**ВВЕДЕНИЕ**

Преддипломная практика является неотъемлемой частью учебного процесса, в ходе которой закрепляются теоретические знания на производстве, направленной на получение практических знаний и навыков профессиональной деятельности предприятия. Производственная практика специализирована для расширения представлений о специальности, полученных при теоретическом обучении, а также для приобретения производственного опыта и конкретных производственных навыков по специальности.

В математической физике имеется достаточно широкий круг задач, которые приводят к изучению краевых или начально-краевых задач для уравнений в частных производных (или интегро-дифференциальных уравнений). Существующие приближённые методы решения либо позволяют получить лишь поточечную аппроксимацию подобно сеточным методам, либо предъявляют специальные требования к набору аппроксимирующих функций и требуют решения важной вспомогательной задачи разбиения исходной области подобно тому, как это происходит в методе конечных элементов. Применение МКЭ (метода конечных элементов) для решения задач теории упругости, как правило, приводят к СЛАУ (системам линейных алгебраических уравнений) больших размерностей, время решения которых на современных вычислительных системах может измеряться часами. Поэтому, в настоящее время разрабатываются новые методы решения дифференциальных уравнений, позволяющие снизить временные затраты для нахождения решения. Целью данной работы является применение искусственной нейронной сети для определения прогибов тонкой прямоугольной пластины и разработка соответствующего программного обеспечения.

# ОБЩИЕ СВЕДЕНИЯ И СТРУКТУРА ПРЕДПРИЯТИЯ «ИВА-Гомель-Парк»

* 1. **История предприятия**

*IBA Group* – это крупная коалиция компаний, которые специализируются в области разработки, производства и поставки решений и услуг в области информационных технологий на территории сран Восточной и Центральной Европы. Штаб-квартира *IBA Group* расположена в г. Праге (Чехия).

Под данным брендом собрано уже более 20 компаний, которые располагаются на территории Республики Беларусь, Чехии, США, Германии, Казахстана, Украины, Словакии, ЮАР, Великобритании, России и на Кипре. После вступления Чехии в Европейский союз в 2005 году местом размещения главного офиса *IBA Group* стала Прага.

За время своего существования *IBA Group* сумели выполнить более 2000 проектов для заказчиков из множества различных секторов экономики, включая такие как транспорт, энергетику, телекоммуникации и информационные технологии.

В 2001 году на территории свободной экономической зоны «Гомель-Ратон» был открыт *IBA Gomel* (Иностранное унитарное предприятие «ИВА-Гомель») – гомельский центр разработок международного холдинга *IBA Group*.

Данное предприятие было создано ради привлечения перспективных IT-специалистов из гомельского региона к разработке современных программных продуктов для различных стран мирового сообщества.

Со дня основания IBA Group постоянно работает над созданием, внедрением и совершенствованием систем менеджмента с целью обеспечения стабильного качества выполняемых работ и оказываемых услуг, снижения проектных рисков и обеспечения информационной безопасности. Благодаря этому начиная с 2004 года в компании «ИВА-Гомель» внедрена система менеджмента качества проектирования, разработки и сопровождения ПО. Данная система соответствует международному стандарту менеджмента качества *ISO* 9001:2008, распространяющемуся на проектирование, разработку и сопровождение программного обеспечения, что подтверждается Немецким обществом по аккредитации (*DAkkS TGA*), а так же сертификатом Национального организации по сертификации. Так же данная система соответствует требованиям СТБ *ISO/IEC* 27001 для систем менеджмента информационной безопасности применительно к проектированию, разработке, производству и сопровождению программного обеспечения и автоматизированных информационных систем.

В 2005 году компания «ИВА-Гомель» присоединилась к научно-технологической ассоциации «Инфопарк». С ноября 2008 года в состав гомельского центра разработок было включено дополнительное предприятие ООО «ИВА-Гомель-Парк» (*IBA GomelPark),* призванное предоставлять услуги по проектированию, разработке и поставке ПО для корпоративных заказчиков из Беларуси, стран СНГ, Западной Европы и США. В мае 2009 года «ИВА-Гомель-Парк» стала резидентом Парка высоких технологий (ПВТ).

В 2012 году компания «ИВА-Гомель-Парк» получила награду Национального конкурса «Золотой Байт» в номинации «Лучший региональный центр разработки ПВТ».

«ИВА-Гомель-Парк» объединяет в себе более 350 высоко квалифицированных специалистов, обладающих обширным опытом разработки программных решений на самых современных технологий, платформ и программных продуктов для различных отраслей экономики и производства.

Среди основных направлений деятельности компании можно выделить следующие:

* проектирование, разработка и сопровождение приложений для платформы *Java Enterprise Edition, Microsoft SharePoint/.Net*;
* приложений для извлечения, преобразования, загрузки массивов данных с использованием *ETL*-технологий;
* приложений в среде «облачных» вычислений;
* приложений для работы с *BigData* и *SemanticWeb*;
* приложений для мобильных устройств на различных платформах (*Android, iOS, WindowsPhone/Mobile*);
* приложений с использованием концепции *SPA* и сервис-ориентированной архитектуры (*SOA*);
* приложений на языке *ABAP* для систем *SAP*;
* приложений на базе системы коллективной работы *IBMNotes/Domino*;
* продуктов Интернет и *e*-*business* (информационных порталов, *Web*-сайтов, систем ведения электронных платежей и др.);
* решений для обмена данными между корпоративными системами, включая унаследованные, *SAP* и *Web*-приложения (на базе IBM *WebsphereMessageBroker*) ;
* создание и развитие интегрированных систем управления предприятием на базе решений *SAP*;
* миграция программных продуктов между разнообразными платформами и средами;
* проектирование и администрирование баз данных (*DB2, Oracle, MS SQL Server*);
* разработка и внедрение *BI*-решений по отчетности и анализу данных;
* разработка облачных сервисов по управлению НСИ (*MDM, MDG)*;
* анализ качества данных в справочниках (*ISO* 8000/22745);
* построение систем управления НСИ (*SAP MDG, MDM/BPM/Webdynpro*);
* построение онтологий и средств структуризации данных (*RDF/OWL*);
* преобразование данных НСИ к формату *SemanticWeb* и организация доступа (*SPARQL*);
* банковские технологии.

*IBA Group* имеет партнерские статусы ведущих мировых ИТ-корпораций, таких как *IBM, SAP, Hewlett Packard Enterprise (HPE), OpenText, Oracle, Check Point, Glory Global Solutions, Philips, Diebold Nixdorf, PTC, Splunk Inc, Lenovo* и др.

**1.2 Используемое оборудование**

Рабочие кабинеты *IBA Gomel* оснащены современным оборудованием, а также всеми сопутствующими необходимыми аксессуарами. Типичное рабочее место инженера-программиста состоит из следующих частей:

* персональный компьютер на базе процессора *Intel Core* (*i5* или *i7*);
* монитор с диагональю 19-21 дюйм;
* клавиатура, мышь, наушники.

Кроме этого, офис оснащён проводной и беспроводной (*Wi-Fi*) сетью, объединяющей все компьютеры организации. Данные сети, также, предоставляют доступ к сети Интернет. В целом, можно отметить, что материальная база *IBA Gomel* находится на высоком уровне.

**1.3 Охрана труда на предприятии**

Охрана труда – система обеспечения безопасности жизни и здоровья, работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, санитарно-гигиенические, психофизиологические, лечебно-профилактические, реабилитационные, и иные мероприятия и средства.

Руководством *IBA Gomel* уделяется большое внимание улучшению эргономики рабочих мест, обеспечению гигиены и совершенствованию организации труда, регламентации режимов труда и отдыха.

Вся деятельность в области охраны труда в *IBA Gomel* регламентирована действующим законодательством Республики Беларусь, санитарными нормами и правилами, гигиеническими нормативами, предписаниями надзорных органов.

К комплексу мероприятий в области охраны труда, осуществляемых в *IBA Gomel* относятся:

* проведение периодических медицинских осмотров персонала;
* проведение производственного лабораторного контроля за условиями труда на рабочих местах;
* разработка инструкций по охране труда и ознакомление с ними персонала;
* обеспечение средствами индивидуальной защиты работников;
* изменение технологических процессов с целью совершенствования их безопасности для здоровья работников;
* модернизация рабочих мест и технологического оборудования;
* создание безопасных условий труда.

Важным фактором создания безопасных условий труда является оптимизация организации рабочих мест. При правильной организации рабочего места производительность труда инженера возрастает с 8 до 20 процентов.

Основным рабочим местом инженера-программиста является стол для выполнения машинописных работ. Основная поза при выполнении работы – вынужденная, сидячая.

Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования».

Требования к организации работы при использовании персонального компьютера и организационных средств определяются СанПиН 9-131 РБ 2000 «Гигиенические требования к видеодисплейным терминалам, электронно-вычислительным машинам и организации работы».

Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

* оптимальное размещение оборудования, входящего в состав рабочего места;
* достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
* уровень акустического шума не должен превышать допустимого значения.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Помещения для работы программиста должны иметь естественное и искусст­венное освещение.

Площадь на одно рабочее место с видео-дисплейным терминалом (ВДТ) и ПЭВМ для взрослых пользователей должна составлять не менее 6,0 м2, а объем не менее 20,0 м3.

Искусственное освещение в помещениях эксплуатации ВДТ и ПЭВМ должно осуществляться системой общего равномерного освещения. В административно-общественных помещениях, в случаях преимущественной работы с документами, допускается применение системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Местное освещение не должно создавать бликов на по­верхности экрана и увеличивать освещенность экрана более 300 лк.

В качестве источников света при искусственном освещении необходимо использовать преимущественно люминесцентные лампы.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей (размер ВДТ и ПЭВМ, клавиатуры и др.), характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируе­мым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии не менее чем 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Помимо требований к организации рабочего места СанПиН 9-131 РБ 2000 устанавливает требования к микроклимату рабочей зоны: влажности, температуре, скорости потока воздуха и пр.

Техника безопасности – это система организационных мероприятий и технических средств, предотвращающих воздействие на работающих опасных производственных факторов.

Для соблюдения техники безопасности в своей деятельности инженер-программист должен руководствоваться инструкцией по охране труда для программиста при выполнении работ с применением ПЭВМ и ВДТ.

**1.4 Должностная инструкция инженера-программиста**

Общие положения:

* инженер-программист относится к категории специалистов;
* инженер-программист назначается на должность и освобождается от нее приказом генерального директора по представлению технического директора / начальника структурного подразделения;
* инженер-программист подчиняется непосредственно техническому директору / начальнику структурного подразделения;
* на время отсутствия инженера-программиста его права и обязанности переходят к другому должностному лицу, о чем объявляется в приказе по организации;
* на должность инженера-программиста назначается лицо, отвечающее следующим требованиям: высшее профессиональное (техническое) образование, стаж работы от года.

Инженер-программист должен знать:

* руководящие и нормативные материалы, регламентирующие методы разработки алгоритмов и программ и использования вычислительной техники при обработке информации;
* основные принципы структурного программирования;
* виды программного обеспечения;
* технологию автоматической обработки информации и кодирования информации;
* формализованные языки программирования;
* порядок оформления технической документации.

Инженер-программист руководствуется в своей деятельности:

* законодательными актами РБ;
* уставом организации, Правилами внутреннего трудового распорядка, другими нормативными актами компании;
* приказами и распоряжениями руководства;
* настоящей должностной инструкцией.

Инженер-программист выполняет следующие должностные обязанности:

* на основе анализа математических моделей и алгоритмов решения экономических и других задач разрабатывает программы, обеспечивающие возможность выполнения алгоритма и соответственно поставленной задачи средствами вычислительной техники, проводит их тестирование и отладку;
* разрабатывает технологию решения задачи по всем этапам обработки информации;
* осуществляет выбор языка программирования для описания алгоритмов и структур данных;
* определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля;
* выполняет работу по подготовке программ к отладке и проводит отладку;
* осуществляет запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач;
* проводит корректировку разработанной программы на основе анализа выходных данных;
* разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию;
* определяет возможность использования готовых программных продуктов;
* осуществляет сопровождение внедрения программ и программных средств;
* разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации;
* выполняет работу по унификации и типизации вычислительных процессов.

Права инженера-программиста:

* знакомиться с проектами решений руководства предприятия, касающихся его деятельности;
* вносить на рассмотрение руководства предложения по совершенствованию работы, связанной с предусмотренными настоящей инструкцией обязанностями;
* сообщать своему непосредственному руководителю о всех выявленных в процессе осуществления должностных обязанностей недостатках в деятельности предприятия (его структурных подразделениях) и вносить предложения по их устранению;

Ответственность инженера-программиста:

* за невыполнение и/или несвоевременное, халатное выполнение своих должностных обязанностей;
* за несоблюдение действующих инструкций, приказов и распоряжений по сохранению коммерческой тайны и конфиденциальной информации;

На время отсутствия инженера-программиста (отпуск, болезнь, командировка, пр.) его обязанности исполняет лицо, назначенное в установленном порядке, которое несет ответственность за качественное исполнение возложенных на него обязанностей.

1. **ЭТАПЫ РАЗРАБОТКИ ПРИЛОЖЕНИЯ ДЛЯ НЕЙРОСЕТЕВОГО МОДЕЛИРОВАНИЯ ПРОГИБОВ ТОНКИХ ПЛАСТИН ИЗ ОДНОРОДНОГО МАТЕРИАЛА**
   1. **Объективная потребность в приложении для определения прогибов тонких пластин с использованием нейронных сетей**

Моделирование реальных физических или технических механических систем включает широкое использование программных пакетов, базирующихся на методе конечных элементов (МКЭ) для решения краевых задач [1]. Последние достижения в области алгоритмов машинного обучения и их успешное применение в различных областях демонстрируют, что при правильном выборе и обучении эти модели могут значительно улучшить традиционные методы.

Использование метода конечных элементов для решения задач теории упругости, как правило, приводит к возникновению СЛАУ больших размеров, время решения которых на современных компьютерных системах можно измерить в часах [2]. При проведении исследований реальных физических объектов, при решении задач подбора оптимальных значений параметров подобные СЛАУ приходится решать многократно [3]. Поэтому требуется найти алгоритмы и подходы, позволяющие снизить временные затраты для нахождения решения. Нейронные сети, при использовании достаточно большой обучающей выборке, способны находить решения задачи намного быстрее решателей на основе численных методов нахождения прогибов. Недостатком подобного метода нахождения решения задачи является более низкая точность полученных результатов по сравнению с использованием численных методов.

* 1. **Метод конечных элементов для определения прогибов пластин**

Описание законов физики для задач, зависящих от пространства и времени, обычно выражается в терминах дифференциальных уравнений с частными производными (ДУЧП). Для подавляющего большинства геометрий и проблем эти уравнения не могут быть решены аналитическими методами. Вместо этого можно построить аппроксимацию уравнений, обычно основанную на различных типах дискретизаций. Эти методы дискретизации аппроксимируют данные уравнения с помощью численных модельных уравнений, которые могут быть решены с использованием численных методов. Решение уравнений численной модели, в свою очередь, является приближением реального решения уравнений в частных производных. Метод конечных элементов (МКЭ) используется для вычисления таких приближений [4]. Например, функция  может быть зависимой переменной в ДУЧП (температура, электрический потенциал, давление и т. д.). Функция  может быть аппроксимирована функцией  с использованием линейных комбинаций базисных функций в соответствии со следующими выражениями:

, (2.1)

и

. (2.2)

Здесь  обозначает базисные функции, а  обозначает коэффициенты функций, которые аппроксимируют  с помощью . Рисунок 2.1 иллюстрирует этот принцип для одномерной задачи. Можно представить температуру по длине () стержня, который нагревается неравномерно. Здесь линейные базисные функции имеют значение 1 в своих соответствующих узлах и 0 в других узлах. В этом случае вдоль части оси имеется семь элементов, где определена функция  (длина стержня).

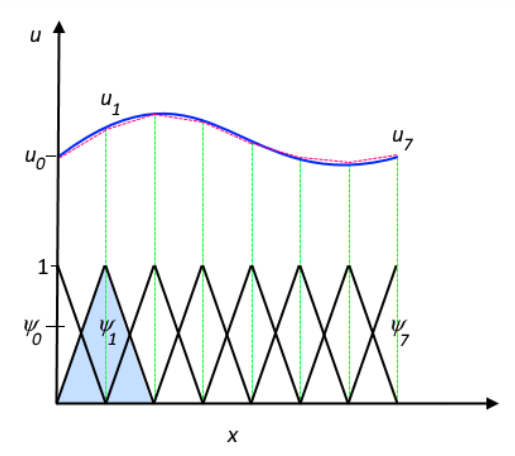


Рисунок 2.1 – Аппроксимация функции  равномерно распределенными по оси  элементами

На рисунке 2.1 показано, что функция  (показана сплошной синей линией) аппроксимируется с помощью  (показано пунктирной красной линией), которая представляет собой линейную комбинацию линейных базисных функций ( представлена сплошными черными линиями). Коэффициенты обозначаются от  до .

Одним из преимуществ использования метода конечных элементов является большая свобода в выборе дискретизации как в элементах, которые могут использоваться для дискретизации пространства, так и в базисных функциях. На рисунке 2.1 элементы равномерно распределены по оси x, хотя это и не обязательно. Можно применить элементы меньшего размера в местах большого значения градиента . На рисунке 2.2 представлена аппроксимация функции  с уменьшением размера элементов.

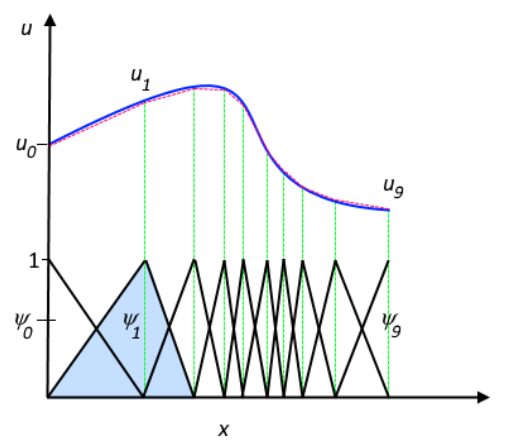


Рисунок 2.2 – Улучшенная аппроксимация функции 

Законы физики часто выражаются на языке математики. Например, законы сохранения, такие как закон сохранения энергии, сохранения массы и сохранения импульса, могут быть выражены в виде дифференциальных уравнений с частными производными (ДУЧП). Дифференциальные уравнения включают в себя выражения, которые определяют небольшое изменение зависимой переменной относительно изменения независимой переменной ( ).

Одним из свойств, которые влияют на изгиб пластины, является соотношение между длиной стороны и толщиной материала [5]. На основании этих соотношений можно выделить следующие классификации пластин:

– толстая пластина;

– тонкая пластина;

– мембрана.

Пластины характеризуются соотношением между длины стороны пластины  и толщиной элемента . Пластина считается толстой, если отношение длины стороны к толщине меньше 8. Если соотношение составляет от 8 до 80, то пластина считается тонкой, а если это соотношение превышает 80, то такая пластина считается мембраной. Все эти различные типы пластин характеризуются основными допущениями соответствующих теорий пластин [6].

Согласно гипотезе Киркгоффа, в любой точке  перемещение  в направлении оси  из-за небольшого вращения характеризуется следующим образом:

. (2.3)

В этой же точке  перемещение  в направлении оси  характеризуется следующим образом:

. (2.4)

Степени кривизны пластины , ,  задаются как скорость изменения угловых смещений нормали и определяются как:

, (2.5)

, (2.6)

. (2.7)

Используя определения для деформаций в плоскости, вместе с зависимостями кривизны, уравнения деформации/перемещения в плоскости можно характеризовать следующим образом:

, (2.8)

, т (2.9)

Согласно одной из гипотез Киркгоффа, уравнения плоских напряжений , , , которые связывают плоскостные напряжения с плоскостными деформациями для изотропного материала, характеризуются следующим образом:

, (2.10)

, (2.11)

, (2.12)

где .

На рисунке 2.3 показаны нормальные напряжения в плоскости и напряжения сдвига, действующие на края пластины.

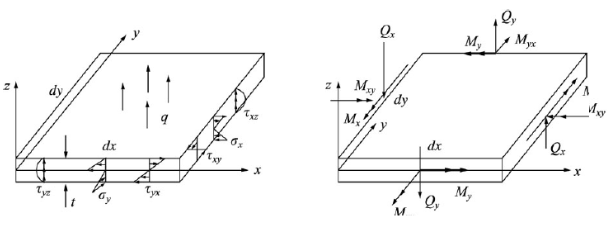


Рисунок 2.3 – Нормальные напряжения в плоскости и напряжения сдвига, действующие на края пластины

Напряжения изменяются линейно в направлении от срединной поверхности пластины. Поперечные напряжения сдвигов  и  также присутствуют, хотя поперечной деформацией сдвига обычно пренебрегают при расчётах. Эти напряжения изменяются квадратично по всей толщине пластины.

Жёсткость при изгибе пластины  рассчитывается по закону:

. (2.13)

Уравнения равновесия для прогиба пластины важны при выборе полей перемещений элементов. Основные дифференциальные уравнения для соблюдения законов равновесия при прогибе тонких пластин характеризуются следующим образом:

, (2.14)

, (2.15)

, (2.16)

где  – поперечная распределенная нагрузка. На рисунке 2.4 показана поперечная распределенная нагрузка на пластину.

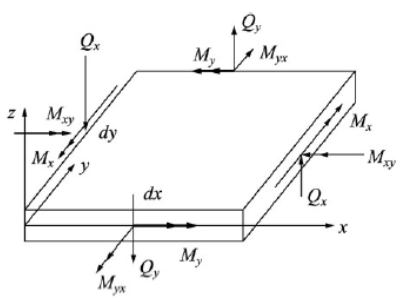


Рисунок 2.4 – Поперечная распределенная нагрузка на пластину

Основное дифференциальное уравнение в частных производных для изгиба тонкой пластины может быть выражено следующим образом:

. (2.17)

Решением задачи определения прогиба тонкой пластины является функция поперечного смещения . Если дифференцированием по направлению оси  пренебречь, уравнение (2.17) упрощается до уравнения для балки, а жесткость при изгибе пластины  приводится к жесткости изгиба балки , когда коэффициент Пуассона равен 0.

Для решения поставленной задачи согласно индивидуальному заданию были построены конечно-элементные модели нескольких типов тонких пластин. На рисунке 2.5 представлен пример построения конечно-элементной сетки одной из тонких пластин.

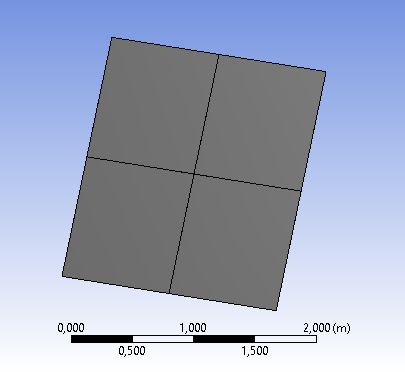


Рисунок 2.5 – Построение конечно-элементной модели тонкой пластины

После построения конечно-элементной модели пластины был проведен анализ прогиба данной пластины. Данный анализ был проведен с несколькими типами тонких пластин с различными размерами, толщиной, материалом и размером конечно-элементной сетки. На рисунке 2.6 показан расчёт прогиба тонкой пластины.

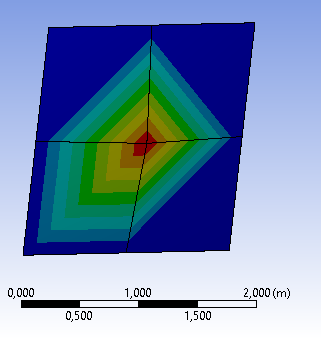


Рисунок 2.6 – Расчёт прогибов тонкой пластины под действием распределенной нагрузки

Результаты данных расчётов были использованы для обучения спроектированной нейронной сети.

* 1. **Архитектура спроектированной нейронной сети**

Регрессионный анализ служит для моделирования отношений между зависимой переменной, которую необходимо предсказать, и одной или несколькими независимыми переменными (входными данными модели) [7]. Простейшее линейное уравнение регрессии выглядит так:

, (2.18)

где  – значение, которое регрессионная модель стремится предсказать;

 – входные данные;

 – веса, определяющие важность переменных для предсказания;

 – расстояние между значением, прогнозируемым моделью, и фактической зависимой переменной .

Основные типы регрессионных моделей:

– линейная регрессия;

– полиномиальная регрессия;

– логистическая регрессия.

Искусственные нейронные сети (ИНС) состоят из простых элементов, называемых нейронами, каждый из которых может принимать простые математические решения. Вместе нейроны могут анализировать сложные проблемы, эмулировать практически любые функции, включая очень сложные, и предоставлять точные ответы. Нейронная сеть в общем случае имеет три слоя нейронов: входной слой, скрытый слой и выходной слой. Глубокая нейронная сеть (*Deep Neural Network*) имеет более одного скрытого слоя, что увеличивает сложность модели и может значительно улучшить предсказуемость. На рисунке 2.7 представлено сравнение архитектуры обычной и глубокой нейронных сетей.

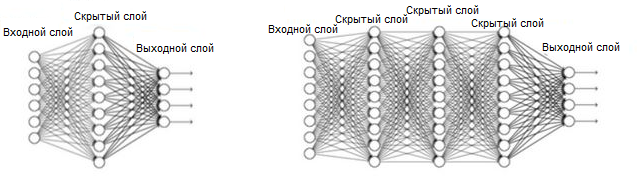


Рисунок 2.7 – Сравнение архитектуры обычной и глубокой нейронных сетей

Обучение нейронной сети в общем случае состоит из следующих этапов:

– инициализация сети;

– прямое распространение весов;

– вычисление ошибки;

– обратное распространение ошибки;

– обновление весов. Затем обновление весов повторяется до достижения сходимости.

Инициализация нейронной сети в начале обучения позволяет снизить время схождения весов нейросети. В некоторых случаях (например, при обучении с использованием стохастического градиентного спуска) без инициализации начальных весов будет невозможно обучить нейронную сеть. Существуют 2 основные техники инициализации нейронной сети:

– инициализация нулями;

– инициализация случайными малыми значениями.

Если все веса инициализированы нулями, производная по функции потерь одинакова для каждого веса  в матрице весов , где  – номер слоя нейронной сети. Таким образом, все веса имеют одинаковое значение в последующих итерациях. В данном случае скрытые нейроны симметричны для всех  итераций. Присвоение случайных значений весам лучше, чем просто присвоение 0. Но при этом важно знать, что является разумной инициализацией значений веса. Если инициализировать веса слишком большими, или слишком малыми значениями, то наклон градиента будет изменяться слишком медленно, и обучение займет очень большое количество времени.

Следующий этап в обучении нейронной сети – прямое распространение значений весов. Данный этап служит для первоначальной оценки работы алгоритма. Расчёт идёт напрямую от входного слоя нейронов, через сеть, до выходного слоя. На каждом нейроне в скрытом или выходном слое обработка сигнала происходит в два этапа:

– предварительная активация;

– активация.

Предварительная активация – это взвешенная сумма входных данных, то есть линейное преобразование весов с доступными входными данными. На основании этой агрегированной суммы и функции активации нейрон принимает решение, передавать эту информацию дальше или нет [8]. Рассчитанная взвешенная сумма входов затем передается в функцию активации. Функция активации – это математическая функция, которая добавляет нелинейность в сеть. Существует несколько широко используемых и популярных функций активации:

– сигмоидальная;

– тангенциальная гиперболическая (*tanh*);

– *ReLU*;

– *Softmax*.

После того, как выход сети посчитан, высчитывается ошибка сети с помощью функции ошибки. В обучении с учителем правильный результат всегда известен. Это позволяет найти разницу между правильным выходным значением и фактическим выходным значением модели с учетом весов текущей модели. Наиболее интуитивно понятной функцией ошибки является разница между правильным значением выхода сети и предсказанным. Если необходимо, чтобы функция ошибки отражала абсолютную ошибку производительности, то необходимо взять абсолютное значение ошибки. В общем случае функция ошибки является среднеквадратичной функцией ошибки (*mean squared error*, *MSE*):

, (2.19)

где  – правильное значение выхода сети;

 – спрогнозированное значение выхода.

Целью обратного распространения является изменение весов нейронов, чтобы свести к минимуму функцию ошибки. На рисунке 2.9 показана схема изменения весов нейронов в зависимости от рассчитанной среднеквадратичной ошибки.

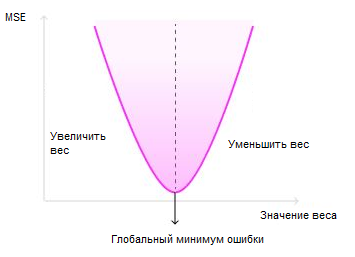


Рисунок 2.9 – Схема изменения весов нейронов в зависимости от рассчитанной среднеквадратичной ошибки

Существует общее правило для обновления весов нейронной сети – «дельта»-правило:

, (2.20)

где  – скорость обучения;

 – производная от функции активации нейрона;

 – взвешенная сумма входов нейрона.

Скорость обучения вводится как постоянная при инициализации сети. Чтобы избежать больших шагов при обучении сети, скорость обучения устанавливают очень маленькой. Веса обновляются по «дельта»-правилу следующим образом:

– если производная положительна, это означает, что увеличение веса увеличит ошибку, поэтому новый вес должен быть меньше;

– если производная отрицательна, это означает, что увеличение веса уменьшит ошибку, поэтому необходимо увеличить веса;

– если производная равна 0, это означает, что достигнут минимум, и обновление весов не требуется [9].

Для определения прогибов была спроектирована нейронная сеть прямого распределения, содержащая в себе 7 скрытых слоев. На рисунке 2.10 представлена архитектура нейронной сети.

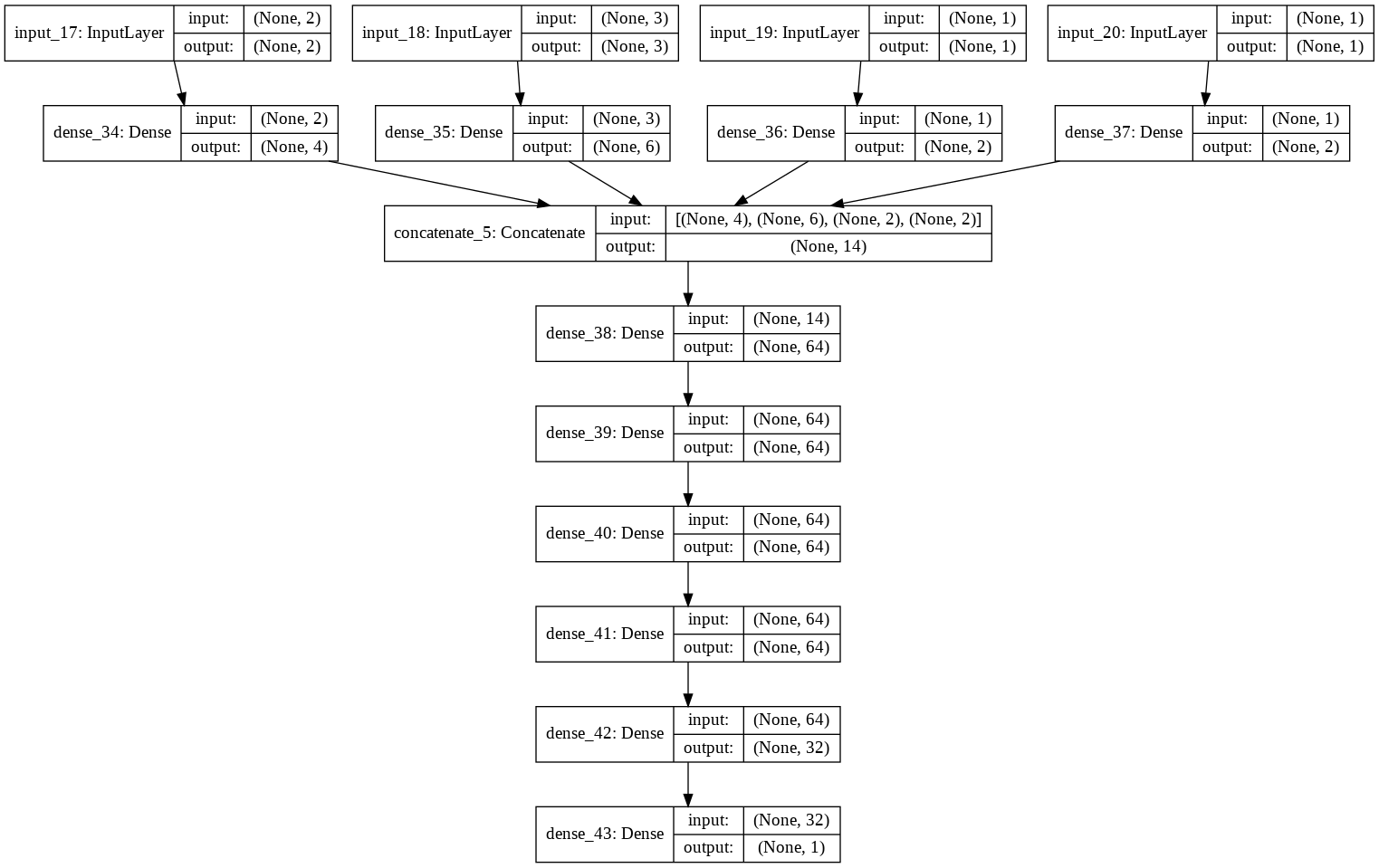


Рисунок 2.10 – Схема спроектированной нейронной сети

На входе сети располагаются 7 нейронов, сгруппированных в несколько связанных групп: координаты точки (, ), материал пластины (модуль упругости, коэффициент Пуассона, толщина), размер сетки и значение нагрузки. Выход сети представляет собой значение прогиба пластины в точке с координатами , . Общее количество параметров для тренировки равно 15597.

Для обучения сети был использован набор из 136 файлов. Общее количество строк данных, использованных для обучения сети, равно 282384. На рисунке 2.11 представлен пример использованных входных данных для обучения.

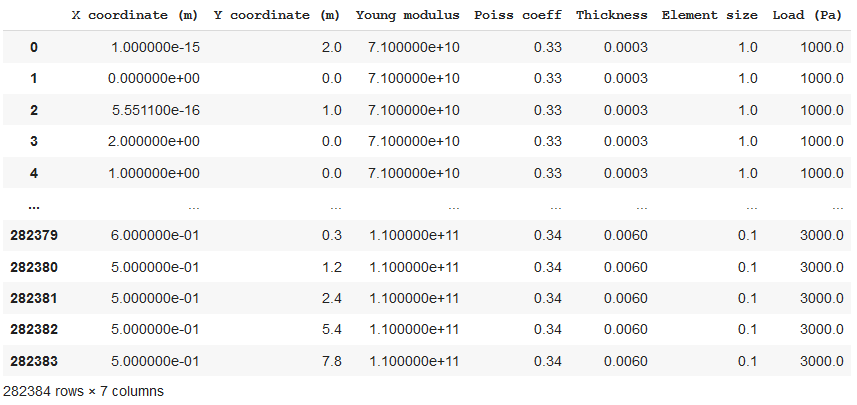


Рисунок 2.11 – Пример входных данных, использованных для обучения

Все данные для обучения были разделены на обучающую и тестовую выборки с соотношением 0.1. На рисунке 2.12 представлен пример выходных данных, использованных для обучения.

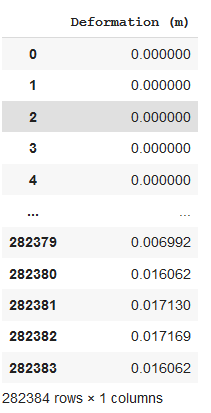


Рисунок 2.12 – Пример выходных данных, использованных для обучения

Метрика *MAPE* (*mean absolute percentage error*) – это одна из используемых функций ошибки для задач регрессии. Данная метрика может указываться в процентах, если умножить полученное значение на 100. Данная метрика выражает точность прогнозирования согласно формуле:

, (2.21)

где  – правильное значение;

 – предсказанное значение.

На рисунке 2.13 представлено значение метрики *MAPE* в ходе обучения нейронной сети на последних эпохах обучения.

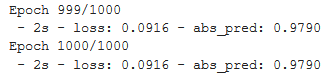


Рисунок 2.13 – Значение метрики *MAPE* на последних эпохах обучения нейронной сети

На основе спроектированной и обученной нейронной сети было разработано десктопное приложение для определения прогибов тонкой пластины.

* 1. **Общий вид разработанного программного обеспечения**

Десктопное приложение, использующее в своей основе спроектированную и обученную нейронную сеть, разработано на языке программирования *Python*. Для данного приложения был разработан графический интерфейс с возможностью просмотра результатов расчёта как в виде трёхмерного графика, так и в виде кратких сведений в текстовом виде. Имеется возможность манипулировать положением графика в пространстве после выполнения необходимых расчётов. Данное приложение позволяет ввести в специальные поля параметры пластины, которую необходимо проанализировать, и, по нажатию кнопки – провести расчёт и вывести результат на экран. На рисунке 2.14 представлен общий вид разработанного приложения после запуска.

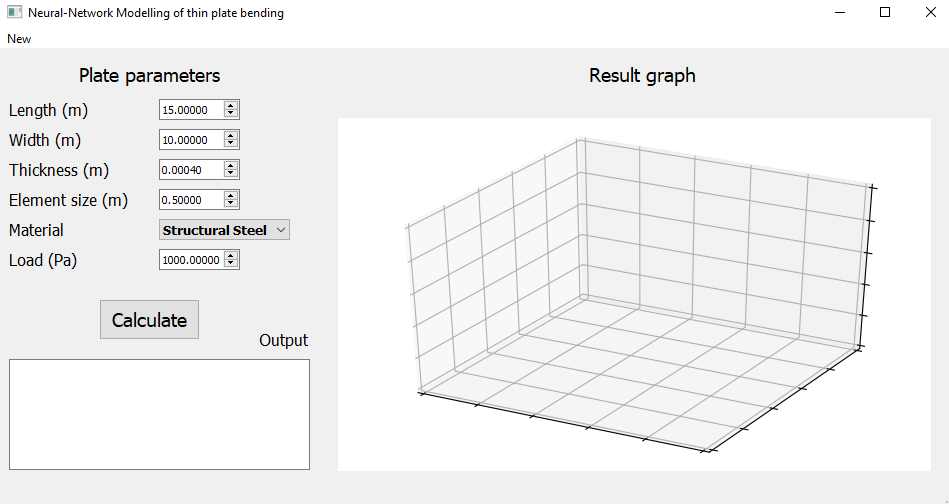


Рисунок 2.14 – Общий вид разработанного приложения

Также присутствует возможность вывода результатов анализа пластины в файл формата *CSV*. После завершения расчёта программа предлагает сохранить результаты расчёта в файл. На рисунке 2.15 представлен пример расчёта пластины.

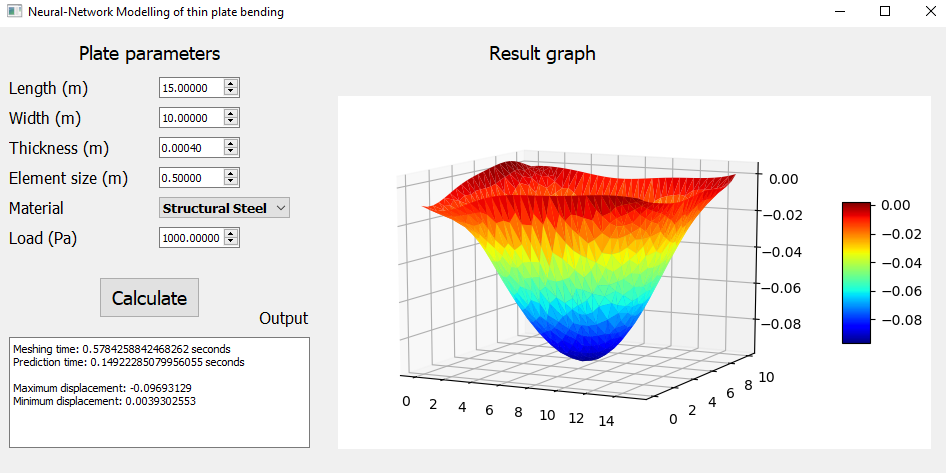


Рисунок 2.15 – Пример расчёта пластины

На рисунке 2.16 представлено диалоговое окно сохранения результатов в файл. По нажатию кнопки «*OK*» открывается диалоговое окно выбора папки, в которую пользователь желает сохранить файл

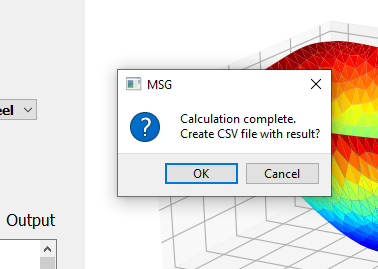


Рисунок 2.16 – Диалоговое окно сохранения результатов в файл

В файл записываются координаты точек пластины и значение прогиба в этих точках. На рисунке 2.17 представлен вид файла результатов.

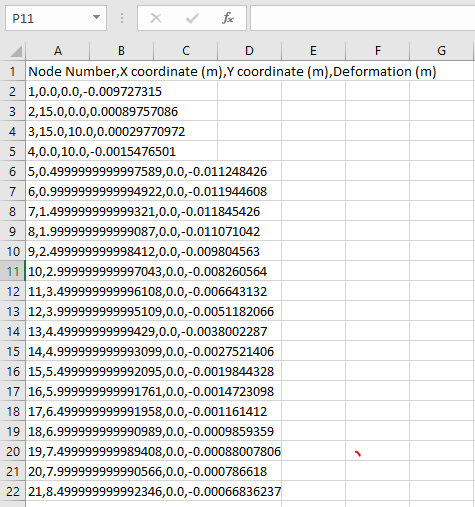


Рисунок 2.17 – Общий вид файла результатов

В окне программы также отображается краткий вывод следующих результатов расчёта:

– время разбиения пластины на сетку элементов (в секундах);

– время расчёта прогиба пластины (в секундах);

– максимальное значение прогиба пластины (в метрах);

– минимальное значение прогиба пластины (в метрах);

– среднее значение прогиба пластины (в метрах).

На рисунке 2.18 представлено поле краткого вывода результатов расчёта пластины.

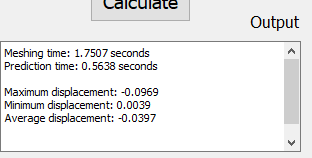


Рисунок 2.18 – Краткий вывод результатов расчёта

* 1. **Сравнение результатов расчёта тонкой пластины нейросетевым методом и методом конечных элементов**

В ходе обучения нейронной сети и её верификации была получена точность расчётов, составляющая 94 процента. Целью данной работы также являлось сравнение времени определения прогибов между методом конечных элементов и нейросетевым методом. Для примера расчётов были взяты две пластины с следующими характеристиками:

Таблица 1

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Пластина №1** | **Пластина №2** |
| Длина | 2 м | 15 |
| Ширина | 2 м | 10 |
| Толщина | 0.0003 м | 0.04 |
| Модуль упругости | 1.1e+11 | 2e+11 |
| Коэфф. Пуассона | 0.34 | 0.3 |
| Размер сетки | 1 м | 0.1 м |
| Значение нагрузки | 2000 *Pa* | 4000 *Pa* |

Сравнение времени расчётов приведено в таблице 2.

Таблица 2

|  |  |  |
| --- | --- | --- |
| **№ пластины** | **Метод конечных элементов** | **Нейронная сеть** |
| 1 | 4.3 с | 0.5782 с |
| 2 | 15.5 с | 0.6581 с |

Из результатов расчёта можно видеть, что размер сетки имеет очень малое влияние на скорость расчёта пластины, тогда как в методе конечных элементов размер сетки имеет прямое влияние на скорость расчётов.

**ЗАКЛЮЧЕНИЕ**

В ходе преддипломной практики был получен опыт работы на предприятии «IBA-Гомель-Парк». Были изучены принципы построения архитектуры нейронных сетей, также было изучено применение метода конечных элементов в строительной механике. Согласно индивидуальному заданию, было разработано программного обеспечение на языке программирования *Python* для определения прогибов тонкой пластины с помощью нейронной сети. Разработанное программное обеспечение является десктопным приложением, содержащим в себе графический интерфейс для управления приложением и модель нейронной сети, с помощью которой производятся необходимые расчёты. Для обучения нейронной сети был сгенерирован большой объем данных, рассчитанных с помощью метода конечных элементов. В ходе обучения сети была получена точность расчётов на тестовых данных, равная 97 процентов. Также в ходе верификации спроектированной нейронной сети было получено время выполнения расчётов, использованное в сравнении с временем расчётов методом конечных элементов. В ходе сравнения времени выполнения расчётов методом конечных элементов и нейросетевым методом было выявлено, что время выполнения расчётов нейронной сетью не зависит от размера используемой сетки, в отличие от метода конечных элементов.

**Список использованных источников**

1. Черноморец А. А. О применении нейронных сетей для решения дифференциальных уравнений в частных производных / А. А. Черноморец, А. Н. Коваленко, М. А. Петина. // Информатика, – 2017, № 9 (258), С. 103-110.

2. Курочка, К. С. Математическое моделирование напряженно-деформированного состояния сложных систем неоднородных упругопластических дисперсных и сплошных твердых тел / К. С. Курочка // Информатика, – 2007, №2 (14), С. 117-128.

3. Курочка, К. С. Исследование математических моделей в ГРИД-средах / К. С. Курочка // Известия Гомельского государственного университета имени Ф. Скорины, – 2008, №5 (50), Ч. 1, С. 69-73.

4. Zienkiewicz, O.C. TheFiniteElementMethod*,* fifthedition*,* Volume *2:* SolidMechanics / O.C*.* Zienkiewicz*,* R.L.Taylor – Butterwort*-*Heinemann – 2000. – 476 p.

5. Reddy, J. Theory andAnalysisofElasticPlatesandShells / J*.* Reddy – CRC Press, – 2006. – 2ndEdition. – 568 p.

6. Ventsel*,* E*.* ThinplatesandShells / E. Ventsel, T. Krauthammer. – Marcel Dekker, Inc. – 2001. – 651 p.

7. Васильев А. Н. Нейросетевой подход к задачам математической физики / А. Н. Васильев, Д. А. Тархов. – СПб.: Нестор-История, 2015. – 260 с.

8. He*,* S*.* Multilayerneuralnetworksforsolvingaclassofpartialdifferentialequations / S. He, K. Reif, R. Unbehauen // Neural Networks, – 2000. – Vol. 13. – P. 385-396.

9. Nguyen*-*Thien*,* T. Approximation offunctionsandtheirderivatives:Aneuralnetworkimplementationwithapplications / T. Nguyen-Thien, T. Tran-Cong // AppliedMathematicalModelling. – 2003. – P. 197— 220.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

*utils.py*:

def average(lst):

    return sum(lst) / len(lst)

materials = {

    'Structural Steel': [2e+11, 0.3],

    'Aluminum Alloy': [7.1e+10, 0.33],

    'Copper Alloy': [1.1e+11, 0.34],

    'Concrete': [3e+10, 0.18]

}

def abs\_pred(y\_true, y\_pred):

        return abs((y\_true - y\_pred) / y\_true) < 0.1

dependencies = {

    'abs\_pred': abs\_pred

}

*mplwidget.py*:

from PyQt5 import QtCore, QtGui, QtWidgets

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg

from matplotlib.figure import Figure

from mpl\_toolkits.mplot3d import Axes3D

class MplWidget(QtWidgets.QWidget):

    def \_\_init\_\_(self, parent=None):

        QtWidgets.QWidget.\_\_init\_\_(self ,  parent)

        self.canvas = FigureCanvasQTAgg(Figure())

        vertical\_layout = QtWidgets.QVBoxLayout()

        vertical\_layout.addWidget(self.canvas)

        self.canvas.axes = Axes3D(self.canvas.figure)

        self.setLayout(vertical\_layout)

*data.py*:

import numpy as np

import utils

def form\_input\_data(coordinates, material, thickness, elementSize, load):

    input\_data = np.zeros([coordinates.shape[0], 7])

    young = utils.materials[material][0]

    poiss = utils.materials[material][1]

    # x

    input\_data[:, 0] = coordinates[:, 0]

    # y

    input\_data[:, 1] = coordinates[:, 1]

    # E

    input\_data[:, 2] = np.array([young] \* coordinates.shape[0])

    # poiss

    input\_data[:, 3] = np.array([poiss] \* coordinates.shape[0])

    # thickness

    input\_data[:, 4] = np.array([thickness] \* coordinates.shape[0])

    # element size

    input\_data[:, 5] = np.array([elementSize] \* coordinates.shape[0])

    # load

    input\_data[:, 6] = np.array([load] \* coordinates.shape[0])

    return input\_data

*form\_mesh.py*:

import calfem.geometry as cfg

import calfem.mesh as cfm

def form\_rectangular\_mesh(length, width, thickness, elementSize):

    g = cfg.Geometry()

    g.point([0.0, 0.0])

    g.point([length, 0.0])

    g.point([length, width])

    g.point([0.0, width])

    g.spline([0, 1])

    g.spline([1, 2])

    g.spline([2, 3])

    g.spline([3, 0])

    g.surface([0, 1, 2, 3])

    mesh = cfm.GmshMesh(g, min\_size=elementSize, max\_size=elementSize)

    mesh.elType = 3

    mesh.dofsPerNode = 2

    mesh.elSizeFactor = 1

    coords, edof, dofs, bdofs, elementmarkers = mesh.create()

    return coords, edof, dofs, bdofs, elementmarkers

*get\_result.py*:

from form\_mesh import form\_rectangular\_mesh

from data import form\_input\_data

import time

def get\_result(length, width, thickness, material, elementSize, force, model, input\_scaler, output\_scaler):

    meshing\_start = time.time()

    coords, edof, dofs, bdofs, elementmarkers = form\_rectangular\_mesh(length, width, thickness, elementSize)

    meshing\_end = time.time()

    input\_data = form\_input\_data(coords, material, thickness, elementSize, force)

    input\_data\_scaled = input\_scaler.transform(input\_data)

    input\_data\_scaled\_xy = input\_data\_scaled[:,0:2]

    input\_data\_scaled\_material = input\_data\_scaled[:,2:5]

    input\_data\_scaled\_elementSize = input\_data\_scaled[:,5]

    input\_data\_scaled\_load = input\_data\_scaled[:,6]

    prediction\_start = time.time()

    scaled\_predict = model.predict([input\_data\_scaled\_xy, input\_data\_scaled\_material, input\_data\_scaled\_elementSize, input\_data\_scaled\_load])

    prediction\_end = time.time()

    log = 'Prediction time: ' + str(round(prediction\_end - prediction\_start, 4)) + ' seconds\n'

    real\_predict = output\_scaler.inverse\_transform(scaled\_predict)

    input\_data\_xy = input\_data[:,0:2]

    input\_data\_x = input\_data\_xy.T[0]

    input\_data\_y = input\_data\_xy.T[1]

    xx=input\_data\_x

    yy=input\_data\_y

    zz=real\_predict \*-1

    x=xx.ravel()

    y=yy.ravel()

    predict=zz.ravel()

    return x, y, predict, log

*main.py*:

# -\*- coding: utf-8 -\*-

from PyQt5 import QtCore, QtGui, QtWidgets

from mplwidget import MplWidget

from matplotlib.backends.backend\_qt5agg import(NavigationToolbar2QT as NavigationToolbar)

import numpy as np

from get\_result import get\_result

from matplotlib import cm

from keras import models

import os

import utils

from pickle import load

import csv

class Ui\_MainWindow(QtWidgets.QMainWindow):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(955, 480)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.label\_inputParams = QtWidgets.QLabel(self.centralwidget)

        self.label\_inputParams.setGeometry(QtCore.QRect(80, 10, 151, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        self.label\_inputParams.setFont(font)

        self.label\_inputParams.setObjectName("label\_inputParams")

        self.label\_graph = QtWidgets.QLabel(self.centralwidget)

        self.label\_graph.setGeometry(QtCore.QRect(590, 10, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        self.label\_graph.setFont(font)

        self.label\_graph.setObjectName("label\_graph")

        self.MplWidget = MplWidget(self.centralwidget)

        self.MplWidget.setGeometry(QtCore.QRect(330, 60, 611, 371))

        self.MplWidget.setObjectName("MplWidget")

        self.pushButton\_calculate = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton\_calculate.setGeometry(QtCore.QRect(100, 250, 101, 41))

        font = QtGui.QFont()

        font.setPointSize(14)

        self.pushButton\_calculate.setFont(font)

        self.pushButton\_calculate.setObjectName("pushButton\_calculate")

        self.plainTextEdit\_output = QtWidgets.QPlainTextEdit(self.centralwidget)

        self.plainTextEdit\_output.setGeometry(QtCore.QRect(10, 310, 301, 111))

        self.plainTextEdit\_output.setObjectName("plainTextEdit\_output")

        self.doubleSpinBox\_length = QtWidgets.QDoubleSpinBox(self.centralwidget)

        self.doubleSpinBox\_length.setGeometry(QtCore.QRect(160, 50, 81, 21))

        self.doubleSpinBox\_length.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.doubleSpinBox\_length.setDecimals(5)

        self.doubleSpinBox\_length.setMaximum(999999.99)

        self.doubleSpinBox\_length.setObjectName("doubleSpinBox\_length")

        self.doubleSpinBox\_width = QtWidgets.QDoubleSpinBox(self.centralwidget)

        self.doubleSpinBox\_width.setGeometry(QtCore.QRect(160, 80, 81, 21))

        self.doubleSpinBox\_width.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.doubleSpinBox\_width.setDecimals(5)

        self.doubleSpinBox\_width.setMaximum(999999.99)

        self.doubleSpinBox\_width.setObjectName("doubleSpinBox\_width")

        self.doubleSpinBox\_thickness = QtWidgets.QDoubleSpinBox(self.centralwidget)

        self.doubleSpinBox\_thickness.setGeometry(QtCore.QRect(160, 110, 81, 21))

        self.doubleSpinBox\_thickness.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.doubleSpinBox\_thickness.setDecimals(5)

        self.doubleSpinBox\_thickness.setMaximum(999999.99)

        self.doubleSpinBox\_thickness.setObjectName("doubleSpinBox\_thickness")

        self.comboBox\_material = QtWidgets.QComboBox(self.centralwidget)

        self.comboBox\_material.setGeometry(QtCore.QRect(160, 170, 131, 21))

        font = QtGui.QFont()

        font.setPointSize(10)

        font.setBold(True)

        font.setWeight(75)

        self.comboBox\_material.setFont(font)

        self.comboBox\_material.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.comboBox\_material.setMaxVisibleItems(10)

        self.comboBox\_material.setObjectName("comboBox\_material")

        self.comboBox\_material.addItem("")

        self.comboBox\_material.addItem("")

        self.comboBox\_material.addItem("")

        self.comboBox\_material.addItem("")

        self.doubleSpinBox\_elementSize = QtWidgets.QDoubleSpinBox(self.centralwidget)

        self.doubleSpinBox\_elementSize.setGeometry(QtCore.QRect(160, 140, 81, 21))

        self.doubleSpinBox\_elementSize.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.doubleSpinBox\_elementSize.setDecimals(5)

        self.doubleSpinBox\_elementSize.setMaximum(999999.99)

        self.doubleSpinBox\_elementSize.setObjectName("doubleSpinBox\_elementSize")

        self.doubleSpinBox\_load = QtWidgets.QDoubleSpinBox(self.centralwidget)

        self.doubleSpinBox\_load.setGeometry(QtCore.QRect(160, 200, 81, 21))

        self.doubleSpinBox\_load.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

        self.doubleSpinBox\_load.setDecimals(5)

        self.doubleSpinBox\_load.setMaximum(999999.99)

        self.doubleSpinBox\_load.setObjectName("doubleSpinBox\_load")

        self.label\_length = QtWidgets.QLabel(self.centralwidget)

        self.label\_length.setGeometry(QtCore.QRect(10, 50, 81, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_length.setFont(font)

        self.label\_length.setObjectName("label\_length")

        self.label\_width = QtWidgets.QLabel(self.centralwidget)

        self.label\_width.setGeometry(QtCore.QRect(10, 80, 81, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_width.setFont(font)

        self.label\_width.setObjectName("label\_width")

        self.label\_thickness = QtWidgets.QLabel(self.centralwidget)

        self.label\_thickness.setGeometry(QtCore.QRect(10, 110, 101, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_thickness.setFont(font)

        self.label\_thickness.setObjectName("label\_thickness")

        self.label\_elementSize = QtWidgets.QLabel(self.centralwidget)

        self.label\_elementSize.setGeometry(QtCore.QRect(10, 140, 121, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_elementSize.setFont(font)

        self.label\_elementSize.setObjectName("label\_elementSize")

        self.label\_material = QtWidgets.QLabel(self.centralwidget)

        self.label\_material.setGeometry(QtCore.QRect(10, 170, 61, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_material.setFont(font)

        self.label\_material.setObjectName("label\_material")

        self.label\_load = QtWidgets.QLabel(self.centralwidget)

        self.label\_load.setGeometry(QtCore.QRect(10, 200, 71, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_load.setFont(font)

        self.label\_load.setObjectName("label\_load")

        self.label\_output = QtWidgets.QLabel(self.centralwidget)

        self.label\_output.setGeometry(QtCore.QRect(260, 280, 51, 21))

        font = QtGui.QFont()

        font.setPointSize(12)

        self.label\_output.setFont(font)

        self.label\_output.setObjectName("label\_output")

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 755, 21))

        self.menubar.setObjectName("menubar")

        self.menuNew= QtWidgets.QMenu(self.menubar)

        self.menuNew.setObjectName("menuNew")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.actionReset = QtWidgets.QAction(MainWindow)

        self.actionReset.setObjectName("actionReset")

        self.menuNew.addAction(self.actionReset)

        self.menubar.addAction(self.menuNew.menuAction())

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.pushButton\_calculate.clicked.connect(self.update\_graph)

        self.actionReset.triggered.connect(self.clear)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "Neural-Network Modelling of thin plate bending"))

        self.label\_inputParams.setText(\_translate("MainWindow", "Plate parameters"))

        self.label\_graph.setText(\_translate("MainWindow", "Result graph"))

        self.pushButton\_calculate.setText(\_translate("MainWindow", "Calculate"))

        self.comboBox\_material.setItemText(0, \_translate("MainWindow", "Structural Steel"))

        self.comboBox\_material.setItemText(1, \_translate("MainWindow", "Concrete"))

        self.comboBox\_material.setItemText(2, \_translate("MainWindow", "Copper Alloy"))

        self.comboBox\_material.setItemText(3, \_translate("MainWindow", "Aluminum Alloy"))

        self.label\_length.setText(\_translate("MainWindow", "Length (m)"))

        self.label\_width.setText(\_translate("MainWindow", "Width (m)"))

        self.label\_thickness.setText(\_translate("MainWindow", "Thickness (m)"))

        self.label\_elementSize.setText(\_translate("MainWindow", "Element size (m)"))

        self.label\_material.setText(\_translate("MainWindow", "Material"))

        self.label\_load.setText(\_translate("MainWindow", "Load (Pa)"))

        self.label\_output.setText(\_translate("MainWindow", "Output"))

        self.menuNew.setTitle(\_translate("MainWindow", "New"))

        self.actionReset.setText(\_translate("MainWindow", "Reset"))

        self.initializePlateCharacteristics()

    def update\_graph(self):

        self.plainTextEdit\_output.clear()

        x, y, z, log = get\_result(

            self.doubleSpinBox\_length.value(),

            self.doubleSpinBox\_width.value(),

            self.doubleSpinBox\_thickness.value(),

            self.comboBox\_material.currentText(),

            self.doubleSpinBox\_elementSize.value(),

            self.doubleSpinBox\_load.value(),

            self.model,

            self.input\_scaler,

            self.output\_scaler

        )

        log += '\n'

        log += 'Maximum displacement: ' + str(round(min(z), 4)) + '\n'

        log += 'Minimum displacement: ' + str(round(max(z), 4)) + '\n'

        log += 'Average displacement: ' + str(round(utils.average(z), 4)) + '\n'

        self.plainTextEdit\_output.insertPlainText(log + '\n')

        self.MplWidget.canvas.axes.clear()

        surface = self.MplWidget.canvas.axes.plot\_trisurf(x, y, z, cmap=cm.jet, linewidth=0.1)

        #self.MplWidget.canvas.figure.colorbar(surface, shrink=0.4, aspect=5)

        self.MplWidget.canvas.draw()

        self.show\_message\_box(x, y, z)

    def initializePlateCharacteristics(self):

        self.doubleSpinBox\_length.setValue(15)

        self.doubleSpinBox\_width.setValue(10)

        self.doubleSpinBox\_thickness.setValue(0.0004)

        self.doubleSpinBox\_elementSize.setValue(0.5)

        self.doubleSpinBox\_load.setValue(1000)

        fileDir = os.path.dirname(os.path.abspath(\_\_file\_\_))

        self.model = models.load\_model(fileDir + '\\model\\model.h5', custom\_objects=utils.dependencies)

        self.input\_scaler = load(open(fileDir + '\\model\\min\_max\_scaler.pkl', 'rb'))

        self.output\_scaler = load(open(fileDir + '\\model\\standard\_scaler.pkl', 'rb'))

        #plot\_model(self.model, to\_file=fileDir + '\\model\\model\_plot.png', show\_shapes=True, show\_layer\_names=True)

    def clear(self):

        self.plainTextEdit\_output.clear()

        self.MplWidget.canvas.axes.clear()

        self.MplWidget.canvas.draw()

    def saveFileDialog(self, x, y, z):

        options = QtWidgets.QFileDialog.Options()

        options |= QtWidgets.QFileDialog.DontUseNativeDialog

        fileName, \_ = QtWidgets.QFileDialog.getSaveFileName(self, "Save result to CSV file", "" ,"All Files (\*);;CSV Files (\*.csv)", options=options)

        if fileName:

            datalist = [['Node Number', 'X coordinate (m)', 'Y coordinate (m)', 'Deformation (m)']]

            for i, row in enumerate(z):

                datarow = [i+1, x[i], y[i], z[i]]

                datalist.append(datarow)

            with open(fileName, 'w', newline='') as file:

                writer = csv.writer(file)

                writer.writerows(datalist)

    def writeCsvFile(self, x, y, z):

        self.saveFileDialog(x, y, z)

    def show\_message\_box(self, x, y, z):

        msg = QtWidgets.QMessageBox()

        msg.setWindowTitle("MSG")

        msg.setText("Calculation complete.\nCreate CSV file with result?")

        msg.setIcon(QtWidgets.QMessageBox.Question)

        msg.setStandardButtons(QtWidgets.QMessageBox.Cancel|QtWidgets.QMessageBox.Ok)

        returnValue = msg.exec()

        if (returnValue == QtWidgets.QMessageBox.Ok):

            self.writeCsvFile(x, y, z)

if \_\_name\_\_ == "\_\_main\_\_":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    MainWindow = QtWidgets.QMainWindow()

    ui = Ui\_MainWindow()

    ui.setupUi(MainWindow)

    MainWindow.show()

    sys.exit(app.exec\_())