НЕЙРОННАЯ СЕТЬ, ИНТЕГРИРОВАННАЯ В СИСТЕМУ АДАПТИВНОГО УПРАВЛЕНИЯ АВТОНОМНЫМ НЕОБИТАЕМЫМ ПОДВОДНЫМ АППАРАТОМ

Романова В. Р.[1]
(*Крымский федеральный университет имени В. И. Вернадского, Симферополь*)
Зуев С. В.[2]
(*Белгородский государственный технологический университет им. В.Г. Шухова, Белгород*)

Данная статья посвящена исследованию и разработке модели интеллектуальной системы адаптивного автоматического управления с заданной целью, основанной на использовании искусственных нейронных сетей. Основной целью работы является подготовка данных для искусственной нейронной сети, которая интегрирует информацию от сонара, лидара и бортовых датчиков, обеспечивая оптимальную подготовку данных для принятия решений в реальном времени. Результатом исследования стала модель интеллектуальной системы, способной адаптироваться к изменяющимся условиям окружающей среды и принимать решения на основе данных из различных источников. Это значительно повышает эффективность управления подводным аппаратом. Значимость данной работы заключается в создании более эффективных и надежных методов управления техническими объектами. Разработанная модель использует телеметрию для подготовки данных для работы искусственной нейронной сети прямого распространения, что может быть полезно в автомобильной и морской промышленности, авиации и других областях, где необходимо точное и быстрое управление в условиях переменной среды. Исследование этой интеллектуальной системы адаптивного автоматического управления представляет собой важный шаг в развитии методов управления техническими объектами и может найти применение в разнообразных отраслях, где актуальными являются задачи точного и быстрого управления.

Ключевые слова: искусственная нейронная сеть, АНПА, модель системы управления, метод обратного распространения ошибки.

1. Введение

Задача адаптивного управления является одной из наиболее важных в области автоматического управления. Она заключает-

[1] *Валерия Романовна Романова, студент* (*lero4ka2004ro@gmail.com*).
[2] *Сергей Валентинович Зуев, к.ф.-м.н., доцент* (*sergey.zuev@bk.ru*).

ся в разработке системы управления, способной быстро и эффективно адаптироваться к изменениям внешних условий.

Адаптивное управление используется во многих областях, таких как промышленность, авиация, робототехника, медицина и другие. В этих областях происходит постоянное изменение внешних условий, например изменение скорости, нагрузки, температуры и т.д. В следующих условиях требуется система управления, способная адаптироваться к изменениям с минимальными потерями производительности и качества работы.

В данной статье будет рассмотрена искусственная нейронная сеть как метод адаптивного управления автономным необитаемым подводным аппаратом (АНПА) [1,2].

2. Постановка задачи

Известно начальное положение аппарата и его желаемая траектория. На основе этих данных нужно обеспечить движение АНПА по этой траектории с минимальными отклонениями, используя движители и показания датчиков скоростей потоков, лидаров и сонаров.

Будем считать, что показания лидаров и сонаров обрабатываются специальной подсистемой, которая после первоначального определения положения АНПА относительно определенных реперных точек (как правило, самых ярких) затем отслеживает устройство и периодически выдает его текущее положение в той же системе координат, в которых задано начальное положение и желаемая траектория АНПА.

Пусть имеются следующие сигналы:

1. $u^j(t)$ – скорости потока воды, измеренные разнонаправленными датчиками в разных местах аппарата; всего пусть имеется J таких сигналов, поступающих в систему с периодичностью θ_u, и имеющих квант η (все датчики одинаковы);

2. $p^k(t)$ – позиционирование, поступающее из подсистемы позиционирования; всего пусть имеется $K = 3$ таких сигналов, поступающих в систему с периодичностью θ_p, и имеющих квант ξ (погрешность подсистемы положим постоянной и одинаковой для всех осей);

3. $f^k(t)$ – проекции скорости течения, не измеряемые и непредсказуемые; очевидно, имеет три компоненты ($K = 3$), не имеет периодичности и установленного кванта;

4. $m^s(t)$ – тяга движителей - сигнал, который является и управляющим, и исходящим одновременно, поскольку никаких других команд, кроме тех, что исходят из управляющей системы, АНПА не имеет (нет связи с оператором); в момент времени t имеются S значений $m^s(t)$, а работа управляющей системы состоит в том чтобы получить столько же значений $m^s(t + \theta_m)$ и применить их в момент $t + \theta_m$; квантование этих сигналов задается возможностями движителей и пусть квант равен μ;

5. $x^k(t)$ – желаемая траектория; задается последовательностью значений с дискретностью θ_x.

В этом случае решением задачи будет такая (зависящая от $f^k(t)$) последовательность векторов $m^s(t)$, которая приведет к минимальному отклонению $p^k(t)$ от $x^k(t)$.

3. Метод решения с помощью алгоритма прямого распространения

Искусственная нейронная сеть – это компьютерная система и алгоритмы машинного обучения, которые пытаются имитировать работу мозга человека. Они состоят из множества связанных между собой элементов, называемых нейронами, которые обрабатывают информацию и передают ее дальше по сети.

Принцип работы нейронной сети заключается в передаче информации через слои нейронов. Каждый слой обрабатывает данные, полученные от предыдущего слоя, и передает их следующему слою. Нейроны внутри каждого слоя связаны между собой весами, которые определяют важность каждого нейрона для обработки данных.

Первый слой нейронной сети называется входным слоем. Он принимает входные данные, например, изображение или звук, и передает их следующему слою. Следующие слои называются скрытыми слоями. Они обрабатывают данные, полученные от предыдущего слоя, и передают их следующему слою. Последний слой нейронной сети называется выходным слоем. Он выдает ответ на основе обработанных данных.

Каждый нейрон в нейронной сети выполняет две операции: умножение входных данных на вес и суммирование результата смещения (bias). Затем результат проходит через функцию активации, которая определяет, должен ли нейрон активироваться или нет. Функция активации может быть различной, например, сигмоид или ReLU.

В процессе обучения нейронной сети веса и смещения в каждом нейроне регулируются таким образом, чтобы минимизировать ошибку выходных данных [7]. Это делается с помощью алгоритмов оптимизации, таких как градиентный спуск.

Прямое распространение (FFNN) – это простой тип нейронной сети, в котором информация передается только в одном направлении, от входных нейронов к выходным. Он обладает преимуществами в виде простоты реализации и обучения, а также хорошей производительности на задачах классификации и регрессии [16]. Однако его недостатком является плохая адаптация к обработке последовательностей или временных данных.

В задаче имеется несколько величин, характеризующих временную дискретность: θ_u, θ_p, θ_m, θ_x, причем θ_x можно выбирать произвольно: непрерывную кривую, соединяющую начальное и конечное положение аппарата, можно представить последовательностью с любой дискретностью. Остальные величины зависят от свойств измерительных датчиков, точности системы позиционирования и вычислительных возможностей системы управления – мы не можем менять эти величины произвольно [5].

Для расчетов на ЭВМ, все величины так или иначе представляются целыми числами. То есть, можно выбрать такую величину кванта времени τ, что все указанные выше величины дискретностей будут целыми положительными числами [4].

Можно выбрать τ настолько малым, что $T_x = \theta_x / \tau$ будет наибольшим общим делителем величин $T_u = \theta_u / \tau$, $T_p = \theta_p / \tau$, $T_m = \theta_m / \tau$. Поэтому далее будем использовать именно целочисленную нотацию, предполагая справедливость этих соотношений:

(1) $T_u, T_p, T_m \in N$.

(2) $T_x = \text{НОД}(T_u, T_p, T_m)$.

Иначе говоря, T_u / T_x, T_u / T_x, $T_u / T_x \in N$, в этом случае любой момент времени, в котором значения величин в системе существенны для состояния системы, будет иметь вид $t_n = n\tau T_x$, $n = 0, 1, \ldots$, а в начале желаемой траектории $t = 0$. Зададим переход системы из состояния, соответствующего времени t_n, в состояние, соответствующее времени t_{n+1}, в виде модели распространения сигнала в полносвязной ИНС прямого распространения с числом входов, на K большим числа нейронов (так как f не предсказывается),), см. рис. 1.

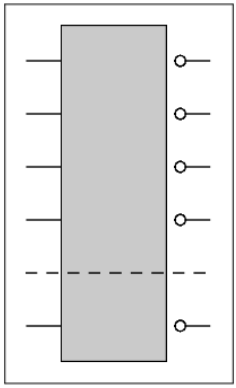

Рис. 1. Схема перехода ИНС

Так как сигналы в предложенной ИНС характеризуют текущее состояние АНПА и воздействия на него, то можно считать эту систему управления адаптивной – при любых внешних воздействиях она будет стремиться удержать АНПА на заданной траектории, если правильно построить целевую функцию. Для упрощения модели, пока построим целевую функцию, исходя из предположения равномерного движения по траектории.

В начале нам известны все величины, кроме $f^k(0)$:

(3) $u_0^j, p_0^k \equiv x_0^k, m_0^s = 0$.

Для нахождения параметров ИНС нужно построить траекторию до точки x_{N-1}^k, где $N = \text{НОК}(T_u, T_p, T_m) / \text{НОД}(T_u, T_p, T_m)$.

В результате N-слойная нейронная сеть и все параметры системы управления будут ее весами и смещениями [13].

4. Случай полносвязной ИНС прямого распространения

Пусть $T_u = 2$, $T_p = 4$, $T_m = 3$, $T_x = 1$, $N = 3$. Тогда полносвязная ИНС прямого распространения будет иметь вид, см. рис. 2.

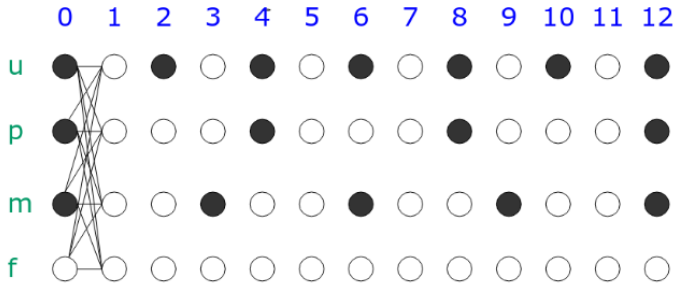

Рис. 2. Частный случай ИНС для системы адаптивного управления

Обозначим веса и смещения через W_l^{da} и b_l^n, соответственно, где l – номер слоя, n – номер нейрона, d, a – номера нейронов донора (в предыдущем слое) и акцептора (в текущем слое). Тогда выход любого слоя запишется в виде:

(4) $$y_l^a = \delta_l^a \left(\sum_{d=0}^{D-1} W_l^{da} y_{l-1}^d + b_l^a \right).$$

(5) $$l = 1, \ldots, N.$$

(6) $$D = J + S + 6.$$

где D – количество признаков, соответствующих разным физическим величинам, J – показания датчиков потока, S – количество движителей.

Кодировка каждого слоя будет следующей:

(4) $$y_0^0 = u_0^0, \ldots, y_0^{J-1} = u_0^{J-1};$$

$$y_0^J = p_0^0, \ldots, y_0^{J+2} = p_0^2;$$

$$y_0^{J+3} = m_0^0, ..., y_0^{J+2+S} = m_0^{S-1};$$
$$y_0^{J+3+S} = f_0^0, ..., y_0^{J+5+S} = f_0^2.$$

Инициируем случайные параметры W_l^{da} и b_l^n. Пройдем ИНС прямо до последнего слоя, получим выходы y_l^a. Размеченные значения выходов будем обозначать со $\hat{y}_l^a$. Начнем обучение с последнего слоя (обратное распространение ошибки) [10].

Коррекция весов последнего слоя (l = 12) будет выглядеть так (дельта-правило):

(8) $\Delta W_l^{da} = -k\sum(y_l^a - \hat{y}_l^a)y_l^a(1-y_l^a)y_{l-1}^d$.

(9) $\Delta b_l^a = -k\sum(y_l^a - \hat{y}_l^a)y_l^a(1-y_l^a)$.

где суммирование ведется по всем имеющимся пакетам сигналов. На практике это эквивалентно нескольким проходам участка или использованием длинной траектории для обучения на ее сегментах. Согласно принятому в методе обратного распространения ошибки предположению, разметка слоя l – 1 будет иметь вид:

(10) $\hat{y}_{l-1}^d = y_{l-1}^d - \sum \delta_l^a W_l^{da}$.

(11) $\delta_l^a = (y_l^a - \hat{y}_l^a)y_l^a(1-y_l^a)$.

При обучении ИНС мы будем пользоваться этим правилом только тогда, когда нет реальных данных (в случае прозрачных точек). Если же реальные данные имеются, будем использовать их вместо соответствующего $\hat{y}_{l-1}^d$, без расчета по обобщенному дельта-правилу [3].

Сигналы, подаваемые в ИНС, будут иметь следующий вид:

1. Показания датчиков потоков u^j в количестве J и каждое может принимать ряд значений (их количество равно мощности P_u признака u^j зависит от кванта и максимального значения);

2. Данные системы позиционирования p^k в виде 3 векторов и каждое может принимать ряд значений (их количество равно мощности P_u признака u^j зависит от кванта и максимального значения);

После этого можно тестировать АНПА: запускать его с разными начальными данными и по ходу его движения он всегда должен стремиться к желаемой траектории, корректируя направление движителями.

5. Численное моделирование работы системы управления

5.1. ДАННЫЕ И СТРУКТУРЫ ДАННЫХ

Для программной реализации используется Python 3.10 с библиотекой NumPy (работа с массивами данных) и matplotlib, осуществляющей построение графиков.

Сигналы от датчиков скорости потоков u представляют собой восемь значений, лежащих в диапазоне от 0 до 9,9 с квантом 0,1, то есть, 8 признаков мощностью 100 каждый. На каждый момент времени, когда эти сигналы доступны, выдаются все 8 значений. Для удобства хранения и работы, входящий сигнал сети записывается в один np.array. Для обучения таких сигналов должно быть много (датасет). Поэтому этот массив представляет собой матрицу, в каждом векторе данных которой (строке) первые 8 (от 0 до 7) значений - показания датчиков скорости потока. Сигналы, получаемые в движении, хранятся в оперативной памяти в виде словарей словарей. Ключом первого словаря является временная метка (номер слоя), значением – второй словарь. Ключом второго словаря является номер сигнала, а значением – массив значений этого сигнала в это время для каждого экземпляра данных.

Сигналы системы позиционирования p, когда они доступны, представляют собой три значения. Их диапазон и квант в разрабатываемом программном обеспечении есть, соответственно, -100, ..., 99,99, и 0,01. То есть, это три признака с мощностями по 20000 каждый. Их записываем аналогично: входящие значения - в массив входящих данных признаками 8, 9, 10. Данные, снятые в движении, - в словарь с теми же ключами признаков. Ясно, что ключи первого словаря не будут охватывать все слои нейронной сети (так как данные появляются не на каждом шаге времени).

Сигналы управления движителями m доступны в каждом слое сети (моменте времени), так как именно ИНС и управляет движителями. В стартовой позиции все движители включены на полную мощность в положительном направлении (вперед): это четыре сигнала, равных 1. Далее они будут принимать значения от –0,9 до 1,0 с квантом 0,1. Таким образом, у нас будет 4 сигнала с мощностью 20 каждый. Их хранение осуществляется аналогично вышеуказанным сигналам: входящий - в массиве, текущие - в словаре.

Параметризация желаемой траектории задается в виде последовательности точек, то есть, в виде матрицы. Структурой данных для неё может быть массив numpy.array. В каждый момент времени (в каждом слое ИНС) будет присутствовать либо предсказанное значение *p*, либо значение, пришедшее из навигационной системы. Поэтому и значения из словарей на этапе первичного обучения никак не будут взаимодействовать с желаемой траекторией [9,11]. Грубо говоря, система должна сначала научиться «рулить», а уже потом должна обучаться движению по траектории.

5.2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

В качестве примера создадим нейронную сеть, конфигурацию см. рис. 3. В ней 12 слоев, 12 нейронов, каждый из которых получает 12 сигналов на входе. Три сигнала, которые каждый раз приходят «извне», – это внешние течения *f*, которые будем полагать задаваемыми точно такими же параметрами, как *u*: с теми же диапазонами и квантованием [15].

```
# Количество образцов данных при обучении ИНС
A = 3
# количество временных отсчетов при обучении ИНС
T = 12
# Количество нейронов сети
N = 12
# период обновления данных с датчиков потоков
Tu = 4
# период обновления данных с датчиков позиционирования
Tp = 2
# гамма, пороговое значение для функции управления при обучении ИНС.
gm = 20
# скорость обучения
mu = 0.05
# количество эпох обучения
epochs = 100
```

Рис. 3. Конфигурация ИНС

Соображением к этому является то, что f – это скорости, которые редко когда могут превысить искусственно создаваемые потоки u. Нам потребуются следующие функции:

1. генерации весов и смещений сети;
2. прохода слоя вперед;
3. проход нескольких слоев вперед;
4. коррекции весов и смещений последнего слоя (по меткам выхода нейронной сети);
5. коррекции весов и смещений скрытых слоев (с помощью гипотезы обобщенного дельта-правила).

Далее генерируются веса и смещения, и вызывается функция ann_train для обучения нейронной сети [12,14], см. рис. 4.

```
npdata = np.array([[start(i) for i in range(T)] for _ in range(A)])
X = get_x(T + 2)
F = generate_f(T, A)
labels_dict = {}
for l in range(1, T + 1):
    labels_dict[l - 1] = umpf(l, F, X, gm)


init_weight_list = [np.array([[0.2 * random() + 0.9 for _ in range(N)] for _ in
range(N)]) for _ in range(T)]
init_bias_list = [np.array([0 for _ in range(N)]) for _ in range(T)]


weight_list, bias_list, stat_list = ann_train(
    npdata,
    labels_dict,
    init_weight_list,
    init_bias_list,
    sigmoid,
    mu,
    epochs,
)
```

Рис. 4. Процесс генерации весов и смещений

Внутри функции ann_train основной цикл проходит по каждой эпохе обучения. Для каждой эпохи проходит цикл по каждому слою ИНС [6, 8]. Обучение каждого слоя методом обратного распространения ошибки с использованием градиентного спуска. Обновленные весовые матрицы и векторы смещений сохраняются в weight_list и bias_list, см. рис. 5.

```
for _ in range(epoch):
    # Для каждого слоя вычисляются выходные данные
    layers_data = layers(layers_data[0], weight_list, bias_list, act_func)

    # метки текущего слоя
    full_labels_dict = dict(labels_dict)

    # статистика ошибок слоя
    layer_stat = list()

    for l in range(layers_number):
        sln = layers_number - l
        labels = labels_dict[sln - 1]

        if l:
            GL = generate_labels(layers_data[sln + 1], full_labels_dict[sln],
weight_list[sln])

        for i, li in enumerate(labels):
            if float('inf') in li:
                labels[i] = GL[i]

        full_labels_dict[sln - 1] = labels

        # Обучение каждого слоя методом обратного распространения ошибки с
использованием градиентного спуска.
        # Обновленные весовые матрицы и векторы смещений сохраняются в
weight_list и bias_list.
        weight_list[sln - 1], bias_list[sln - 1], layer_stat_list = layer_train(
            layers_data[sln - 1],
            full_labels_dict[sln - 1],
            weight_list[sln - 1],
            bias_list[sln - 1],
            act_func,
            mu,
            1)
        layer_stat.append(layer_stat_list[0])
    stat_list.append(layer_stat)
```

Рис. 5. Функция ann_train. Процесс обучения

В результате обучения имеется информация о T шагах движения аппарата. Для каждого шага имеется матрица, ее строки соответствуют экземплярам данных (то есть разным местам или временам испытаний), а столбцы - показаниям датчиков в этом испытании.

С помощью библиотеки matplotlib производим построение графиков абсолютной величины отклонения аппарата от желаемой траектории в каждый момент времени в каждом испытании.

6. Заключение

В данной исследовательской работе были рассмотрены методы решения адаптивных задач для интеллектуальной системы адаптивного автоматического управления с заданной целью.

Целью настоящей работы является исследование возможности разработки модели интеллектуальной системы адаптивного автоматического управления с заданной целью, а именно – подготовка данных для искусственной нейронной сети, объединяющих показания сонара, лидара и бортовых датчиков в виде, оптимально подготовленном к решению задач управления в реальном времени.

Адаптивные системы и нейронные сети могут помочь улучшить производительность и эффективность неопределенных динамических систем, а также сделать управление ими более гибким и адаптивным к изменениям внешних условий.

Применение элементов искусственного интеллекта существенно расширяет возможности и скорость обработки информации, а при применении механизмов самообучения и снижает количество ошибок при ответах.

Приложение 1

Результаты выполнения испытаний в соответствии с разработанным методом:

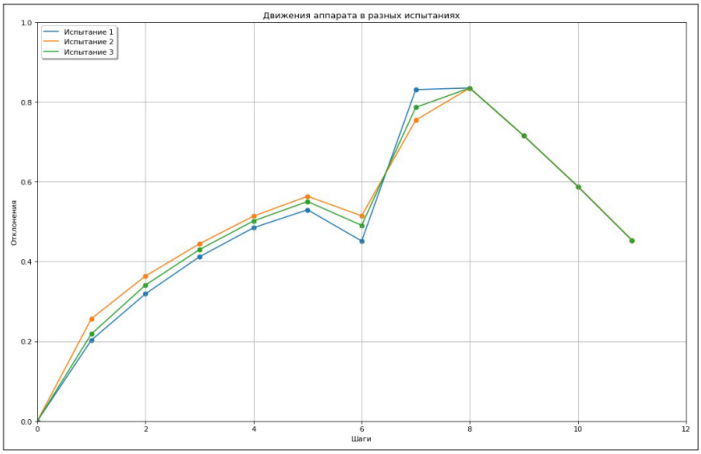

Рис. 6. Абсолютные величины отклонения аппарата от желаемой траектории в каждый момент времени в каждом испытании

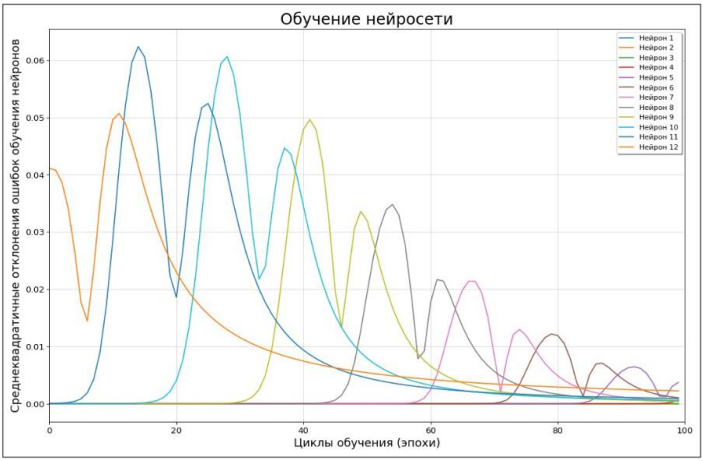

Рис. 7. Процесс обучения слоя нейронов с коррекцией ошибок

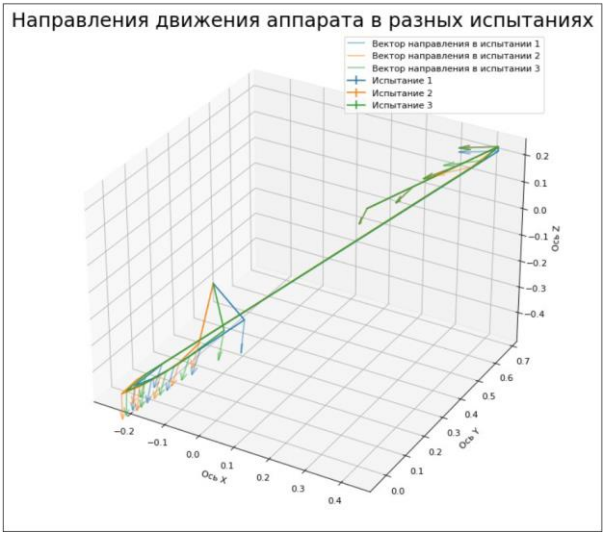

Рис. 8. График направления движения аппарата в плоскости XYZ

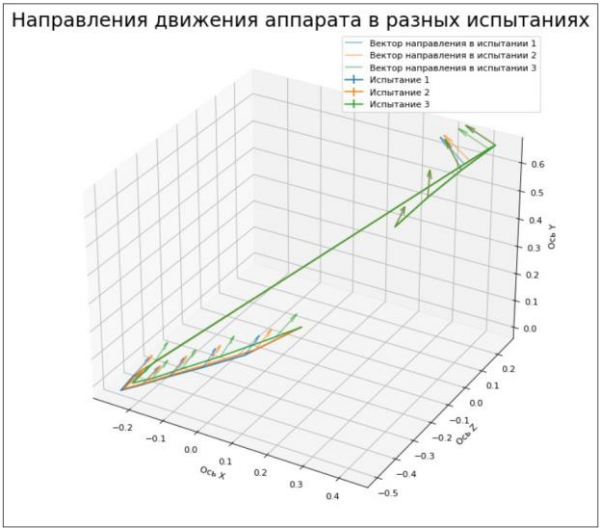

Рис. 9. График направления движения аппарата в плоскости XZY

Литература

1. БАХАРЕВ С.А., КАРАСАЕВ В.В., КАРАСАЕВ А.В. *Использование автономных необитаемых подводных аппаратов в процессе изучения Мирового океана* // Научные труды Дальрыбвтуза. – 2012. – №26 – С. 63–72.
2. САРУХАНЯН А.И. *Перспективы использования искусственных нейронных сетей при управлении группировкой автономных необитаемых подводных аппаратов* // Техника средств связи. – 2018. – №4 (144) – С. 127–129.
3. ЦЫПКИН Я.З. *Адаптация и обучение в автоматических системах* // Главная редакция физико-математической литературы изд-ва «Наука», М. – 1968. – 400 с.
4. IOANNOU P.A., TSAKALIS K.S. *Robust discrete-time adaptive control* // Adaptive and Learning Systems: Theory and Applications. Plenum Press. – 1986. – P. 73–85.
5. JENKINS B., KRUPADANAM A., ANNASWAMY A.M. *Fast adaptive observers for battery management systems* // IEEE Transactions on Control Systems Technology. – 2019. – P. 1–14.
6. KAELBLING L.P., LITTMAN M.L., MOORE A.W. *Reinforcement Learning: A Survey* // Journal of artificial intelligence research. – 1996. – Vol. 4. – P. 237–285.
7. KRSTIC M. *Control has met learning: Aspirational lessons from adaptive control theory* // Online Event: Control Meets Learning Seminar. – 2021.
8. LEWIS F.L., VRABIE D.L. *Reinforcement learning and adaptive dynamic programming for feedback control* // IEEE Circuits and Systems magazine. – 2009. – Vol. 9. – P. 32–50.
9. LI W., KRSTIC M. *Stochastic adaptive nonlinear control with filterless least squares* // IEEE Transactions on Automatic control. – 2020. – Vol. 66, No. 9. – P. 3893–3905.
10. PETERSON B.B., NARENDRA K.S. *Bounded error adaptive control* // IEEE Transactions on Automatic Control. – 1982. – Vol. 27, – P. 1161–1168.
11. POLYCARPOU M.M. *Stable adaptive neural control scheme for nonlinear systems* // IEEE Transactions on Automatic control. – 1996. – Vol. 41, – P. 446–451.

12. POWELL W.B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality* // IIE Transactions. – 2007. – Vol. 41, – P. 168–169.
13. ROVITHAKIS G.A., CHRISTODOULOU M.A. *Adaptive control of unknown plants using dynamical neural networks* // IEEE Transactions on Systems, Man, and Cybernetics. – 1994. – Vol. 24, – P. 400–412.
14. SUTTON R.S., BARTO A.G. *Reinforcement learning: An introduction* // MIT press. – 1998. – 342 p.
15. QU Z., THOMSEN B., ANNASWAMY A.M. *Adaptive control for a class of multi-input multi-output plants with arbitrary relative degree* // IEEE Transactions on Automatic Control. – 2020. – Vol. 65, – P. 3023–3038.
16. *What's New in Artificial Intelligence from the 2022 Gartner Hype Cycle* [Электронный ресурс] // Gartner. URL: https://tinyurl.com/4z4vrhcy (дата обращения: 01.06.2023).

NEURAL NETWORK INTEGRATED INTO THE ADAPTIVE CONTROL SYSTEM OF AN AUTONOMOUS UNINHABITED UNDERWATER VEHICLE

Valeria Romanova, V. I. Vernadsky Crimean Federal University, Simferopol, student (lero4ka2004ro@gmail.com).
Sergei Zuev, Belgorod State Technological University named after V.G. Shukhov, Belgorod, Candidate of Physical and Mathematical Sciences, Associate Professor (sergey.zuev@bk.ru).

Abstract. This paper is devoted to the research and development of a model of an intelligent system of adaptive automatic control with a given target based on the use of artificial neural networks. The main goal of the work is to prepare data for an artificial neural network that integrates information from sonar, lidar and onboard sensors, providing optimal data preparation for real-time decision making. The result of the research is a model of an intelligent system capable of adapting to changing environmental conditions and making decisions based on data from various sources. This significantly improves the efficiency of underwater vehicle control. The significance of this work is to create more effective and reliable methods of controlling technical objects. The developed model uses telemetry to prepare data for the operation of a feed-forward artificial neural network, which can be useful in the automotive and marine industries, aviation and other areas where accurate and fast control in a variable environment is required. The study of this intelligent adaptive automatic control system represents an important step in the development of

methods for controlling technical objects and may find application in a variety of industries where accurate and fast control tasks are relevant.

Keywords: artificial neural network, AUV, control system model, error back propagation.

УДК 004.8
ББК 32.813.5

Статья представлена к публикации
членом редакционной коллегии …заполняется редактором…

Поступила в редакцию …заполняется редактором…
Опубликована …заполняется редактором…