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.
ФИО руководителя Подпись

Студент Кузнецов И.А.
ФИО студента

Специальность (направление подготовки) 01.03.01 Математика

Группа МЕН-490102

Екатеринбург 2023

Содержание

1 Введение

В последние годы нейронные сети получили широкое распространение, они широко используются для анализа и генерации изображений и видео, обработки естественных языков (перевод, чат-боты), медицинской диагностике, 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 ограничения, 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.

Впервые терми PINN был введён в статье [1]. В ней автор дал формальное определение PINN'ам и рассмотрел решение нескольких задач: уравнение Шрёдингера, Навье-Стокса, Ален-Чана.

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

В статье [2] рассматривается задача симуляции циклической вольтаметрии, исследователями было рассмотрено несколько случая: 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 изучения биологических процессов существует разновидность PINN'ов – BINN (Biologically-informed neural network) [10]

2 Постановка задачи

2.1 Цель работы

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

2.2 Описание задачи

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

3 Теоретическое описание PINN

Пусть система описывается некой системой дифференциальных уравнений:

$$F_j(t, x, u, \lambda_j) = 0, x \in \Omega, t > 0, j = \overline{1, N} \tag{1}$$

с граничными условиями

$$B_k(t_0, x_0, u) = 0, t_0, x_0 \in \text{граничные точки} \tag{2}$$

где t – время, x – пространственные координаты, Ω – некоторая область в пространстве $\mathbb{R}^n$, $u(t, x)$ – 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.

Определим $f(t, x)$ следующим образом:

$$f(t, x) := \begin{pmatrix} F_1(t, x, u) \\ F_2(t, x, u) \\ \dots \\ F_N(t, x, u) \end{pmatrix} \tag{3}$$

где $u(t, x)$ аппроксимируется с помощью глубокой нейронной сети. $f(t, x)$ назовём физически-информированной нейросетью, или же PINN, она может быть получена с помощью автоматического дифференцирования сложных функций. Данная имеет все те же параметры, что и сеть $u(t, x)$, а так же дополнительно набором параметров λ.

Для обучения нейросети составим следующую функцию потерь:

$$MSE = MSE_f + MSE_u \tag{4}$$

где

$$MSE_f = \sum_{j=1}^{N} \frac{1}{N_f} \sum_{i=1}^{N_f} F_j^2(t_f^i, x_f^i, u(t_f^i, x_f^i)) \tag{5}$$

требует соблюдения дифференциальных уравнений, описывающих про-

цесс, здесь $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ – точки коллокации для F_j, N_f – количеств этих точек и

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u}((u(t_u^i, x_u^i)) - u_0^i)^2 \tag{6}$$

требует соблюдения граничных условий $\left(t_0^i, x_0^i, u_0^i\right)_{i=1}^{N_0}$ для функции $u(t, x)$. Принципиальная схема работы PINN изображена на рисунке 1

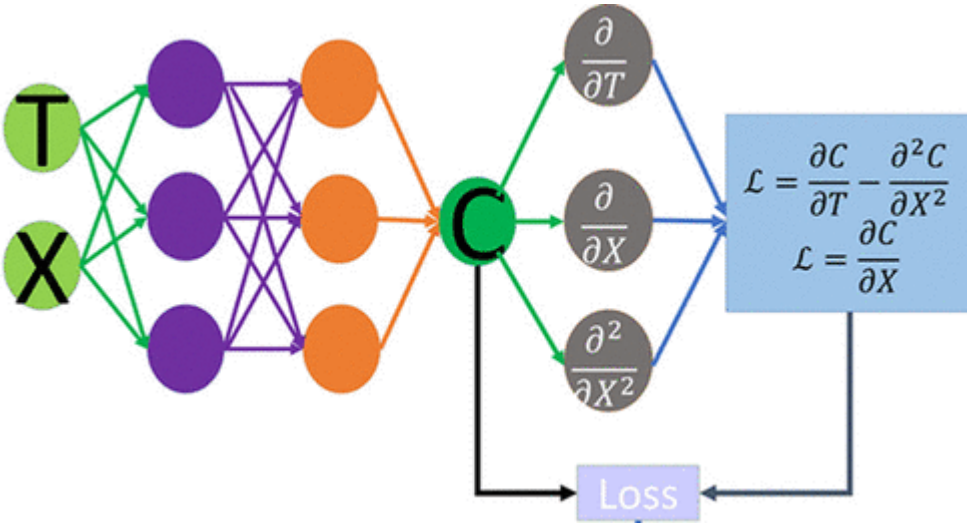

Рис. 1. Принципиальная схема работы PINN

4 Техническая реализация

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, высокой производительностью, а так же встроенным автоматическим дифференцированием, которое и позволит нам обучить нейросеть дифференциальным уравнениям в частных производных.

4.2 Общая архитектура PINN

В качестве архитектуры нейронной сети, аппроксимирующей функцию $u(t, x)$ возьмём многослойный перцептрон, точное число слоёв и нейронов в каждом слое будем выбирать экспериментально, в качестве функции активации слоя будем использовать tanh. Для создания нейронной сети воспользуемся классом `tensorflow.keras.Model`, для создания слоёв классом `tensorflow.keras.layers.Dense`. PINN $f(t, x)$ будет иметь всего один слой. Параметрами этого слоя и есть параметры λ_j из системы (1). На вход слой получает N_f точек (1) и N_u точек для граничных условий. Внутри этого слоя мы считаем частные производные $u(t, x)$ с помощью `tensorflow.GradientTape`, и составлять из них и параметров λ_j систему уравнений (1). На выход из данного слоя будем выдавать значения $u(t, x)$ и сами эти уравнения. В силу вида системы (1) все выходы, соответствующие уравнениям системы (1) должны быть равны 0. В качестве оптимизатора будем использовать Adam, а в качестве метрики MSE

5 Эксперименты

5.1 Установившееся распределение тепла в кольце

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

$$\begin{aligned} u(1,\phi) &= \cos\phi + \sin\phi + \sin(2\phi) + 5\sin(3\phi) + 1 \\ u(2,\phi) &= \sin(2\phi) + \sin(3\phi) + \cos(4\phi) \end{aligned} \tag{7}$$

Установившееся распространение тепла описывается уравнением Пуассона, учитывая что внутренних источников тепла нет получаем:

$$\Delta u = 0 \tag{8}$$

Для полярных координат оно принимает вид

$$\frac{\partial^2 u}{\partial r} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \phi^2} = 0 \tag{9}$$

Данная задача имеет аналитическое решение:

$$\begin{aligned} u(r,\phi) = 1 &- \frac{\ln r}{\ln 2} \\ &+ \left(\frac{-r}{3} + \frac{4}{3r}\right)\sin(\phi) + \left(\frac{-r}{3} + \frac{4}{3r}\right)\cos(\phi) \\ &+ \left(\frac{r^2}{5} + \frac{4}{5r^2}\right)\sin(2\phi) \\ &+ \left(\frac{3r^3}{63} + \frac{312}{64r^3}\right)\sin(3\phi) \\ &+ \left(\frac{16r^4}{255} - \frac{16}{255r^4}\right)\cos(4\phi) \end{aligned} \tag{10}$$

Для одной итерации обучения возьмём N_f внутренних точек для уравнения (10) и N_u точек для внутреннего и внешнего условий, обозначим их количество как N_i и N_o соответственно, всего $N_f + 2N_u$ точек.

В данном случае функция потерь будем выглядеть следующим образом:

$$
\begin{aligned}
MSE =& \frac{1}{N_f}\sum_{i=1}^{N_f}\left(\frac{\partial^2 u}{\partial r}(r_i,\phi_i)+\frac{1}{r}\frac{\partial u}{\partial r}(r_i,\phi_i)+\frac{1}{r^2}\frac{\partial^2 u}{\partial \phi^2}(r_i,\phi_i)\right)^2+ \\
& \frac{1}{N_i}\sum_{i=1}^{N_i}(u(1,\phi_i)-\cos\phi+\sin\phi+\sin(2\phi_i+5\sin(3\phi_i+1)^2+ \\
& \frac{1}{N_o}\sum_{i=1}^{N_o}(u(2,\phi_i)-\sin(2\phi_i+\sin(3\phi_i+\cos(4\phi_i)^2
\end{aligned}
\tag{11}
$$

Возьмём сеть со скрытыми слоями [20,20,20,20] и проведём вычисления при различных значениях для N_f и N_u. Эпох 10000, размер батча 32 – стандартный размер батча в `tensorflow`. Валидацию буем следующим образом: для уравнения (10) разобьём область на радиальную сетку, с числом узлов N_f и сторонами $dr, d\phi$, так что бы dr $d\phi$, для граничных условий (11) равномерно расположим на $[0, 2\pi]$ N_u точек.

По графикам обучения, изображённым на рисунке 2 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 совпадаю.

Время обучения, указанное в таблице 1 практически не зависит от количества точек для граничных условий N_u, 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_u изображено на рисунках 3,4,5. Для всех случаев характерно то, что внутренне условие при $r = 1$ выполняется почти идеально, исключение составляет случай $N_f = 1000, N_u = 500$, у которого изгибается конец, связанно это вероятно со стохастической природой оптимизатора Adam. Внешнее условие однако выполняется относительно плохо. Связанно такое различие веро-

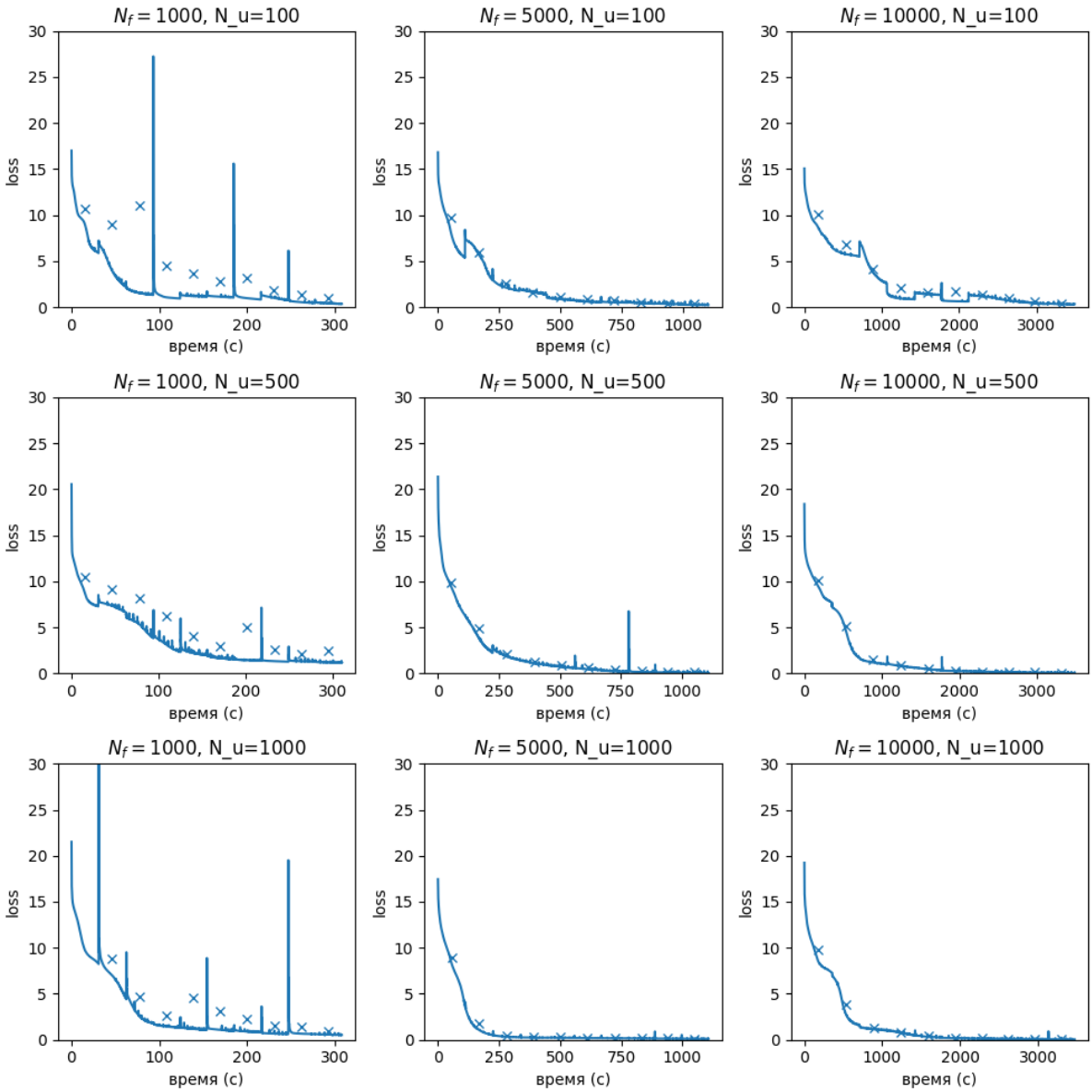

Рис. 2. Графики функций потерь для различных сочетаний N_f и N_u, крестиками изображаются результаты валидации в определённые моменты времени

ятнее 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 $\frac{1}{\max_\phi u(r_0,\phi)}$.

N_u \ 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 1. Время обучения в секундах для различных сочетаний N_f и N_u

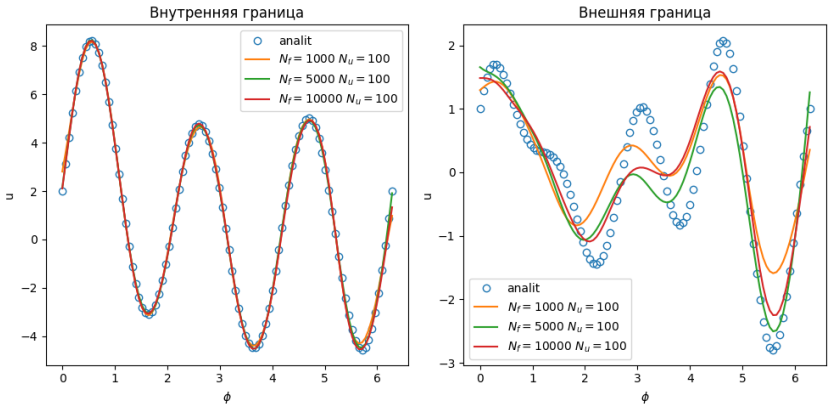

Рис. 3. Граничные условия для случая $N_u = 100$

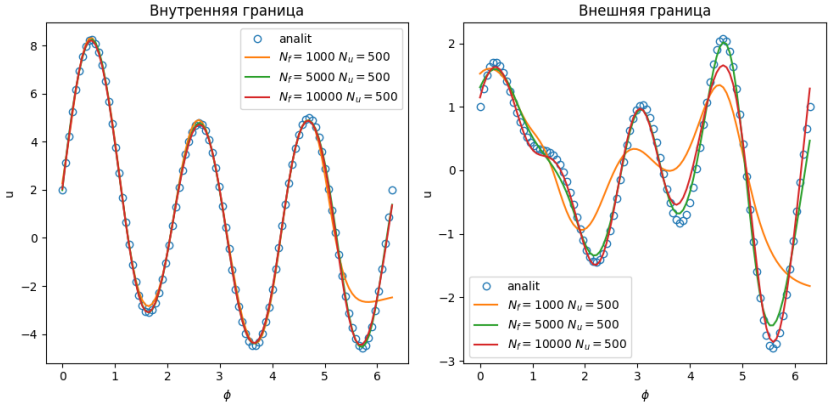

Рис. 4. Граничные условия для случая $N_u = 500$

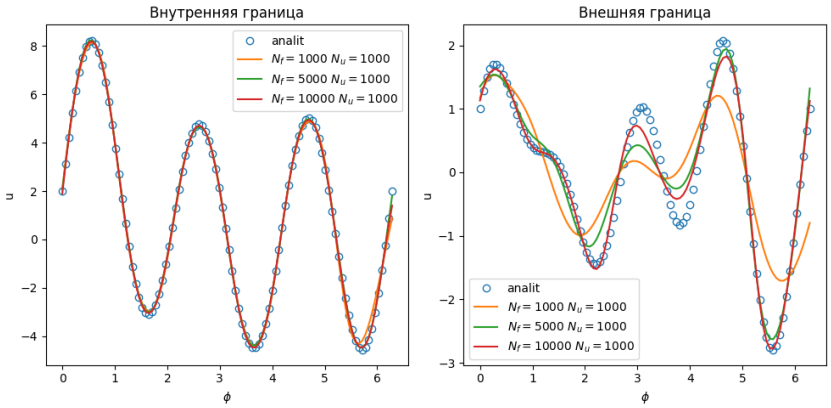

Рис. 5. Граничные условия для случая $N_u = 1000$

Наконец посмотрим на решение в целом. Точное решение изображено на графике 6, решение при различных N_f и N_u изображено на графике 7, ошибка между аналитическим решением и предсказаниями нейросети на графике 8. 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:

$$MSE_s = \frac{1}{N_s} \sum_{i=1}^{N_s} (u(r_i, 0) - u(r_i, 2\pi))^2 \tag{12}$$

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

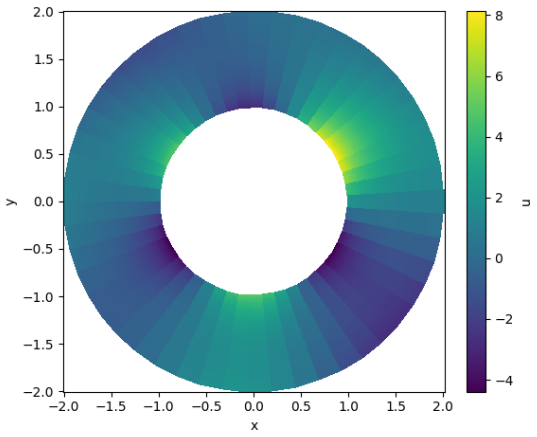

Рис. 6. Аналитическое решение

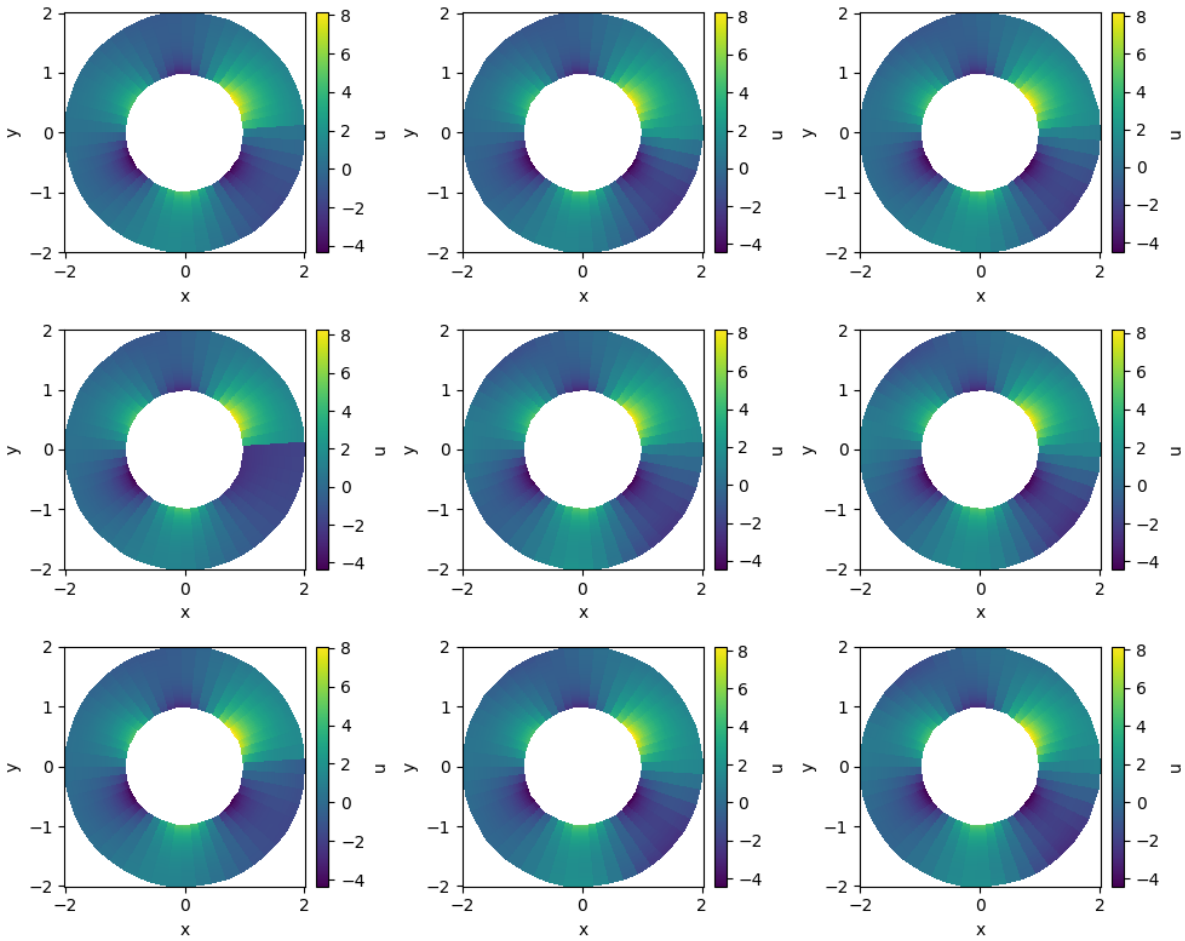

Рис. 7. Решение при различных N_f и N_u

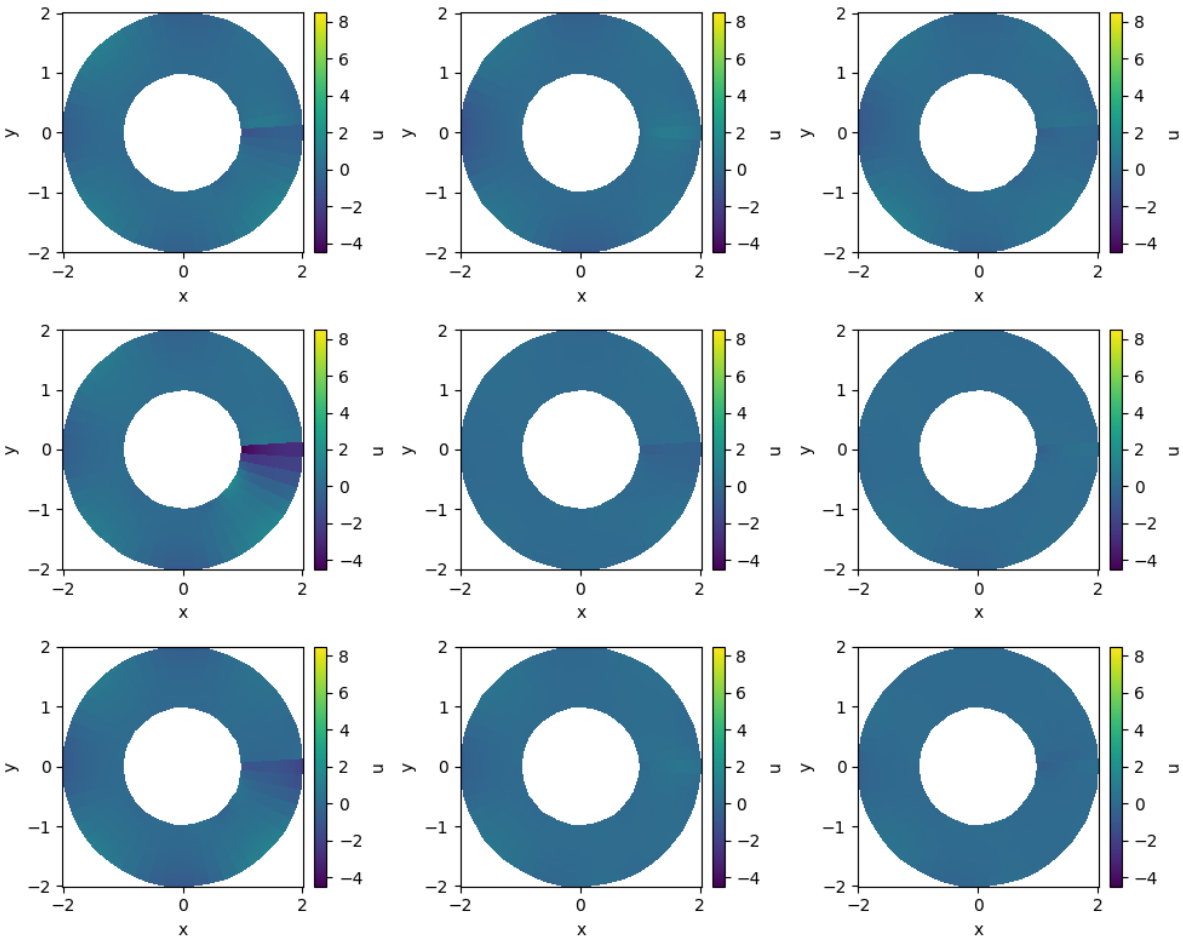

Рис. 8. разница между аналитическим решением и решением нейросети при различных N_f и N_u

6 Заключение

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. Так же нами были высказаны предположения по улучшению точности решения и ускорению работы PINN.

7 Список 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, Bruis 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] 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.