

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
(НИЯУ МИФИ)

ОТЧЕТ

по дисциплине «Проектная практика»
на тему «Исследование влияния весового коэффициента в функции
потерь PINN в двумерном уравнении Пуассона»

Группа

Б23-215

Студенты

Арнаутов П.А, Ефимов В.А, Меркулов А.М

Руководители работы
к.ф.-м.н., доценты

Карачурин Р.Н, Ладыгин С.А

Москва 2024

Аннотация

Отчет посвящен влиянию весового коэффициента в функции потерь PINN на точность решения

Двумерное уравнение Пуассона с граничными условиями Дирихле:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), & \text{для } (x, y) \in \Omega \\ u(x, y) = g(x, y), & \text{для } (x, y) \in \partial\Omega \end{cases} \quad (1)$$

Задача: Реализовать PINN метод решения краевой задачи для уравнения Пуассона. Исследовать влияние весового коэффициента λ в функции потерь PINN на точность и эффективность решения.

Содержание

| | | |
|----|-------------------------------|---|
| 1. | Физическое описание | 3 |
| 2. | Обзор метода | 5 |
| 3. | Реализация | 6 |
| 4. | Исследование | 7 |

Введение

Наше исследование посвящено решению краевой задачи с помощью PINN (Physics-Informed Neural Networks) Дифференциальные уравнения — очень важная часть математики, это целый математический аппарат (Дифференциальные уравнения в частных производных) широко применяемый при разработке моделей в разных областях науки и техники.

К сожалению, явное решение этих уравнений в аналитическом виде оказывается возможным только в частных простых случаях. Поэтому для анализа математических моделей, основанных на дифференциальных уравнениях, используются приближенные численные методы.

В нашем исследовании мы используем метод PINN, он отличается от классических методов (Метод конечных разностей, метод конечных элементов, метод конечных объемов и пр.) и объединяет методы машинного обучения с фундаментальными физическими законами.

При этом для численных методов характерны: большой объем вычислений, использование вычислительных систем. Поэтому с появлением нейропроцессоров и цифровых сигнальных процессоров метод становится особенно интересным, ведь может дать выигрыш в скорости выполнения, а также он может помочь в случаях, когда традиционные численные методы сталкиваются с трудностями. (Этот метод предлагает новый взгляд на решение сложных физических задач, особенно в случаях, когда традиционные численные методы сталкиваются с трудностями.)

1. Физическое описание

Уравнение Пуассона — эллиптическое дифференциальное уравнение в частных производных.

Где можно встретить данное уравнение:

Электростатика:

$$\nabla^2 \varphi = -\frac{\rho}{\varepsilon_0}, \quad (2)$$

где

- φ — электростатический потенциал (В),
- ρ — объемная плотность заряда (Кл/м³),
- ε_0 — диэлектрическая проницаемость вакуума (Ф/м),
- $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$ — оператор Набла.

Вывод уравнения из закона Гаусса $\operatorname{div} \mathbf{E} \equiv \nabla \cdot \mathbf{E} = \rho/\varepsilon_0$ и определения статического потенциала ($\mathbf{E} = -\nabla \varphi$) (\mathbf{E} — напряженность электрического поля):

$$\frac{\rho}{\varepsilon_0} = \nabla \cdot \mathbf{E} = \nabla \cdot (-\nabla \varphi) = -\nabla \cdot \nabla \varphi = -\nabla^2 \varphi. \quad (3)$$

(Еще одно применение, но такое же по сути) Если мы имеем объемную сферически симметричную плотность гауссова распределения заряда $\rho(r)$:

$$\rho(r) = \frac{Q}{\sigma^3(\sqrt{2\pi})^3} e^{-r^2/(2\sigma^2)}, \quad (4)$$

где

- E — (В),
 - Q — общий заряд системы,
 - σ — поверхностная плотность заряда,
- то решение $\Phi(r)$ уравнения Пуассона:

$$\nabla^2 \Phi = -\frac{\rho}{\varepsilon_0}, \quad (5)$$

дается в виде:

$$\Phi(r) = \frac{1}{4\pi\varepsilon_0} \frac{Q}{r} \operatorname{erf}\left(\frac{r}{\sqrt{2}\sigma}\right), \quad (6)$$

где $\operatorname{erf}(x)$ — функция ошибок.

Это решение может быть проверено напрямую вычислением $\nabla^2 \Phi$.

Гравитационные силы:

Расчет гравитационного потенциала φ . Его градиент определяет напряженность гравитационного поля. Потенциал φ , создаваемый точечной массой m , расположенной в начале координат, равен

$$\varphi_m = -\frac{Gm}{r}, \quad (7)$$

где

- G — гравитационная постоянная,
- r — расстояние от начала координат.

На бесконечности потенциал такого вида обращается в ноль. В общем случае произвольного распределения массы, описываемого координатно-зависимой плотностью ρ (кг/м³), уравнение Пуассона записывается:

$$\nabla^2 \varphi = 4\pi G \rho. \quad (8)$$

С точностью до замены $-1/\varepsilon_0 \rightarrow 4\pi G$ и изменения смысла величины ρ («плотность заряда» \rightarrow «плотность массы»), уравнение подобно соответствующему электростатическому уравнению. Правда, в случае гравитационных сил не бывает ситуации отталкивания, но на решении этот факт никак не сказывается.

Решение имеет вид:

$$\varphi(x, y, z) = \int G \frac{\rho(\xi, \eta, \zeta)}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + (z-\zeta)^2}} d\xi d\eta d\zeta. \quad (9)$$

Рассмотрение уравнения Пуассона в остальных областях физики (в которых он встречается) может быть выполнено аналогично, только со специфическим для конкретной области смыслом входящих в него величин.

2. Обзор метода

Идея состоит в том, чтобы решать краевые задачи (определенные на Декартовой прямоугольной системе координат), представляя численное решение с помощью нейронной сети и обучая полученную сеть удовлетворять условиям, требуемым дифференциальным уравнением и граничными условиями. Это решение включает нейронную сеть в качестве основного элемента аппроксимации, параметры которой (веса и смещения) подстраиваются для минимизации функции ошибки (Loss). Для обучения мы используем методы оптимизации, которые требуют вычисления градиента ошибки по параметрам сети.

Разберем метод на нашем уравнении Пуассона:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y).$$

Возьмем, например, $\forall x, y \in \Omega = [0, 2] \times [0, 2]$. Вид граничных условий Дирихле такой:

$$u(x, y) = g(x, y), \text{ для } (x, y) \in \partial\Omega.$$

Чтобы решить эту проблему, мы аппроксимируем решение с помощью нейронной сети:

$$NN(x, y) \approx u(x, y).$$

Если $NN(x, y)$ является истинным решением, то должно выполняться равенство

$$\frac{\partial^2 NN(x, y)}{\partial x^2} + \frac{\partial^2 NN(x, y)}{\partial y^2} = f(x, y) \quad \forall (x, y) \in \Omega.$$

Таким образом, мы превращаем это условие в нашу функцию потерь. Это обосновывает выбор функции потерь. Обозначив параметры нейросети (веса и смещения) буквой ω , запишем:

$$L(\omega) = \frac{1}{n} \sum_{x_i, y_i \in \Omega \setminus \partial\Omega} \left(\frac{\partial^2 NN(x_i, y_i)}{\partial x^2} + \frac{\partial^2 NN(x_i, y_i)}{\partial y^2} - f(x_i, y_i) \right)^2$$

— среднеквадратичная функция потерь

Выбор x_i, y_i может быть выполнен различными способами, в нашем случае, времена множество точек представлено в виде сетки прямоугольной равномерной сетки. В любом случае, когда эта функция потерь минимизируется (в нашем случае вычисляем градиенты с помощью стандартного обратного дифференцирования tensorflow), мы имеем, что

$$\frac{\partial^2 NN(x_i, y_i)}{\partial x^2} + \frac{\partial^2 NN(x_i, y_i)}{\partial y^2} \approx f(x_i, y_i),$$

и, следовательно, $NN(x, y)$ аппроксимирует решение краевой задачи.

Но нам все еще нужно учесть граничное условие. Один из простых способов сделать это — добавить потери на границе в функцию стоимости.

$$L(\omega, \lambda) = \frac{1}{n} \sum_{x_i, y_i \in \Omega \setminus \partial\Omega} \left(\frac{\partial^2 NN(x_i, y_i)}{\partial x^2} + \frac{\partial^2 NN(x_i, y_i)}{\partial y^2} - f(x_i, y_i) \right)^2 + \\ + \frac{\lambda}{n} \sum_{x_i, y_i \in \partial\Omega} (NN(x_i, y_i) - u(x_i, y_i))^2$$

где λ — гиперпараметр, определяющий нейронную сеть NN , аппроксимирующую u . Таким образом, решение задачи снова сводится к нахождению весов, которые минимизируют функцию потерь!

3. Реализация

Область определения и граница

В нашей задаче не даны определенные функции, они могут варьироваться в зависимости от того, что именно уравнение описывает. Мы решили выбрать уравнение

$$u = \sin(\pi x) \sin(\pi y), \quad x, y \in \text{int}(\Omega),$$

где $\Omega = [0, 2] \times [0, 2]$, тогда $\forall (x, y) \in \partial\Omega \implies g \equiv 0$

$$f(x) = \sin(5x) \tag{10}$$

$$f(x) = \sin(x) \tag{11}$$

Описание модели нейронной сети

Решение основывается на библиотеке Python TensorFlow и Keras (который встроен в TensorFlow) — это открытые программные библиотеки для машинного обучения.

Для эффективного обновления параметров модели был использован оптимизатор Adam.

Архитектура модели

- Количество слоев: 6
 - 1 входной слой
 - 5 скрытых слоев
 - 1 выходной слой
- Количество нейронов:
 - Входной слой: 2 нейрона
 - Скрытые слои: 32 нейрона в каждом
 - Выходной слой: 1 нейрон

Функции активации

В модели используются следующие функции активации для скрытых слоев:

- 1-й скрытый слой: $\sin(5x)$, договоримся называть `custom_function2`,
- 2-5 скрытые слои: $\sin(x)$, договоримся называть `custom_function1`.

Такие функции были выбраны, чтобы учесть период решения, то есть множитель π в \sin в решении, но для упрощения вычислений на следующих слоях мы решили использовать обычный $\sin(x)$.

Выходной слой использует линейную активацию. Это означает, что он просто выдает результат полученный прохождением через предыдущие слои нейросети.

Визуализация архитектуры

Ниже представлена текстовая визуализация архитектуры нейронной сети:

| Layer (type) | Output Shape | Param # |
|------------------|--------------|---------|
| dense_6 (Dense) | (None, 32) | 96 |
| dense_7 (Dense) | (None, 32) | 1,056 |
| dense_8 (Dense) | (None, 32) | 1,056 |
| dense_9 (Dense) | (None, 32) | 1,056 |
| dense_10 (Dense) | (None, 32) | 1,056 |
| dense_11 (Dense) | (None, 1) | 33 |

Total params: 4,353 (17.00 KB)

Trainable params: 4,353 (17.00 KB)

Non-trainable params: 0 (0.00 B)



Figure 1. Нейронная сеть - состояние

Наша нейронная сеть будет обучена для нахождения функции, которая удовлетворяет этому уравнению, используя соответствующие граничные условия и данные. Нетренируемых параметров, иначе гиперпараметров нет, потому что коэффициент λ мы подбираем сами, тренируя нейросеть с одинаковыми изначальными параметрами в цикле.

4. Исследование

Для оценки натренированных моделей мы использовали следующие функции ошибки:

$$\text{MSE} = \frac{1}{n} \sum_{x_i, y_i \in \Omega} (NN(x_i, y_i) - u(x_i, y_i))^2,$$

$$\text{MAX} = \max_{x_i, y_i \in \Omega} |NN(x_i, y_i) - u(x_i, y_i)|.$$

При подборе гиперпараметра часто используется достаточно небольшое количество эпох, чтобы определить динамику сходимости и погрешности. Мы не всегда придерживались этого принципа. Например, для демонстрации какой-никакой работоспособности метода для диапазона параметров в промежутке $[0.3, 20000]$ полученного с помощью функции `numpy.logspace`, генерирующей последовательность в заданном промежутке, возрастающую экспоненциально, мы тренировали модель в течение 15 000 эпох.

Это дало следующие результаты:

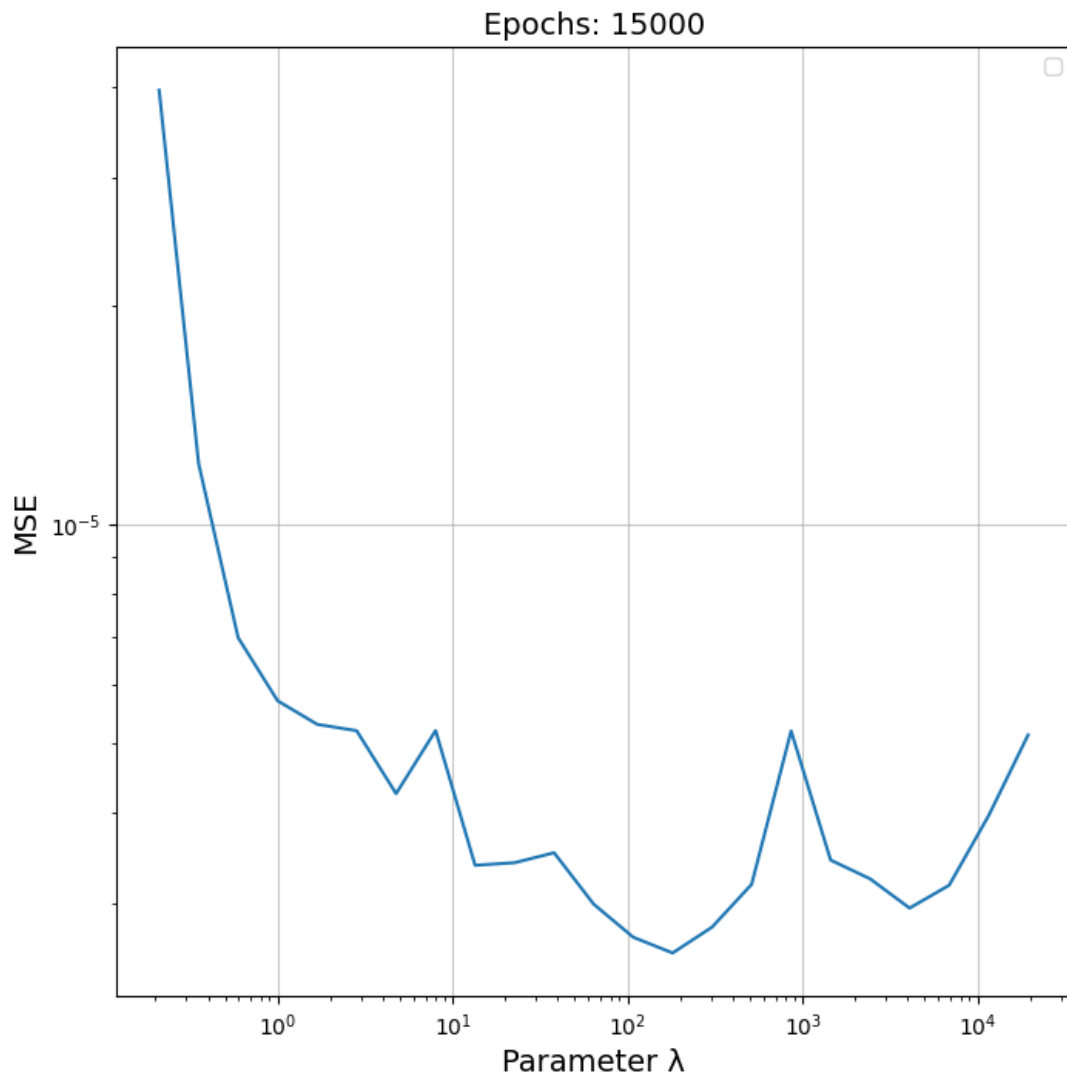


Figure 2. 15000 эпох, среднеквадратичная ошибка

Точные цифры привести, к сожалению, не приведены. Но ошибка, будет видно позже по тепловой карте, не превышает $5e - 3$.

Можно сделать, очевидный вывод о том, что меньшее значение функции потерь необязательно приводит к уменьшению ошибки аппроксимации. Очевидный в силу того, что для больших лямбда функция $L(\omega, \lambda)$ больше из-за роста множителя при второй сумме.

Различие достаточно большое, но видно, что нейросеть приближает то, что нужно.

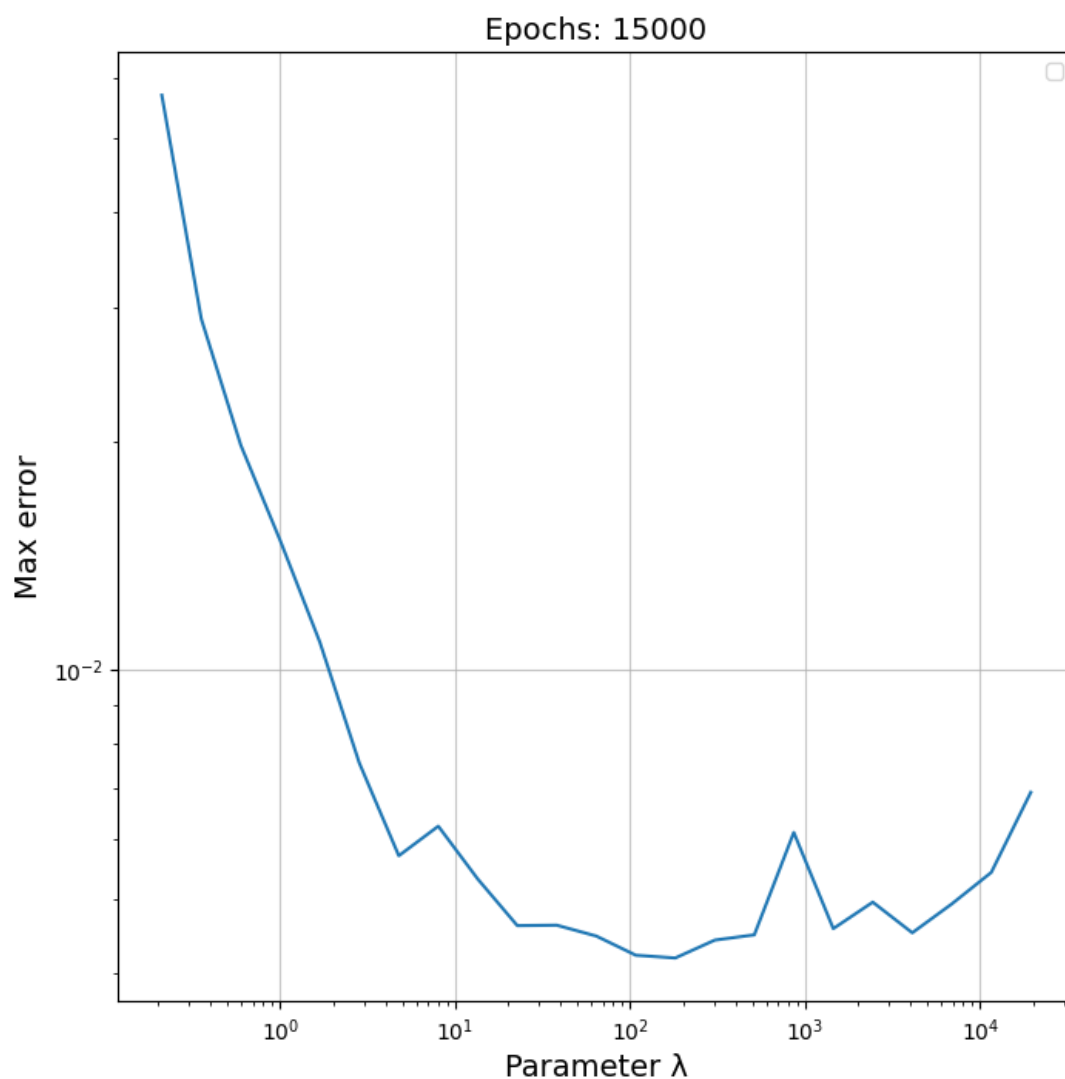


Figure 3. 15000 эпох, максимальная ошибка

Список литературы и интернет-ресурсов

- [1] https://www.tensorflow.org/api_docs — TensorFlow documentation
- [2] <https://keras.io/api/> — Keras documentation
- [3] Raissi M., Perdikaris P., Karniadakis G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations // Journal of Computational Physics. — 2019. — Т. 378. — С. 686-707.
- [4] Lagaris I.E., Likas A., Fotiadis D.I. Artificial neural networks for solving ordinary and partial differential equations // IEEE transactions on neural networks. — 1998. — Т. 9. — No. 5. — С. 987-1000.
- [5] <https://pytorch.org/docs/stable/generated/torch.optim.Adam.html> Pytorch optimizator Adam documentation

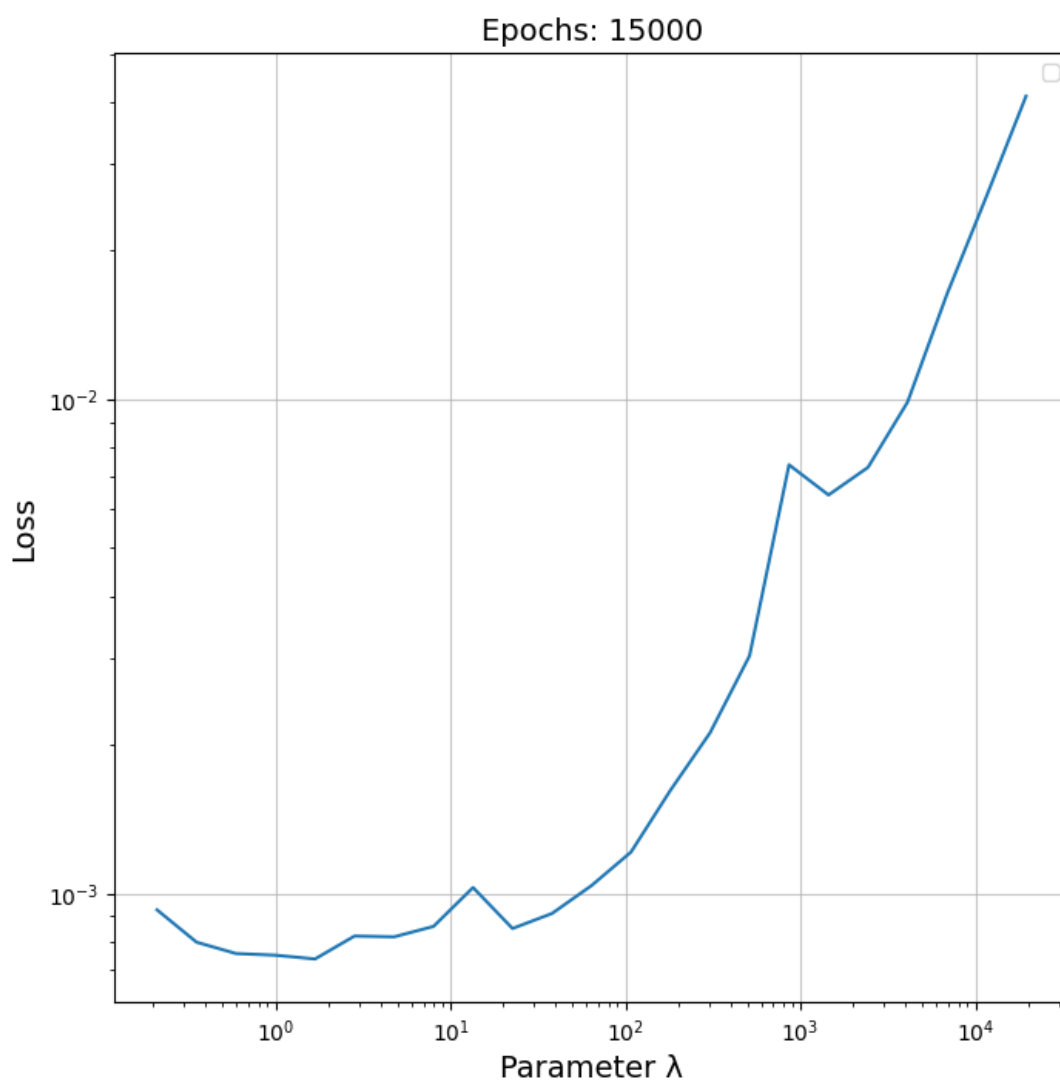


Figure 4. 15000 эпох, потери, сходимость

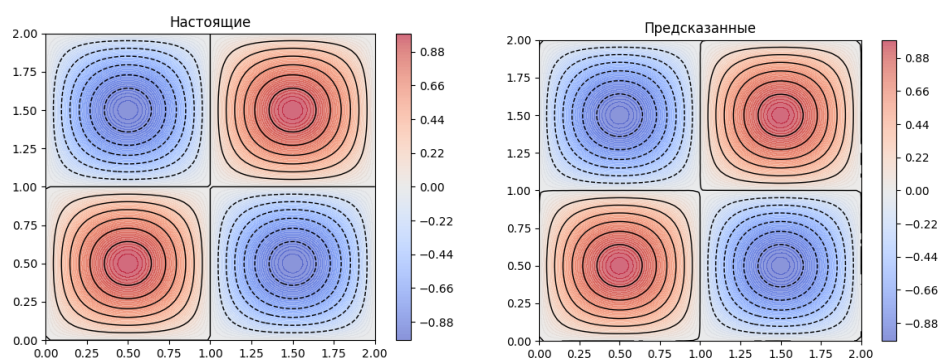


Figure 5. Тепловая карта настоящей функции и значений модели (гиперпараметр неизвестен)

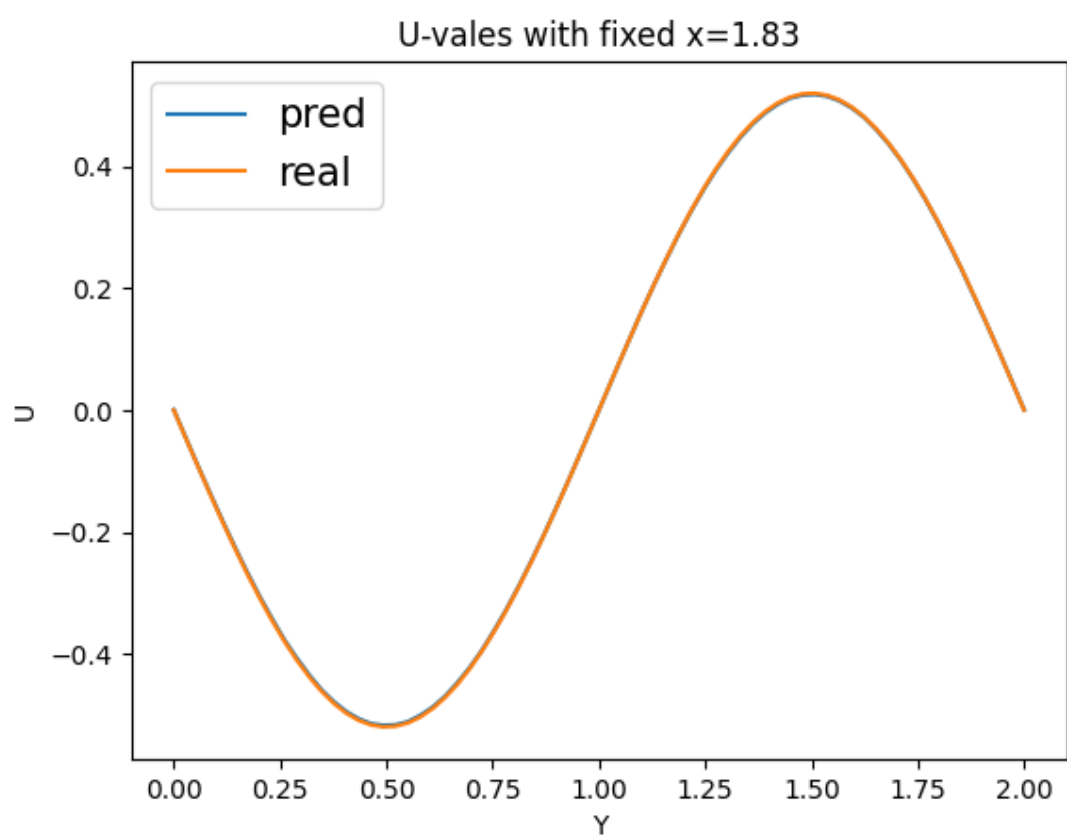


Figure 6. Разница приведенных сверху карт