МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

Институт автоматики и информационных технологий

Кафедра «Высшая математика и моделирование»

Асыл Жанат

Решение обратной задачи колебаний нити в вязкой среде

ДИПЛОМНАЯ РАБОТА

6B06103 – Математическое и компьютерное моделирование

Алматы 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

Институт автоматики и информационных технологий

Кафедра «Высшая математика и моделирование»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ВМиМ
канд. физ-мат. наук,
ассоциированный профессор
_____ Г. А. Тулешева
«06» ИЮНЯ 2023 г.

ДИПЛОМНАЯ РАБОТА

На тему: «Решение обратной задачи колебаний нити в вязкой среде»

6В06103 – Математическое и компьютерное моделирование

Выполнил Асыл Жанат

Рецензент
канд. физ-мат. наук,
PhD in Computer Science, профессор Школы информационных технологий, Руководитель Проектных Групп, АО «КБТУ»
_____ А. Ж. Акжалова
"05" 06 2023 г.

Научный руководитель
канд. физ-мат. наук, ассоциированный профессор
_____ Р. Н. Зимин
"06" ИЮНЯ 2023 г.

Алматы 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество «Казахский национальный исследовательский технический университет имени К.И.Сатпаева»

Институт автоматики и информационных технологий

Кафедра «Высшая математика и моделирование»

6В06103 – Математическое и компьютерное моделирование

УТВЕРЖДАЮ
Заведующий кафедрой ВМиМ
канд. физ-мат. наук,
ассоциированный профессор
_____ Г. А. Тулешева
«06» июня 2023 г.

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся: Асыл Ж.
Тема: «Решение обратной задачи колебаний нити в вязкой среде»
Утверждена приказом проректора по академической работе: № 408-П/Ө
от «23» ноября 2022 г.
Срок сдачи законченной работы: «22» мая 2023 г.
Исходные данные к дипломной работе: техническая документация к Python, TensorFlow и Keras
Краткое содержание дипломной работы:
А) Теоретическая часть;
Б) Математическая постановка задачи;
В) Решение прямой задачи;
Г) Выбор архитектуры нейронной сети.
Рекомендуемая основная литература: из 2 наименований.

ГРАФИК
подготовки дипломной работы

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
Построение математической модели обратной задачи	16.03.23	Выполнено
Решение прямой задачи и создание базы данных	23.03.23	Выполнено
Поиск оптимальной архитектуры для нейронной сети	27.04.23	Выполнено
Заключение	04.05.23	Выполнено

Подписи
консультантов и нормоконтролёра на законченную дипломную работу
с указанием относящихся к ним разделов работы

Наименования разделов	**Консультанты, Ф.И.О. (уч. степень, звание)**	**Дата подписания**	**Подпись**
Построение математической модели обратной задачи	Зимин Р. Н. канд. физ-мат. наук, ассоц. профессор	30.03.23г.	
Решение прямой задачи и создание базы данных	Зимин Р. Н. канд. физ-мат. наук, ассоц. профессор	06.04.23г.	
Поиск оптимальной архитектуры для нейронной сети	Зимин Р. Н. канд. физ-мат. наук, ассоц. профессор	11.05.23г.	
Заключение	Зимин Р. Н. канд. физ-мат. наук, ассоц. профессор	18.05.23г.	
Нормоконтролёр	Шатманов Ж. Ж. канд. физ-мат. наук, ассоц. профессор	05.06.2023	

Научный руководитель _____ Р. Н. Зимин

Задание принял к исполнению обучающийся _____ Ж. Асыл

Дата «02» февраля 2023 г.

Протокол

о проверке на наличие неавторизованных заимствований (плагиата)

Автор: Асыл Жанат

Соавтор (если имеется):

Тип работы: Дипломный проект

Название работы: Решение обратной задачи колебаний нити в вязкой среде.doc

Научный руководитель: Решат Зимин

Коэффициент Подобия 1: 4.6

Коэффициент Подобия 2: 1.5

Микропробелы: 50

Знаки из здругих алфавитов: 2

Интервалы: 0

Белые Знаки: 311

После проверки Отчета Подобия было сделано следующее заключение:

☐ Заимствования, выявленные в работе, является законным и не является плагиатом. Уровень подобия не превышает допустимого предела. Таким образом работа независима и принимается.

☐ Заимствование не является плагиатом, но превышено пороговое значение уровня подобия. Таким образом работа возвращается на доработку.

☐ Выявлены заимствования и плагиат или преднамеренные текстовые искажения (манипуляции), как предполагаемые попытки укрытия плагиата, которые делают работу противоречащей требованиям приложения 5 приказа 595 МОН РК, закону об авторских и смежных правах РК, а также кодексу этики и процедурам. Таким образом работа не принимается.

☐ Обоснование:

Дата

02.06.23г.

проверяющий эксперт

Протокол

о проверке на наличие неавторизованных заимствований (плагиата)

Автор: Асыл Жанат

Соавтор (если имеется):

Тип работы: Дипломный проект

Название работы: Решение обратной задачи колебаний нити в вязкой среде.doc

Научный руководитель: Решат Зимин

Коэффициент Подобия 1: 4.6

Коэффициент Подобия 2: 1.5

Микропробелы: 50

Знаки из здругих алфавитов: 2

Интервалы: 0

Белые Знаки: 311

После проверки Отчета Подобия было сделано следующее заключение:

☑ Заимствования, выявленные в работе, является законным и не является плагиатом. Уровень подобия не превышает допустимого предела. Таким образом работа независима и принимается.

☐ Заимствование не является плагиатом, но превышено пороговое значение уровня подобия. Таким образом работа возвращается на доработку.

☐ Выявлены заимствования и плагиат или преднамеренные текстовые искажения (манипуляции), как предполагаемые попытки укрытия плагиата, которые делают работу противоречащей требованиям приложения 5 приказа 595 МОН РК, закону об авторских и смежных правах РК, а также кодексу этики и процедурам. Таким образом работа не принимается.

☐ Обоснование:

Дата
02.06.23г.

Заведующий кафедрой

Университеттің жүйе администраторы мен Академиялық мәселелер департаменті директорының ұқсастық есебіне талдау хаттамасы

Жүйе администраторы мен Академиялық мәселелер департаментінің директоры көрсетілген еңбекке қатысты дайындалған Плагиаттың алдын алу және анықтау жүйесінің толық ұқсастық есебімен танысқанын мәлімдейді:

Автор: Асыл Жанат

Тақырыбы: Решение обратной задачи колебаний нити в вязкой среде.doc

Жетекшісі: Решат Зимин

1-ұқсастық коэффициенті (30): 4.6

2-ұқсастық коэффициенті (5): 1.5

Дәйексөз (35): 0.8

Әріптерді ауыстыру: 2

Аралықтар: 0

Шағын кеңістіктер: 50

Ақ белгілер: 311

Ұқсастық есебін талдай отырып, Жүйе администраторы мен Академиялық мәселелер департаментінің директоры келесі шешімдерді мәлімдейді :

☑ Ғылыми еңбекте табылған ұқсастықтар плагиат болып есептелмейді. Осыған байланысты жұмыс өз бетінше жазылған болып санала отырып, қорғауға жіберіледі.

☐ Осы жұмыстағы ұқсастықтар плагиат болып есептелмейді, бірақ олардың шамадан тыс көптігі еңбектің құндылығына және автордың ғылыми жұмысты өзі жазғанына қатысты күмән тудырады. Осыған байланысты ұқсастықтарды шектеу мақсатында жұмыс қайта өңдеуге жіберілсін.

☐ Еңбекте анықталған ұқсастықтар жосықсыз және плагиаттың белгілері болып саналады немесе мәтіндері қасақана бұрмаланып плагиат белгілері жасырылған. Осыған байланысты жұмыс қорғауға жіберілмейді.

Негіздеме:

Күні 02.06.23 г.

Кафедра меңгерушісі

РЕЦЕНЗИЯ

на дипломную работу студента
Асыл Жанат
6В06103 – Математическое и компьютерное моделирование

На тему: Решение обратной задачи колебаний нити в вязкой среде

ЗАМЕЧАНИЯ К РАБОТЕ

Перед дипломантом ставилась такая задача как, решение обратной задачи колебаний нити в вязкой среде.

В первой главе рассмотрена теоретическая часть для решения данного проекта. А также описаны инструменты, которые были использованы.

Во второй главе были описаны шаги проведенные для решения задачи. Первым шагом было построение математической модели прямой задачи. Были определены все силы, действующие на точку, и используя законы Ньютона, была получена математическая модель. Дальнейшим шагом было решение полученной задачи, при условии, что в начальный момент времени нить покоилась, а скорость падающей точки равнялась единице. Следующим шагом было формирование базы данных, на которой пройдет обучение нейронной сети. Последним этапом было нахождение оптимальной архитектуры нейронной сети.

На данном этапе возникли некоторые трудности с подбором оптимальной архитектуры для нейронной сети. Отталкиваясь от рекомендаций с различных ресурсов и официальную документацию к библиотекам, были выбраны различные варианты архитектур нейронных сетей. Далее каждая архитектура тестировалась для выявления наиболее эффективной, а именно выбиралась та, у которой была высокая точность и низкий показатель ошибки.

В результате была получена наиболее эффективная архитектура нейронной сети. Тем самым было найдено решение поставленной обратной задачи.

Графический и текстовый материал оформлен в соответствии с требованиями ГОСТ, предъявляемыми к оформлению учебных работ.

Данный дипломный проект отличает проработанность, научно-исследовательский подход и полноту изложенного теоретического материала. Приведенные исследования доказывают отличную теоретическую подготовку дипломанта.

Оценка работы

Считаю, что дипломный проект заслуживает оценки «95», а студент

Асыл Ж. присуждения академической степени бакалавра по специальности 6B06103 – Математическое и компьютерное моделирование.

Рецензент
к.ф.-м.н., PhD in Computer Science,
профессор Школы информационных
технологий, Руководитель Проектных
Групп, АО «КБТУ»
_____ А. Ж. Акжалова
"___" _____ 2023 г.

ДЕПАРТАМЕНТ ПО РАЗВИТИЮ ПЕРСОНАЛА

НЕКОММЕРЧЕСКОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени К.И.САТПАЕВА»

ОТЗЫВ

НАУЧНОГО РУКОВОДИТЕЛЯ

На дипломную работу
Асыл Жанат
6В06103 – Математическое и компьютерное моделирование

Тема: «Решение обратной задачи колебаний нити в вязкой среде»

Перед Асыл Ж. стояла следующая задача. В вязкой среде имеется невесомая нить (не является струной) с закрепленными концами. На нить падает материальная точка с неизвестной массой. Требуется определить место падения и массу материальной точки. В качестве инструмента решения было предложено использовать нейронные сети, в том числе искусственные.

Для обучения искусственной нейронной сети была создана база данных на основе решения параметрического семейства прямых задач колебания нити. Отталкиваясь от рекомендаций с различных ресурсов и официальную документацию к библиотекам, были выбраны различные варианты архитектур нейронных сетей. Далее каждая архитектура тестировалась для выявления наиболее эффективной, а именно выбиралась та, у которой была высокая точность и низкий показатель ошибки.

В результате была получена наиболее эффективная архитектура нейронной сети. Тем самым было найдено решение поставленной обратной задачи.

Дипломная работа Асыл Жанат состоит трех основных частей: теоретической части, основного результата и заключения.

Считаю, что Асыл Ж. справилась с поставленной ей задачей, ее дипломная работа соответствует выдвигаемым к таким работам требованиям. Считаю, что дипломная работа заслуживает оценки «96», а Асыл Ж. — присуждения ей академической степени бакалавра по специальности 6В06103 — «Математическое и компьютерное моделирование».

Научный руководитель
Ассоц. профессор кафедры ВМиМ,
к.ф.-м.н.
(должность, уч. степень, звание)
Зимин Р.Н.
(подпись)
«05» июня 2023 г.

АҢДАТПА

Бұл дипломдық жұмыста, екі жағынан бекітілген жіптің үстіне массасы бар нүктенің түскен кезіндегі, бұл механикалық жүйенің тұқыр ортадағы тербелісі қарастырылады. Жұмыстың мақсаты нейрондық желі арқылы кері есепті шешу болып табылады. Кіріс параметрлері ретінде меншікті жиілік, жіптің бекітілген ұштарындағы ауытқу бұрыштары, жіптің керілу күші, уақыт және период қолданылады, ал шығыс параметрлері нүктенің жіпке түскен нүктесі және массасы.

Дипломдық жұмыс теориялық бөлімді және атқарылған жұмыстың негізгі нәтижесін қамтиды. Бірінші кезең, тура есептің математикалық моделін құру болып табылады. Келесі кезең, құрастырылған тура есепті шешу және дерекқор құру. Соңғы кезең – нейрондық желі үшін, кері есепті шешетін, оңтайлы құрылымды табу.

АННОТАЦИЯ

В данной дипломной работе рассматривается колебание механической системы, состоящей из закрепленной нити и точечной массой в вязкой среде. Цель работы – решение обратной задачи, с помощью нейронной сети. В качестве входных параметров выступают: собственная частота, углы отклонения на концах крепления нити, сила натяжения нити, время и период колебаний, а искомыми значений являются выходные параметры: точечная масса и ее точка падения на нить.

Дипломная работа содержит теоретическую часть, на которую опиралась работа и основной результат. Первым этапом является построение математической модели прямой задачи. Далее решение полученной задачи и составление базы данных. Заключительный этап — это нахождение оптимальной архитектуры для нейронной сети, который решает нашу обратную задачу.

ABSTRACT

This thesis considers the mechanic system of oscillation of a string with a mass in the viscous domain. The purpose of the work is to solve the inverse problem, with using a neural network. The input parameters are natural frequency, deflection angles at the ends of the string, string tension force, time and period of oscillation, and the output parameters we are looking for are the mass and its dropping point on the string.

The thesis contains the theoretical part on which the work was based and the main result. The first step is to find a mathematical model of a direct problem. The next step is finding the solution to the model and the generation of the database. The final step is finding the optimal architecture for a neural network, which will solve our inverse problem.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Машинное обучение, бесспорно, является одной из самых влиятельных и мощных технологий в современном мире. Сегодня машинное обучение работает повсюду вокруг нас. Когда мы взаимодействуем с банками, совершаем покупки онлайн или пользуемся социальными сетями в игру вступают алгоритмы машинного обучения, которые делают наш процесс эффективным, плавным и безопасным. Машинное обучение и технологии, связанные с ним, быстро развиваются, и мы только начинаем знакомиться с его возможностями. За последнее десятилетие в попытках развить искусственный интеллект, были разработаны всевозможные задачи. От таких простых вещей как распознавание речи, рукописного текста и изображений до самоуправляемых автомобилей и умных помощников по дому.

В данной дипломной работе нейронная сеть используется для решения задачи из классической механики. Цель работы – решение обратной задачи, которая состоит в том, чтобы с помощью нейронной сети на основе таких параметров как: период колебаний, углы отклонения на концах крепления нити, сила натяжения нити, время и собственная частота, определять точечную массу и ее точку падения на нить.

1 Теоретическое представление о работе

1.1 Классическая механика, затухающие гармонические колебания

Колебания являются особенно важной частью механики, и даже физики в целом. Это объясняется их широким распространением и практической важностью проблем колебаний.

1.1.1 Тело на пружине

Предположим, что тело массой m прикреплено к одному концу легкой пружины. Другой конец пружины прикреплен к неподвижной точке A на гладком горизонтальном столе, и тело скользит по столу по прямой линии, проходящей через A. Пусть x – перемещение, а V – скорость тела в момент времени t, как показано на рисунке 1.1.

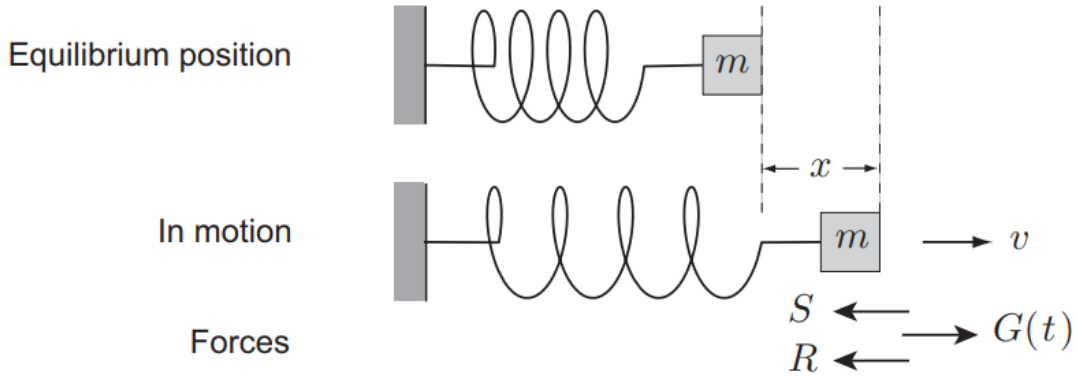

Рисунок 1.1 – Корпус прикреплен к одному концу легкой пружины и перемещается по прямой линии

Рассмотрим теперь силы, действующие на тело. Когда пружина растягивается, она прикладывает силу упругости S в направлении, противоположном растяжению. Кроме того, тело может столкнуться с силой сопротивления R, действующей в направлении, противоположном его скорости. Наконец, может существовать внешняя движущая сила $G(t)$, которая является заданной функцией времени. Тогда уравнение движения тела имеет вид

$$m\frac{dv}{dt} = -S - R + G(t) \tag{1.1}$$

Сила упругости определяется конструкцией пружины и удлинителем x. При достаточно малых деформациях зависимость между S и x приблизительно линейна, то есть,

$$S = \alpha x \tag{1.2}$$

где α – положительная константа, называемая прочностью пружины.

Мощная пружина, такая как те, что используются в автомобильных подвесках, имеет большое значение α; пружина за дверным звонком имеет небольшое значение α. Формула (1.2) называется законом Гука, а пружина, которая в точности подчиняется закону Гука, называется линейной пружиной.

Сила сопротивления R зависит от физического процесса, который вызывает сопротивление. Что касается сопротивления жидкости, то законы линейного или квадратичного сопротивления тоже могут быть подходящими. Однако ни один из этих законов не отражает силу трения, создаваемую грубым столом. Поэтому здесь предполагается закон линейного сопротивления,

$$R = \beta v \tag{1.3}$$

где β – положительная константа, называемая константой сопротивления; это мера силы сопротивления. Нет смысла скрывать тот факт, что основная причина для предположения о линейном сопротивлении заключается в том, что (вместе с законом Гука) это приводит к линейному уравнению движения, которое может быть решено явно. Тем не менее это дает представление об общем эффекте всех сопротивлений и на самом деле уместно, когда сопротивление возникает из-за медленного вязкого потока (например, автомобильные амортизаторы); это также уместно в аналоге электрической цепи, где это эквивалентно закону Ома.

С учетом закона Гука и линейного сопротивления уравнение движения (1.1) для тела становится

$$m\frac{d^2x}{d^2t} + \beta\frac{dx}{dt} + \alpha x = G(t) \tag{1.4}$$

где α – постоянная пружины, β – постоянная сопротивления, а G(t) – заданная движущая сила. Это линейное обыкновенное дифференциальное уравнение второго порядка с постоянными коэффициентами для неизвестного смещения *x(t)*. Можно продолжить решение этого уравнения в его нынешнем виде, но алгебра значительно упрощается за счет введения двух новых констант Ω и *K* (вместо α и β), определяемых соотношением

$$\alpha = m\Omega, \tag{1.5}$$

$$\beta = 2mK \tag{1.6}$$

Тогда уравнение движения тела становится

$$\frac{d^2x}{d^2t} + 2K\frac{dx}{dt} + \Omega^2 x = F(t) \tag{1.7}$$

где $F(t) = G(t)/m$ – движущая сила на единицу массы. Это стандартная форма уравнения движения для тела.

Любая система, приводящая к уравнению такого вида, называется линейным осциллятором с затуханием. Когда сила *F(t)* отсутствует, колебания считаются свободными; когда сила *F(t)* присутствует, колебания считаются управляемыми.

1.1.2 Классическое обыкновенное гармоническое движение

Линейный осциллятор, который одновременно не затухает и не приводится в движение, называется классическим линейным осциллятором. Это простейший случай, но, возможно, самая важная система в физике. Уравнение (1.7) сводится к

$$\frac{d^2x}{d^2t} + \Omega^2 x = 0 \tag{1.8}$$

которое называется уравнением простого гармонического движения.

Решение ищется в виде $x = e^{\lambda t}$. Тогда λ должно удовлетворять уравнению

$$\lambda^2 + \Omega^2 = 0 \tag{1.9}$$

что дает $\lambda = \pm i\Omega$. Таким образом, находится пара комплексных решений

$$x = e^{\pm i\Omega t} \tag{1.10}$$

которые формируют основу для пространства комплексных решений. Действительной и мнимой частями первого комплексного решения являются

$$x = \begin{cases} \cos \Omega t \\ \sin \Omega t \end{cases} \tag{1.11}$$

данные функции формируют основу для пространства реальных решений. Таким образом, общее вещественное решение уравнения простого гармонического движения является

$$x = A \cos \Omega t + B \sin \Omega t \tag{1.12}$$

где A и B – вещественные произвольные константы. Это общее решение может быть записано в альтернативной форме

$$x = C \cos(\Omega t - \gamma) \tag{1.13}$$

где C и γ – вещественные произвольные константы и $C > 0$.

Общая форма движения наиболее легко следует из формы (1.13) и показана на рисунке 1.2. Это называется простым гармоническим движением (SHM). Тело

совершает бесконечно много колебаний с постоянной амплитудой *C*; постоянная γ – это просто "фазовый коэффициент", который сдвигает весь график на λ/Ω в направлении *t*. Поскольку косинусная функция повторяется, когда аргумент Ω*t* увеличивается на 2π, из этого следует, что период колебаний задается формулой

$$\tau = \frac{2\pi}{\Omega} \tag{1.14}$$

Величина Ω, которая связана с частотой *v* посредством Ω = *2πv*, называется угловой частотой колебаний.

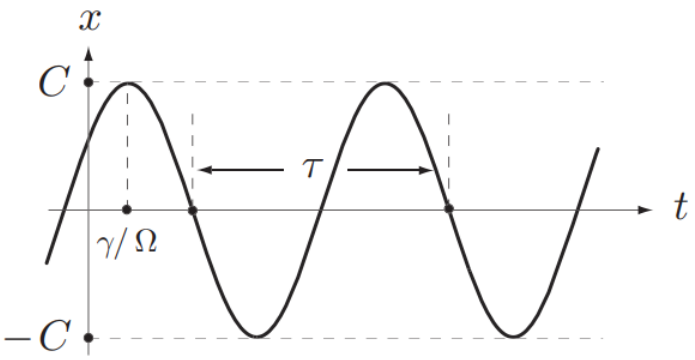

Рисунок 1.2 – Классическое обыкновенное гармоническое движение

1.1.3 Затухающее обыкновенное гармоническое движение

В задаче, которая рассматривается в данной работе, затухание присутствует, но внешняя сила отсутствует, и общее уравнение (1.4) сводится к уравнению затухающего простого гармонического движения

$$\frac{d^2x}{d^2t} + 2K\frac{dx}{dt} + \Omega^2 x = 0 \tag{1.15}$$

Решение ищется в виде $x = e^{\lambda t}$. Тогда λ должно удовлетворять уравнению

$$\lambda^2 + 2K\lambda + \Omega^2 = 0 \tag{1.16}$$

то есть

$$(\lambda + K)^2 = K^2 - \Omega^2 \tag{1.17}$$

В зависимости от того, является ли $K < \Omega$, $K = \Omega$ или $K > \Omega$, возникают разные случаи. Эти случаи приводят к различным видам решений и должны рассматриваться отдельно.

Рассмотрим периодически затухающие колебания (under-damping), когда $K < \Omega$. В этом случае уравнение для λ записывается в виде

$$(\lambda + K)^2 = -\Omega_D^2 \tag{1.18}$$

где $\Omega_D = (\Omega^2 - K^2)^{1/2}$, положительное вещественное число. Тогда значения λ равны $\lambda = -K \pm i\Omega_D$. Таким образом, можно найти пару комплексных решений

$$x = e^{-Kt}e^{\pm i\Omega_D t} \tag{1.19}$$

которые формируют основу для пространства комплексных решений. Действительной и мнимой частями первого комплексного решения являются

$$x = \begin{cases} e^{-Kt}\cos(\Omega_D t) \\ e^{-Kt}\sin(\Omega_D t) \end{cases} \tag{1.20}$$

и эти функции формируют основу для пространства реальных решений. Таким образом, общее вещественное решение уравнения затухающего простого гармонического движения в этом случае является

$$x = e^{-Kt}(A\cos\Omega_D t + B\sin\Omega_D t) \tag{1.21}$$

где A и B – вещественные произвольные константы. Это общее решение может быть записано в альтернативной форме

$$x = Ce^{-Kt}\cos(\Omega_D t - \gamma) \tag{1.22}$$

где C и γ – вещественные произвольные константы и $C > 0$.

Общая форма движения наиболее легко следует из формы (1.22) и показана на рисунке 1.3. Это называется простым гармоническим движением с периодическим затуханием. Тело по-прежнему совершает бесконечно много колебаний, но теперь данные колебания имеют экспоненциально убывающую амплитуду Ce^{-Kt}. Предположим, что период τ колебаний определен так, как показано на рисунке 1.3. Введение затухания уменьшает угловую частоту колебаний с Ω до Ω_D, что увеличивает период колебаний с $2\pi/\Omega$ до

$$\tau = \frac{2\pi}{\Omega_D} = \frac{2\pi}{(\Omega^2 - K^2)^{1/2}} \tag{1.23}$$

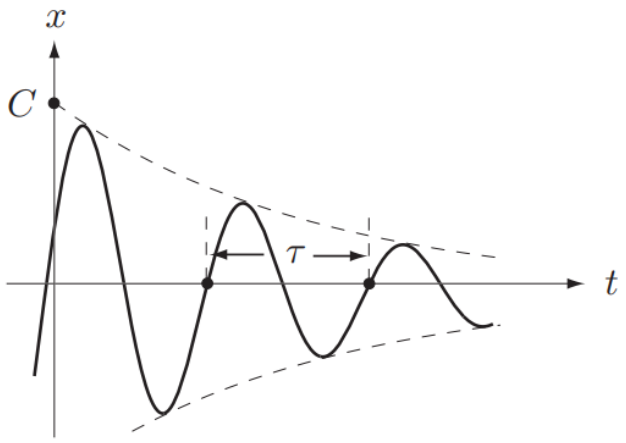

Рисунок 1.3 – Обыкновенное гармоническое движение с переодическим затуханием

Так же можно рассмотреть как выглядит чрезмерное затухание (over-damping), когда $K < \Omega$. В этом случае уравнение для λ записывается в виде

$$(\lambda + K)^2 = \delta^2 \tag{1.24}$$

где $\delta = (K^2 - \Omega^2)^{1/2}$, положительное вещественное число. Тогда значения λ равны $\lambda = -k \pm \delta$. Таким образом, можно найти пару реальных решений

$$x = e^{-Kt} e^{\pm \delta t} \tag{1.25}$$

которые формируют основу для пространства реальных решений. Таким образом, общее вещественное решение уравнения простого гармонического движения с затуханием в этом случае является

$$x = e^{-Kt}\left(Ae^{\delta t} + Be^{-\delta t}\right) \tag{1.26}$$

где A и B – вещественные произвольные константы.

Три типичные формы движения показаны на рисунке 1.4. Это называется простым гармоническим движением с избыточным затуханием. Немного удивительно, но тело вообще не колеблется. Например, если тело выходит из состояния покоя, то оно просто дрейфует обратно к положению равновесия. С другой стороны, если тело проецируется в положение равновесия с достаточной скоростью, то оно проходит положение равновесия один раз, а затем возвращается к нему с другой стороны [1].

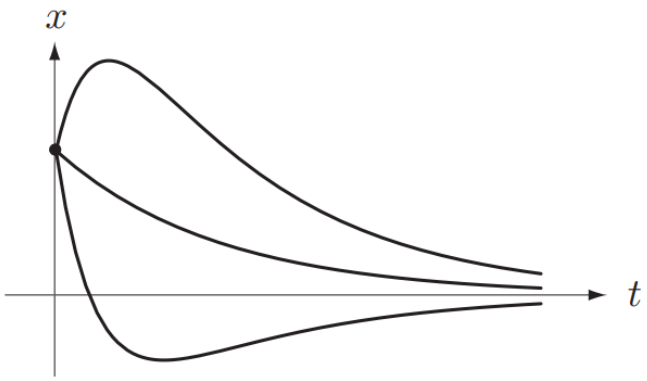

Рисунок 1.4. – Три типичных случая, чрезмерно затухающего простого гармонического движения.

1.2 Машинное обучение и искусственная нейронная сеть

Машинное обучение, бесспорно, является одной из самых влиятельных и мощных технологий в современном мире. Настолько мощная, что люди далеки от того, чтобы узреть весь его потенциал. За последние пару лет термины как “Машинное обучение”, “Искусственный интеллект”, “Нейросеть” стали, так сказать, модными и встречаются почти везде. Причина этого в том, что прогресс никогда не стоит на месте: например, возросла вычислительная мощность процессоров, были разработаны более совершенные алгоритмы, благодаря большому объему обрабатываемых приложениями данных искусственный интеллект стал более доступен для широкой массы.

За последнее десятилетие в попытках развить искусственный интеллект, были разработаны всевозможные задачи. От таких простых вещей как распознавание речи, рукописного текста и изображений, до самоуправляемых автомобилей и умных помощников по дому. Машинное обучение также используется в таких сферах как медицина и финансы. Например, как распознавать раковые ткани или при расследовании мошенничества и проверки кредитоспособности. Машинное обучение сегодня настолько широко распространено, что люди сами того не подозревая, пользуются им ежедневно.

В данной работе для решения поставленной задачи используются нейронные сети, а не машинное обучение. Главная особенность нейронной сети в том, что данная сеть работает как нейроны в мозгу человека. В моделях машинного обучения нам нужно вводить созданные вручную алгоритмы в качестве входных данных для выполнения прогнозирования, а в нейронных сетях, модель извлекает функции из входных данных и предсказывает на их основе выходные данные. Точно так же, как биологическая нейронная сеть, искусственная нейронная сеть постоянно учится и обновляет свои знания и понимание окружающей среды на основе опыта, с которым столкнулась. Нейронные сети могут помочь нам понять взаимосвязи между сложными структурами данных. Нейронные сети могут использовать полученные знания

для прогнозирования поведения сложных структур. Поэтому в данном случае нейронная сеть подходит больше.

1.3 Инструменты для работы с нейронной сетью

1.3.1 Python

Для создания искусственной нейронной сети использовался язык программирования Python, а также библиотека TensorFlow. Python очень легкий в изучении язык программирования, который так же является очень универсальным и подходит для решения различных задач. Интуитивно понятен и прост в использовании. Код вполне подходит для написания практически на любом редакторе и представляет собой простой текстовый файл. Данный мощный язык программирования был создан Гвидо ван Россумом в 1991 году. Python был разработан как интерпретируемый язык программирования для общего использования. Интерпретируемый язык означает, что программный код преобразуется в байт-код и затем выполняется интерпретатором, которым в данном случае является виртуальная машина Python. С годами популярность и функциональность Python росли, что привело к гибкости данного языка программирования в использовании. Возможность быстро вносить и тестировать изменения в программный код программного обеспечения упрощает задачу, с которой при необходимости можно справиться “на лету”. Python работает практически на всех системных архитектурах и может использоваться для широкого спектра приложений в различных областях, от веб-разработки до машинного обучения. Помимо своей универсальности, данный язык также удобен для начинающих, поэтому считается одним из самых популярных доступных языков программирования. Python включает в себя модули, исключения, динамическую типизацию, динамические типы данных очень высокого уровня и классы. Имеет интерфейсы ко многим системным вызовам и библиотекам, а также к различным оконным системам и является расширяемым на C или C++. Также может использоваться в качестве языка расширения для приложений, которым требуется программируемый интерфейс. Наконец, Python переносим, что значит, что данный язык программирования работает на многих версиях Unix, включая Linux и macOS, а также на Windows [2]. Python не только прост в использовании, но и в освоении. Эти два фактора привели к тому, что этот язык стал основным для начинающих, изучающих разработку программного обеспечения. Кроме того, универсальность данного языка как языка программирования общего назначения делает Python подходящим для нужд многих отраслей промышленности.

Но это не главные причины почему для создания нейронной сети выбор пал на язык программирования Python. Для более быстрой и эффективной работы с нейронной сетью было принято решение использовать библиотеку

TensorFlow. А так как главный язык, который поддерживает TensorFlow это Python, то было принято решение использовать данный язык программирования.

1.3.2 TensorFlow

TensorFlow – программная библиотека открытого вида разработана компанией Google для машинного обучения, построения и тренировки нейронных сетей. Целью обычно служит классификация и автоматическое нахождение образов с точностью до человеческой оценки. При создании и обучении моделей не жертвуется производительность или скорость, платформа предоставляет нужный контроль и гибкость благодаря функциям для создания сложных сетевых топологий. Платформа TensorFlow это более или менее простой инструмент для создания нейронных сетей любой сложности. Подходит для использования как для тех, кто только начинает свое знакомство с машинным обучением и нейронной сетью, так и для тех, кто давно освоился в этой области. Так же в сети можно встретить достаточное количество различных примеров и уже готовых моделей, что облегчает обучение и понимание данной библиотеки. Главный язык, который поддерживает TensorFlow это Python, но также помимо него есть отдельные дополнения и для таких языков как C, C++, C#, Java, Go, Swift, R [3].

Все вычисления в TensorFlow используют тензор, собственно, поэтому платформа и называется TensorFlow. Тензор – это вектор или матрица из n измерений, представляющая типы данных. Значения в тензоре содержат идентичные типы данных с известной формой. Эта форма и есть размерность матрицы. Машинное обучение работает с большими объемами данных в очень сложных форматах. Тензоры обеспечивают отличное решение для работы с этими разнообразными данными простым способом.

1.3.3 Keras

Keras – это программный интерфейс приложения (API), предназначенный для людей, а не для машин. Keras следует рекомендациям по снижению когнитивной нагрузки: предлагает согласованные и простые API-интерфейсы, сводит к минимуму количество пользовательских действий, необходимых для обычных случаев использования, и предоставляет четкие и действенные сообщения об ошибках. Цель Keras – предоставить помощь любому разработчику, который хочет выпускать приложения на базе машинного обучения. Keras фокусируется на скорости отладки, элегантности и лаконичности кода, ремонтопригодности и развертываемости. При выборе Keras, кодовая база становится меньше, более читабельной и легче для повторного использования. Построенная поверх платформы TensorFlow, Keras – это мощная в отрасли платформа, которая может масштабироваться до больших кластеров графических процессоров или целого модуля TPU. Это не только возможно, но и легко. Keras является центральной частью тесно связанной

экосистемы TensorFlow, охватывающей каждый этап рабочего процесса машинного обучения, от управления данными до обучения гиперпараметрам и решений для развертывания. Keras используется CERN, NASA, NIH и многими другими научными организациями по всему миру. Обладает низкоуровневой гибкостью для реализации произвольных исследовательских идей, предлагая при этом дополнительные высокоуровневые функции удобства для ускорения циклов экспериментов. Keras используют около 200 000 пользователей, начиная от академических исследователей и инженеров как в стартапах, так и в крупных компаниях и заканчивая аспирантами и любителями. Данная библиотека используется в Google, Netflix, Uber, Microsoft, Square и многих стартапах, работающих над широким спектром задач машинного обучения [4].

2 Основной результат

2.1 Математическая постановка задачи колебания нити в вязкой среде

Имеется невесомая нить, закрепленная на обоих концах. На нить падает точка некоторой массы, тем самым возбуждая колебания нити. Система находится в вязкой среде, за счет чего рассматриваются затухающие колебания. На рисунке 2.1 показана данная система, которая совершает колебания при падении массы m на нить.

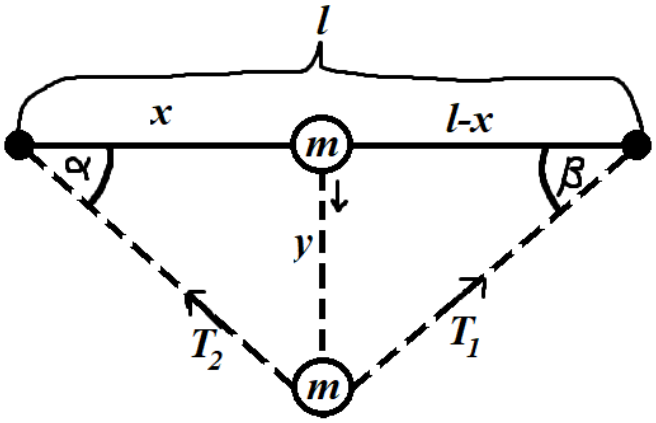

Рисунок 2.1 – Закрепленная на двух концах нить

Для построения уравнения данной системы понадобятся II и III законы Ньютона. Второй закон Ньютона гласит что под действием силы тело приобретает такое ускорение, что его произведение на массу тела равно приложенной силе

$$F = ma \tag{2.1}$$

Третий закон отражает тот факт, что силы, с которыми действуют друг на друга два тела, равны по величине и противоположны по направлению

$$F = -F \tag{2.2}$$

Колебания механической системы происходят по оси y. Рассмотрим направление сил на ось y

$$Oy: ma = -(\bar{T}_1 + \bar{T}_2) \tag{2.3}$$

Из тригонометрии известно, что синус угла – это отношение противолежащего катета на гипотенузу. По рисункам 2.2 и 2.3 видно, что

$\sin\alpha = \frac{y}{T_2}$ и $\sin\beta = \frac{y}{T_1}$. Проецируя силы натяжения на ось y, получаем

$$Oy: ma = -(T\sin\alpha + T\sin\beta) \quad (2.4)$$

Предполагаем, что углы отклонения малы и синус можно заменить на тангенс

$$ma = -T(\tan(\alpha) + \tan(\beta)) \quad (2.5)$$

Катет, противолежащий углу α, равен произведению второго катета на $\tan(\alpha)$, это показано на рисунке 2.2

$$y = x\tan(\alpha), \quad (2.6)$$

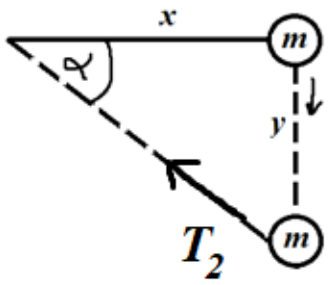

Рисунок 2.2 – Левая сторона нити от точки падения

Так же и с углом β, как показано на рисунке 2.3

$$y = (l - x)\tan(\beta) \quad (2.7)$$

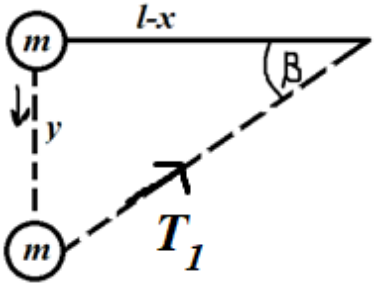

Рисунок 2.3 – Правая сторона нити от точки падения

Из формул (2.6) и (2.7) получаем формулы для $\tan(\alpha)$ и $\tan(\beta)$

$$\tan(\alpha) = \frac{y}{l-x}, \quad (2.8)$$

$$\tan(\beta) = \frac{y}{x} \tag{2.9}$$

Подставляем формулы (2.8) и (2.9) в уравнение (2.5) и делим на массу m

$$a = -\frac{T}{m}\left(\frac{y}{x} + \frac{y}{l-x}\right) \tag{2.10}$$

Ускорение – это вторая производная координаты по времени. Переносим все слагаемые в уравнении (2.10) в одну сторону и выносим y

$$y'' + \frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right)y = 0 \tag{2.11}$$

Это обыкновенное дифференциальное уравнение 2-го порядка без затухающих колебаний. Это значит, что нить рассматривается в среде без учета вязкости, где на нее не действуют внешние силы. Нить и падающая на нее точка в данной задаче находятся в вязкой среде, поэтому добавляется дополнительный параметр вязкость данной среды η

$$y'' + \frac{\eta}{m}y' + \frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right)y = 0 \tag{2.12}$$

Так выглядит конечный вид уравнения, моделирующее затухающие обыкновенные гармонические колебания, где η – вязкость среды, m – масса точки, T – сила натяжения нити, x – точка падения и l – длина нити. Задача замыкается такими начальными условиями как, $y(0) = 0$, $y'(0) = 1$.

Характеристический многочлен рассматриваемого уравнения (2.12) имеет либо два комплексно-сопряженных корня, один повторяющийся вещественный корень, либо два различных вещественных корня в зависимости от того, является ли дискриминант отрицательным, нулевым или положительным. Эти три состояния соответствуют соотношениям между K и Ω, как было рассказано в выше в главе 1.1.3. В данной работе рассматривается соотношение $K < \Omega$. В таком случае система начинает совершать гармонические колебания, но амплитуда колебаний уменьшается экспоненциально, как показано на рисунке 1.3.

Так как $\Omega = \sqrt{\frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right)}$, а $K = \frac{\eta}{m}$, учитывая соотношение следует, что

$$\frac{\eta}{m} < \sqrt{\frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right)} \tag{2.13}$$

Из неравенства (2.13) получаем

$$T > \frac{\eta^2}{lm}x(l-x) \tag{2.14}$$

где T – это сила натяжения нити.

Так же учитывая данное соотношение составляется уравнение для нахождения собственной частоты и периода колебаний. Из уравнения (1.18) можно определить формулу для собственной частоты Ω_D

$$\Omega_D = (\Omega^2 - K^2)^{1/2} \tag{2.15}$$

где $\Omega = \sqrt{\frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right)}$, а $K = \frac{\eta}{m}$. Тогда соотношение (2.15) будет иметь вид

$$\Omega_D = (\frac{T}{m}\left(\frac{1}{x} + \frac{1}{l-x}\right) - (\frac{\eta}{m})^2)^{1/2} \tag{2.16}$$

Подставляем значение собственной частоты Ω_D в уравнение (1.23) и находим период колебаний τ

$$\tau = \frac{2\pi}{\Omega_D} \tag{2.17}$$

2.2 Решение прямой задачи

Теперь переходим к решению уравнения (2.12). Для этого нужно обозначить некоторые параметры. Например, длина нити l = 1, это константа. Теперь составим и решим характеристическое уравнение:

$$\begin{gathered} r^2 + \frac{\eta}{m}r + \frac{T}{mx(1-x)} = 0, \\ D = (\frac{\eta}{m})^2 - \frac{4T}{mx(1-x)}, \\ r_1 = -\frac{1}{2}\left(\frac{\eta}{m} + \sqrt{D}\right), \\ r_2 = \frac{1}{2}\left(\sqrt{D} - \frac{\eta}{m}\right) \end{gathered} \tag{2.18}$$

Другие параметры определим временно следующим образом:

$$m = 1;\ x = 0{,}5;\ T = 0{,}5;\ \eta = 1;\ t = 1,$$

и подставим их в формулу (2.18), чтобы получить определенные значения. Таким образом дискриминант равен $D = -7$, значит, уравнение имеет сопряжённые комплексные корни $r_1 = \mu - \varphi i,\ r_2 = \mu + \varphi i$. Параметры μ и φ можно использовать для нахождения углов отклонения при колебании системы. Теперь общее решение однородного уравнения принимает вид

$$y = e^{\mu t}(C_1 \cos(\varphi t) + C_2 \sin(\varphi t)) \tag{2.19}$$

где $C_1, C_2 = const$. Находим корни $r_1 = -\frac{1}{2} - \frac{\sqrt{7}}{2}i$, $r_2 = -\frac{1}{2} + \frac{\sqrt{7}}{2}i$ и получаются действительная и мнимая части: $\mu = -\frac{1}{2}$, $\varphi = \frac{\sqrt{7}}{2}$. Подставляем всё в общее решение (2.19)

$$y(t) = e^{-\frac{1}{2}t}\left(C_1 \cos\left(\frac{\sqrt{7}}{2}t\right) + C_2 \sin\left(\frac{\sqrt{7}}{2}t\right)\right) \quad (2.20)$$

Далее нужно найти решение задачи Коши, то есть, то, которое соответствует заданным начальным условиям. Нахождение решения задачи Коши требуется для того, чтобы можно было определить поведение рассматриваемой механической системы в любой момент времени. Позже найденная функция $y(t)$ будет использоваться для построения графика колебаний и наглядно покажет затухание этих самых колебаний.

Алгоритм нахождения частного решения следующий. Сначала используем начальное условие $y(0) = 0$ для уравнения (2.20)

$$y(0) = e^{-\frac{1}{2}*0}\left(C_1 \cos\left(\frac{\sqrt{7}}{2} * 0\right) + C_2 \sin\left(\frac{\sqrt{7}}{2} * 0\right)\right) \quad (2.21)$$

Находим первую константу из уравнения (2.21)

$$y(0) = C_1,$$

$$C_1 = 0 \quad (2.22)$$

Далее берем общее решение (2.19) и находим производную

$$y' = \mu e^{\mu t}(C_1 \cos(\varphi t) + C_2 \sin(\varphi t)) + e^{\mu t}(-C_1 \varphi \sin(\varphi t) + C_2 \varphi \cos(\varphi t)) \quad (2.23)$$

Используем второе начальное условие $y'(0) = 1$, подставляем его в уравнение (2.23) и получаем

$$y' = \mu C_1 + C_2 \varphi \quad (2.24)$$

Находим вторую константу из уравнения (2.24)

$$C_2 \varphi = 1,$$

$$C_2 = \frac{1}{\varphi} \quad (2.25)$$

Осталось подставить найденные константы из уравнений (2.22) и (2.25) в общее уравнение (2.19). Получаем

$$y(t) = e^{\mu t}\frac{1}{\varphi}\sin(\varphi t) \quad (2.26)$$

Уравнение (2.26) будет использоваться для построения графика колебаний нити [5].

2.3 Формирование базы данных

Для формирования базы данных требуется перенести проделанные в предыдущей главе вычисления на компьютерный язык. С помощью языка программирования Python повторяем все вычисления (приложение А) и запускаем цикл в нужное количество итераций. Тут уже некоторые параметры задаются случайным образом в определенном промежутке с помощью модуля random в библиотеке языка.

Как показано на рисунке 2.4, есть цикл, который создает две таблицы с входными (input_data) и выходными данными (output_data) на 500 тысяч строк. Масса m, точка падения x и время t выбираются случайным образом. Так как длина нити 1, чтобы точка не попадала на закрепленные концы нити, точка падения задается в промежутке от 0,001 до 0,999. Масса в промежутке от 0,001 до 1 и время до 50 секунд. Вязкость задается относительно того, какая среда была выбрана, в данном примере это оливковое масло ($\eta = 84 * 10^{-3}$ Па·с). Сила натяжения рассчитывается по формуле (2.14), исходя из точки падения и вязкости. Так как в формуле (2.14) показано, что значение силы натяжения должно быть выше, к посчитанному значению добавляем 1. Для нахождения остальных параметров используем функцию difEquation (приложение А), которая описывает решение прямой задачи. Функция difEquation использует для вычисления все указанные параметры и рассчитывает амплитуду, углы отклонения, период и изменения уравнения (2.24) относительно времени.

```
i = 0
while i<500000:
    i=i+1
    m = uniform(0.001, 1)
    x = uniform(0.001, 0.999)
    t = uniform(0.001, 50)
    n = 0.084 #n=tenacity
    F = n**2/m*x*(1-x)+1 #F=tencion force

    input_data['mass'].append(m); input_data['droping point'].append(x);
    output_data['time'].append(t); output_data['tencion force'].append(F);
    lst = list(difEquation(m, x, t, n, F))
    output_data['amplitude'].append(lst[0]); output_data['angle alpha'].append(lst[1]);
    output_data['angle beta'].append(lst[2]); output_data['period'].append(lst[3]);
    df_input_data = pd.DataFrame(input_data)
    df_output_data = pd.DataFrame(output_data)
```

Рисунок 2.4 – Код для генерирования базы данных

Чтобы проверить правильно ли написано и решено уравнение (2.12), а также код, строится график, который покажет, присутствует ли затухание. На

рисунке 2.5 показано, что когда нить находится в оливковом масле, присутствуют и колебания, и затухание, то есть затухающие колебания. Примерно на сотой секунде нить возвращается в свое первоначальное состояние. Для сравнения нить так же помещалась в различные среды помимо оливкового масла. Естественно, чем вязкость среды меньше, тем дольше и интенсивней происходят колебания. На рисунках 2.6 и 2.7 показано насколько интенсивней колебания, а на рисунках 2.8 и 2.9 — сколько секунд требуется для затухания. Например, когда нить находится в воде ($\eta = 1{,}718 * 10^{-3}$ Па·с) или в воздухе ($\eta = 0{,}0172 * 10^{-3}$ Па·с), требуется больше времени чтобы вернуться в первоначальное состояние покоя. Используется динамическая вязкость среды, так как это стандартное измерение вязкости, используемое в большинстве расчетов.

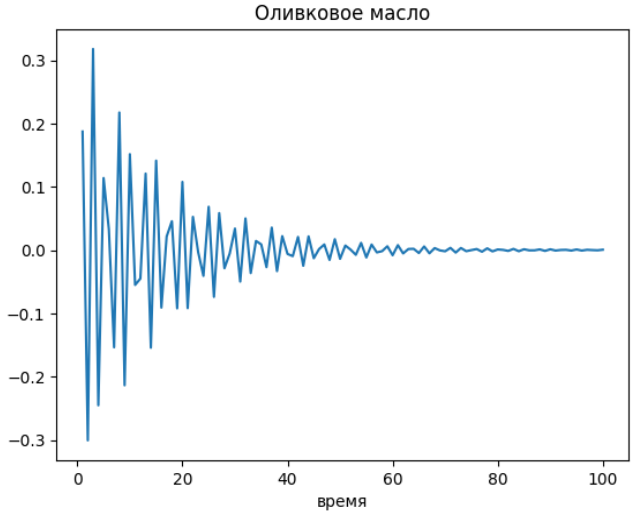

Рисунок 2.5 – Колебания нити в оливковом масле

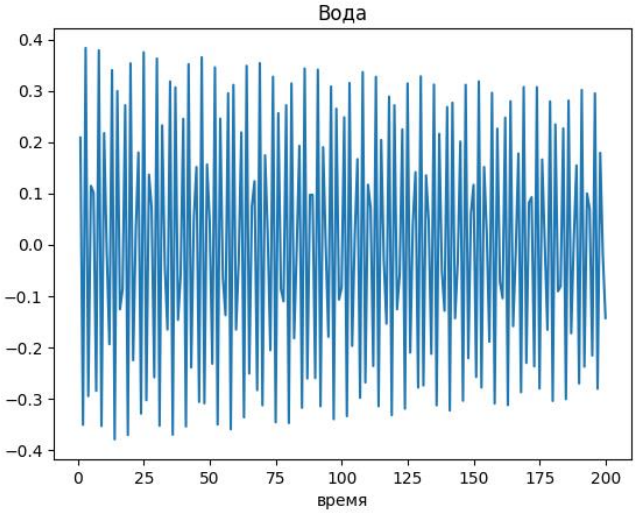

Рисунок 2.6 – Интенсивность колебаний нити в воде

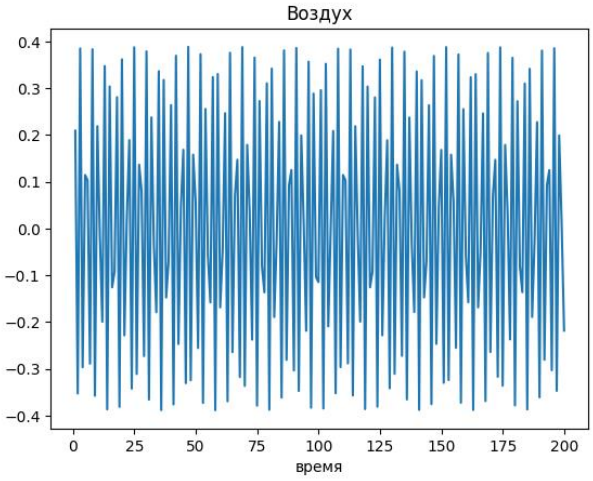

Рисунок 2.7 – Интенсивность колебаний нити в воздухе

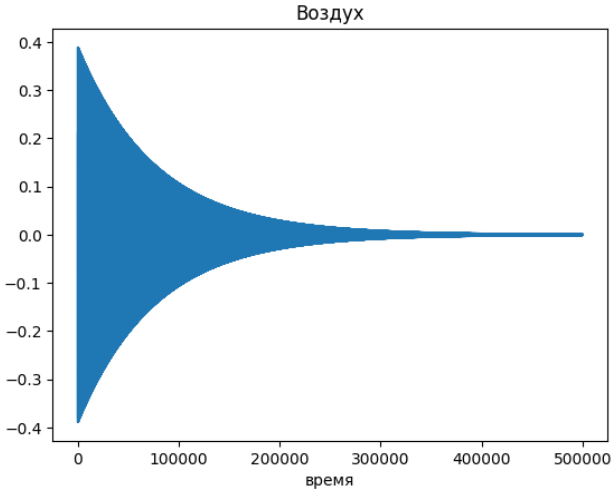

Рисунок 2.8 – Продолжительность колебаний нити в воздухе

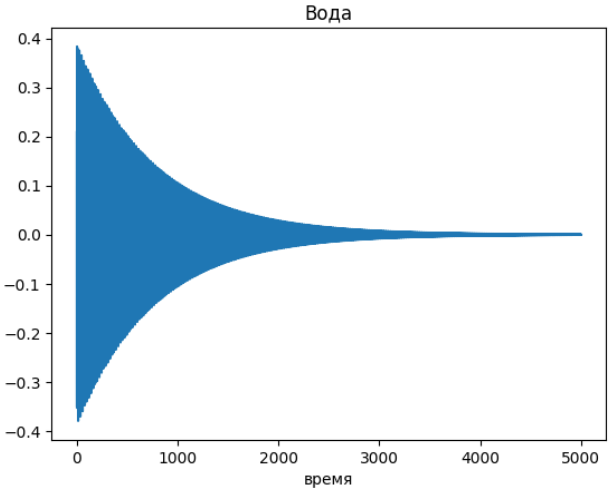

Рисунок 2.9 – Продолжительность колебаний нити в воде

Для сред с более высокой вязкостью, таких как бензин ($\eta = 128 * 10^{-3}$ Па·с) и серная кислота ($\eta = 27 * 10^{-3}$ Па·с), требуется меньше времени и интенсивность колебаний в данных средах гораздо ниже. Рисунки 2.10 и 2.11 наглядно это показывают.

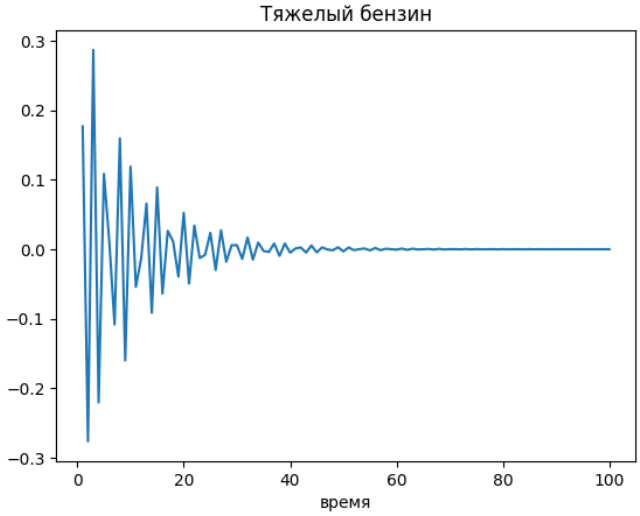

Рисунок 2.10 – Колебания нити в тяжёлом бензине

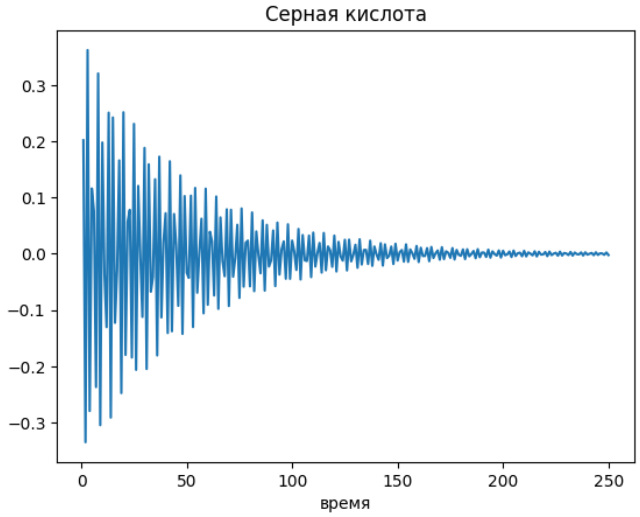

Рисунок 2.11 – Колебания нити в серной кислоте

На этом базу данных можно считать готовой. В таблице данных input_data у нас записаны масса и точка падения, а в таблице output_data время, период, собственная частота, сила натяжения и два угла α и β. На примере первых пяти значений показано, как выглядят таблица input_data на рисунке 2.12 и таблица output_data на рисунке 2.13.

```
df_input_data.head()
```

	масса	точка падения
0	0.719565	0.841361
1	0.603154	0.411004
2	0.532163	0.476154
3	0.309107	0.497061
4	0.446957	0.116000

Рисунок 2.12 – Код для вывода первых пяти значений таблицы input_data

```
df_output_data.head()
```

	время	период	собственная частота	угол альфа	угол бета	натяжение
0	0.343802	1.937633	3.242712	0.311949	1.041708	1.011212
1	23.512200	2.380155	2.639822	-0.114743	-0.080249	1.020335
2	2.308711	2.269235	2.768856	0.073626	0.066944	1.020952
3	31.903996	1.733410	3.624754	0.002497	0.002468	1.020999
4	9.822268	1.340462	4.687328	0.566214	0.083217	1.008614

Рисунок 2.13 – Код для вывода первых пяти значений таблицы output_data

2.4 Решение обратной задачи путем обучения и дальнейшего использования обученной нейронной сети

Для решения обратной задачи требуется выбрать оптимальную архитектуру нейронной сети, натренировать с помощью составленной базы данных и посмотреть на сколько точно нейронная сеть будет предсказывать значения. Нейронная сеть представляет собой математическую модель нейронов человеческого мозга. Для ее создания используется программная библиотека открытого вида TensorFlow, которая была создана компанией Google. Данная библиотека помогает создавать как сложные, так и простые модели нейронных сетей, а также проста в понимании и использовании.

Первый шаг – разделить базу данных на обучающие и тестовые выборки. Такое разделение данных необходимо для того, чтобы оценить способность модели к обобщению на данных, которых не было в обучающей выборке. На вход в нейронную сеть будут подаваться следующие параметры: время, период, собственная частота колебаний, сила натяжения и два угла отклонения на концах крепления нити, а на выходе: масса и точка падения. На рисунке 2.14 показан код для загрузки ранее созданной базы данных и разделение на обучающую и тестовую выборки.

```
file_input = pd.read_csv(r'D:\Desktop D\4 курс\new_in_500.000.txt', header=None, sep=' ')
file_output = pd.read_csv(r'D:\Desktop D\4 курс\new_out_500.000.txt', header=None, sep=' ')

m_x = pd.DataFrame.to_numpy(file_input)
degrees = pd.DataFrame.to_numpy(file_output)

output_train = m_x[:500000]
input_train = degrees[:500000]

output_test = m_x[500000:]
input_test = degrees[500000:]
```

Рисунок 2.14 – Код разделения базы данных на обучающую и тестовую выборки

Следующий шаг – определение архитектуры нейронной сети. Для этого будет использоваться библиотека Keras [6]. Для решения нашей задачи будет использоваться класс Model.

Далее идет импорт нужных библиотек, таких как numpy, pandas, TensorFlow и Keras, также модель Sequential и слои CNN из Keras. Sequential – это последовательная группировка линейного набора слоев в Keras, а CNN – это сверточные слои, которые помогут обучаться нейронной сети на базе данных. Все это показано на рисунке 2.15.

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Рисунок 2.15 – Код загрузки библиотек

Теперь можно начать построение архитектуры нейронной сети, как показано на рисунке 2.16.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])
```

Рисунок 2.16 – Код построения архитектуры нейронной сети

Архитектуру нейронной сети строят последовательно, начиная с входного слоя и заканчивая выходным. Для описания такой сети создаем класс Sequential. С помощью команды model.add добавляем первый слой. Units – это количество нейронов, которые определяют форму слоя. Первый слой – это входной слой и количество нейронов в нем зависит от того какое количество параметров задается. В данной работе на вход подается шесть параметров (время, период,

собственная частота, сила натяжения и два угла α и β), поэтому в первом слое будет шесть нейронов.

Далее идут скрытые слои. Нейронные сети с большим количеством слоев, а также с большим количеством нейронов могут выражать более сложные функции. Структура нейронной сети с такой архитектурой может работать лучше, чем сети меньшего размера, но еще могут с легкостью переобучиться – сеть не будет хорошо предсказывать. На практике бывает так, что сеть с двумя и более скрытыми слоями может превосходить простую сеть с одним слоем, но углубление не всегда помогает. За счет большого количества данных, в данной архитектуре нейронной сети используются два скрытых слоя, так как с бóльшим количеством скрытых слоев нейронная сеть недоучивалась. Что касается количества нейронов, то оптимальный размер нейронов в скрытом слое обычно находится между размером входных и размером выходных слоев. Получается, что количество нейронов для скрытых слоев, в данной архитектуре нейронной сети, должно варьироваться между шестью и двумя нейронами. При тестировании различных вариантов самая высокая точность была при количестве в шесть нейронов в первом скрытом слое и двух нейронов во втором скрытом слое.

И последний слой – это выходные данные, он ответственен за предсказания, которые будет делать нейронная сеть [7]. В выходном слое будет два нейрона, так как на выходе нейронная сеть должна предсказывать два параметра – массу и точку падения.

Что касается команды activation, то это библиотека функций активации, которая преобразует входные сигналы в выходные, необходимые для функционирования нейронной сети, и передает их на следующий слой. В зависимости от правила или порогового значения она включает или выключает нейроны в слое. Без конкретизации функции активации сеть будет изучать только линейные объекты. К сожалению, для выбора активационных функций нет каких-либо определенных правил, все зависит от задачи и используемых данных. Поэтому, изучив наиболее подходящие и наиболее распространенные функции как sigmoid, ReLU (Rectified Linear Unit), tanh и Softmax, было принято решение использовать функцию активации tanh. Гиперболический тангенс (tanh) используется при решении различных задач, таких как классификация, регрессия и даже распознавание образов [8], [9]. Данная функция лучше распознает сложные зависимости в данных, так как имеет более пологую кривую. Тоже самое можно сказать и про функцию sigmoid, но для данной задачи она не совсем подходит.

Последний этап создания нейронной сети, это - компиляция готовой архитектуры. При компиляции объявляется оптимизатор и функция потерь, их так же можно выбрать в зависимости от типа задачи и используемых данных.

В библиотеке Keras имеются различные варианты оптимизаторов для всевозможных задач. На практике самый распространенный из них это оптимизатор (optimizer) Adam. Метод прост в реализации, эффективен в вычислительном отношении, предъявляет небольшие требования к памяти,

инвариантен к диагональному масштабированию градиентов и хорошо подходит для задач, которые являются большими с точки зрения данных или параметров. Эмпирические результаты демонстрируют, что Adam хорошо работает на практике и выгодно отличается от других методов стохастической оптимизации [10]. Также были рассмотрены и другие функции оптимизации, но при тестировании наилучший результат показал именно данный оптимизатор Adam.

Что касается функции потерь (loss function), то ее цель состоит в вычислении величины, которую нейронная сеть должна стремиться минимизировать во время обучения. Функцию потерь так же можно выбрать среди большого количества в библиотеке Keras. Выбор пал на функцию среднеквадратичной ошибки (Mean Squared Error), которая в ходе тестирования оказалась наиболее подходящей. Данный метод используется для вычисления среднеквадратичных ошибок между предсказаниями и истинными значениями [11].

И последнее. Команда model.fit, как показано на рисунке 2.17, обучает нейронную сеть для фиксированного количества итераций (epochs). В команду подается обучающая выборка сначала входных, потом выходных данных и выбирается количество итераций. Одна итерация или эпоха (epochs) — это один полный проход через обучающий набор данных. Оптимальное количество выбирается так, чтобы нейронная сеть не попала в состояние недообучения или переобучения. Для этого оптимально установить максимальное количество эпох и прервать обучение на моменте возрастания частоты ошибки.

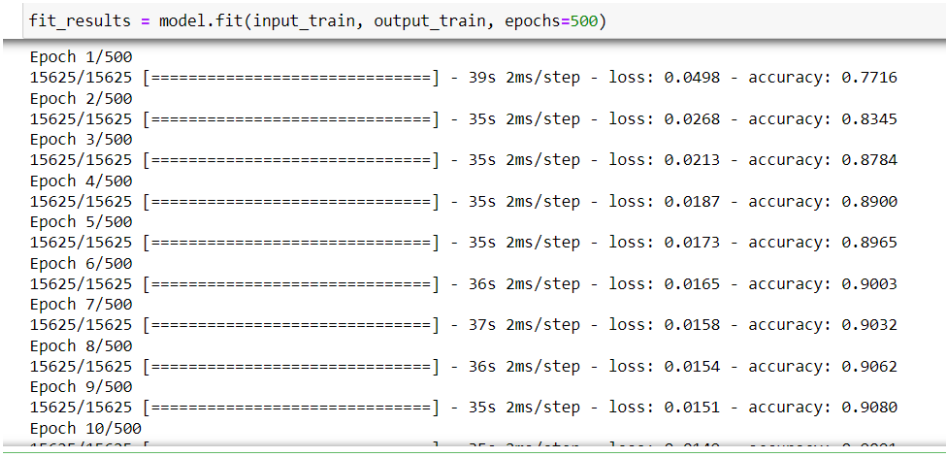

Рисунок 2.17 – Код запуска команды model.fit

Архитектура нейронной сети готова, теперь можно оценить, насколько точно сеть работает на тестовой выборке. Для этого используется команда model.evaluate, которая показывает насколько велико отклонение и на сколько точно нейронная сеть предсказывает ответы. Данная архитектура нейронной сети работает вполне успешно, точность составляет свыше 95%, что является хорошим показателем, смотри рисунок 2.18.

```
In [6]: testing = model.evaluate(input_test, output_test)
        testing

        1563/1563 [==============================] - 4s 2ms/step - loss: 0.0077 - accuracy: 0.9551

Out[6]: [0.007719121873378754, 0.9550600051879883]
```

Рисунок 2.18 – Код запуска команды model.evaluate

В качестве еще одной проверки, было принято решение создать дополнительную базу данных в 50 строк. Новая база данных отличается от обучающей и тренировочной выборки, и дает понять, насколько хорошо нейронная сеть будет работать на неизвестных ей данных.

Загружаем новую базу данных и воспользуемся командой model.predict, который предскажет значение массы и точку падения. Создаем таблицу из предсказанных данных. Как видно из рисунка 2.19 нейронная сеть работает.

```
In [7]: predictions = model.predict(test_out)
        pred = pd.DataFrame(predictions)

        2/2 [==============================] - 0s 0s/step
```

Рисунок 2.19 – Код запуска команды model.predict

Как показано на рисунке 2.20, при повторном запуске команды model.evaluate на дополнительно созданных данных, видно что точность составляет 98% и погрешность 0,0056.

```
model.evaluate(test_out, test_in)

2/2 [==============================] - 0s 11ms/step - loss: 0.0056 - accuracy: 0.9800

[0.005583403166383505, 0.9800000190734863]
```

Рисунок 2.20 – Код запуска команды model.evaluate на дополнительно созданных данных

Для более наглядной демонстрации работы нейронной сети, были созданы два графика которые показывают, насколько сильно предсказанные данные отличаются от реальных данных. Данная погрешность равна модулю от разности предсказанных и реальных параметров. На рисунках 2.21 и 2.22 видно, что погрешность присутствует, правда она не столь велика, а значит архитектуру нейронной сети можно считать вполне эффективной.

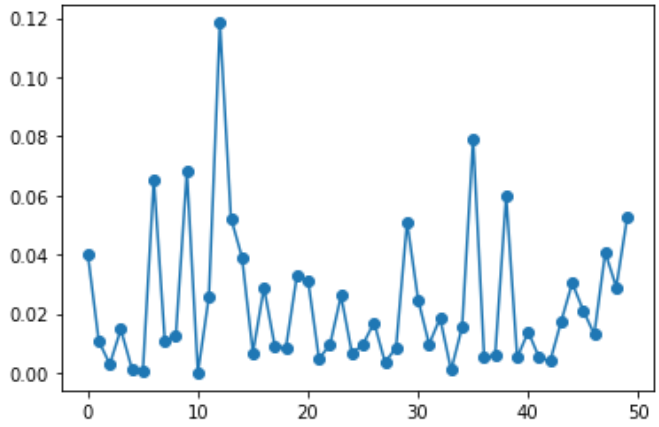

Рисунок 2.21 – График погрешности массы

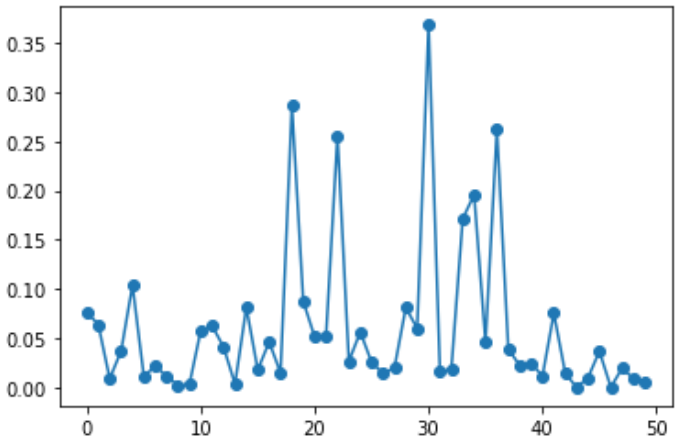

Рисунок 2.22 – График погрешности точки падения

2.5 Сложности возникшие в ходе выбора архитектуры для нейронной сети

Как упоминалось выше, при выборе подходящей архитектуры для нейронной сети возникли некоторые сложности. Дело в том, что нет определенных правил при выборе количества скрытых слоев, нейронов, функций активации, оптимизаторов, функции потерь и количества эпох. Поэтому для выбора подходящей архитектуры с наибольшей производительностью, основной упор делался на рекомендации с различных ресурсов и официальную документацию к библиотекам. Таким образом были выбраны несколько вариантов для тестирования.

Первым делом была выбрана простая нейронная сеть с входящим, одним скрытым и выходящим слоем, как показано на рисунке 2.23. Количество нейронов во входном слое шесть и выходном слое два, что определяется количеством параметров. В скрытом слое пока было установлено максимальное значение в шесть нейронов. Для первого теста использовались функция

активации sigmoid и оптимизатор SGD. Количество эпох было установлено как 1000. По завершению был получен результат в 71% точности. Это была первая пробная попытка.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='sigmoid'))
model.add(keras.layers.Dense(units=6, activation='sigmoid'))
model.add(keras.layers.Dense(units=2, activation='sigmoid'))
model.compile(optimizer='SGD', loss='mean_squared_error', metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=1000)
```

```
Epoch 1/1000
15625/15625 [==============================] - 36s 2ms/step - loss: 0.0801 - accuracy: 0.6148
Epoch 2/1000
15625/15625 [==============================] - 35s 2ms/step - loss: 0.0712 - accuracy: 0.6737
Epoch 3/1000
15625/15625 [==============================] - 34s 2ms/step - loss: 0.0628 - accuracy: 0.7020
Epoch 4/1000
15625/15625 [==============================] - 34s 2ms/step - loss: 0.0608 - accuracy: 0.7053
Epoch 5/1000
15625/15625 [==============================] - 34s 2ms/step - loss: 0.0605 - accuracy: 0.7067
```

Рисунок 2.23 – Код архитектуры нейронной сети с функцией активации sigmoid и оптимизатором SGD

Примерно после 100-й эпохи стало понятно, что точность не поднимется выше 71%, а функция потерь даже начала возрастать. Значит сеть начала переобучаться и для данной нейронной сети 1000 эпох очень много. Отталкиваясь от этого результата, сперва было решено попробовать перебрать все выбранные функции активации, как sigmoid, ReLU (Rectified Linear Unit), tanh и Softmax, с различными оптимизаторами, как SGD, Adam и Adamax. Это был очень долгий процесс. На рисунке 2.24 видно, что с первой эпохи сочетание функции активации tanh и оптимизатора Adam работает эффективнее. Правда 1000 эпох для данной архитектуры так же приводит к переобучению сети, поэтому данное число было сокращено до 500 эпох, что является самым оптимальным количеством для обучения нейронной сети.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=1000)
```

```
Epoch 1/1000
15625/15625 [==============================] - 25s 2ms/step - loss: 0.0480 - accuracy: 0.7785
Epoch 2/1000
15625/15625 [==============================] - 23s 1ms/step - loss: 0.0240 - accuracy: 0.8476
Epoch 3/1000
15625/15625 [==============================] - 23s 1ms/step - loss: 0.0187 - accuracy: 0.8822
Epoch 4/1000
15625/15625 [==============================] - 23s 1ms/step - loss: 0.0171 - accuracy: 0.8913
Epoch 5/1000
15625/15625 [==============================] - 23s 1ms/step - loss: 0.0164 - accuracy: 0.8945
```

Рисунок 2.24 – Код архитектуры нейронной сети с функцией активации tanh и оптимизатором Adam

Также были попытки с различными функциями активации и количеством нейронов в скрытом слое. Как видно на рисунке 2.25, использовались функции ReLU и Softmax, а также четыре нейрона в скрытом слое. Процент точности нейронной сети был довольно хорошим, однако функция потерь была выше, чем в архитектуре, показанной на рисунке 2.24.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='relu'))
model.add(keras.layers.Dense(units=4, activation='softmax'))
model.add(keras.layers.Dense(units=2, activation='softmax'))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=500)
```

```
Epoch 1/500
15625/15625 [==============================] - 24s 1ms/step - loss: 0.0612 - accuracy: 0.7837
Epoch 2/500
15625/15625 [==============================] - 23s 1ms/step - loss: 0.0538 - accuracy: 0.8464
Epoch 3/500
15625/15625 [==============================] - 25s 2ms/step - loss: 0.0527 - accuracy: 0.8604
Epoch 4/500
15625/15625 [==============================] - 26s 2ms/step - loss: 0.0523 - accuracy: 0.8666
Epoch 5/500
15625/15625 [==============================] - 27s 2ms/step - loss: 0.0522 - accuracy: 0.8676
```

Рисунок 2.25 – Код архитектуры нейронной сети с функциями активации ReLU и Softmax.

На этом этапе архитектура с наивысшей точностью содержала один скрытый слой, шесть нейронов в скрытом слое, функцию активации tanh, оптимизатор Adam и 500 эпох. Точность предсказаний данной архитектуры составляла 94%, что можно считать очень хорошим показателем, но возможно не максимальным. Поэтому далее были эксперименты с количеством скрытых слоев. На рисунке 2.26 показан один из вариантов с количеством в два скрытых слоя: количество нейронов в первом скрытом слое равно шести, а во втором скрытом слое – два. Такой вариант архитектуры оказался наиболее эффективным. Точность составила свыше 95%, что оказалось наивысшим показателем среди всех протестированных архитектур, а погрешность была 0,068. На новых данных нейронная сеть предсказывала с точностью 98% и погрешностью 0,0056. Поэтому было принято решение продолжать работу с данной архитектурой нейронной сети.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=500)
```

```
Epoch 1/500
15625/15625 [==============================] - 37s 2ms/step - loss: 0.0559 - accuracy: 0.7297
Epoch 2/500
15625/15625 [==============================] - 35s 2ms/step - loss: 0.0306 - accuracy: 0.8191
Epoch 3/500
15625/15625 [==============================] - 36s 2ms/step - loss: 0.0260 - accuracy: 0.8414
Epoch 4/500
15625/15625 [==============================] - 36s 2ms/step - loss: 0.0236 - accuracy: 0.8559
Epoch 5/500
15625/15625 [==============================] - 36s 2ms/step - loss: 0.0223 - accuracy: 0.8662
```

Рисунок 2.26 – Код архитектуры нейронной сети с наивысшим показателем точности

Так же были протестированы варианты с тремя скрытыми слоями. Но точность у данной архитектуры снова упала до 94%, а значение ошибки даже немного возросло до 0,0090. На рисунке 2.27 показан один из вариантов. Помимо этого, было протестировано различное количество нейронов в каждом слое.

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=5, activation='tanh'))
model.add(keras.layers.Dense(units=4, activation='tanh'))
model.add(keras.layers.Dense(units=3, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.compile(optimizer='Adam', loss='mean_squared_error', metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=500)
```

```
Epoch 1/500
15625/15625 [==============================] - 39s 2ms/step - loss: 0.0599 - accuracy: 0.7060
Epoch 2/500
15625/15625 [==============================] - 37s 2ms/step - loss: 0.0339 - accuracy: 0.8029
Epoch 3/500
15625/15625 [==============================] - 37s 2ms/step - loss: 0.0252 - accuracy: 0.8431
Epoch 4/500
15625/15625 [==============================] - 37s 2ms/step - loss: 0.0227 - accuracy: 0.8589
Epoch 5/500
15625/15625 [==============================] - 37s 2ms/step - loss: 0.0218 - accuracy: 0.8679
```

Рисунок 2.27 – Код архитектуры с тремя скрытыми слоями и разным количеством нейронов в каждом слое

Так же были созданы два графика которые показывают, насколько сильно предсказанные данные отличаются от реальных данных, в архитектуре с тремя скрытыми слоями. На рисунке 2.28 показана погрешность массы, а на рисунке 2.29 точки падения.

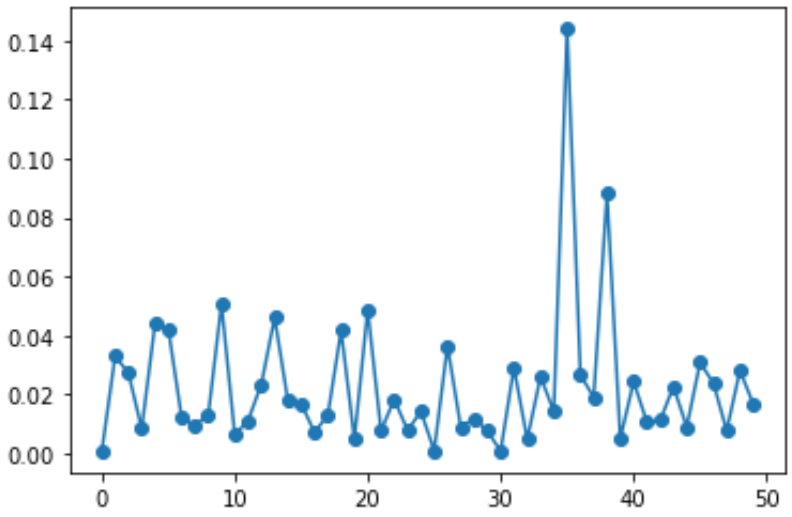

Рисунок 2.28 – погрешность массы при архитектуре с тремя скрытыми слоями

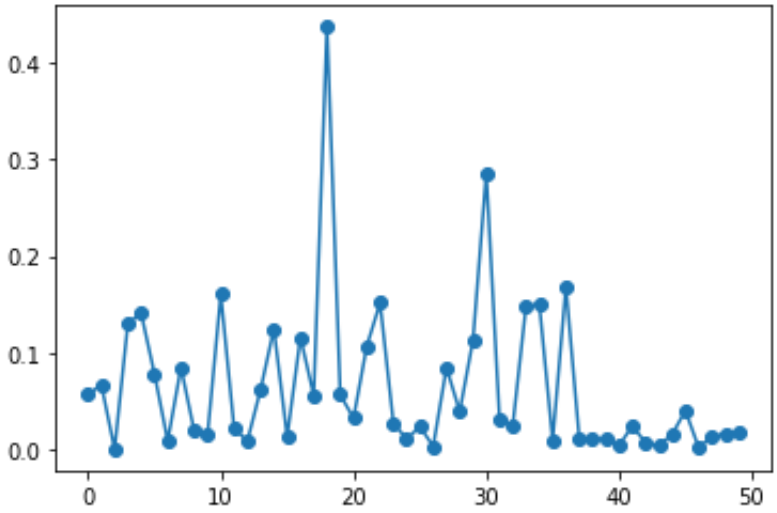

Рисунок 2.29 – погрешность точки падения при архитектуре с тремя скрытыми слоями

В общей сложности, при тестировании различных количеств скрытых слоев в нейронной сети, эксперименты проводились над семью различными архитектурами. Во всех проведенных экспериментах, точность не превышала 95%, а погрешность составляла не менее 0,0085.

ЗАКЛЮЧЕНИЕ

Целью данной дипломной работы было решение обратной задачи колебаний закрепленной нити с точечной массой в вязкой среде. Решение обратной задачи состоит в том, что с помощью нейронных сетей на основе таких параметров как: период колебания, углы отклонения на концах крепления нити, сила натяжения нити, время и собственная частота определять точечную массу и ее точку падения на нить. Вязкость среды, в которой находится нить и падающая на нее точка, также является важным параметром. В ходе написания дипломной работы, были выполнены серии экспериментов с использованием различных сред. Задача была составлена так, что при необходимости можно менять вязкость среды. В данной работе в качестве вязкой среды использовалось оливковое масло.

Первым этапом выполнения дипломной работы было построение математической модели прямой задачи. Были определены все силы, действующие на точку, и используя законы Ньютона, была получена математическая модель. Дальнейшим шагом было решение полученной задачи, при условии, что в начальный момент времени нить покоилась, а скорость падающей точки равнялась единице. Данное решение использовалось далее на следующем этапе для составления базы данных, на которой пройдет обучение нейронной сети.

Следующим шагом было формирование базы данных. Для этого был написан код на языке Python. Код состоит из функции difEquation и цикла. Входными параметрами для функции difEquation являются точечная масса, точка падения, время, вязкость среды и сила натяжения нити. С помощью этих параметров данная функция находит решение прямой задачи и на выходе выдает значения таких параметров, как собственная частота, углы отклонения нити на концах и период колебаний, а цикл позволяет сформировать базу данных на основе полученных значений. Точечная масса, точка падения и время подбираются случайным образом в определенном промежутке, чтобы в базе данных были рассмотрены все возможные варианты событий. Объединяя всё это в одну таблицу была получена база данных в 500 тысяч строк. Далее для нейронной сети были определены входные и выходные данные. Во входные данные вошли: время, период, собственная частота, сила натяжения нити и два угла отклонения на концах крепления нити, а в выходные – масса и точка падения.

Последним этапом было нахождение оптимальной архитектуры нейронной сети. Для работы с нейронной сетью использовалась библиотека TensorFlow. Архитектура нейронной сети в данной работе состоит из входного, двух скрытых и выходного слоя. Нейронная сеть с большим количеством скрытых слоев и нейронов может работать эффективнее. Но в данном случае при добавлении трех и более скрытых слоев, из-за недостаточного количества данных нейронная сеть показывала более низкую точность. Количество нейронов во входном слое шесть, что обусловлено количеством выбранных входных параметров, а в выходном два,

так как нейронная сеть предсказывает значения для двух параметров. На данном этапе возникли некоторые трудности с подбором оптимальной архитектуры для нейронной сети. Отталкиваясь от рекомендаций с различных ресурсов и официальную документацию к библиотекам, были выбраны различные варианты архитектур нейронных сетей. Далее каждая архитектура тестировалась для выявления наиболее эффективной, а именно выбиралась та, у которой была высокая точность и низкий показатель ошибки.

В результате была получена наиболее эффективная архитектура нейронной сети. Выбранная архитектура состоит из входного слоя, двух скрытых слоев в шесть и два нейрона, выходного слоя, функции активации tanh, оптимизатора Adam, функции потерь MSE и количеством эпох 500. Точность предсказаний данной архитектуры нейронной сети составила более 95%, а погрешность 0,0068. Тем самым было найдено решение поставленной обратной задачи.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Gregory R.D. Classical Mechanic: Solutions Manual, Cambridge University Press, 2006. – 105 с.

2 Официальный сайт Python // Электронная версия на сайте: https://wiki.python.org/moin/BeginnersGuide/Overview.

3 Официальный сайт TensorFlow // Электронная версия на сайте: https://www.tensorflow.org/learn.

4 Статья “Introduction to Keras – Deep Learning Library” // Электронная версия на сайте: https://medium.com/mlait/introduction-to-keras-deep-learning-library-2844b39f0496.

5 А. П. Рябушко. Сборник индивидуальных заданий по высшей математике. ч. 2: Минск «Вышэйшая школа», 1991. – 243 с.

6 Официальный сайт документации к библиотеке Keras // Электронная версия на сайте: https://keras.io/about/.

7 Статья “How deep should neural nets be?” // Электронная версия на сайте: https://kharshit.github.io/blog/2018/04/27/how-deep-should-neual-nets-be.

8 Официальный сайт документации функций активации в библиотеке Keras // Электронная версия на сайте: https://keras.io/api/layers/activations/.

9 Статья “Выбор слоя активации в нейронных сетях: как правильно выбрать для вашей задачи” // Электронная версия на сайте: https://habr.com/ru/articles/727506/.

10 Официальный сайт документации оптимизаторов в библиотеке Keras // Электронная версия на сайте: https://keras.io/api/optimizers/.

11 Официальный сайт документации функций потерь в библиотеке Keras // Электронная версия на сайте: https://keras.io/api/losses/.

Приложение А

Текст программы

Импорт библиотек:

```
import random
from math import sqrt
from math import e
from math import sin
from math import atan
from random import uniform
from random import randint
import pandas as pd
from numpy import pi
```

Функция для решения обычного дифференциального уравнения 2-го порядка:

```
def difEquation(mass, drop_point, time, tenacity, tencion):
    D = (tenacity/mass)**2-(4*tencion)/(mass*drop_point*(1-drop_point))
    r_a1_2 = 0.5*(-1*(tenacity/mass))
    if (D<0):
      D=-1*D
    r_b1 = 0.5*(-1*(sqrt(D)))
    r_b2 = 0.5*(sqrt(D))
    if (r_b1*(-1)==r_b2):
       a=r_a1_2
       b=r_b2
    else:
       print("ERROR")
    C2 = 1/b

    yt = (e**(a*time))*C2*sin(b*time)
    tna = atan(yt/drop_point)
    tnb = atan(yt/(1-drop_point))
    w=sqrt(abs((tencion/(mass*drop_point*(1-drop_point)))-
((tenacity/mass)**2)))
    if(w==0):
      print(w, mass, drop_point)
    T=(2*pi)/w #T=period
    return w, tna, tnb, T, yt

input_data = {'mass': [], 'droping point': []}
output_data = {'time': [], 'period': [], 'amplitude': [], 'angle alpha': [],
    'angle beta': [], 'tencion force': []}
```

Продолжение А

Цикл для генерации базы данных:

```
i = 0
while i<500000:
   i=i+1
   m = uniform(0.001, 1)
   x = uniform(0.001, 0.999)
   t = uniform(0.001, 50)
   n = 0.084 #n=tenacity
   F = n**2/m*x*(1-x)+1 #F=tencion force

   input_data['mass'].append(m); input_data['droping point'].append(x);
   output_data['time'].append(t); output_data['tencion force'].append(F);
   lst = list(difEquation(m, x, t, n, F))
   output_data['amplitude'].append(lst[0]);
   output_data['angle alpha'].append(lst[1]);
   output_data['angle beta'].append(lst[2]); output_data['period'].append(lst[3]);
   df_input_data = pd.DataFrame(input_data)
   df_output_data = pd.DataFrame(output_data)
```

Сохранение данных на компьютер:

```
df_input_data.to_csv(r'input.txt', header=None, index=None, sep=' ', mode='a')
df_output_data.to_csv(r'output.txt', header=None, index=None, sep=' ',
mode='a')
```

Выгрузка данных из компьютера:

```
file_input = pd.read_csv('input_500.000.txt', header=None, sep=' ')
file_output = pd.read_csv('output_500.000.txt', header=None, sep=' ')
```

Импорт библиотек для нейронной сети:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Продолжение А

Разделение данных на обучающую и тестовую выборки, а также создание и обучение нейронной сети:

```
m_x = pd.DataFrame.to_numpy(file_input)
degrees = pd.DataFrame.to_numpy(file_output)

output_train = m_x[:500000]
input_train = degrees[:500000]

output_test = m_x[500000:]
input_test = degrees[500000:]

model = tf.keras.Sequential()
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=6, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.add(keras.layers.Dense(units=2, activation='tanh'))
model.compile(optimizer='Adam', loss='mean_squared_error',
metrics=['accuracy'])

fit_results = model.fit(input_train, output_train, epochs=500)
```

Тестирование модели на тестовых данных:

```
model.evaluate(input_test, output_test)
```

Загрузка новых данных из компьютера:

```
test_in = pd.read_csv('test_in.txt', header=None, sep=' ')
test_out = pd.read_csv('test_out.txt', header=None, sep=' ')
```

Проверка работы сети на новых данных:

```
model.evaluate(test_out, test_in)
```

Получение предсказанных параметров на новых данных:

```
predictions = model.predict(test_out)
pred = pd.DataFrame(predictions)
```

Высчитывание разницы между предсказанными и настоящими данными:

```
sub = test_in.subtract(pred)
```

Создание графиков погрешности для массы и точки падения:

```
import matplotlib.pyplot as plt
plt.plot(abs(sub[0]), '-o')
plt.plot(abs(sub[1]), '-o')
```

Продолжение А

Создание графика колебаний нити в оливковом масле:

```
import matplotlib.pyplot as plt  #Oliv oil
time = []
a = []
t=0
while t<100:
  t=t+1
  F = 0.084*0.352375*(1-0.352375)+1
  y = list(difEquation(0.661841, 0.352375, t, 0.084, F))
  time.append(t)
  a.append(y[4])

plt.title('Оливковое масло')
plt.xlabel('время')
plt.plot(time, a)
plt.show()
```