Нелинейные системы

© 2023 г. А.А. ГАЛЯЕВ, чл.-корр. РАН (galaev@ipu.ru),
А.И. МЕДВЕДЕВ (medvedev.ai18@physics.msu.ru),
И.А. НАСОНОВ (nasonov.ia18@physics.msu.ru)
(Институт проблем управления им. В.А. Трапезникова РАН, Москва)

НЕЙРОСЕТЕВОЙ АЛГОРИТМ ПЕРЕХВАТА МАШИНОЙ ДУБИНСА ЦЕЛЕЙ, ДВИЖУЩИХСЯ ПО ИЗВЕСТНЫМ ТРАЕКТОРИЯМ[1]

Задача перехвата движущейся по прямолинейной или круговой траектории цели машиной Дубинса сформулирована как задача оптимального управления по критерию быстродействия с произвольным направлением скорости машины при перехвате. Для решения данной задачи и синтеза траекторий перехвата использовались нейросетевые методы обучения без учителя на основе алгоритма Deep Deterministic Policy Gradient. Проведен анализ полученных законов управления и траекторий перехвата по сравнению с аналитическими решениями задачи перехвата, проведено моделирование для параметров движения цели, которые нейросеть не видела при обучении. Проведены модельные эксперименты по проверке устойчивости решения. Показана эффективность применения нейросетевых методов синтеза траекторий перехвата для заданных классов движений цели.

Ключевые слова: задача перехвата, машина Дубинса, алгоритм DDPG, нейросетевой синтез траекторий.

DOI: 10.31857/S0005231023030017, **EDN:** ZYOFFZ

1. Введение

Задачи перехвата подвижных целей, движущихся по известным траекториям, вызывают интерес исследователей с середины 50-х годов прошлого века [1]. Одной из базовых моделей для описания динамики перехватывающего объекта является модель машины Дубинса.

Первые работы по нахождению линии с ограниченной кривизной и минимальной длины, соединяющей две заданные точки, принадлежат А.А. Маркову. Первая его задача в [2] была посвящена поиску кривой, соединяющей две точки на плоскости с минимальной длиной и ограниченной кривизной

[1] Работа выполнена при финансовой поддержке гранта Молодежной научной школы ИПУ РАН «Методы оптимизации и планирования движения управляемых объектов». Работа Галяева А.А. и Насонова И.А. была частично поддержана Российским научным фондом (проект № 23-19-00134).

с фиксированным направлением выхода из первой точки. Такая задача нашла применение в решении проблем прокладки железных дорог. В 1957 г. Л. Дубинс опубликовал похожую работу [3] о нахождении линии кратчайшей длины с ограниченным радиусом кривизны, соединяющей две точки на плоскости с заданным направлением выхода из первой точки и заданным направлением входа во вторую. Результаты оказались полезными при исследовании объектов с ограниченным радиусом разворота и постоянной по величине скоростью передвижения.

В [4] рассмотрена неигровая задача наискорейшего перехвата подвижной цели машиной Дубинса. Предполагалось, что цель движется по произвольной и заранее известной непрерывной траектории. Для нахождения решения были найдены алгебраический критерий оптимальности перехвата по геодезической линии и оптимальное значение критерия времени перехвата.

В ранних исследованиях [5] установлены достаточные условия того, что оптимальной траекторией являются кривые «дуга–прямая». Эти условия накладывают ограничения на отношение минимального радиуса кривизны траектории машины и расстояния между целью и машиной в начальный момент времени. В [6] синтезировано управление для перехвата цели по геодезической линии, проведенной из начала движения машины в точку перехвата, причем полагается, что цель движется по прямой с постоянной скоростью.

Практические применения задач перехвата машиной Дубинса довольно обширны: построение оптимальных траекторий беспилотных летательных аппаратов, выполняющих наблюдение за несколькими наземными целями [7], разработка алгоритмов, решающих задачу коммивояжера [8], построение траекторий обхода при движении с препятствиями [9]. Также модель машины Дубинса используется в дифференциальной игре преследование-уклонение. Такая игра предполагает наличие двух агентов: преследователь должен поймать цель, а убегающий должен уклониться от преследователя. Аналитическое решение задачи нахождения оптимального времени перехвата и синтеза оптимальной траектории для такой игры было получено в [4]. Проблема синтеза траекторий перехвата для объектов, движущихся по круговой траектории, рассматривалась в [10].

Решение задач перехвата машиной Дубинса также можно получить с помощью вычислительных машин. В последнее время для подобных задач активно применяются нейросетевые методы обучения с подкреплением, которые представляют технологию машинного обучения без моделей и применяются в случаях, когда данных для тренировки нейронной сети мало или их нет вообще. В отличие от обучения с учителем [11], которому необходимо наличие набора размеченных данных, обучение с подкреплением основано на взаимодействии агента со средой [12]. Такой метод наиболее эффективен для поиска решения задачи преследования-уклонения.

Метод Actor-Critic используется во многих соответствующих исследованиях. Например, в [13] Actor-Critic использовался с Convolutional Neural Network

(CNN) и Long Short-Term Memory (LSTM) в качестве кодировщика состояния для гоночных игр. В [14] использовали нечеткий детерминированный алгоритм градиента политики для получения конкретного физического смысла при обучении политике в игре преследования-уклонения. В [15] впервые был представлен метод Deep Deterministic Policy Gradient (DDPG) для взаимодействия с непрерывным пространством действий. Именно этот алгоритм будет использоваться в данной работе для нейросетевого синтеза траектории перехвата машиной Дубинса цели, движущейся с постоянной скоростью по прямолинейной и круговой траекториям. Благодаря DDPG удалось впервые получить субоптимальную траекторию, основанную на нейросетевом решении.

Актуальность работы обусловлена как востребованностью на практике алгоритмов перехвата одной и множества движущихся целей, так и возможностью получения некоторых новых теоретических результатов, связанных с синтезом траекторий перехвата. Отдельный интерес представляет так называемая задача коммивояжера с подвижными целями — Moving Target Traveling Salesman Problem (MTTSP) [16]. В данном случае точки, которые требуется обойти, движутся с заданной скоростью. Примером такого сценария является перехват нескольких уклоняющихся (или атакующих) целей, что весьма актуально для приложений двойного назначения. Очевидно, что поиск наилучшего маршрута для перехвата нескольких подвижных целей является особенно сложной задачей ввиду постоянного изменения положения целей, что значительно увеличивает вычислительные затраты на поиск оптимальных решений. Известно, что для решения MTTSP в литературе был предложен эвристический подход.

Авторы предлагают синтез траектории перехвата, основанный на нейросетевом решении, поскольку аналитические результаты и оптимальные траектории для групп целей практически отсутствуют или неизвестны. Этот метод авторы планируют масштабировать на подобные задачи.

Структура работы включает в себя 6 разделов. В разделе 2 предлагается математическая постановка задачи, адаптированная к дальнейшему применению. Раздел 3 посвящен описанию алгоритма DDPG, также готового к применению в данной постановке. В разделе 4 описывается структура нейронной сети, а раздел 5 содержит результаты моделирования. В заключении представлено направление дальнейших исследований.

2. Постановка задачи нейросетевого перехвата

На плоскости рассматривается задача наибыстрейшего δ-перехвата машиной Дубинса (преследователь) подвижного объекта (цель), движущегося по двум заданным траекториям с постоянной скоростью. Как и в [4], динамика для преследователя была выбрана в виде

$$\begin{cases} \dot{x_P} = \cos\varphi, \\ \dot{y_P} = \sin\varphi, \\ \dot{\varphi} = u, \ |u(t)| \leqslant 1. \end{cases} \tag{1}$$

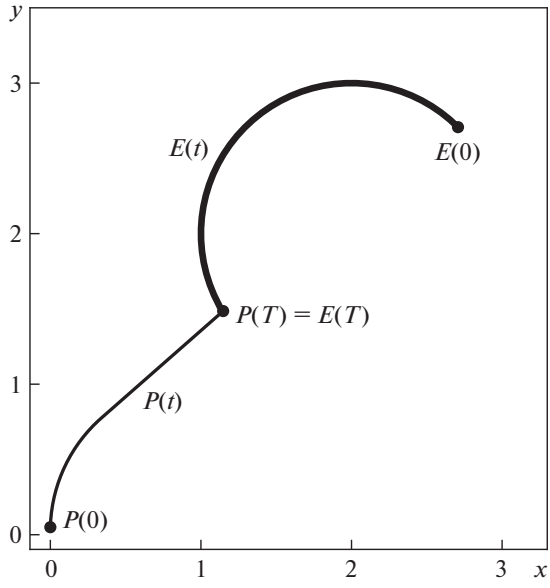

Рис. 1. Взаимное расположение объектов.

Здесь $x_P(t)$ и $y_P(t)$ — координаты машины Дубинса на декартовой плоскости, $\varphi(t)$ — угол между направлением скорости преследователя и осью абсцисс, а $u(t)$ — управление, зависящее от времени, что показано на рис. 1. Координаты и угол машины обозначим через вектор-функцию $P(t) = (x_P(t), y_P(t), \varphi(t))$.

Начальные условия системы (1) фиксированы:

$$x_P(0) = 0, \quad y_P(0) = 0, \quad \varphi(0) = \frac{\pi}{2}. \tag{2}$$

Непрерывная вектор-функция $E(t) = (x_E(t), y_E(t))$ определяет траекторию цели на декартовой плоскости.

Терминальное условие δ-перехвата для нейросетевого решения имеет следующий вид:

$$(x_P(T) - x_E(T))^2 + (y_P(T) - y_E(T))^2 \leqslant \delta^2, \tag{3}$$

где $T \in \mathbb{R}_0^+$ — время движения из начальной точки в точку перехвата, а δ — заданный радиус перехвата — максимально допустимое расстояние между преследователем и целью, при котором перехват можно считать совершенным. Данный параметр вводится для определенности понятия перехвата именно для нейросетевого решения.

Поставим задачу перехвата цели за минимальное время как задачу оптимального управления в классе кусочно-постоянных функций:

$$J[u] \stackrel{def}{=} \int_0^T dt \to \min_u. \tag{4}$$

Приступим к описанию динамики цели. По условию задачи цель движется с постоянной скоростью прямолинейно или по окружности. Тогда параметризованные уравнения координат будут иметь следующий вид:

$$\begin{cases} x_E(t) = R\cos(\omega t + \phi) + x_0, \\ y_E(t) = R\sin(\omega t + \phi) + y_0; \end{cases} \tag{5}$$

$$\begin{cases} x_E(t) = v_x t + x_0, \\ y_E(t) = v_y t + y_0, \end{cases} \tag{6}$$

где x_0 и y_0 являются начальными условиями координат цели и выбираются произвольно.

Для учета взаимного расположения преследователя и цели введем формулу для нахождения угла между осью абсцисс и прямой, соединяющей координатные точки цели и преследователя. Пусть (x_P, y_P) и (x_E, y_E) — координаты преследователя и цели соответственно в некоторый момент времени t. Тогда искомая величина угла находится по формуле

$$\psi = \arctan\left(\frac{y_E - y_P}{x_E - x_P}\right).$$

Также введем формулу для расчета расстояния L между агентами:

$$L = \sqrt{(x_P - x_E)^2 + (y_P - y_E)^2}.$$

Далее, для упрощения исследования задачи совершим переход в новые координаты. Для этого необходимо уметь сравнивать текущее состояние агентов $S = (x_P, y_P, \varphi, x_E, y_E)$ и состояние, предсказанное нейронной сетью $S' = (x'_P, y'_P, \varphi', x'_E, y'_E)$.

Получим значения для функций углов ψ и ψ' от состояний S и S' соответственно, а также рассчитаем расстояние L', когда агенты находятся в состоянии S'. Введем угол между направлением скорости преследователя и линией, соединяющей координатные точки агентов:

$$\Theta = \varphi' - \psi'.$$

Введем скорость поворота как частное разности $\psi' - \psi$ и промежутка времени Δt, за которое произошел переход от состояния S в состояние S':

$$\omega = \frac{\psi' - \psi}{\Delta t}.$$

Совокупность (L', ω, Θ) и есть искомые координаты, в которых будем строить нейросетевое решение. В начальный момент времени, когда результат

нейросети еще не получен, координаты $(L'(0), \omega(0), \Theta(0))$ рассчитываются следующим образом:

$$\begin{cases} L'(0) = L(S), \\ \omega(0) = 0, \\ \Theta(0) = \varphi(0) - \psi(0), \end{cases} \tag{7}$$

где $\psi(0) = \arctan\left(\frac{y_E(0)-y_P(0)}{x_E(0)-x_P(0)}\right)$.

3. Алгоритм Deep Deterministic Policy Gradient

DDPG — это Actor-Critic алгоритм, основанный на градиенте детерминированной политики. Алгоритм DPG (Deterministic Policy Gradient) состоит из параметризованной функции Actor $\mu\left(s \mid \theta^{\mu}\right)$, которая задает управление в текущий момент времени путем детерминированного сопоставления состояний с конкретным действием. Функция Critic $Q(s, a)$ обновляется с помощью уравнения Беллмана так же, как и при Q-обучении. Actor обновляется путем применения цепного правила к ожидаемому вознаграждению от начального распределения J по отношению к параметрам Actor:

$$\begin{aligned} \nabla\theta^{\mu}\mathrm{J} &\approx \mathbb{E}_{s_{\mathrm{t}}\sim\rho^{\beta}}\left[\nabla_a Q\left(s, a \mid \theta^{Q}\right)\big|_{s=s_{\mathrm{t}}, a=\mu(s_{\mathrm{t}}|\theta^{\mu})}\right] \\ &= \mathbb{E}_{s_{\mathrm{t}}\sim\rho^{\beta}}\left[\nabla_{\theta^{\mu}} Q\left(s, a \mid \theta^{Q}\right)\big|_{s=s_{\mathrm{t}}, a=\mu(s_{\mathrm{t}})} \nabla_{\theta_{\mu}}\mu\left(s \mid \theta^{\mu}\right)\Big|_{s=s_{\mathrm{t}}}\right]. \end{aligned} \tag{8}$$

DDPG сочетает в себе достоинства своих предшественников, что делает его более устойчивым и эффективным в обучении. Так как разные траектории могут очень сильно отличаться друг от друга, DDPG использует идею DQN [17], называемую буфером воспроизведения. Буфер воспроизведения — это буфер конечного размера, в который сохраняются данные о среде в каждый момент времени. Он необходим для достижения равномерного распределения выборки переходов и дискретного контроля обучения нейронных сетей. Actor и Critic обновляются путем равномерной выборки mini-batch из буфера воспроизведения. Еще одним дополнением к DDPG стала концепция обновлений программных целей вместо прямого копирования весов в целевую сеть. Обновляемая сеть $Q\left(s, a \mid \theta^{Q}\right)$ также используется для расчета целевого значения, поэтому обновление Q подвержено расхождению. Это возможно, если сделать копию сетей Actor и Critic, $Q'\left(s, a \mid \theta^{Q'}\right)$ и $\mu'\left(s, a \mid \theta^{\mu'}\right)$. Веса этих сетей следующие: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ с $\tau \ll 1$. Проблема исследования решается путем добавления шума, полученного от шумового процесса N, в управление актора. В данном исследовании выбран процесс Орнштейна–Уленбека [18].

Общая структура DDPG показана на рис. 2. Поскольку в задаче требуется, чтобы управления были заключены в числовом интервале, то необходимо ввести ограничения. Для этого в программе была использована функция $clip()$, которая ограничивает область значений действий в диапазоне $[-1; 1]$.

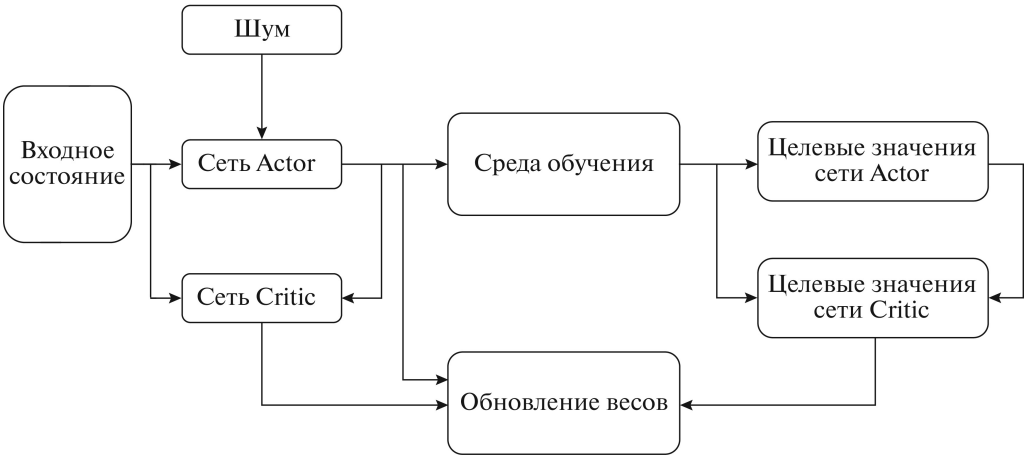

Рис. 2. Общая структура алгоритма Deep Deterministic Policy Gradient.

Алгоритм 1.

Входные данные: коэффициент дисконтирования γ, количество эпизодов M, количество шагов обучения T в каждом эпизоде, batch size N, коэффициенты обучения нейронных сетей Actor и Critic r_a и r_c соответственно.

1. Произвольная инициализация сетей Actor $\mu(s|\theta^\mu)$ и Critic $Q(s,a|\theta^Q)$

2. Инициализация целевых сетей Q' и μ' весовыми параметрами $\theta^Q = \theta^{Q'}$ и $\theta^\mu = \theta^{\mu'}$

3. Инициализация буфера R

4. for episode $= 1$ to M do

5. Инициализация случайного действия $a_t = \mu(s_t|\theta^\mu) + \eta_t$ в соответствии с текущим управлением и исследовательским шумом

6. Получение начального состояния среды s_1

7. for $t = 1$ to T do

8. Выполнение действия a_t, приобретение награды r_t и получение нового состояния среды s_{t+1}

9. Сохранение перехода (s_t, a_t, r_t, s_{t+1}) в буфере R

10. Случайная выборка N переходов (s_i, a_i, r_i, s_{i+1}) из R

11. Получение $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

12. Обновление весов сети Critic путем минимизирования функции потерь

$$\widehat{L} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

13. Обновление политики Actor с помощью градиента эффективной политики:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

14. Обновление целевых сетей

$$\theta^{Q'} = \tau\theta^{Q} + (1-\tau)\theta^{Q}, \quad \theta^{\mu'} = \tau\theta^{\mu} + (1-\tau)\theta^{\mu}$$

15. endfor

16. endfor

Выходные данные: Управление $u = \mu(s|\theta^{\mu})$

В табл. 1 показаны различия между сетями Actor, Critic и их целевыми сетями. В ней приведены входные и выходные значения, а также формулы, по которым производится расчет этих величин.

Подробное описание работы метода DDPG приведено в алгоритме 1.

Таблица 1. Различия между сетями Actor, Critic и их целевыми сетями

Сеть	Формула	Входные данные	Выходные данные
Critic целевая	$Q'\left(s_{t+1}, \mu'\left(s_{t+1}\|\theta^{\mu'}\right)\|\theta^{Q'}\right)$	следующее состояние среды; выходные данные целевой сети Actor	значение Q', которое используется для вычисления y_i
Critic	$Q\left(s_t, a\|\theta^{Q}\right)$	текущее состояние среды; текущее действие	значение Q, которое необходимо для расчета ошибки и обновления сети Actor
Actor целевая	$\mu'\left(s_{t+1}\|\theta^{\mu'}\right)$	следующее состояние среды	действие μ', используемое как входное значение целевой сети Critic
Actor	$\mu\left(s_t\|\theta^{u}\right)$	текущее состояние среды	действие μ, которое используется для обновления сети Actor

4. Нейронная сеть

4.1. Структура сети

Для реализации алгоритма Deep Deterministic Policy Gradient были написаны две нейросети для каждого метода: Critic и Actor. Их архитектуры изображены на рис. 3 и 4.

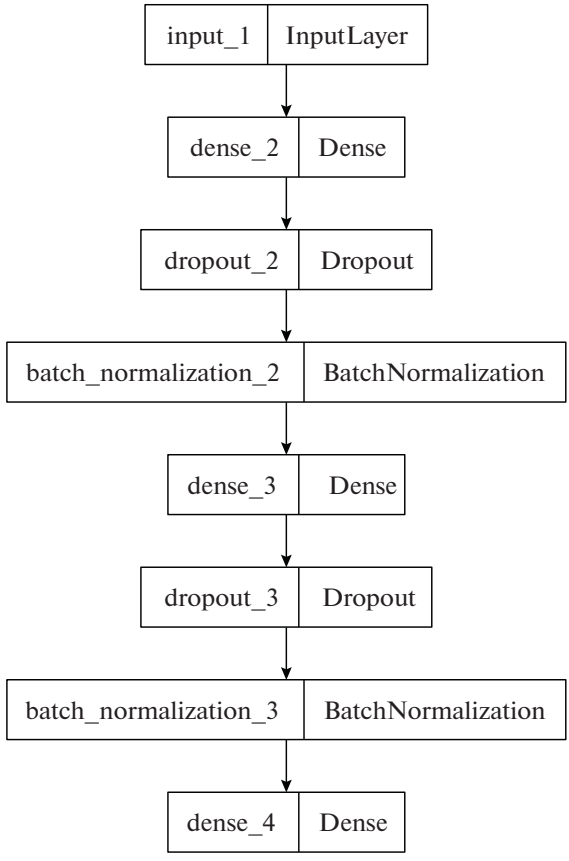

Рис. 3. Архитектура нейросети Actor.

Сеть Actor имеет четыре полносвязных скрытых слоя с 256 нейронами, с функцией активации $SELU$. Так как возможные действия находятся в интервале $[-1, 1]$, то функцию активации для выходного слоя удобно взять как $tanh$. Сеть Critic имеет пять полносвязных скрытых слоев с 16, 32, 32 и два слоя с 512 нейронами, с функцией активации $SELU$.

Сети Critic и Actor составлены из полносвязных слоев $Dense$, для выходных значений которых применяется операция нормализации и метод $Dropout$ [19], который эффективен в борьбе с проблемой переобучения нейросетей. Для вычисления выхода сети Actor из последнего слоя выбрана функция активации гиперболический тангенс.

Сеть Critic имеет сложную структуру, поскольку она принимает два входных значения: состояние среды и действия преследователя. Далее происходит соединение слоев с помощью метода $Concatenate$ и значения проходят через полносвязные слои сети до выхода, который является слоем единичной размерности.

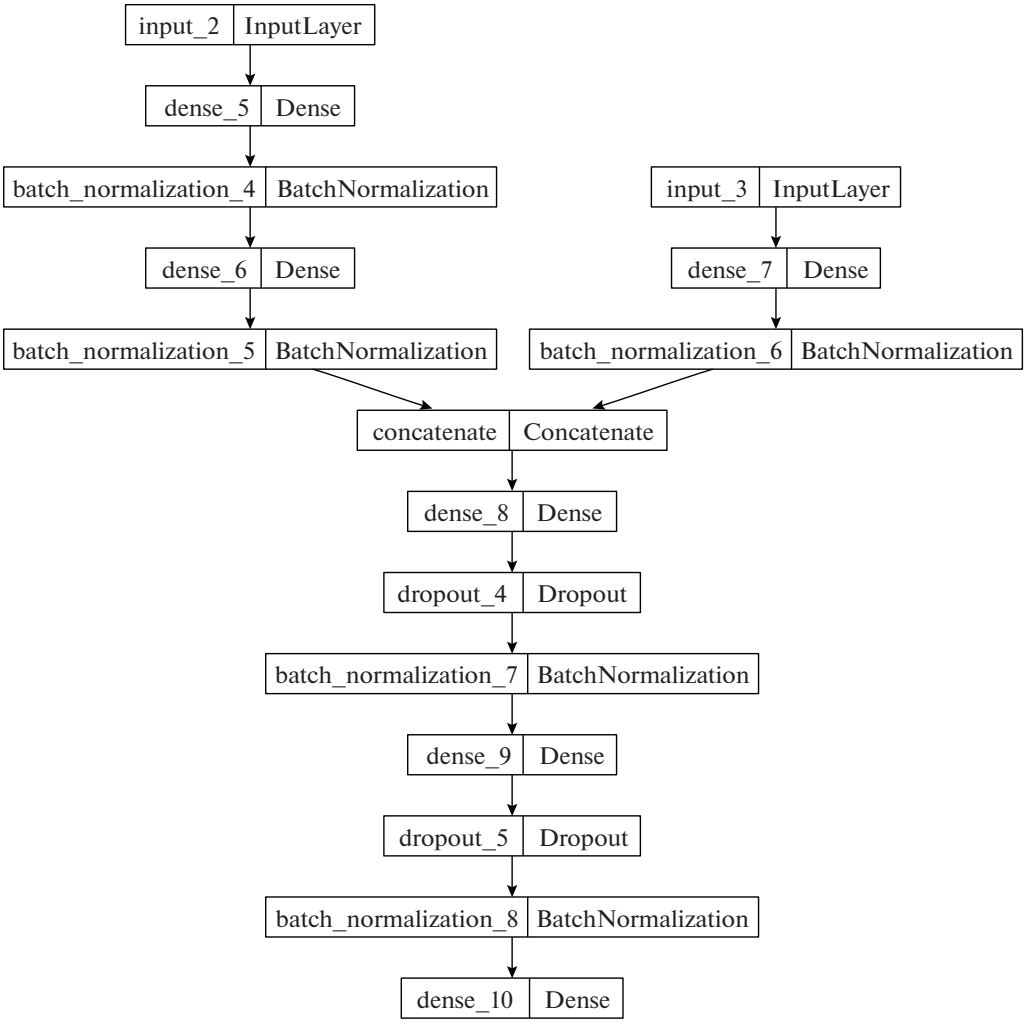

Рис. 4. Архитектура нейросети Critic.

4.2. Гиперпараметры

В качестве функции активации в скрытых слоях нейросетей Critic и Actor была выбрана функция $SELU$ [20], которая задается следующим уравнением:

$$SELU(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha e^x - \alpha, & x \leqslant 0, \end{cases}$$

где $\lambda \approx 1{,}0507$, а $\alpha \approx 1{,}6732$.

График функции $SELU$ приведен на рис. 5.

Функция $SELU$ обадает свойством самонормализации входных данных при использовании метода инициализации $LeCun$, который инициализирует параметры сети как нормальное распределение. Поэтому выходные значения этой функции имеют нулевое среднее и единичное стандартное отклонение.

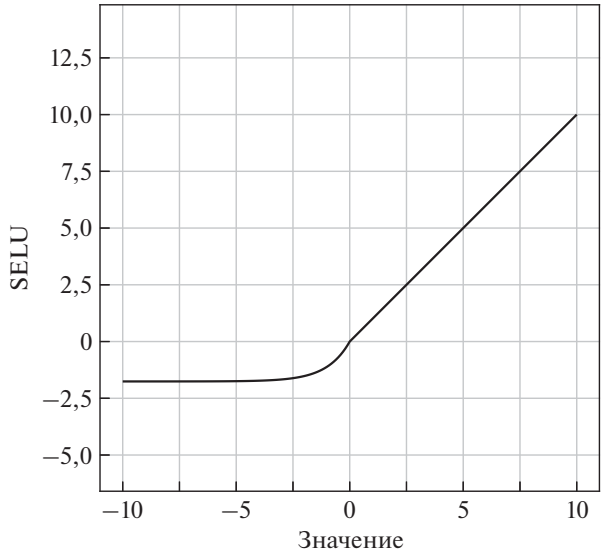

Рис. 5. График функции активации $SELU$.

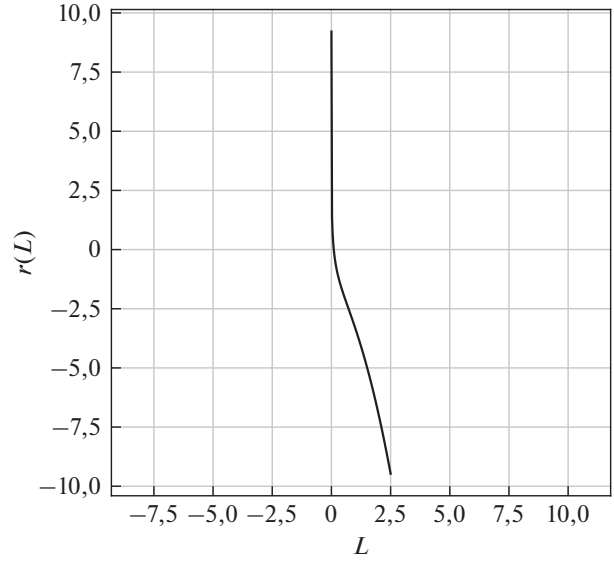

Рис. 6. График зависимости вознаграждения от расстояния между агентами.

В виде функции вознаграждения преследователя было выбрано следующее выражение, зависящее только от расстояния L между агентами:

$$r(L) = -\lg(10L) - L^2. \tag{9}$$

График этой функции приведен на рис. 6. На нем можно видеть, что значение r быстро растет при уменьшении L, а когда расстояние принимает нулевое значение, то агент получает максимальную награду.

Таблица 2. Значения параметров нейронной сети

Параметр	Значение	Описание
γ	0,98	Коэффициент дисконтирования, используемый в уравнении Беллмана
τ	0,01	Коэффициент мягкого обновления целевых сетей
Размер mini-batch	64	Количество выборок для обновления весов
Объем буфера R	10 000	Количество данных, из которых выбираются примеры для обновления
Размер эпизода	1000	Количество эпизодов, используемых для обучения
Размер шага	400	Количество шагов обучения в каждом эпизоде
Интервал времени	0,1	Время каждого шага обучения
Коэффициент обучения сети Actor	5e-5	Коэффициент обучения, используемый для обновления сети Actor
Коэффициент обучения сети Critic	1e-4	Коэффициент обучения, используемый для обновления сети Critic

Значения гиперпараметров нейронных сетей приведены в табл. 2. Параметры γ, τ, размер эпизода и интервал времени были подобраны в результате анализа в соответствии с [14]. Однако значения размера mini-batch, объема буфера R, размера шага и коэффициентов обучения сетей Actor-Critic были подобраны эмпирическим путем — сеть синтезировала траектории перехвата движения цели, а затем проводился их анализ на предмет соответствия физической задаче. Например, если график среднего вознаграждения не увеличивался в течение 100–200 эпизодов обучения, а значения функций ошибок нейросетей Actor-Critic не убывали за тот же период, то значения коэффициентов обучения сетей уменьшались, а размер mini-batch увеличивался.

5. Результаты моделирования

5.1. Процесс обучения нейросети

Моделирование было произведено при помощи языка Python и фреймворка TensorFlow. Начальные параметры движения цели и преследователя во время обучения нейросети приведены в табл. 3.

Начальные координаты движения цели выбираются случайным образом с помощью функции numpy.random.uniform() в диапазоне $(-3; 3)$, чтобы сеть тренировалась на разных примерах и эффективно работала после процесса обучения. Скорости цели всегда имели константное значение на всем процессе обучения $v_x = 0{,}5$ и $v_y = 0{,}5$.

Тренировка нейросети проводилась на процессе с характеристиками, указанными в табл. 4. В силу сложности нейросетевой модели процесс обучения длился около четырех часов.

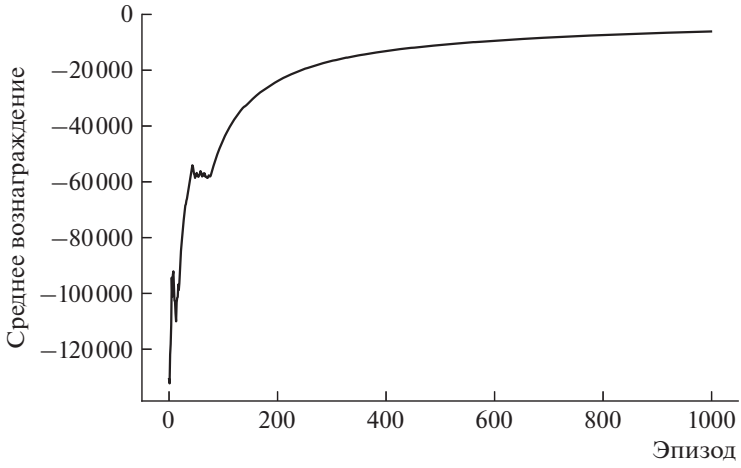

Рис. 7. Зависимость среднего вознаграждения от номера эпизода.

На рис. 7 показан график среднего вознаграждения за весь период обучения. Во время тренировки модели наблюдается резкое возрастание значения награды агента в первые 100–150 эпизодов. Данному процессу соответствует заполнение буфера воспроизведения R. Далее примеры для тренировки

Таблица 3. Начальные параметры движения цели и преследователя во время обучения сети

Параметр	Значение
Начальная координата движения цели $x_E(0)$	Произвольное значение в промежутке $(-3; 3)$
Начальная координата движения цели $y_E(0)$	Произвольное значение в промежутке $(-3; 3)$
Начальные координаты движения преследователя $(x_P(0); y_P(0))$	$(0; 0)$
Начальная ориентация преследователя $\varphi(0)$	$\frac{\pi}{2}$
Постоянная скорость преследователя v	1
Радиус перехвата δ	0,2

Таблица 4. Характеристики оборудования, где проходило обучение сети

Параметр	Значение
Процессор	Intel(R) Core(TM) i7-8565U
Литография	14 нм
Количество ядер	4
Количество потоков	8
Базовая тактовая частота процессора	1,80 ГГц
Кэш-память	8 Мб
Оперативная память компьютера	16 Гб

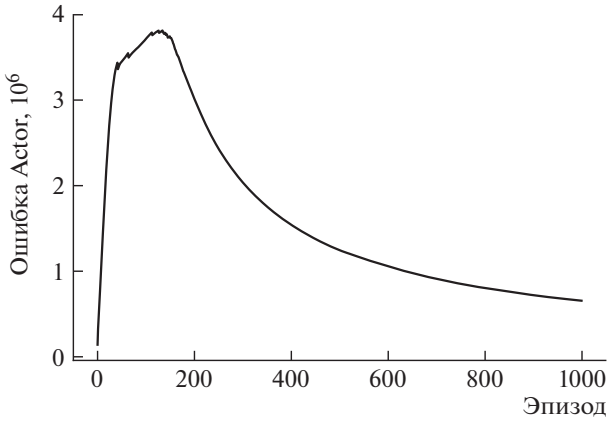

Рис. 8. Зависимость значения функции ошибки от эпизода сети Actor.

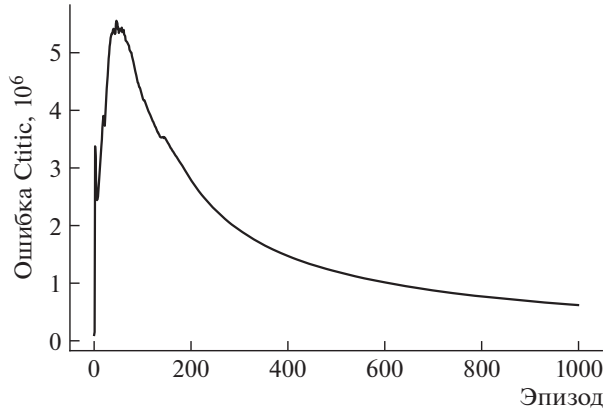

Рис. 9. Зависимость значения функции ошибки от эпизода сети Critic.

случайным образом берутся из R, происходит процесс обучения сети и полученный кортеж состояний заменяет старый образец данных в R. На этом этапе происходит медленный рост среднего вознаграждения, см. рис. 7.

Также были получены графики зависимостей функции ошибки нейронных сетей Actor и Critic. Они приведены на рис. 8 и 9 соответственно.

На графиках видно плавное уменьшение значения функции потерь с увеличением эпизодов обучения, что свидетельствует о правильном выборе коэффициентов обучения.

5.2. Результат обучения

На рис. 10 изображены траектории, полученные с помощью нейросети и аналитического решения. Начальные параметры цели и преследователя в этом случае имели значения, указанные в табл. 5.

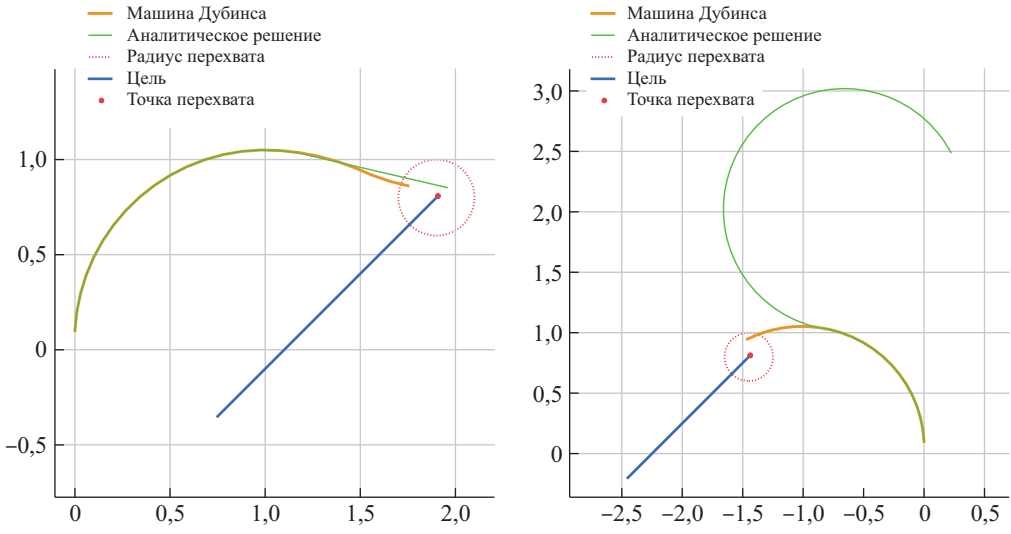

Рис. 10. Сравнение траекторий перехвата прямолинейно движущейся цели при разных начальных параметрах.

По траекториям, изображенным на рис. 10, можно видеть, что сеть смогла построить более эффективную траекторию. В этом случае оптимальное время перехвата, полученное с помощью аналитического решения, равняется $T_{opt} \approx 5{,}42$ с. А время, за которое сеть смогла перехватить цель, $T_{nn} \approx 2{,}1$ с. Такой результат объясняется наличием радиуса перехвата $\delta = 0{,}2$.

На рис. 11 справа можно увидеть сравнение графиков нейросетевого управления с аналитическим. Как видно, управления значительно отличаются на конечном участке траектории вследствие того, что нейросеть подстраивает терминальные условия перехвата. Оптимальный синтез в задаче с нефиксированным углом перехвата состоит из участков “Дуга–прямая” либо “Дуга–дуга” [4], а в задаче с фиксированным углом перехвата — в общем случае из участка “Дуга–прямая–дуга” [21]. Именно последний вариант и синтезирует нейросеть. При этом, как видно из рис. 10, есть участок траектории, где нейросеть выбирает не оптимальное, но близкое к оптимальному значение радиуса разворота. Второй причиной отличия является то, что нейросеть

Таблица 5. Начальные параметры движения цели и преследователя

Параметр	Значение	Значение
Начальные координаты движения цели	$(0{,}8; -0{,}4)$	$(-2{,}5; -0{,}25)$
Постоянная скорость цели v_x	0,5	0,5
Постоянная скорость цели v_y	0,5	0,5
Начальные координаты движения преследователя	$(0; 0)$	$(0; 0)$
Начальная ориентация преследователя $\varphi(0)$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
Постоянная скорость преследователя v	1	1
Радиус перехвата δ	0,2	0,2

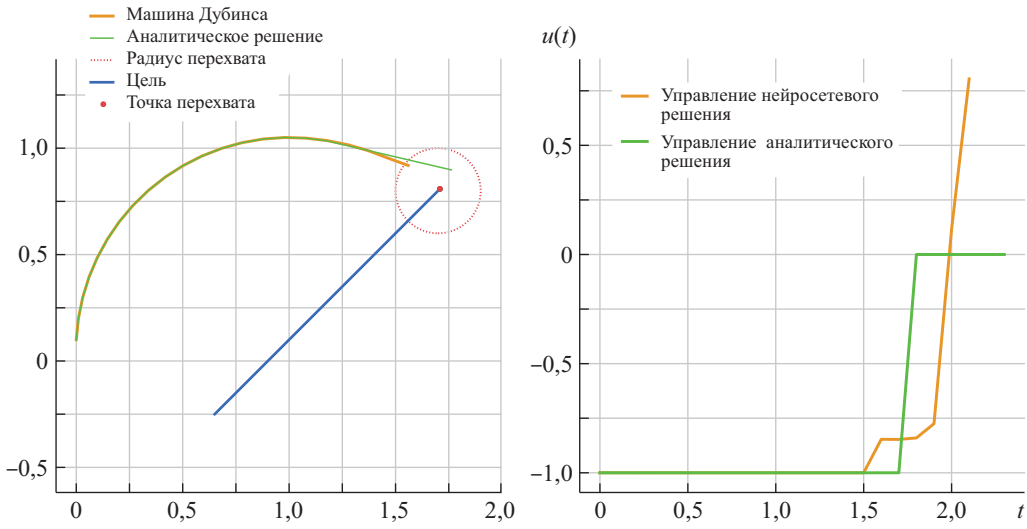

Рис. 11. Сравнение функций управления от времени.

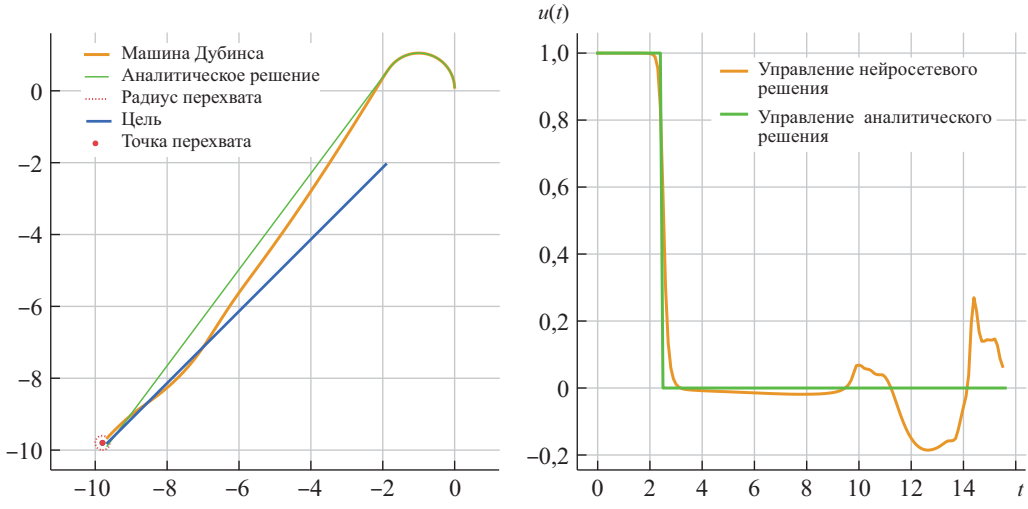

Рис. 12. Сравнение функций управления от времени.

оптимизирует локальную функцию вознаграждения, отличную от функционала быстродействия, который использовался при постановке задачи.

На рис. 12 изображены траектории перехвата цели и зависимость функции управления от времени. На правом графике можно наблюдать, что функция нейросетевого управления имеет значения, близкие к оптимальному, на участке, где аналитическое решение дает нулевое управление. К тому же отклонения нейросетевого управления не превышают значение радиуса перехвата δ. Времена перехвата в этом случае практически идентичны: $T_{opt} \approx \approx T_{nn} \approx 21$ с.

Проанализируем, насколько нейросетевое решение зависит от входных параметров, так как в теории нейросеть должна хорошо обобщать полученное решение на состояния и параметры, которые еще “не видела” при обучении.

Для перехвата цели, движущейся по окружности, обучим нейросеть только на цели с единичным радиусом и единичной угловой скоростью и проверим, может ли она успешно ловить цель с другими параметрами. Как видно на рис. 13, сеть успешно справляется с задачей, на левом рисунке угловая

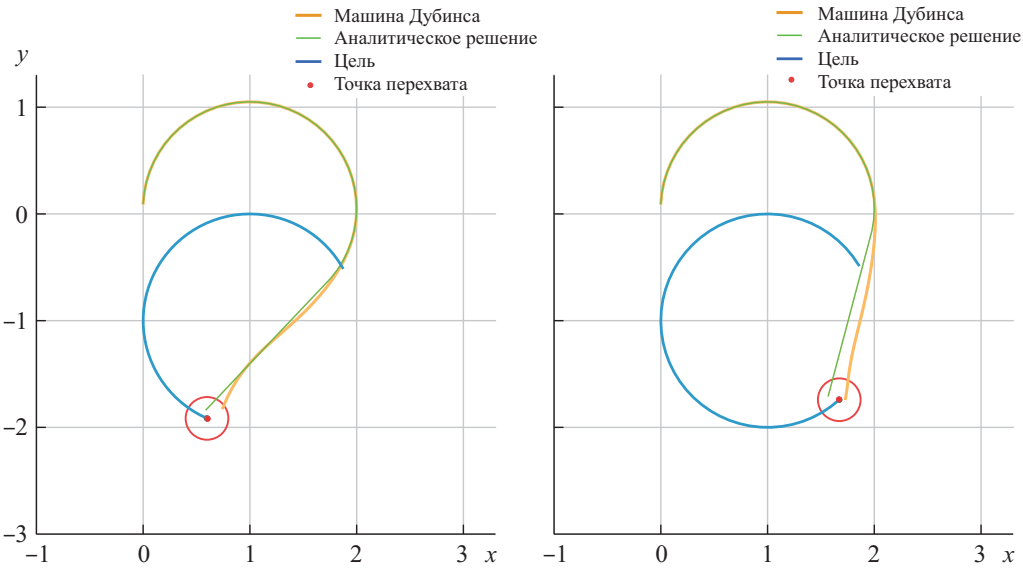

Рис. 13. Сравнение траекторий кругового перехвата при разных начальных параметрах.

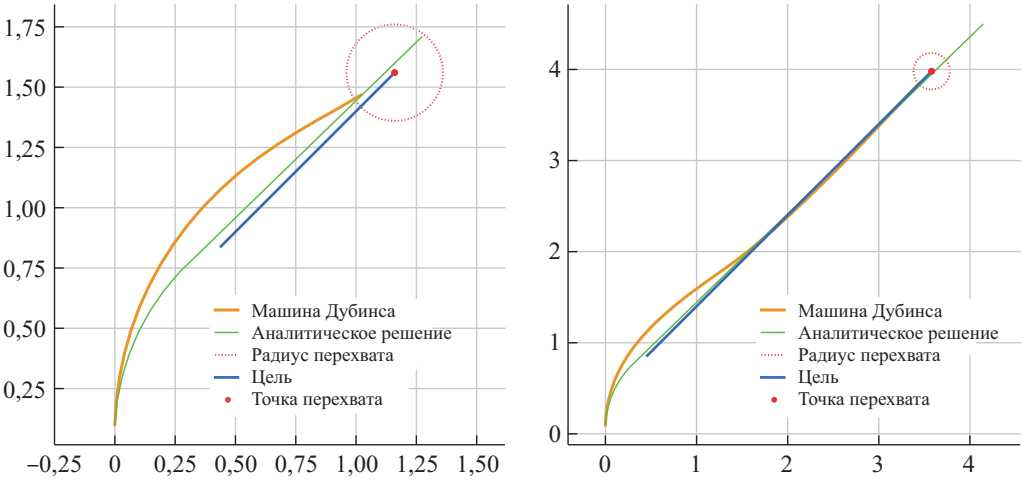

Рис. 14. Сравнение траекторий перехвата цели при разных начальных параметрах.

скорость у цели равна 0,7 от угловой скорости, используемой при обучении, на правом рисунке изображен перехват обычной цели. Были проведены эксперименты для значений угловой скорости от 0,7 до 1,3, в которых нейросеть успешно перехватывала цель.

В случае перехвата прямолинейно движущейся цели нейросеть обучалась при значениях скорости цели $v_x = v_y = 0{,}5$. На рис. 14 изображены результаты тестирования сети со скоростями, различающимися на 20% — на левом рисунке цель имеет скорость $v_x = v_y = 0{,}4$, а на правом $v_x = v_y = 0{,}6$.

Из полученных результатов следует, что сеть хорошо обобщает решение. Это может быть полезно для прикладных задач, так как в них параметры зачастую известны с некоторой погрешностью.

6. Заключение

В работе были предложены два основанных на DDPG нейросетевых алгоритма синтеза траекторий перехвата машиной Дубинса целей, движущихся по прямолинейной и круговым траекториям. Особенностями предложенных алгоритмов является их способность работать с пространством непрерывных действий, гарантированность обучения и работы с различными относительными начальными положениями целей и машины Дубинса. Показано, что сеть успешно обобщает решение и в некоторых ситуациях предлагает лучшее по быстродействию решение задачи перехвата.

Несомненные преимущества предложенных алгоритмов могут быть использованы, а сами алгоритмы доработаны для получения барьерной поверхности в дифференциальной игре двух автомобилей.

СПИСОК ЛИТЕРАТУРЫ

1. *Isaacs R.* Differential games. New York: John Wiley and sons, 1965.
   Айзекс Р. Дифференциальные игры. М.: Мир, 1967.
2. *Markov A.A.* A few examples of solving special problems on the largest and smallest values. / The communications of the Kharkov mathematical society. 1889. Ser. 2. V. 1. P. 250–276.
3. *Dubins L.E.* On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents // Amer. J. Math. 1957. No. 79. P. 497–516.
4. *Галяев А.А., Бузиков М.Э.* Перехват подвижной цели машиной Дубинса за кратчайшее время // АиТ. 2021. № 5. С. 3–19.
   Galyaev A.A., Buzikov M.E. Time-Optimal Interception of a Moving Target by a Dubins Car // Autom Remote Control. 2021. V. 82. P. 745–758.
5. *Glizer V.Y., Shinar J.* On the structure of a class of time-optimal trajectories // Optim. Control Appl. Method. 1993. V. 14. No. 4. P. 271–279.
6. *Бердышев Ю.И.* О задачах последовательного обхода одним нелинейным объектом двух точек // Тр. ИММ УрО РАН. 2005. Т. 11. № 1. С. 43–52.

7. *Xing Z.* Algorithm for Path Planning of Curvature-constrained UAVs Performing Surveillance of Multiple Ground Targets // Chin. J. Aeronaut. 2014. V. 27. No. 3. P. 622–633.
8. *Ny J.L., Feron E., Frazzoli E.* On the Dubins Traveling Salesman Problem // IEEE Transactions on Automatic Control. 2014. V. 57. P. 265–270.
9. *Yang D., Li D., Sun H.* 2D Dubins Path in Environments with Obstacle // Math. Problem. Engineer. 2013. V. 2013. P. 1–6.
10. *Manyam S.G. et al.* Optimal dubins paths to intercept a moving target on a circle // Proceedings of the American Control Conference. 2019. V. 2019-July. P. 828–834.
11. *Caruana R., Niculescu-Mizil A.* An empirical comparison of supervised learning algorithms // ICML Proceedings of the 23rd international conference on Machine learning. June 2006. P. 161–168.
12. *Arulkumaran K., Deisenroth M.P., Brundage M., Bharath A.A.* Deep Reinforcement Learning: A Brief Survey // IEEE Signal Processing Magazine. 2017. V. 34. No. 6. P. 26–38.
13. *Perot E., Jaritz M., Toromanoff M., de Charette R.* End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning // IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2017. P. 474–475.
14. *Al-Talabi A.A., Schwartz H.M.* Kalman fuzzy actor-critic learning automaton algorithm for the pursuit-evasion differential game // IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). 2016. P. 1015–1022.
15. *Hartmann G., Shiller Z., Azaria A.* Deep Reinforcement Learning for Time Optimal Velocity Control using Prior Knowledge // IEEE 31st International Conference on Tools with Artificial Intelligence. 2019. P. 186–193.
16. *Helvig C.S., Gabriel Robins, Alex Zelikovsky* The moving-target traveling salesman problem // J. Algorithm. 2003. V. 49. No. 1. 2003. P. 153–174.
17. *Mnih V., Kavukcuoglu K., Silver D. et al.* Human-level control through deep reinforcement learning // Nature. 2015. V. 518. P. 529–533.
18. *Uhlenbeck G.E., Ornstein L.S.* On the theory of the brownian motion // Physic. Rev. 1930. V. 36. P. 823–841.
19. *Hinton G.E., Srivastava N., Krizhevsky A. et al.* Improving neural networks by preventing co-adaptation of feature detectors. arXiv. 2012.
20. *Klambauer G., Unterthiner T., Mayr A. et al.* Self-normalizing neural networks // Advances in Neural Information Processing Systems. 2017. P. 972–981.
21. *Buzikov M.E., Galyaev A.A.* Minimum-time lateral interception of a moving target by a Dubins car // Automatica. 2022. V. 135. 109968.

Статья представлена к публикации членом редколлегии О.П. Кузнецовым.

Поступила в редакцию 28.07.2022

После доработки 17.11.2022

Принята к публикации 30.11.2022