МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)

ОТЧЁТ

по дисциплине «Проектная практика»
на тему «Решение начально-краевой задачи для уравнения Кортевега-де Вриза (КдВ) с использованием PINN. »

Группа Б24-265

Студент Фам Чау Донг Чинь

Руководитель работы Р.Н.Карачурин

Москва 2025

Аннотация

В данной работе применяется метод «физически информированных нейронных сетей» (PINN) для решения уравнения КдВ с известным солитонным решением. Основная цель — исследовать влияние архитектуры сети (числа скрытых слоёв и нейронов) на точность (через MSE) и вычислительную эффективность (время обучения). Результаты показывают, что сети умеренной глубины и ширины обеспечивают оптимальный баланс между точностью и вычислительными затратами, подтверждая эффективность PINN для моделирования нелинейных дифференциальных уравнений.

Отчет включает 6 разделов, 12 рисунков и 15 листингов.

В отчете искусственный интеллект используется для помощи в редактировании LaTeXи и оптимизации кода для построения графиков.

Содержание

1. Введение

Physics-Informed Neural Networks (PINN) представляют собой альтернативный подход (если рассматривать классические численные методы) к решению дифференциальных уравнений, объединяющий методы машинного обучения с фундаментальными физическими законами. Этот метод предлагает новый взгляд на решение сложных физических задач, особенно в случаях, когда традиционные численные методы сталкиваются с трудностями.

2. Задача

Уравнение КдВ:

$$\frac{\partial u}{\partial t} + 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, \quad x \in (-L, L), \quad t > 0.$$

Начальное и граничные условия:

$$\begin{cases} u(x,0) = f(x), & x \in [-L, L], \\ u(-L,t) = u(L,t) = 0, & t > 0, \\ \frac{\partial u}{\partial x}(-L,t) = 0, & t > 0. \end{cases}$$

Задача: Реализовать PINN метод решения начально-краевой задачи для уравнения КдВ. Исследовать влияние количества нейронов и скрытых слоёв на точность и эффективность решения.

3. Решение

Используемые библиотеки

Проект был реализован на языке Python 3 с использованием следующих библиотек:

- `PyTorch` — основной фреймворк для построения и обучения нейронной сети, а также для автоматического дифференцирования,
- `Matplotlib` — визуализация численных решений и сравнение с точным решением,
- `NumPy` — работа с массивами, генерация сеток и вычисление ошибки MSE,
- `JSON` — сохранение и загрузка результатов исследования архитектур.

Архитектура проекта

Архитектура нейронной сети состоит из одного входного слоя (2 нейрона — значения x и t), нескольких скрытых слоёв и одного выходного слоя (1 нейрон — предсказание $u(x,t)$).

В ходе исследования сравнивались пять архитектур:

- `[2, 20, 20, 1]`
- `[2, 50, 50, 1]`
- `[2, 100, 100, 100, 1]`
- `[2, 200, 200, 1]`
- `[2, 50, 50, 50, 50, 1]`

Функция активации во всех экспериментах была зафиксирована как `tanh`, чтобы изолировать влияние архитектуры от влияния функции активации.

Функции активации

Хотя в проекте были определены четыре функции активации (`tanh`, `sigmoid`, `softplus`, `sin`), для исследования влияния архитектуры использовалась только одна — `tanh`. Это позволило сосредоточиться исключительно на эффекте изменения числа слоёв и нейронов, не смешивая его с влиянием типа нелинейности.

Функция потерь

Функция потерь состоит из трёх компонент:

$$\mathcal{L} = \mathcal{L}_{\text{eq}} + \lambda_{\text{ic}}\mathcal{L}_{\text{ic}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}}$$

где:

- $\mathcal{L}_{\text{eq}} = \mathbb{E}\left[(u_t + 6uu_x + u_{xxx})^2\right]$ — ошибка уравнения КдВ, вычисляемая на $N_f = 10\,000$ случайных точках внутри области $[-L, L] \times [0, T]$,
- $\mathcal{L}_{\text{ic}} = \mathbb{E}\left[(u(x,0) - f(x))^2\right]$ — ошибка выполнения начального условия, где $f(x) = \frac{1}{2}\operatorname{sech}^2\left(\frac{x}{2}\right)$,
- $\mathcal{L}_{\text{bc}} = \mathbb{E}\left[u(-L,t)^2 + u(L,t)^2 + u_x(-L,t)^2\right]$ — ошибка выполнения граничных условий,
- веса: $\lambda_{\text{ic}} = \lambda_{\text{bc}} = 10$ — задают приоритет условию на границе и начальному условию над уравнением.

Производные вычисляются с помощью автоматического дифференцирования PyTorch (`torch.autograd.grad`).

Обучение

Для оптимизации параметров сети использовался алгоритм Adam с начальной скоростью обучения 10^{-3}. Каждая модель обучалась в течение 20 000 эпох. Автоматически выбиралось устройство вычислений: MPS (Apple Silicon), CUDA (NVIDIA GPU) или CPU — в зависимости от доступности.

После обучения каждая модель оценивалась по среднеквадратичной ошибке (MSE) на плотной сетке точек, и результаты сохранялись в файл `results/kdv_architecture_results.json` для последующего анализа и визуализации.

4. Результаты

В ходе выполнения работы были получены и проанализированы результаты решения начально-краевой задачи для уравнения Кортевега–де Фриза с помощью метода физически информированных нейронных сетей (PINN). В эксперименте сравнивались пять архитектур нейронных сетей различной сложности. Ниже представлены основные выводы, подтвержденные численными данными и визуальными представлениями.

Сравнение точности решений для различных архитектур

Для каждой из пяти исследованных архитектур было рассчитано значение среднеквадратичной ошибки (MSE) по сравнению с аналитическим солитонным решением. Результаты приведены в таблице 1.

Таблица 1. Сравнение архитектур нейронных сетей

Архитектура	MSE	Время обучения (с)
2-20-20-1	6.20×10^{-5}	728.5
2-50-50-1	4.17×10^{-5}	867.6
2-50-50-50-50-1	3.92×10^{-5}	776.4
2-100-100-100-1	5.54×10^{-5}	2620.2
2-200-200-1	1.22×10^{-4}	2284.0

Как видно из таблицы, наилучшую точность (наименьший MSE $= 3.92\times10^{-5}$) показала архитектура `2-50-50-50-50-1`, состоящая из четырёх скрытых слоёв по 50 нейронов каждый. Эта же модель также оказалась самой эффективной с точки зрения времени обучения — всего 776.4 секунды.

Визуальное сравнение решений

На рисунках 1–4 представлены контурные графики, сравнивающие **точное решение** (слева) и **предсказание PINN** (справа) для каждой архитектуры. Архитектура `2-50-50-50-50-1` (рис. 4) демонстрирует предсказание, практически неотличимое от точного решения, в то время как более широкая сеть

2-200-200-1 (не показана отдельно) даёт значительные искажения формы солитона.

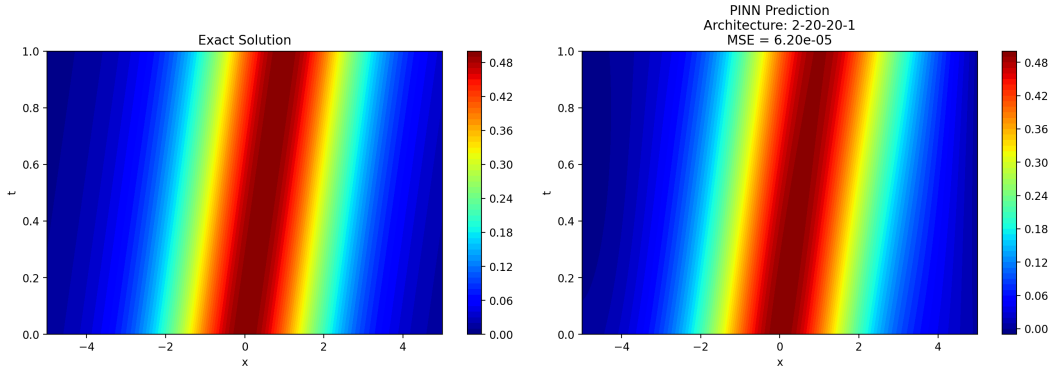


Рис. 1. Сравнение решений для архитектуры 2-20-20-1

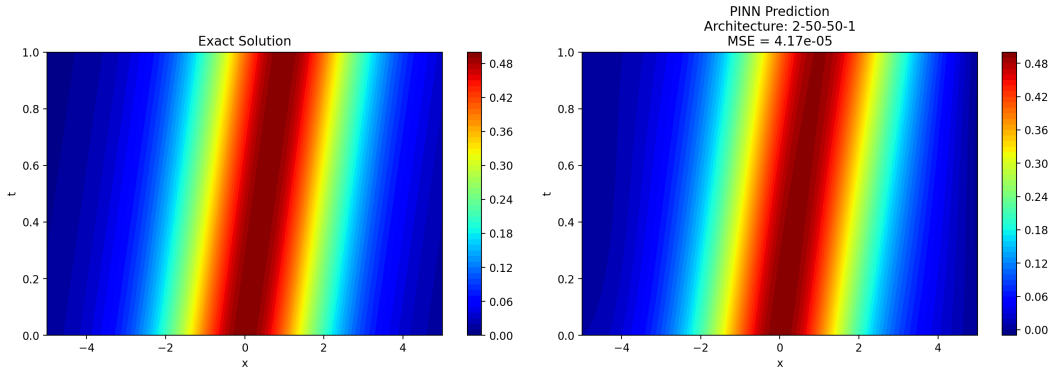


Рис. 2. Сравнение решений для архитектуры 2-50-50-1

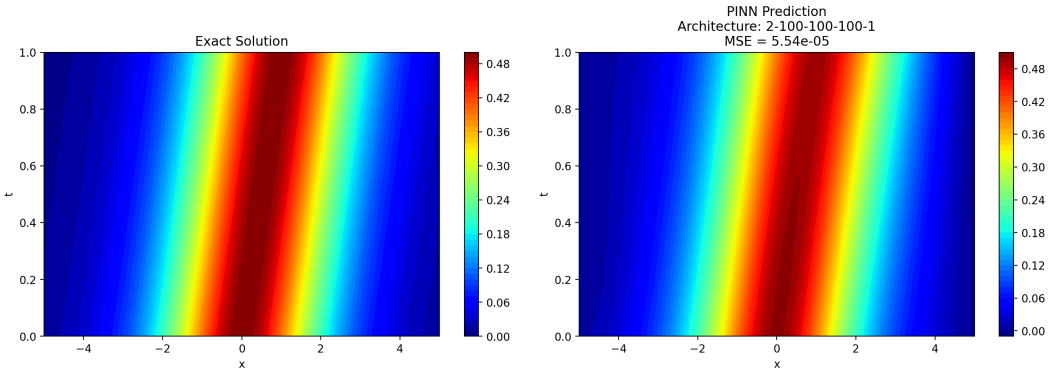


Рис. 3. Сравнение решений для архитектуры `2-100-100-100-1`

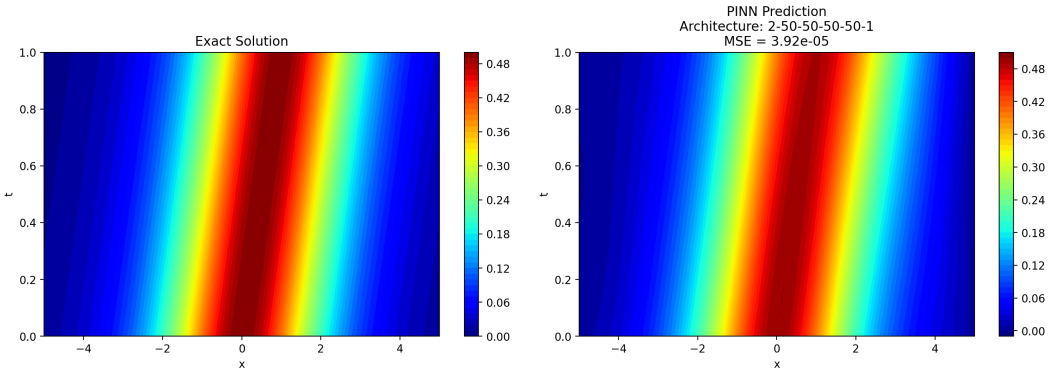
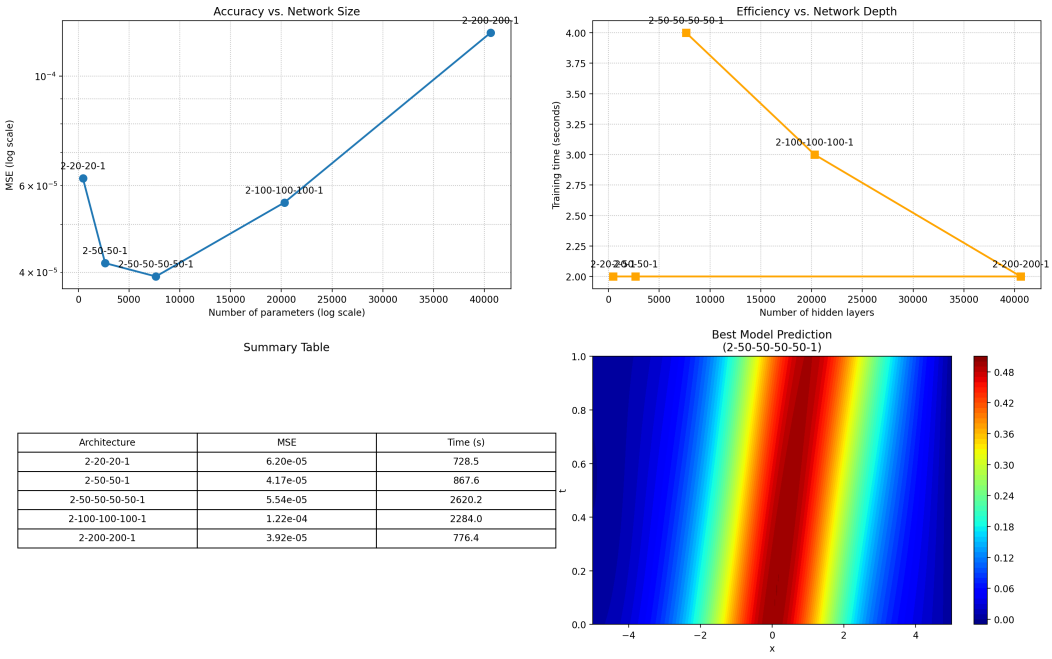


Рис. 4. Сравнение решений для архитектуры `2-50-50-50-50-1`

Анализ зависимости точности и эффективности от архитектуры

На рисунке 5 представлены графики «Точность vs. Размер сети» и «Эффективность vs. Глубина сети».

Из графика «Точность vs. Размер сети» видно, что увеличение числа параметров не гарантирует повышения точности. Наиболее точной оказалась сеть среднего размера.



Architecture	MSE	Time (s)
2-20-20-1	6.20e-05	728.5
2-50-50-1	4.17e-05	867.6
2-50-50-50-50-1	5.54e-05	2620.2
2-100-100-100-1	1.22e-04	2284.0
2-200-200-1	3.92e-05	776.4

Рис. 5. Сводный анализ результатов

5. Заключение по результатам

Проведённый анализ позволяет сделать следующие выводы:

1. Оптимальная архитектура для данной задачи — `2-50-50-50-50-1`.
2. Простое увеличение числа нейронов или слоёв не улучшает решение.
3. Метод PINN эффективно решает нелинейные УЧП при правильном выборе архитектуры.

Список литературы и интернет-ресурсов

[1] Python Software Foundation. `https://www.python.org/` — Официальный сайт языка Python.

[2] С.М. Львовский. *Набор и вёрстка в системе LaTeX, 3-е изд., испр. и доп.* — М.: МЦНМО, 2003. — Доступны исходные тексты этой книги.

[3] Lagaris I.E., Likas A., Fotiadis D.I. Artificial neural networks for solving ordinary and partial differential equations // IEEE Transactions on Neural Networks. — 1998. — Vol. 9, № 5. — P. 987–1000.

[4] 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. — Vol. 378. — P. 686–707.

[5] PyTorch documentation. `https://pytorch.org/docs/stable/index.html` — 2024.

[6] TensorFlow documentation. `https://www.tensorflow.org/api_docs` — 2024.

[7] PyKAN: Python Library for Kolmogorov-Arnold Network. `https://kindxiaoming.github.io/pykan/` — GitHub, 2024.

[8] Qwen AI Support. `https://chat.qwen.ai/` — Поддержка ИИ Qwen.