



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» **Кафедра** 806
Группа М8О-407Б-19 **Направление подготовки** 01.03.02 «Прикладная математика и
информатика»

Профиль Информатика

Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему: Решение дифференциальных уравнений в частных производных
бессеточным методом с использованием нейросетей

Автор ВКРБ: Воронов Кирилл Михайлович (_____)
Руководитель: Ревизников Дмитрий Леонидович (_____)
Консультант: — (_____
Консультант: — (_____
Рецензент: — (_____)

К защите допустить

Заведующий кафедрой № 806 Крылов Сергей Сергеевич (_____
_____ мая 2023 года

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 57 страниц, 93 рисунков, 15 использованных источников, 1 приложения.

НЕЙРОСЕТЕВОЙ МЕТОД, БЕССЕТОЧНЫЙ МЕТОД, ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ В ЧАСТНЫХ ПРОИЗВОДНЫХ

Цель работы – создание, обоснование и демонстрация работы бессеточного метода решения дифференциальных уравнений в частных производных с использованием нейросетей.

Для достижения поставленной цели были проведены исследования в области численных методов для решения дифференциальных уравнений в частных производных.

Основное содержание работы состояло в разработке бессеточного метода решения дифференциальных уравнений в частных производных.

Основными результатами работы, полученными в процессе разработки, являются нейросети, обученные решать конкретные дифференциальные уравнения в частных производных, а также различного рода графики, позволяющие оценить их эффективность.

Данные результаты разработки предназначены для использования на предприятиях, где используются классические численные методы.

Использование результатов данной работы позволяет более удобно получать численное решение дифференциальных уравнений в частных производных, без сетки.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	4
ВВЕДЕНИЕ	5
1 РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ БЕССЕТОЧНЫМ МЕТОДОМ	7
1.1 Сетка, налагаемая классическими методами. Численное дифференцирование	7
1.2 Анализ литературы	8
1.3 Цели и задачи для бессеточного метода решения дифференциальных уравнений в частных производных	8
2 НЕЙРОСЕТЕВЫЕ АЛГОРИТМЫ РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ	9
2.1 Теоретическая часть	9
2.1.1 Структура полносвязной нейронной сети	9
2.1.2 Радиально-базисные сети (нормализованные радиально-базисные сети)	12
2.1.3 Рекуррентные нейронные сети	13
2.1.4 Автоматическое дифференцирование	14
2.1.5 Обучение нейросетей. Алгоритм обратного распространения ошибки	17
2.1.6 Нейросетевой метод решения дифференциальных уравнений в частных производных	17
2.1.7 Гибридный метод решения дифференциальных уравнений в частных производных	19
2.2 Используемые технологии	20
2.3 Реализация. Структура кода	21
3 РЕЗУЛЬТАТЫ РАБОТЫ	22
3.1 Характеристика условий и места применения разработки	22
3.2 Результаты	22
3.2.1 Уравнение параболического типа	22
ЗАКЛЮЧЕНИЕ	54
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	55
ПРИЛОЖЕНИЕ А Исходный код	57

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

МАИ — Московский авиационный институт

НРБС — нормализованные радиально-базисные сети

РБС — радиально-базисные сети

РБФ — радиально-базисные функции

ВВЕДЕНИЕ

Актуальность темы данной работы связана с широким применением дифференциальных уравнений в частных производных. С их помощью составляются сложные математические модели, которые описывают многие физические явления. Например, моделирование электромагнитных полей, теплопроводности, распространения звука и т.д. Притом, их использование не ограничивается областью физики. Динамика популяций, диффузия и дрейф генов, распространение болезней из биологии, моделирование финансовых рынков из экономики, проектирование и оптимизация систем электроники, теплоснабжения и транспорта из инженерии также описываются дифференциальными уравнениями в частных производных.

Поэтому, одной из важных и непростых задач является решение этих уравнений. Самый лучший и надёжный способ — нахождение решения аналитически. Однако, в связи со сложностями процесса, не всегда получается это сделать. Поэтому, прибегают к получению решения на ЭВМ приближённо с помощью различных численных методов. Но все они завязаны на получении решения в виде сетки, что не всегда удобно и практично. Поэтому, создание бессеточного метода решения является одной из первостепенных задач в этой области.

Таким образом, выполненная работа актуальна и с теоретической, и с практической точек зрения.

Цель работы — создание, обоснование и демонстрация работы бессеточного метода решения дифференциальных уравнений в частных производных с использованием нейросетей.

Для достижения поставленной цели в работе были решены следующие задачи:

- обоснование выбора бессеточного метода решения дифференциальных уравнений в частных производных с использованием нейросетей;
- реализация и решение дифференциальных уравнений в частных производных на моделях различной архитектуры;
- оценка и сравнение результатов решения дифференциальных уравнений в частных производных на моделях различной архитектуры.

Работа основывалась на следующих инструментах и методах:

- язык программирования Python,
- библиотека PyTorch,
- полносвязные нейронные сети,
- автоматическое дифференцирование,
- радиально-базисные сети (нормализованные радиально-базисные сети),
- рекуррентные нейросети,
- гибридный метод решения дифференциальных уравнений в частных производных.

Основными результатами, полученными в работе, являются:

- нейросети, обученные решать конкретные дифференциальные уравнения в частных производных;
- различного рода графики, по которым возможно оценить погрешность моделей.

Результаты работы предназначены для использования внедрения в различного рода предприятия, где производится решение различных дифференциальных уравнений в частных производных с помощью численных методов.

Данная разработка позволяет решать дифференциальные уравнения в частных производных не опираясь на сетку.

1 РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ БЕССЕТОЧНЫМ МЕТОДОМ

1.1 Сетка, налагаемая классическими методами. Численное дифференцирование

Решение дифференциальных уравнений в частных производных является непростой задачей, так как необходимо численно посчитать производную. Для этого применяют различные методы, суть которых заключается в замене производной одной из конечно разностных схем. Рассмотрим определение производной:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h} \quad (1)$$

Если предположить, что h очень мал и убрать предел, мы получим формулу для вычисления производной функции с некоторым шагом. Производится дискретизация области. Тот же самый результат можно получить, разложив $f(x + h)$ в ряд Тейлора:

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \quad (2)$$

$$f'(x) = \frac{f(x + h) - f(x)}{h} - \frac{h}{2}f''(x) + O(h^2) \quad (3)$$

Здесь мы получаем, как базовую формулу вычисления производной, так и слагаемое ($\frac{h}{2}f''(x)$), по которому можно оценить погрешность — она составляет $O(h)$. Также отсюда по второй производной можно понять, что чем кривее будет функция, тем больше будет погрешность. Разложим теперь $f(x + h)$ и $f(x - h)$ в ряды Тейлора:

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5) \quad (4)$$

$$f(x - h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5) \quad (5)$$

Вычтем 5 из 4 и выразим $f'(x)$:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) + O(h^4) \quad (6)$$

Получаем формулу для первой производной с погрешностью $O(h^2)$. Теперь сложим 5 и 4 и выразим $f''(x)$:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{h^2}{24} f^{(4)}(x) + O(h^6) \quad (7)$$

Получаем формулу для второй производной с погрешностью $O(h^2)$.

Расписывать таким образом производные удобно в плане контроля погрешности, однако, мы сильно зависим от шага h , который приводит к дискретизации области. В результате решением дифференциальных уравнений в частных производных является сетка с заданным шагом, что не всегда удобно.

1.2 Анализ литературы

Анализ литературы показал, что особых исследований в данной области очень мало. Единственный подход, описываемый в [1], [2], [3] реализуется и совершенствуется в данной работе. Это связано с отсутствием технологий, кроме нейросетевого подхода, позволяющих полноценно численно аппроксимировать производные без шага.

1.3 Цели и задачи для бессеточного метода решения дифференциальных уравнений в частных производных

В связи с вышеописанной проблемой бессеточный метод решения дифференциальных уравнений в частных производных должен обладать следующими свойствами:

- возможность получать решение дифференциального уравнения в частных производных в любой точке заданной области,
- хорошая точность полученного решения,
- быстрое время решения,
- конкурентоспособность с классическими методами,
- лёгкая интеграция в предприятия.

2 НЕЙРОСЕТЕВЫЕ АЛГОРИТМЫ РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ

2.1 Теоретическая часть

2.1.1 Структура полносвязной нейронной сети

Математически глубокая нейронная сеть — это особый выбор композиционной функции. Простейший вид нейронной сети — односторонняя, суть которой заключается в том, что она применяет линейные и нелинейные преобразования к входным данным. Основу такой нейросети составляют перцептроны [4], [5]. Перцептрон, или как его ещё называют, нейрон — это простейшая математическая модель. Фактически, это линейная функция, которая входным данным x (число, вектор) сопоставляет некое число y используя функцию $f(x) = W \cdot x + w_0$. Нейрон представлен на рисунке 1.

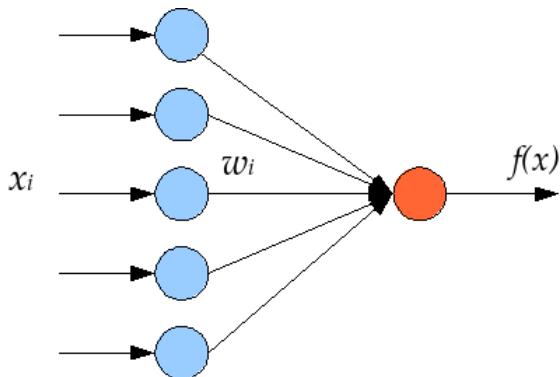


Рисунок 1 – Нейрон

Несколько нейронов стоящих в ряд образуют полносвязный слой, для обеспечения вектора на выходе (один нейрон на каждый выход).

Для того, чтобы придать вычислениям нелинейность, к выходу слоя применяются различные функции активации. Некоторые из них представлены на рисунках 2, 3 и 4.

$$ReLU(x) = \max(x, 0) \quad (8)$$

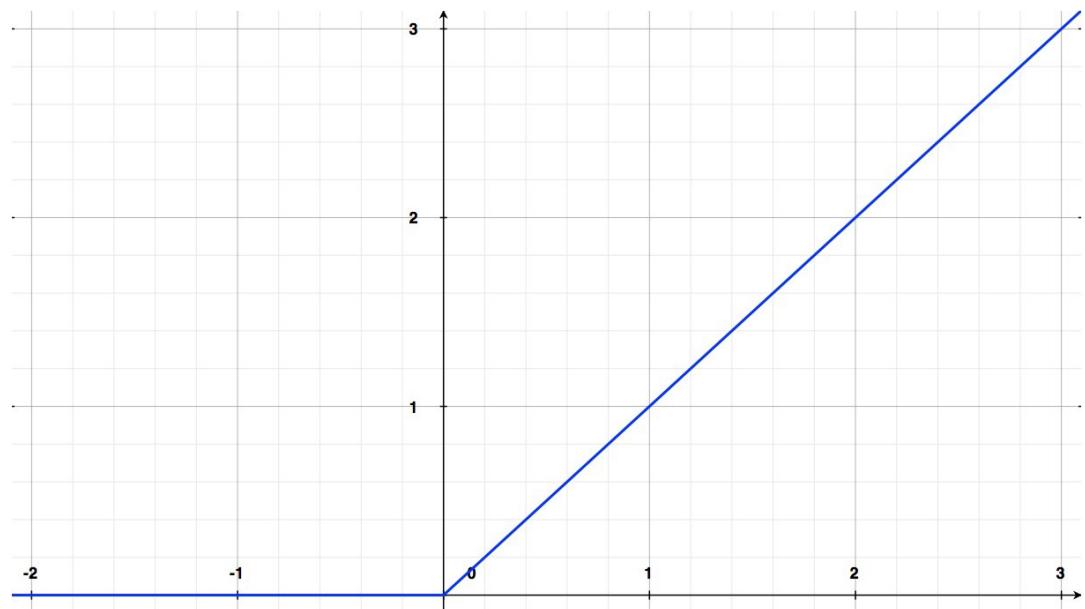


Рисунок 2 – ReLu (8)

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (9)$$

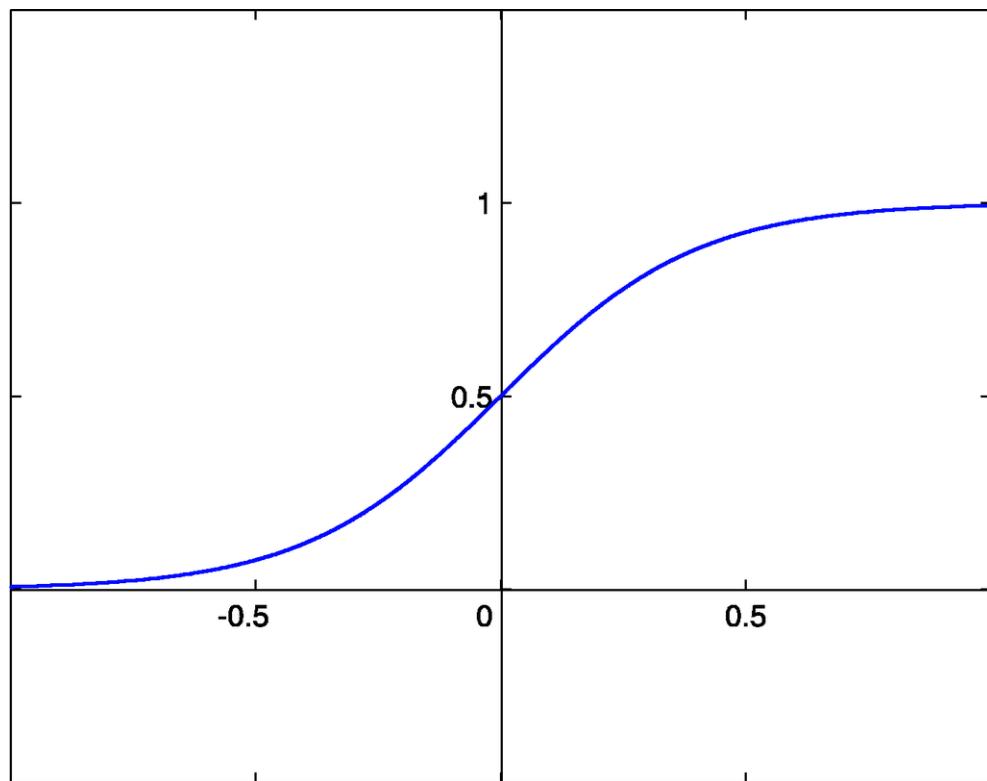


Рисунок 3 – Сигмоида (9)

$$\tanh(x) = 2\sigma(2x) - 1 \quad (10)$$

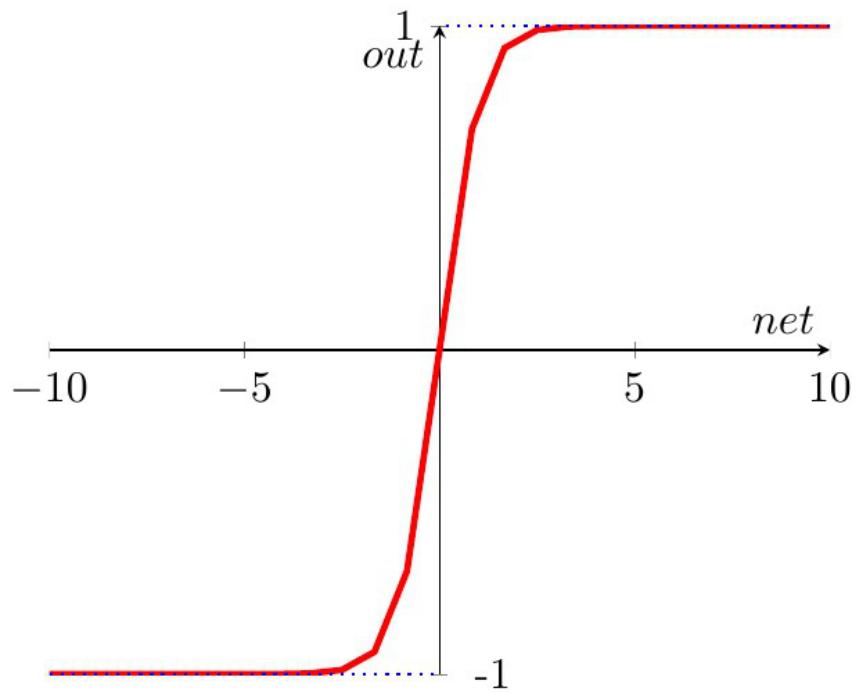


Рисунок 4 – Гиперболический тангенс (10)

Объединение слоев с функциями активаций (выходы одних идут на входы к другим) образуют многослойную (полносвязную) нейронную сеть [6]. Слои, находящиеся между входным и выходным слоями называют скрытыми. Пример многослойной сети изображён на рисунке 5.

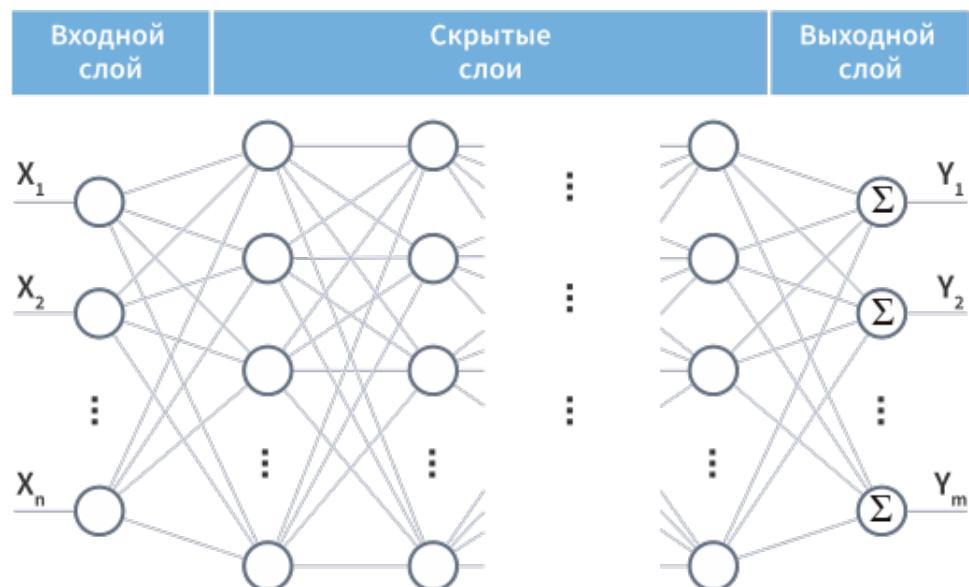


Рисунок 5 – Многослойная нейронная сеть

2.1.2 Радиально-базисные сети (нормализованные радиально-базисные сети)

Еще один вид нейронных сетей — радиально-базисные сети, суть которых заключается в том, что вместо линейной функции в слое там используется радиально базисные функции $\phi(\|x - x_c\|, \sigma)$, где x_c — центр, а σ так называемый параметр "ширины"[3]. В качестве радиально базисных функций чаще всего используются:

$$\phi(\|x - x_c\|, \sigma) = \exp\left(-\frac{\|x - x_c\|^2}{2\sigma^2}\right) \text{ — функция Гаусса,} \quad (11)$$

$$\phi(\|x - x_c\|, \sigma) = \sqrt{1 + \frac{\|x - x_c\|^2}{2\sigma^2}} \text{ — мультиквадрик,} \quad (12)$$

$$\phi(\|x - x_c\|, \sigma) = \left(1 + \frac{\|x - x_c\|^2}{2\sigma^2}\right)^{-\frac{1}{2}} \text{ — обратный мультиквадрик,} \quad (13)$$

$$\phi(\|x - x_c\|, \sigma) = \left(1 + \frac{\|x - x_c\|^2}{2\sigma^2}\right)^{-1} \text{ — обратный квадрик (функция Коши).} \quad (14)$$

Чаще всего после слоя с радиально-базисными функциями ставится полносвязный слой. Пример радиально-базисной сети представлен на рисунке 6.

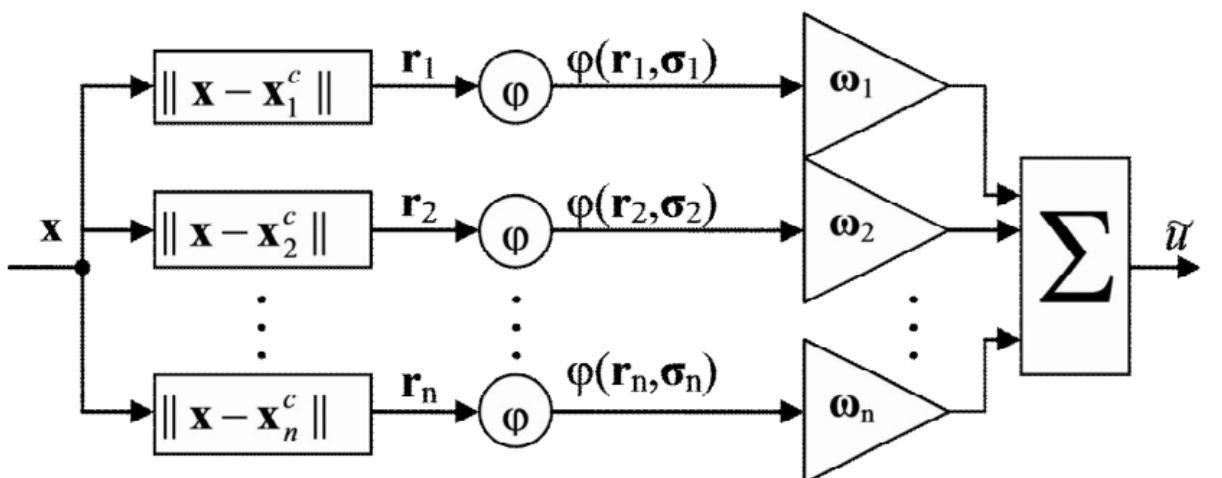


Рисунок 6 – Радиально-базисная сеть

Существует также модифицированная версия радиально-базисных сетей — нормализованные радиально-базисные сети. Отличие от обычных РБС

состоит в том, что результат полно связного слоя делится на сумму результатов РБФ (фактически, тот же полно связный слой с весами, равными единицам). Пример нормализованной радиально-базисной сети представлен на рисунке 7.

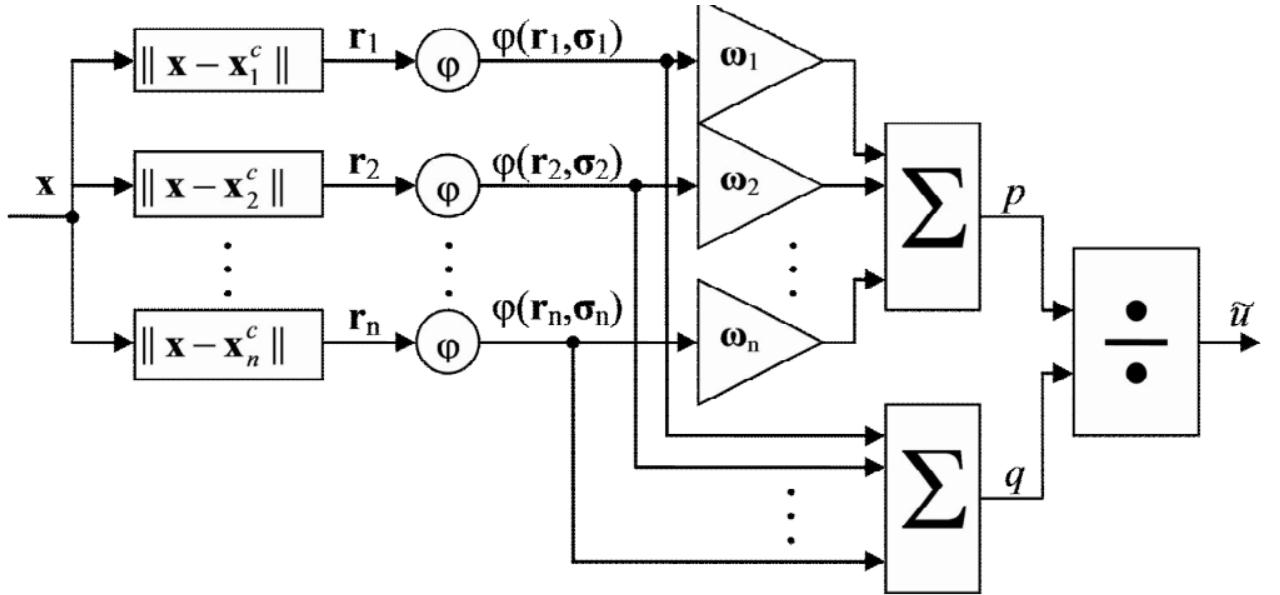


Рисунок 7 – Нормализованная радиально-базисная сеть

Нормализованные радиально-базисные сети чаще всего показывают более быструю сходимость.

2.1.3 Рекуррентные нейронные сети

Существует также вид нейронных сетей, которые используют информацию, полученную на прошлом слое. Их называют рекуррентными [7]. Обычно, их используют в случаях, когда данные последовательно зависят друг от друга. Один из видов рекуррентных сетей — нейронная сеть Элмана. Она использует выход от внутренних нейронов. Структурная сеть сети изображена на рисунке 8.

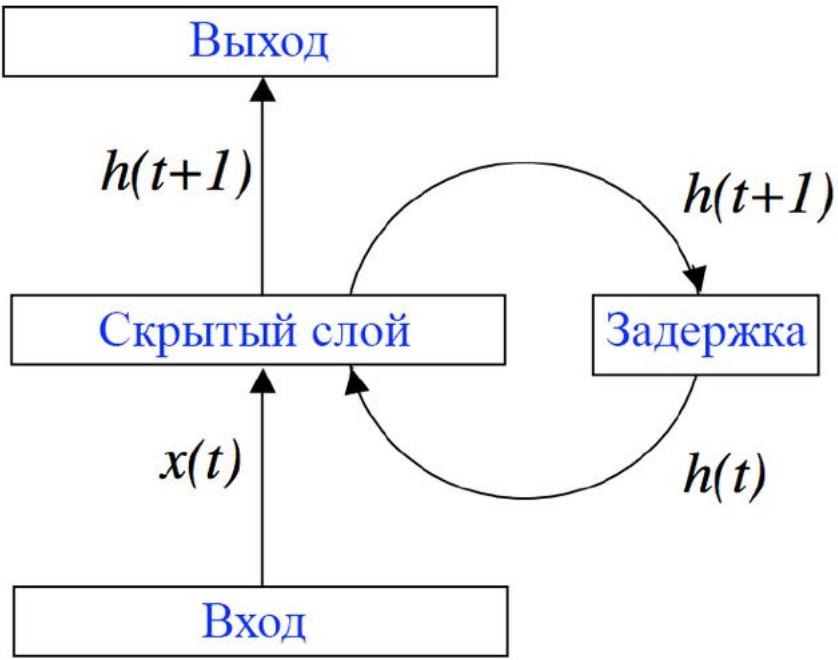


Рисунок 8 – Структура сети Элмана

Фактически, каждый нейрон имеет дополнительную матрицу, которая умножается на результат предыдущего нейрона и прибавляется к результату. В итоге результат сети выглядит следующим образом:

$$f_a(W_1 * x + W_2 * y + w_0), \quad (15)$$

где x — текущий вход,

y — результат выполнения на прошлом слое,

f_a — функция активации.

После всего это также используется полносвязный слой.

2.1.4 Автоматическое дифференцирование

В альтернативу различным методам дифференцирования, а именно нахождению аналитического решения вручную, центральной разности или другим численным приближениям, а также символльному дифференцированию, существует так называемое автоматическое дифференцирование [8]. Оно основано на свойстве $\frac{dx}{dz} = \frac{dx}{dy} \cdot \frac{dy}{dz}$ и позволяет численно вычислить производную выходов по отношению ко входам в определенной точке. Автоматическое дифференцирование бывает вперед и назад и отличается только ходом вычисления производных.

Рассмотрим работу автоматического дифференцирования на примере. Пусть нам надо найти производную функции $f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$ по x_1 в точке $(2, 5)$. Для начала построим граф вычислений, т.е. разделим вычисление функции на этапы. В каждой вершине графа будет происходить вычисление соответствующей операции. Граф вычислений функции $f(x_1, x_2)$ представлен на рисунке 9.

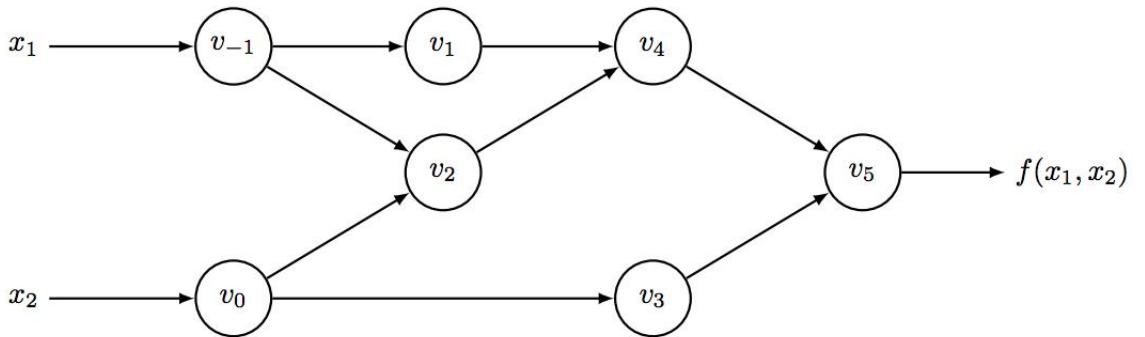


Рисунок 9 – Граф вычислений функции $f(x_1, x_2)$

Здесь $v_0 = x_2$,

$$v_{-1} = x_1,$$

$$v_1 = \ln(v_{-1}),$$

$$v_2 = v_{-1} \cdot v_0,$$

$$v_3 = \sin(v_0),$$

$$v_4 = v_1 + v_2,$$

$$v_5 = v_4 - v_3.$$

Зная, что производная x_1 по x_1 это 1, а производная x_2 по x_1 это 0, мы можем параллельно с вычислением значения функции считать значение её производной, пользуясь правилом вычисления производной сложной функции. Например, производная \dot{v}_1 — это производная $\ln(v_{-1})$, то есть $v_1 = \dot{v}_{-1} \cdot (\ln(v_{-1}))' = \dot{v}_{-1}/v_{-1}$, где \dot{v}_{-1} — производная v_{-1} , которую мы уже знаем. Продолжая таким образом идти по графу, в вершине v_5 мы получим значение производной функции в искомой точке. Полное вычисление производной по этому принципу представлено на рисунке 10.

Forward Evaluation Trace			Forward Derivative Trace		
$v_{-1} = x_1$	= 2		$\dot{v}_{-1} = \dot{x}_1$	= 1	
$v_0 = x_2$	= 5		$\dot{v}_0 = \dot{x}_2$	= 0	
$v_1 = \ln v_{-1}$	= ln 2		$\dot{v}_1 = \dot{v}_{-1}/v_{-1}$	= 1/2	
$v_2 = v_{-1} \times v_0$	= 2×5		$\dot{v}_2 = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	= $1 \times 5 + 0 \times 2$	
$v_3 = \sin v_0$	= sin 5		$\dot{v}_3 = \dot{v}_0 \times \cos v_0$	= $0 \times \cos 5$	
$v_4 = v_1 + v_2$	= $0.693 + 10$		$\dot{v}_4 = \dot{v}_1 + \dot{v}_2$	= $0.5 + 5$	
$v_5 = v_4 - v_3$	= $10.693 + 0.959$		$\dot{v}_5 = \dot{v}_4 - \dot{v}_3$	= $5.5 - 0$	
$y = v_5$	= 11.652		$\dot{y} = \dot{v}_5$	= 5.5	

Рисунок 10 – Автоматическое дифференцирование вперёд

Раскладывая вычисление функции на части, мы добиваемся несложного вычисления производной. Для автоматического дифференцирования в обратном режиме первый этап, а именно вычисление значения функции в точке, остается таким же, однако вычисление производной происходит с конца. Оно позволяет получить производную выхода ко всем входам сразу. Так как в v_5 хранится значение функции, то $\bar{v}_5 = \frac{\partial f}{\partial v_5} = 1$. Для вершины v_4 , соответственно, получаем $\bar{v}_4 = \bar{v}_5 \cdot \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \cdot \frac{\partial(v_4-v_3)}{\partial v_4} = \bar{v}_5 \cdot 1 = 1$. Продолжая таким образом на выходе значение производных функций(выхода) ко всем входам. Полный расчет представлен производных представлен на рисунке 11.

Forward Evaluation Trace			Reverse Adjoint Trace		
$v_{-1} = x_1$	= 2		$\bar{x}_1 = \bar{v}_{-1}$	= 5.5	
$v_0 = x_2$	= 5		$\bar{x}_2 = \bar{v}_0$	= 1.716	
$v_1 = \ln v_{-1}$	= ln 2		$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	= $\bar{v}_{-1} + \bar{v}_1/v_{-1} = 5.5$	
$v_2 = v_{-1} \times v_0 = 2 \times 5$			$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	= $\bar{v}_0 + \bar{v}_2 \times v_{-1} = 1.716$	
$v_3 = \sin v_0$	= sin 5		$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	= $\bar{v}_2 \times v_0 = 5$	
$v_4 = v_1 + v_2 = 0.693 + 10$			$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	= $\bar{v}_3 \times \cos v_0 = -0.284$	
$v_5 = v_4 - v_3 = 10.693 + 0.959$			$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	= $\bar{v}_4 \times 1 = 1$	
$y = v_5$	= 11.652		$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	= $\bar{v}_4 \times 1 = 1$	
			$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	= $\bar{v}_5 \times (-1) = -1$	
			$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	= $\bar{v}_5 \times 1 = 1$	
			$\bar{v}_5 = \bar{y}$	= 1	

Рисунок 11 – Автоматическое дифференцирование назад

2.1.5 Обучение нейросетей. Алгоритм обратного распространения ошибки

Для того, чтобы нейросеть выдавала на выходе правильный результат, необходимо производить коррекцию её весов. Этот процесс называется обучением. Обучение бывает с учителем, когда мы можем явно сказать, как и насколько нейросеть ошиблась, и без учителя, когда нейросеть сама пытается понять какие-то зависимости. В данной работе рассматривается обучение с учителем. Чтобы дать понять нейросети, насколько она ошиблась и как ей поправить веса, чтобы получить более точные результаты, используют так называемую функцию ошибки. Она соопоставляет выходу нейросети и правильным ответам некое число. Рассмотрим алгоритм обратного распространения ошибки, который позволяет найти градиент ошибки по весам нейронной сети. Зная градиент, мы можем уменьшить веса в сторону антиградиента, двигаясь тем самым к минимуму функции ошибки [9]. А уменьшение функции ошибки приближает ответы сети к правильным. Нахождение производных ошибки по весам происходит с помощью автоматического дифференцирования. Вычитая значение производной из весов, его домножают на некий коэффициент η , называемый скорость обучения (learning rate). То есть выглядит это следующим образом:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \quad (16)$$

где E - значение функции ошибки.

Такой алгоритм изменения весов называется градиентным спуском. Он позволяет получить локальный минимум функции, что имеет ряд недостатков. Также его результат сильно зависит от начального приближения. Существуют различные модификации данного алгоритма, одна из них — Adam [10], которая и будет использоваться в данной работе.

2.1.6 Нейросетевой метод решения дифференциальных уравнений в частных производных

Пусть дано дифференциальное уравнение вида:

$$F(x, y, z, t, u, u_t, u_x, u_z, u_y, u_{xx}, u_{xz}, u_{xy}, \dots) = 0, \quad (17)$$

с граничными условиями и начальными условиями

$$B_i(x, y, z, t, u...) = b, \quad (18)$$

где $i = \overline{0, n}$.

Суть нейросетевого подхода в том, что нейросеть, получая параметры уравнения, а именно x, y, z, t, \dots , будет выдавать значение функции u при этих параметрах [1; 2]. Здесь, в отличие от классических методов, отсутствует сетка, то есть модель полноценно аппроксимирует уравнение. Выбор архитектуры нейросети может быть различным. Функция ошибки будет состоять из двух частей L_f и L_b . L_f отвечает за удовлетворение функции исходному уравнению, а L_b — граничным условиям. Они складываются с определёнными весами w_f и w_b , которые регулируют их вклад в общую ошибку.

$$L_f = \frac{1}{|T_f|} \sum_{x \in T_f} \|F(x, \frac{\partial \hat{u}}{\partial x_1}, \dots, \frac{\partial \hat{u}}{\partial x_n}, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_n}, \dots)\|_2^2 \quad (19)$$

$$L_b = \frac{1}{|T_b|} \sum_{x \in T_b} \|B(\hat{u}, x)\|_2^2 \quad (20)$$

$$L = w_f \cdot L_f + w_b \cdot L_b. \quad (21)$$

Производные находятся с помощью автоматического дифференцирования. T_f и T_b — случайно распределенные или кластеризованные точки в области. Нейросетевой метод решения для уравнения теплопроводности $\frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2}$ с граничными условиями и начальными условиями $u(x, t) = g_D(x, t)$ и $\frac{\partial u}{\partial n}(x, t) = g_R(u, x, t)$ изображён на рисунке 12.

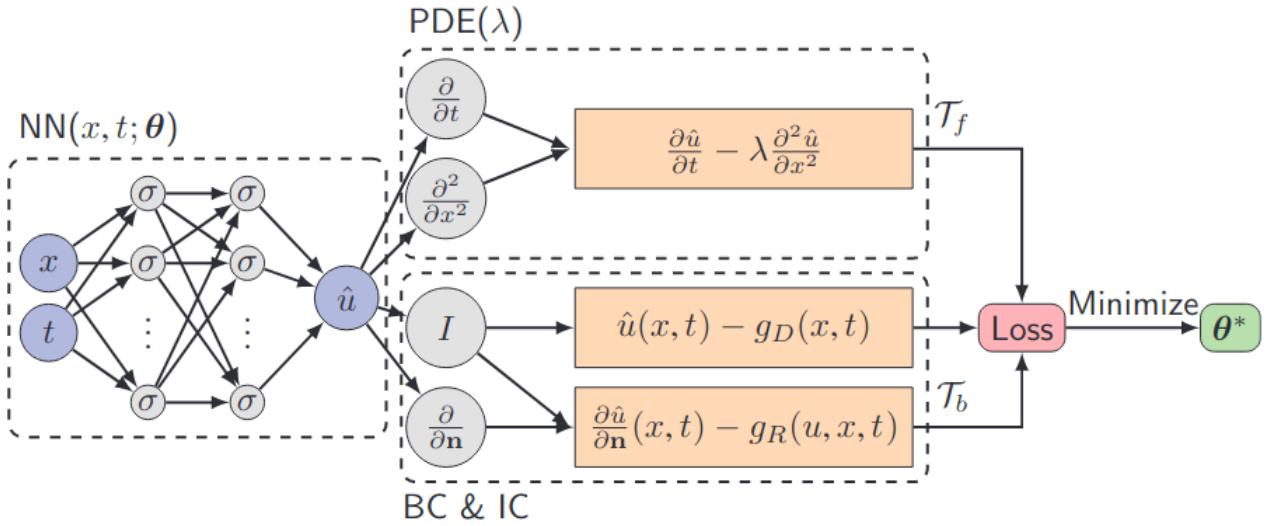


Рисунок 12 – Нейросетевой метод решения уравнения теплопроводности

На практике, так как результат зависит от случайно распределенных весов (и точек, в случае их выборки случайно), обучают несколько нейросетей и выбирают из них одну наилучшую.

2.1.7 Гибридный метод решения дифференциальных уравнений в частных производных

Суть данного метода заключается в том, что производная по времени аппроксимируется конечно-разностной схемой, а пространственные параметры — нейросетями [3]. Рассмотрим уравнение вида $u_t = A(x) + f$, с граничными условиями вида $B(u) = g$ и начальным условием $u|_{t=0} = y$. Здесь A — дифференциальный оператор, f , g , y — некоторые заданные функции. Воспользуемся явно-неявной схемой Кранка-Николсона:

$$\frac{u^{k+1} - u^k}{\tau} = A(u^{k+1}) + f^{k+1} \quad (22)$$

$$\frac{u^{k+1} - u^k}{\tau} = A(u^k) + f^k. \quad (23)$$

Поделим пополам и просуммируем, получим следующее:

$$u^{k+1} - \frac{\tau}{2}(A(u^{k+1}) + f^{k+1}) = u^k + \frac{\tau}{2}(A(u^k) + f^k). \quad (24)$$

Границные условия запишутся в виде:

$$B(u^{k+1}) = g^{k+1}. \quad (25)$$

Получаем выражения, которые будут играть роль L_f и L_b в функциях ошибки нейросетей. Сетка в этом методе накладывается только на время, и для каждого времени обучается нейросеть. Начальное условие обучается явно. Так как дифференциальные уравнения в частных производных — это эволюция процесса во времени, на каждом новом слое u^{k+1} копируется модель на прошлом, чтобы обучение происходило быстрее и точнее. В примере представлено уравнение параболического типа, однако этот метод можно распространить и на уравнения гиперболического типа.

2.2 Используемые технологии

В работе использовался язык программирования Python [11], так как он обладает некоторыми преимуществами для реализации нейросетей:

- динамическая типизация позволяет не задумываться об объявлении типов, что значительно упрощают работу с нейросетями;
- имеет огромное количество библиотек для машинного обучения и нейросетей, таких как TensorFlow, PyTorch, и др., что делает его очень гибким и мощным инструментом;
- библиотека numpy, реализованная в Python, легко позволяет выполнять непростые операции с массивами [12].

Выбранная же библиотека PyTorch [13] тоже имеет ряд преимуществ:

- удобность создания различных архитектур нейросетей. Возможность использовать как готовые слои, так и реализовывать свои;
- обладает гибкостью в работе с данными различных типов и форматов;
- позволяет легко и быстро искать производные, с помощью `torch.autograd`[14];
- позволяет использовать GPU для ускорения вычислений;
- легко перебирать параметры и структуры нейросетей.

Подход к этой задачи с использованием нейросетей объясняется их аппроксимационными свойствами. В связи со сложностями посчитать производную численно, идея её аппроксимации звучит очень заманчиво.

Несмотря на то, что идея нейросетей начала зарождаться еще в 60-е годы 20-го века, их полный потенциал открылся с недавнего времени, в связи с появлением интернета (возможностью пользования огромным количеством данных), и ростом вычислительных мощностей. И на данный момент идут попытки их интеграции в огромное количество областей.

2.3 Реализация. Структура кода

Весь код разделён на определённые разделы. В начале идёт импорт библиотек и подключение к GPU, если такое возможно. Для отображения результатов далее реализованы соответствующие функции, а именно:

- отрисовка поверхности (`sol_sur`);
- отрисовка графика функции ошибки (`loss_print`);
- отрисовка графика зависимости максимальной ошибки от количества нейронов (`loss_n_print`);
- отрисовка графика зависимости максимальной ошибки от размеров сетки для обучения (`loss_n_n_print`);
- отрисовка графиков зависимости средней арифметической ошибки по x от времени и поверхности погрешности (разница между полученным решением и аналитическим на какой-то сетке) (`error_rate`);
- отрисовка аналитического и полученного решения для определённых времён (`sol_ttt`).

Там же написаны функция начальной инициализации весов (`weights_init`) и класс конкатенации датасетов (`ConcatDataset`). Следующий раздел — реализация различных архитектур нейросетей. Здесь реализованы радиально-базисные слои и сети (а также нормализованные радиально-базисные) (`RBFLayer`, `RBF`), полносвязные (`PINN`, `PINN2`) и рекуррентные сети (`RNN_PINN_LIN2`).

Следующие разделы — работа с конкретными дифференциальными уравнениями в частных производных. Для каждого из моделей написаны функции ошибки, аналитическое решение, функции обучения на случайных точках (`random_train`) и на сетке (`data_train`). Если использовался гибридный метод, то для него также реализованы соответствующие функции. Далее для каждой модели реализованы и обучены нейросетевые модели с различными архитектурами, произведено отображение и анализ результатов.

3 РЕЗУЛЬТАТЫ РАБОТЫ

3.1 Характеристика условий и места применения разработки

Работа имеет широкий спектр применения, так как дифференциальные уравнения в частных производных описывают многие процессы. Её использование может быть на разного рода промышленных предприятиях, где используют классические методы.

3.2 Результаты

3.2.1 Уравнение параболического типа

В работе использовано следующее уравнение параболического типа с граничными и начальными условиями:

$$\frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial x^2} \quad (26)$$

$$u(0, t) = e^{-at} \quad (27)$$

$$u(1, t) = -e^{-at} \quad (28)$$

$$u(x, 0) = \cos(x). \quad (29)$$

Задача решается с коэффициентом температуропроводности $a = 0.5$.

Аналитическое решение данного уравнения представлено на рисунке 13.

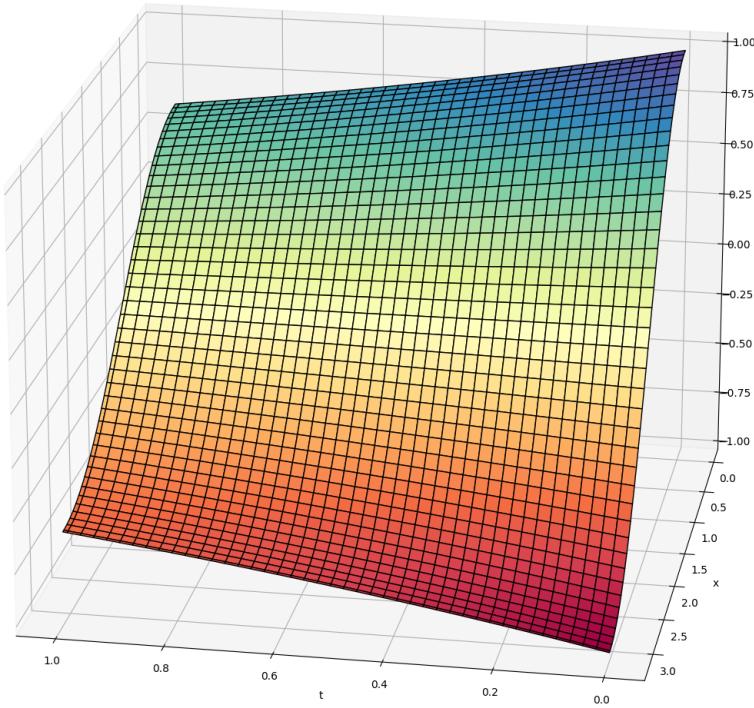


Рисунок 13 – Аналитическое решение уравнения (26)

3.2.1.1 Полносвязные модели

В начале были испробованы полносвязные модели. Эмпирическим путём было выявлено, что наилучшие результаты дает модель, состоящая из четырёх слоёв, разделённых функцией активации гиперболический тангенс. Были испробованы различные параметры, обучение на случайных точках и на сетке. В качестве оптимизатора использовался Adam.

Обучение на случайных точках было произведено на моделях с 32, 64, 128, 400 нейронами в слоях. Результаты приведены на рисунках 14 - 25.

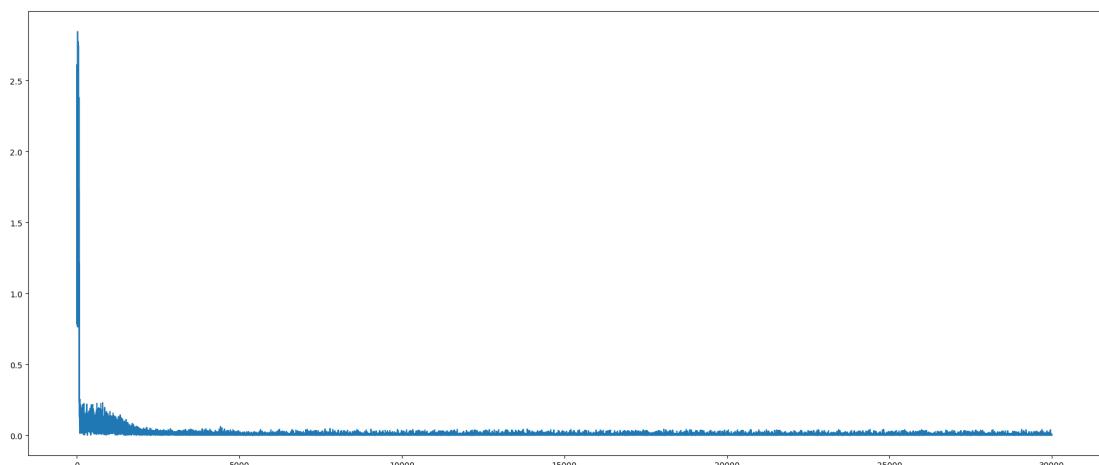


Рисунок 14 – Модель из 32-х нейронов. Функция ошибки

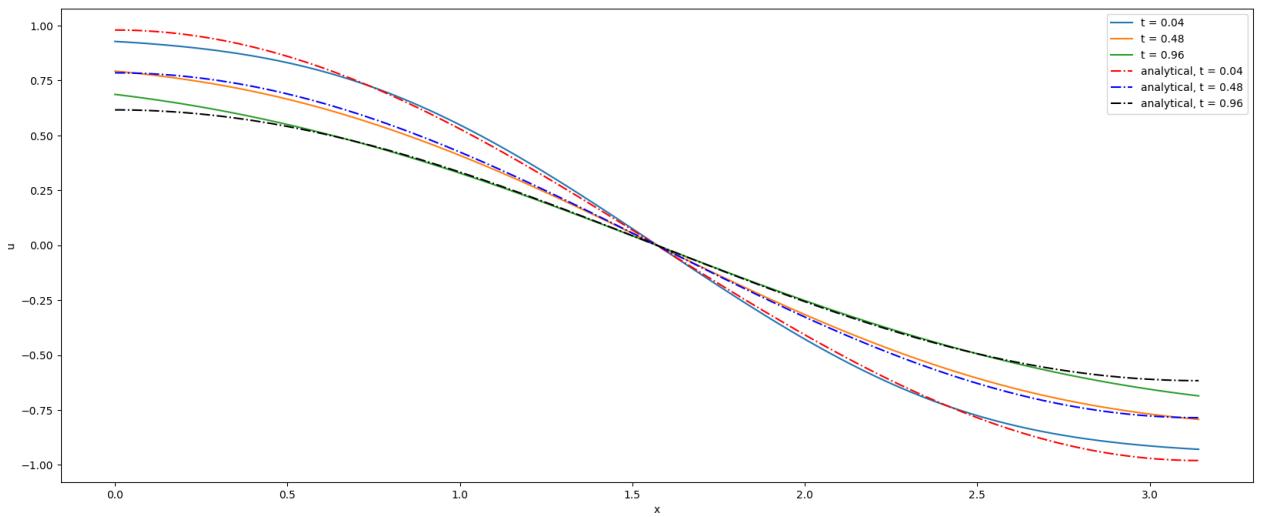


Рисунок 15 – Модель из 32-х нейронов. Ошибка на некоторых моментах времени

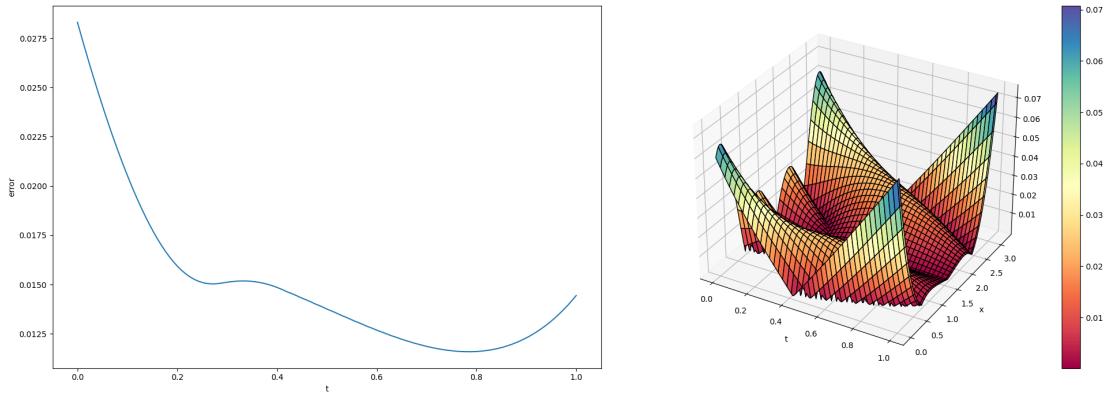


Рисунок 16 – Модель из 32-х нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

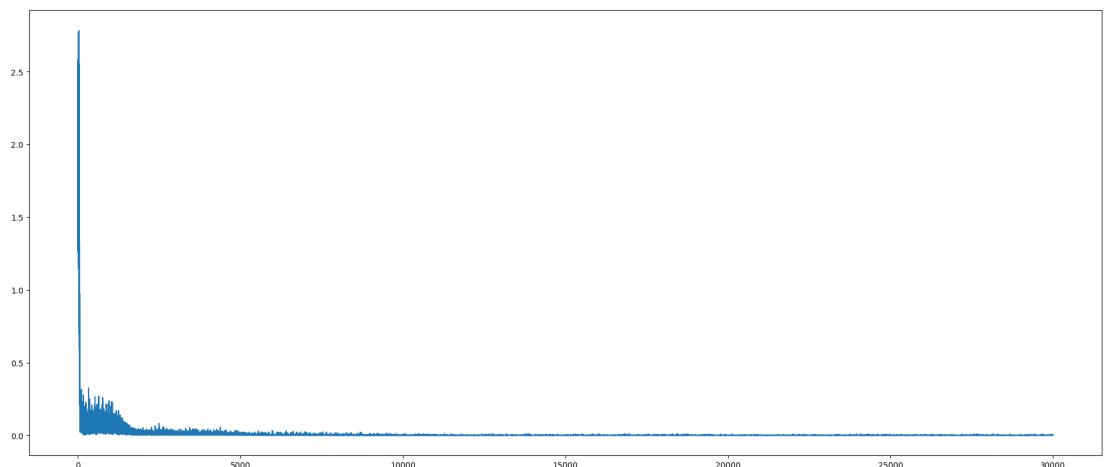


Рисунок 17 – Модель из 64-х нейронов. Функция ошибки

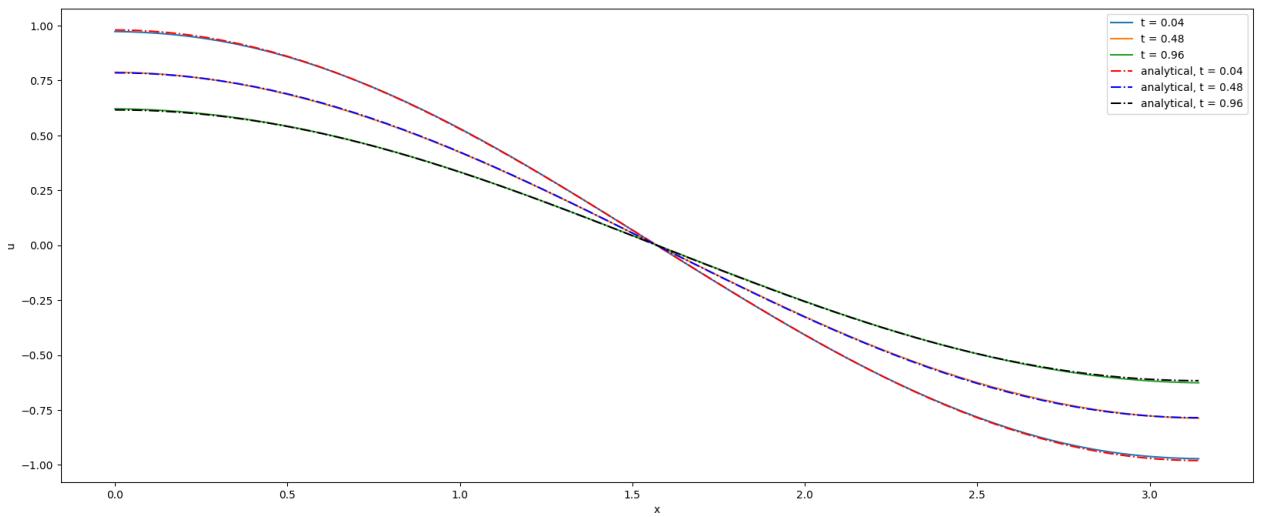


Рисунок 18 – Модель из 64-х нейронов. Ошибка на некоторых моментах времени

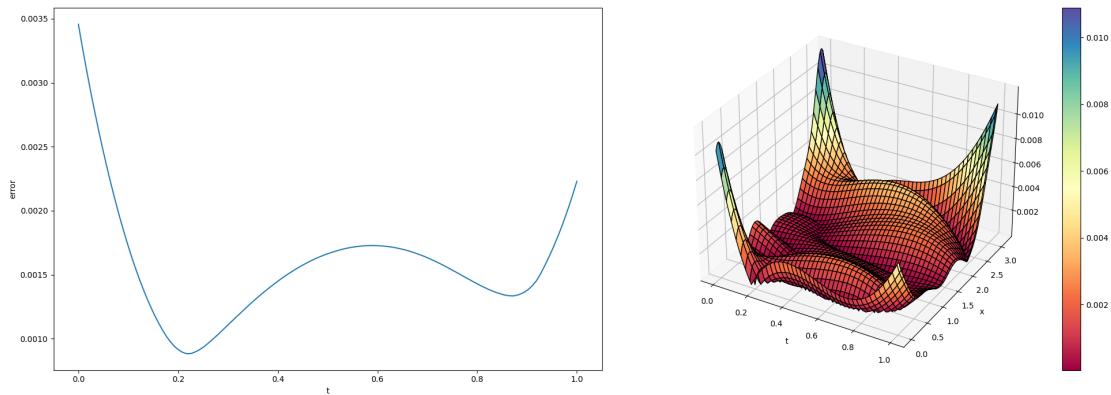


Рисунок 19 – Модель из 64-х нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

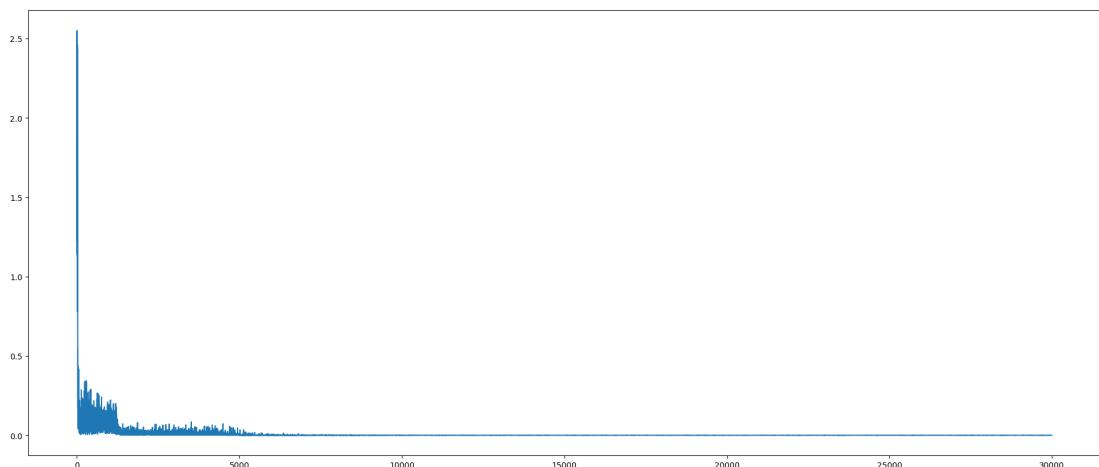


Рисунок 20 – Модель из 128-ми нейронов. Функция ошибки

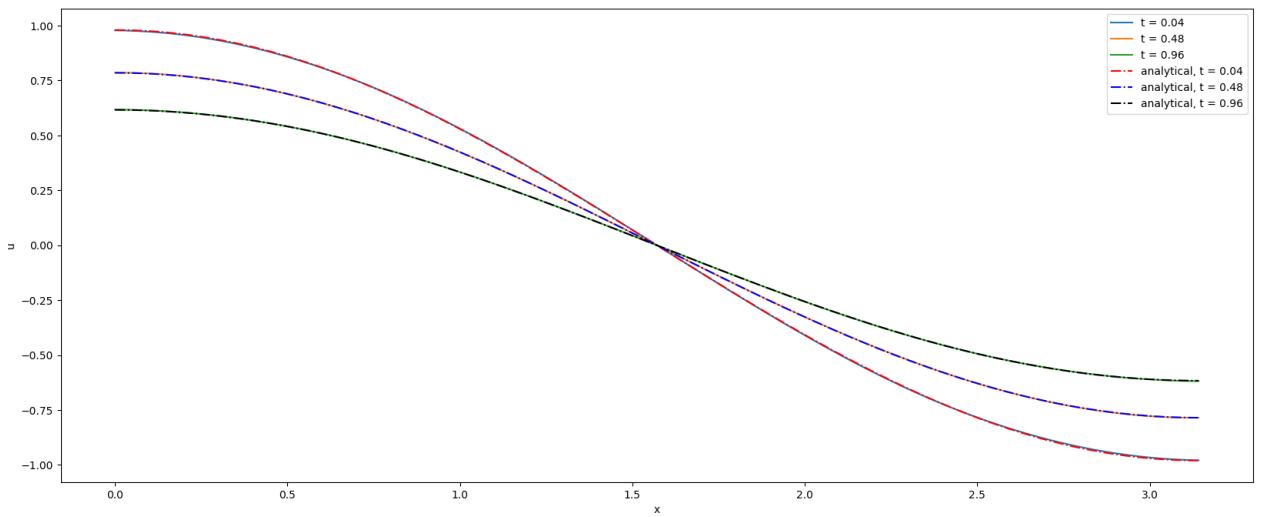


Рисунок 21 – Модель из 128-ми нейронов. Ошибка на некоторых моментах времени

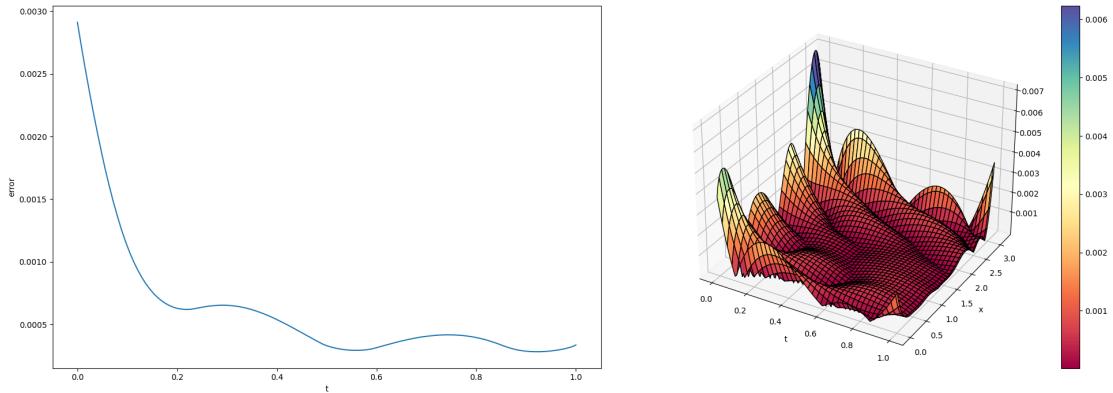


Рисунок 22 – Модель из 128-ми нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

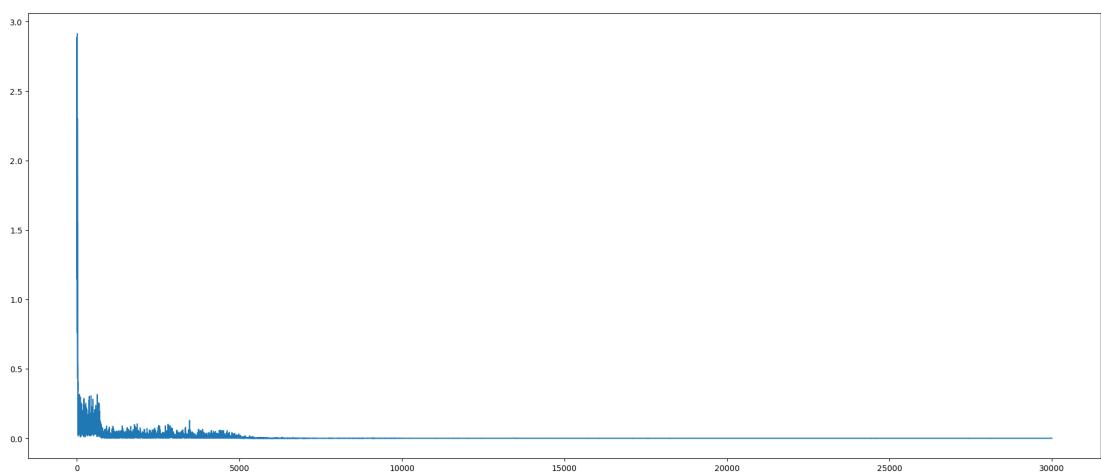


Рисунок 23 – Модель из 400-от нейронов. Функция ошибки

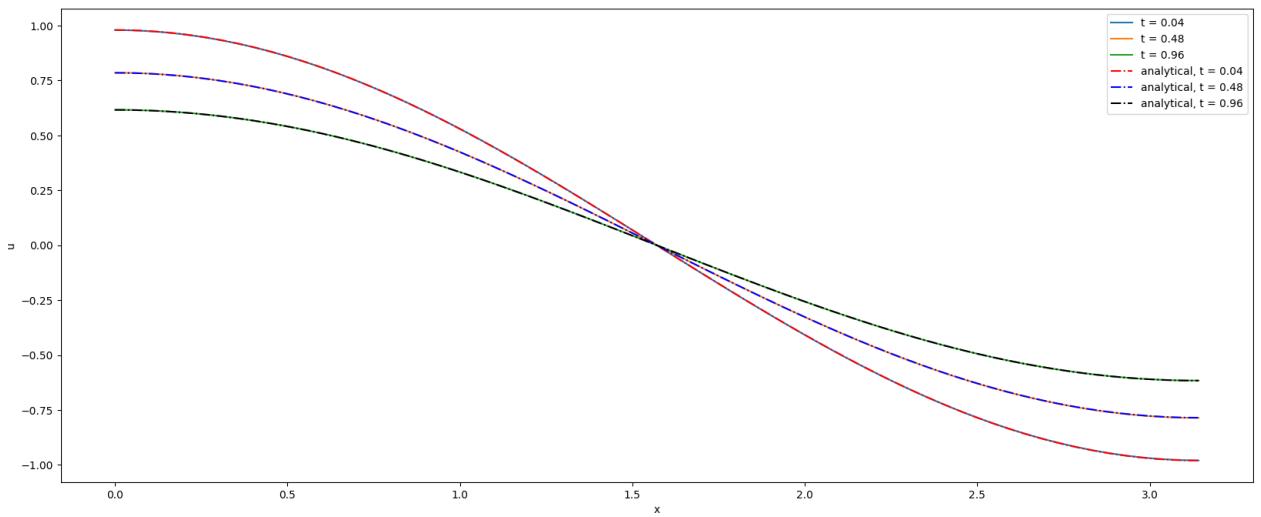


Рисунок 24 – Модель из 400-от нейронов. Ошибка на некоторых моментах времени

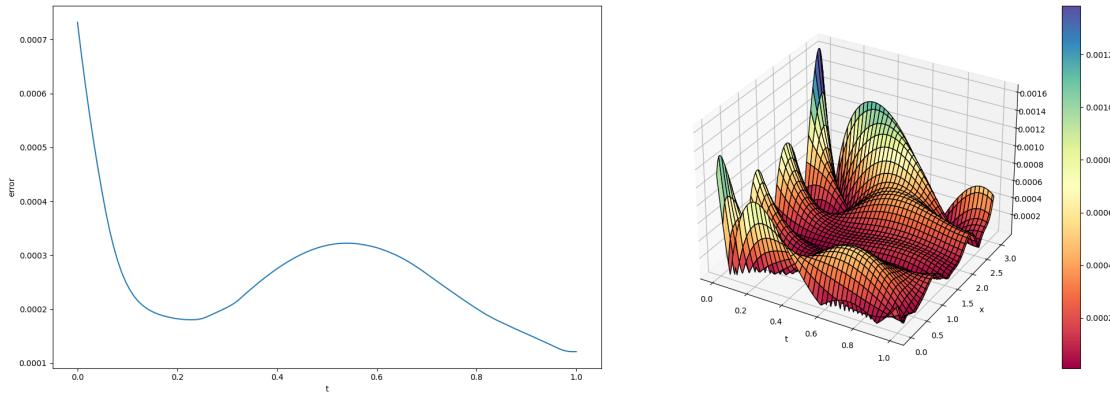


Рисунок 25 – Модель из 400-от нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

Время обучения составляет приблизительно 5 минут для всех моделей. График зависимости максимальной ошибки от количества нейронов изображён на рисунке 26.

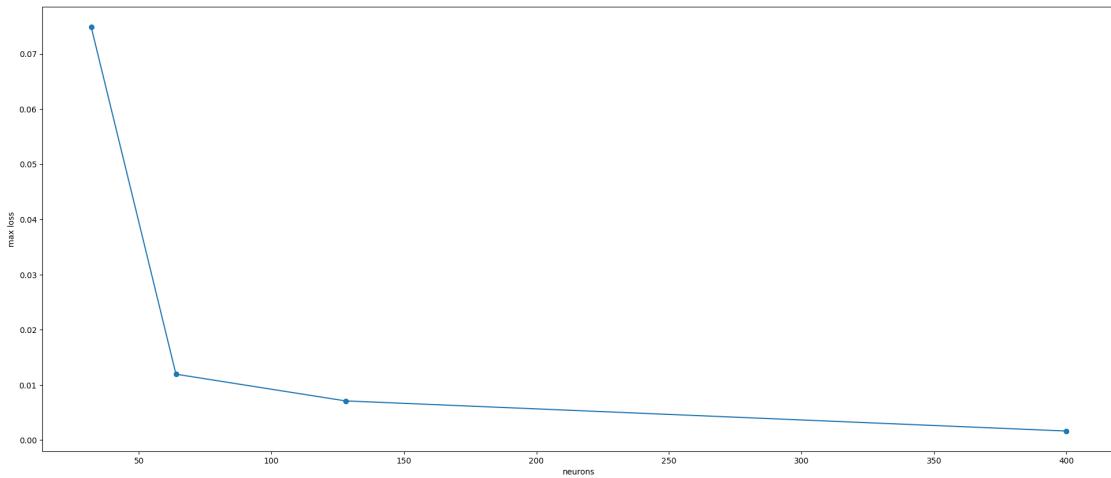


Рисунок 26 – Зависимость максимальной ошибки от количества нейронов

Как видно, при увеличении нейронов, ошибка падает, но в какой-то момент это начинает происходить медленно. Это объясняется тем, огромному количеству параметров не хватает информативности для хорошего обучения.

Обучение на сетке было произведено на модели, состоящей из 128-ми нейронов. Результаты приведены на рисунках 27 - 38.

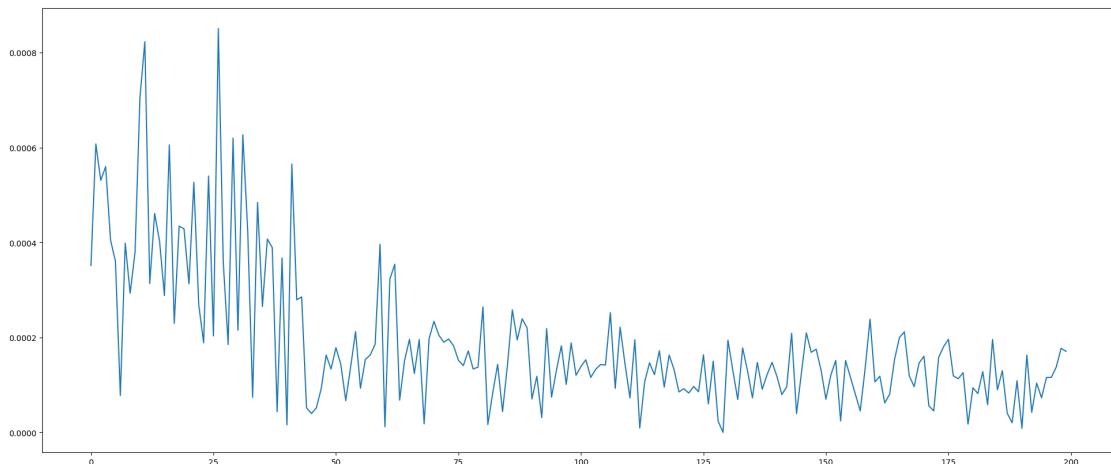


Рисунок 27 – Сетка для обучения 15x15. Функция ошибки

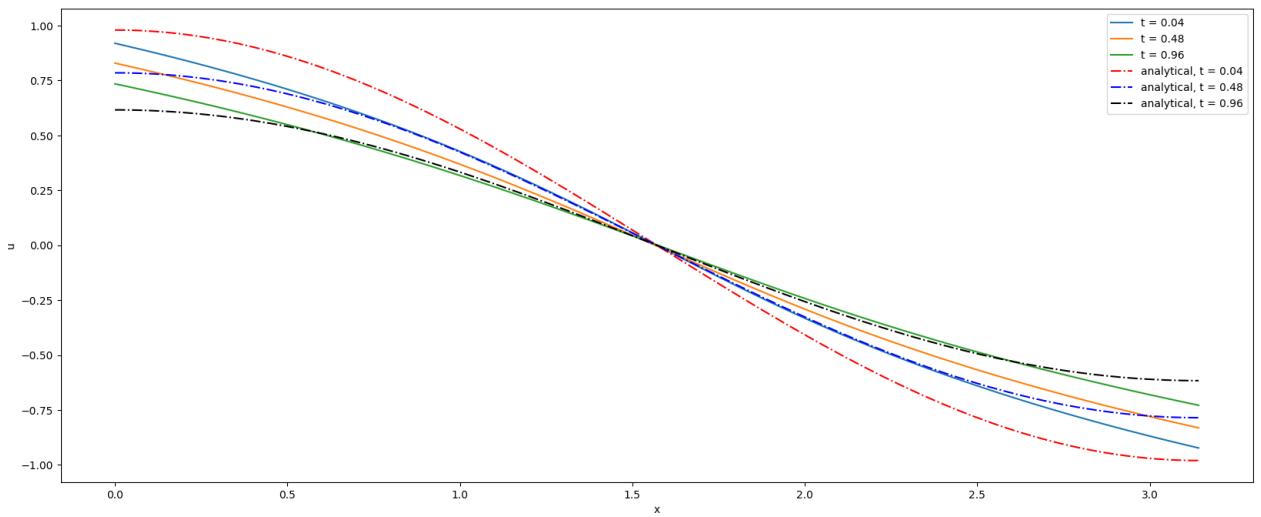


Рисунок 28 – Сетка для обучения 15x15. Ошибка на некоторых моментах времени

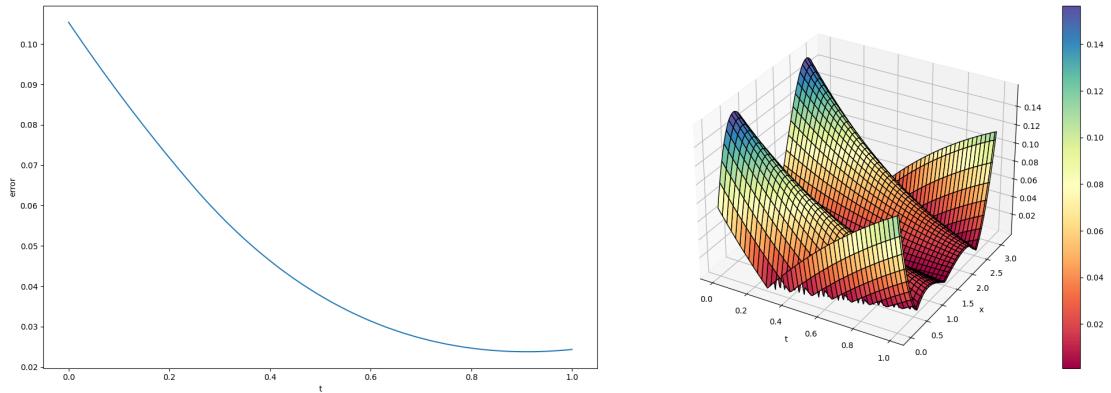


Рисунок 29 – Сетка для обучения 15x15.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

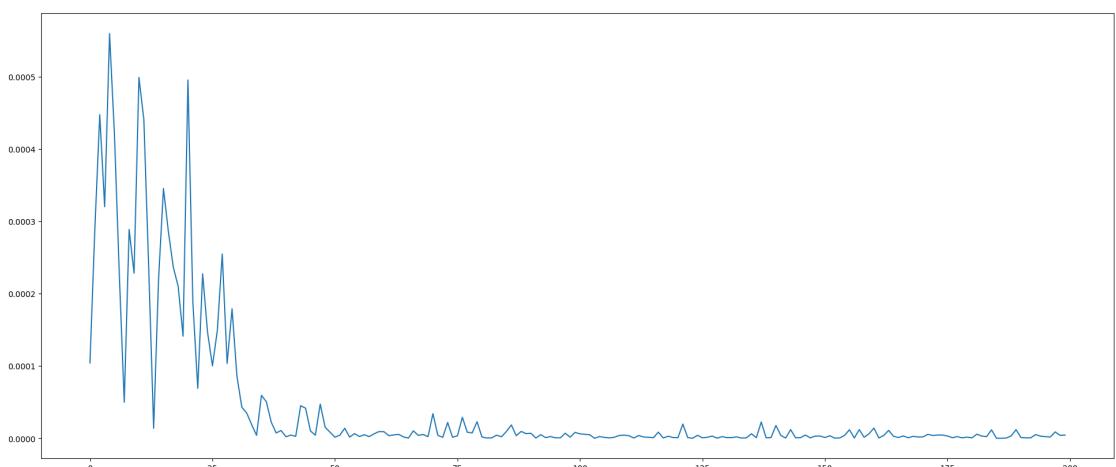


Рисунок 30 – Сетка для обучения 19x19. Функция ошибки

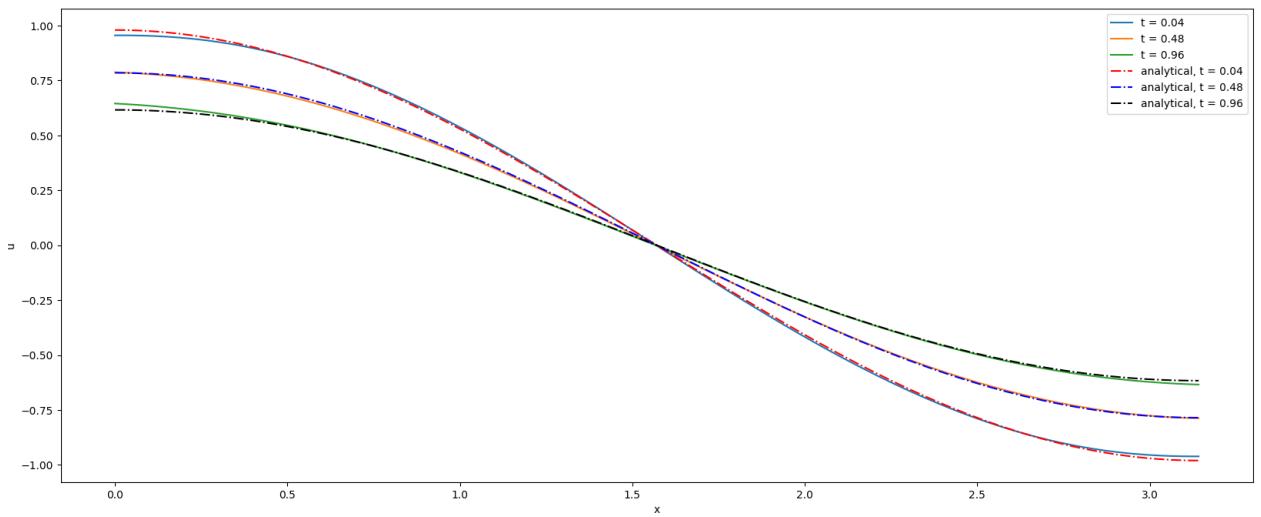


Рисунок 31 – Сетка для обучения 19x19. Ошибка на некоторых моментах времени

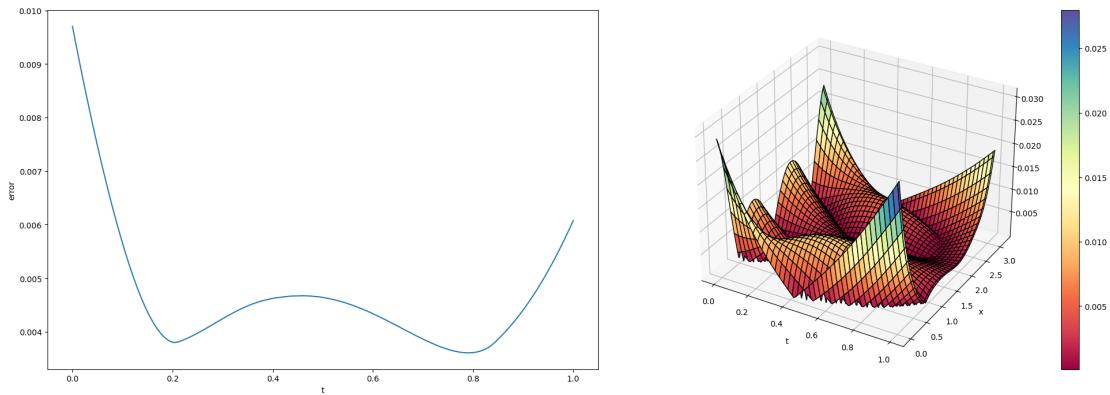


Рисунок 32 – Сетка для обучения 19x19.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

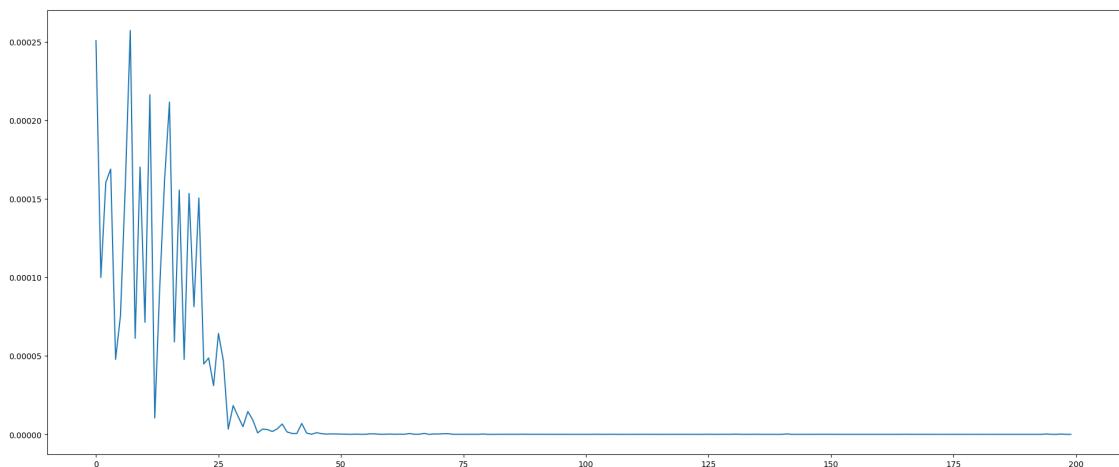


Рисунок 33 – Сетка для обучения 22x22. Функция ошибки

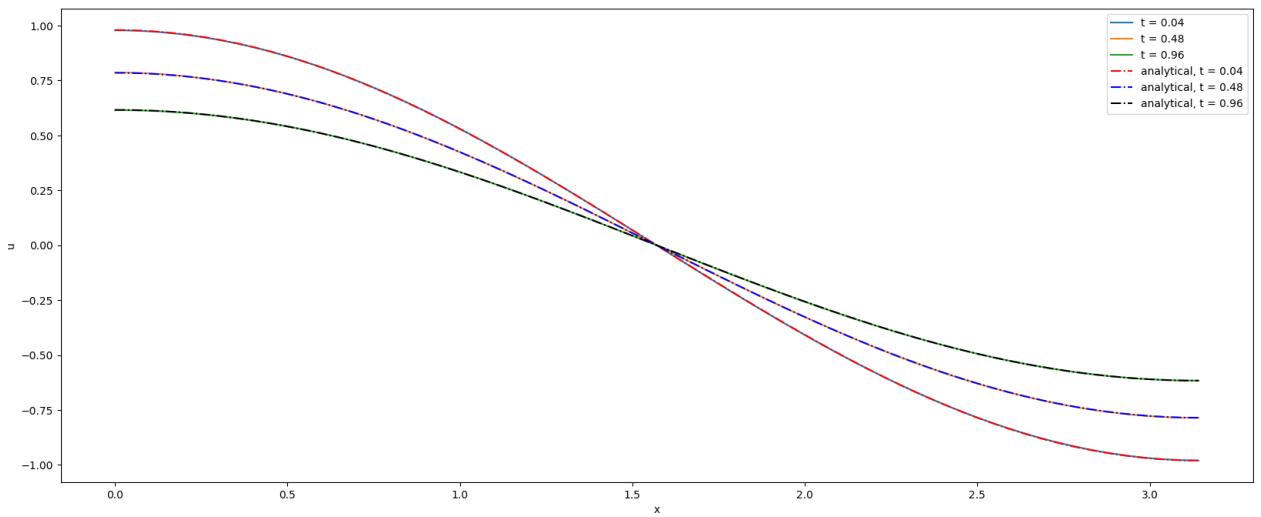


Рисунок 34 – Сетка для обучения 22x22. Ошибка на некоторых моментах времени

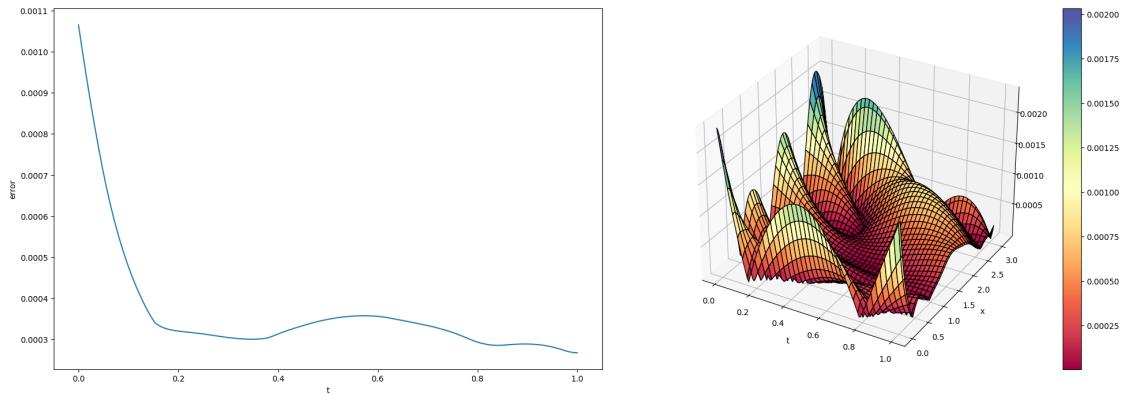


Рисунок 35 – Сетка для обучения 22x22.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

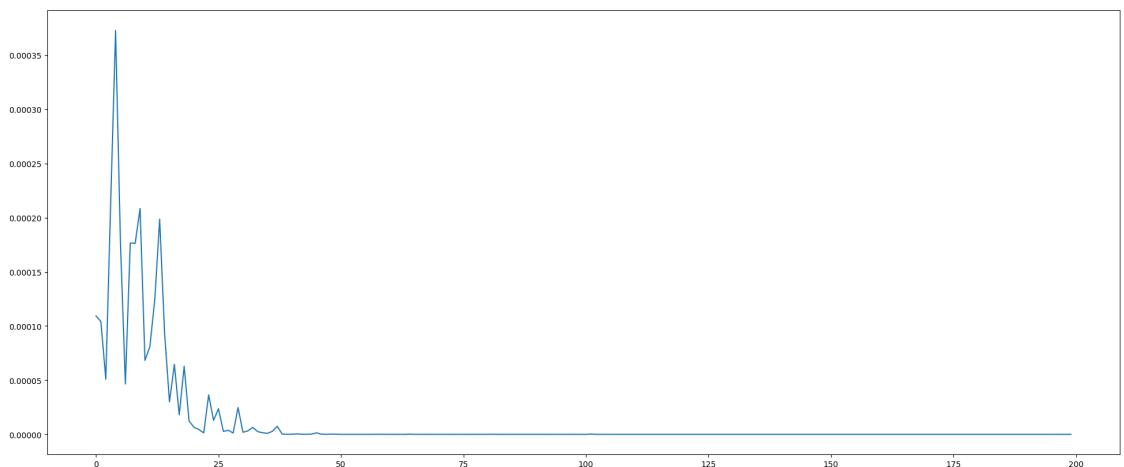


Рисунок 36 – Сетка для обучения 25x25. Функция ошибки

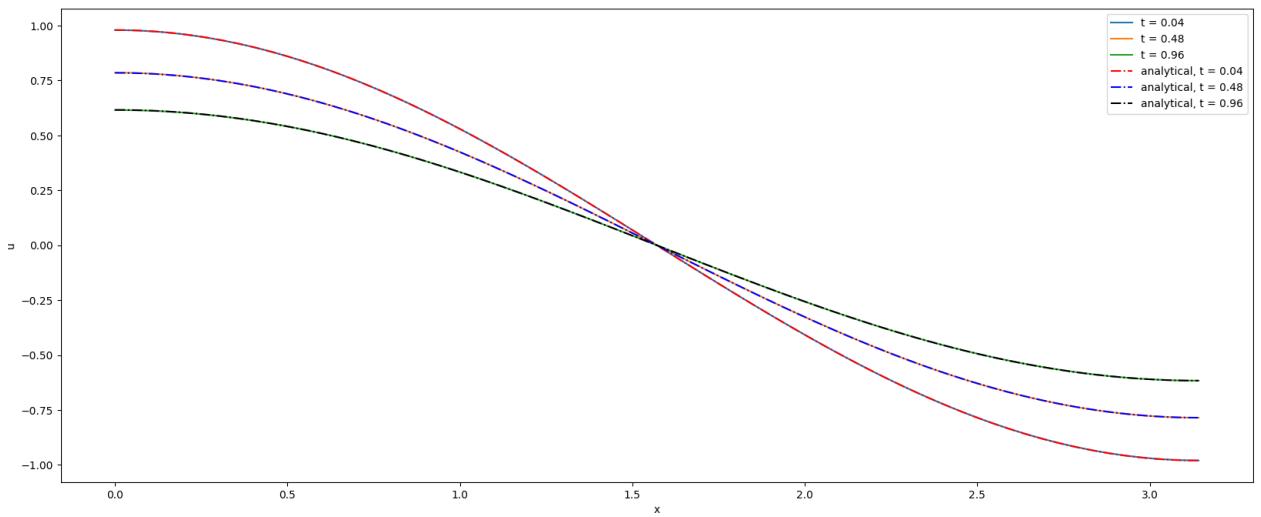


Рисунок 37 – Сетка для обучения 25×25 . Ошибка на некоторых моментах времени

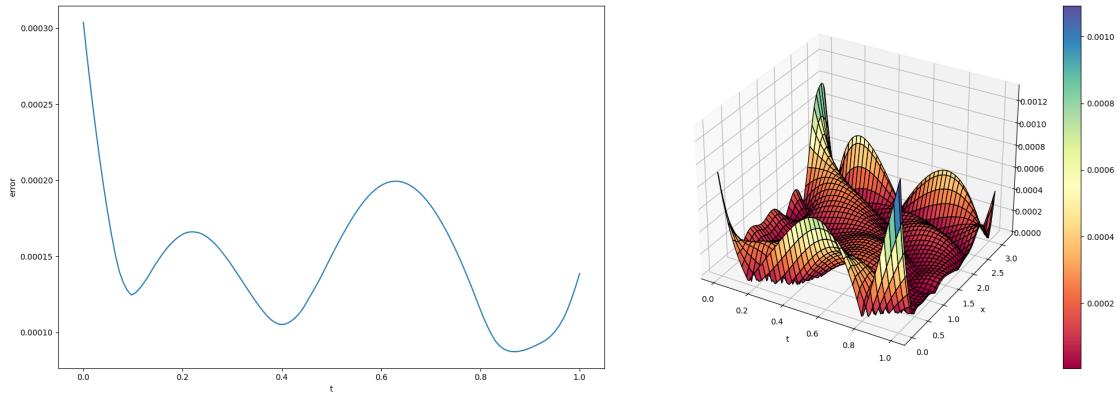


Рисунок 38 – Сетка для обучения 25×25 .
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

При обучении на сетке получаются более хорошие результаты, однако время обучения здесь намного выше — от 5 до 15 минут, в зависимости от размеров сетки. График зависимости максимальной ошибки от размера сетки для обучения изображён на рисунке 39.

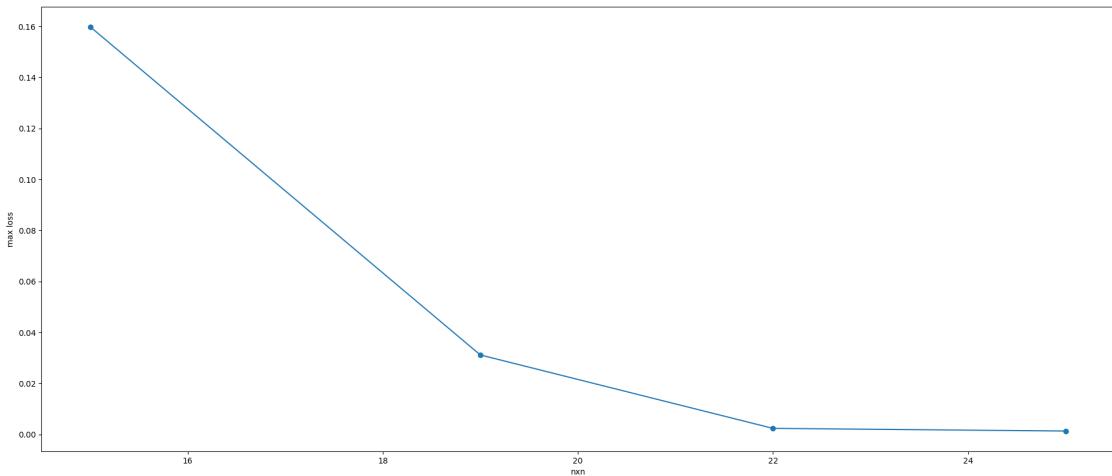


Рисунок 39 – График зависимости максимальной ошибки от размера сетки для обучения

3.2.1.2 LBFGS

Далее был опробован оптимизатор второго порядка — LBFGS [15]. Однако, он быстро впадал в локальный минимум и обучал модель очень плохо. Тогда перед тем, как использовать LBFGS, модель обучалась оптимизатором Adam. Однако, сильного выигрыша это не дало, а только заняло дополнительное время. Результаты представлены на рисунках 40 - 45.

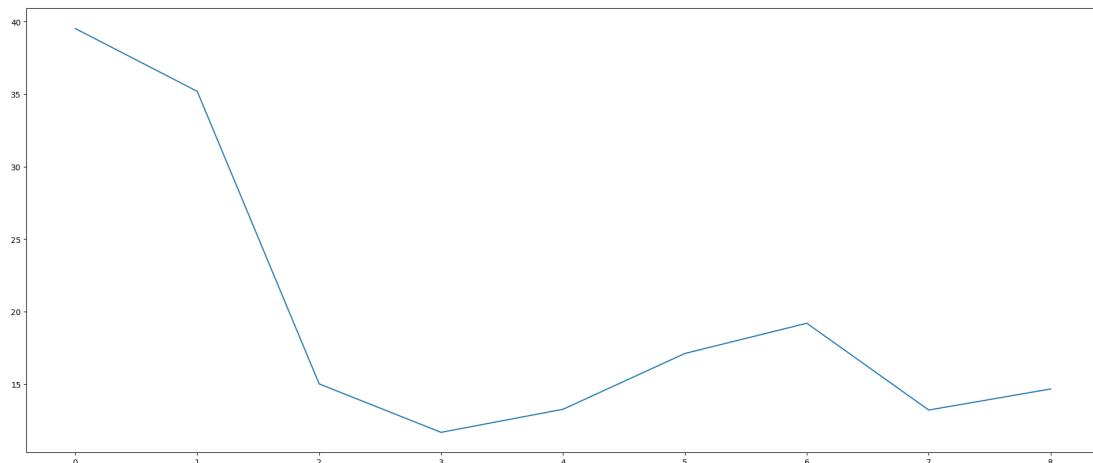


Рисунок 40 – Оптимизатор LBFGS. Функция ошибки

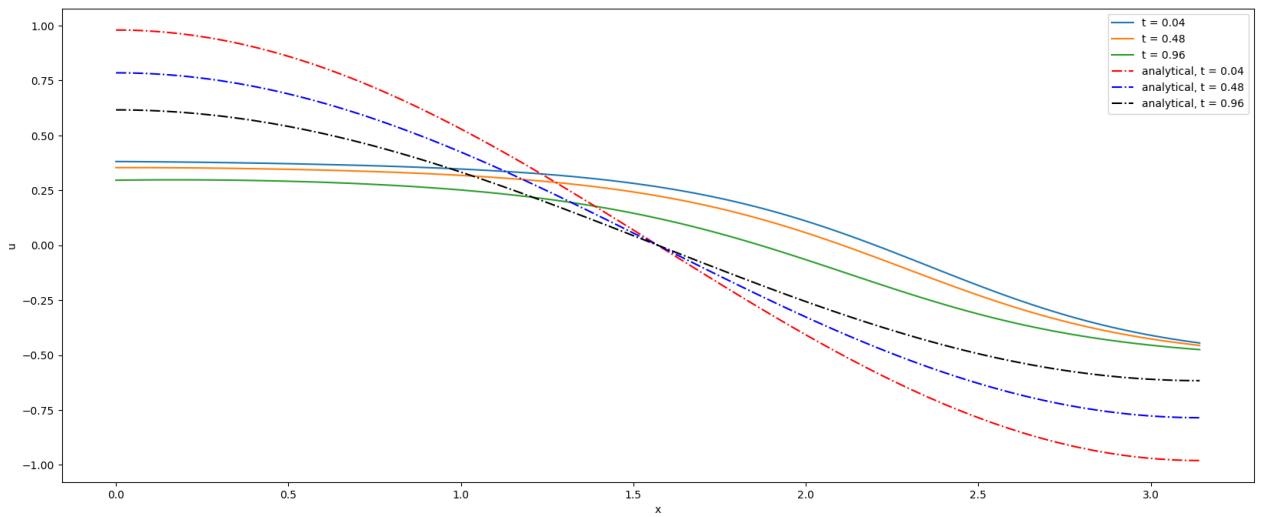


Рисунок 41 – Оптимизатор LBFGS. Ошибка на некоторых моментах времени

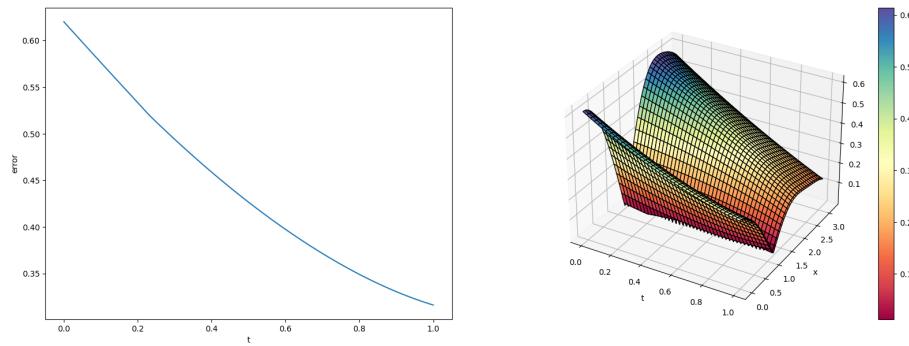


Рисунок 42 – Оптимизатор LBFGS. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

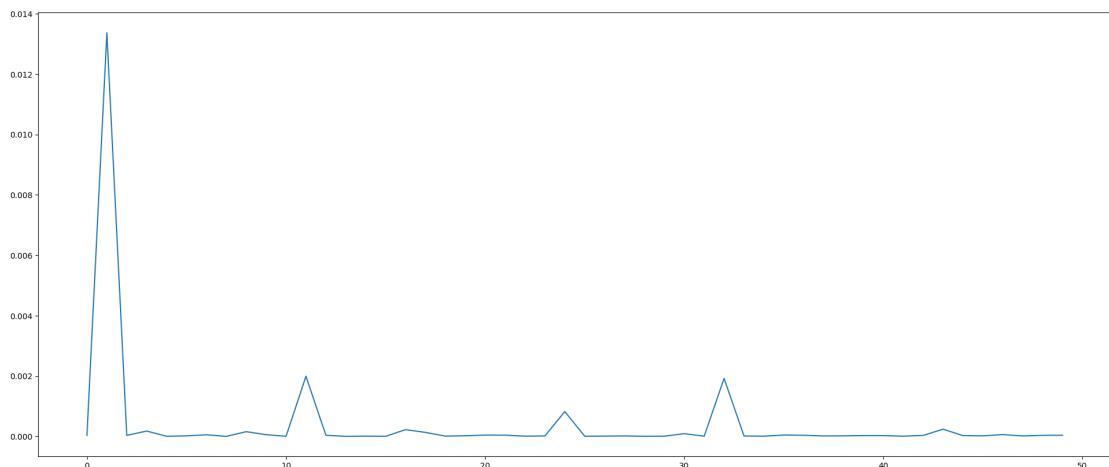


Рисунок 43 – Оптимизатор Adam + LBFGS. Функция ошибки

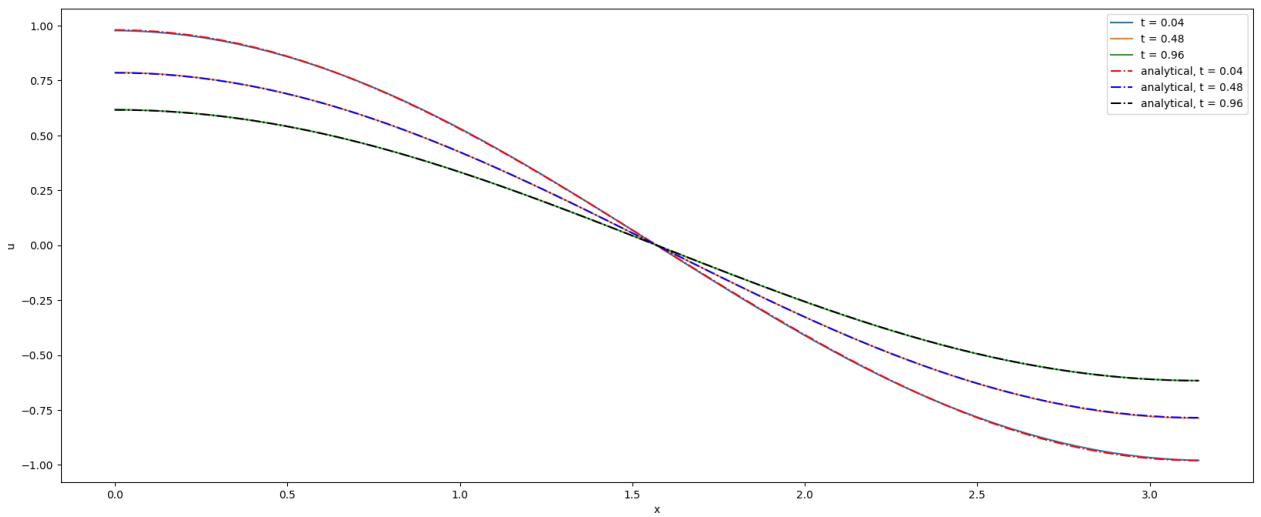


Рисунок 44 – Оптимизатор Adam + LBFGS. Ошибка на некоторых моментах времени

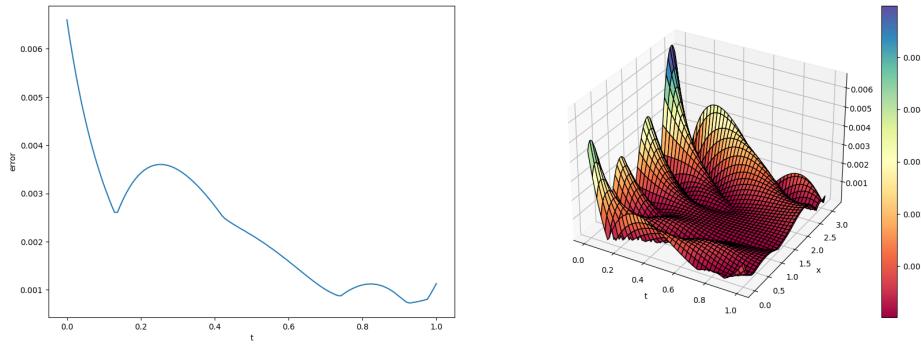


Рисунок 45 – Оптимизатор Adam + LBFGS.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

3.2.1.3 Рекуррентная сеть

Тогда было выдвинуто предположение, что хорошо себя покажут рекуррентные сети, так как налицо эволюция во времени. Однако, они способны запоминать 10-15 токенов, в представленной задаче точек намного больше. Поэтому результаты были тоже не впечатляющими. Результаты модели из 128 нейронов и 3-х слоев изображены на рисунках 46 - 48.

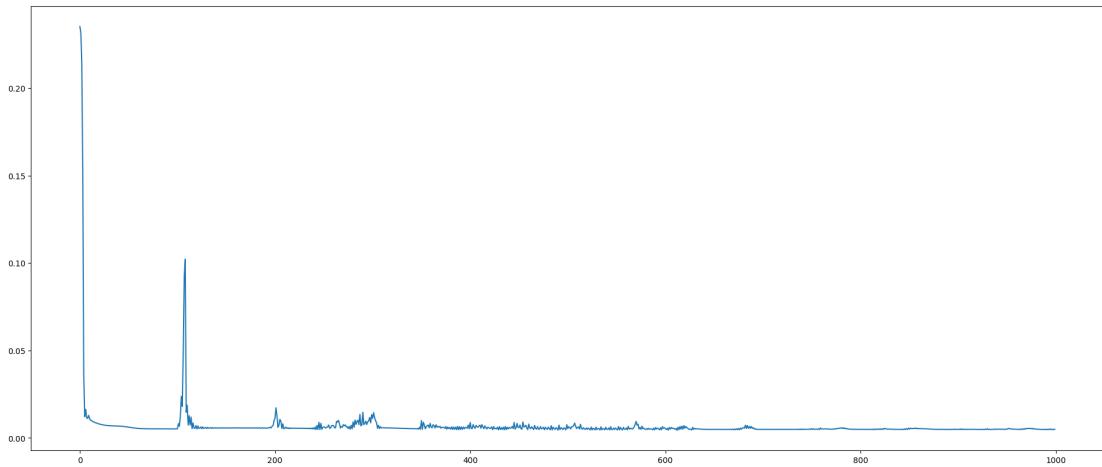


Рисунок 46 – Рекуррентная сеть. Функция ошибки

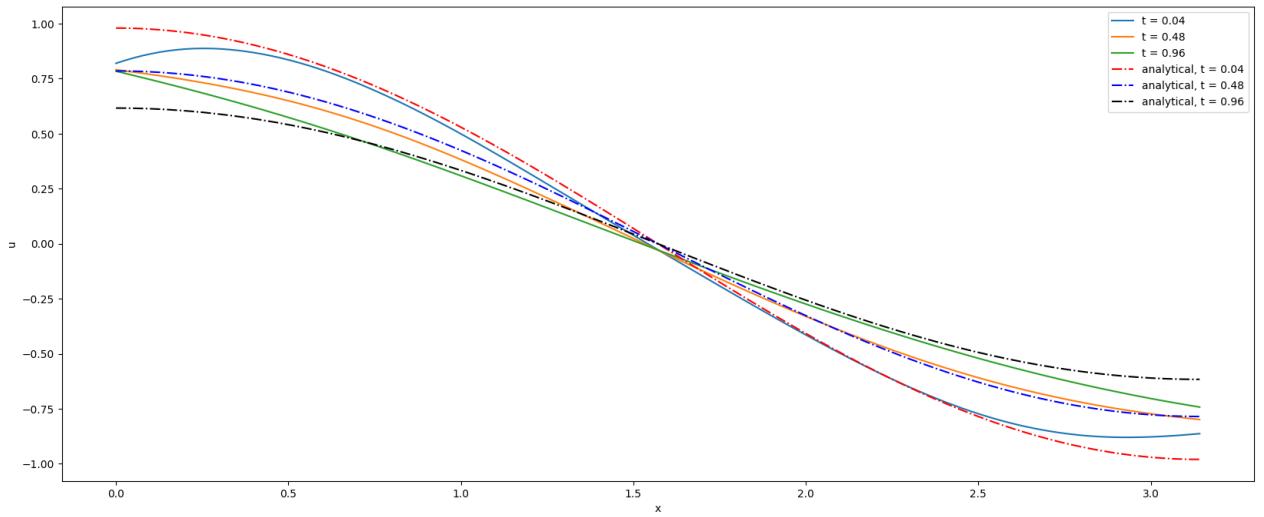


Рисунок 47 – Рекуррентная сеть. Ошибка на некоторых моментах времени

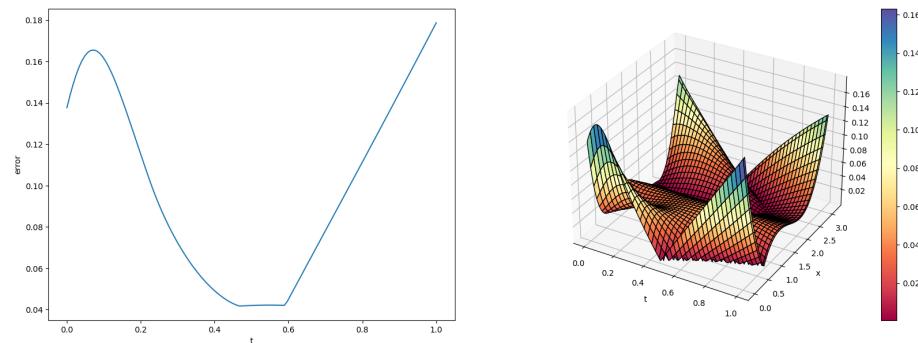


Рисунок 48 – Рекуррентная сеть. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

3.2.1.4 Нормализованные радиально-базисные сети

Несмотря на то, что полносвязные модели дают приемлемую точность, время обучения слишком большое. Поэтому были использованы нормализованные радиально-базисные сети (радиально-базисные сети показали себя хуже), как использовалось в [3]. Обучение также производилось на случайных точках и сетке. В качестве базисной функции использовалась функция Гаусса. В качестве оптимизатора использовался Adam.

Обучение на случайных точках было произведено на моделях с 16, 32, 64, 128 нейронами в слоях. Результаты приведены на рисунках 49 - 60.

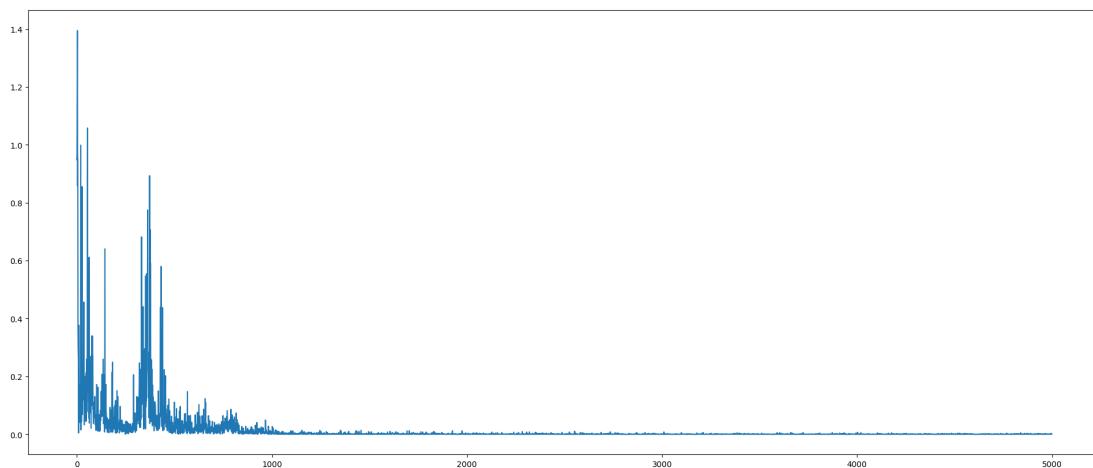


Рисунок 49 – Модель из 16-ти нейронов. Функция ошибки

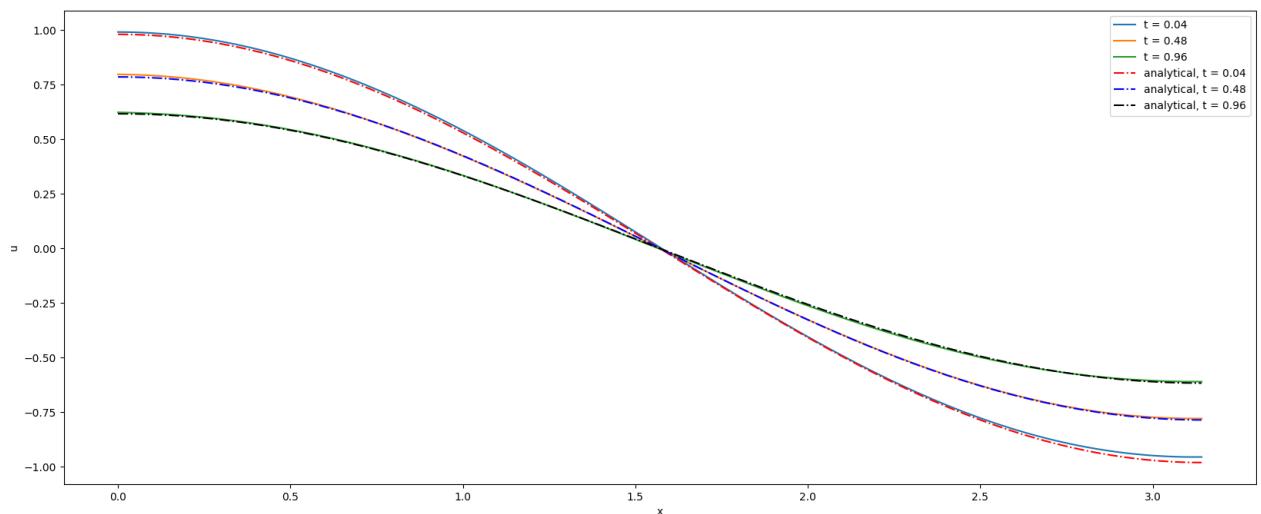


Рисунок 50 – Модель из 16-ти нейронов. Ошибка на некоторых моментах времени

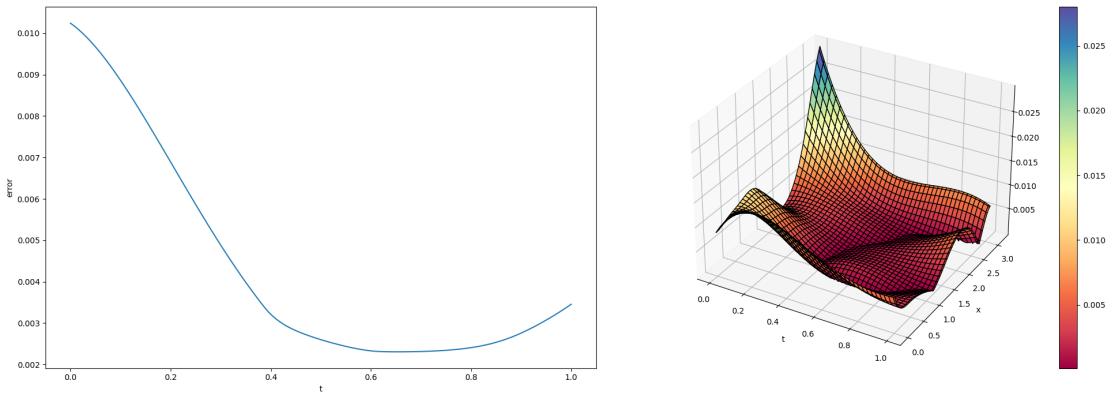


Рисунок 51 – Модель из 16-ти нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

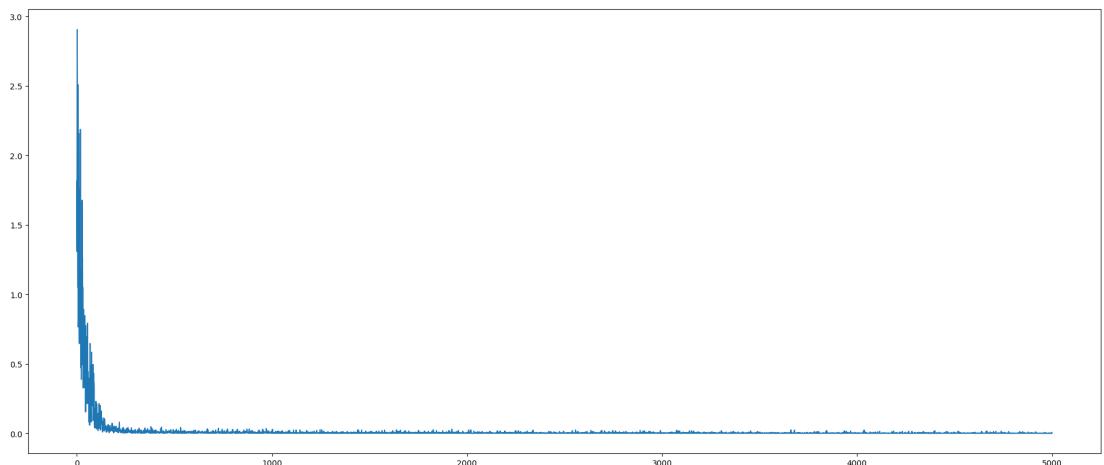


Рисунок 52 – Модель из 32-х нейронов. Функция ошибки

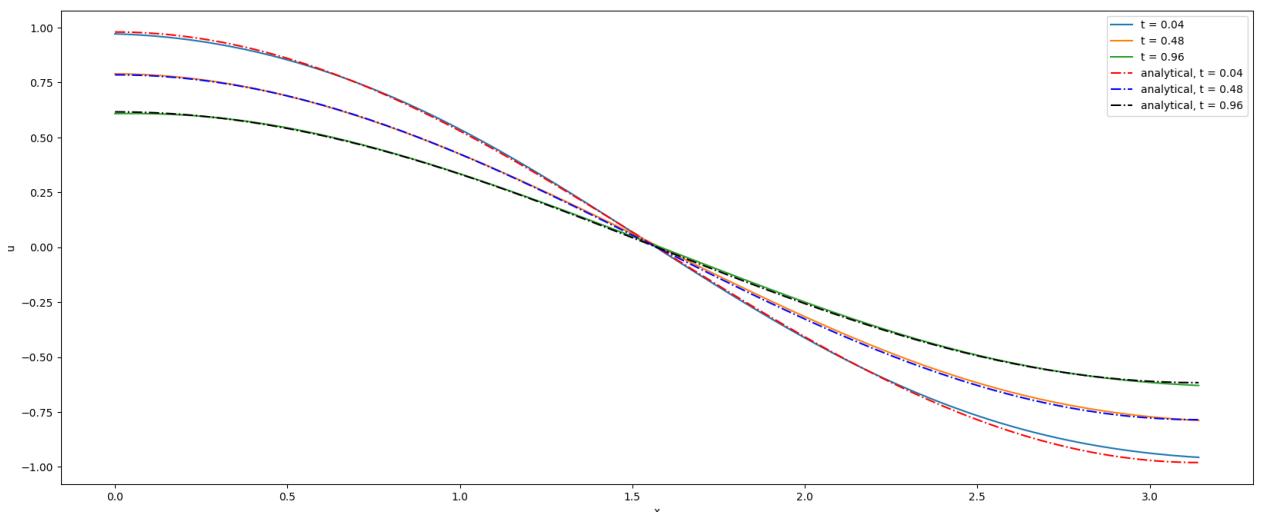


Рисунок 53 – Модель из 32-х нейронов. Ошибка на некоторых моментах времени

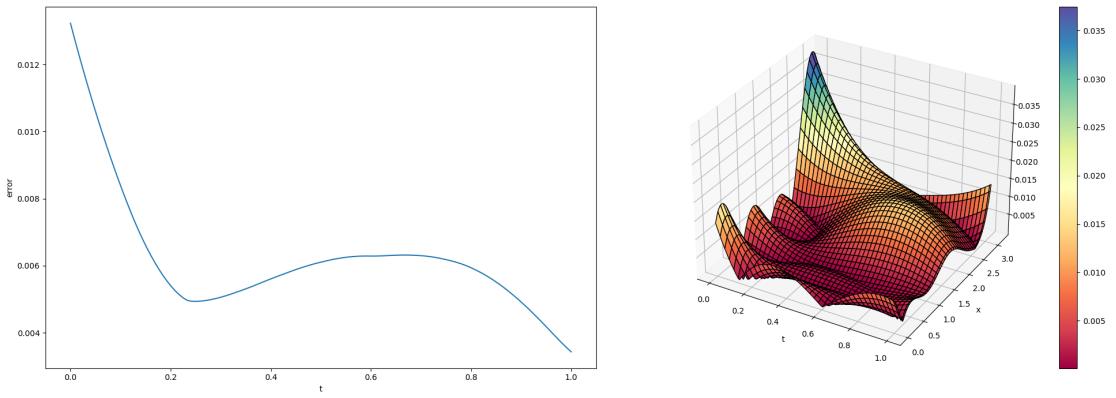


Рисунок 54 – Модель из 32-х нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

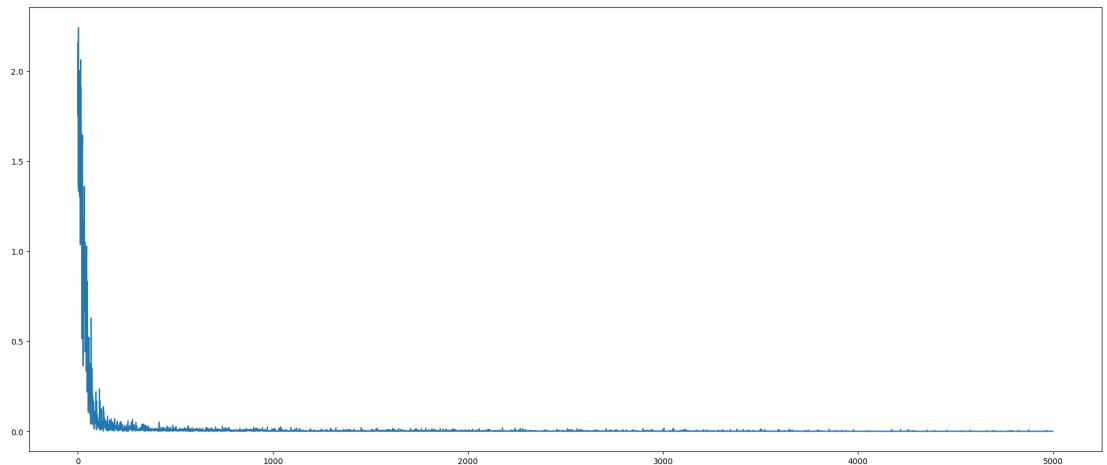


Рисунок 55 – Модель из 64-х нейронов. Функция ошибки

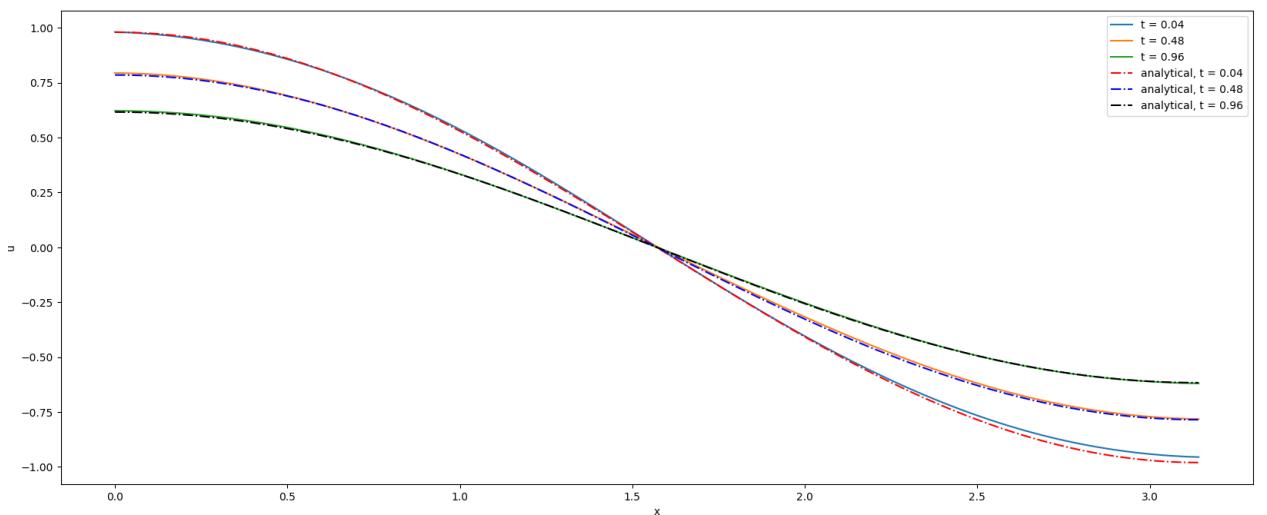


Рисунок 56 – Модель из 64-х нейронов. Ошибка на некоторых моментах времени

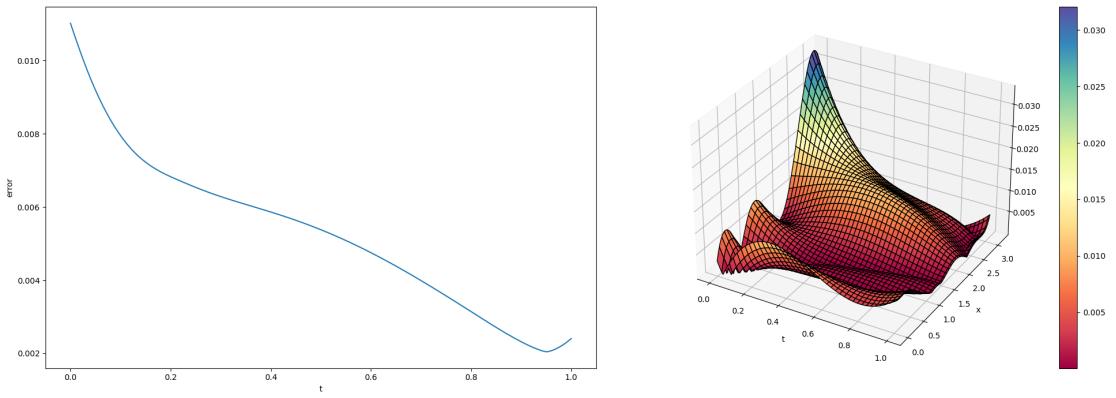


Рисунок 57 – Модель из 64-х нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

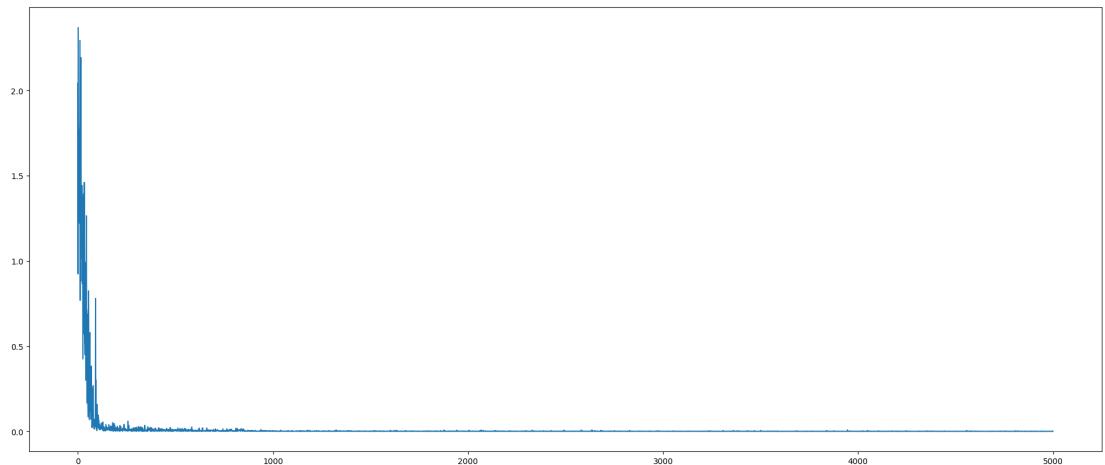


Рисунок 58 – Модель из 128-ми нейронов. Функция ошибки

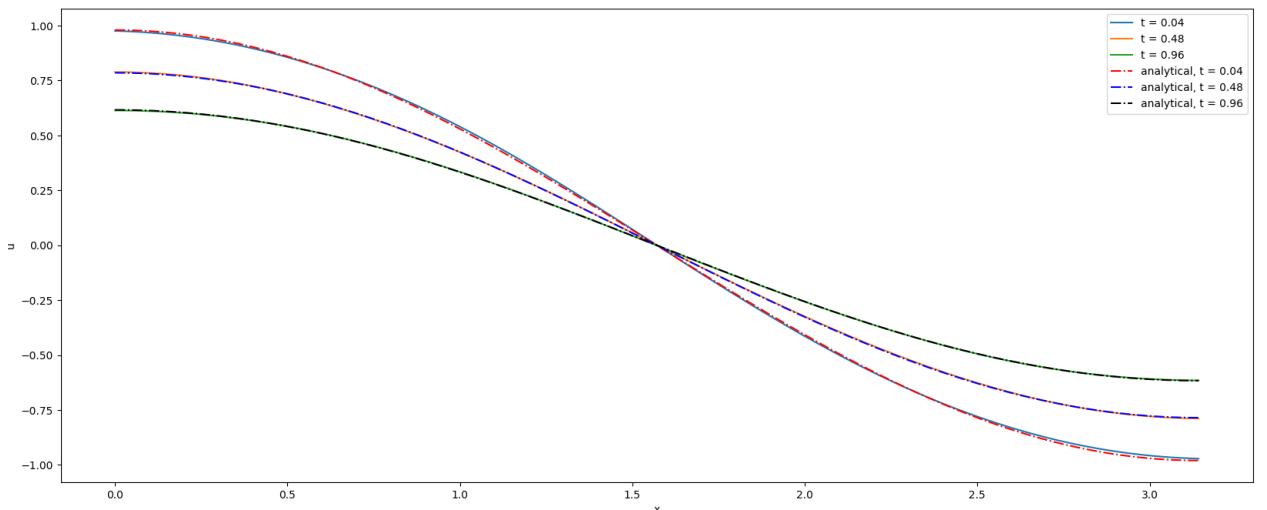


Рисунок 59 – Модель из 128-ми нейронов. Ошибка на некоторых моментах времени

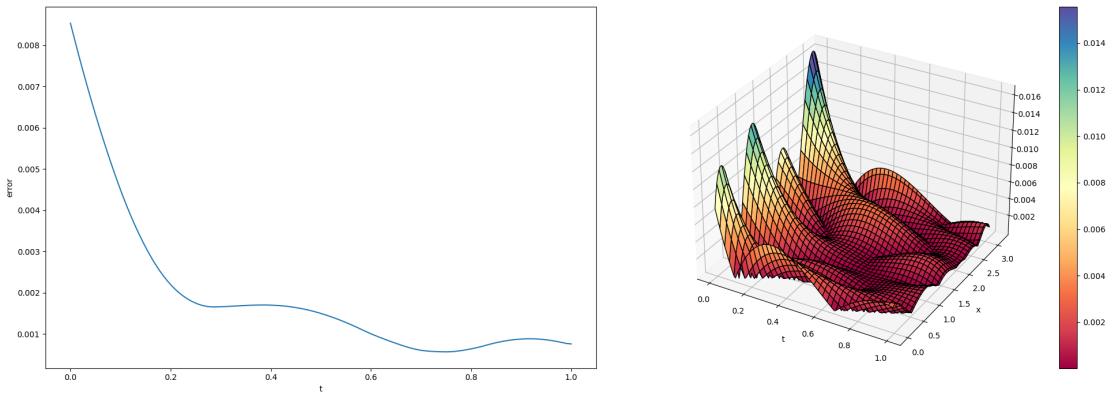


Рисунок 60 – Модель из 128-ми нейронов.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

Нормализованные радиально-базисные сети показали себя лучше, чем полносвязные. Время обучения сократилось до 1.5 - 3-х минут. Обучение на сетке также дало более точные результаты за более долгое время. Однако, все также быстрее, чем полносвязные модели. График зависимости максимальной ошибки от количества нейронов изображён на рисунке 61.

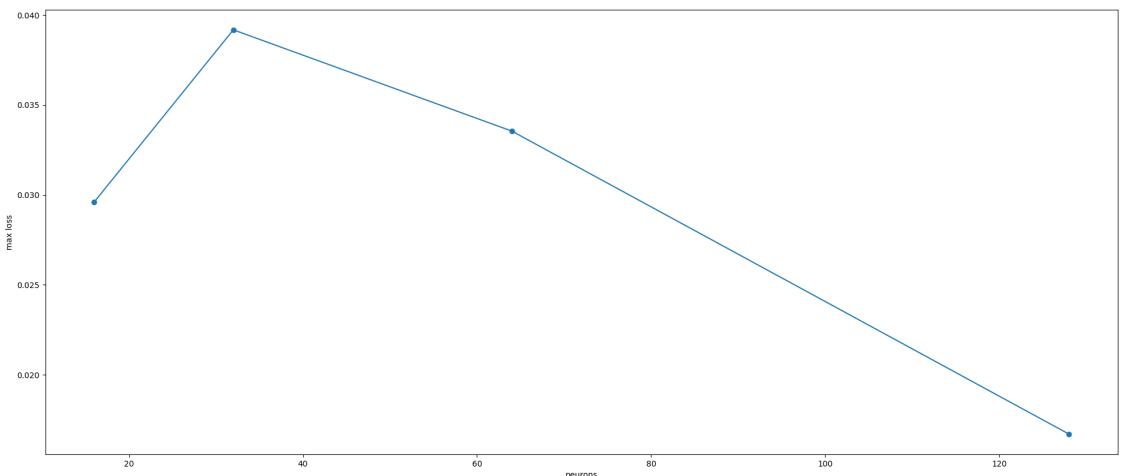


Рисунок 61 – Зависимость максимальной ошибки от количества нейронов

Результаты обучения модели, состоящей из 256 нейронов, на сетке приведены на рисунках 62 - 73.

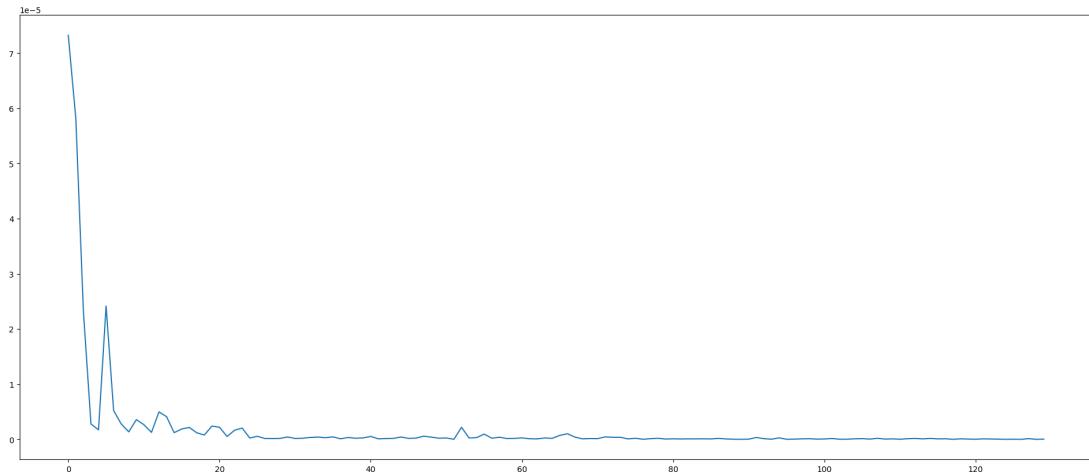


Рисунок 62 – Сетка для обучения 15x15. Функция ошибки

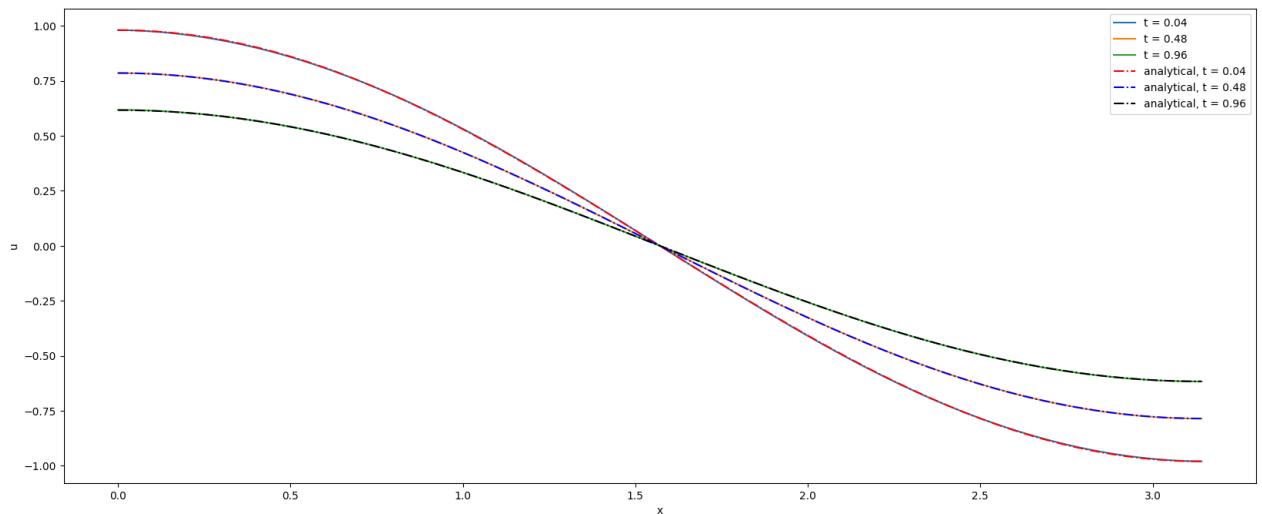


Рисунок 63 – Сетка для обучения 15x15. Ошибка на некоторых моментах времени

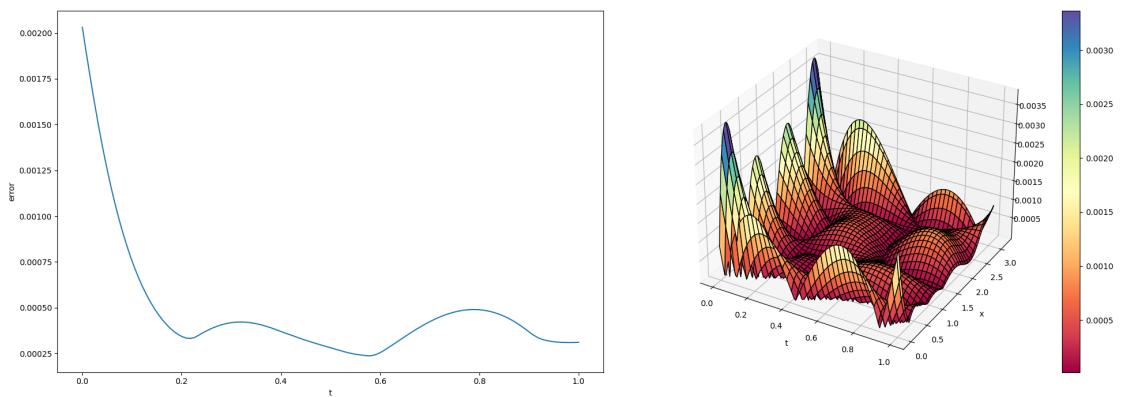


Рисунок 64 – Сетка для обучения 15x15.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

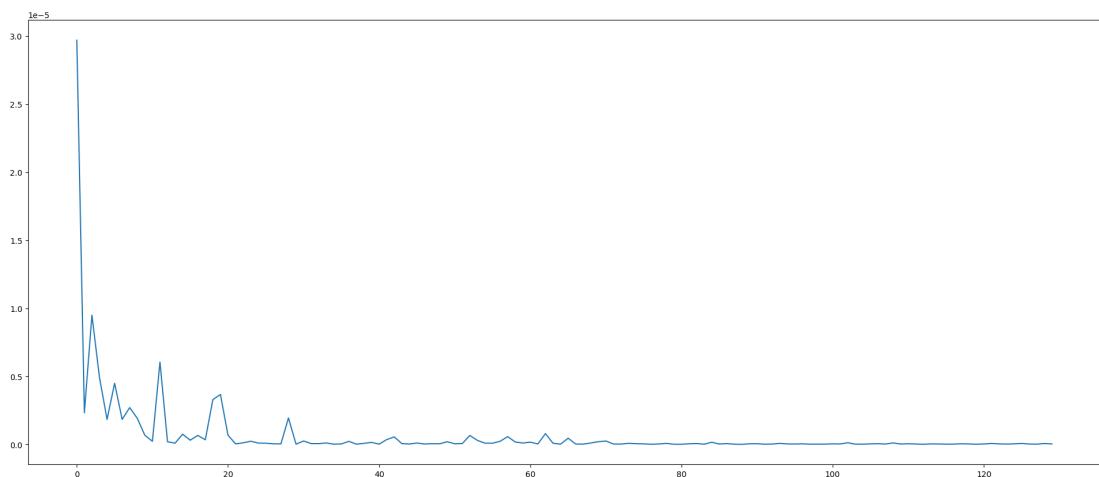


Рисунок 65 – Сетка для обучения 19x19. Функция ошибки

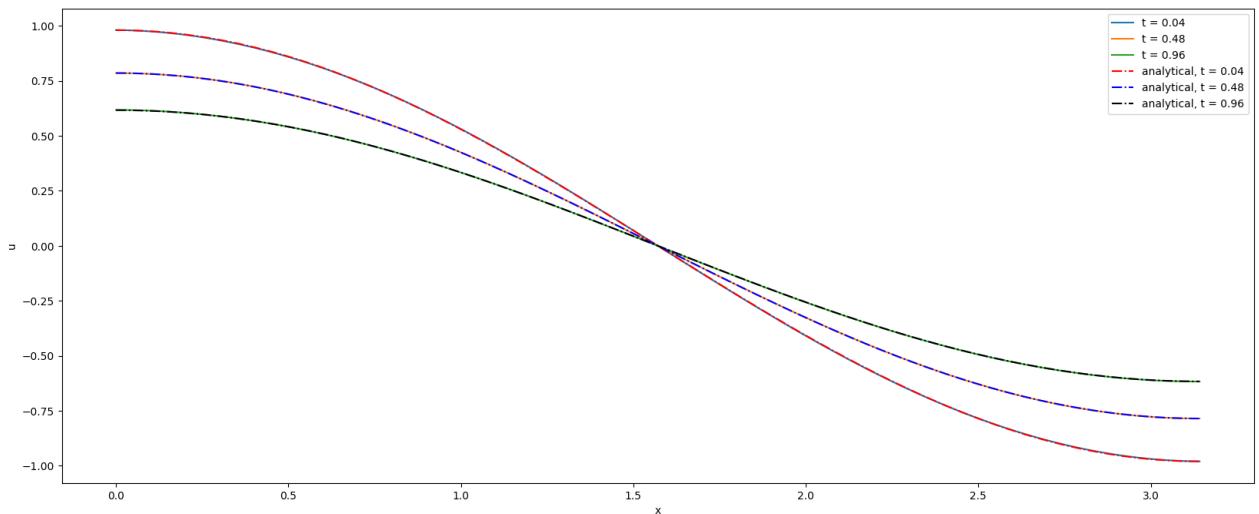


Рисунок 66 – Сетка для обучения 19x19. Ошибка на некоторых моментах времени

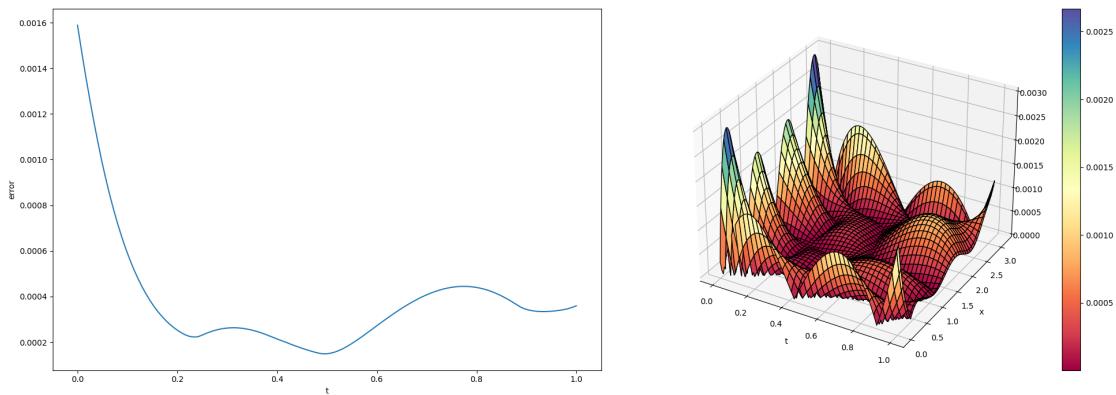


Рисунок 67 – Сетка для обучения 19x19.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

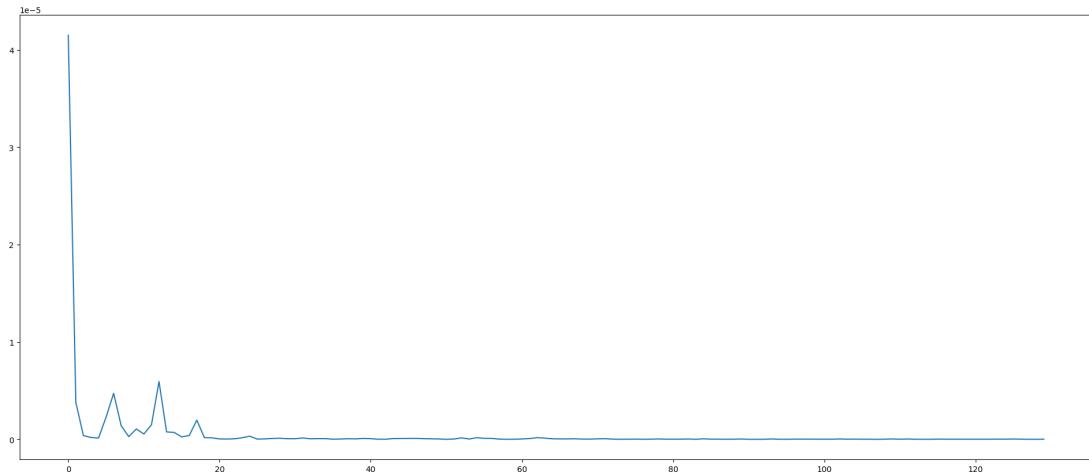


Рисунок 68 – Сетка для обучения 22x22. Функция ошибки

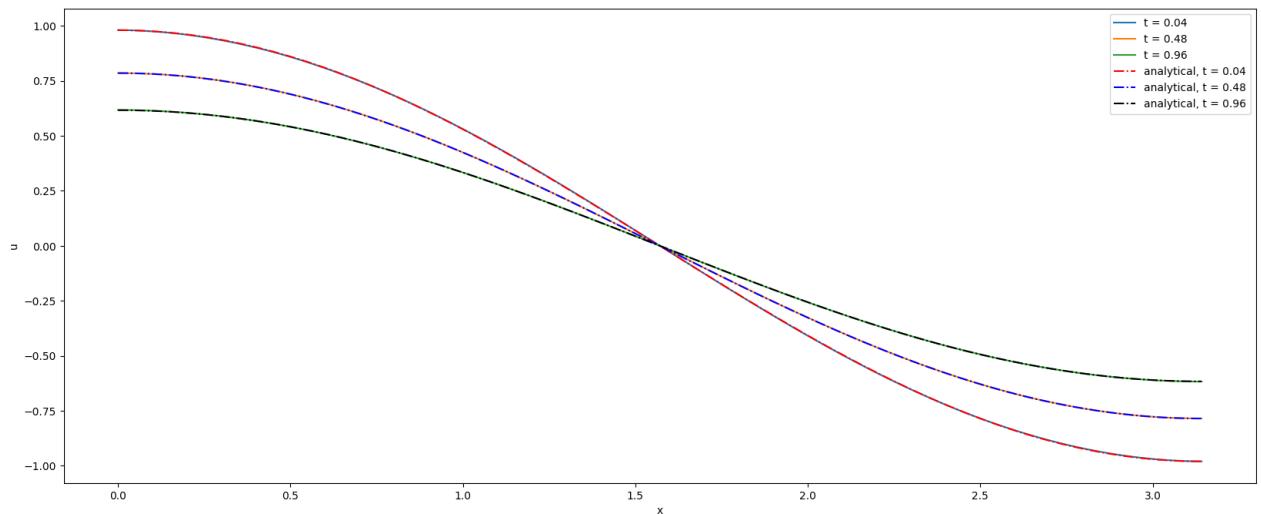


Рисунок 69 – Сетка для обучения 22x22. Ошибка на некоторых моментах времени

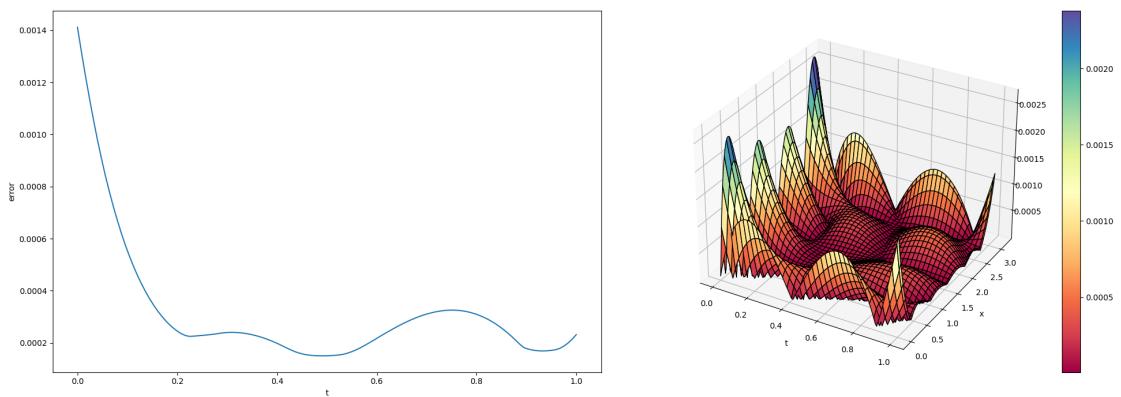


Рисунок 70 – Сетка для обучения 22x22.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

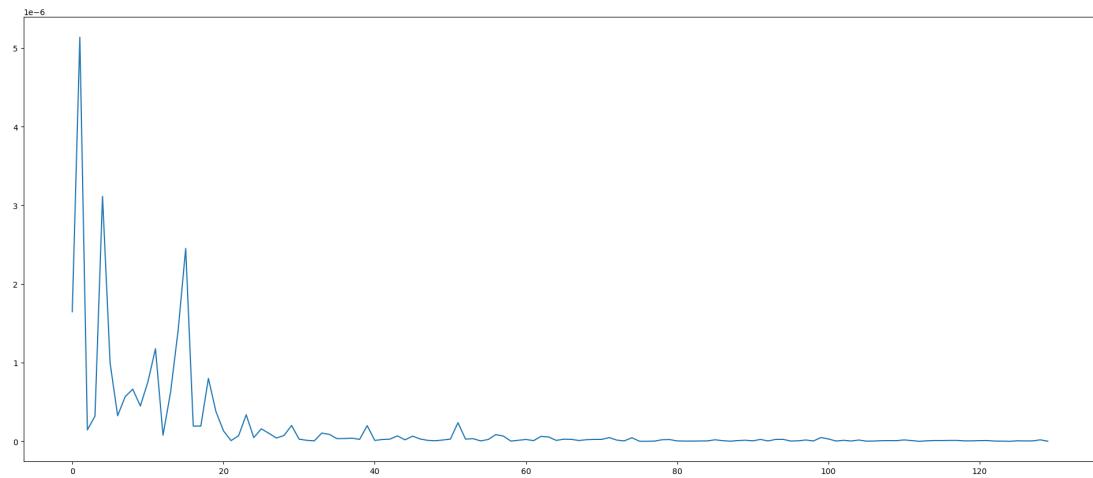


Рисунок 71 – Сетка для обучения 25x25. Функция ошибки

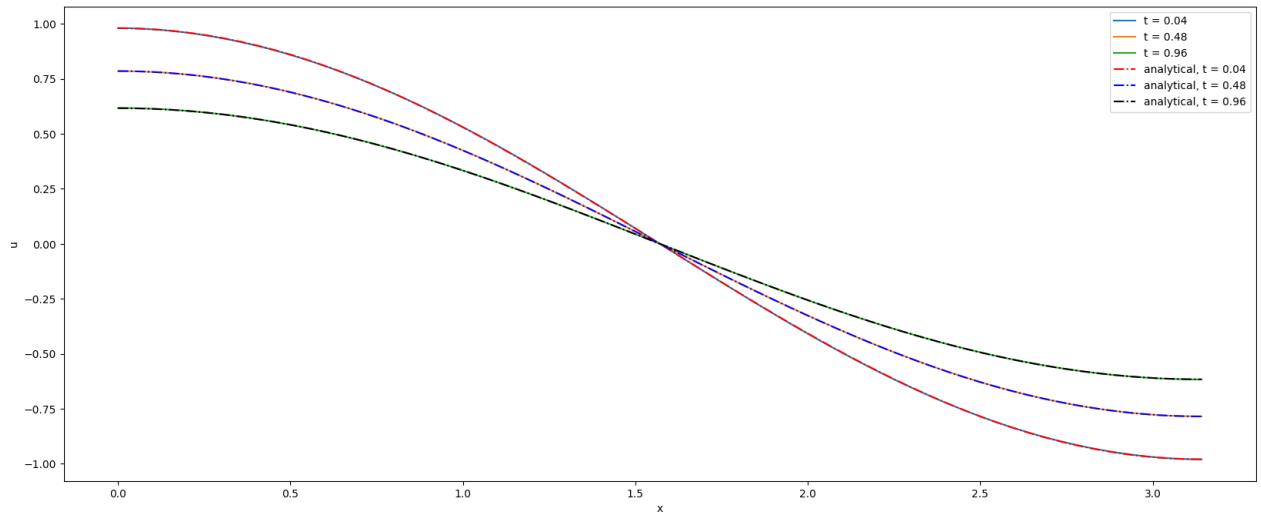


Рисунок 72 – Сетка для обучения 25x25. Ошибка на некоторых моментах времени

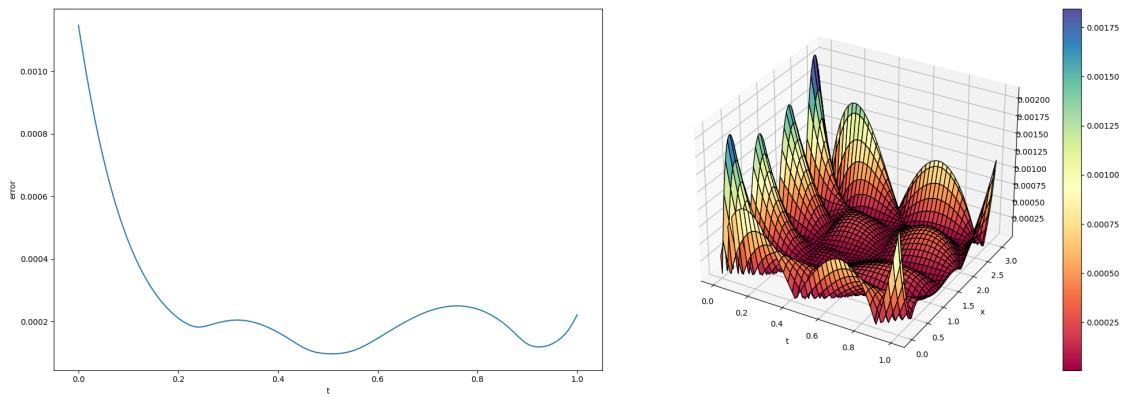


Рисунок 73 – Сетка для обучения 25x25.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

Время обучения составило от 4 минут до 11, в зависимости от размеров сетки. Зависимость максимально полученной ошибки от размера сетки представлена на рисунке 74.

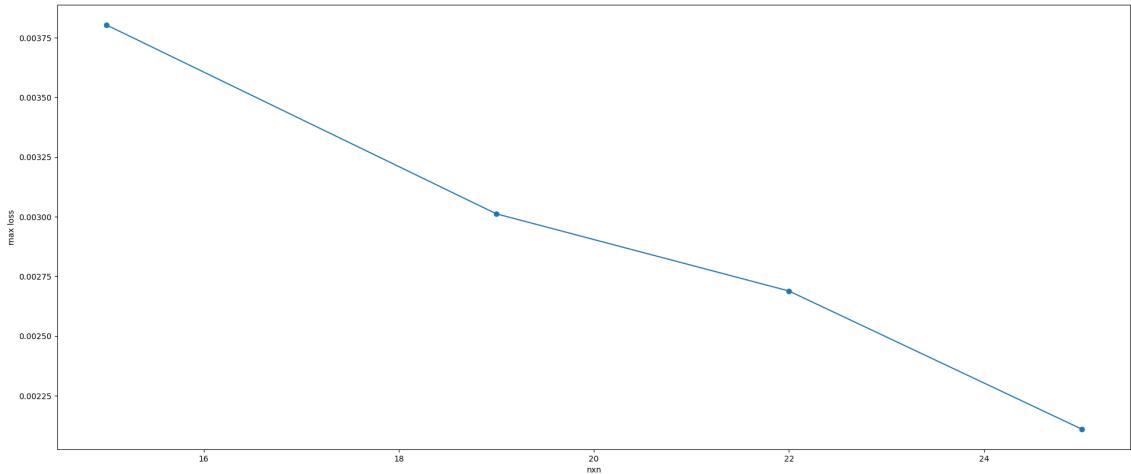


Рисунок 74 – График зависимости максимальной ошибки от размера сетки для обучения

3.2.1.5 Предсказание по времени

Была попытка предсказывать значение функции на промежутке времени, на котором не проводилось обучение. Но, как полносвязные, так и радиально-базисные сети дали плохие результаты. Они изображены на рисунках 75, 76.

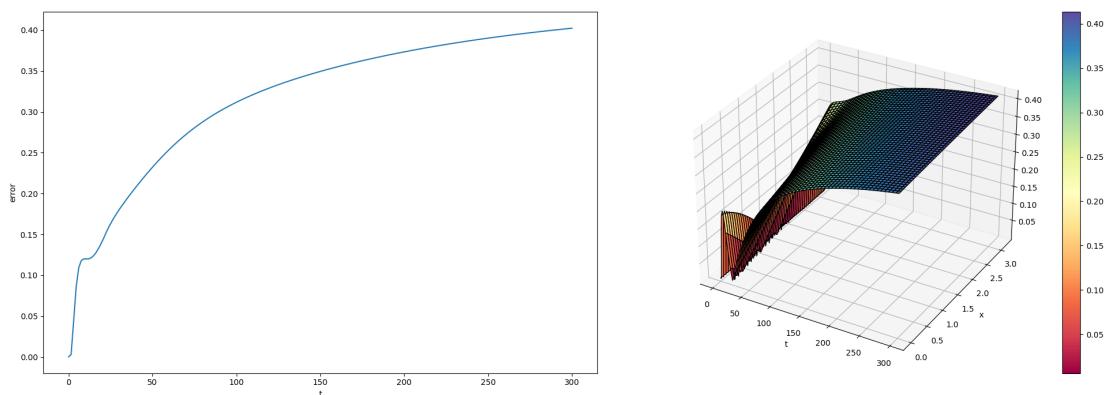


Рисунок 75 – Полносвязная сеть. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

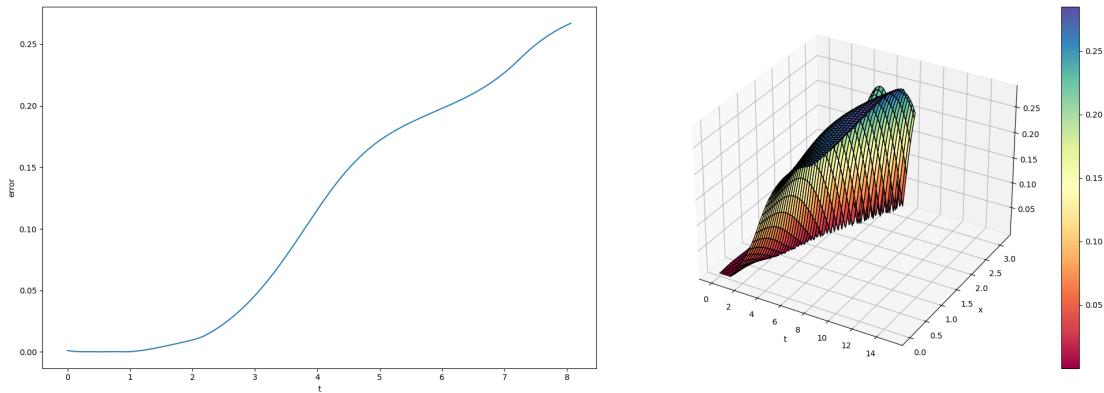


Рисунок 76 – НРБС. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

Однако, как обучение моделей на длинном промежутке времени, так и дообучение дали хорошие результаты. Дообучение происходило в 3 раза быстрее. Результаты обучения и дообучения нормализованной радиально-базисной сети на промежутке времени от 0 до 5 секунд, состоящей из 128 нейронов, представлены на рисунках 77 - 80.

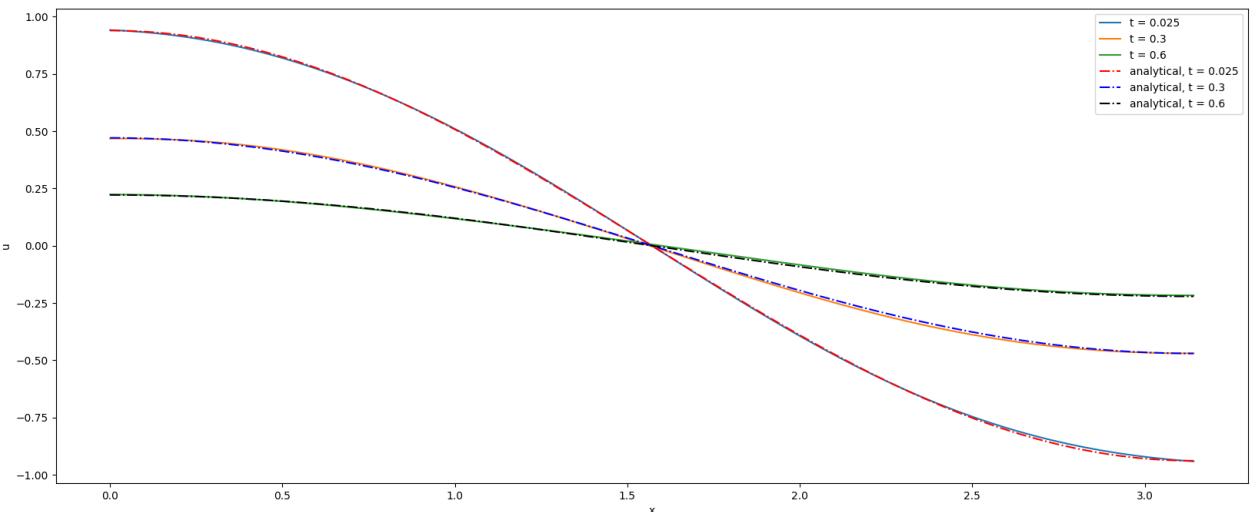


Рисунок 77 – Обучение с нуля. Ошибка на некоторых моментах времени

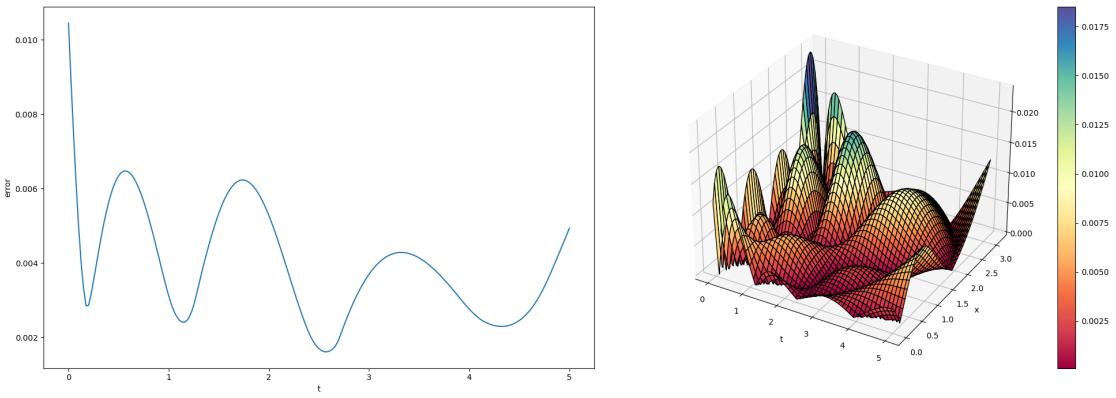


Рисунок 78 – Обучение с нуля. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

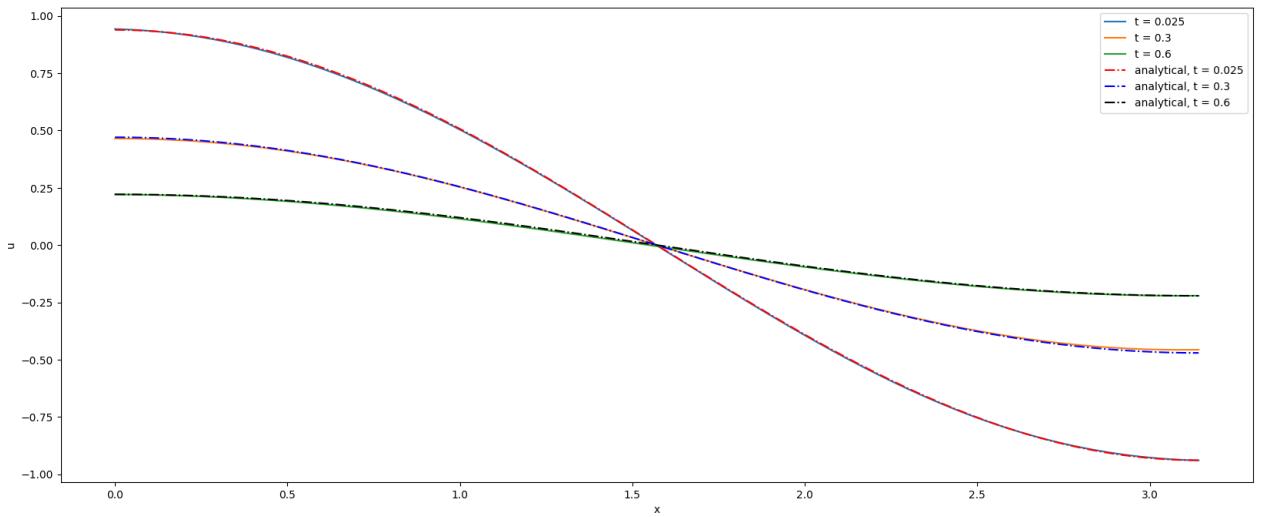


Рисунок 79 – Дообучение. Ошибка на некоторых моментах времени

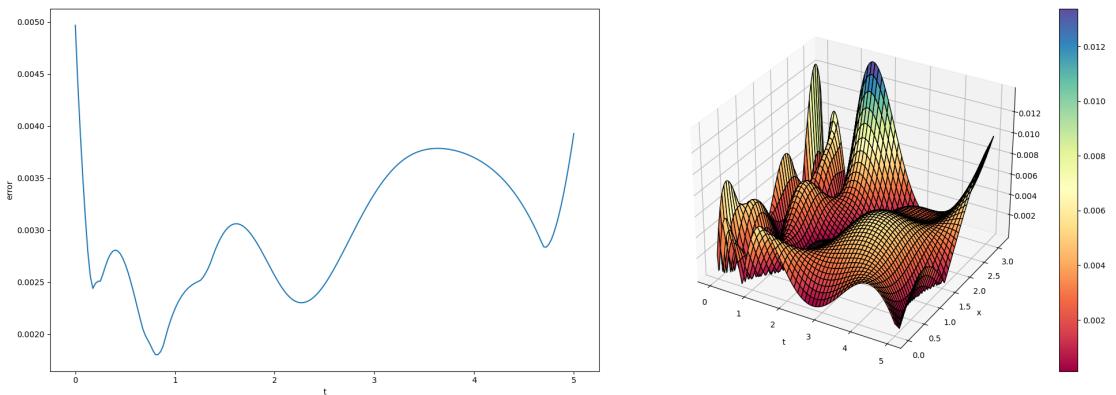


Рисунок 80 – Дообучение. Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

3.2.1.6 Гибридный метод

Далее был рассмотрен гибридный метод. В качестве нейросетей использовались как полносвязные, так и нормализованные радиально-базисные. В сумме получился неплохой результат, однако, время обучения занимало от 4.5 до 10 минут в зависимости от шага τ и архитектуры сети (полносвязные все также медленнее). Результаты представлены на рисунках 81 - 92.

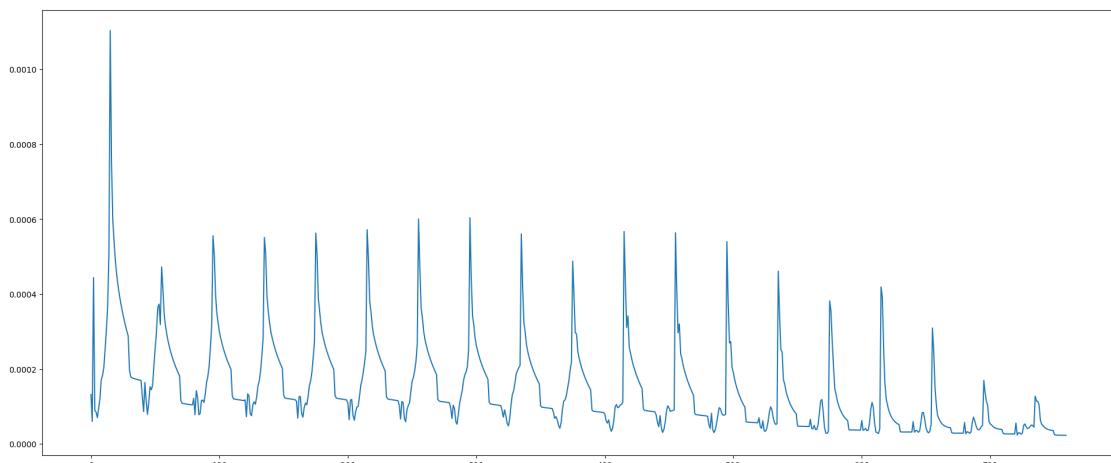


Рисунок 81 – НРБС, 16 нейронов, $\tau = 0.05$. Функция ошибки, все слои поочередно

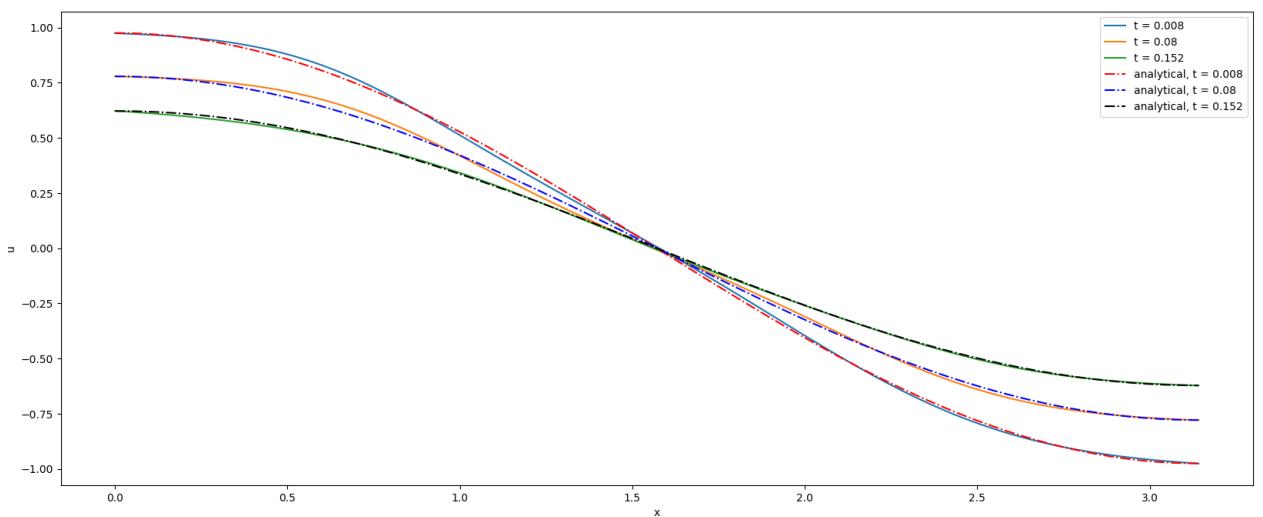


Рисунок 82 – НРБС, 16 нейронов, $\tau = 0.05$. Ошибка на некоторых моментах времени

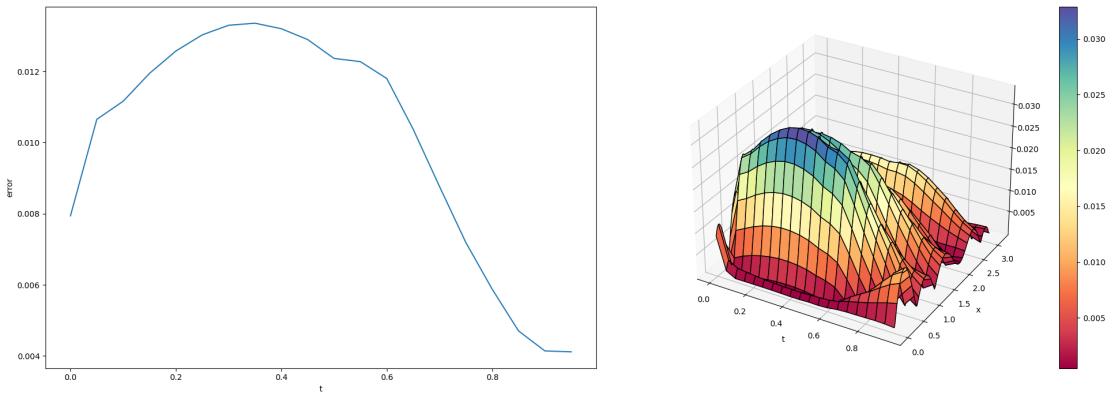


Рисунок 83 – НРБС, 16 нейронов, $\tau = 0.05$.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

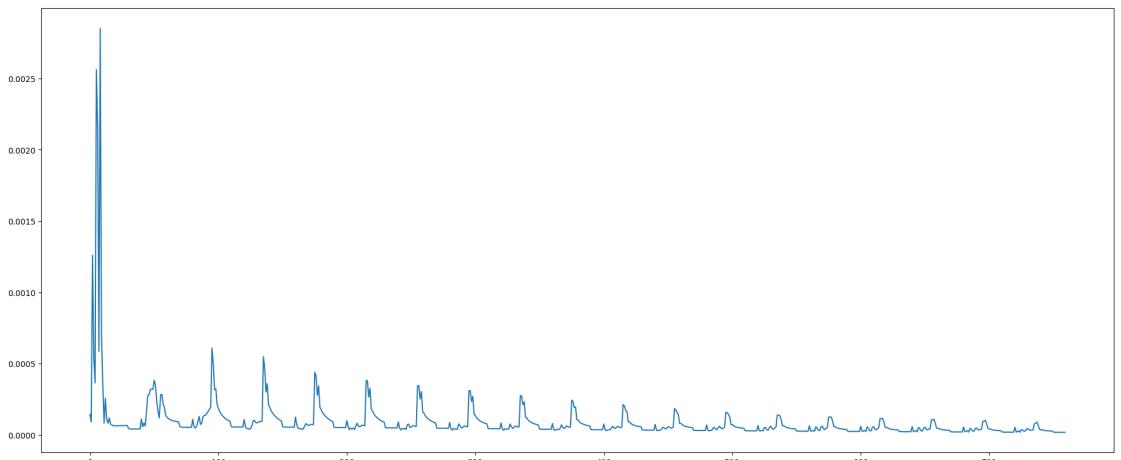


Рисунок 84 – НРБС, 32 нейрона, $\tau = 0.05$. Функция ошибки, все слои поочередно

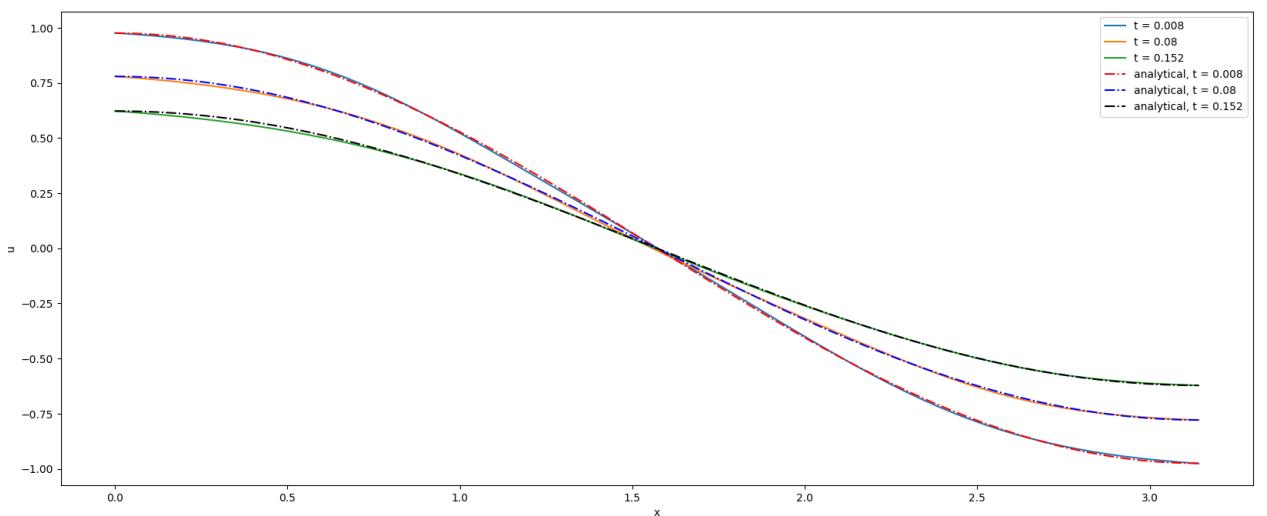


Рисунок 85 – НРБС, 32 нейрона, $\tau = 0.05$. Ошибка на некоторых моментах времени

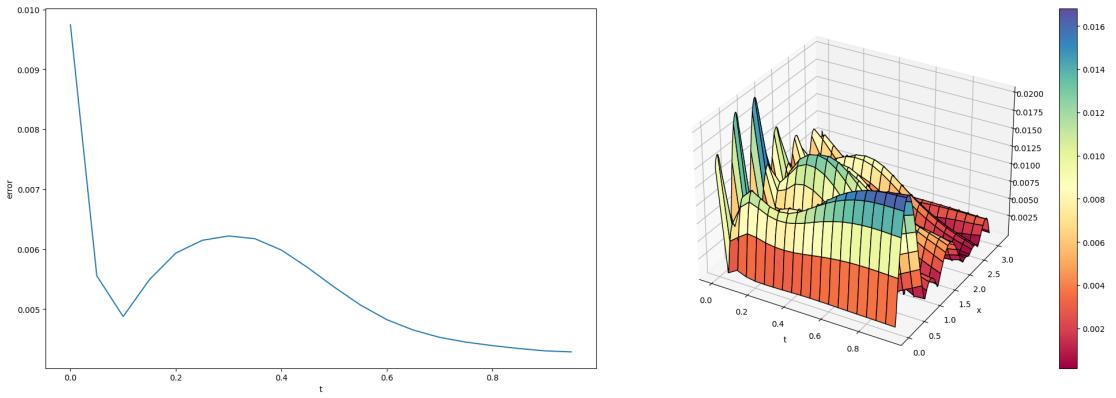


Рисунок 86 – НРБС, 32 нейрона, $\tau = 0.05$.
 Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

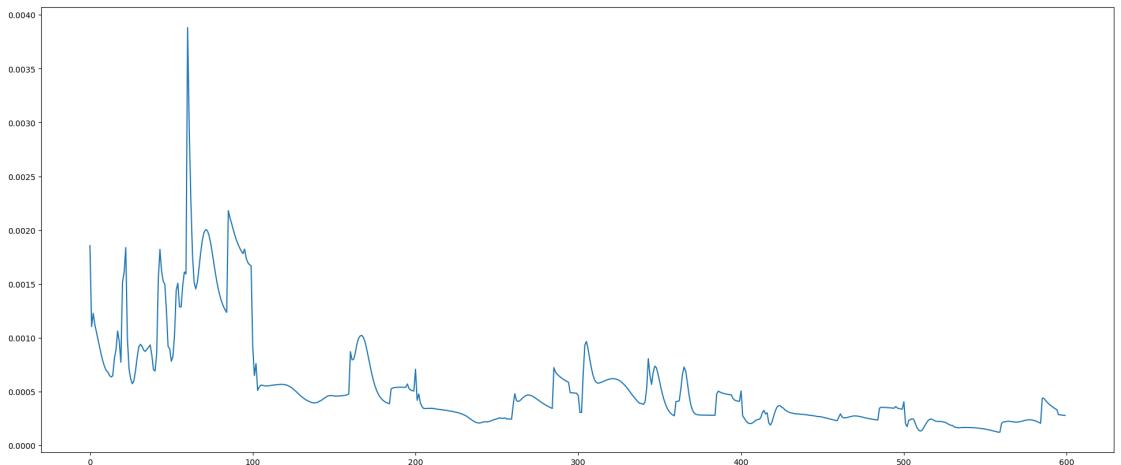


Рисунок 87 – Полносвязная сеть, 16 нейронов , $\tau = 0.15$.
 Функция ошибки, все слои поочередно

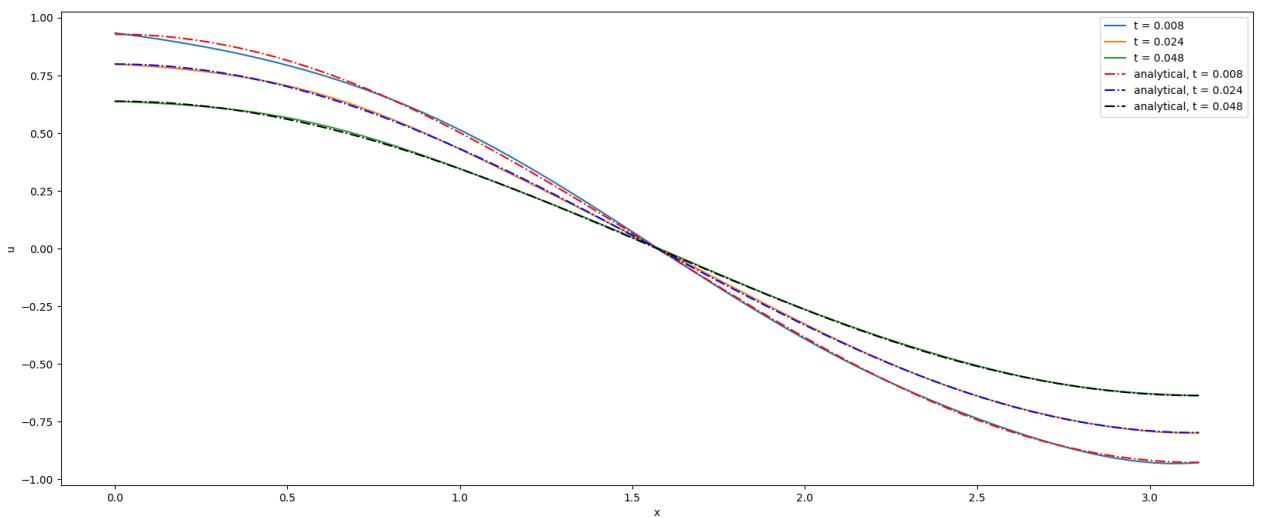


Рисунок 88 – Полносвязная сеть, 16 нейронов, $\tau = 0.15$.
 Ошибка на некоторых моментах времени

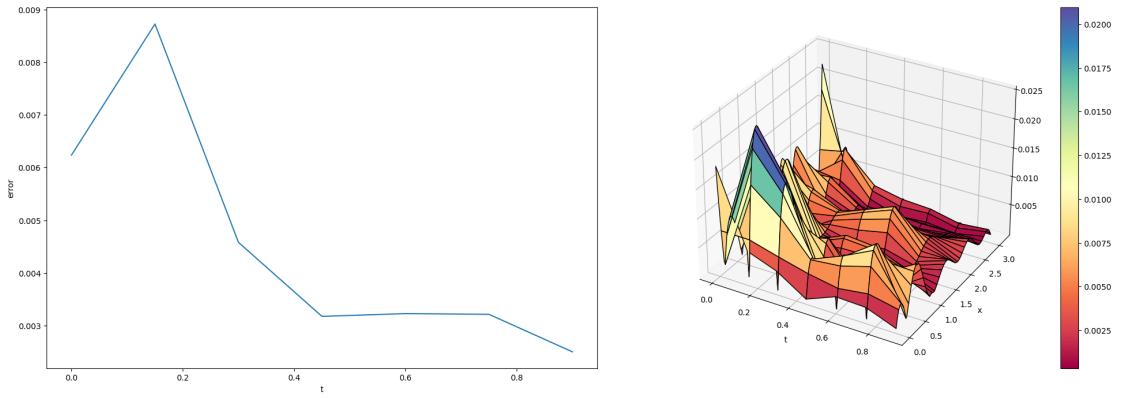


Рисунок 89 – Полносвязная сеть, 16 нейронов, $\tau = 0.15$.
Среднеарифметическая ошибка по x для каждого t . Модуль разности полученного решения нейросети и аналитического

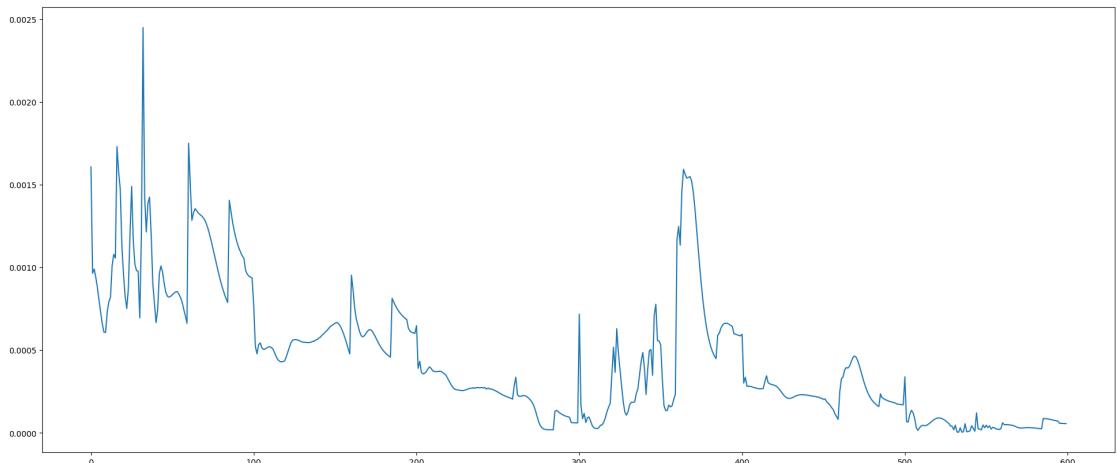


Рисунок 90 – Полносвязная сеть, 32 нейрона , $\tau = 0.15$.
Функция ошибки, все слои поочередно

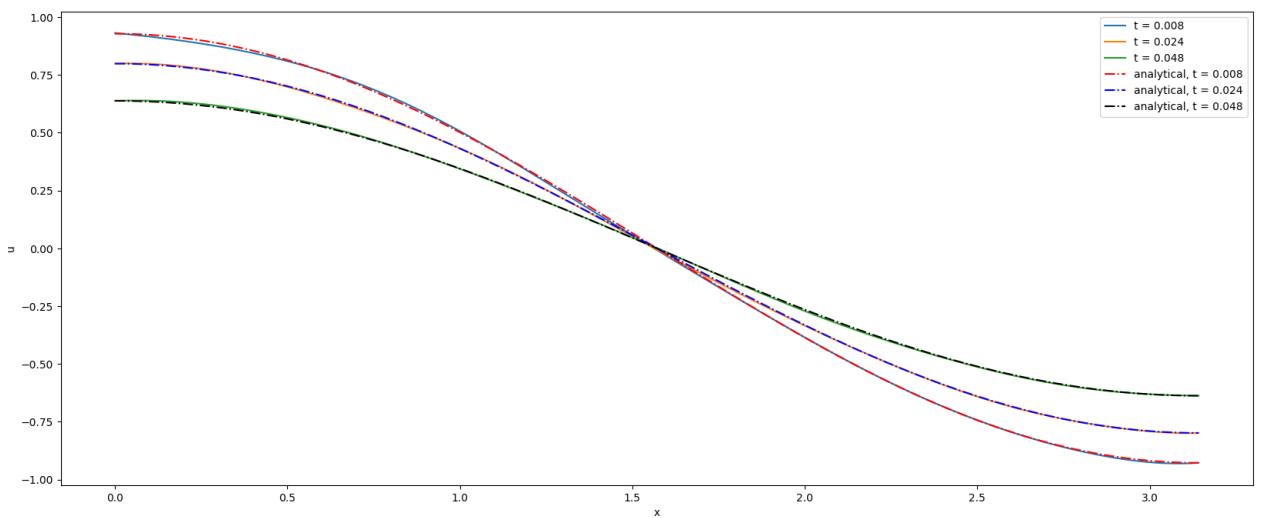


Рисунок 91 – Полносвязная сеть, 32 нейрона, $\tau = 0.15$.
Ошибка на некоторых моментах времени

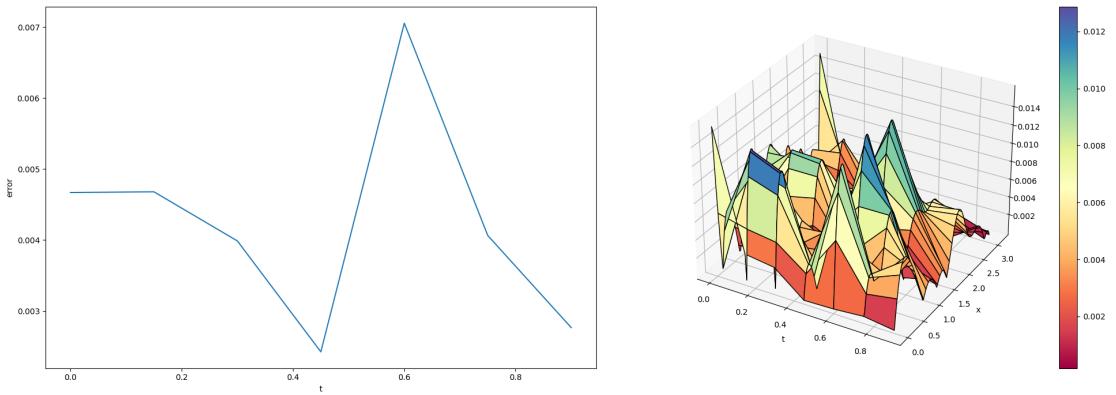


Рисунок 92 – Полносвязная сеть, 32 нейрона, $\tau = 0.15$.
Среднеарифметическая ошибка по x для каждого t . Модуль
разности полученного решения нейросети и аналитического

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы бакалавра был реализован и проанализирован нейросетевой метод решения дифференциальных уравнений в частных производных. Он уникален тем, что является бессеточным. Решение уравнения можно найти в любой точке интересующей нас области.

При реализации возникает множество нюансов, неочевидных в начале. Методы оптимизации первого порядка очень часто попадают в локальный минимум, и очень хороших результатов добиться не выходит. Также стоит отметить долгое время обучения в некоторых случаях. Поэтому, существуют его различные модификации, например, гибридный метод, суть которого заключается в том, что параметр времени в нейросети не учитывается, а считается с помощью какой-нибудь конечно-разностной схемы.

Нейросети предсказывают плохо, поэтому, стоит их использовать на том участке, на котором они обучались. При необходимости можно дообучить сеть, это происходит быстрее.

Хоть победить классические методы на одномерном уравнении параболического вида не удалось, метод имеет потенциал на решение дифференциальных уравнений в частных производных в многомерных областях. При использовании оптимизаторов более высоких порядков, например, метод Хегера-Чжана, можно добиться более быстрой и точной сходимости. Также для радиально-базисных сетей с функцией Гаусса возможен предподсчет экспоненты, что уменьшит время вычислений.

Несмотря на то, что идея нейросетей начала зарождаться еще в 60-е годы 20-го века, их полный потенциал открылся с недавнего времени, в связи с появлением интернета (возможностью пользования огромным количеством данных) и мощными вычислительными ресурсами. Поэтому, их использование является актуальным с точки зрения науки и может привести к технологическому прорыву. В частности, решение дифференциальных уравнений в частных производных с помощью нейросетей имеет потенциал. Вполне возможно, что завтра будет придумано нечто новое, что в значительной степени ускорит процесс сходимости и точность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Lu L., Meng X., Mao Z.* DeepXDE: A Deep Learning Library for Solving Differential Equations // SIAM Review. — 2021. — Т. 63, № 1. — С. 208—228.
2. *Madeira M.* Heat 2: The Heat Equation and a Physics-Informed Neural Network. — 2022. — URL: <https://inductiva.ai/blog/article/heat-2-pinn> (дата обращения 25.10.2022).
3. *Колбин И. С., Ревизников Д. Л.* РАЗРАБОТКА МЕТОДОВ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ НА ОСНОВЕ НОРМАЛИЗОВАННЫХ РАДИАЛЬНО-БАЗИСНЫХ СЕТЕЙ : диссертация / Колбин Илья Сергеевич, Ревизников Дмитрий Леонидович. — Московский авиационный институт (национальный исследовательский университет), 2013.
4. *Rosenblatt F.* The Perceptron, a Perceiving and Recognizing Automaton Project Para. — Cornell Aeronautical Laboratory, 1957. — (Report: Cornell Aeronautical Laboratory). — URL: <https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf> (дата обращения 28.09.2022).
5. Нейронные сети, перцептрон. — 2022. — URL: <https://neerc.ifmo.ru/wiki/index.php?title=%D0%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%81%D0%B5%D1%82%D0%B8%D0%BF%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD> (дата обращения 01.10.2022).
6. *Филипп С., Радослав Н.* Первое знакомство с полно связанными нейросетями. — URL: <https://academy.yandex.ru/handbook/ml/article/pervoe-zнакомство-s-polnosvyaznymi-nejrosetyami> (дата обращения 03.10.2022).
7. Рекуррентные нейронные сети. — 2022. — URL: [https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D0%BD%D1%82%D0%BD%D1%8B%D0%B5%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D0%BD%D1%81%D0%BD%D0%BD%D1%8B%D0%B8](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D0%BD%D1%82%D0%BD%D1%8B%D0%B5%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D0%BD%D1%81%D0%B5%D1%82%D0%B8%D0%B5%D1%81%D0%BD%D0%BD%D1%8B%D0%B8) (дата обращения 10.11.2022).

8. Введение в автоматическое дифференцирование. — 2022. — URL: <https://russianblogs.com/article/29231578190/> (дата обращения 19.10.2022).

9. Радослав Н., Станислав Ф. Метод обратного распространения ошибки. — URL: <https://academy.yandex.ru/handbook/ml/article/metod-obratnogo-rasprostraneniya-oshibki> (дата обращения 14.10.2022).

10. PyTorch - Adam. — URL: <https://pytorch.org/docs/stable/generated/torch.optim.Adam.html#torch.optim.Adam> (дата обращения 03.01.2023).

11. Python. — URL: <https://www.python.org/> (дата обращения 20.11.2022).

12. NumPy. — URL: <https://numpy.org/> (дата обращения 01.01.2023).

13. PyTorch. — URL: <https://pytorch.org/> (дата обращения 25.11.2022).

14. PyTorch - torch.autograd.grad. — URL: <https://pytorch.org/docs/stable/generated/torch.autograd.grad.html> (дата обращения 04.01.2023).

15. PyTorch - LBFGS. — URL: <https://pytorch.org/docs/stable/generated/torch.optim.LBFGS.html#torch.optim.LBFGS> (дата обращения 15.01.2023).

ПРИЛОЖЕНИЕ А

Исходный код

На рисунке А.1 изображен QR-код со ссылкой на GitHub репозиторий с исходным кодом.



Рисунок А.1 – QR-код на репозиторий