Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ Фундаментальные науки

КАФЕДРА Прикладная математика

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Распознавание графиков решения одномерного линейного уравнения переноса

Студент ФН2-61Б (Группа) _____ (Подпись, дата) О. Д. Климов (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) М. П. Галанин (И. О. Фамилия)

2024 г.

Оглавление

Введение

Необходимость распознавания графика функций возникает в совершенно разных прикладных задачах науки и техники. Например, она непосредственно связана с проблемой восстановления графика решений уравнений по неточно заданным данных о решениях. В силу нелинейности распознавания изображений нахождение точных алгоритмов для такой задачи испытывало ряд трудностей. Однако с развитием программирования и вычислительной техники стало возможным решать данную задачу методами нейронных сетей.

1. Постановка задачи

1.1. Формулировка

Необходимо изучить методы численного решения линейного одномерного уравнения переноса. Составить и отладить программу для нахождения численного решения задачи Коши для указанного уравнения. Использовать шесть различных разностных схем:

1) Явная схема с левой разностью на двух точках,
2) Явная схема с левой разностью на трех точках,
3) Неявная схема с левой разностью на двух точках,
4) Неявная схема с левой разностью на трех точках,
5) Схема Лакса,
6) Схема Лакса-Вендрофа.

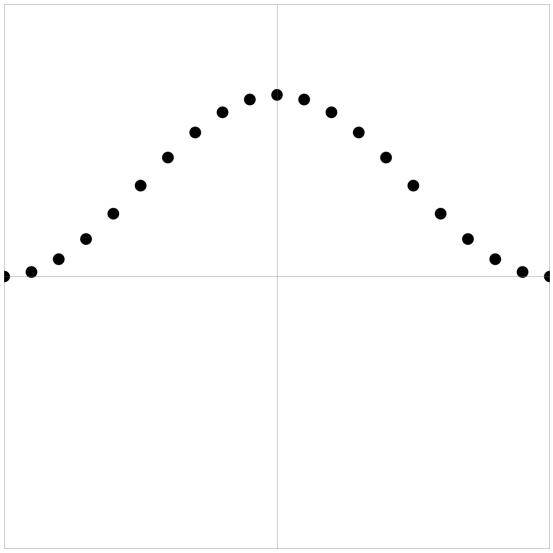
Для всех схем использовать одинаковую систему тестов:

1) Левый треугольник,
2) Правый треугольник,
3) Прямоугольник,
4) Косинус,
5) «Зуб».

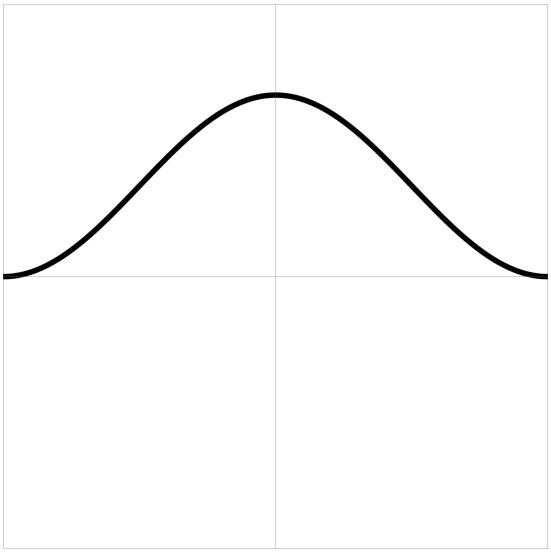
Реализовать модель нейронной сети для распознавания решения на языке программирования Python. На основе модели разработать программу, которая по неточному решению возвращает более точное известное решение.

1.2. Пример

Основной задачей работы является подготовка программы, которая получает с изображения неточный график решения уравнения переноса (рис. 1a) и в ходе своей работы возвращает улучшенное решение. (рис. 1b).



(a) График неточного решения



(b) График точного решения

Рис. 1. Иллюстрация задачи

К поставленной задаче, в силу свойств выбранного метода решения, ставятся некоторые ограничения. График будем рассматривать только в определенной области, а все возможные решения будем считать известными и их количество ограниченным. Тем самым, будем рассматривать только определенные наборы решений в выбранные моменты времени на некотором интервале.

2. Задача Коши для линейного одномерного уравнения переноса

Уравнение переноса является одним из фундаментальных уравнений математической физики, которое широко используется для описания движения сплошной среды. В то же время оно является простейшим представителем семейства гиперболических уравнений, которые определяются наличием действительных характеристик, число которых совпадает с числом неизвестных для системы уравнений первого порядка.

Рассмотрим задачу Коши для уравнения переноса следующего вида:

$$\begin{cases} \dfrac{\partial u}{\partial t} + a\dfrac{\partial u}{\partial x} = 0, \\ u(x,0) = u_0(x), \end{cases} \quad \text{где } u = u(x,t), \quad a = const > 0, \quad t \in (0,T), \quad x \in (-\infty, +\infty). \tag{1}$$

Приведем аналитическое решение. Уравнение можно записать в виде:

$$u_t + au_x = 0$$

Запишем и решим характеристическое уравнение:

$$\frac{dt}{1} = \frac{dx}{a} = \frac{du}{0} \quad \Longrightarrow \quad \begin{cases} u = C_1, \\ x = at + C_2 \end{cases} \quad \Longrightarrow \quad \begin{cases} C_1 = u, \\ C_2 = x - at \end{cases} \quad \Longrightarrow \quad u = \psi(x - at).$$

Применим граничные условия:

$$u(x,0) = u_0(x), \quad \Longrightarrow \quad \psi(x) = u_0(x), \quad \Longrightarrow \quad u = u_0(x - at).$$

Получим аналитическое решение уравнения $u = u_0(x - at)$. Решение заключается в сносе неизменного профиля по характеристикам.

Важнейшим свойством рассматриваемого решения будет являться сохранение начального профиля: если начальное решение представляет собой, например, профиль буквы «М», то оно будет сохранять его таким всегда.

3. Численные методы решения задачи

Целью построения методов численного решения данного уравнения является не собственно нахождение его решения, а исследование численного алгоритма на простейшем примере. Аналитическое решение данного уравнения известно и тривиально. При этом в чрезвычайно большое количество математических моделей уравнение переноса входит в качестве составной части. К таким относятся модели газодинамики, гидродинамики, переноса частиц и излучения, электродинамики и многие другие. Разработать метод численного решения для этих моделей можно только в том случае, если метод будет построен и успешно применен для данного простейшего уравнения, описывающего обыкновенный перенос. [1]

3.1. Описание методов

Обозначим $\gamma = \dfrac{a\tau}{h}$ — число Куранта. Для численного решения задачи для линейного одномерного уравнения переноса рассмотрим следующие схемы:

1) **Явная схема с левой разностью по двум точкам**:

$$\widehat{y} = (1 - \gamma)y + \gamma y_{-1}.$$

Погрешность аппроксимации схемы: $\psi_h = O(\tau + h)$.

Схема устойчива при $\gamma \leq 1$, причем при $\gamma = 1$ схема является точной.

2) **Неявная схема с левой разностью по двум точкам**:

$$\widehat{y} = \frac{\gamma}{1+\gamma}\widehat{y}_{-1} + \frac{1}{1+\gamma}y.$$

Погрешность аппроксимации схемы: $\psi_h = O(\tau + h)$.

Данная схема является безусловно устойчивой.

3) **Явная схема с левой разностью по трем точкам**:

$$\widehat{y} = (1 - \frac{3}{2}\gamma)y + \gamma(2y_{-1} - \frac{1}{2}y_{-2}).$$

Погрешность аппроксимации схемы: $\psi_h = O(\tau + h^2)$.

Схема является безусловно неустойчивой.

4) **Неявная схема с левой разностью по трем точкам**:

$$\widehat{y} = \frac{2}{2+3\gamma}y + \frac{\gamma}{2+3\gamma}(4\widehat{y}_{-1} - \widehat{y}_{-2}).$$

Погрешность аппроксимации схемы: $\psi_h = O(\tau+h^2)$. Схема является безусловно неустойчивой. В подтверждение далее приведены результаты расчетов.

5) **Схема Лакса**:

$$\widehat{y} = \frac{(y_{+1} + y_{-1}) - \gamma(y_{+1} - y_{-1})}{2}.$$

Погрешность аппроксимации схемы: $\psi_h = O(\tau + h + \frac{h^2}{\tau})$.

Схема устойчива при $\gamma \leq 1$ и сходится при стремлении $h^2 \to 0$ быстрее, чем τ.

6) **Схема Лакса-Вендрофа**:

$$\widehat{y} = y - \gamma(\frac{(y_{+1} + y) - \gamma(y_{+1} - y)}{2} - \frac{(y + y_{-1}) - \gamma(y - y_{-1})}{2}).$$

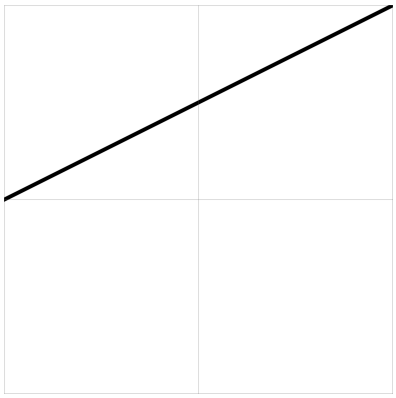
Погрешность аппроксимации схемы: $\psi_h = O(\tau^2 + h^2)$.

Схема устойчива при $\gamma \leq 1$, а при $\gamma = 1$ схема является точной.

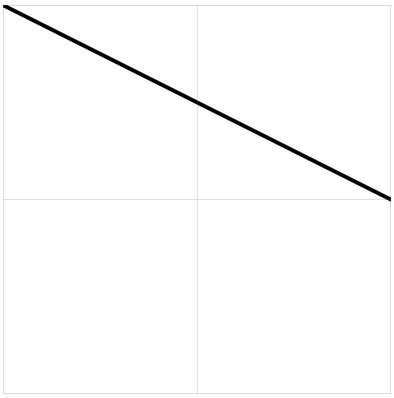
3.2. Реализация

Все эти схемы реализованы в программе на языке программирования C++. Был создан единый программный класс, который включает в себя все необходимые данные об уравнении и методе, а также сами реализации методов. Программа возвращает массив с результатами, записанный в текстовый файл.

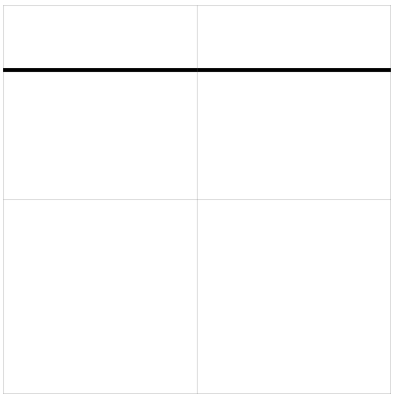
Для тестирования были взяты следующие начальные условия:



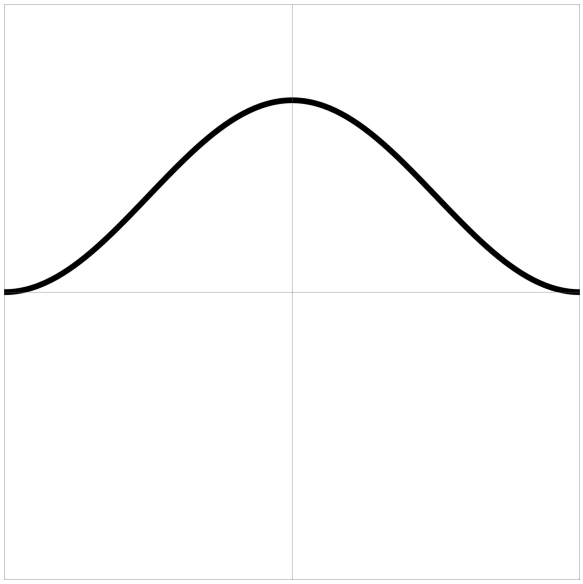
(a) Левый треугольник
$u_0(x) = \frac{x-l_1}{l_2-l_1}$



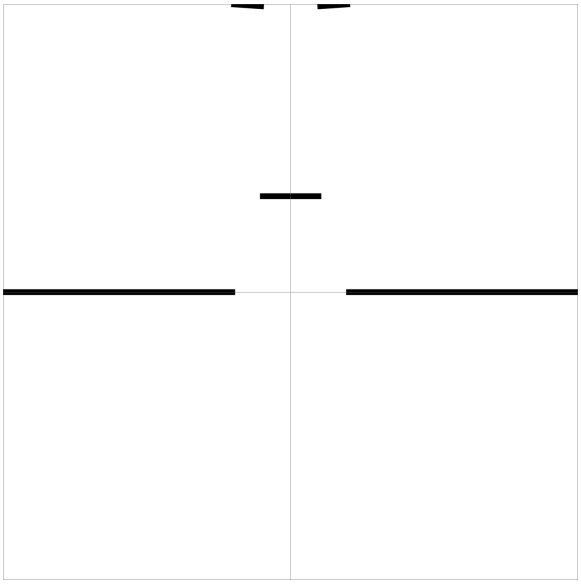
(b) Правый треугольник
$u_0(x) = \frac{l_2-x}{l_2-l_1}$



(c) Прямоугольник
$u_0(x) = \frac{2}{3}$



(d) Косинус
$u_0(x) = \frac{1}{3}(1 - \cos(\frac{2\pi(x-l_1)}{l_2-l_1}))$



(e) «Зуб»

$$u_0(x) = \begin{cases} -\frac{2}{3}(l_{11} - l_1)(x - l_1) + 1, & l_1 \le x < l_{11} \\ \frac{1}{3}, & l_{11} \le x \le l_{22} \\ \frac{2}{3}(l_2 - l_{22})(x - l_2) + 1, & l_{22} < x \le l_2 \end{cases}$$

Для расчета по схемам организован итерационный процесс. Создается сетка с шагами h по пространству и τ по времени. Первой итерацией в объект std::vector записывается начальное условие на сетке. На каждом шаге по времени записывается новый вектор значений, который изменяется по соответствующей схеме. Для явных схем процесс организован в виде последовательного вычисления верхнего слоя значений исходя из нижнего. Для работы всех схем задается левое граничное условие, так как без него счет по схемам невозможен. Для схемы (3) для получения второй верхней точки используется на первой итерации схема (2), причем, хоть и схема теоретически неустойчива, но результаты счета по ней иногда являются вполне корректными.

В случае неявной схемы (2) вычисление ведется также как и по явной, так как информация в левой верхней точке известна из левого граничного условия. Для схем (5) и (6) также использовалось правое граничное условие, из-за необходимости информации о крайней левой верхней точке. В схеме (4) на первой итерации используется схема (2).

Покажем сходимость или расходимость схем путем вычисления нормы ошибки.

Таблица 1. Нормы ошибки разностных схем, $\gamma = 0.5$

Схема / h, τ	Явная ЛДР 1-го пор.	Неявная ЛДР 1-го пор.	Явная ЛДР 2-го пор.
$h = 0.1, \tau = 0.05$	0.042	0.109	0.107
$h = 0.01, \tau = 0.005$	0.0045	0.013	159
Схема / h, τ	Неявная ЛДР 2-го пор.	Лакса	Лакса-Вендрофа
$h = 0.1, \tau = 0.05$	10^{10}	0.107	0.0067
$h = 0.01, \tau = 0.005$	10^{18}	0.013	0.000065

Норма вычислялась как норма в L_1 в пространстве сеточных функций по формуле

$$\|\psi\|_{L_1} = \sum_{i=0}^{N_x} \sum_{j=0}^{N_t} |u(x_i, t_j) - y(x_i, t_j)| \cdot h \cdot \tau,$$

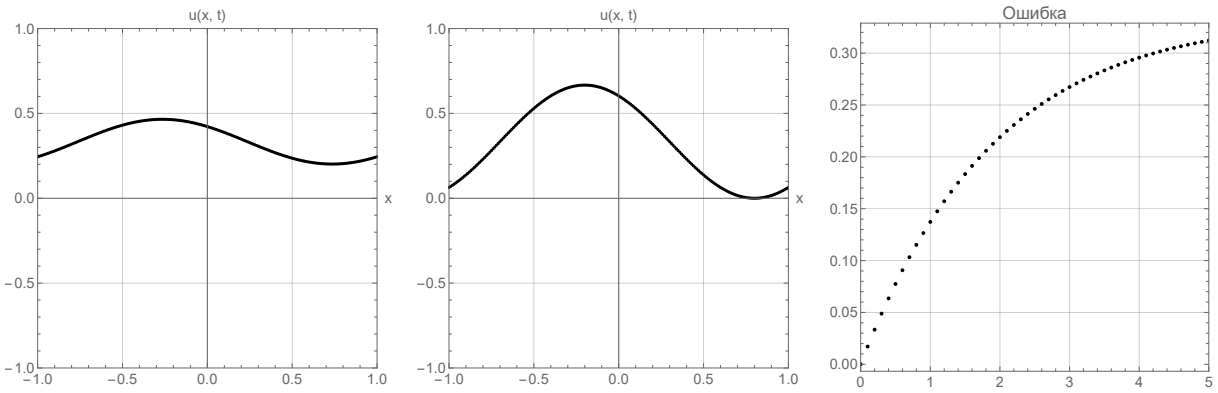
где u - точное решение, y - численное решение. Вычисления проводились на тесте «Косинус». Сетка строилась до момента времени $T = 1$.

Отдельно приведем таблицу с результатами расчетов по схеме (4). По таблице можно сделать вывод о неустойчивости данной схемы.

Таблица 2. Норма ошибки разностной схемы (4) при различных параметрах

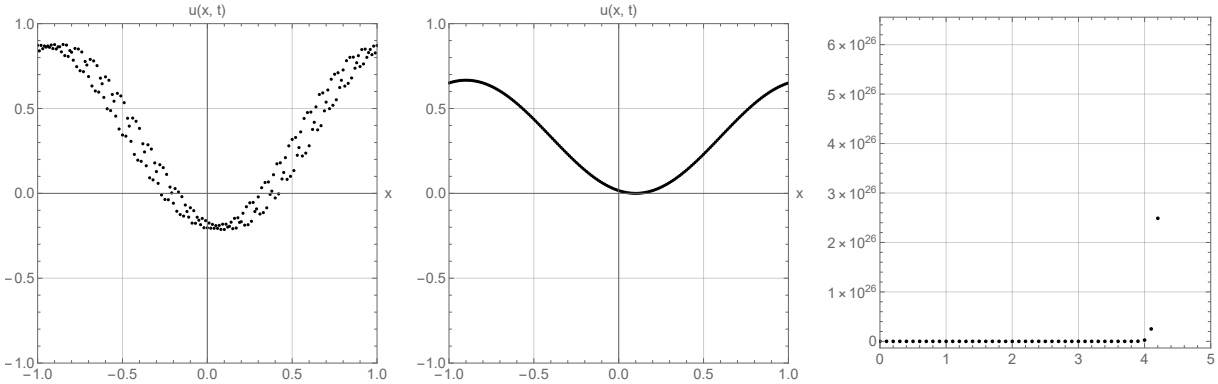
Схема / h, τ	Неявная ЛДР 2-го пор.
$h = 0.1, \tau = 0.01, \gamma = 0.1$	10^{12}
$h = 0.1, \tau = 0.02, \gamma = 0.2$	10^{19}
$h = 0.1, \tau = 0.05, \gamma = 0.5$	10^{18}
$h = 0.1, \tau = 0.1, \gamma = 1$	10^{15}
$h = 0.05, \tau = 0.025, \gamma = 2$	10^{22}

Ниже приведены иллюстрации полученных результатов для некоторых схем.



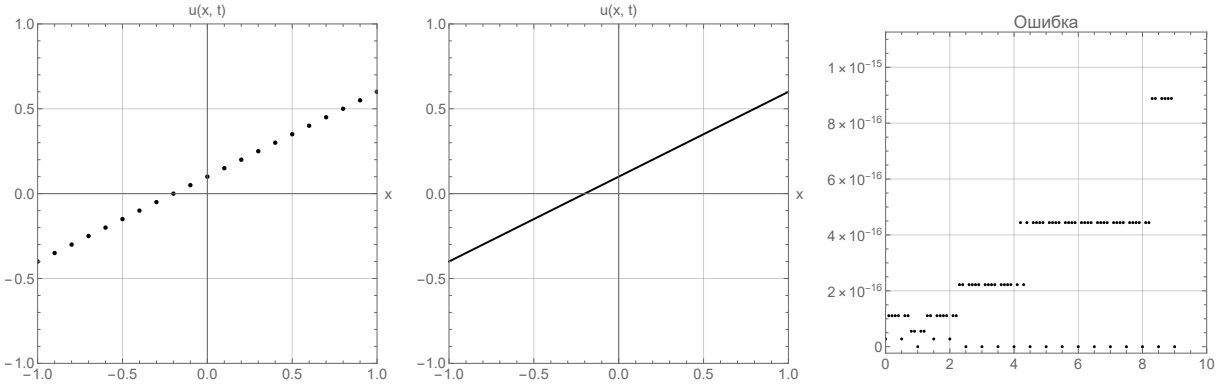
(a) Численное решение , при $h = 0.01, \tau = 0.1$ (b) Точное решение (c) Изменение ошибки со временем

Рис. 3. Тест косинус по схеме (2), $t = 2$



(a) Численное решение при $h = 0.01, \tau = 0.1, t = 20$ (b) Точное решение (c) Изменение ошибки со временем

Рис. 4. Иллюстрация расходимости схемы (1) на тесте косинус, $t = 1.3$

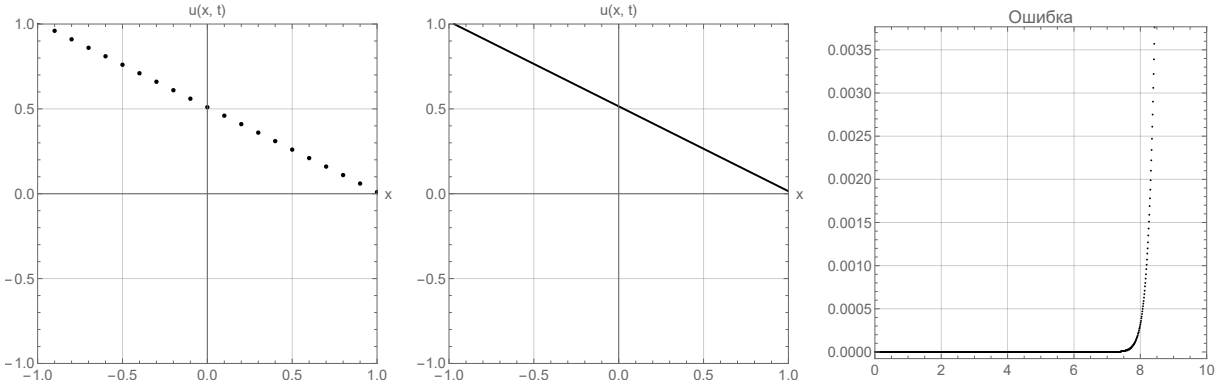


(a) Численное решение при $h = 0.1, \tau = 0.1$

(b) Точное решение при тех же параметрах

(c) Изменение нормы со временем

Рис. 5. Тест левый треугольник по схеме (1), $t = 1.8$



(a) Численное решение при $h = 0.1, \tau = 0.01$

(b) Точное решение при тех же параметрах

(c) Изменение ошибки со временем

Рис. 6. Тест правый треугольник по схеме (6), $t = 0.5$

Для визуализации решения использовался математический пакет Wolfram Mathematica, в котором были созданы функции для создания анимации изменения графика во времени. Также в этом пакете были созданы функции для визуализации аналитического решения и для сравнения его с численным.

4. Метод улучшения решения с помощью нейронной сети

Основной целью данной работы является создание программы, которая по известному точному решению сможет улучшить входное неточное. Для реализации такой программы был выбран метод, заключающийся в распознавании изображения нейронной сетью. Алгоритм модели заключается в следующем: в сеть, обученную на

исключительно точных решениях, подается на вход неточное, которое она классифицирует. Как результат, нейросеть выдает метку соответствующего точного решения.

4.1. Модель сверточной нейронной сети

Сверточная нейронная сеть - модель нейронной сети, предложенная в 1988 году Яном Лекуном, которая и сегодня является одной из самых эффективных для распознавания образов на изображении. Структура сети однонаправленная и для обучения в большинстве случаев используется метод обратного распространения ошибки. [2]

Основной особенностью сверточной нейронной сети является наличие комбинаций слоев свертки и пуллинга. Преимущества модели, заключается в способности улавливать пространственные взаимосвязи на изображении, что позволяет классифицировать одни и те же объекты независимо от их расположения и размера на изображении. Не менее значимым является меньшее количество настраиваемых весов, так как один набор фильтров как весы используется целиком для всего изображения, вместо того, чтобы создавать для каждого пикселя входного изображения свои коэффициенты как это делает перцептрон. Благодаря такой особенности, сверточная нейронная сеть требует меньшее количество времени для обучения. Также данная модель устойчива к повороту и сдвигу распознаваемого изображения.

Для реализации своей модели выберем библиотеку $TensorFlow$ в силу ее простоты и полноты документации. Опишем основные шаги при создании модели сверточной нейросети с помощью библиотеки TensorFlow.

Модель нейросети представляет собой объект класса из TensorFlow, который мы назовем $Model$. При создании модели необходимо указать ее тип как $sequental$, что означает, что процесс вычислений будет последовательно идти по слоям. После создания модели необходимо добавить в нее слои, что можно сделать с помощью метода $.add$. Добавляя нужное нам количество уже встроенных в библиотеку слоев свертки $Conv2D$, указываем количество фильтров, их размерность и функцию активации. Добавляем слои пуллинга $MaxPooling2D$, указывая в параметрах слоя необходимую размерность подвыборки. Последними слоями будут полносвязные слои $Dense$, где указываем количество нейронов и также функцию активации. Количество вариантов ответа нейросети обозначим $output$.

Наконец, соберем модель с помощью метода $.compile$, в котором необходимо указать параметры функции оптимизатора, потерь и метрику обучения. Для использования модели остается только ее скомпилировать с помощью метода $.compile$.

Библиотека позволяет сохранить параметры скомпилированной модели, что делает возможным встраивать уже созданный алгоритм распознавания в другие про-

граммы. Таким образом можно создать свою модель нейронной сети для совершенно разных задач.

Листинг 1. Модель нейронной сети

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential()

model.add(Conv2D(32, (4, 4), activation='relu', input_shape=(128, 128, 1)))
model.add(MaxPooling2D(pool_size=(4, 4)))

model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(output + 50, activation='relu'))
model.add(Dense(output, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

4.2. Распознавание решения

Чтобы распознать на изображении определенное решение уравнения, необходимо произвести расчет модели нейронной сети и получить ее ответ. Этим ответом будет метка, которая была назначена во время обучения каждому решению. Зная метку, можно поставить в соответствие исходное для обучения изображение, которое и будет являться улучшенным решением.

Для работы такого подхода была реализована программа на языке программирования Python, которая включает в себя все прежде описанные действия. На вход программа получает изображение (путь к изображению в операционной системе), которое она преобразовывает в объект, который можно подать в модель нейронной сети для получения предсказания метки. Программа вызывает заранее обученную нами модель нейронной сети и получает ее ответ. При обучении модели автоматически создается кодификатор, который ставит соответствие между каждой меткой и изображении, на котором производилось обучение. Программа использует этот коди-

фикатор и возвращает тем самым изображение, которое является точным решением.

4.3. Результаты работы программы

Для проверки работы программы был созданы наборы изображений для обучения и тестирования. Наборы содержали решения для промежутка времени $t \in [0, 1]$ с шагом 0.1. Рассматривалась область $x \in [-1, 1], y \in [-1, 1]$, а изображения составлялось разрешением в $128 \cdot 128$ пикселей. Для теста использовались решения, полученные с помощью определенных схем, которые решают уравнение хотя бы примерно.

Приведем таблицу результатов точности работы программы в зависимости от шагов по времени и пространству.

Таблица 3. Результаты тестирования модели нейронной сети

Номер теста	Шаг по времени τ	Шаг по пространству h	Точность программы %
1	0.1	0.1	82
2	0.05	0.1	84
3	0.05	0.05	89
4	0.01 ‘	0.01	93

Стоит отметить, что результаты могут отличатся в зависимости от множества факторов, к которым относятся выбор коэффициентов сети, количество эпох обучения и особенности выбора набора для обучения.

5. Актуальность и перспективы задачи

Технологии нейронных сетей пробираются во все сферы деятельности человека и применение их в математике может привести к неожиданно полезным результатам. Благодаря распознаванию графика функции нейронной сетью, можно научить компьютер определять одновременно огромное количество графиков за сравнено небольшое количество времени, чтобы применять данный метод в совершенно разных сферах науки и техники. Перспективой развития данного метода является создания такого же алгоритма восстановления графика решений, но с заранее неизвестными данными о типе графика. Улучшение распознавания типов графиков приведет к возможностям классифицировать практически любое решение уравнения и уточнить его на любом малом промежутке исходную кривую.

Заключение

Таким образом в ходе работы были изучены численные методы решения уравнения переноса, проведено тестирование численных методов на 5 тестах, а также созданы несколько программ. Была создана программа, которая по неточному решения уравнения переноса возвращает точное. Была реализована программа на языке программирования C++17 для численного решения уравнения переноса различными схемами, а также реализованы 5 тестов. Реализована модель сверточной нейронной сети для распознавания решений уравнения на языке программирования Python 3.9.2 с помощью библиотеки машинного обучения TensorFlow 2.15.0. Изображения графиков решения создавались с помощью математического пакета Wolfram Mathematica 14. с реализацией соответствующих функций.

Как итог, реализована программа, которая с помощью модели сети улучшает неточное решение данного уравнения до точного.

Список использованных источников

1. Галанин М.П., Савенков Е.Б. Методы численного анализа математических моделей. М.: Изд-во МГТУ им. Н.Э. Баумана. 2018. 592 с.
2. Tariq Rashid Make Your Own Neural Network // CreateSpace Independent Publishing Platform; 1st edition SAND96-0583 (March 31, 2016)
3. TensorFlow documentation. URL: https://www.tensorflow.org/?hl=ru (Дата обращения 08.06.2024)
4. Гафаров Ф.М. Искусственные нейронные сети и приложения. г.Казань: Изд-во Казан. ун-та, 2018. 121 с