DOI 10.37882/2223-2966.2024.8.07

СРАВНЕНИЕ КЛАССИЧЕСКИХ МЕТОДОВ ЧИСЛЕННОГО РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С МЕТОДОМ НЕЙРОННЫХ СЕТЕЙ

Ахметов Ильшат Зуфарович
Казанский (Приволжский) федеральный университет
ilshat.achmetov@gmail.com

Аннотация. В данной работе построен метод искусственных нейронных сетей для численного решения различных типов дифференциальных уравнений. Данный метод так же известен как PINN, то есть physics-informed neural networks. Суть метода заключается в минимизации квадрата невязки уравнения, в котором решение уравнения ищется с помощью искусственной нейронной сети. В настоящее время активно ведутся исследования в области данного метода, в связи с чем возникла необходимость в его детальном изучении и сравнении с другими методами. Приведен ряд примеров для сравнения точности данного метода с классическими методами численного решения дифференциальных уравнений. Для имплементации метода автором была разработана программа на языке Python с применением фреймворка глубокого обучения PyTorch.

Ключевые слова: искусственные нейронные сети, дифференциальные уравнения, аппроксимация, численные методы.

COMPARISON OF CLASSICAL METHODS FOR NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS WITH THE NEURAL NETWORK METHOD

I. Akhmetov

Summary. In this paper, the method of artificial neural networks was constructed for the numerical solution of various differential equation types. This method is also known as the PINN, i.e., physics-informed neural network. The main idea of the method is to minimize the squared residual of the equation, in which the solution of the equation is sought using an artificial neural network. Currently there are much research related to this method, thus there is need to examine this method in detail and compare with other methods, first the classical ones. Several examples were provided to compare this method with classical methods for the numerical solution of differential equations in terms of accuracy. To implement the neural network method author developed a program on python programming language using PyTorch — a framework for deep learning.

Keywords: artificial neural networks, differential equations, approximation, numerical methods.

Введение

В настоящее время активно ведутся исследования в области применения нейронных сетей для численного решения дифференциальных уравнений. На западе данный метод известен прежде всего под названием PINN, что является сокращением от physics-informed neural network. На русский язык это можно перевести как физически обусловленную нейронную сеть, то есть нейронную сеть, параметры которой определены физическими законами. Такое название объясняется тем, что данный метод применялся в основном для решения физических уравнений, выведенных на основе законов физики.

Исследования в данной области ведутся еще с 90-х годов прошлого столетия. В настоящее время исследователи применяют данный метод для решения обратной задачи уравнений Максвелла [1], для решения уравнений Навье-Стокса [2], а также для решения нелинейных интегро-дифференциальных уравнений [3]. Так же данный метод может применяться для решения дифференциальных уравнений дробного порядка [4], [5].

У данного метода очень много других применений. Все это вызывает необходимость исследовать вопрос сходимости данного метода, а также его точности. Цель данной работы — построить алгоритм решения дифференциальных уравнений методом нейросетей и сравнить точность метода с точностью таких классических методов, как методы Галеркина и МКЭ для решения краевых задач, а также с методами Эйлера и Рунге-Кутта 4-го порядка для решения задачи Коши. Для сравнения будет использован ряд типовых примеров с известным аналитическим решением.

Постановка задачи

Рассмотрим дифференциальное уравнение в операторной форме

$$Du = f \tag{1}$$

определенное на некоторой области Ω, а также оператор A, определяющий граничные и начальные условия на $\partial\Omega$

$$Au = g \tag{2}$$

Искомую функцию u будет аппроксимировать с помощью искусственной нейросети $\hat{u} = NN$. Выберем n узлов на области Ω. К примеру, в одномерном случае для уравнения, определенного на $x \in [a,b]$ можно выбрать узлы равномерно с шагом $h = \frac{b-a-2\varepsilon}{n-1}$. Здесь ε — некая очень малая положительная константа $b - a \gg \varepsilon$, нужная для того, чтобы не включать граничные узлы в основную область. Таким образом, узлы будут определяться формулой $x_i = a + \varepsilon + hi, i \in \{0,1,\ldots n-1\}$. На $\partial\Omega$ аналогичным образом выберем m узлов.

Пусть $R(\hat{u}, x_i)$ — значение невязки в точке x_i на Ω при аппроксимации решения с помощью $\hat{u}$. Аналогично, $r(\hat{u}, x_i)$ — значение невязки на границе области $\partial\Omega$ либо в узлах, значение функции u в которых определяется начальными условиями. Построим так называемую функцию потерь

$$L(\hat{u}) = \frac{\sum_{i=1}^{n} R^2(\hat{u}, x_i) + \alpha \sum_{j=1}^{m} r^2(\hat{u}, x_j)}{N} \quad (3)$$

По сути, это функционал от аппроксимации $\hat{u}$, который необходимо минимизировать для нахождения аппроксимации. Здесь $\alpha > 0$ — коэффициент, отвечающий за то, насколько сильно стоит штрафовать функционал за нарушение граничным условий. Выбирать его можно по-разному, например так, чтобы выполнилось условие $\alpha m = 0.1n$. Можно сказать, необходимо найти функцию $\hat{u}$, такую, что

$$\hat{u} = \underset{u}{argmin}\left[L(u)\right] \quad (4)$$

Если есть возможность подобрать аппроксимацию $\hat{u}$, в точности удовлетворяющую граничным условиям, то уравнение (3) упрощается. К примеру, если функция определена на $x \in [0,1]$ и заданы граничные условия такие, что $u(0) = 1, u(1) = 2$, то аппроксимацию можно представить в виде $\hat{u}(x) = 1 + x + (x-1)(2-x)NN(x)$. Не трудно видеть, что в данной форме аппроксимация в точности удовлетворяет граничным условиям. Тогда уравнение (3) запишется в виде

$$L(\hat{u}) = \frac{\sum_{i=1}^{n} R^2(\hat{u}, x_i)}{N}. \quad (5)$$

К примеру, именно в такой форме, удовлетворяющей граничным условиям нейронные сети были применены исследователями в работе [6].

Таким образом, решение дифференциального уравнения было сведено к задаче оптимизации.

Теоретические сведения

В 1989 Джордж Цыбенко доказал [7] универсальную теорему аппроксимации, утверждающую, что любая непрерывная функция $f(x)$ на кубе $[0,1]^n$ может быть сколь угодно точно аппроксимирована с помощью функции вида

$$G(x) = \sum_{i=1}^{N} \alpha_i \varphi(w_i^T x + \theta_i) \quad (6)$$

при надлежащем выборе значений $\alpha \in R^N, w_i \in R^N, \theta \in R^N$ и достаточно большого значения N. Здесь φ — сигмоида. В данной формуле нетрудно увидеть простейшую полносвязную нейронную сеть с одним скрытым слоем. Здесь и далее такие сети будет обозначать как MLP, что значит multilayer perceptron.

Несмотря на то, что данная теорема указывает на возможность аппроксимации непрерывной на $[0,1]^n$ функции в форме (6) она не дает никаких указаний, как данную аппроксимацию найти.

Рассматриваемый метод (3) является по сути вариацией дискретного варианта метода наименьших квадратов. Нужно подчеркнуть — именно вариацией, поскольку в классическом методе наименьших квадратов искомая функция аппроксимируется линейной комбинацией неких базисных линейно независимых функций, например полиномов.

В данном же методе аппроксимацией служит нейронная связь. Ниже представлена функция, определяющая простую MLP с двумя скрытыми слоями и N нейронами в скрытых слоях, такой что $NN : R^n \to R$

$$NN(x) = W_3 \varphi(W_2 \varphi(W_1 x + b_1) + b_2) + b_3 \quad (7)$$

Здесь

$$W_3 \in R^{1,N}, W_2 \in R^{N,N}, W_1 \in R^{N,n}, b_1, b_2 \in R^n, b_3 \in R.$$

В качестве функции активации φ могут использоваться, к примеру гиперболический тангенс или сигмоида, хотя на этом выбор не ограничен. На рис. 1, представлена схема такой нейросети.

Для нахождения производной данной функции по одному из параметров, например для нахождения производной $NN(x)$ по компоненте $w_{i,j}^2$ в i-строке и j-столбце матрицы W_2 нужно использовать правило цепочки дифференцирования сложной функции. Даже для небольшой двуслойной сети число параметров может исчисляться тысячами. Очевидно, что если применять стандартный подход МНК, предполагающий нахождение

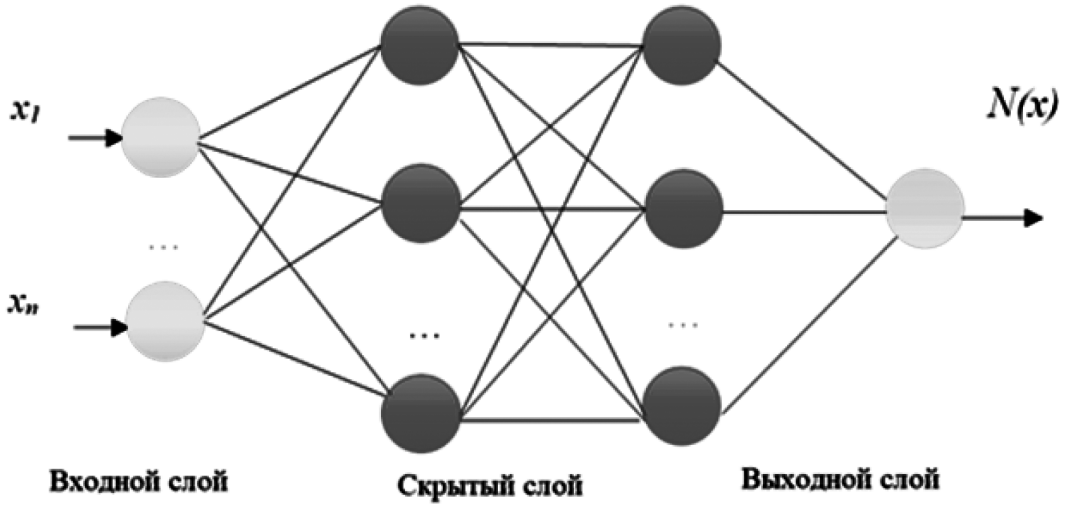

Рис. 1. Схема искусственной нейронной сети с 2 скрытыми слоями

производных функционала по параметрам аппроксимации и приравнивания нулю, то получится система из тысяч нелинейных уравнений, которая практически нерешаема. Поэтому в данном методе используют другой подход, использующий модификацию градиентного спуска либо метода Ньютона, о чем будет идти речь в следующем параграфе.

Вообще говоря, можно выбрать различные типы архитектуры нейронной сети, главное, чтобы получившаяся аппроксимация была достаточное число раз дифференцируема. В данной статье будет использована только MLP.

В книге Марчука Г.И. [8] на стр. 75 дано условие, при котором МНК применительно к дифференциальным уравнениям сходится. Автор утверждает, что последовательные приближения u^h метода наименьших квадратов сходятся в F к точному решению u уравнения (1), если уравнение однозначно разрешимо, последовательность DF^h полна в множестве значений $\ddot{O}(D)$ и обратный оператор D^{-1} существует и ограничен. Здесь F — гильбертово пространство, в котором определена искомая функция u, F^h — конечномерное пространство с некоторым базисом $\phi_1^h, \phi_2^h, \ldots \phi_N^h$, такое, что $F^h \in \ddot{O}(D)$ — область определения D. Однако здесь нужно сделать замечание, что в данном методе аппроксимация определяется не как линейная комбинация линейно-независимых функций, а как сложная функция, суперпозиция простых функций.

Описание алгоритма

Пусть θ — параметры нейросетевой аппроксимации (7), такие как W_3, W_2, b_1 и т. д. Для решения задачи (4) необходимо их каким-то образом настроить. Как уже отмечалось, нахождение частных производных и приравнивание к нулю в данном случае не подойдет. Но есть другие методы, такие как модификации градиентного спуска и метода Ньютона. Их суть заключается в итеративном подборе параметров

$$\theta_n = \theta_{n-1} + \alpha_n \bar{d} \tag{8}$$

Здесь α_n — некое положительное значение, так называемый learning rate, то есть скорость обучения. В частном случае можно принять это значение за константу α, а $\bar{d}$ — некое приращение параметров θ. В классическом градиентном спуске это просто антиградиент функции потерь (3) или (5) по параметрам аппроксимации, то есть $-L'_\theta\left(\hat{u}\right)$, а в методе Ньютона — это произведение матрицы, обратной к гессиану на антиградиент функции, то есть $-H\left(L\left(\hat{u}\right)\right)^{-1} L'_\theta\left(\hat{u}\right)$. В методе LBFGS, который будет использован в данной работе и который является модификацией метода Ньютона вместо данного значения, будет находиться его аппроксимация. Теперь опишем алгоритм настройки параметров аппроксимации θ.

1. Построим аппроксимацию $\hat{u}$, функцию потерь (3) либо (5) и инициализируем параметры аппроксимации θ_0 некоторыми начальными значениями, выберем α, максимальное количество итераций N и некоторые очень малые положительные константы ϵ_1, ϵ_2.
2. На шаге n алгоритма, $n \geq 1$ находим новое значение параметров аппроксимации θ_n по правилу (8) с помощью алгоритма LBFGS
3. Если шаг $n \geq N$ либо выполняется $\|\theta_n - \theta_{n-1}\| < \epsilon_1$ или $\left\|L(\hat{u}(\theta_n)\right\| < \epsilon_2$, то прекращаем итеративный процесс, в противном случае возвращаемся к шагу 2.

Как видно, данный алгоритм предполагает, что если значение квадрата невязки достаточно близко к нулю, то

аппроксимация является хорошим приближением к искомому решению.

Нужно так же отметить, что для нахождения производных и обновления параметров нейросети θ в современной практике используется алгоритм backpropagation, суть которого заключается в последовательном нахождении производных от слоя к слою, дифференцируя функцию потерь сразу по векторам либо матрицам параметров нейросети, поскольку нахождение производной функции потерь по каждому параметру-скаляру крайне долго и неэффективно. К примеру, производная $\frac{\partial L}{\partial W_2}$ функции (7) находится по правилу цепочки, используя найденные раннее значения производных $\frac{\partial L}{\partial W_3}, \frac{\partial L}{\partial \delta_2}$, где $\delta_2 = W_2\varphi(W_1x + b_1) + b_2$.

Численные эксперименты

Далее будет приведен ряд примеров для сравнения метода нейронных сетей с классическими методами. В данной работе в методе Галеркина аппроксимация выбирается так, чтобы в точности удовлетворять граничным условиям. Для МКЭ в одномерном случае используются простые кусочно-линейные полиномы, а в двумерном — прямоугольные элементы. В МКЭ интегрирование производится с помощью метода средних прямоугольников. Во всех случаях, включая методы Эйлера и Рунге-Кутта 4-го порядка узлы выбираются равномерно на области Ù.

Для оценки точности решения будет использоваться норма $\|x\|_\infty = \max(|x_1|, |x_2|, .. |x_n|)$, так как данная норма хорошо интерпретируема и понятна человеку.

Для применения метода нейросетей автором была разработана программа на языке Python с применением фреймворка глубокого обучения PyTorch.

Всюду в примерах ниже будут использованы параметры нейросетевой аппроксимации и настройки ее параметров, представленные в таблице 1.

Таблица 1.

Параметры обучения и нейросетевой аппроксимации

Количество скрытых слоев	1
Количество нейронов в слое	20
Скорость обучения á	0.1
Количество итераций метода	40

Теперь рассмотрим примеры для сравнения нейросетевого метода с классическими численными методами.

Пример 1.

$$u_{tt} + 0.2u_t + u = -0.2e^{(-0.2t)}\cos(t)$$
$$u(0) = 0$$
$$u(1) = \sin(1)e^{-0.2}$$
$$t \in [0,1]$$

Точное решение данной задачи

$$u(t) = e^{-0.2t}\sin(t)$$

В методе Галеркина решение аппроксимировалось с помощью функции

$$\tilde{u}(t) = tsin(1)e^{-0.2} + \sum_{i=1}^{10}\alpha_i(1-t)t^i$$

То есть количество базисных функций в методе Галеркина — 10.

В таблице 2 представлено сравнение максимальных абсолютных погрешностей рассмотренных методов. Здесь и далее под ИНС понимается метод искусственных нейронных сетей.

Количество узлов в методе ИНС большее, чем 15 не приводило к повышению точности.

Таблица 2.

Сравнение погрешностей методов

Метод	$\|u - \hat{u}\|_\infty$	Количество базисных функций/узлов
ИНС	$7.6e-7$	15
МКЭ	$1.83e-8$	1000
Метод Галеркина	$2.37e-13$	10

Пример 2.

Решение задачи Коши для системы ОДУ

$$x = -y$$
$$y' = x + \cos(t)$$
$$x(0) = 0$$
$$y(0) = 0$$
$$t \in [0,1]$$

Для данной задачи известно точное решение

$$x = -0.5tsin(t)$$
$$y = tcos(t) + 0.5\sin(t)$$

Нейросетевую аппроксимацию выберем так, чтобы она в точности удовлетворяла граничным условиям.

$$\hat{u}_1(t) = tNN_1(t)$$
$$\hat{u}_2(t) = tNN_2(t)$$

Таблица 3.

Сравнение погрешностей методов

Метод	$\|u - \hat{u}\|_\infty$	Количество узлов
ИНС	$3.05e-5$	20
Метод Эйлера	$6.07e-4$	1001
Метод Рунге-Кутта 4	$1.09e-7$	101

Точность методов Рунге-Кутта и Эйлера можно улучшить, увеличивая число узлов. В методе ИНС увеличение числа узлов не привело к улучшению точности.

Пример 3.

Рассмотрим уравнение в частных производных

$$u_{xx} + u_{yy} + 1 = 0$$
$$u(x,-1) = u(1,y) = 0$$
$$u(-1,y) = u(x,1) = 0$$
$$x \in [-1,1] y \in [-1,1]$$

Данное уравнение описывает движение ламинарной жидкости в трубе. Пример взят из книги Флетчера [9] на стр. 24. В ней же показано, что решением уравнения является функция

$$u(x,y) =$$
$$= \left(\frac{8}{\pi^2}\right)\sum_{i=1,3,5..}^{\infty}\sum_{j=1,3,5..}^{\infty}\frac{(-1)^{(i+j)/2-1}}{ij(i^2+j^2)}\cos\left(ix\frac{\pi}{2}\right)\cos\left(jy\frac{\pi}{2}\right)$$

В методе Галеркина искомая функция будет аппроксимироваться выражением

$$\tilde{u}(t) = \sum_{i=1}^{6}\sum_{j=1}^{6}\alpha_{i,j}\left(1-x^2\right)^i\left(1-y^2\right)^j$$

Такая аппроксимация состоит из 36 базисных функций.

В МКЭ число узлов равно $50X50 = 2500$. Для нейросетевого метода количество узлов выбрано равным $20X20 = 400$, а аппроксимация была представлена в виде, удовлетворяющем граничным условиям:

$$\hat{u}(x,y) = NN(x,y)\left(1-x^2\right)\left(1-y^2\right)$$

В таблице 4 представлено сравнение точности методов на данном примере.

Таблица 4.

Сравнение погрешностей методов

Метод	$\|u - \hat{u}\|_\infty$	Количество узлов/базисных функций
ИНС	$1.51e-3$	400
МКЭ	$2.13e-4$	2500
Метод Галеркина	$7e-5$	36

Заключение

Здесь было приведено лишь небольшое число примеров, показывающих, что нейросетевой метод сильно проигрывает в точности классическим методам. Сравнительно небольшое число узлов в методе ИНС может произвести ложное впечатление о точности метода, однако в методе ИНС увеличение количества узлов приводило к увеличению точности лишь до некоторого момента, причем число узлов при этом было сравнительно невелико, в то время как для классических методов увеличением числа базисных функций/узлов теоретически можно достичь какой угодно точности, на практике, разумеется, ограниченной способом хранения и обработки числе в ЭВМ. Автором было проведено намного больше экспериментов, во всех них результат был аналогичен. Стоит отметить, что выбор параметров аппроксимации и обучения ИНС сильно влияет на результат, и здесь выбор параметров — процесс скорее эвристический.

Однако у данного метода есть и свои преимущества. Первое — это некоторая универсальность, т.к алгоритм минимизации функции потерь (3) и (5) остается тем же, независимо от класса уравнения, будь то задача Коши для ОДУ, эллиптическое, параболическое либо любое другое уравнение, в то время как для классических методов схемы решения могу существенно отличаться для разных классов уравнений, и следовательно нужно писать различные программы для их решения. Более того, алгоритм нейросетевого метода практически без изменений может быть применен для решения интегральных и интегро-дифференциальных уравнений [10]. Так же данный метод может быть применен для решения жестких ОДУ [11]. Дополнительно о преимуществах и недостатках нейросетевого метода можно прочитать в обзорной статье [12].

В настоящее время статьи на данную тему публикуются в таких ведущих журналах, как Nature [13]. Это говорит о том, что исследования свойств и поиски путей улучшения, а также изучение вопросов практического применения данного метода активно продолжаются.

В качестве возможных путей модернизации данного метода можно отметить поиск более совершенных функций потерь (3), алгоритмов минимизации, а также разработка новых архитектур нейросетей, к примеру недавно вышедшая архитектура KANN (Kolmogorov-Arnold neural network) [14]

ЛИТЕРАТУРА

1. Shiyuan Piao and Hong Gu and Aina Wang and Pan Qi. A Domain-adaptive Physics-informed Neural Network for Inverse Problems of Maxwell's Equations in Heterogeneous Media // APL Photonics 7, 011301 (2022). arXiv:2308.06436
2. Xiaowei Jin, Shengze Cai, Hui Li, George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations // Journal of Computational Physics, Volume 426, 2021, 109951, ISSN 0021-9991, \url{https://doi.org/10.1016/j.jcp.2020.109951}
3. Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, Shuo Hao. «A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations» // Journal of Computational Physics, Volume 462, 2022.
4. N.T. Duc, A.F. Galimyanov, I.Z. Akhmetov. Artificial neural network method for solving a fractional order differential equation with the cauchy-type problem // 2023 International Russian Smart Industry Conference (SmartIndustryCon) / IEEE. — 2023. — Pp. 329–334, doi: 10.1109/SmartIndustryCon57312.2023.10110823.
5. N.T. Duc, A.F. Galimyanov, I.Z. Akhmetov. Neural network method for solving fractional differential equations á with the dirichlet problem // 2023 International Russian Smart Industry Conference (SmartIndustryCon) / IEEE. — 2023. — Pp. 295–300, doi: 10.1109/SmartIndustryCon57312.2023.10110785.
6. Lagaris I.E, Likas A. and Fotiadis D.I. Artificial neural networks for solving ordinary and partial differential equation // IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 987–1000, Sept. 1998.
7. Cybenko G.V. Approximation by Superpositions of a Sigmoidal function // Mathematics of Control Signals and Systems. — 1989. — Т. 2, № 4. — С. 303–314.
8. Марчук Г.И. Методы вычислительной математики. Издательство «Наука», 1977.
9. Флетчер К. Численные методы на основе метода Галёркина. Издательство «Мир», Москва, 1988.
10. Effati Sohrab & Buzhabadi, Reza. (2012). A neural network approach for solving Fredholm integral equations of the second kind // Neural Comput & Applic. 21. 1–10. 10.1007/s00521-010-0489-y.
11. Pouyan Nasiri, Roozbeh Dargazany. 2022. «Reduced-PINN: An Integration-Based Physics-Informed Neural Networks for Stiff ODEs». arXiv:2208. 12045.ibitem
12. Cuomo S., Di Cola V.S., Giampaolo F. et al. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. J Sci Comput 92, 88 (2022). https://doi.org/10.1007/s10915-022-01939-z.
13. Wang F., Zhai Z., Zhao Z. et al. Physics-informed neural network for lithium-ion battery degradation stable modeling and prognosis. Nat Commun 15, 4332 (2024). https://doi.org/10.1038/s41467-024-48779-z
14. Liu Ziming, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljaиж, Thomas Y.Hou, and Max Tegmark. «Kan: Kolmogorov-arnold networks.» arXiv preprint arXiv:2404.19756 (2024).

Журнал «Современная наука: актуальные проблемы теории и практики»