Записки научных
семинаров ПОМИ
Том 499, 2021 г.

А. В. Музалевский, С. И. Репин

АПОСТЕРИОРНЫЙ КОНТРОЛЬ ТОЧНОСТИ ПРИБЛИЖЕННЫХ РЕШЕНИЙ КРАЕВЫХ ЗАДАЧ, ПОЛУЧЕННЫХ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

§1. Введение

Теория и методы аппроксимации уравнений в частных производных изучены достаточно хорошо. Имеется большое количество вычислительных методов, которые позволяют получать приближенные решения для основных классов таких уравнений. Они основаны на использовании теории аппроксимации функций и известных методов вычислительной математики. Соответствующие процедуры реализованы в различных пакетах прикладных программ (таких как, например, MATLAB, ANSYS, COMSOL). В последние годы стал развиваться принципиально иной подход к решению прикладных задач, основанный на обучении нейронных сетей (Deep Neural Networks (DNN)). Появляется все больше работ (см., например, [1–8] и цитированную там литературу), где DNN используются для решения краевых и начально–краевых задач (далее для краткости мы будем называть их DNN–решениями). Сравнение таких решений с решениями, построенными классическими методами не является темой данной статьи. По точности и эффективности такие методы как метод конечных элементов, конечных объемов, или разностные методы пока остаются вне серьезной конкуренции. Однако методы обучения нейронных сетей быстро развиваются и совершенствуются, так что вполне можно прогнозировать их широкое использование в недалеком будущем. При этом неизбежно возникает вопрос о точности DNN–решений. Мы показываем, что надежный контроль точности таких решений обеспечивает теория апостериорных оценок функционального типа, которая была ранее разработана для уравнений в частных производных различных типов (подробное изложение соответствующей теории можно найти в [9]).

Ключевые слова: апостериорные оценки, машинное обучение, приближенное решение уравнений в частных производных.

В статье обсуждаются эллиптические уравнения, но изложенный метод допускает широкие обобщения на другие классы уравнений. Применяя эти оценки мы анализируем DNN–решения, построенные для уравнения Пуассона и обсуждаем как их точность коррелирует с величиной функции потерь. Статья организована следующим образом. В §2 мы даем краткий обзор результатов по апостериорному контролю точности приближенных решений. §3 посвящен адаптации апостериорных оценок к специфическим особенностям DNN–решений. Проблема машинного обучения нейронных сетей в приложении к решению краевых задач обсуждается в §4, а §5 содержит результаты экспериментов и основные выводы.

§2. Апостериорный контроль приближенных решений

Достоверность приближенных решений математических моделей, полученных при помощи тех или иных методов является важной проблемой, которая изучается достаточно давно. Если говорить о моделях, связанных с дифференциальными уравнениями, то существует два основных подхода к решению этого вопроса, которые основаны на априорных и апостериорных оценках точности приближенных решений. Априорные оценки обычно возникают, когда дифференциальное уравнение (или система уравнений) приближенно аппроксимируется некоторой конечномерной задачей (например в рамках метода конечных элементов). Эти оценки имеют асимптотический характер и устанавливают скорость убывания погрешности, когда размерность конечномерной задачи стремится к бесконечности. В данной работе мы обсуждаем приближенные решения, построенные нейронными сетями. В этом случае приближенная модель всегда имеет конечную размерность. Это обстоятельство (а также ряд других особенностей DNN–решений) принципиально не позволяет использовать априорные оценки.

Апостериорные методы нацелены на анализ конкретного приближенного решения. Однако большинство из них также не подходят для анализа результатов работы нейронных сетей. Это связано с тем, что апостериорные оценки обычно используют специфические особенности приближенных решений. Так хорошо известный метод невязок (explicit residual method) использует свойство Галеркинской ортогональности. Подробное изложение этого метода можно найти в монографии [10]. Индикаторы ошибок, применяемые для МКЭ и других

методов, также используют особые свойства точного решения и (или) конечномерных подпространств, в которых строится приближеноое решение (например свойство суперсходимости некоторых составляющих приближенных решений, которое имеет место для квазирегулярных сеток). Для DNN–решений это подходы неприменимы, поскольку последние не отвечают таким требованиям. Существует только один класс апостериорных оценок (апостериорные оценки функционального типа), который можно использовать в этом случае. Эти оценки позволяют оценить расстояние между точным решением краевой задачи и любой функцией из соответствующего класса допустимых функций с ограниченной энергетической нормой. В этом случае проблема анализируется на функциональном уровне без использования особенностей аппроксимаций и способов их получения.

В общей форме идея функционального подхода к проблеме апостериорного контроля точности состоит в следующем. Пусть $\mathcal{A} : V \to V'$ – дифференциальный оператор, областью определения которого является банахово пространство V, а областью значений – банахово пространство V'. Предположим что задача

$$\mathcal{A}u = f, \qquad f \in V', \tag{1}$$

порожденная этим оператором имеет единственное решение u. Предположим, что функция $v \in V$ является приближением u. Нам необходимо построить функционалы $M_\oplus : V \to \mathbb{R}_{\geqslant 0}$ и $M_\ominus : V \to \mathbb{R}_{\geqslant 0}$, которые зависят только от v и известных данных $\mathcal{D}$ (геометрии области, коэффициентов уравнения, краевых условий, дополнительных параметров и т.п.), обращаются в ноль только на точном решении задачи и, кроме того,

$$M_\ominus(v, \mathcal{D}) \leqslant \|u - v\|_V \leqslant M_\oplus(v, \mathcal{D}) \qquad \forall v \in V, \tag{2}$$

$$M_\oplus(v_k, \mathcal{D}), M_\ominus(v_k, \mathcal{D}) \to 0 \tag{3}$$

$$\text{для любой последовательности } v_k \to u \text{ в } V. \tag{4}$$

Величина $I_{eff} := M_\oplus(v)/M_\ominus(v)$ называется индексом эффективности и дает представление о качестве оценок и величине реальной погрешности.

Задача построения функционалов $M_\ominus$ и $M_\oplus$ для задач с выпуклых вариационных задач была поставлена и решена в [11]. В дальнейшем были разработаны два метода построения таких функционалов: первый основан на использовании теории двойственности вариационного

исчисления, а второй (невариационный) использует специальные преобразования интегральных тождеств, определяющих обобщенное решение задачи. Первый метод подробно изложен в [12], а второй в [13] и [9] (см. также [14–16]).

В качестве самого простого примера можно взять задачу $\mathrm{div} A\nabla u + f = 0$ с симметричной положительно определенной матрицей A. Задача рассматривается в ограниченной липшицевой области $\Omega \in \mathbb{R}^d$, $d \geqslant 1$ с краевым условием $u = 0$ на границе $\partial\Omega$. В этом случае V совпадает с Соболевским пространством $W_0^{1,2}(\Omega)$, а отклонение от точного решения естественно измерять в норме

$$\|\nabla(u-v)\|_A := \left(\int\limits_\Omega A\nabla(u-v)\cdot\nabla(u-v)dx \right)^{1/2}.$$

Существует несколько различных форм функционалов $M_\ominus$ и $M_\oplus$, которые контролируют эту норму ошибки. Наиболее простые из них задаются выражениями

$$M_\oplus^2(v,y,\gamma) = (1+\gamma)\int\limits_\Omega (A\nabla v\cdot\nabla v + A^{-1}y\cdot y - 2\nabla v\cdot y)dx + \frac{1+\gamma}{\gamma}C_\Omega^2\|\mathrm{div} y + f\|^2 \quad (5)$$

и

$$M_\ominus^2(v,w) = -\|\nabla w\|_A^2 - 2\int\limits_\Omega A\nabla v\cdot\nabla w\,dx + 2\int\limits_\Omega fw\,dx \quad (6)$$

Здесь

$$y \in H(\Omega,\mathrm{div}) := \{y \in L^2(\Omega,\mathbb{R}^d),\ |\ \mathrm{div} y \in L^2(\Omega)\},$$

$\gamma > 0$, $w \in V$, $\|\cdot\|$ обозначает норму в пространстве $L^2(\Omega)$, а постоянная C_Ω соответствует постоянной в неравенстве $\|v\| \leqslant C_\Omega\|\nabla v\|_A$, которое выполняется для для любой функции $v \in V$. Нетрудно видеть, что мажоранта и миноранта точны в том смысле что

$$\sup_{w\in W_0^{1,2}(\Omega)} M_\ominus^2(v,w) = \|\nabla(u-v)\|_A^2 = \inf_{\substack{y\in H(\Omega,\mathrm{div})\\ \gamma>0}} M_\oplus^2(v,y,\gamma), \quad (7)$$

причем точность двусторонних оценок зависит от выбора "свободных" функций y и w. На практике эти функции строятся на основе реконструкции (post–processing) приближенного решения так, чтобы они

приближали вектор функцию ∇u и функцию ошибки $e = u - v$ соответственно. Аналогичные оценки построены для многих уравнений в частных производных (например для уравнений конвекции–диффузии, теории упругости, теории вязких несжимаемых жидкостей, уравнений Максвелла и многих других). Их использование для контроля точности приближенных решений, полученных с помощью вариационно-разностных и других методов исследовалось в большом количестве работ (см. монографию [17] и цитированные там статьи). Важной особенностью оценок является то, что они обладают максимальной универсальностью и применимы для любой функции $v \in V$ независимо от того, как она была получена. Конечно само по себе условие принадлежности пространству V накладывает определенные ограничения, которые в ряде случаев нарушаются. Например при использовании так называемых "еконформных" аппроксимаций (например в рамках метода конечных разностей или метода конечных объемов) приближенное решение не принадлежит энергетическому пространству. Однако обычно это ограничение легко обходится с помощью подходящего оператора проектирования. Аналогичный подход может быть использован и для приближенных решений построенных с помощью DNN.

§3. Контроль точности решения, полученного с помощью DNN

Проблему оценки точности решения построенного DNN можно формализовать следующим образом. Сеть позволяет получить значения v_i в заданном множестве точек $X_n := \{x_i\} \in \Omega$, $i = 1, 2, ..., n$, т.е. фактически результатом работы является сеточная функция. Пусть $\mathfrak{T}_n$ обозначает пространство сеточных функций, принимающих вещественные значения в этих точках. Сеточную функцию со значениями v_i будем обозначать $\mathbf{v_n}$ и называть DNN–решением, а сеточную функцию, построенную по точным значениям $u_i = u(x_i)$ обозначим $\mathbf{u_n}$. На первый взгляд естественно взять в качестве критерия точности некоторую сеточную меру $|\cdot|_{\mathfrak{T}_n}$, например величину

$$| \mathbf{v_n} - \mathbf{u_n} |_{\mathfrak{T}_n} := \left(\sum_{i=1}^{n} |v_i - u(x_i)|^2 \right)^{1/2}. \tag{8}$$

Однако в общем случае такой критерий не является вполне корректным потому что решение уравнения в отдельной точке может быть не

определено. Для корректного сравнения нам надо использовать критерии интегрального типа, которые могут измерять точность решения в локальных или глобальных нормах. Ниже мы рассматриваем один из возможных подходов к измерению глобальной точности в терминах энергетической нормы. Рассмотрим конечномерное пространство $V_h \subset V$ и линейный оператор $\Pi_n : \mathbf{v_n} \to V_h$, который удовлетворяет условию $\Pi_n \mathbf{v_n}(x_i) = v_i$ непрерывен в том смысле, что из стремления $\widetilde{\mathbf{v}}_\mathbf{n}$ к $\mathbf{v_n}$ в сеточной норме пространства $|\cdot|_{\mathfrak{T}_n}$ следует, что $\|\Pi_n(\widetilde{\mathbf{v}}_\mathbf{n} - \mathbf{v_n})\|_V \to 0$. Оператор $\Pi_n : \mathbf{v_n} \to V_h$ назовем оператором восполнения сеточной функции $\mathbf{v_n}$.

Определение 1. *DNN–решение* $\mathbf{v_n}$ *является* ϵ*–точным, если существует восполнение* $\Pi_n \mathbf{v_n}$ *такое, что*

$$\|u - \Pi_n \mathbf{v_n}\|_V \leqslant \epsilon. \tag{9}$$

Методы построения операторов восполнения сеточных функций хорошо изучены, но вопрос о построении наилучшего оператора $\Pi_n \mathbf{v_n}$ не решается однозначно, поскольку такой оператор зависит от структуры сетки и типа функций (аффинные, полиномиальные, тригонометрические), которые используются для продолжения между узлами. Далее мы будем использовать простейшие кусочно–аффинные восполнения.

Используя мажоранту (5) и миноранту (6), получаем вычисляемые выражения для верней и нижней границ нормы (9) :

$$e_\oplus(\mathbf{v_n}) \leqslant \inf_{\substack{y_\tau \in Y_\tau \\ \beta > 0}} \left\{ (1+\beta)\|\nabla \Pi_n \mathbf{v_n} - y_\tau\|^2 + \frac{1+\beta}{\beta} C_\Omega^2 \|\mathrm{div} y_\tau + f\|^2 \right\}, \tag{10}$$

$$e_\ominus(\mathbf{v_n}) \geqslant \sup_{w_{h'} \in V_{h'}} \left\{ -\|\nabla w_{h'}\|^2 - 2 \int_\Omega (\nabla \Pi_n \mathbf{v_n} \cdot \nabla w_{h'} - f w_{h'})\, dx \right\}, \tag{11}$$

где использованы конечномерные подпространства $Y_\tau \subset H(\Omega, \mathrm{div})$ и $V_{h'} \subset V$. Базис в подпространстве Y_τ удобно строить при помощи конечных элементов Равьяра–Тома [18]. Они образуют кусочно–полиномиальное векторное поле, которое сохраняют непрерывность нормальной компоненты при переходе через любую грань. Соответствующие базисные функции ассоциированы с гранями элементов (в простейшем случае с гранью элемента связывается одна степень свободы). При вычислении оценки снизу (6) существенным является выбор подпространства $V_{h'}$ для функции w. Это подпространство должно быть

шире, чем подпространство V_h. В примерах для $V_{h'}$ мы используем кусочно–полиномиальные функции 2-го порядка.

§4. Использование нейронных сетей и машинного обучения для решения краевых задач

В общем виде идею deep learning для решения краевых задач можно формализовать следующим образом. Мы хотим получить приближенное решение задачи $\mathcal{A}u = f$ (т.е. функцию $u = \mathcal{A}^{-1}f$) построив упрощенную модель обратного оператора. Модель представляет собой граф с весами, которые определяются с помощью процедур "обучения". Такие модели объектов или процессов часто называют суррогатными (surrogate model). Суррогатные модели могут строится для решения прямых и обратных краевых задач. Для этого используется та или иная форма невязки $\mathcal{A}v - f$, с помощью которой формируется целевой функционал $J(v)$. В терминологии теории оптимального управления $J(v)$ представляет собой целевой функционал в задаче оптимизации весов на графе (нейронной сети). В терминологии, принятой в методах "обучения" сетей (deep learning), такой функционал называется "функцией потерь" (loss function).

Вычисление невязки не требует обращения оператора и поэтому сколь угодно длинная последовательность обучающих примеров и соответствующая ей функция $J(v)$ может быть построена без больших затрат. В результате оптимизации образуется сеть, которая воспроизводит решение задачи с той или иной точностью. Эффективность процесса обучения оценивается с помощью функции потерь и считается, что если последняя стагнирует вокруг некоторого малого значения, то процесс оптимизации можно прекратить, а сеть признать построенной наилучшим образом.

Изложенная выше схема является типичной для методов машинного обучения нейронных сетей вообще и сетей ориентированных на дифференциальные уравнения в частности. Успешность этой общей схемы зависит от правильного выбора функционала $J(v)$ и от того насколько эффективен выбранный метод оптимизации. В работах [19] и [4] предлагается использовать в качестве $J(v)$ сумму невязок уравнения, посчитанную в некотором наборе точек, т.е. фактически используется та же идея, что и в методе коллокаций. В [4] данный подход к

построению нейросетевых моделей предлагается для дифференциальных уравнений вида

$$G(x, u(x), \nabla u(x), \nabla^2 u(x)) = 0.$$

Фукционал $J(v)$ задается в форме

$$\sum_{x_i \in \Omega} G^2(x_i, v(x_i, \theta_m), \nabla v(x_i, \theta_m), \nabla^2 v(x_i, \theta_m)), \tag{12}$$

где θ_m определяет набор параметров нейросети на итерации m. Приближенное решение v предлагается разыскивать в виде

$$v = F(x, Net(x, \theta_m)) + \beta(x). \tag{13}$$

Функция $\beta(x)$ задает краевые условия, а функция F определяется выходными параметрами сети $Net(x, \theta_m)$ и обращается в ноль на границе области.

В [19] аналогичный подход используется для параболических уравнений

$$u_t + Lu(x,t) = 0, \quad (t,x) \in Q_T := [0,T] \times \Omega, \tag{14}$$

$$u(x,0) = u_0(x), \qquad x \in \Omega, \tag{15}$$

$$u(x,t) = g(x,t), \quad (t,x) \in S_T := [0,T] \times \partial\Omega, \tag{16}$$

При этом целевой функционал J сначала задается в виде взвешенной суммы интегральных норм, которые соответствуют (14)–(16). Однако затем он заменяется на функционал

$$\hat{J}(v, \theta_m, x_m) = (v_t(\theta_m, x_m, t_m) + Lv(\theta_m, x_m, t_m))^2 + (v(\theta_m, z_m, \tau_m) - g(z_m, t_m))^2 + (v(\theta_m, \zeta_m, 0) - u_0(\zeta_m))^2, \tag{17}$$

где x_m, t_m – набор точек в Q_T, которые выбираются случайным образом. Аналогично (z_m, τ_m) и выбираются случайным образом в S_T, а ζ_m – в Ω. Таким образом, целевой функционал задается невязкой уравнения в некотором случайном наборе точек. Если параметры θ_m фиксированы, но сеть генерирует на выходе некоторую функцию, производные которой считается алгоритмами стандартными для для глубоких нейронных сетей с использованием алгоритмов символьных вычислений.

С точки зрения теории коллокационного метода такой подход вызывает сомнения поскольку точки коллокации не могут выбираться

произвольным (случайным) образом (см., например, [20], где показано, что выбор точек коллокации связан с формой носителей базисных функций, которые используются для аппроксимации решения). Его недостатки иллюстрирует следующий простой пример. Рассмотрим задачу $\frac{d^2u}{dx^2} = 0$ с условиями $u(0) = u(1) = 0$. Допустим, что при каком-то m (например при $m = 1$) сеть с параметрам θ_m генерирует функцию $v(x) = 0.5 - |x - 0.5|$ или очень близкую гладкую функцию, которая отличается от v только в малой окрестности точки $x = 0.5$. В этом случае величины $\frac{d^2v}{dx^2}(x_i)$ в произвольно выбранных точках x_i могут быть очень малы (или просто равны нулю) и функционал цели будет показывать, что решение практически получено. Однако ясно, что такая функция v никакого отношения к точному решению $u = 0$ не имеет. Ниже мы приводим примеры, показывающие что подобные ситуации "блокировки" алгоритма оптимизации могут происходить и для уравнений с частными производными.

В пакете `pydens` [3], подход [19] используется в приложении к эллиптическим уравнениям. Приведенные ниже примеры считались с помощью этого пакета для краевой задачи

$$\Delta u + f = 0 \ \text{ в}\,\Omega, \quad u = u_0 \ \text{на}\ \partial\Omega \tag{18}$$

Для того чтобы точно удовлетворить граничные условия используется простой вариант представления (13)

$$v = \alpha(x)Net(x, \theta) + \beta(x), \quad \alpha|_{\partial\Omega} = 0, \quad \beta|_{\partial\Omega} = u_0,$$

где $Net(x, \theta)$ – значение, полученное в точке x при помощи нейронной сети с весовыми коэффициентами θ (вид такой сети с 4 слоями из 50 нейронов представлен на рис. 2). У сети два входа x_1, x_2 и один выход: решение $v(x_1, x_2)$. Также в пакете `pydens` могут использоваться и более сложные сети, в том числе и те, что содержат т.н. residual block, который пропускает один или несколько скрытых слоев и его выход добавляется к выходу полносвязного слоя. Считается, что в таком случае нейронную сеть легче оптимизировать.

В качестве α используют так называемую bubble-функцию (гладкую функцию, обращающуюся в ноль на границе области Ω). Например для области $\Omega = [0, 1]^2$ можно положить (см., [4]) $\alpha(x) = x_1(1 - x_1)x_2(1 - x_2)$. Функция β является непрерывным продолжением краевых условий внутрь области. Задача нахождения весовых

коэффициентов сети θ состоит в оптимизации

$$\hat{J}(v) = \frac{1}{N}\sum_{i=1}^{N}(\Delta v + f)^2|_{x_i} \tag{19}$$

где N – число точек некоторым образом (случайным) выбранным в области Ω. На рис. 1 представлен пример 100 таких точек. В рамках этого подхода Δv считается алгоритмами стандартными для машинного обучения для глубоких нейронных сетей с использованием алгоритмов символьных вычислений пакета TensorFlow, которые позволяют автоматическое вычисление производных функции.

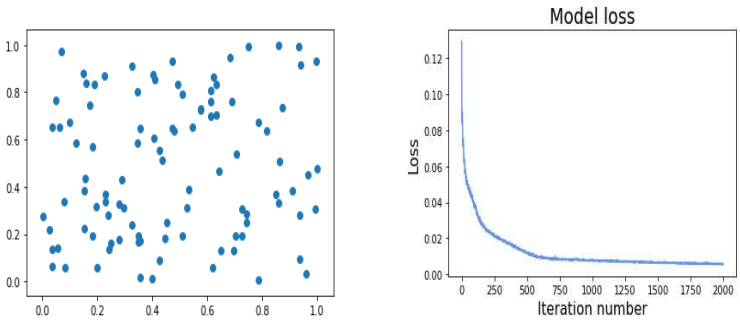

Рис. 1. Пример работы программы sampler (слева) и типичный вид функция потерь в процессе обучения нейронной сети (справа).

Функция $\hat{J}(v)$ можно интерпретировать как некоторую усредненную невязку дифференциального уравнения (18), представленного в классической форме. Заметим, что аналогичные формы невязки используется и в некоторых численных методах (например в известном методе коллокаций). Считается, что если сеть построена правильно, то на любом новом наборе точек функционал $\hat{J}(v)$ мал. На каждом шаге “обучения” (оптимизации) эти точки выбираются заново (с помощью некоторой процедуры – Sampler). В конце “обучения” функционал $\hat{J}(v)$ перестает убывать (см. рис. 1). Такая стагнация функции потерь вполне естественна, поскольку нейронная сеть имеет конечное число параметров и конечно может приближать решение с ограниченной точностью. В конце процесса должна образоваться сеть, которая

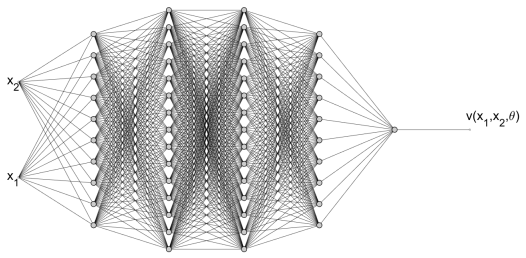

Рис. 2. Нейронная сеть, применяемая в примерах (606 весовых коэффициентов θ).

в любой точке области определяет решение с некоторой точностью. Фактически речь идет о построении сетевой модели формулы обращения

$$u(x) = \int\limits_{\Omega} G(x,\zeta) f(\zeta) d\zeta \tag{20}$$

для уравнения Пуассона с соответствующей функцией Грина G.

На качество DNN-решения влияет выбор количества точек N и алгоритм который строит эти точки (Sampler), а также количество итераций метода оптимизации весов сети (их называют “эпохами”) и структура нейронной сети (количество нейронов и связи между ними).

В данной статье мы не обсуждаем могут ли DNN– решения конкурировать с теми, что получаются при помощи хорошо известных численных методов (по мнению авторов, по точности и экономичности стандартные вычислительные методы пока намного эффективнее). Нас интересует иной вопрос, принципиально важный с точки зрения самого процесса оптимизации (обучения) нейронных сетей ориентированных на решение дифференциальных уравнений, а именно “Какую величину можно считать объективной характеристикой качества построенной сети?” В частности является ли таковой функция (19), которая аналогична(12) и (17)? Этот вопрос тесно связан с другим:

“Всегда ли процесс обучения приводит к сетевой модели, которая дает приемлемое решение задачи?”

Заметим, что вопрос о выборе эффективного целевого функционала для процесса оптимизации возникают практически во всех случаях, когда методы машинного обучения используются для построения суррогатных моделей (например при распознавании изображений). Однако на них не всегда можно дать однозначные ответы ввиду множественности критериев качества решений. В рассматриваемом нами случае в качестве объекта используется дифференциальное уравнение, для которого можно применить методы апостериорного контроля, что позволяет объективно оценить полученные DNN–решения и сделать обоснованные выводы.

§5. Примеры

Приведенные ниже примеры можно разделить на три группы. Первая группа содержит те случаи, когда DNN–решение $\mathbf{v_n}$ можно признать достаточно хорошим. Методы объективного контроля, основанные на использовании оценок (10) и (11) подтверждают этот факт. Решения задач из второй группы нельзя признать столь удачными. Оценки показывают, что приближения оказались весьма грубыми и указывают места, где соответствующие ошибки максимальны. Третья группа содержит $\mathbf{v_n}$, которые можно считать неверными.

5.1. Прямоугольная область. Пример 1. Рассмотрим вначале самую простую задачу, где $\Omega = (0,1)^2$, $u_0 = 0$, а правая часть задается гладкой функцией $f = -2\pi^2 \sin(\pi x_1)\sin(\pi x_2)$. Процесс построения сетевой модели представлен графиком функции потерь (Рис. 3).

В данном примере множество X_n задает равномерную сетку с шагом h, а оператор восполнения Π_n осуществляет кусочно–аффинное продолжение значений функции в узлах так, что $\Pi_n \mathbf{v_n}$ является непрерывной функцией и принадлежит пространству $V_h \in H^1(\Omega)$, (см. рис. 4). Для равномерной сетки в квадрате $h = 1/n$, общее число узлов $N_p = (n+1)^2$, а число треугольных элементов $N_t = 2n^2$.

Используя (5), получаем верхнюю границу погрешности и распределение ошибки по области. Для вычисления соответствующих интегралов по симплексам применялся метод Гаусса 6 порядка, который позволяет производить интегрирование практически точно.

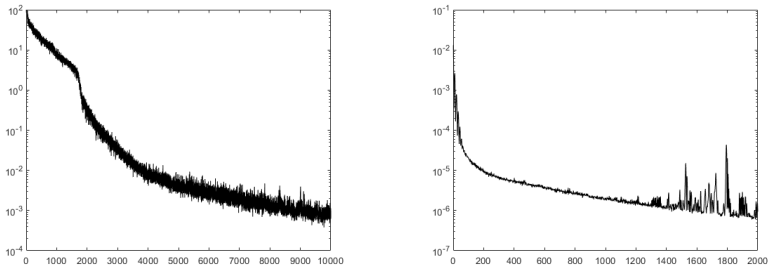

Рис. 3. Функция потерь. Примеры 1 (слева) и 2 (справа).

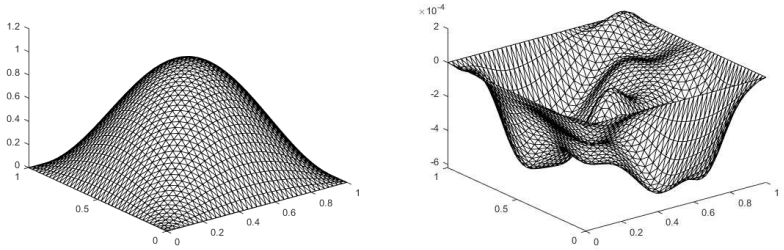

Рис. 4. Пример 1. Функции $\Pi_n \mathbf{v_n}$ и $u - \Pi_n \mathbf{v_n}$ на равномерной сетке с $h = 1/40$, и $N_p = 1681$.

В этом примере DNN–решение является достаточно точным, причем уменьшение h приводит к пропорциональному уменьшению погрешности (см. Таб. 1). Отклонение DNN–решения от точного показано на рис. 4 справа (оно достаточно мало). Мажоранта и миноранта дают весьма точные границы погрешности, причем анализ первого слагаемого мажоранты позволяет выявить зоны с максимальной величиной локальных ошибок. Они показаны на рис. 5), где для маркировки распределения ошибки используется несколько градаций. Темно-серый цвет показывает элементы, которые в сумме дают 50% суммарной величины ошибки. Для тех элементов, которые в сумме дают 70% ошибки используется серый цвет, а для тех элементов что в сумме дают 90% – светло-серый. Белым цветом отмечены оставшиеся элементы.

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	3.92	1.96	0.98
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$, %	3.92	1.96	0.98
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	7.40	3.70	1.85
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	1.34		
$\hat{J}(v)$	0.00089		

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	79.05	78.97	78.95
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$,%	79.07	78.98	78.96
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	90.75	89.96	89.74
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	0.26		
$\hat{J}(v)$	7.3e-07		

Таблица 1. Ошибка в зависимости от шага сетки h. Примеры 1 (слева) и 2 (справа).

Распределение изображенное слева (построенное при помощи мажоранты) мало отличается от изображения справа, которое использует точное решение.

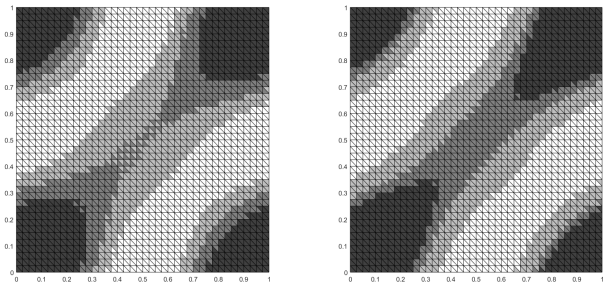

Рис. 5. Пример 1. Распределение ошибки по области, построенное по мажоранте (слева) и по точной ошибке(справа).

Пример 2. Возьмем ту же самую область Ω и $f = 0$, но краевые условия выберем так, чтобы исключить гладкое решение: $u_0 = |x_1 - 0.5|\,|x_2 - 0.5|$. В этом случае результат оказывается совершенно иным.

Процесс "обучения" производился с $N = 1000$. Нахождение DNN-решения представлен графиком функции потерь в логарифмическом масштабе (Рис. 3, справа). Отметим, что значения функционала $\hat{J}$ не показывают, что возникли какие-то проблемы и финальная функция потерь очень близка к нулю.

На самом деле выбор таких граничных условий приводит к появлению DNN-решения, которое можно назвать неверным. Это решение мало отличается от u_0 и имеет внутри области "ребра", хотя внутри области точное решение должно быть гладким (в силу известных свойств локальной регулярности). Заметим, что для таких граничных условий точное решение неизвестно. Поэтому для того, чтобы оценить погрешность полученного решения, мы используем два независимых метода. В первом вместо точного решения используется так называемое reference solution, вычисленное при помощи МКЭ на очень тонкой сетке с с 299313 узлами. Погрешность такого решения намного меньше той, что можно ожидать для $\Pi_n \mathbf{v_n}$, так что его можно считать практически точным.

На рис. 6 видно, DNN-решение существенно отличается от точного и количественно и качественно. При уменьшении h никаких существенных изменений в DNN-решении не происходит (Таб. 1).

Было исследовано, каким получается это DNN-решение в зависимости от различных начальных параметрах и от N. Заметим, что при каждом запуске пакета начальные параметры сети инициируются случайным образом. Результаты тестов после 2000 итераций приведены в таблице 2.

Тест №	N	$\hat{J}(v)$	$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$,%
1	1000	9.7e-07	0.30	78.97
2	1000	1.0e-06	0.32	78.97
3	1000	3.2e-07	0.17	78.97
4	5000	2.5e-06	0.49	78.96
5	5000	5.2e-07	0.22	78.97
6	5000	2.4e-06	0.49	79.05
7	10000	1.2e-07	0.11	78.97
8	10000	3.2e-06	0.55	78.96
9	10000	2.2e-06	0.46	78.96

Таблица 2. Пример 2. Финальная функция потерь и точная ошибка для различных N.

Этот эффект вероятно связан с выбором функционала $\hat{J}$ и способом вычисления его значения в отдельных точках. Если функция v в окрестности x_i близка к гармонической, то формальное применение

символьных вычислений приводит к очень малым значениям Δv в x_i. Однако кусочно–гармоническая непрерывная функция может не быть решением даже если она точно удовлетворяет краевым условиям. Таб. 1 показывает, что использование функции потерь не позволяет обнаружить эту проблему, а функционалы $M_\ominus$ и $M_\oplus$ точно указывают на истинную величину погрешности.

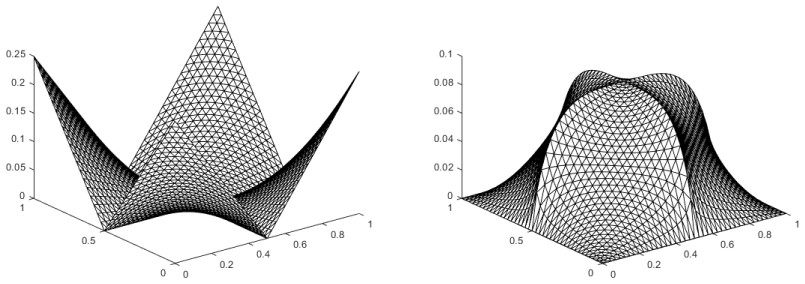

Рис. 6. Пример 2. Функции $\Pi_n \mathbf{v_n}$ и $u - \Pi_n \mathbf{v_n}$ на равномерной сетке с $h = 1/40$ и $N_p = 1681$.

Пример 3. Этот пример показывает, что если в процессе обучения используется функция (19), то возможно возникновение проблем иного рода. Рассмотрим задачу в прямоугольной области Ω с $f = 0$ и функцией

$$u_0 = \sqrt{\varepsilon^2 + (x_1 - 0.5)^2}\sqrt{\varepsilon^2 + (x_2 - 0.5)^2}$$

для $\varepsilon = 0.01$. Эта функция является гладкой, а ее значения мало отличаются от значений функции, использовавшейся в примере 2. Для анализа результатов вместо точного решения использовалось reference solution, вычисленное при помощи МКЭ на очень тонкой сетке с с 299313 узлами. Погрешность этого решения очень мала (намного меньше, чем погрешность DNN-решения) и в рамках нашего анализа его можно рассматривать как точное.

Процесс "обучения" сети, производился с $N = 1000$. Это незначительное изменение краевого условия привело к сильному изменению результата в том смысле, что теперь функция потерь не стремится к нулю (Рис. 7). Таким образом, как и в предидущем примере, построить приемлемую аппроксимацию решения не удалось. Однако здесь функция потерь по крайней мере показывает, что результат не достигнут.

При этом ни абсолютное, ни нормализованное значение функционала $\hat{J}$ не дает представления о качестве полученного решения. Вероятно такое поведение функционала связано с попаданием одной или нескольких точек x_i в те (небольшие по площади) области, где необходимо аппроксимировать функцию с большими значениями вторых производных.

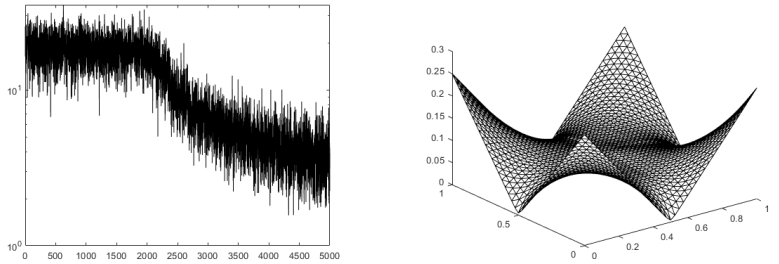

Рис. 7. Пример 3. Функция потерь (слева) и функция $\Pi_n \mathbf{v_n}$ (справа).

Как видно из таблицы (3), мажоранта и миноранта правильно оценивают интегральную точность решения, которое было построено сетью. Также анализ первого слагаемого мажоранты позволяет весьма точно установить характер распределения ошибок по области Ω. На рис. (8) сравниваются два распределения локальных ошибок, одно из которых вычислено при помощи $M_{\oplus}$, а второе получено путем непосредственного вычисления точной ошибки $\nabla(u - \Pi_n \mathbf{v_n})$.

Пример 4. В этом примере рассматривается случай негладкой правой части

$$f = \begin{cases} 0 & \text{если} \quad x_1 < 0.5 \\ 1 & \text{если} \quad x_1 \geqslant 0.5 \end{cases}$$

В качестве точного используется reference solution, построенное на сетке из 149144 элементов с 299313 узлами. Обучении сети производилось при $N = 5000$.

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	12.03	9.79	9.13
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$,%	12.17	9.82	9.14
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	20.60	14.86	12.96
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	628.50		
$\hat{J}(v)$	4.01		

Таблица 3. Пример 3. Ошибка в зависимости от шага сетки h.

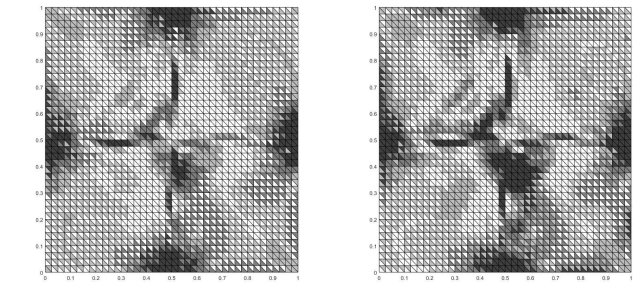

Рис. 8. Пример 3. Распределение ошибки по области, построенное по мажоранте (слева) и по точной ошибке(справа).

Результаты представлены в Таб. 4. Как и первом примере, здесь точное решение является достаточно регулярной функцией и соответствующее DNN–решение можно признать вполне удачным. При измельчении сетки вплоть до значений $h = 1/160$ соответствующие DNN–решения демонстрируют почти пропорциональное уменьшение ошибки. Однако величина функции $\hat{J}(v)$ не дает представления о величине погрешности.

5.2. L–образная область. Задачи в L–образных областях ("L-shape domain") часто используются для тестирования методов численного

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	5.46	3.07	2.13
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$, %	5.52	3.18	2.24
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	9.41	5.14	3.30
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	75.32		
$\hat{J}(v)$	0.0070		

Таблица 4. Пример 4. Ошибка в зависимости от шага сетки h.

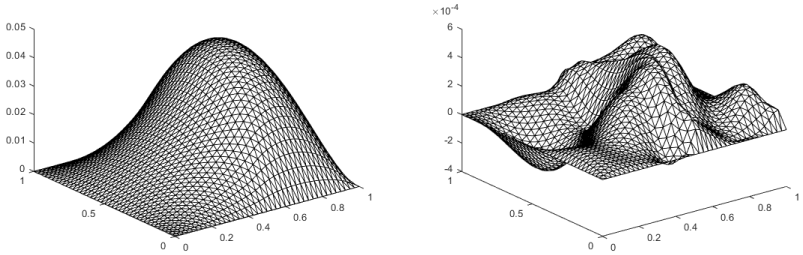

Рис. 9. Пример 4. Функции $\Pi_n \mathbf{v_n}$ и $u - \Pi_n \mathbf{v_n}$ на равномерной сетке с $h = 1/40$, $N_p = 1681$.

решения дифференциальных уравнений в частных производных потому, что во внутреннем угле этой невыпуклой области возможно возникновение различных особенностей. Приведем два примера, в которых $\Omega = (0,1)^2 \backslash [1/2,1]^2$.

Пример 5. В этом примере

$$f = (6 - 12x_1)x_2(1 - x_2)(2x_2 - 1) + (6 - 12x_2)x_1(1 - x_1)(2x_1 - 1),$$

а точное решение задается полиномом $u = x_1(1 - x_1)(2x_1 - 1)x_2(1 - x_2)(2x_2 - 1)$.

Процесс "обучения" производился с $N = 1000$ и представлен графиком функции потерь в логарифмическом масштабе (Рис. 10). Левая часть таблицы 5 характеризует качество построенной сетевой модели. Соответствующие DNN–решения можно считать вполне успешными.

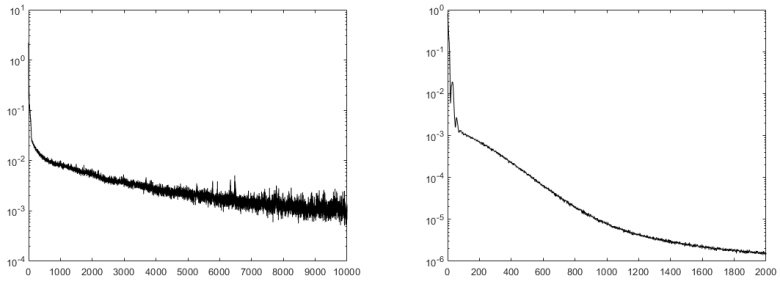

Рис. 10. Функция потерь. Пример 5 (слева) и пример 6 (справа).

При измельчении сетки точность функции $v = \Pi_n \mathbf{v_n}$ увеличивается и достигает 3%. Это следует из вычисленных значений мажоранты и миноранты и подтверждается сравнением с точным решением). При этом величина $\hat{J}(v)$ никакой надежной информации не дает.

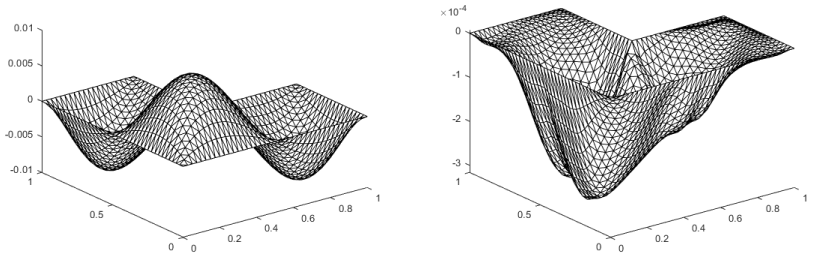

Рис. 11. Пример 5. Функции $\Pi_n \mathbf{v_n}$ и $u - \Pi_n \mathbf{v_n}$ на равномерной сетке с $h = 1/40$, $N_p = 1281$.

На рис. (12) сравниваются два распределения локальных ошибок, одно из которых вычислено при помощи $M_\oplus$, а второе получено путем непосредственного вычисления точной ошибки $\nabla(u - \Pi_n \mathbf{v_n})$.

Пример 6. Здесь область Ω такая же, как и в Примере 4, $f = 0$, а краевые условия задаются функцией

$$u_0 = \max(0, x_1 - 0.25).$$

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	8.91	4.84	3.06
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$, %	8.92	4.84	3.06
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	18.42	9.50	5.27
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	105.40		
$\hat{J}(v)$	0.0015		

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	48.56	48.56	48.57
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$,%	48.57	48.57	48.57
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	60.19	60.05	60.01
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	0.20		
$\hat{J}(v)$	1.7e-06		

Таблица 5. Ошибка в зависимости от шага сетки h. Пример 5 (слева) и пример 6 (справа).

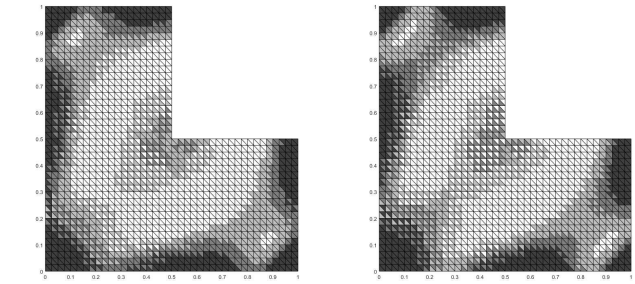

Рис. 12. Пример 5. Распределение ошибки по области, построенное по мажоранте (слева) и по точной ошибке(справа).

Как и в Примере 2 выбор таких граничных условий приводит к появлению DNN-решения, существенно отличного от точного, хотя финальная функция потерь близка к нулю (Рис. 10). На рис. 13 видно, что разность DNN-решения и референсного (которое было вычислено на сетке с 225765 узлами) существенно отличается от нуля. Процесс обучения сети производился с $N = 1000$. В результате была получена сеть, эффективность которой отражена в Таб. 5. Несмотря на то, что функция потерь мала, решение, которое строит сеть (оно изображено на рис. 13) нельзя признать разумной аппроксимацией точного решения.

Было исследовано, каким получается это DNN-решение в зависимости от различных начальных параметрах и от N. Заметим, что при

каждом запуске пакета начальные параметры сети инициируются случайным образом. Результаты тестов после 5000 итераций приведены в таблице 6.

Тест №	N	$\hat{J}(v)$	$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|},\%$	$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|},\%$
1	1000	9.4e-08	0.048	48.56
2	1000	4.5e-06	0.33	48.56
3	1000	9.6e-07	0.15	48.57
4	5000	3.6e-07	0.094	48.56
5	5000	3.5e-06	0.29	48.56
6	5000	2.2e-06	0.23	48.55
7	10000	9.3e-08	0.048	48.56
8	10000	3.8e-06	0.30	48.56
9	10000	3.1e-06	0.27	48.56

Таблица 6. Пример 6. Финальная функция потерь и точная ошибка.

Функционалы $M_{\oplus}$ и $M_{\ominus}$ фиксируют это и дают правильные оценки погрешности.

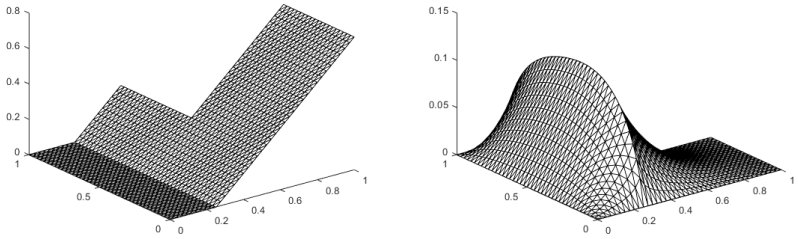

Рис. 13. Пример 6. Функции $\Pi_n \mathbf{v_n}$ и $u - \Pi_n \mathbf{v_n}$ на равномерной сетке с $h = 1/40$, $N_p = 1281$.

Более того, использование методов апостериорного контроля позволяет не только оценить глобальную норму погрешности, но и достаточно точно указать подобласти, где ошибки наиболее значительны (см. Рис. 14).

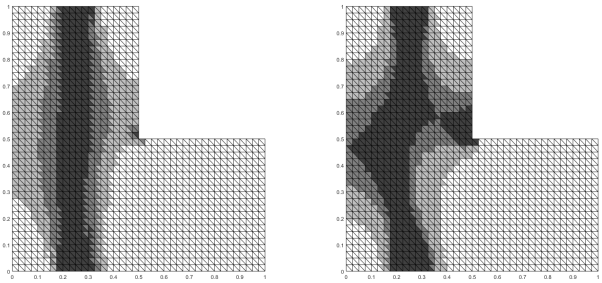

Рис. 14. Пример 6. Распределение ошибки по области, построенное по мажоранте (слева) и по точной ошибке(справа).

5.3. П–образная область. Также мы рассмотрим несколько примеров в области $\Omega = (-1,1) \times (0,1) \backslash ([-0.5, 0.5] \times [0, 0.5])$.

Пример 7. Рассматривается задача с однородными краевыми условиями,
$f = -(12x_1^2 - 2.5)(x_2^3 - 1.5x_2^2 + 0.5x_2) - (x_1^4 - 1.25x_1^2 + 0.25)(6x_2 - 3)$, а точное решение $u = (x_1 - 1)(x_1 + 1)(x_1 - 0.5)(x_1 + 0.5)x_2(x_2 - 1)(x_2 - 0.5)$. Построение сетевой модели производилось при $N = 5000$ и представлено графиком функции потерь в логарифмическом масштабе (Рис. 15).

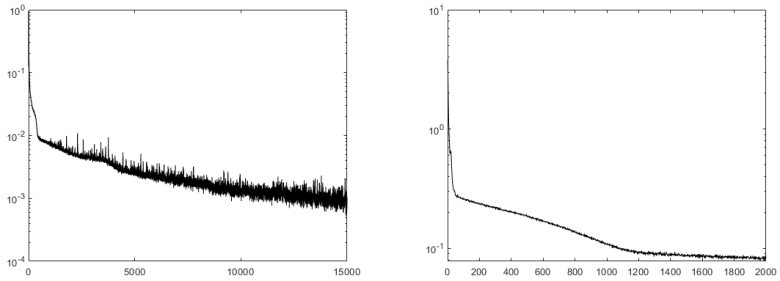

Рис. 15. Функция потерь. Функция потерь. Пример 7 (слева) и пример 8 (справа).

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	8.15	4.69	3.30
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$, %	8.15	4.69	3.30
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	17.78	9.41	5.62
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	74.19		
$\hat{J}(v)$	0.0010		

h	1/40	1/80	1/160
$\frac{\sqrt{M_{\ominus}(\hat{v})}}{\|\nabla u\|}$,%	31.77	31.76	31.76
$\frac{\|\nabla(u-\hat{v})\|}{\|\nabla u\|}$,%	31.78	31.77	31.76
$\frac{\sqrt{M_{\oplus}(\hat{v})}}{\|\nabla u\|}$,%	42.37	42.11	42.03
$\frac{\sqrt{\hat{J}(v)}}{\|\nabla u\|}$,%	24.20		
$\hat{J}(v)$	0.081		

Таблица 7. Ошибка в зависимости от шага сетки h. Пример 7 (слева) и пример 8 (справа).

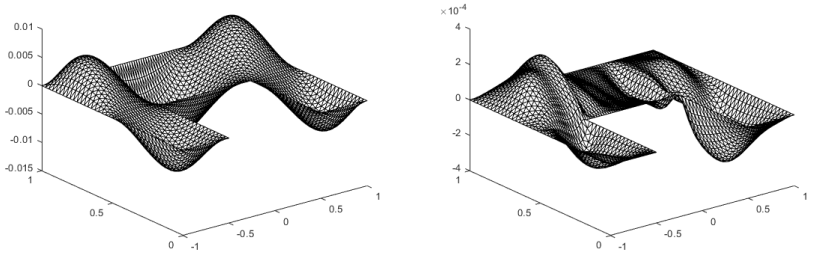

Рис. 16. Пример 7. Функции $\Pi_n \mathbf{v}_\mathbf{n}$ и $u - \Pi_n \mathbf{v}_\mathbf{n}$ на равномерной сетке с$h = 1/40$, $N_p = 2541$.

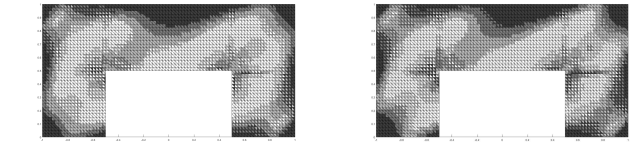

Рис. 17. Пример 7. Распределение ошибки по области, построенное по мажоранте (слева) и по точной ошибке(справа).

Пример 8. Здесь рассматривалась задача $\Delta u + 1 = 0$ с краевым условием $u_0 = |x_1|$. Как и в предыдущих примерах выбор таких граничных условий приводит к появлению DNN-решения, существенно

отличного от точного, хотя финальная функция потерь близка к нулю. На рис. 18 видно, что разность DNN-решения и референсного существенно отличается от нуля. В качестве точного используется референсное решение с 451581 узлами. Здесь $N = 10000$. Нахождение DNN-решения представлен графиком функции потерь в логарифмическом масштабе (Рис. 15).

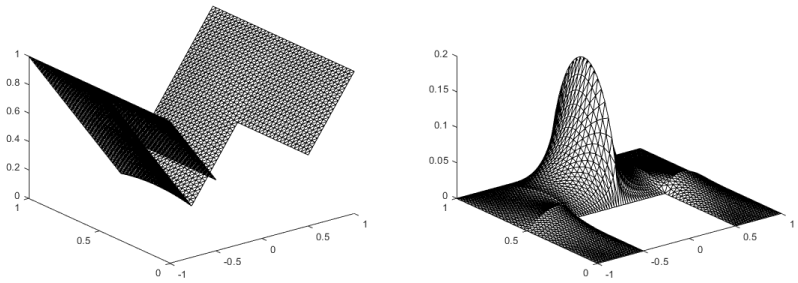

Рис. 18. Пример 8. Интерполянт DNN-решения. Равномерная сетка $h = 1/40$, $N_p = 2541$ и разность референсного решения и DNN-решения.

5.4. Выводы. Анализ примеров позволяет сделать следующие выводы. Прежде всего надо отметить, что апостериорные оценки функционального типа позволяют уверенно контролировать точность DNN–решений. Они успешно работают во всех возможных ситуациях и позволяют уверенно контролировать достоверность решений и отсеивать те из них, которые содержат большие погрешности. Важно, что оценки дают как гарантированные оценки глобальной погрешности так и качественные характеристики решения (показывают места сосредоточения ошибки).

Другой вывод заключается в том, что функция потерь (19) не является надежной харктеристикой качества решения. Выбор функционала цели $\hat{J}(v)$ в виде невязки уравнения, представленного в классической форме и вычисленного на конечном и случайно выбираемом наборе точек, не даёт надежной информации ни о глобальной величине погрешности ни о ее распределении по области. Это может приводить к сильному искажению решения и неадекватной оценке результатов

обучения сети. Они могут возникать в тех случаях, когда используются функции потерь коллокационного типа.

По мнению авторов более правильным было бы использование функционалов интегрального типа. В частности, сама мажоранта $M_{\oplus}$ может быть использована в этом качестве. Например для задачи (18) можно использовать функционал

$$\hat{J}(v,q) = \|\nabla v - q\|^2 + C\|\mathrm{div} q + f\|^2, \tag{21}$$

где q обозначает векторное поле, а C положительная постоянная, значение которой должно быть близким к постоянной C_Ω. Соответствующая сетевая модель будет иметь три выходных параметра: $v(x_1, x_2)$, $q_1(x_1, x_2)$ и $q_2(x_1, x_2)$. Такое представление оперирует только с первыми производными и отражает физически естественное представление уравнения в терминах функции и ее потока.

Список литературы

1. W. E, J. Han, A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations.* — Commun. Math. Stat. **5** (2017), 349–380.
2. J. He, L. Li, J. Xu, C. Zheng, *ReLU deep neural networks and linear finite elements.* `arXiv:1807.03973v2`.
3. R. Khudorozhkov, S. Tsimfer, A. Koryagin, *PyDEns framework for solving differential equations with deep learning.* — `arXiv:1909.11544 [cs.LG], arXiv`, 2019.
4. I. E. Lagaris, A. Likas, D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations.* — IEEE Transactions on Neural Networks **9**, No. 5, 987–1000.
5. W. E, B. Yu, *The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems.* — Commun. Math. Stat. **6** (2018), 1–12.
6. O. Pironneau, *Parameter identification of a fluid-structure system by deep-learning with an Eulerian formulation.* — Methods Appl. Anal. **26**, No. 3 (2019), 281–290.
7. F. Regazzonia, L.Dede, A.Quarteroni, *Machine learning for fast and reliable solution of time-dependent differential equations.* — J. Comput. Phys., **397** (2019), 108852.
8. E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuka, *An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications.* — Comput. Methods Appl. Mech. Engrg., **362** (2020), 112790.

9. S. Repin, *A posteriori estimates for partial differential equations.* Berlin: Walter de Gruyter GmbH & Co. KG, 2008.
10. R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Stuttgart: Wiley-Teubner, 1996.
11. S. Repin, *A posteriori error estimation for nonlinear variational problems by duality theory.* — Zap. Nauchn. Semin., V POMI **243** (1997), 201–214.
12. S. Repin, *A posteriori error estimation for variational problems with uniformly convex functionals.* — Math. Comp. **69** (2000), 481–500.
13. С. И. Репин, *Двусторонние оценки отклонения от точного решения для равномерно эллиптических уравнений.* — Труды С.-Петербургского математического общества **9** (2001), 148–179.
14. S. Repin, *Estimates of deviation from exact solutions of initial-boundary value problems for the heat equation.* — Rend.Mat. Acc. Lincei.**13** (2002), 121–133.
15. S. Repin, S. Sauter, A. Smolianski, *Two-sided a posteriori error estimates for mixed formulations of elliptic problems.* — SIAM J. Num. Analysis. **45** (2007), 928–945.
16. С. И. Репин, М. Е. Фролов, *Апостериорные оценки погрешности приближенных решений краевых задач эллиптического типа.* — ЖВМиМФ. **42**, No. 12 (2002), 1704–1716.
17. O. Mali, P. Nettaanmäki, S. Repin, *Accuracy verification methods. Theory and Algorithms.* Berlin: Springer, 2014.
18. P. A. Raviart, J. M. Thomas, *A mixed finite element method for 2-nd order elliptic problems.* — In: Galligani I., Magenes E. (eds) Mathematical Aspects of Finite Element Methods. Lecture Notes in Mathematics, vol 606. Springer, Berlin, Heidelberg, 1977.
19. J. Sirignano, K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations.* — J. Comput. Phys. **375** (2018), 1339–1364.
20. H. Gomez, L. Lorenzis, *The variational collocation method.* Computer Methods in Applied Mechanics and Engineering **309** (2016), 152–181.

Muzalevsky A. V., Repin S. I. A posteriori error control of approximate solutions to boundary value problems constructed by neural networks.

The paper discusses how to verify the quality of approximate solutions to partial differential equations constructed by deep neural networks. A posterior error estimates of the functional type, that have been developed for a wide range of boundary value problems, are used to solve this problem. It is shown, that they allow one to construct guaranteed two-sided estimates of global errors and get distribution of local errors the error over the domain. The corresponding results of numerical experiments are presented for a boundary value problem of an elliptic type. They show that the estimates provide much more reliable information than the so-called loss

function, which is commonly used as a quality criterion training neural network models.

С.-Петербургский
политехнический университет Петра Великого,
195251, Санкт-Петербург, Россия

С.-Петербургское отделение
Математического института
им. В. А. Стеклова РАН,
191023, Санкт-Петербург, Россия

Поступило 14 декабря 2020 г.