

Министерство науки и высшего образования Российской Федерации
ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»
Кафедра «школы бакалавриата (школа)»

Оценка работы 100
Руководитель от УрФУ Кошелев А.А.

Тема задания на практику

Возможность использования искусственных нейронных сетей для решения
задач математической физики

ОТЧЕТ

Вид практики Производственная практика
Тип практики Производственная практика, Преддипломная

Руководитель практики от предприятия (организации) Кошелев А.А. Кош
ФИО руководителя Подпись

Студент Кузнецов И.А.
ФИО студента

Специальность (направление подготовки) 01.03.01 Математика

Группа МЕН-490102

Екатеринбург 2023

РЕФЕРАТ

Кузнецов Игорь александрович «Возможность использования искусственных нейронных сетей для решения задач математической физики»: работа содержит: страниц 22, иллюстраций 7, таблиц 1, использованных источников 12

Ключевые слова: PINN, дифференциальные уравнения, нейронные сети, Python

PINN (Physics-Informed Neural Networks) - это метод, который сочетает в себе преимущества нейронных сетей и физических моделей для решения задач научного моделирования. В данном дипломном проекте исследуется применение метода PINN для решения задачи установившегося распределения тепла в кольце. В работе проводится анализ эффективности точности и скорости метода PINN. Также исследуется влияние различных параметров на точность решения задачи и высказаны идеи по улучшению полученных результатов. Результаты исследования показывают, что метод PINN может решить поставленную задачу. Для решения задачи были использованы следующие технологии:

- 1) язык программирования Python
- 2) библиотека для работы с глубокими нейронными сетями Tensorflow

Содержание

РЕФЕРАТ	2
ВВЕДЕНИЕ	4
ОСНОВНАЯ ЧАСТЬ	5
1 Постановка задачи	5
1.1 Цель работы	5
1.2 Описание задачи	5
2 Обзор литературы	5
3 Способы и методы решения задачи	6
3.1 Нейронные сети	6
3.2 Автоматическое дифференцирование	8
3.3 Теоретическое описание PINN	8
3.4 Техническая реализация	10
3.4.1 Используемые технологии	10
3.4.2 Программная реализация PINN	10
4 Результаты и их обсуждение	11
4.1 Установившееся распределение тепла в кольце	11
4.2 Задача электрокинетики	18
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ	21

ВВЕДЕНИЕ

В последние годы нейронные сети получили широкое распространение, они используются для анализа и генерации изображений и видео, обработки естественных языков (перевод, чат-боты), медицинской диагностики, финансовых прогнозах и так далее.

Одним из перспективных направлений в этой области являются так называемые PINN – Physics-Informed Neural Networks, физически-информированные нейронные сети. Классические нейронные сети используют большую выборку реальных данных, однако в естественно-научных областях, таких как физика, химия, биология и т.д. зачастую может просто не хватать нужного объёма данных для обучения. PINN способны обойти это ограничение, используя в обучении знания законов физики, описываемые дифференциальными уравнениями в частных производных. Это позволяет использовать неполные и зашумленные данные, что делает их полезными в реальных научных задачах. Однако, вычислительная сложность PINN выше, чем у классических нейронных сетей, что требует большого количества вычислительных мощностей.

ОСНОВНАЯ ЧАСТЬ

1 Постановка задачи

1.1 Цель работы

Рассмотреть задачу распределения тепла в диске, решить её с помощью PINN и сравнить полученные данные с аналитическим решением, оценить целесообразность применения PINN к задаче.

1.2 Описание задачи

Создать нейросеть, которой на вход подаются пространственные координаты. На выходе хотим получить температуру в данной точке. Обучить данную нейросеть используя методику PINN. Провести оценку результатов обучения: скорость, точность ответа.

2 Обзор литературы

Одна из первых работ, посвященных PINNs, была опубликована в 2019 году Мазьяром Райсси и его коллегами [1]. В этой работе авторы представили метод, который позволяет использовать уравнения в частных производных в качестве ограничений для обучения нейронной сети. Они показали, что этот метод может быть использован для решения широкого спектра задач, включая уравнения Навье-Стокса, уравнения Максвелла и тому подобное.

В настоящее время PINN широко применяются моделировании, анализе широкого спектра физических явлений:

В статье [2] рассматривается задача симуляции циклической вольтаметрии, исследователями было рассмотрено несколько случаев: одномерная вольтаметрия на дисковом электроде с полубесконечными или тонкослойными граничными условиями, двумерная вольтаметрия на микрополосковом электроде и наконец вольтаметрия на края квадратного электрода,

количественно определяя неравномерное распределение тока вблизи угла электрода. Для моделирования был использован перцептрон использующий от трёх до шести скрытых слоёв, и гиперболический тангенс в качестве функции активации. Полученные исследователями данные хорошо согласуются с решениями этих же задач, полученными другими способами.

Так же PINN применяются для: анализа литий-ионных батарей [3, 4], для моделирование теплопереноса в системах со сложной геометрией [5, 6], решения уравнения Навье-Стокса для моделирования турбулентности [7], химической кинематике [8, 9], для решения задач оптимизации с ограничениями, таких как задачи оптимизации формы [10]. Для изучения биологических процессов существует разновидность PINN'ов – BINN (Biologically-informed neural network) [11]

3 Способы и методы решения задачи

3.1 Нейронные сети

Задачу решения системы дифференциальных уравнений можно рассматривать как задачу регрессии, то есть задачу нахождения непрерывного соотношения между зависимой переменной u и одной или несколькими независимыми переменными $x = (x^1, x^2, \dots, x^n)^T$ по обучающей выборке $T = \{x_i, u_i\}_{i=1}^{N_f}$ (в нашем случае начальным и граничным условиям). Следовательно нашей задачей является поиск функции $\bar{u} = \bar{u}(x, \theta)$, аппроксимирующей истинную функцию $u(x)$ для любых значений x . В качестве \bar{u} обычно выступает нелинейная функция, определяемая соответствующим методом регрессии и зависящая от своего набора параметров $\theta = (\theta_1, \theta_2, \dots, \theta_k)^T$. Набор параметров θ подбирается таким образом, чтобы

$$\theta = \min_{\theta} \|\bar{u}(x_i, \theta)\|, i = \overline{1, N_f} \quad (1)$$

процесс подбора этих параметров называется обучением модели.

Определение 1. Определим функцию регрессии \bar{u} следующим обра-

ЗОМ:

$$\bar{u}(x, \theta) = h\left(\sum_{j=1}^p w_j x^j + b\right) = h(wx + b) \quad (2)$$

где h – функция активации (обычно берётся сигмоида, \tanh или ReLu), $w = (w_1, w_2, \dots, w_p)$ – веса, b – смещение, w и b вместе являются параметрами θ функции \bar{u} , $\theta = (w_1, w_2, \dots, w_p, b)^T$.

Определение 2. Искусственной нейронной сетью называют последовательность нескольких функций из определения 1. Сеть с L скрытыми слоями и N нейронами на слой может быть записана следующим образом:

$$\begin{aligned} q^{(l,n)} &= h\left(\sum_{i=1}^N w_i^{(l,n)} q^{(l-1,i)} + b^{(l,n)}\right), n = 1, \dots, N, l = 1, \dots, L-1 \\ q^{(L)} &= \sum_{i=1}^N w_i^{(L)} q^{(L-1,i)} + b^{(L)} \end{aligned} \quad (3)$$

здесь $q^{(0,i)} = x^i$. Вся искусственная нейронная сеть может быть записана как

$$\bar{u} = q^{(L)} \quad (4)$$

Для описанной выше нейронной сети справедлива следующая теорема:

Теорема 1. Нейронная сеть, описанная в определении 2, имеющая по крайней мере один скрытый слой и сигмоидную функцию активации может аппроксимировать любую непрерывную функцию со сколь угодно большой точностью.

Данная теорема хотя и утверждает, что мы можем приблизить любую непрерывную функцию сколь угодно точно, это может потребовать большого количества обучающих данных и а так же большого количества нейронов, что приводит к большому размеру множества параметров θ . Более того в теореме ничего не сказано о том как находить параметры θ искомой нейронной сети, следовательно данный результат скорее теоретический чем практический. Как же в таком случае искать эти параметры.

Как было показано в уравнение (1) мы хотим подобрать параметры таким образом что бы расстояние между предсказанным решением и ис-

тинным решением на обучающих данных было минимизировано по какой то норме. Мы будем использовать один из наиболее распространённых вариантов – среднеквадратичную ошибку:

$$MSE(\theta, T) = \frac{1}{N_f} \sum_{i=1}^{N_f} (\bar{u}(x, \theta) - u_i)^2 \quad (5)$$

Одним из простейших методов поиска минимума $MSE(\theta, T)$ является градиентный спуск, записать его можно следующим образом:

$$\theta^{(i+1)} = \theta^{(i)} - \gamma(\nabla_{\theta} MSE(\theta^{(i)}, T)) \quad (6)$$

здесь γ – длина шага градиентного спуска или же скорость обучений. На основе данного метода был создан Adam, его мы и будем использовать.

3.2 Автоматическое дифференцирование

Для задания уравнений в частных производных нам понадобится уметь находить частные производные нейронной сети \bar{u} по входам x . Для этой цели мы воспользуемся автоматическим дифференцированием. Автоматическое дифференцирование использует тот факт, что любая функция является последовательностью элементарных операций (сложение, умножение и т.д.) совмещённых с элементарными функциями (синус, экспонента, логарифм и т.д.). Используя правило дифференцирования сложных функций и известные производные элементарных функций мы можем вычислить интересующую нас производную. Частным случаем автоматического дифференцирования является обратное распространение ошибки.

3.3 Теоретическое описание PINN

Пусть дана система дифференциальных уравнений:

$$F_j(x, u, \lambda_j) = F_j(x, u, u'_{x1}, u''_{x1}, \dots, \lambda_j) = 0, x \in \Omega, j = \overline{1, N} \quad (7)$$

с граничными условиями

$$B_k(x_0, u, u'_{x^1}, u''_{x^1}, \dots) = 0, x_0 \in \partial\Omega, k = \overline{1, K} \quad (8)$$

$x = (x^1, x^2, \dots, x^n)$ – независимые переменные из \mathbb{R}^n , Ω – некоторая область в пространстве \mathbb{R}^n , $\partial\Omega$ – её граница, $u(x)$ – искомая функция описывающая интересующие нас свойства системы (скорость, концентрация, потенциал и т.п.), λ_j – векторы постоянных параметров системы, такие как плотность вещества, заряд частиц, теплопроводность материала, температура окружающей среды и тому подобное.

Пусть $u(\bar{x}, \theta)$ – нейронная сеть, аппроксимирующая истинное решение $u(x)$. Для тренировки нейронной сети у нас имеется некоторое количество обучающих данных $T = \{x_b, u_b\}_{b=1}^{N_b}$, полученных из граничных условий. На данном этапе наша функция потерь выглядит следующим образом:

$$MSE_b = \sum_{k=1}^K \frac{1}{N_b} \sum_{b=1}^{N_b} (\bar{u}(x_b) - u_b)^2 \quad (9)$$

Теоретически мы можем приблизиться к решению u сколь угодно близко, если T достаточно велика. Однако решение может быть крайне сложным и следовательно требовать большого количества обучающих данных.

Что бы преодолеть это ограничение включим уравнение системы 7 в функцию потерь следующим образом

$$MSE_f = \sum_{j=1}^N \frac{1}{N_f} \sum_{i=1}^{N_f} F_j^2(x_i, \bar{u}(x_i), \lambda_j) \quad (10)$$

Тогда итоговая функция потерь примет следующий вид:

$$\begin{aligned} MSE &= MSE_f + MSE_b \\ &= \sum_{k=1}^K \frac{1}{N_b} \sum_{b=1}^{N_b} (\bar{u}(x_b) - u_b)^2 + \sum_{j=1}^N \frac{1}{N_f} \sum_{i=1}^{N_f} F_j^2(x_i, \bar{u}(x_i), \lambda_j) \end{aligned} \quad (11)$$

Так как физические законы, описываемые (7) напрямую включены в функцию потерь данную нейронную сеть можно назвать physics informed

neural network (PINN). Принципиальная схема работы PINN изображена на рисунке 3.1

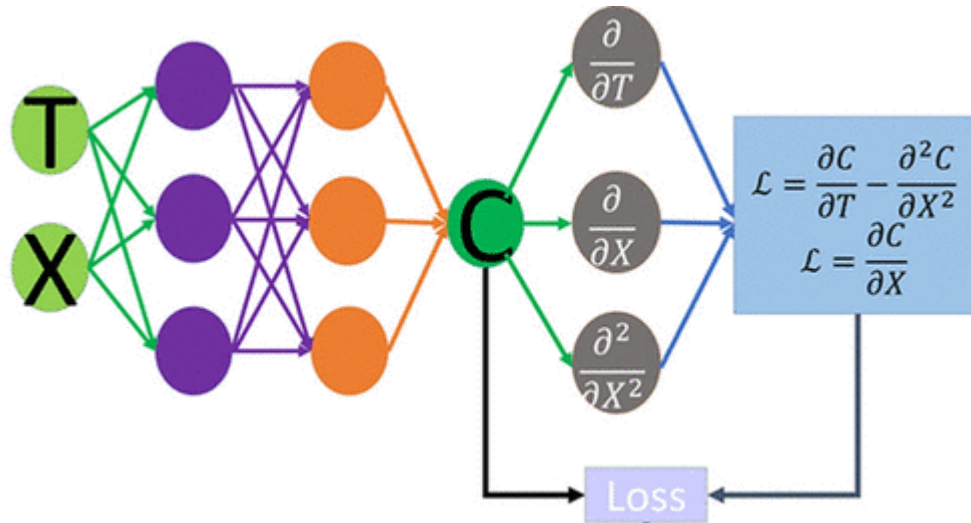


Рисунок 3.1 Принципиальная схема работы PINN

3.4 Техническая реализация

3.4.1 Используемые технологии

Для разработки будем использовать язык Python 3.10.6 – высокоуровневый язык программирования общего назначения, один из наиболее популярных языков в области машинного обучения и Tensorflow – библиотеку для создания и обучения нейронных сетей. Выбор данной библиотеки обусловлен простотой создания нейронных сетей с помощью Keras API, высокой производительностью, а так же встроенным автоматическим дифференцированием, которое и позволит нам обучить нейросеть дифференциальным уравнениям в частных производных.

3.4.2 Програмная реализация PINN

Общая архитектура сети $\bar{u}(x)$, аппроксимирующей функцию $u(x)$ была описана в разделе 3.1, точное число слоёв и нейронов в каждом слое будем выбирать экспериментально, в качестве функции активации слоя будем использовать \tanh . Для создания нейронной сети \bar{u} воспользуемся классом `tensorflow.keras.Model`, слоёв классом `tensorflow.keras.layers.Dense`.

Для удобства использования вынесем вычисление производных и функцию потерь в отдельную нейросеть-обёртку $f(\bar{u}, x)$. На вход сети f подаём N_f точек для системы (7) и N_b точек для граничных условий (8). Для этих точек вычисляем значения \bar{u} , а так же её частные производные с помощью автоматического дифференцирования, реализуемого классом `tensorflow.GradientTape`. Затем составляем из них и параметров λ_j систему уравнений (7) и граничные условия (8) они и будут являться выходами нейронной сети f . Функцией потерь у f будет (1). Именно эту сеть мы будем обучать, при этом так как она включает в себя \bar{u} , то одновременно будет происходить и её обучение. Как уже было сказано в разделе 3.1 в качестве оптимизатора будем использовать Adam, а в качестве метрики MSE.

4 Результаты и их обсуждение

4.1 Установившееся распределение тепла в кольце

Рассмотрим для начала относительно простую задачу: определить распределение тепла, установившееся в кольце $1 < r < 2$. $0 < \phi < 2\pi$, с граничными условиями:

$$\begin{aligned} u(1, \phi) &= \cos \phi + \sin \phi + \sin(2\phi) + 5 \sin(3\phi) + 1 \\ u(2, \phi) &= \sin(2\phi) + \sin(3\phi) + \cos(4\phi) \end{aligned} \tag{12}$$

Установившееся распределение тепла описывается уравнением Пуассона, учитывая что внутренних источников тепла нет получаем уравнение Лапласа:

$$\Delta u = 0 \tag{13}$$

Для полярных координат оно принимает вид

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} + \frac{1}{r^2} \frac{\partial^2 u}{\partial \phi^2} = 0 \tag{14}$$

Данная задача имеет аналитическое решение:

$$\begin{aligned}
u(r, \phi) = & 1 - \frac{\ln r}{\ln 2} \\
& + \left(\frac{-r}{3} + \frac{4}{3r} \right) \sin(\phi) + \left(\frac{-r}{3} + \frac{4}{3r} \right) \cos(\phi) \\
& + \left(\frac{r^2}{5} + \frac{4}{5r^2} \right) \sin(2\phi) \\
& + \left(\frac{3r^3}{63} + \frac{312}{64r^3} \right) \sin(3\phi) \\
& + \left(\frac{16r^4}{255} - \frac{16}{255r^4} \right) \cos(4\phi)
\end{aligned} \tag{15}$$

Для одной итерации обучения возьмём N_f внутренних точек для уравнения (15) и N_b точек для внутреннего и внешнего условий, обозначим их количество как N_i и N_o соответственно, всего $N_f + 2N_b$ точек. В данном случае функция потерь будем выглядеть следующим образом:

$$\begin{aligned}
MSE = & \frac{1}{N_f} \sum_{i=1}^{N_f} \left(\frac{\partial^2 u}{\partial r^2}(r_i, \phi_i) + \frac{1}{r} \frac{\partial u}{\partial r}(r_i, \phi_i) + \frac{1}{r^2} \frac{\partial^2 u}{\partial \phi^2}(r_i, \phi_i) \right)^2 + \\
& \frac{1}{N_i} \sum_{i=1}^{N_i} (u(1, \phi_i) - \cos \phi + \sin \phi + \sin(2\phi_i) + 5 \sin(3\phi_i) + 1)^2 + \\
& \frac{1}{N_o} \sum_{i=1}^{N_o} (u(2, \phi_i) - \sin(2\phi_i) + \sin(3\phi_i) + \cos(4\phi_i))^2
\end{aligned} \tag{16}$$

Возьмём сеть со скрытыми слоями [20,20,20,20] и проведём вычисления при различных значениях для N_f и N_b . Эпох 10000. Валидацию будем проводить следующим образом: для уравнения (15) разобьём область на радиальную сетку, с числом узлов N_f и сторонами $dr, d\phi$, так что бы $dr d\phi$, для граничных условий (16) равномерно расположим на $[0, 2\pi]$ N_b точек. Все расчёты производились на компьютере под управлением Windows 10, с процессором 12th Gen Intel(R) Core(TM) i5-12400 2.50 GHz.

По графикам обучения, изображённым на рисунке 4.1 видно, что при малом количестве внутренних точек N_f значения функции потерь на валидационной выборке в несколько раз больше значений функции потерь

на обучающей выборке, с увеличением N_f же значения функции потерь на обучающей и валидационной выборках практически совпадают.

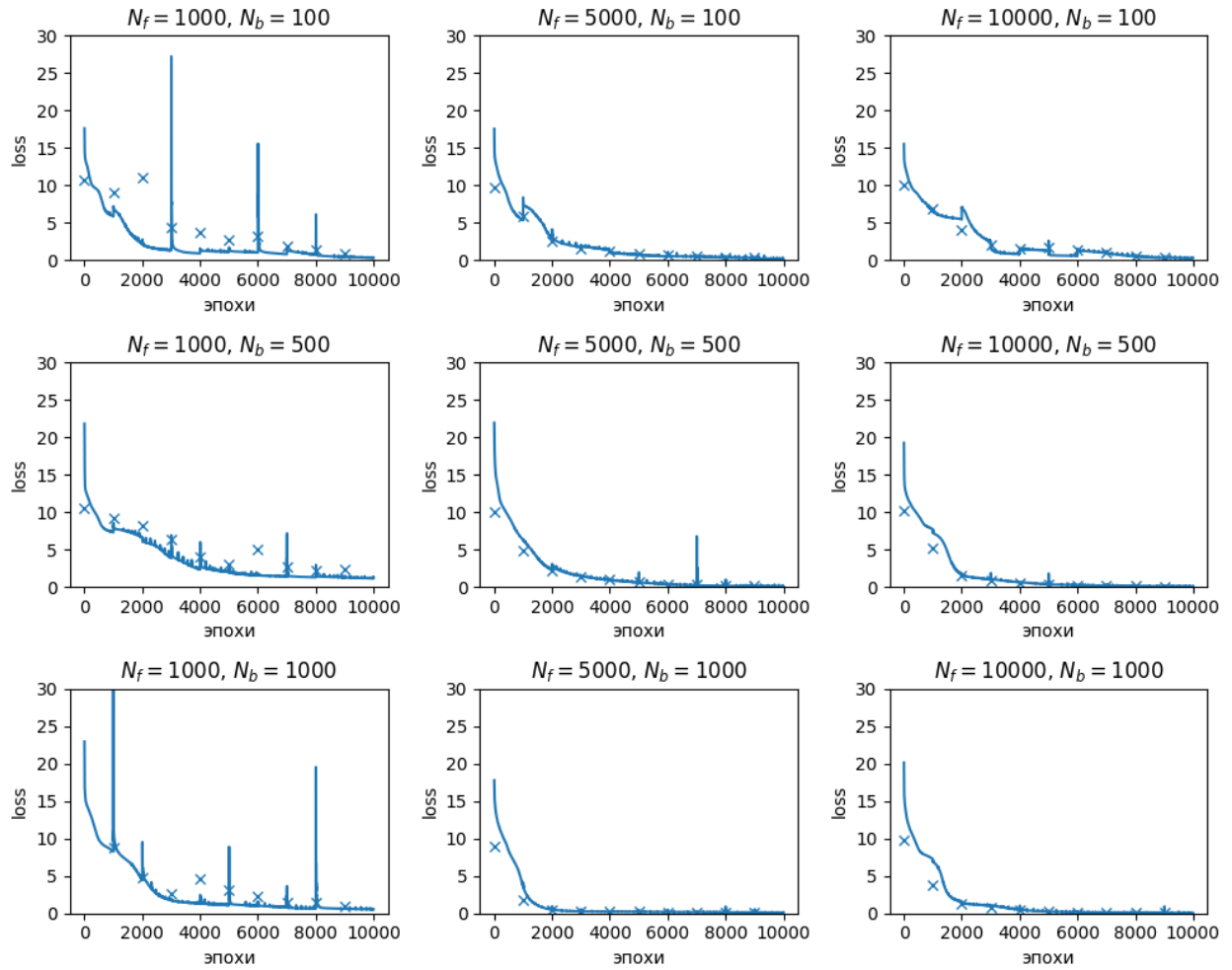


Рисунок 4.1 Графики функций потерь для различных сочетаний N_f и N_b , крестиками изображаются результаты валидации в определённые моменты времени

Время обучения, указанное в таблице 4.1 практически не зависит от количества точек для граничных условий N_b , вероятно потому что с ними не производится вычисление производных, а так же потому что их в целом меньше. В целом время обучения получается достаточно большим, возможным решением данной проблемы может быть изменение числа слоёв и количества нейронов в слоях.

Выполнение граничных условий при различных N_b изображено на рисунках 4.2, 4.3, 4.4. Для всех случаев характерно то, что внутренне условие при $r = 1$ выполняется почти идеально, исключение составляет случай $N_f = 1000, N_b = 500$, у которого изгибается конец, связано это вероятно

$N_b \backslash N_f$	1000	5000	10000
100	311.65	1108.51	3494.39
500	314.31	1112.09	3487.54
1000	312.27	1109.55	3472.66

Таблица 4.1 Время обучения в секундах для различных сочетаний N_f и N_b

со стохастической природой оптимизатора Adam. Внешнее условие однако выполняется относительно плохо. Связанно такое различие вероятнее всего с тем, что разброс величин у внутреннего условия больше – от -4 до 8, в то время как у внешнего в два раза меньше от -3 до 2, для решения данной проблемы в дальнейшем можно попробовать внести весовой коэффициент для граничного условия равный $\frac{1}{\max_{\phi} u(r_0, \phi)}$.

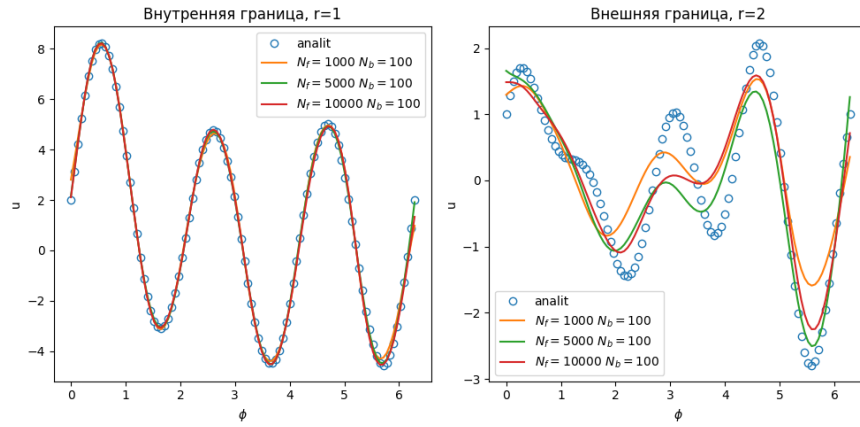


Рисунок 4.2 Граничные условия для случая $N_b = 100$

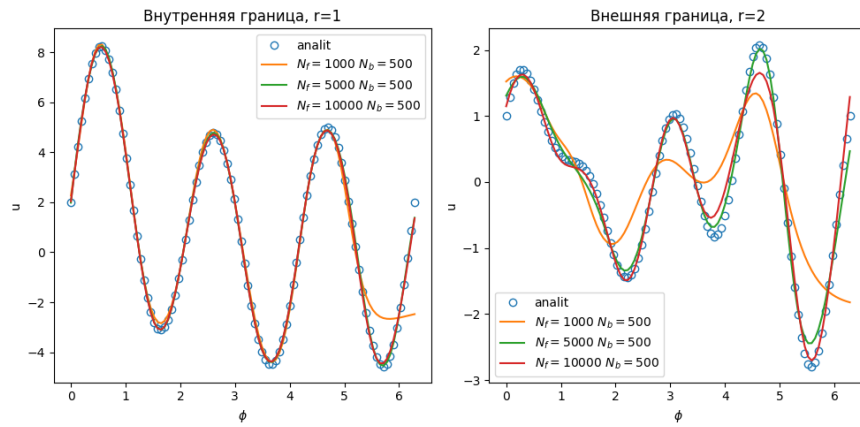


Рисунок 4.3 Граничные условия для случая $N_b = 500$

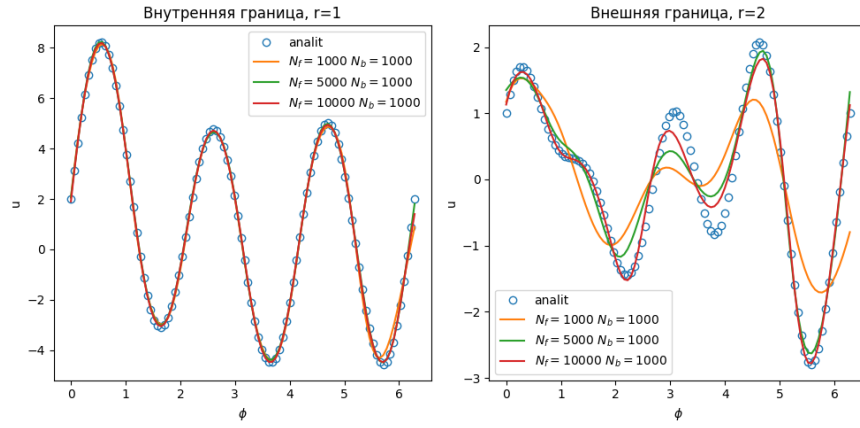


Рисунок 4.4 Граничные условия для случая $N_b = 1000$

Наконец посмотрим на решение в целом. Точное решение изображено на графике 4.5, решение при различных N_f и N_b изображено на графике 4.6, ошибка между аналитическим решением и предсказаниями нейросети на графике 4.7. В целом графики достаточно хорошо совпадают в области. Расхождения наблюдаются в основном на стыке, где $\phi = 0$. Что бы решить данную проблему можно попробовать добавить условие равенства $u(r, 0) = u(r, \phi)$, такому условию будет соответствовать следующая функция потерь:

$$MSE_s = \frac{1}{N_s} \sum_{i=1}^{N_s} (u(r_i, 0) - u(r_i, 2\pi))^2 \quad (17)$$

Здесь $\{r_i\}_{i=0}^{N_s}$ – точки для данного условия, N_s – их количество.

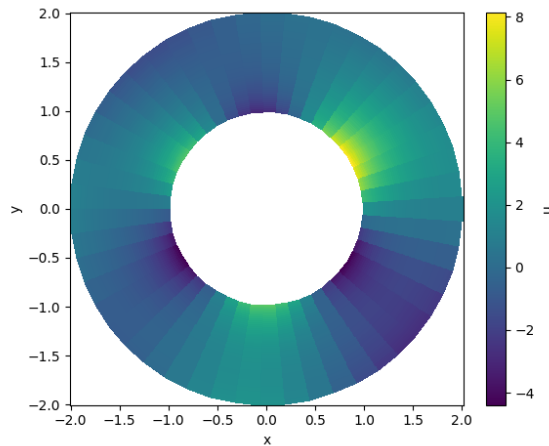


Рисунок 4.5 Аналитическое решение

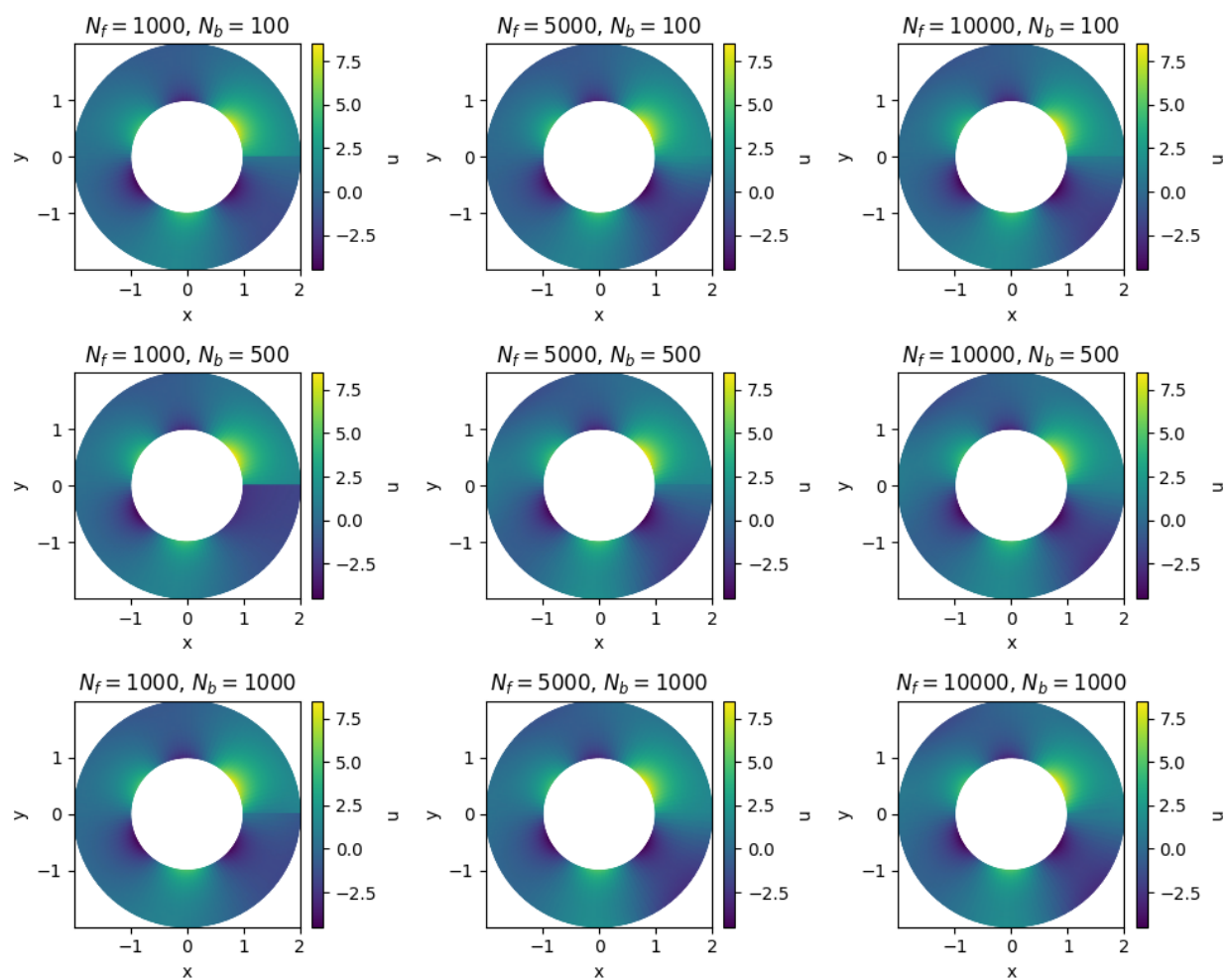


Рисунок 4.6 Решение при различных N_f и N_b

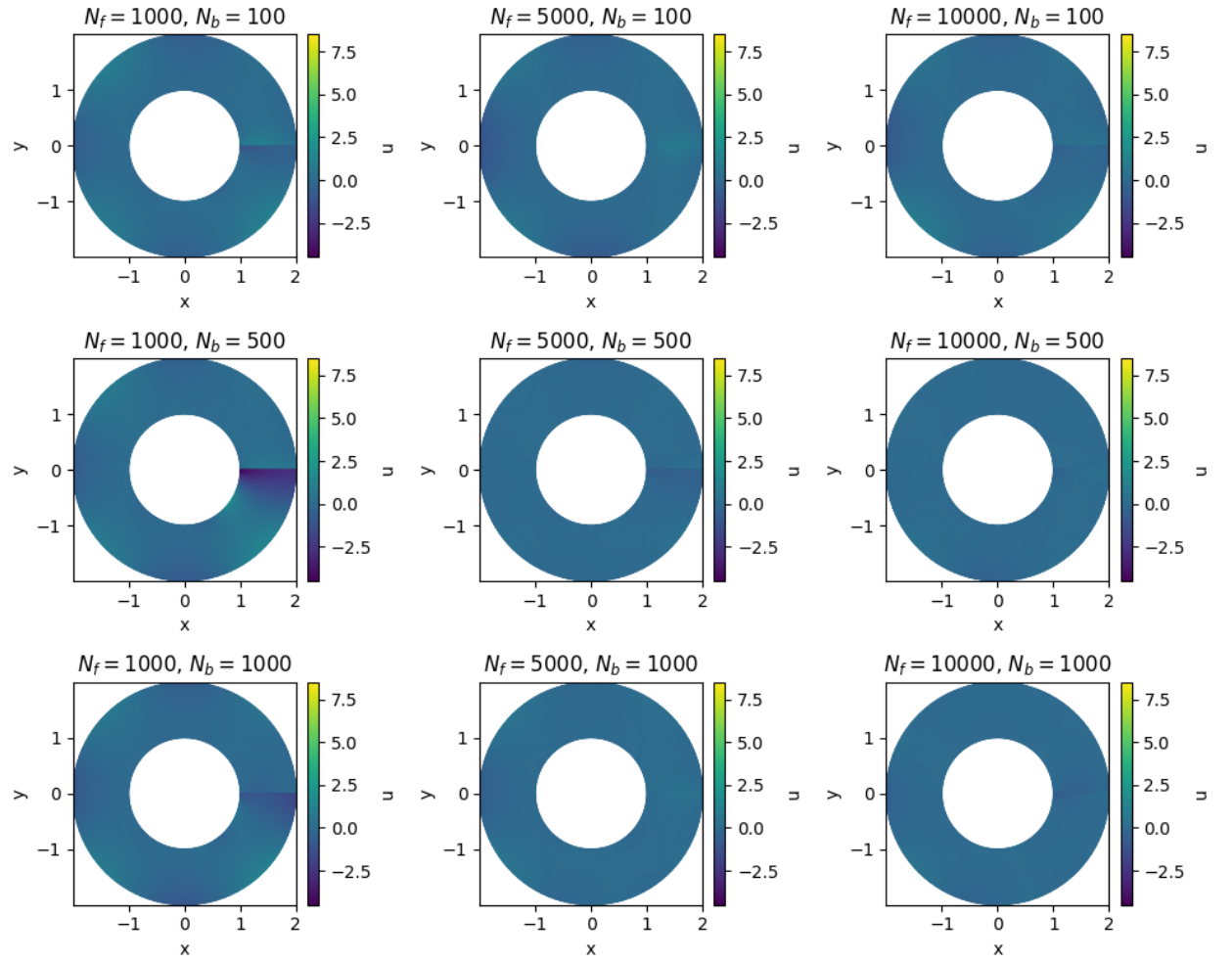


Рисунок 4.7 разница между аналитическим решением и решением нейросети при различных N_f и N_b

4.2 Задача электрокинетики

В качестве второго примера возьмём систему из [12]. Она описывается следующими уравнениями

$$\begin{aligned}
 \vec{j} &= -D\nabla c - \xi z e c \nabla \Phi + c \vec{v} \\
 \partial_t c &= -\nabla \cdot \vec{j} \\
 \nabla^2 \Phi &= -4\pi l_B k_B T z c \\
 \rho(\partial_t v + (v \cdot \nabla)v) &= -\nabla p_H + \eta \nabla^2 v - (k_B T \nabla c + z e c \nabla \Phi) \\
 \nabla \cdot v &= 0
 \end{aligned} \tag{18}$$

здесь c – концентрация ионных частиц, j – поток плотности, \vec{v} – адвективная скорость жидкости, e – заряд электрона z – валентность частиц, Φ – электростатический потенциал, ξ – подвижность частиц, D – коэффициент диффузии частиц, l_B – длина Бьеррума, $l_B = \frac{e^2}{4\pi\epsilon k_B T}$ k_B – постоянная Больцмана, T – температура, ρ – плотность жидкости p_H – гидродинамическое давление. Первое уравнение в системе описывает поток плотности, второе электростатику, третье гидродинамику с помощью уравнения Навье-Стокса, четвёртое уравнение несжимаемости жидкости.

Рассмотрим систему щелевых пор, состоящую из двух одноимённо заряженных бесконечных пластин. Выпишем для такой системы граничные условия

$$\begin{aligned}
 c(t, X_l) &= 0.01 \\
 c(t, X_r) &= 0.01 \\
 c(0, X) &= 0.002 \\
 v(t, X_l) &= 0 \\
 v(t, X_r) &= 0 \\
 v(0, X) &= 0 \\
 \Phi(t, X_l) &= -0.05 \\
 \Phi(t, X_r) &= -0.05 \\
 \Phi(0, X) &= -0.009x^2 + 2
 \end{aligned}$$

здесь t – время, X_l – пространственные координаты, соответствующие ле-

вой стенке, X_r – правой, x в формул для $\Phi(0, x)$ соответствует оси, перпендикулярной пластинам.

В данной системе мы имеем 3 неизвестные – концентрацию c , скорость v и потенциал Φ , соответственно наша сеть \bar{u} будет иметь три выхода \bar{c} , \bar{v} и $\bar{\Phi}$. Составим функцию потерь. Из системы (18) получаем:

$$\begin{aligned}
MSE_f = & \sum_{i=1}^{N_f} (\partial_t c + \nabla \cdot (-D \nabla c - \xi z e c \nabla \Phi + c v))^2 \\
& + \sum_{i=1}^{N_f} (\nabla^2 \Phi + 4\pi l_B k_B T z c)^2 \\
& + \sum_{i=1}^{N_f} (\rho(\partial_t v + (v \cdot \nabla) v) + \nabla p_H - \eta \nabla^2 v + (k_B T \nabla c + z e c \nabla \Phi))^2 \\
& + \sum_{i=1}^{N_f} (\nabla \cdot v)^2
\end{aligned} \tag{19}$$

ЗАКЛЮЧЕНИЕ

В ходе работы мы написали, обучили и протестировали PINN, который решает задачу распределения тепла в кольце. Нам удалось получить в целом достаточно хорошее решение, близкое к истинному, однако решение занимает относительно много времени, что делает нецелесообразным использование PINN в текущем варианте.

Возможные улучшения:

- 1) Подобрать весовые коэффициенты для функций потерь
- 2) Поэкспериментировать с числом слоёв и нейронов в каждом слое
- 3) Добавить дополнительное условие, требующее равенства на границе при $\phi = 0, 2\pi$

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

- 1 Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.
- 2 Haotian Chen, Enno Kätelhön, and Richard G. Compton. Predicting voltammetry using physics-informed neural networks. *The Journal of Physical Chemistry Letters*, 13(2):536–543, 2022. PMID: 35007069.
- 3 Muratahan Aykol, Chirranjeevi Balaji Gopal, Abraham Anapolsky, Patrick K. Herring, Buis van Vlijmen, Marc D. Berliner, Martin Z. Bazant, Richard D. Braatz, William C. Chueh, and Brian D. Storey. Perspective-combining physics and machine learning to predict battery lifetime. *Journal of the Electrochemical Society*, 168(3), 2021.
- 4 Renato G. Nascimento, Matteo Corbetta, Chetan S. Kulkarni, and Felipe A. C. Viana. Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis. *Journal of Power Sources*, 1(513), 2021.
- 5 Navid Zobeiry and Keith D. Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021.
- 6 Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6), 04 2021. 060801.
- 7 Fabian Pioch, Jan Hauke Harmening, Andreas Maximilian Müller, Franz-Josef Peitzmann, Dieter Schramm, and Ould el Moctar. Turbulence modeling for physics-informed neural networks: Comparison of different rans models for the backward-facing step flow. *Fluids*, 8(2), 2023.

- 8 Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021. PMID: 34463510.
- 9 Gabriel S. Gusmão, Adhika P. Retnanto, Shashwati C. da Cunha, and Andrew J. Medford. Kinetics-informed neural networks. *Catalysis Today*, 2022.
- 10 Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation, 2020.
- 11 John H. Lagergren, John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data, 2020.
- 12 Jean-Noël Grad. Electrokinetics.