Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 27.04.04 «Управление в технических системах»
Отделение автоматизации и робототехники

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Синтез нейронного регулятора для управления бесколлекторным двигателем постоянного тока

УДК 681.515:621.313.292.024

Студент

Группа	ФИО	Подпись	Дата
8АМ61	Усольцев Денис Вячеславович		

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Руководитель ВКР	Яковлева Елена Максимовна	к.т.н.		
Руководитель ООП	Пушкарев Максим Иванович	к.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Баннова Кристина Алексеевна	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Авдеева Ирина Ивановна			

ДОПУСТИТЬ К ЗАЩИТЕ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Руководитель ОАР	Леонов Сергей Владимирович	к.т.н.		

Томск – 2018 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

по направлению 27.03.04 «Управление в технических системах»

Код результата	Результат обучения (Выпускник должен быть готов)
	Профессиональные компетенции
Р1	Применять глубокие естественнонаучные и математические знания для решения научных и инженерных задач в области анализа, синтеза, проектирования, производства и эксплуатации средств автоматизации и систем управления техническими объектами.
Р2	Уметь обрабатывать, анализировать и обобщать научно-техническую информацию, передовой отечественный и зарубежный опыт в области теории, проектирования, производства и эксплуатации средств автоматизации и систем управления техническими объектами.
Р3	Ставить и решать инновационные задачи инженерного анализа, связанные с разработкой технических систем управления с использованием аналитических методов и сложных моделей.
Р4	Выполнять инновационные инженерные проекты по разработке программно-аппаратных средств автоматизированных систем различного назначения с использованием современных методов проектирования, систем автоматизированного проектирования, передового опыта разработки конкурентно способных изделий.
Р5	Планировать и проводить теоретические и экспериментальные исследования в области проектирования аппаратных и программных средств автоматизированных систем с использованием новейших достижений науки и техники, передового отечественного и зарубежного опыта. Критически оценивать полученные данные и делать выводы.
Р6	Осуществлять авторское сопровождение процессов проектирования, внедрения и эксплуатации программно-аппаратных средств автоматизированных систем различного назначения.
Р7	Владеть иностранным языком на уровне, позволяющем работать в интернациональной профессиональной среде с пониманием культурных, языковых и социально-экономических различий партнеров.
Р8	Осуществлять коммуникации в профессиональной среде и в обществе в целом, активно владеть иностранным языком, разрабатывать документацию, презентовать и защищать результаты инновационной инженерной деятельности, в том числе на иностранном языке.
Р9	Эффективно работать индивидуально и в качестве члена и руководителя группы, в том числе междисциплинарной и международной, при решении инновационных инженерных задач.
Р10	Демонстрировать личную ответственность и ответственность за работу возглавляемого коллектива, приверженность и готовность следовать профессиональной этике и нормам ведения инновационной инженерной деятельности. Демонстрировать глубокие знания правовых, социальных, экологических и культурных аспектов инновационной инженерной деятельности.
Р11	Демонстрировать способность к самостоятельному обучению, непрерывному самосовершенствованию в инженерной деятельности, способность к педагогической деятельности.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 27.04.04 «Управление в технических системах»
Отделение автоматизации и робототехники

УТВЕРЖДАЮ:
Руководитель ООП

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Магистерской диссертации

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	**ФИО**
8АМ61	Усольцеву Денису Вячеславовичу

Тема работы:

Синтез нейронного регулятора для управления бесколлекторным двигателем постоянного тока	
Утверждена приказом директора (дата, номер)	№2181/с от 28 марта 2018 г.

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Микроконтроллер Robotdyn Nano V3, бесколлекторный двигатель постоянного тока X2216 KV880, электронный регулятор хода Skywalker 40A, датчик освещенности.

Перечень подлежащих исследованию, проектированию и разработке вопросов	Анализ существующих технических решений на основе нейронных регуляторов. Разработка стенда для испытания нейронного регулятора. Программная реализация и исследование нейронного регулятора.
Перечень графического материала	Презентация в формате *.pptx

Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Баннова Кристина Алексеевна
Социальная ответственность	Авдеева Ирина Ивановна
Обязательное приложение на иностранном языке	Шепетовский Денис Владимирович

Названия разделов, которые должны быть написаны на русском и иностранном языках:

1. ОБЗОР ЛИТЕРАТУРЫ

2. ОБЪЕКТЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Яковлева Елена Максимовна	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8АМ61	Усольцев Денис Вячеславович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8АМ61	Усольцев Денис Вячеславович

Инженерная школа информационных технологии и робототехники		Отделение автоматизации и робототехники	
Уровень образования	магистратура	Направление/специальность	27.04.04 «Управление в технических системах»

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ)	Затраты на сырье, материалы, комплектующие изделия, специальное оборудование, основную и дополнительную заработную платы исполнителей, отчисления на социальные нужды, накладные расходы.
2. Нормы и нормативы расходования ресурсов	Статистические бюллетени и издания, нормативно-правовые документы.
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	В соответствии со стандартной системой налогообложения, отчислений, кредитования.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого и инновационного потенциала НТИ	Определение потенциальных потребителей результатов исследования, анализ конкурентных технических решений.
2. Инициация проекта	Планирование комплекса предполагаемых работ и формирование рабочей группы.
3. Планирование процесса управления НТИ	Составление перечня этапов и работ по выполнению НТИ, расчет бюджета НТИ.
4. Определение ресурсной, финансовой, экономической эффективности	Оценка научно технического уровня НТИ.
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. Карта сегментирование рынка *2.* Оценка конкурентоспособности технических решений *3.* Диаграмма Исикавы *4.* Матрица SWOT *5.* График проведения и бюджет НТИ *6.* Оценка ресурсной, финансовой и экономической эффективности НТИ	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Баннова Кристина Алексеевна	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8АМ61	Усольцев Денис Вячеславович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8АМ61	Усольцев Денис Вячеславович

Инженерная школа информационных технологии и робототехники		Отделение автоматизации и робототехники	
Уровень образования	магистратура	Направление/специальность	27.04.04 «Управление в технических системах»

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения.	1. Исследуется возможность управления двигателем посредством нейронного регулятора. Разработка программной и аппаратной частей проходит в аудитории 117-А 10-го корпуса ТПУ. Результаты работы могут быть использованы в учебных целях.
2. Перечень законодательных и нормативных документов по теме.	2. МС CSR-08260008000 ГОСТ 12.0.003-2015 СН 2.2.4/2.1.8.562–96 СанПиН 2.2.2/2.4.1340-03 СанПиН 2.2.4/2.1.8.055-96 СанПиН 2.2.4.548–96 СНиП 23-05-95 СанПиН 2.2.1/2.1.1.1278–03 ГОСТ Р 12.1.019-2009 ССБТ ГОСТ 30772-2001 ГОСТ 12.1.004–91 ССБТ ФЗ-№197 от 30.12.2001 ГОСТ 12.2.032-78
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения.	1. Выявленные вредные факторы: – повышенный уровень шума; – повышенный уровень электромагнитных излучений; – отклонение показателей микроклимата; – недостаточная освещенность; – психофизические нагрузки.
2. Анализ выявленных опасных факторов при разработке и эксплуатации проектируемого решения.	2. Выявленные опасные факторы: – поражение электрическим током; – статическое электричество; – короткое замыкание.
3. Охрана окружающей среды.	3. Источники загрязнения окружающей среды: – коллекторный двигатель постоянного тока; – лампы осветительных приборов.
4. Защита в чрезвычайных ситуациях.	4. Возможно ЧС техногенного характера: – пожар.

5. Правовые и организационные вопросы обеспечения безопасности.	5. Трудовая деятельность осуществляется в соответствии с ТК РФ, организация рабочего места в соответствии с ГОСТ 12.2.032-78 и СанПиН 2.2.2/2.4.1340–03.
Перечень графического материала:	
При необходимости представить эскизные графические материалы к расчётному заданию (обязательно для специалистов и магистров)	План эвакуации при пожаре, зоны для выполнения ручных операций и размещения органов управления.

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Авдеева Ирина Ивановна			01.03.2018

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8АМ61	Усольцев Денис Вячеславович		01.03.2018

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 27.04.04 «Управление в технических системах»
Уровень образования – магистр
Отделение автоматизации и робототехники
Период выполнения (осенний / весенний семестр 2017/2018 учебного года)

Форма представления работы:

Магистерская диссертация

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	8.06.2018 г.

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
	Основная часть	60
	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	15
	Социальная ответственность	15
	Обязательное приложение на иностранном языке	10

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Яковлева Елена Максимовна	к.т.н.		

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Пушкарев Максим Иванович	к.т.н.		

РЕФЕРАТ

Выпускная квалификационная работа содержит 150 страниц, 67 рисунков, 24 таблицы, 31 литературный источник, 3 приложения.

Ключевые слова: искусственная нейронная сеть, бесколлекторный двигатель постоянного тока, метод обратного распространения ошибки, нейроэмулятор, мультимодульный нейроконтроллер, Robotdyn, C#.

Объектом исследования является бесколлекторный двигатель постоянного тока.

Цель работы – синтез системы управления бесколлекторным двигателем постоянного тока на основе нейронной сети.

В процессе исследования проводился анализ существующих методов нейроуправления, разработка учебного стенда, идентификация двигателя, программная реализация нейронного регулятора, синтез различных структур нейрорегуляторов.

В результате исследования была написана и отлажена программа для обучения и исследования нейронных сетей, произведено сравнение и выявлены преимущества и недостатки различных структур нейрорегуляторов, реализована методика идентификации состояния объекта.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: испытательный стенд на основе бесколлекторного двигателя постоянного тока, работающий в пределах от 1 до 20 оборотов в секунду, устройство для динамического изменения нагрузки на валу двигателя от холостого хода до полной блокировки с возможностью фиксации положения, программный комплекс способный генерировать обучающие выборки, обучать нейронные сети и сохранять результаты в файл.

Степень внедрения: опытная эксплуатация.

Область применения: образовательный процесс для изучения структуры и принципа действия нейронных сетей. Разработка и исследование структур нейрорегуляторов для реальных объектов.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИНС – искусственная нейронная сеть.

ПИД – пропорционально-интегрально-дифференцирующий.

ПИ – пропорционально-интегрирующий.

ШИМ – широтно-импульсная модуляция.

ПК – персональный компьютер.

ООП – объектно-ориентированное программирование.

USB – universal serial bus.

UART – universal asynchronous receiver-transmitter.

GND – ground.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Применение искусственных нейронных сетей для задач управления является перспективной направлением, находящимся на стыке таких дисциплин, как автоматическое управление и искусственный интеллект.

Нейронные сети обладают рядом уникальных свойств, которые выделяют их на фоне иных средств управления: способностью к обобщению данных, обучению на примерах и адаптации к изменяющимся свойствам объекта управления и внешней среды. Кроме того нейронные сети пригодны для синтеза нелинейных регуляторов и обладают высокой надежностью, ввиду заложенного в структуру сети параллелизма.

В данной работе рассматривается возможность синтеза нейронного регулятора для задачи управления бесколлекторным двигателем постоянного тока. В качестве искусственной нейронной сети используется многослойный персептрон Розенблатта. Обучение производится методом обратного распространения ошибки.

В ходе выполнения работы разработан учебный стенд с возможностью динамического изменения нагрузки на валу бесколлекторного двигателя и написана программа для обучения нейронных и исследования полученных решений. Полученное решение может быть использовано в учебных целях или модифицировано для задач идентификации состояний реальных производственных объектов.

1. ОБЗОР ЛИТЕРАТУРЫ

Начало исследований в области искусственных нейронных сетей было положено в начале 1940-х годов, как следствие исследований, направленных на изучение биологических процессов в головном мозге. Первая математическая модель нейронной сети была создана в 1943 нейролингвистом Уолтером Питтсом и нейропсихологом Уорреном Мак-Каллоком. Сеть работала с двоичными числами и обладала способностью к обучению. Первая аппаратная реализация искусственной нейронной сети была сконструирована в 1958 Френком Розенблаттом. Нейрокомпьютер «Марк-I» решал задачу распознавания букв на картах с помощью встроенных фотоэлементов [1].

Разработанные модели искусственных нейронных сетей имели серьезные ограничения, связанные с невозможностью представления линейно неразделимых множеств. Ограничения могли быть преодолены с помощью использования многослойных нейронных сетей с использованием скрытых слоев, но на тот момент не были выработаны методы обучения таких сетей. Данное обстоятельство заморозило развитие теории нейронных сетей вплоть до 1982, когда Джоном Хопфилдом были разработаны методы, реализующие двухсторонний обмен информацией между нейронами [2].

В настоящее время искусственные нейронные сети получили большое распространение благодаря развитию вычислительной техники. Основным приложением искусственных нейронных сетей является обработка и анализ данных различного рода. Наибольшую популярность снискала задача распознавания изображений. Например, нейронные сети применяются для идентификации пользователя смартфона [3]. Так же нейронные сети применяются для поиска закономерностей во временных рядах и с успехом используются для прогноза курса акций и валют [4]. Многие компании используют нейронные сети в составе своих продуктов. Так, например, компания Google использует систему нейронного машинного перевода в своем

продукте Google Translate. Система способна переводить с более чем ста языков на английский и обратно [5].

Искусственные нейронные, в том числе, применяются для управления технологическими процессами. В частности, для управления динамическими объектами в составе нейрорегулятора. Впервые идея использования нейронной сети для управления динамическим объектом была предложена Бернардом Видроу в 1964 году [6]. Разработанная система представляла собой адаптивный регулятор, меняющий свои параметры в зависимости от закономерностей в структуре объекта, распознаваемых нейронной сетью.

Основной проблемой при решении задач управления динамическими объектами является получение инверсной модели объекта. Трудности получения инверсной модели обусловлены сложными причинно-следственными связями поведения и общей нелинейностью реального объекта. Применение нейронных сетей позволяет частично решить данную проблему, путем поиска приближенных значении на основе обучения на примерах поведения реального объекта.

В процессе развития теории нейроуправления, были рассмотрены нейроконтроллеры с использованием различных типов нейронных сетей: однослойных персептронов, сетей Кохонена, многослойных персептронов. Лучшие результаты показали многослойные нейроны с линией задержек [7]. Для построения нейрорегуляторов применяются различные архитектуры, которые можно разделить на прямые, при которых объект управляется непосредственно нейронной сетью, и косвенные, при которых нейронная сеть служит, для подстройки параметров основного регулятора или решает задачу минимизации определенного критерия. Среди прямых методов выделяются обобщенный инверсный метод, при котором нейронная сеть выступает в роли инверсной нейромодели объекта, и метод многомодульного нейроуправления, в котором используются пары инверсной и прямой нейромоделей, адекватных для определенных состояний объекта.

В литературе рассмотрено множество вариантов применения нейронных сетей для управления динамическими объектами. В статье [8] автор рассматривает возможность управления электроприводом с помощью адаптивного ПИД–регулятора, коэффициенты которого настраиваются с помощью нейронной сети. Среди преимуществ выявлены повышенное, в сравнении с классическим ПИД–регулятором, качество управления, а так же постоянная адаптация к изменяющимся параметрам объекта управления. В тоже время отмечается высокая сложность процедуры синтеза такого регулятора и непригодность реализованного решения для другого объекта управления.

В статье [9] рассматривается применение нейронной сети для создания модели двигателя внутреннего сгорания. Отмечается высокая адекватность полученной модели и возможность получения новой информации об объекте из модели, что говорит о способности нейронной сети к обобщению.

В статье [10] описано применение нейронного регулятора для управления насосной станцией. Задачей управления является минимизация суммарного энергопотребления станции. Отмечается способность нейронного регулятора учитывать трудно формализуемые ограничения и правила.

В статье [11] рассмотрена реализация адаптивного регулятора для управления двигателем постоянного тока. В рассмотренной системе нейронная сеть задает коэффициент скорости адаптации. Отмечается значительное повышение качества переходных процессов.

В статье [12] описана система управления угловой скорости двигателя постоянного тока. Нейронная сеть используется для косвенной идентификации тока двигателя. Отмечается сложность в построении обучающей выборки.

В рассмотренных источниках описано применение искусственных нейронных сетей для повышения качества управления различными динамическими объектами, но не рассматривается возможность непосредственного управления посредством нейронного регулятора.

На основании рекомендаций [7], в качестве типа нейронной сети нейроконтроллера был выбран многослойный персептрон ввиду простоты описания математической модели и наличия действенных методов для процесса обучения. В качестве структуры нейроконтроллера была выбран многомодульный нейроконтроллер. Выбор обусловлен нестационарностью объекта управления и сложностью описания математической модели для исследуемого интервала состояний. Таким образом, бесколлекторный двигатель постоянного тока рассматривается как модель черного ящика.

2. ОБЪЕКТЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

2.1. Искусственная нейронная сеть

Искусственные нейронные сети (ИНС) – направление в кибернетике, изучающее моделирование процессов в структурах, соответствующих процессам в биологических нейронных сетях, а также их применение для различных областей искусственного интеллекта и теории автоматического управления. В общем виде нейронная сеть состоит из последовательности нейронов, соединенных между собой синапсами.

Нейрон является простейшей структурно-функциональной единицей, способной хранить, обрабатывать и передавать информацию. Биологический нейрон состоит из, собственно, соматической клетки, множества коротких разветвленных отростков – дендритов, и одного длинного отростка – аксона. Посредством дендритов осуществляется прием сигналов, аксон передает информацию другим нейронам или исполнительным органам. Место контакта между двумя нейронами называется синапсом [13]. На рисунке 1 представлено изображения клетки биологического нейрона.

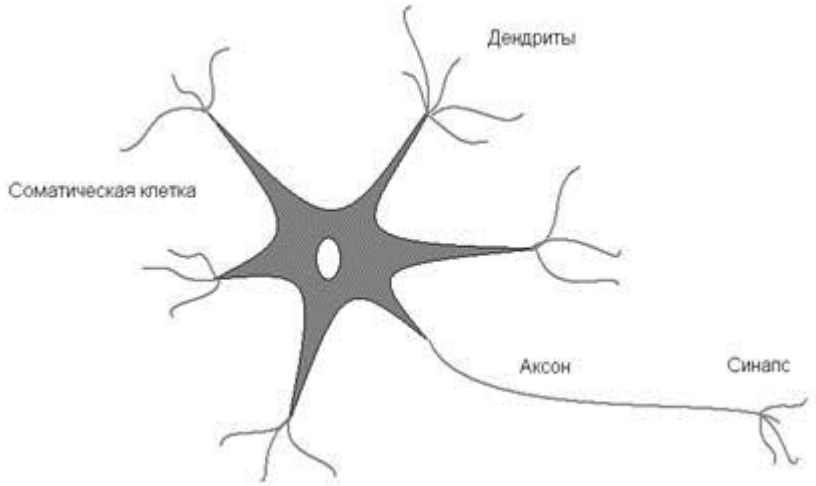


Рисунок 1 – Биологический нейрон

В искусственной модели нейрона дендриты представлены входами с весовыми коэффициентами. Произведения входных сигналов суммируются с начальным смещением нейрона и поступают на вход активационной функции,

что имитирует возбуждение нейрона. Изображение модели нейрона представлено на рисунке 2.

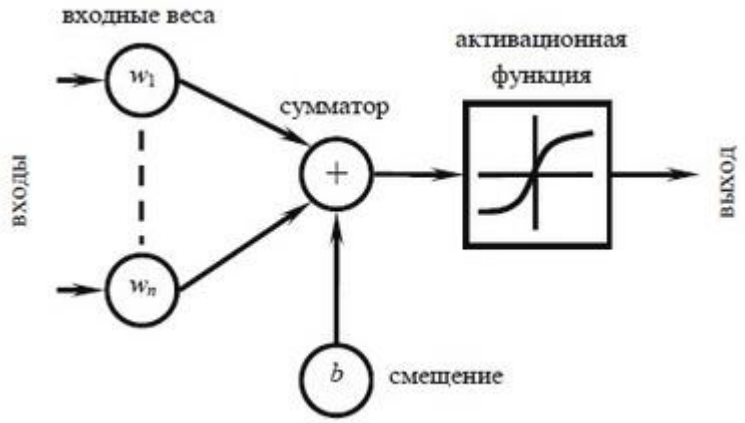


Рисунок 2 – Модель нейрона

Математическое описание такой модели может быть представлено как (1):

$$y = F\left(b + \sum_{i=0}^{n} w_i \cdot x_i\right), \quad (1)$$

где F(x) – функция активации;

b – начальное смещение нейрона;

n – количество входов;

w_i – вес i-го входа.

В качестве функции активации, как правило, применятся различные нелинейные и реже линейные функции. Различные виды функций активации представлены на рисунке 3.

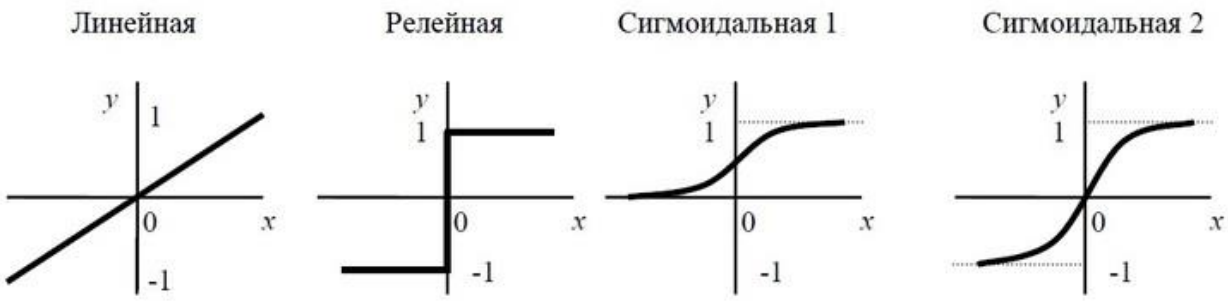


Рисунок 3 – Активационные функции

Наибольший интерес представляет сигмоидальная функция. Данная функция обладает избирательной чувствительностью к входным сигналам различной интенсивности, что соответствует реальному поведению биологического нейрона [13]. Наибольшая чувствительность к изменению входа наблюдается вблизи порога. При крайних и малых значениях входа функция не чувствительна, так как производная стремится к нулю.

Одной из первых моделей нейронных сетей явился персептрон Розенблатта. Простейший персептрон состоит из элементов трех типов. Входные S-элементы воспринимают, преобразуют и передают входную информацию на элементы следующего слоя. A-элементы, являющиеся формальными моделями нейрона, нелинейно преобразуют полученные сигналы и передают их на R-элементы, которые формируют сигналы реакции персептрона на входные воздействия. Изображение персептрона представлено на рисунке 4.

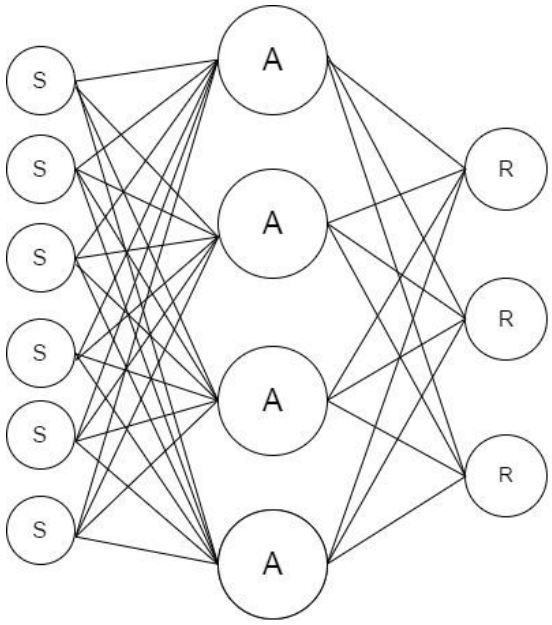


Рисунок 4 – Модель персептрона Розенблатта

Существенным недостатком такой нейронной сети является невозможность представить линейно неразделимые множества, что сводит на нет его возможность управления инерционным динамическим объектом [1].

Классическим примером линейно неразделимой функции может служить логическое исключающее ИЛИ (2).

$$Y = A \oplus B. \quad (2)$$

Графическое представление линейной неразделимости представлено на рисунке 5.

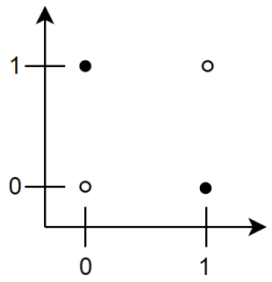


Рисунок 5 – Графическое представление функции исключающего ИЛИ

Невозможно провести линию таким образом, чтобы разделить группы множеств пар A и B при которых функцию принимает значение истина или ложь.

Для того чтобы преодолеть данную проблему, используется многослойный персептрон Розенблатта. На низших уровнях такой нейронной сети входные сигналы преобразуются таким образом, чтобы сформировались линейно разделимые множества, которые будут обработаны последующими слоями сети. Схематическое изображение многослойного персептрона Розенблатта представлено на рисунке 6.

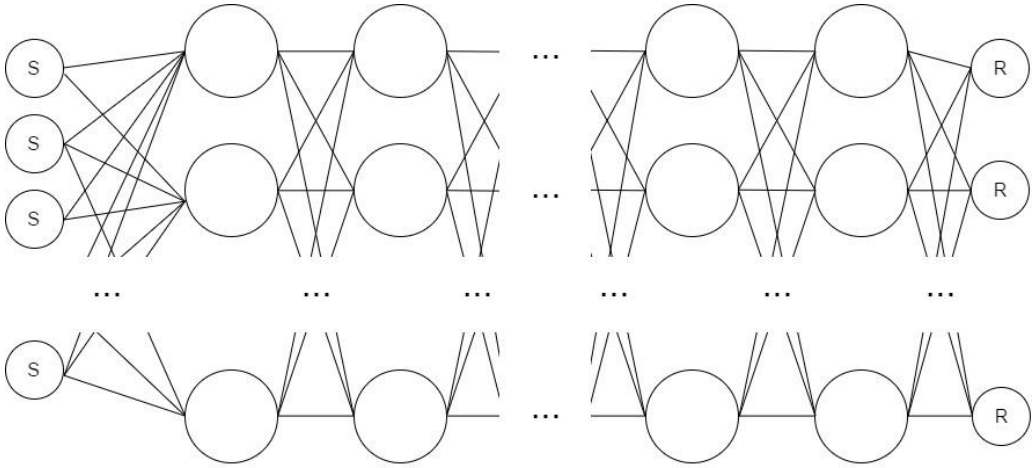


Рисунок 6 – Многослойный персептрон Розенблатта

Слой, воспринимающий входные сигналы, называется входным. Слой, формирующий реакцию, называется выходным. Находящиеся между ними слои, называются скрытыми.

Существуют множество иных видов нейронных сетей, например таких как сети Кохонена или сверточные сети [1], но они в основном применяются для задач классификации или кластеризации и не применимы для рассматриваемой задачи.

2.2. Обучение

Для того чтобы нейронная сеть смогла выполнить поставленную задачу необходимо произвести её обучение. Процесс обучения заключается в подстройке весов сети таким образом, чтобы выполнялся некоторый критерий. Можно разделить алгоритмы обучения ИНС на обучение с учителем и без учителя.

При обучении без учителя обучающая выборка состоит только из входных воздействий. Задача нейронной сети заключается в формировании однородных сигналов на схожие входные сигналы. Таким образом, выполняется задача кластеризации.

Во втором варианте предполагается наличие некоторой системы учителя, которая сообщает нейронной сети корректность ее реакций на заданные стимулы. Сети предоставляется обучающая выборка с входными воздействиями и требуемыми реакциями. На основе полученных данных нейронная сеть корректирует свои веса.

Одним из самых распространенных методов обучения многослойного персептрона является метод обратного распространения ошибки. Данный метод относится к группе методов обучения с учителем [14]. Суть метода состоит в решении оптимизационной задачи поиска глобального минимума ошибки сети в пространстве всех весов с помощью метода градиентного спуска

Рассмотрим данный метод для случая обучения персептрона с одним скрытым слоем. Изображение рассматриваемого персептрона представлено на рисунке 7.

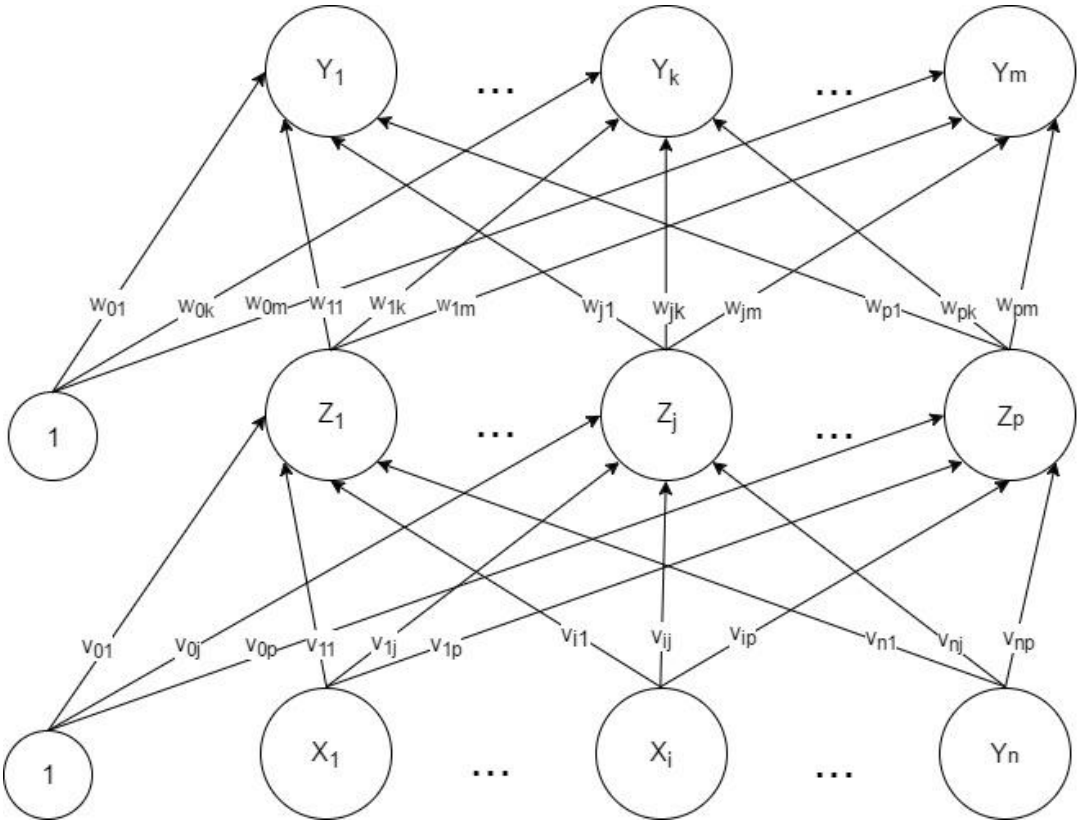


Рисунок 7 – Трехслойный персептрон Розенблатта

Введем обозначения:

x – входной вектор обучающей выборки размерностью n;

y – вектор целевых значений обучающей выборки размерностью m;

X_i – нейрон во входном слое с индексом i;

Z_j – нейрон в скрытом слое с индексом j;

Y_k – нейрон в выходном слое с индексом k;

n – количество нейронов во входном слое;

p – количество нейронов в скрытом слое;

m – количество нейронов в выходном слое;

v_{0j} – начальное смещения нейрона Z_j;

w_{0k} – начальное смещения нейрона Y_k;

v_{ij} – весовой коэффициент связи между нейронами X_i и Z_j;

w_{jk} – весовой коэффициент связи между нейронами Z_j и Y_k;

α – коэффициент скорости обучения.

В качестве функции активации выберем часто используемый бинарный сигмоид, представленный следующей формулой (3):

$$F(x) = \frac{1}{1 + e^{-x}}. \quad (3)$$

Выбор данной функции обусловлен схожестью с поведением биологического нейрона и простотой вычисления производной (4):

$$F'^{(x)} = F(x) \cdot (\, 1 - F(x)\,). \quad (4)$$

Первым шагом алгоритма является инициализация всех весовых коэффициентов некоторыми малыми случайными значениями.

Каждый нейрон входного слоя получает значения входного вектора обучающей выборки (5) и передает на нейроны скрытого слоя, в результате чего нейроны формируют выходное воздействие, в виде взвешенной суммы со смешением, и применяют функцию активации (6).

$$X_{j\,out} = x_j. \quad (5)$$

$$Z_{j\,out} = F(\, v_{0j} + \sum_{i=0}^{n} X_{i\,out} \cdot v_{ij}\,). \quad (6)$$

Каждый выходной нейрон так же формирует выходное воздействие (7):

$$Y_{k\,out} = F(\, w_{0k} + \sum_{j=0}^{p} Z_{j\,out} \cdot w_{jk}\,). \quad (7)$$

Для оценки ошибки нейронной сети для текущей пары входного и выходного векторов воспользуемся функцией среднеквадратичного отклонения (8):

$$E = \frac{1}{2}\sum_{k=1}^{m} (Y_{k\,out} - y_k)^2. \quad (8)$$

Для минимизации данной ошибки необходимо решить оптимизационную задачу поиска минимума. Возьмем производную данной

ошибки по выходу нейронной сети (9):

$$\frac{\partial E}{\partial Y_{k\ out}} = Y_{k\ out} - y_k = E_Y. \quad (9)$$

Обозначим полученное выражение как E_Y – ошибка выходного слоя нейронной сети. Определим производную ошибки ИНС по выходу скрытого слоя (10) и обозначим ее как E_Z – ошибку скрытого слоя:

$$\frac{\partial E}{\partial Z_{j\ out}} = \frac{1}{2}\sum_{k=1}^{m}(F(\ w_{0k} + \sum_{j=1}^{p} Z_{j\ out} \cdot w_{jk}\ ) - y_k)^2 =$$

$$= \sum_{k=1}^{m}\left(E_Y \cdot F'\left(\ w_{0k} + \sum_{j=1}^{p} Z_{j\ out} \cdot w_{jk}\ \right) \cdot\ w_{jk}\right) = E_Z \quad (10)$$

Определим производную ошибки ИНС по весу связей нейронов выходного слоя (11) и обозначим ее как σ_{jk} – ошибку веса связи от j-го нейрона скрытого слоя к k-му нейрону выходного слоя:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial Y_{k\ out}} \cdot \frac{\partial Y_{k\ out}}{\partial w_{jk}} = E_Y \cdot F'\left(\ w_{0k} + \sum_{j=1}^{p} Z_{j\ out} \cdot w_{jk}\ \right) \cdot\ Z_{j\ out} = \sigma_{jk} \quad (11)$$

Определим производную ошибки ИНС по весу связей нейронов скрытого слоя (12) и обозначим ее как δ_{ij} – ошибку веса связи от i-го нейрона входного слоя к j-му нейрону скрытого слоя:

$$\frac{\partial E}{v_{ij}} = \frac{\partial E}{\partial Z_{j\ out}} \cdot \frac{\partial Z_{j\ out}}{v_{ij}} = E_Z \cdot F'\left(v_{0j} + \sum_{i=1}^{n} X_{i\ out} \cdot v_{ij}\ \right) \cdot\ X_{i\ out} = \delta_{ij} \quad (12)$$

Таким образом, для минимизации целевой функции ошибки необходимо сделать градиентный шаг в сторону уменьшения градиента целевой функции для каждой его частной производной. Это можно сделать путем изменения значений весов между нейронами.

Произведем корректировку всех весов (13, 14) сети:

$$v_{ij} = v_{ij} + \alpha \cdot \delta_{ij} \quad (13)$$

$$w_{jk} = w_{jk} + \alpha \cdot \sigma_{jk} \quad (14)$$

Последовательность шагов выполняется для каждой пары элементов обучающей выборки. Одна итерация обучения на основе всех элементов обучающей выборки называется эпохой. Суммарная ошибка в эпохе может быть определена как суммарное среднеквадратичное отклонение (15):

$$E_{epoch} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{2}\sum_{k=1}^{m}(Y_{k\,out} - y_k)^2\right)}, \qquad (15)$$

где M – количество элементов обучающей выборки. Обучение персептрона продолжается пока не выполнится один из критериев останова. Например не выйдет время обучения, пройдет определенное количество эпох обучения или ошибка за эпоху не превысит минимально необходимого порогового значения.

Таким образом, во время обучения ошибка передается от выходного слоя к срытым слоям, что и отражено в названии метода. Недостатками метода является низкая скорость сходимости опасность попадания в локальный минимум ошибки, а так же опасность появления ситуации, при которой в ходе обучения веса сети могут принять очень большое значение, следовательно, градиент функции активации будет стремиться к нулю, что фактически парализует обучение [14]. Для противодействия данным недостаткам не выявлено точных методик и, как правило, при обучении используются приближенные эвристики, зависящие от задач, которые решает ИНС.

2.3. Структуры нейрорегуляторов

Применение ИНС является целесообразным, когда сложно произвести математическое описание объекта управления. В таком случае объект управления рассматривается как черный ящик, для которого известны ответные реакции на ограниченное количество входных воздействий в ограниченном диапазоне. Преимуществом нейронных сетей в таких случаях является их способность к интерполяции и экстраполяции, выявлению общих закономерностей.

При разработке систем управления на основе искусственных нейронных сетей используется их способность к обучению – воспроизведению заданных реакций, а так же к адаптации к меняющимся параметрам объекта и процесса управления в процессе функционирования. Нелинейности в составе функций активации ИНС позволяют применять ее в качестве блока линеаризации объекта управления или нелинейного регулятора.

Не рационально использовать обучение нейронной сети, последовательно подавая пары воздействий и входных реакций объекта, так как в таком случае сеть не сможет определить временную взаимосвязь входных значений. Для представления динамических свойств объекта управления может быть применена модель авторегрессии представленная модулем линии задержек Tapped Delay Line (TDL) (16):

$$TDL = \begin{pmatrix} y[n] \\ y[n-1] \\ y[n-2] \\ \dots \\ y[n-N] \end{pmatrix}. \quad (16)$$

Все методы применения ИНС для нейроуправления можно условно разделить на прямые и непрямые методы. При прямых методах нейронная сеть непосредственно влияет на объекта управления. При непрямых методах ИНС используется только как часть системы управления, выполняющая определенную функцию. Так же нейрорегуляторы можно разделить на одномодульные и многомодульные. В составе одномодульных нейрорегуляторов используется только одна ИНС, многомодульные же являются комбинацией нескольких сетей [7].

2.3.1. Подражающее нейроуправление

При подражающем нейроуправлении ИНС эмулирует поведение иного регулятора. На этапе обучения входными сигналами являются сигнал задания и сигнал обратной связи, задержанный линией задержки. В качестве целевых значений используется выход регулятора. Обучение нейрорегулятора

производят для нескольких рабочих точек объекта управления во всем диапазоне регулирования. Структурная схема обучения подражающего нейроуправления представлена на рисунке 8.

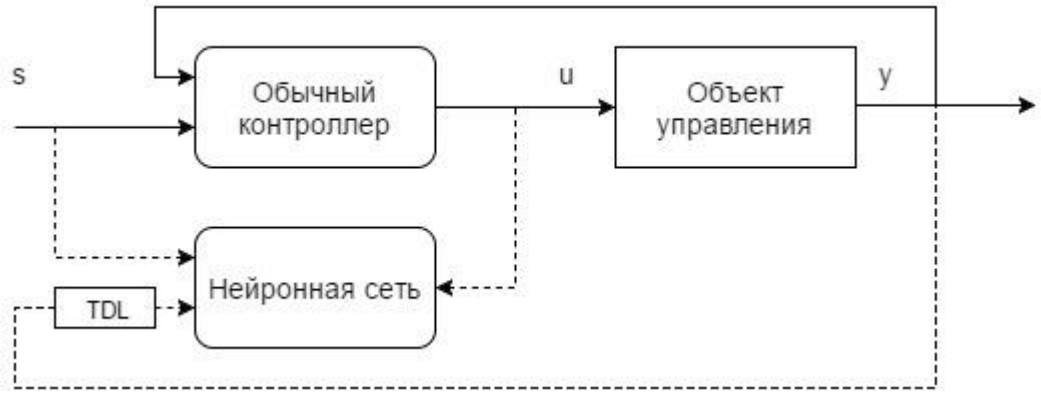


Рисунок 8 – Обучение подражающего нейроконтроллера

После обучения нейроконтроллер используется вместо исходного. При управлении объектом с переменными параметрами сеть может запомнить различные настройки регулятора и продолжить управления без переобучения, в то время как обычный регулятор нуждается в подстройке. Структурная схема управления подражающим нейроконтроллером представлена на рисунке 9.

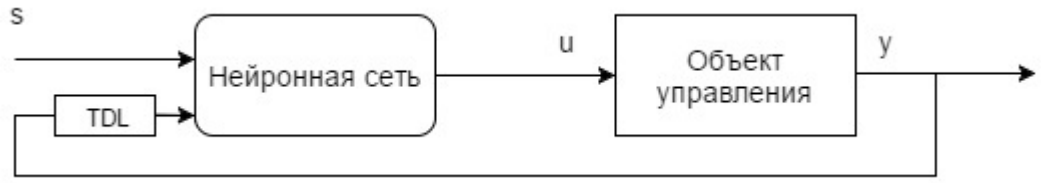


Рисунок 9 –Подражающий нейроконтроллер

Недостатком такого метода является принципиальная невозможность обеспечить качество управление лучшее, чем исходный регулятор. Метод имеет смысл для объектов с периодически меняющимися параметрами.

2.3.2. Обобщенное инверсное нейроуправление

При инверсном нейроуправлении ИНС выполняет функции инверсной модели объекта управления. Такая сеть называется инверсным нейромодулятором. Обучение сети производится в режиме оффлайн на основе накопленной статической информации об объекте управления. Для получения

такой информации на объект управления подают некоторые случайные воздействия и регистрируют ответные реакции. На основе полученных данных формируют обучающую выборку. В процессе обучения на вход искусственной нейронной сети подается выходные значения объекта управления, задержанные линией задержек. В качестве целевых значений используется вход объекта управления. Структурная схема обучения инверсного нейроэмулятора представлена на рисунке 10.

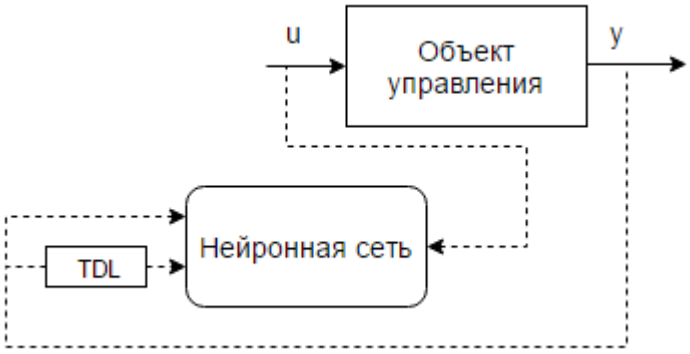


Рисунок 10 – Обучение инверсного нейроэмулятора

При управлении объектом, на инверсный нейромодулятор подается желаемое значение реакции объекта управления и выходной сигнал, задержанный во времени, в ответ на которые нейроэмулятор выдает необходимые управляющие воздействия. Структурная схема управления с использованием инверсного нейроэмулятора представлена на рисунке 11.

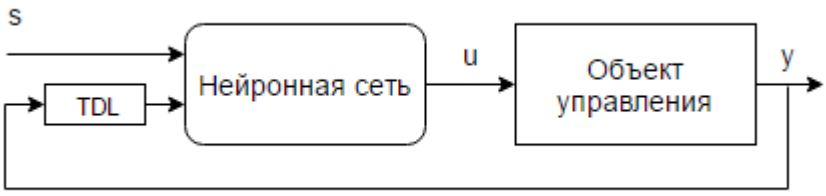


Рисунок 11 – Управление инверсным нейроэмулятором

Благодаря стабилизирующему влиянию обратной связи, достигается достаточно высокое качество управления динамическим объектом. В тоже

время возникают сложности при обучении нейроэмулятора, когда задающее воздействие меняется дискретно.

2.3.3. Метод многомодульного нейроуправления на основе пар прямых и инверсных моделей

В данном методе нейроконтроллер состоит из множества пар инверсных нейроэмуляторов и прямых моделей объекта – прямых нейроэмуляторов. Каждая пара служит для управления объектов в одной конкретной рабочей точке.

Для обучения прямого нейроэмулятора, объект управления переводится в определенное фиксированное состояние. На вход объекта подается случайное управляющее воздействие для формирования обучающей выборки. Нейроэмулятор учится повторять реакции объекта на основе задающего воздействия и задержанных реакций. Схема обучения прямого нейроэмулятора представлена на рисунке 12. Инверсный нейроэмулятор обучается схожим образом по схеме обобщенного инверсного нейроуправления.

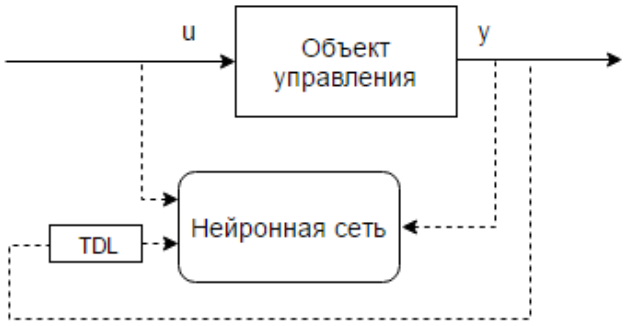


Рисунок 12 – Обучение прямого нейроэмулятора

В процессе управления прямой нейроэмулятор вычисляет текущее значение на основе входного воздействия и задержанных выходных воздействий. Предполагается, что каждая пара нейроэмуляторов обучается на своем примере динамики объекта управления и специализируется именно на нем. Поэтому, если прямой нейроэмулятор правильно предсказывает динамику объекта управления, то соответствующий ему инверсный нейроэмулятор

хорошо управляет объектом. Предполагается также, что применяющиеся для обучения пар эмуляторов траектории состояний управляемого объекта существенно отличаются между собой. На каждом этапе управления система вычисляет меру адекватности каждой пары эмуляторов текущему состоянию объекта и выбирает наиболее оптимальную пару. Схема многомодульного нейроконтроллера представлена на рисунке 13.

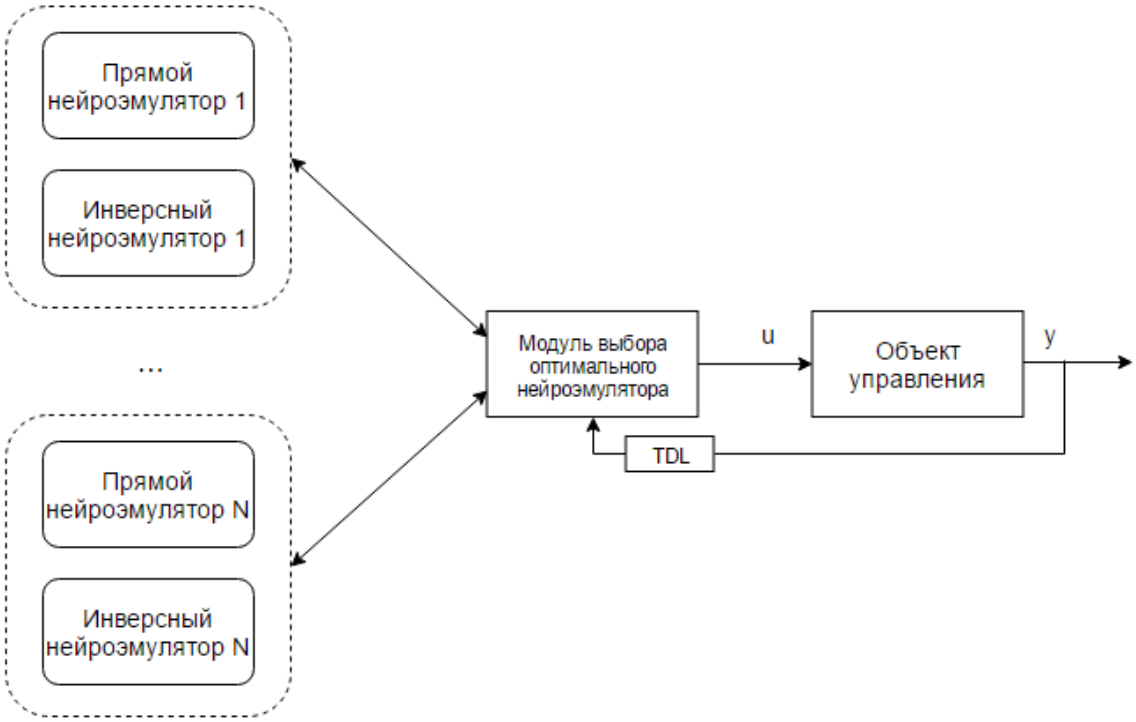


Рисунок 13 – Многомодульный нейроконтроллер

Существенным минусом систем многомодульного нейроуправления является непрозрачная процедура разделения обучающей выборки на подвыборки для обучения прямых и инверсных нейроэмуляторов разных модулей. А так же скачкообразное изменение управляющего сигнала при смене действующего модуля нейроэмуляторов.

2.4. Бесколлекторный двигатель постоянного тока

Бесколлекторные двигатели постоянного тока являются разновидностью двигателей постоянного тока. Изображение бесколлекторного двигателя представлено на рисунке 14.

Рисунок 14 – Бесколлекторный двигатель постоянного тока

Основное отличие бесколлекторного двигателя от коллекторного двигателя постоянного тока заключается в отсутствии механического коллектора – узла, который передает напряжение на якорь двигателя через щетки. Переключение обмоток якоря коллекторного двигателя осуществляется механически, что влечет за собой быстрый износ и как следствие, низкую надежность системы.

В бесколлекторном двигателе обмотки переключаются посредством внешнего устройства – контроллера. Сложность управления заключается в необходимости определения положения ротора для подключения соответствующих обмоток к источнику постоянного тока. Бесколлекторный двигатель является синхронной машиной и если частота вращения ротора и статора не совпадает, то двигатель может выпасть из синхронизма и остановиться. Проблема может быть решена путем использования датчиков положения, что в свою очередь усложняет конструкцию двигателя. Так же применяются бездатчиковые методы основанные на определении ЭДС самоиндукции обмоток [13]. Для решения данной задачи чаще всего используются специальные устройства – электронные регуляторы хода.

3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА

3.1. Аппаратная часть

3.1.1. Микроконтроллер

Для управления двигателем постоянного тока и расчета скорости вращения вала двигателя с помощью энкодера было решено выбрать аппаратную платформу Robotdyn Nano V3. Примененная аппаратная платформа является полным аналогом Arduino Nano. Выбор обусловлен максимальной простотой и достаточной функциональностью. В контексте данной работы, платформа используется как доступный USB модуль дискретного ввода-вывода. Внешний вид аппаратной платформы представлен на рисункеРисунок 15.

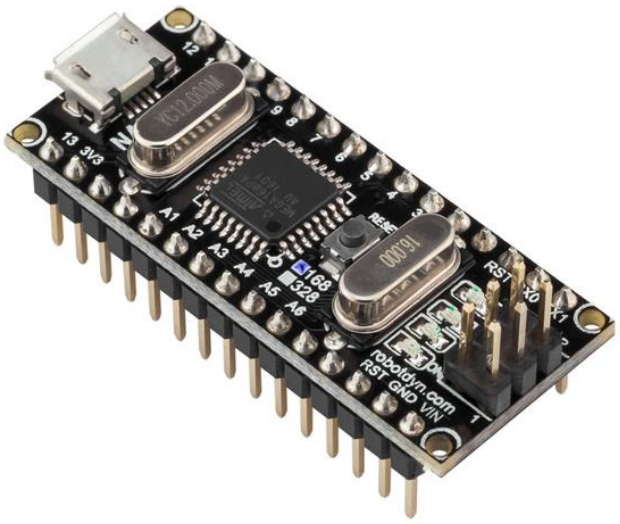


Рисунок 15 – Внешний вид платформы Robotdyn NANO V3

Характеристики аппаратной платформы представлены ниже:

- микроконтроллер – ATmega328;
- рабочее напряжение – 5В;
- рекомендуемое входное напряжение – 7-12 В;
- 14 цифровых входов/выходов (D0 – D14);
- 6 портов поддерживают ШИМ (D3, D5, D6, D9, D10, D11);
- 8 аналоговых входов A0 – A7;

- выходной ток 40 мА;
- 32 Кб flash-памяти;
- 2 Кб оперативной памяти;
- частота встроенного тактового генератора 16 МГц [16].

В платформу встроен загрузчик, который позволяет программировать микроконтроллер без использования сторонних программаторов. Программирование происходит в среде разработки Arduino, которая состоит из встроенного текстового редактора, окна вывода текста, области сообщений, панели инструментов, содержащей кнопки часто используемых команд и ряд меню. Язык программирования основан на языках C и C++. Связь с компьютером осуществляется посредством USB через виртуальный COM порт.

3.1.2. Бесколлекторный двигатель постоянного тока

Бесколлекторный двигатель постоянного тока KV (БДПТ) характеризуется количеством оборотов двигателя на один вольт без нагрузки на валу. Умножив этот показатель на максимальное напряжение, можно получить максимальное число оборотов двигателя без нагрузки на валу. Внешний вид БДПТ представлен на рисунке 16.

Рисунок 16 – Внешний вид бесколлекторного двигателя

Основные характеристики бесколлекторного двигателя постоянного тока сведены в таблицу 1.

Таблица 1 – Характеристики БДПТ [17]

Внешний диаметр статора	22 мм
Внутренний диаметр статора	16 мм
Количество обмоток	12
Количество полюсов ротора	14
Величина kv	880
Ток холостого хода	0.5A
Сопротивление мотора	0.117 Ом
Максимальный непрерывный ток	20A/30S
Максимальная непрерывная мощность	320 ват
Вес	72 г
Диаметр якоря	27.7мм
Диаметр вала	3.175мм
Длина вала	34мм
Общая длина вала	36.5мм

3.1.3. Электронный регулятор хода

Электронный регулятор скорости позволяет управлять двигателем посредством переключения обмоток в моменты времени, определенные положением ротора. Идентификация положения ротора производится на основе обратной ЭДС, формируемой в обмотках статора магнитами ротора. Внешний вид электронного микроконтроллера скорости представлен на рисунке 17.

Рисунок 17 – Внешний вид электронного регулятора

Характеристики электронного регулятора приведены в таблице 2.

Таблица 2 - Характеристики электронного регулятора [18]

Тип	SkyWalker – 40A
Выходной ток	40A (в пиках до 55A в течение 10 сек)
Входное напряжение	2-4S Lipo или 5 - 12 ячеек NiMH
Вид BEC (Battery eliminator circuit)	Линейный регулятор
Выход BCE (импульсный)	5B/3A
Частота обновления	50 - 432 Гц
Максимальная скорость	70000 об/мин для 6 полюсных бесщёточных двигателей,
Вес	39 г
Размер	68x25x8 мм

3.1.4. Датчик

Для отслеживания оборотов на валу двигателя был использован простейший инкрементный энкодер. Энкодер состоит из пластикового диска с 10 прорезями, порогового датчика освещенности и светодиода. Изображение датчика приведено на рисунке 18.

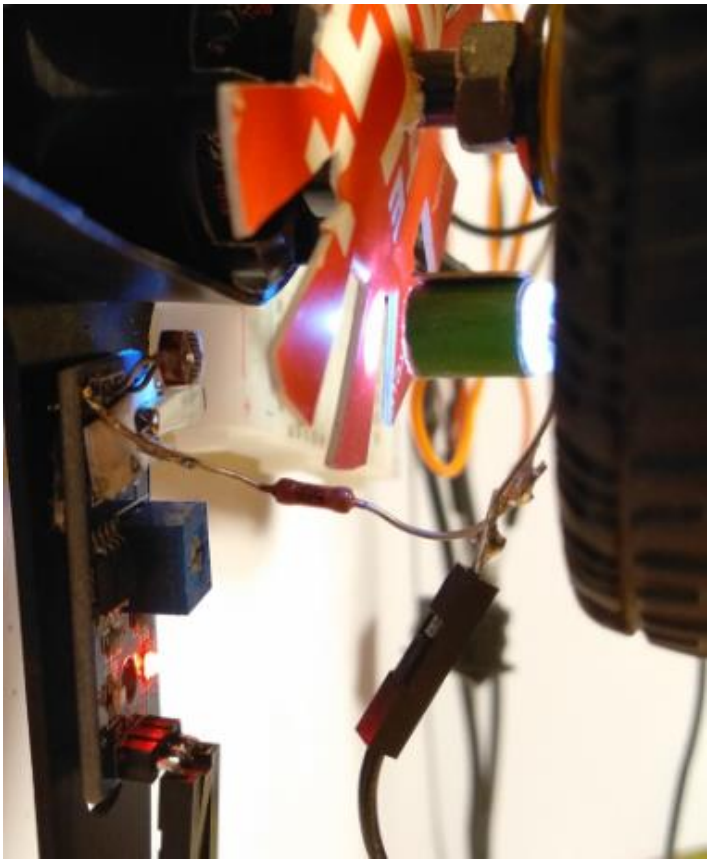


Рисунок 18 – Внешний вид инкрементного энкодера

Диск устанавливается на вал двигателя. Край диска помещается между фоторезистором датчика и светодиодом. По количество положительных и отрицательных фронтов за единицу времени можно отслеживать скорость вращения вала двигателя.

3.1.5. Сборка устройства

Таким образом, для аппаратной реализации стенда было решено использовать следующие компоненты:

- аппаратная платформа Robodyn Nano V3;
- бесколлекторный двигатель постоянного тока;
- ESC SkyWalker – 40A;
- датчик освещенности;
- светодиод;
- тумблер;
- два нагрузочных резистора номиналом 10 кОм;
- винт с барашком и подголовником квадратного сечения;
- пружина;
- резиновая лента.

В итоге был собран испытательный стенд, для управления бесколлекторным двигателем постоянного тока. Структурная схема стенда представлена на рисунке 19.

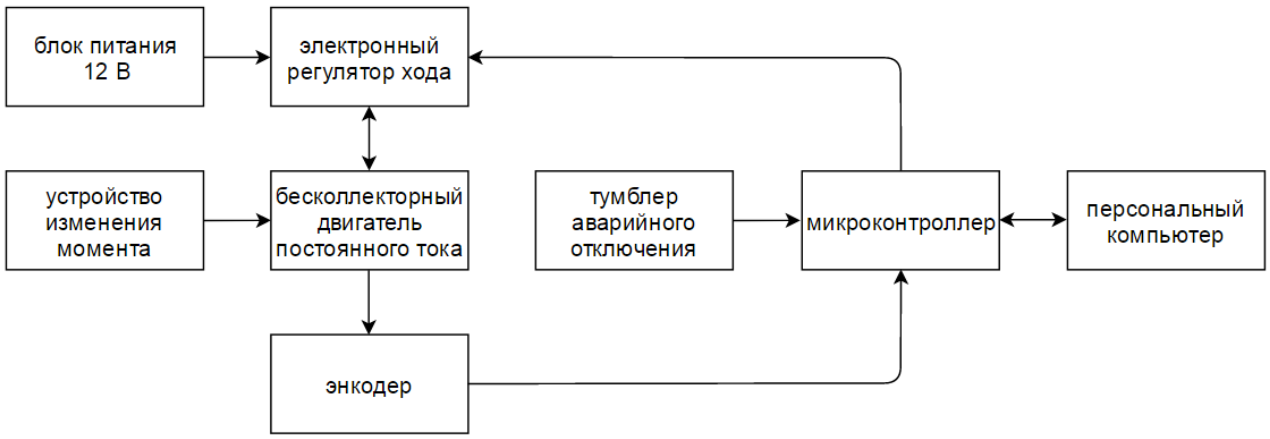


Рисунок 19 – Структурная схема испытательного стенда

Персональный компьютер содержит в себе основную программу для работы с нейронными сетями. В процессе работы ПК формирует задание и пересылает его на микроконтроллер посредством интерфейса USB-UART. Микроконтроллер пересылает задание на электронный регулятор хода, подключенный к блоку питания 12 В. Регулятор формирует импульсы на обмотках бесколлекторного двигателя постоянного тока и приводит его в движение. Двигатель начинает вращаться, испытывает сопротивление устройства изменения момента. Энкодер считывает обороты двигателя и передает значения на микроконтроллер. Микроконтроллер производит первичную обработку данных и пересылает значение на персональный компьютер. В случае нештатной работы стенда возможно произвести аварийное отключение двигателя посредством тумблера. Принципиальная электрическая схема предоставлена на рисунке 20.

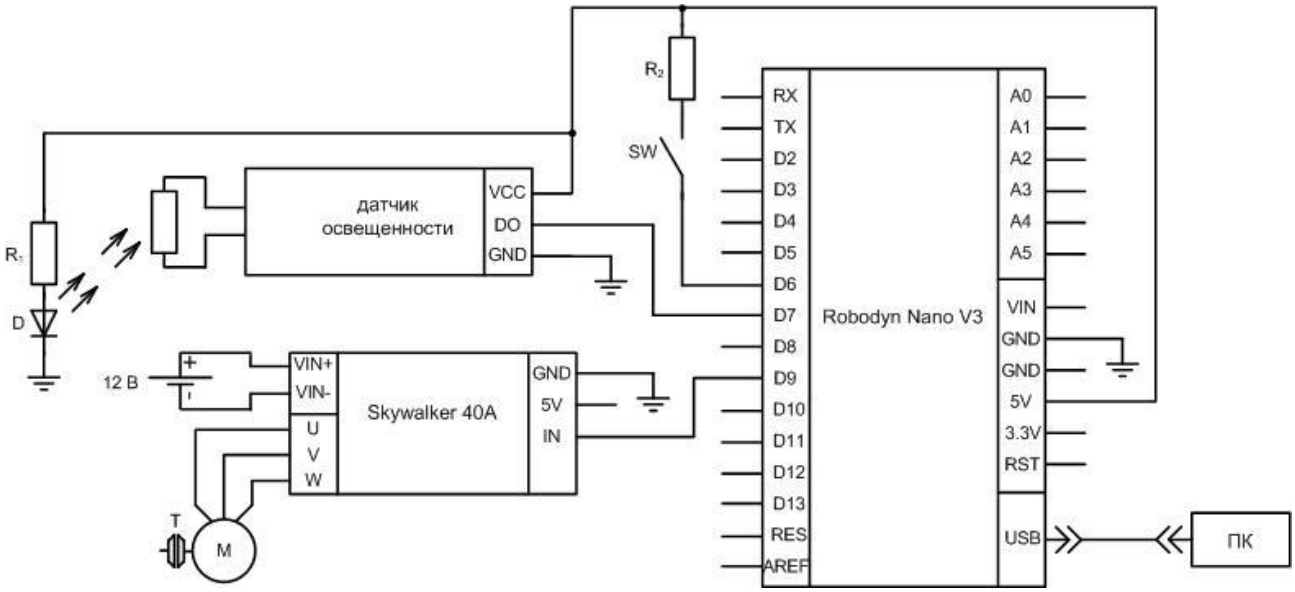


Рисунок 20 – Принципиальная электрическая схема

На схеме буквой D обозначен светодиод, R_1 и R_2 – нагрузочные резисторы, M – бесколлекторный двигатель постоянного тока, T – устройства для динамического изменения нагрузки на валу

По разработанным структурной и принципиальной электрической схемам был собран учебный стенд управления бесколлекторным двигателем. Изображение стенда приведено на рисунке 21.

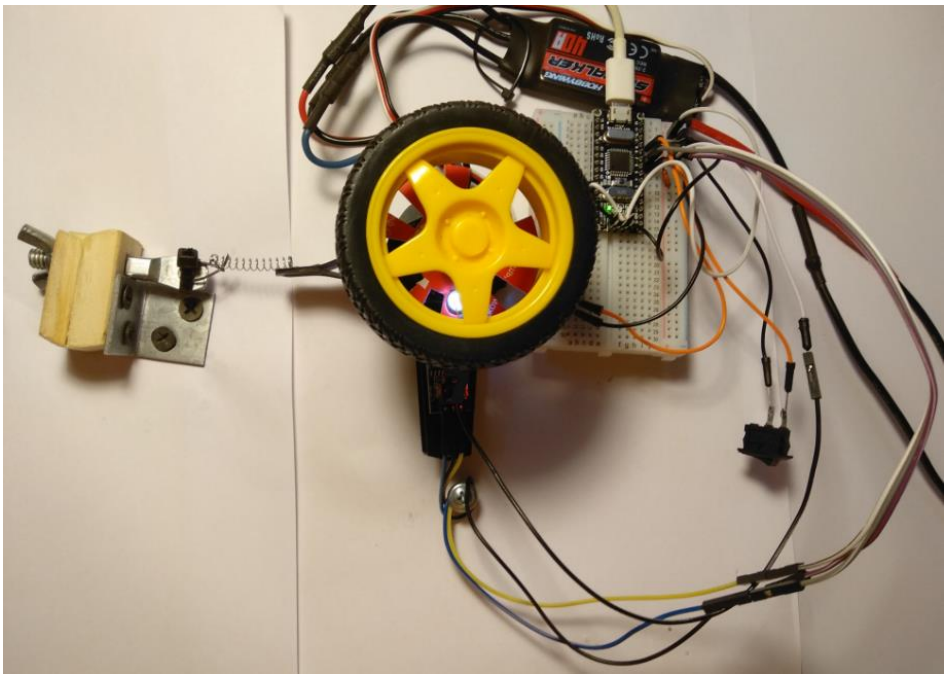


Рисунок 21 – Внешний вид испытательного стенда

Бесколлекторный двигатель постоянного тока жестко закреплен на массивном основании, на вал двигателя прикреплен пластиковый диск энкодера. На шасси двигателя прикреплен датчик освещенности со светодиодом. Выводы обмоток двигателя присоединены к электронному регулятору хода. Электрическая схема собрана с помощью макетной платы.

Для обеспечения возможности динамической смены нагрузки на валу двигателя было изготовлено специальное устройство. Изображение устройства предоставлено на рисунке 22.

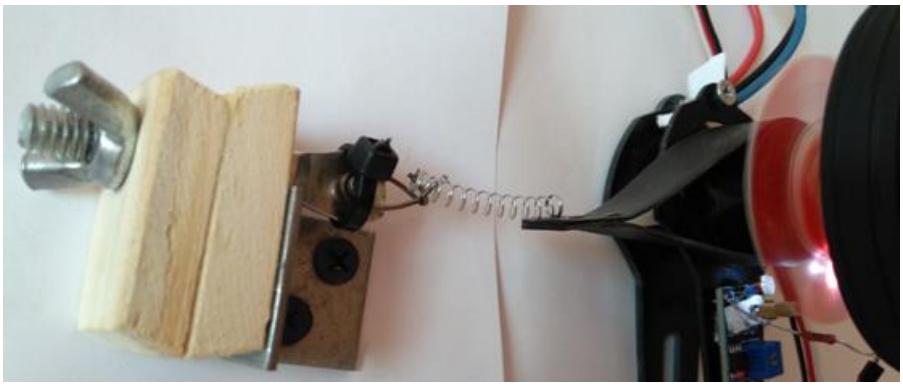


Рисунок 22 – Устройства динамического задания нагрузки

Для динамического изменения нагрузки была применена резиновая лента, которая меняла свое натяжение в зависимости от натяжения пружины, которая в свою очередь натягивалась посредством жестко зафиксированной системы из болта с прямоугольным подголовником, упора и барашка. Полученное устройство позволило плавно менять нагрузку на двигатель в широком диапазоне и фиксировать определенные состояния нагрузки посредством вращения барашка на определенный угол.

3.2. Программная часть

3.2.1. Микроконтроллер

Аппаратная платформа Robotdyn NANO V3 используется в качестве дискретного модуля ввода вывода. Программа реализована на языке C/C++ и выполняет следующие задачи:

- счет количества передних и задних фронтов с датчика освещенности за такт измерения и преобразование к оборотам в секунду;
- первичная фильтрация данных с датчика с целью минимизации ошибки округления;
- формирование текущего задания для контролера бесколлекторного двигателя постоянного тока;
- асинхронный прием и передача данных с персонального компьютера посредством интерфейса UART;
- аварийный останов программы посредством переключателя.

Полный исходный код программы представлен в приложении Б.

3.2.2. Персональный компьютер

На персональном компьютере под управлении операционной системы Windows выполняется основная часть программы. Разработка программы проводилась в Microsoft Visual Studio 2017.

Программа реализована в виде приложения WinForms. Приложение разделено на три параллельно выполняющихся потока для максимальной

производительности и исключения влияния работы второстепенных циклов на время выполнения основного цикла. На главной форме приложения в виде графика выводятся значения с энкодера в реальном времени. Ниже расположен график текущего состояния объекта. В верхней части формы расположены окно динамического отображения текущих ошибок обучения в эпохе и окно результатов проверки полученного нейроэмулятора. Так же на форме реализован ввод уставки для ДПТ и отображение различной диагностической информации, такой как время цикла значения СКО и эпохи при обучении, текущее состоянии и составляющие регулятора.

Основное окно программы представлено на рисунке 23.

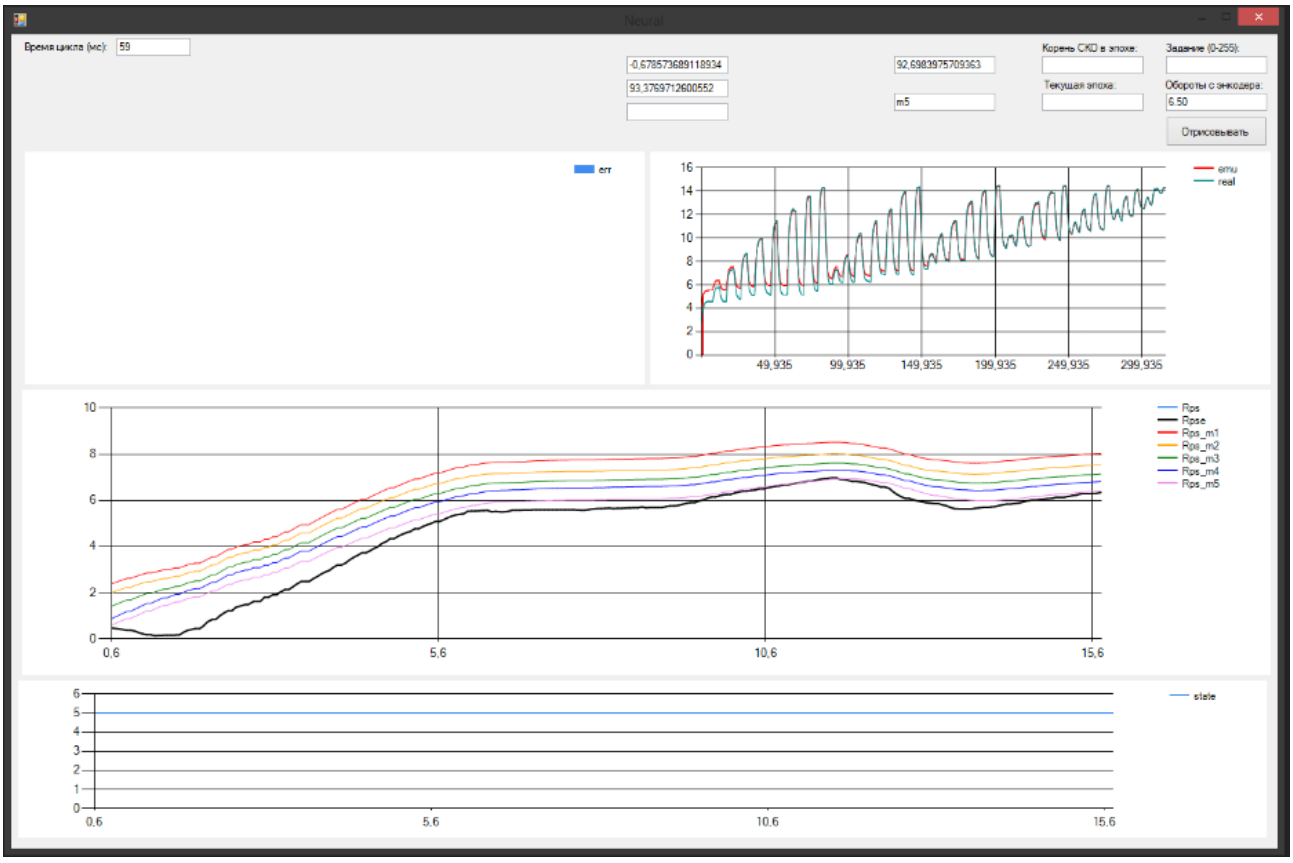


Рисунок 23 – Главное окно приложения

В приложении реализована асинхронная передача данных с аппаратной платформой Robodyn Nano. Алгоритмы передачи данных содержатся в методах PortRead и PortWrite. Инициализация последовательного порта содержится в методе InitializePort. Циклический опрос порта содержится в методе ComRead.

Для вывода графиков был использован элемент WinForms – Chart. Инициализация графиков содержится в методе InitizlizePlot. Обновление значений графиков содержится в методе Loop который вызывается таймером раз в 50 мс. Инициализация таймера содержится в методе InitializeLoopTimer.

Ввод уставки реализован по событию нажатия клавиши Enter в соответствующем текстбоксе. Уставка напрямую отправляется в микроконтроллер или поступает на вход контроллера в линию задержек в зависимости от текущего выбранного метода управления.

Для работы инверсного нейромодулятора был программно реализован многослойный персептрон Розенблатта. Нейросеть представлена четырьмя классами. Класс Neuron хранит в себе поля и конструктор инициализации базового элемента нейронной сети – нейрона. Класс NetInput хранит конструктор инициализации входного слоя, количество входов в котором определяется входной выборкой. Класс NetLayer хранит конструктор инициализации скрытых или выходных слоев. В классе реализованы методы для обучения методом обратного распространения ошибки для разных типов слоев. Класс Network хранит конструктор для инициализации трехслойного персептрона и методы для обучения нейронной сети. Класс легко может быть модифицирован для создания персептрона с любым количеством слоев и нейронов. Каждый из описанных классов имеет модификатор Serializable, что позволяет записывать экземпляры классов в файл. Методы сериализации описаны в классе NetSerialize. Алгоритмы чтения и записи нейронных сетей в файлы содержатся в методах NetToFile и NetFromFile.

Метод Timer реализует измерения времени между тактами для отрисовки графика. Метод DbTimer аналогичен предыдущему, но используется для отладки измерения цикла управления в другом потоке.

Для формирования обучающей выборки прямого нейроэмулятора реализован метод TrainsetGen, который формирует различные управляющие воздействия, дожидаясь, пока переходный процесс стабилизируется. Алгоритмы стабилизации содержатся во встроенном методе

WaitForStabilization. Полученные значения записываются в файл. Результат работы метода для ограниченной области управляющих воздействий представлен на рисунке 24.

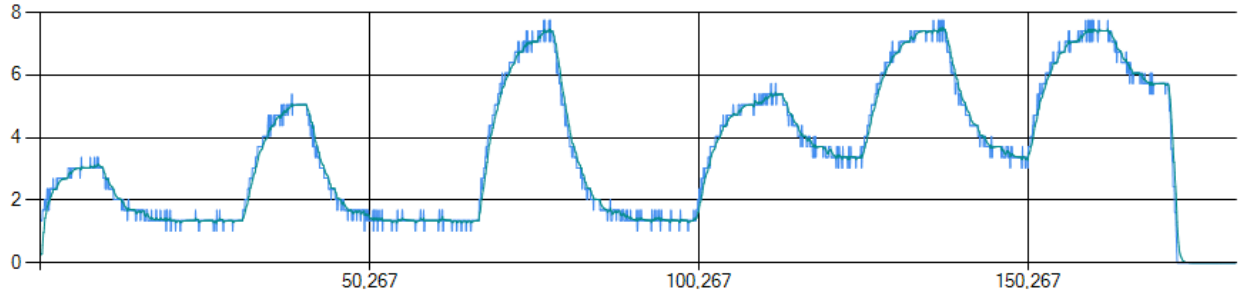


Рисунок 24 – Пример работы генератора обучающей выборки

Методы ToEmulatorTransform и ToInvertedTransform преобразует обучающую выборку из файла к виду, пригодному для нейронной сети вводя линию задержек. Пример нормализованной выборки с глубиной линейной задержки 5 для прямого нейроэмулятора представлен на рисунке 25.

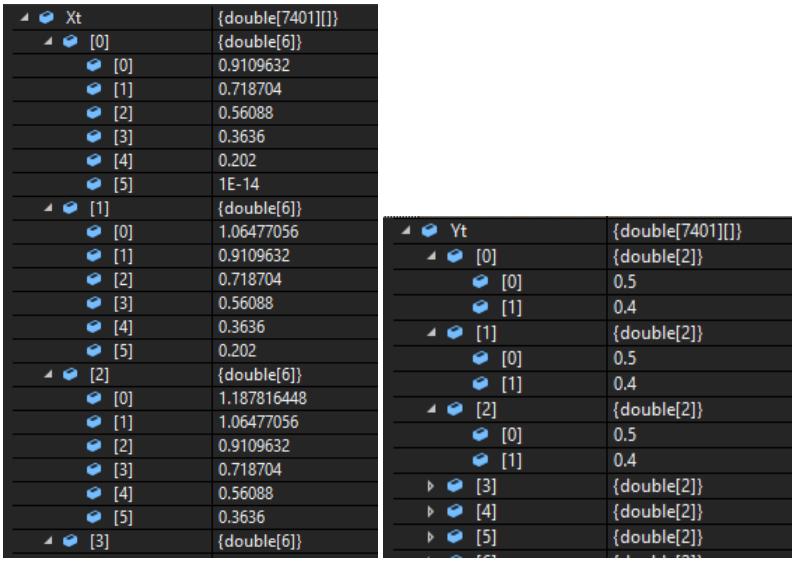


Рисунок 25 – Нормализованная обучающая выборка

Обучение или загрузка из файлов происходит в методе Net. Инициализация компонентов происходит при запуске формы.

Для уменьшения скачков значения с энкодера при ошибках округления был применен экспоненциальный фильтр (17):

$$u_e(N) = \alpha \cdot \mathrm{u(N)} + (1-\alpha) \cdot u_e(N-1), \qquad (17)$$

где α – постоянная экспоненциального фильтра;

u(N) – исходное значение на текущем такте.

График результатов фильтрации представлен на рисунке 26. Серым цветом обозначено исходно значение, черным – отфильтрованное экспоненциальным фильтром с коэффициентом α равным 0.8.

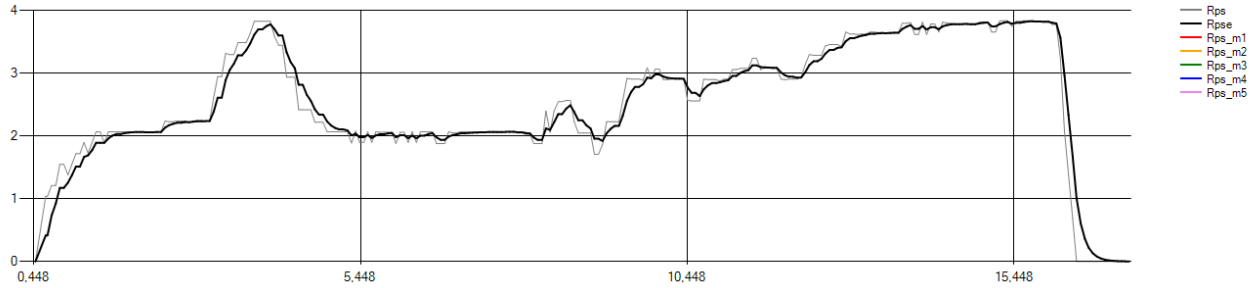


Рисунок 26 – Результат работы экспоненциального фильтра

Режимы работы программы выбираются перед запуском посредством присвоения определенных логических переменных-флагов. Список таких переменных с коментариями приведен на рисунке 27.

```
// конфигурация запуска программы
public bool Emulator = false; // флаг обучения нейроэмулятора
public bool Inverse = false; // флаг обучения инверсного нейроэмулятора
public bool DirectGen = false; // флаг генерации обучающей выборки для нейроэмулятора
public bool InverseGen = false; // флаг генерации обучающей выборки для инверсного нейроэмулятора
public bool NetControl = false; // флаг управления нейроконтроллером
public bool PIDControl = false; // флаг управления ПИД-регулятором
public bool CombinedControl = true; // флаг управления комбинированным нейроконтроллером
//
```

Рисунок 27 – Флаги выбора режимы работы программы

Таким образом, написана программа, реализующая нейронную сеть в виде совокупности классов. Реализована возможность записи и чтения обученных нейронных сетей из файла. Реализованная возможность чтения обучающей выборки из файла позволяет применять иные методы для ее формирования. Разработанная программа является удобным и эффективным инструментом для экспериментов с нейронными сетями выбранной архитектуры.

4. ИССЛЕДОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ

4.1. Влияние параметров обучения на качество искусственной нейронной сети

Адекватность модели, представленной нейронной сетью, напрямую зависит от процесса обучения. Можно выделить ряд параметров, которые влияют на качество обучения:

- коэффициент скорости обучения (LearningRate);
- количество нейронов в скрытом слое (HiddenNeurons);
- длина линии задержки входных сигналов (DelayLineDeep);
- количество пройденных эпох обучения (EpochTreshold).

В качестве критерия оценки качества обучения можно использовать итоговое суммарное среднеквадратичное отклонение за эпоху, которое определяется по формуле (15). Представленные в программе параметры обучения приведены на рисунке 28.

```
public double EpochTreshold = 8000; // порог ошибки при обучении
public double LearningRate =0.01 ; // коэффициент скорости обучения
public int DelayLineDeep = 5; // глубина линии задержки
public int HiddenNeurons = 5; // колличество нейронов в скрытом слое
```

Рисунок 28 – Параметры обучения

В качестве обучающей выборки выберем набор для обучения нейроэмулятора двигателя постоянного тока под нагрузкой. Размер обучающей выборки 1100 значений.

4.1.1. Влияние коэффициента скорости обучения

Исследуем влияние коэффициента скорости обучения на итоговое среднеквадратичное отклонение. Зафиксируем количество пройденных эпох обучения на значении 8000, количество нейронов на значении 5 и длину линии задержки на значении 5.

Результаты исследований приведены в таблице 3.

Таблица 3 – Влияние коэффициента скорости обучения на результирующую ошибку

Коэффициент скорости обучения	0.01	0.1	1	10	100
Итоговое СКО	0.005201	0.000921	0.000601	0.000404	0.000482

Графическое изображение зависимости представлено на рисунке 29.

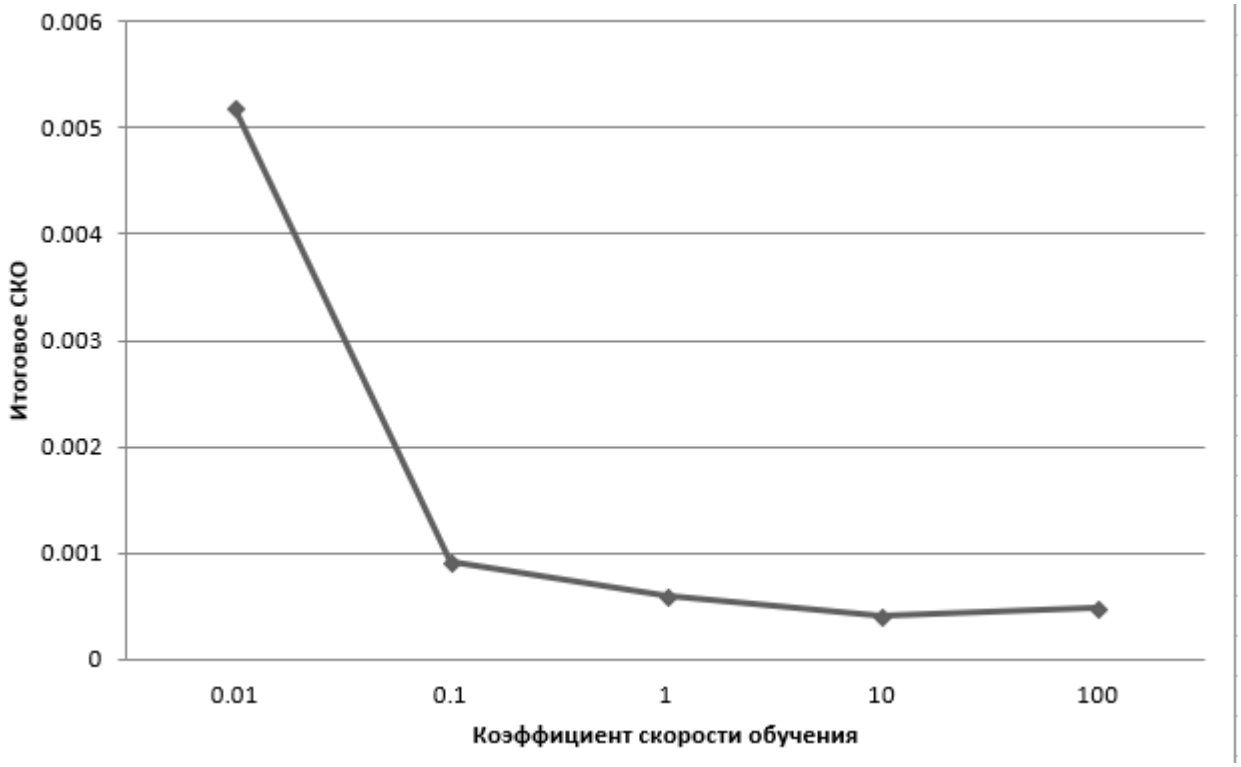


Рисунок 29 – Влияние скорости обучения на итоговое СКО

При малых значениях коэффициента скорости обучения наблюдается медленная сходимость и появляется риск попадание в локальный минимум, в тоже время повышается точность попадания в точку экстремума. При больших значениях невозможно добиться малой ошибки, так как каждый шаг проскакиваем мимо экстремума.

4.1.2. Влияние количества нейронов в скрытом слое

Исследуем влияние количества нейронов в скрытом слое на итоговое среднеквадратичное отклонение. Зафиксируем количество пройденных эпох

обучения на значении 8000, коэффициент скорости обучения на значении 1 и длину линии задержки на значении 5. Результаты исследований приведены в таблице 4

Таблица 4 – Влияние количества нейронов в скрытом слое на результирующую ошибку

Количество нейронов в скрытом слое	1	2	5	10	50
Итоговое СКО	0.000631	0.000609	0.000601	0.000597	0.000566

Графическое изображение зависимости представлено на рисунке 30.

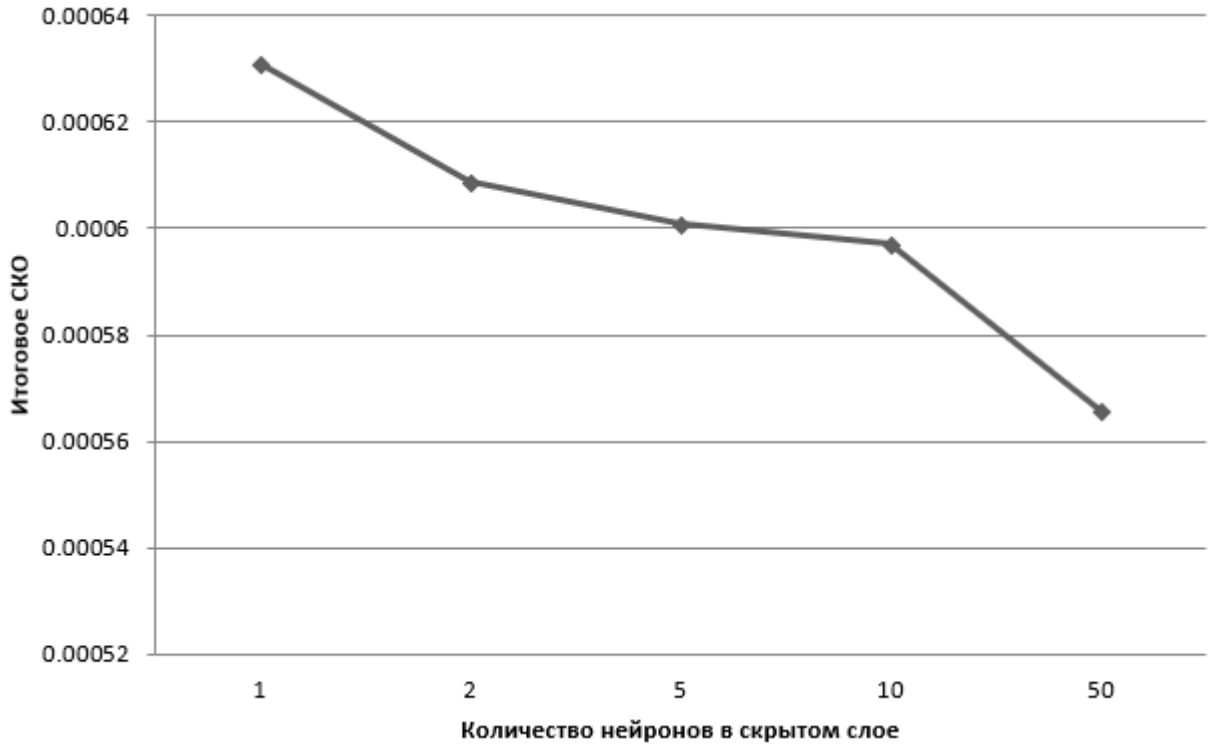


Рисунок 30 – Влияние количества скрытых нейронов на итоговое СКО

Большее количество нейронов в скрытом слое позволяет более точно описывать обучающую выборку ценой вычислительных мощностей. В тоже время появляется опасность переобучения, когда нейронная сеть воспроизводит шумы и искажения в обучающей выборке и не способна адекватно представить реальный физический объект.

4.1.3. Влияние длины задержки входных сигналов

Исследуем влияние задержки входного сигнала на итоговое среднеквадратичное отклонение. Зафиксируем количество пройденных эпох обучения на значении 8000, коэффициент скорости обучения на значении 1 и количество нейронов в скрытом слое на значении 5. Результаты исследований приведены в таблице 5

Таблица 5 – Влияние длины линии задержки на результирующую ошибку

Длина линии задержек	1	2	5	10	50
Итоговое СКО	0.000271	0.000351	0.000601	0.000695	0.000617

Графическое изображение зависимости представлено на рисунке 31.

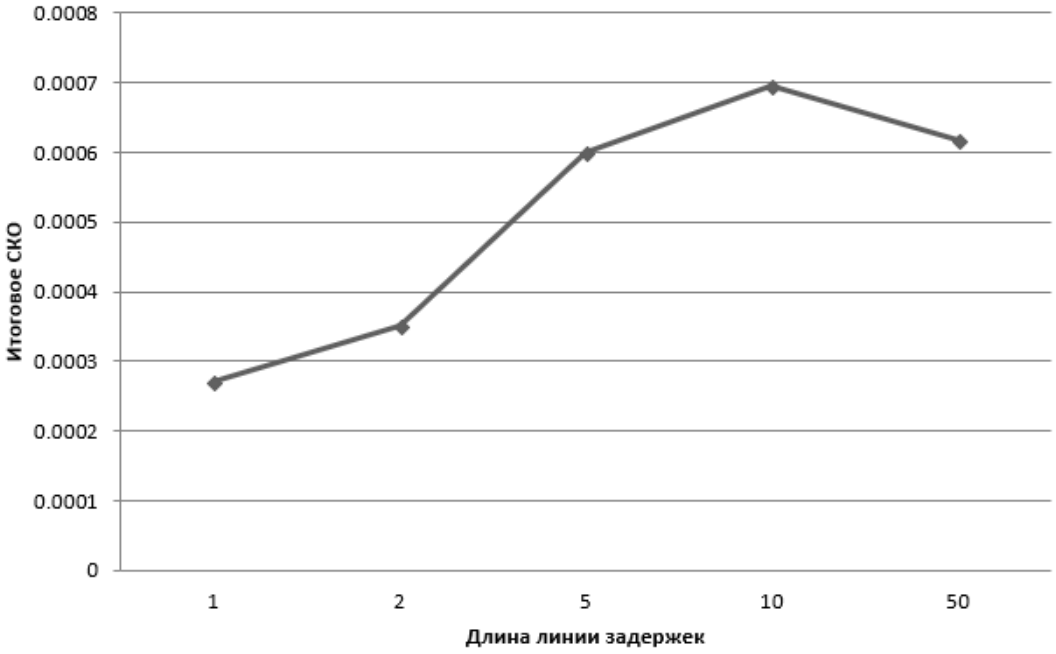


Рисунок 31 – Влияние длины линии задержки на итоговое СКО

Большая длина линии задержки увеличивает количество входных нейронов и, следственно, вычислительную нагрузку. В тоже время большая длина линии задержки лучше описывает динамические свойства объекта и позволяет избежать противоречий при обучении, когда объект может менять свое поведение в зависимости от прошлых воздействий.

4.1.4. Влияние количества пройденных эпох обучения

Исследуем влияние количества пройденных эпох обучения на итоговое среднеквадратичное отклонение. Зафиксируем коэффициент скорости обучения на значении 1, количество нейронов в скрытом слое на значении 5 и длину линии задержки на значении 5. Результаты исследований приведены в таблице 6

Таблица 6 – Влияние длины линии задержки на результирующую ошибку

Пройденные эпохи обучения	1	1000	8000	16000	50000
Итоговое СКО	0.0311799	0.000872	0.000601	0.000561	0.000486

Графическое изображение зависимости представлено на рисунке 32.

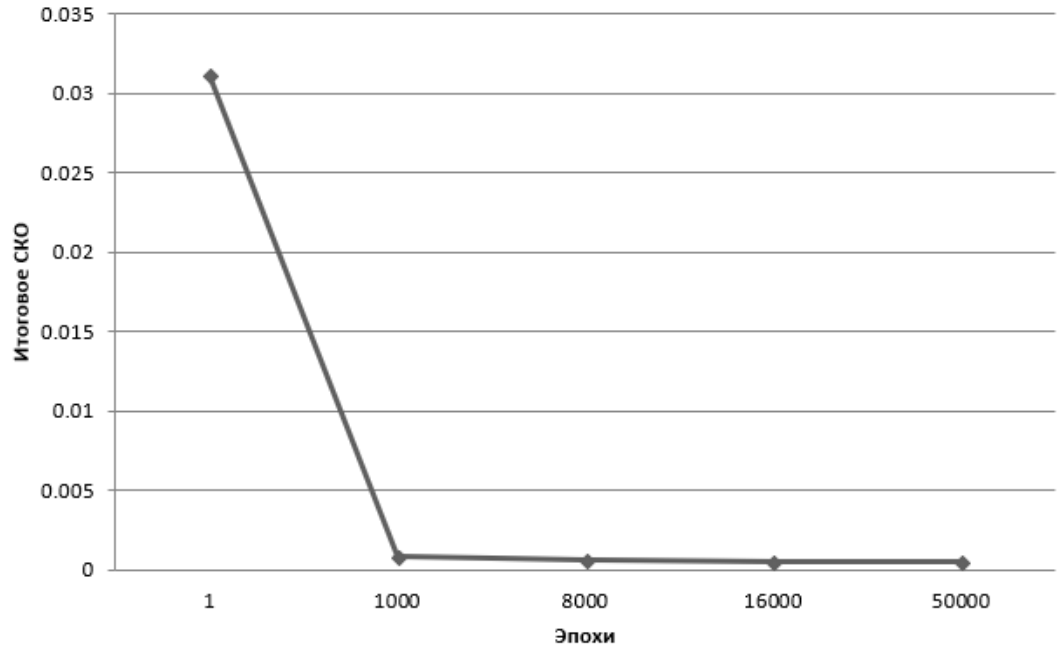


Рисунок 32 – Влияние количества пройденных эпох на итоговое СКО

Итоговая ошибка логарифмически зависит от количества пройденных эпох, вследствие чего не рационально использовать большое количество эпох, так как экспоненциально возрастает время вычисления и возникает риск переобучения сети.

4.2. Синтез ПИД-регулятора

Для сравнения полученных результатов с типовым регулятором был синтезирован дискретный ПИД-регулятор. Разностная формула ПИД-регулятора может быть представлена как (18):

$$u(N) = K_p \cdot \mathrm{e(N)} + T \cdot K_i \cdot \sum_{i=1}^{n} e(N-i) + \frac{1}{T} \cdot K_d \cdot \big(e(N) - e(N-1)\big), \qquad (18)$$

где K_p – пропорциональный коэффициент;

K_i – интегральный коэффициент;

K_i – дифференциальный коэффициент;

T – период дискретизации;

e(N) – ошибка на N-ом такте.

Объект управления был зафиксирован в одном состоянии для сбора данных для идентификации. Полученные данные были введены в ППП MatLAB. График задающих воздействий и ответных реакций представлен на рисунке 33.

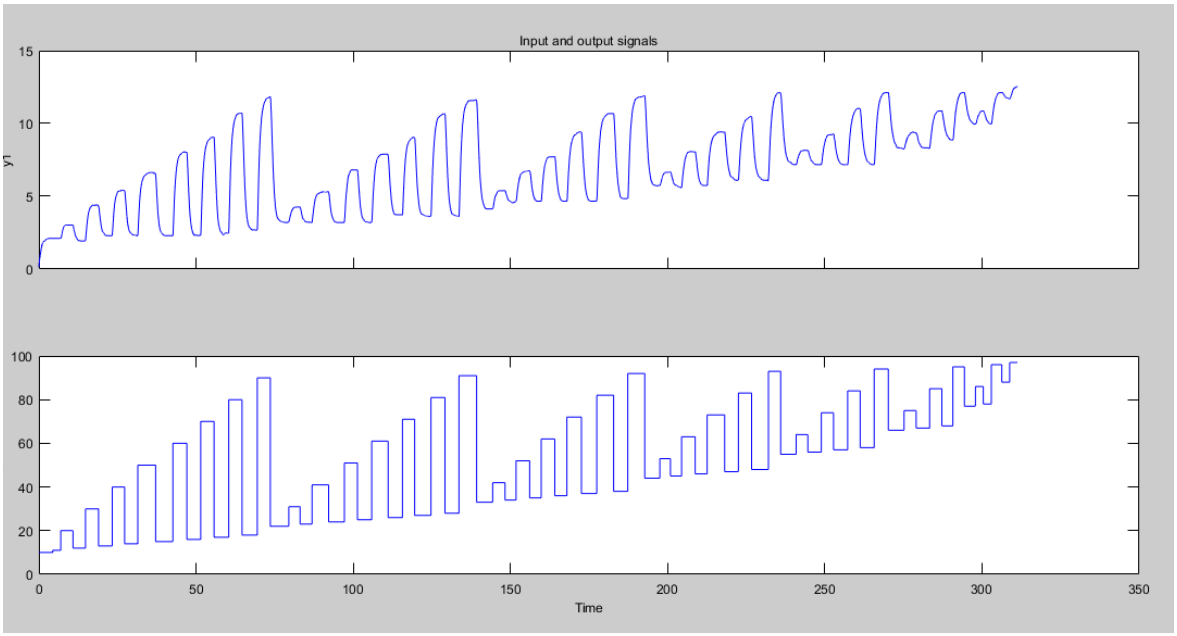


Рисунок 33 – Данные для идентификации

С помощью встроенного приложения System Identification была произведена идентификация объекта. В качестве передаточных функций были

рассмотрены функции разных порядков. Результаты идентификации представлены на рисунке 34.

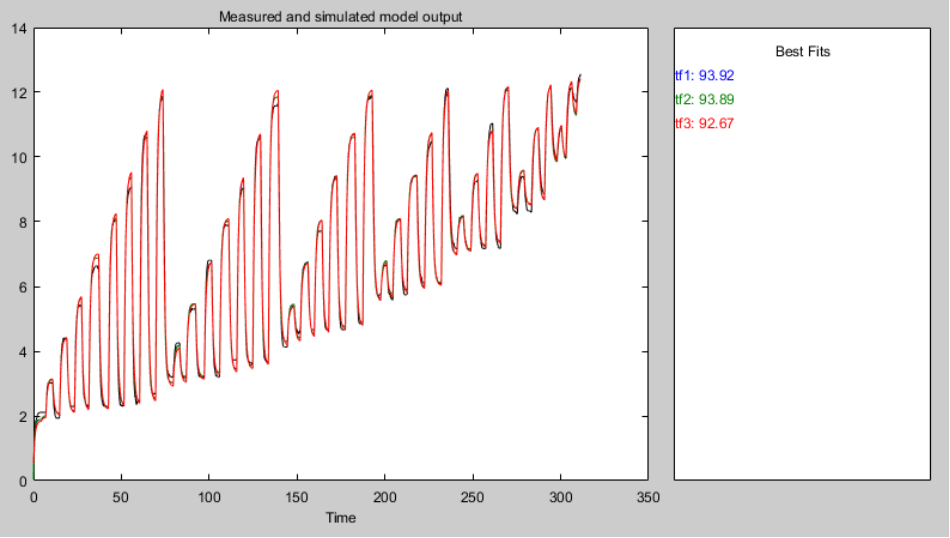


Рисунок 34 – Результаты идентификации

Полученная передаточная функция для объекта в заданном фиксированном состоянии представлена на рисунке 35.

```
From input "u1" to output "y1":
   0.153 s + 0.00206
-----------------------
s^2 + 1.139 s + 0.01635
```

Рисунок 35 – Передаточная функция объекта для заданного состояния

Таким образом, передаточная функция для данного состояния может быть записана как (19):

$$W(s) = \frac{0.153 \cdot s + 0.0021}{s^2 + 1.139 + 0.01635} \quad (19)$$

На основе полученной передаточной функции была составлена модель системы управления с дискретным ПИД-регулятором в ППП Simulink. Структурная схема модели представлена на рисунке 36.

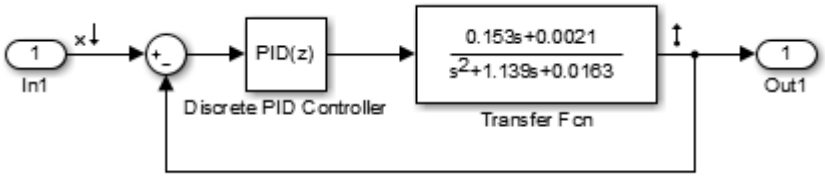


Рисунок 36 – Структурная схема системы управления

С помощью встроенной утилиты были получены оптимальные настройки ПИД-регулятора. Результаты авто настройки регулятора представлены на рисунке 37.

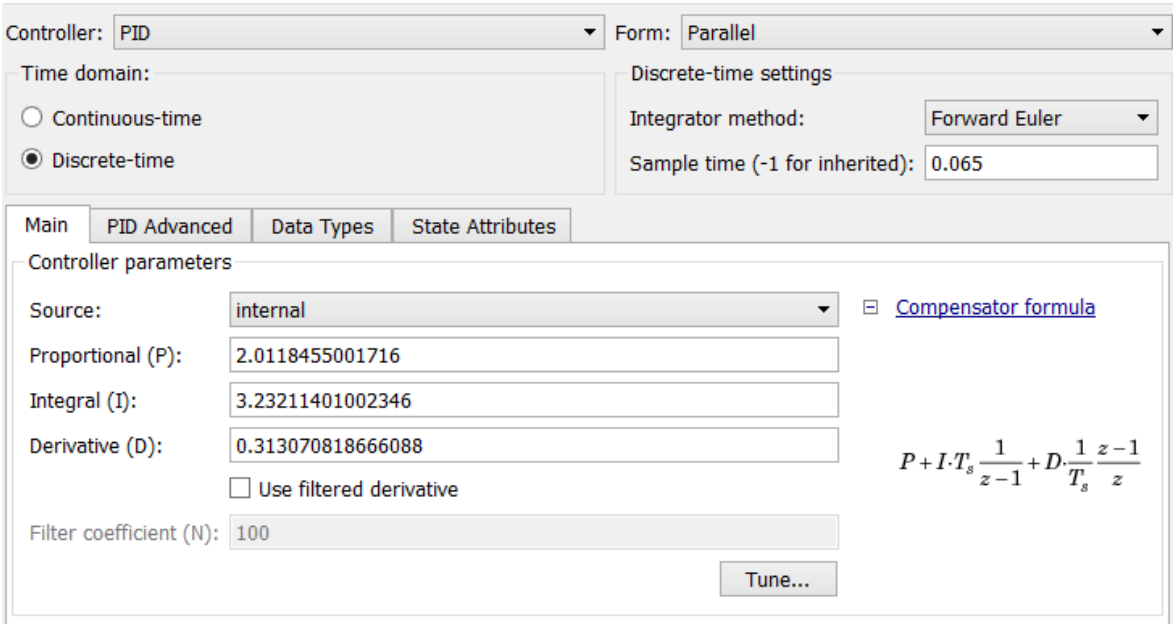


Рисунок 37 – Настройки ПИД-регулятора

На основе полученных данных и формулы (18) была реализована программная версия ПИД-регулятора на языке C#.

4.3. Синтез нейромодулятора

Для синтеза нейроэмулятора объект управления был зафиксирован в том же состоянии, что и ПИД-регулятор. После была произведена генерация обучающей выборки. Изображение обучающей выборки представлено на рисунке 38. По оси абсцисс обозначено время, по оси ординат – значение с энкодера.

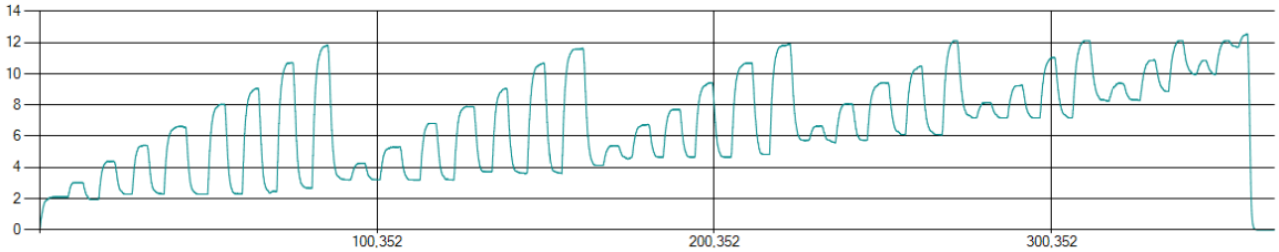


Рисунок 38 – Обучающая выборка для нейроэмулятора

На основе обучающей выборки было произведено обучение нейронной сети. График ошибок нейроэмулятора для каждого элемента из обучающей выборки до обучения представлен на рисунке 39. По оси абсцисс представлен элемент обучающей выборки, по оси ординат – ошибка элемента обучающей выборки, выраженная в оборотах в секунду с энкодера.

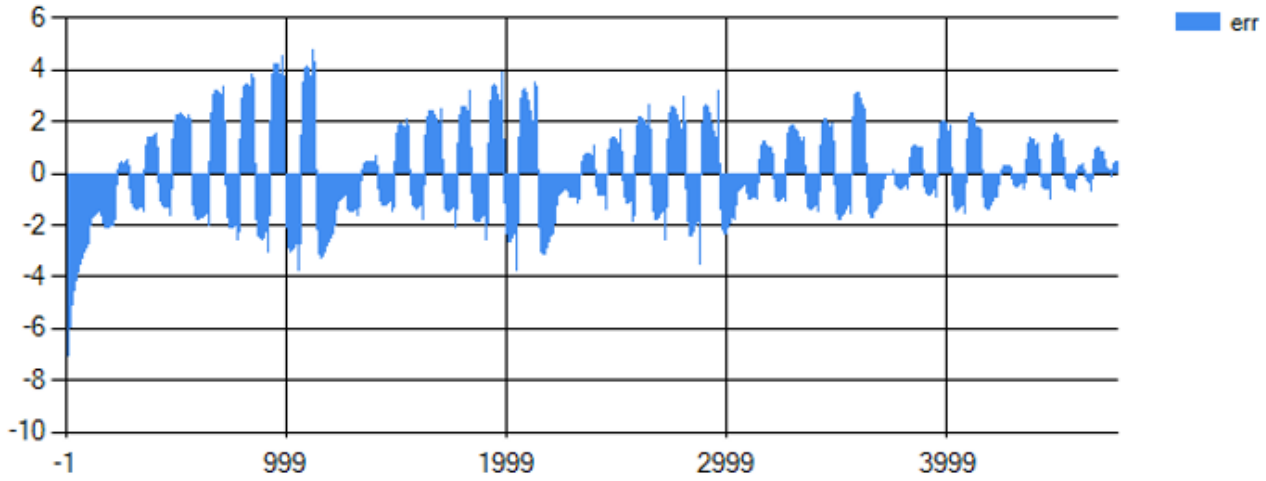


Рисунок 39 – График ошибок нейросети до обучения

График ошибок нейроэмулятора для каждого элемента из обучающей выборки после обучения представлен на рисунке 40.

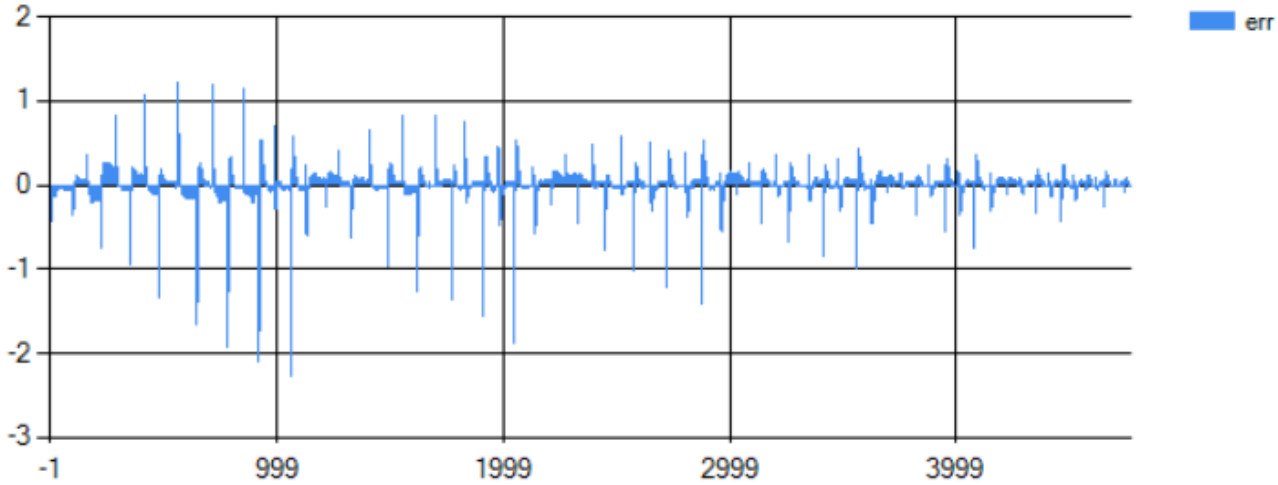


Рисунок 40 – График ошибок нейросети после обучения

Локальные пики ошибок обусловлены скачкообразным изменением значения на входе обучающей выборки при смене уставки. Нейронная сеть состоит из непрерывных функций и не в состоянии обротать такой скачок.

С помощью обученной нейронной сети был восстановлен сигнал на основе заданий, содержащихся в обучающей выборке. Результаты проверки представлены на рисунке 41

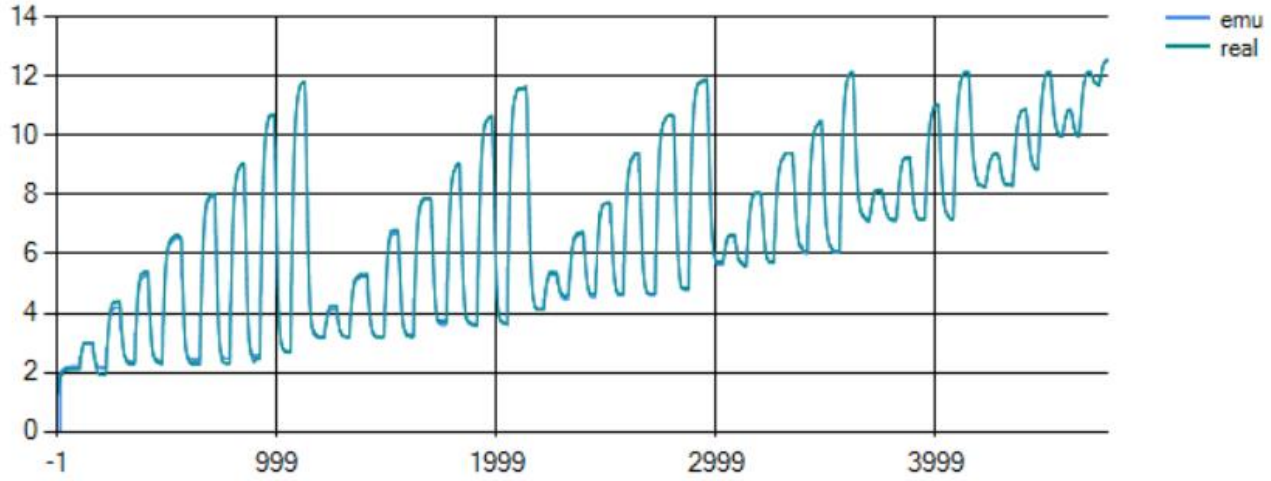


Рисунок 41 – Восстановленный сигнал

Таким образом, синтез нейроэмулятора прошел успешно, и полученная модель отличается высокой достоверностью.

4.4. Синтез инверсного нейромодулятора

На основе полученной в предыдущем пункте обучающей выборки была произведена попытка обучения инверсного нейроэмулятора. Результаты обучения после 10000 эпох представлены на рисунке 42.

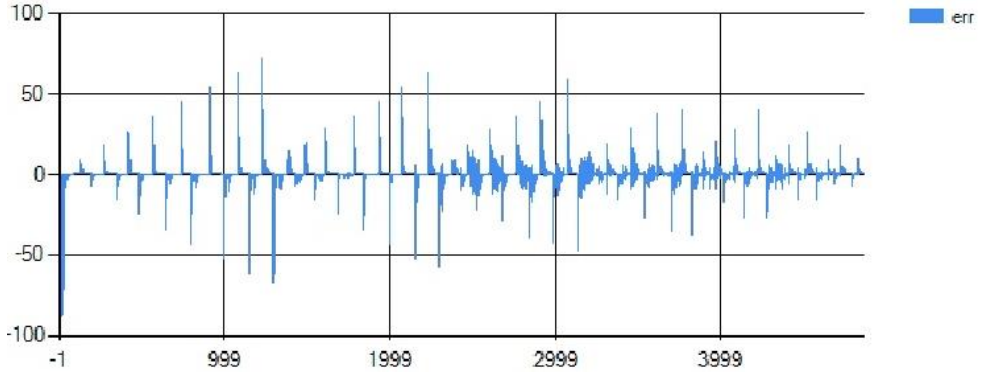


Рисунок 42 – Обучение инверсного нейроэмулятора

Обучение не удалось так как нейронная сеть на основе непрерывных элементов не может обучиться генерировать ступенчатый выходной сигнал на основе непрерывного входного сигнала. Таким образом, в выборке содержатся

противоречия, которые нейронная сеть выбранной архитектуры не может обработать. В связи с данным фактом была модифицирована методика генерации обучающей выборки. Процесс генерации обучающей выборки для инверсного нейроэмулятора представляет собой сбор информации о всех возможных задающих воздействий и соответствующим им установившимся значениям на выходе объекта. Генерация обучающей выборки представлена на рисунке 43.

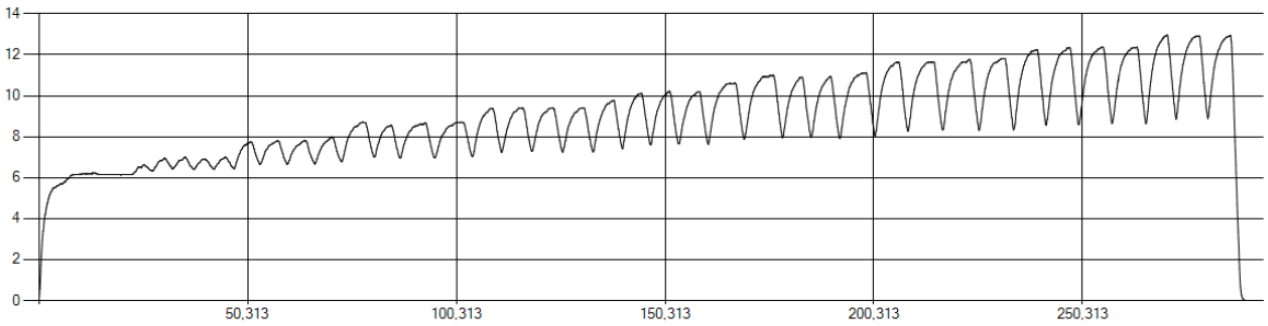


Рисунок 43 – Генерация обучающей выборки для инверсного нейроэмулятора

На основе модифицированной обучающей выборки было произведено обучение нейронной сети. График ошибок инверсного нейроэмулятора для каждого элемента из обучающей выборки представлен на рисунке 44. По оси абсцисс представлен элемент обучающей выборки, по оси ординат – ошибка элемента обучающей выборки, выраженная в задании для ДПТ.

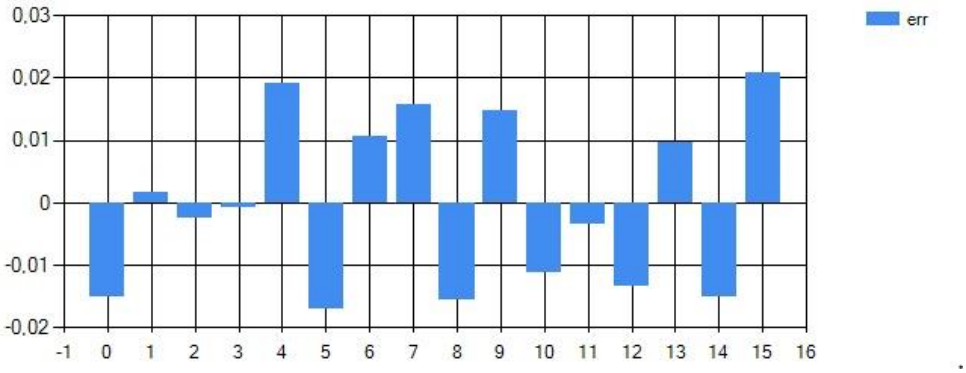


Рисунок 44 – График ошибок нейросети после обучения

На основе полученного инверсного нейроэмулятора и синтезированного ранее ПИД-регулятора был произведен сравнительный анализ переходных

процессов. На вход контроллеров подавались уставки с периодом 10 секунд в следующей последовательности: 7, 11, 6, 12. Графики переходных процессов представлены на рисунке 45 для инверсного нейрорегулятора и на рисунке 46 для ПИД-регулятора.

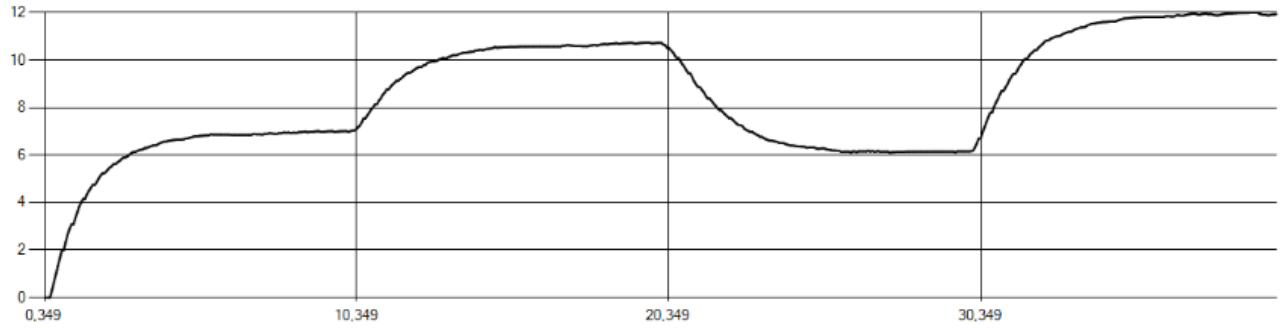


Рисунок 45 – График переходного процесса для инверсного нейроэмулятора

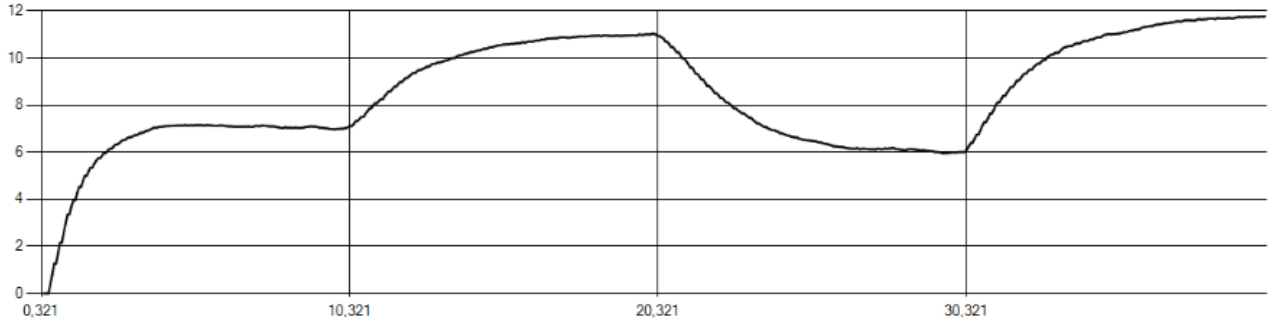


Рисунок 46 – График переходного процесса для ПИД-регулятора

Из графиков можно сделать вывод, что регуляторы обладают сходными показателями, но при больших скачках уставки ПИД-регулятор обеспечивает меньшее время переходного процесса, так как его выход зависит от текущей ошибки. В тоже время инверсный нейрорегулятор обеспечивает лучшее время переходного процесса при малых изменения уставки, так как его выход мгновенно устанавливается в соответствии с уставкой. Существенным минусом инверсного нейроэмулятора является то, что синтезированный контроллер действителен только для одного конкретного состояния объекта, что сильно ограничивает его область применения. По графику переходного процесса видно, что на каждом этапе присутствует незначительная статическая ошибка обусловленная ошибками в обучении и смешением состояния объекта во время испытаний.

4.5. Синтез многомодульного нейроконтроллера

Для компенсации недостатков инверсного нейроэмулятора была произведена модификация структуры регулятора в соответствии с многомодульной моделью. Пространство состояний объекта было разделено на пять состояний. Фиксация состояний обеспечивалась вращением барашка на половину оборота, контроль производился посредством рисок на винте. Для каждого из состояний объекта была сгенерирована обучающая выборка для нейроэмулятора и соответствующая ему выборка для инверсного нейроэмулятора. Были введены обозначения для состояний объекта и соответствующих им нейроэмуляторов: m1 – состояние с наименьшей нагрузкой на вал двигателя; m2, m3, m4 – промежуточные состояния; m5 – состояние с наибольшей нагрузкой. Интервалы состояний были выбраны таким образом, чтобы при наибольшей нагрузке и наименьшем задании, а также при наименьшей нагрузке и наибольшем задании стенд сохранял постоянство оборотов. Перечень состояний и соответствующие им обороты двигателя при фиксированном задании сведены в таблицу 7.

Таблица 7 – Обороты двигателя в секунду для фиксированных состояниях объекта и соответствующих заданиях

Задание	10	50	100
Состояние m1	4.57	9.50	14.76
Состояние m2	3.17	7.17	11.75
Состояние m3	2.11	6.10	10.24
Состояние m4	1.41	5.02	9.07
Состояние m5	0.53	4.31	8.40

Обучающие выборки прямых нейроэмуляторов приведены на рисунках 47 – 51.

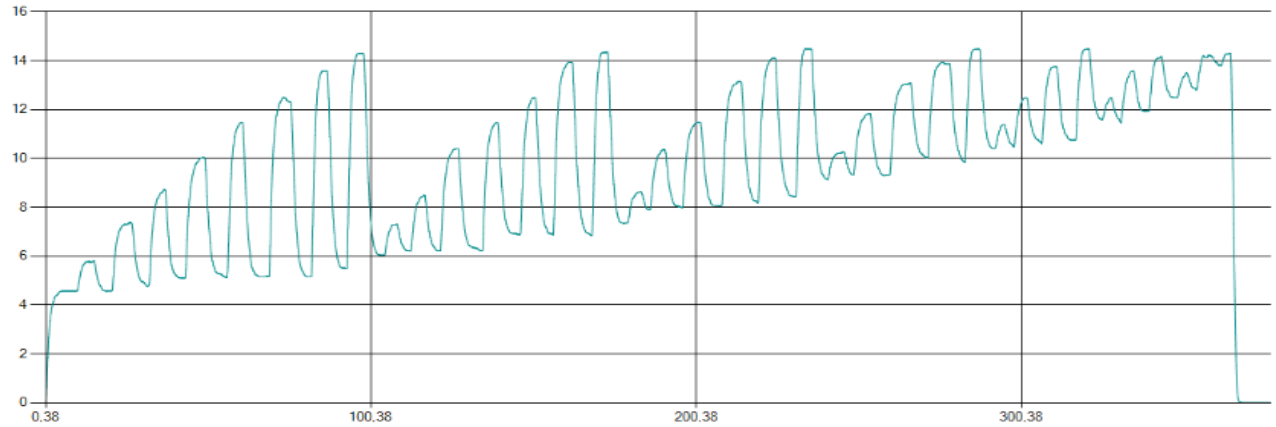


Рисунок 47 – Обучающая выборка прямого нейроэмулятора для состояния m1

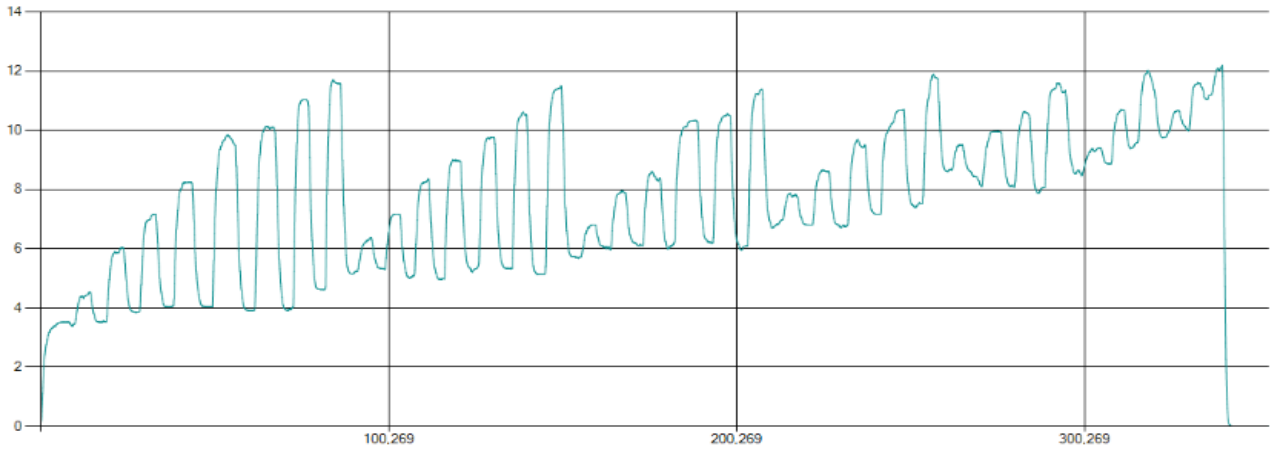


Рисунок 48 – Обучающая выборка прямого нейроэмулятора для состояния m2

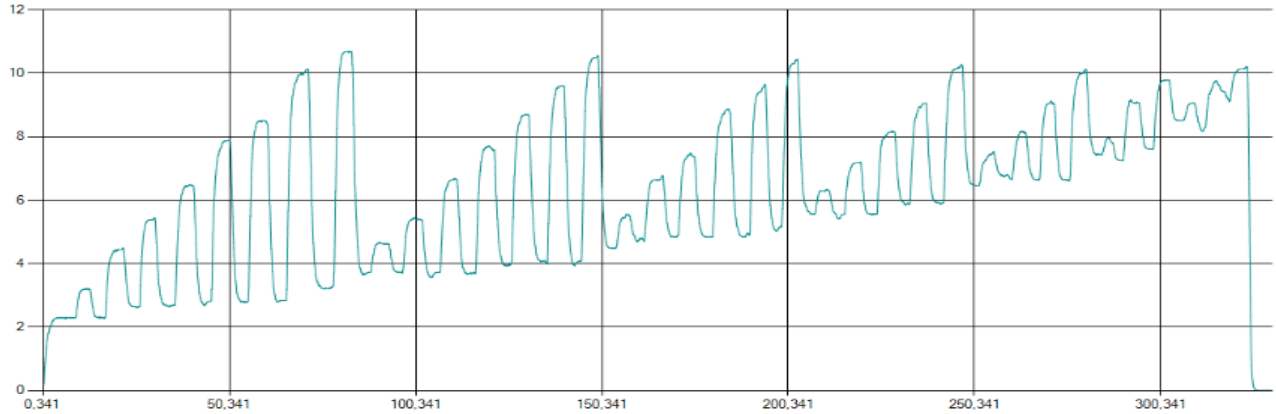


Рисунок 49 – Обучающая выборка прямого нейроэмулятора для состояния m3

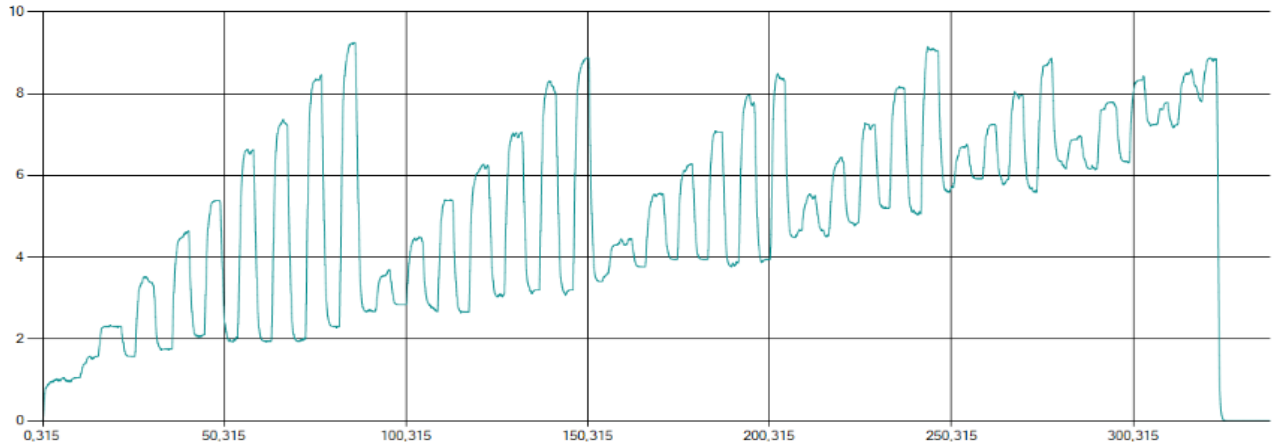


Рисунок 50 – Обучающая выборка прямого нейроэмулятора для состояния m4

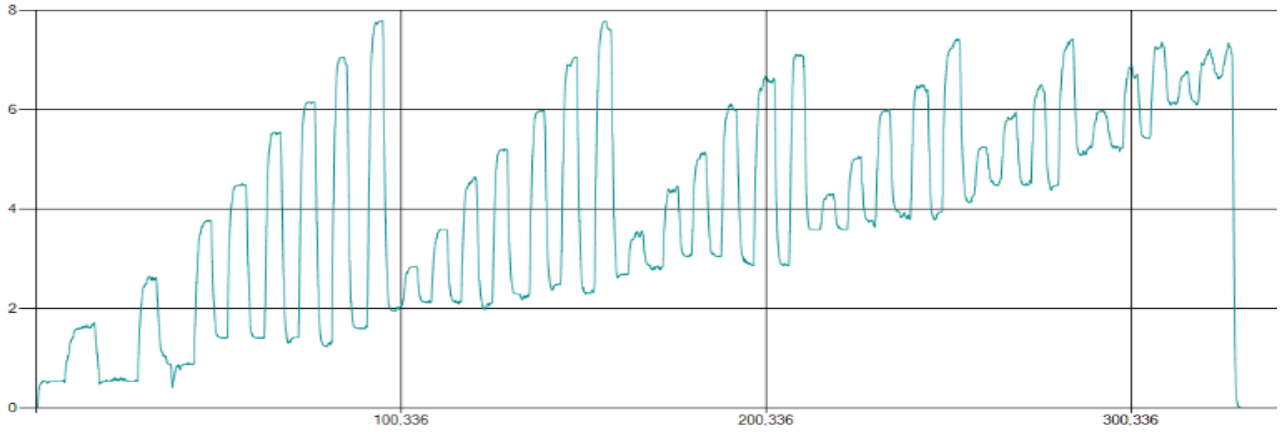


Рисунок 51 – Обучающая выборка прямого нейроэмулятора для состояния m5

Обучающие выборки инверсных нейроэмуляторов представлены на рисунках 52 –56.

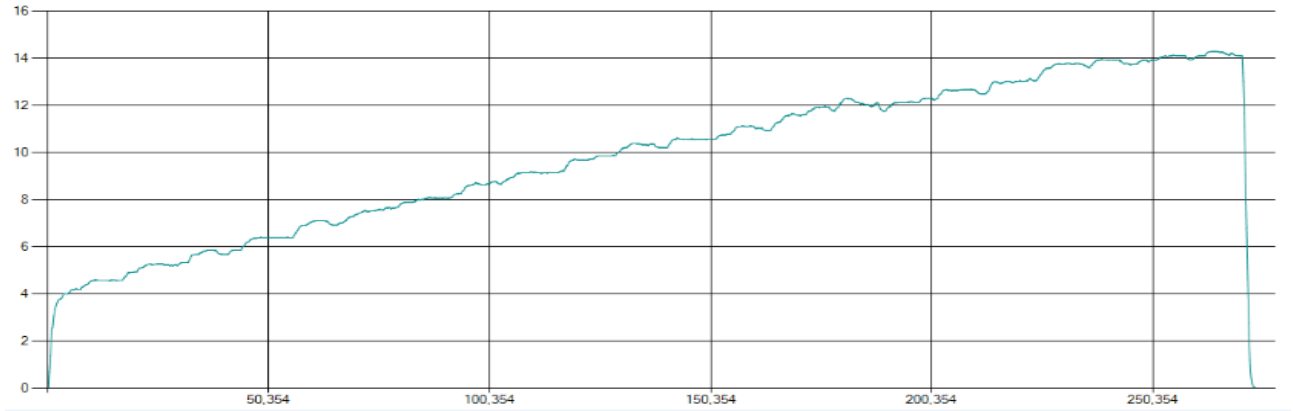


Рисунок 52 – Обучающая выборка инверсного нейроэмулятора для состояния m1

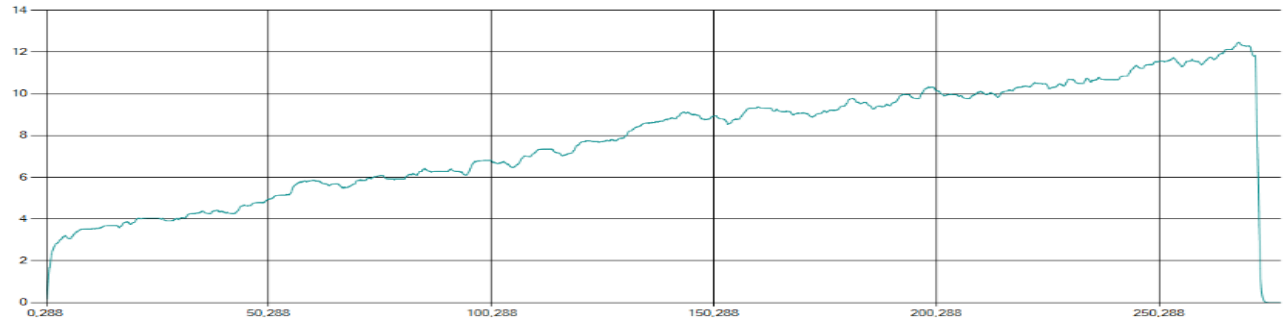


Рисунок 53 – Обучающая выборка инверсного нейроэмулятора для состояния m2

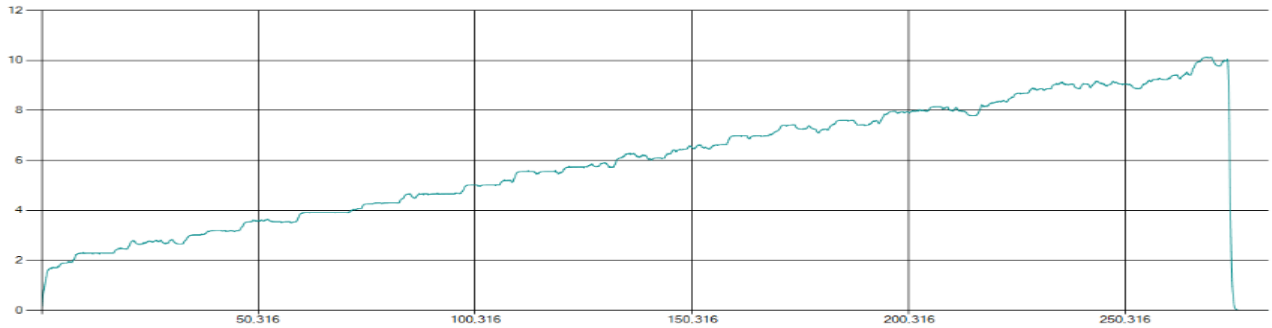


Рисунок 54 – Обучающая выборка инверсного нейроэмулятора для состояния m3

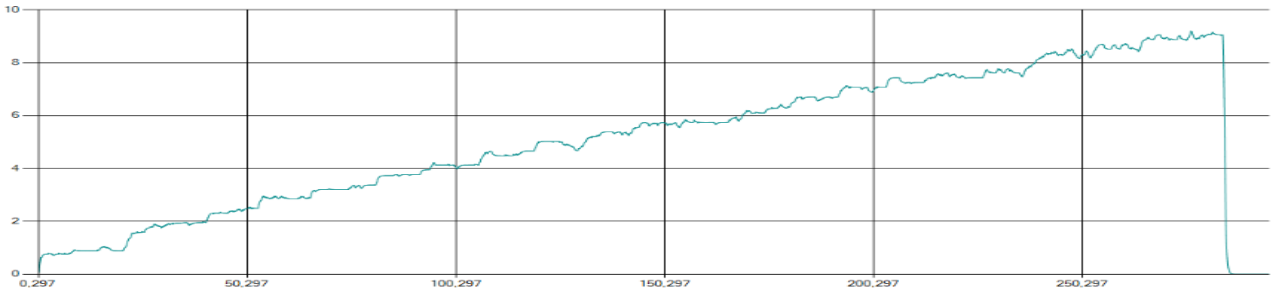


Рисунок 55 – Обучающая выборка инверсного нейроэмулятора для состояния m4

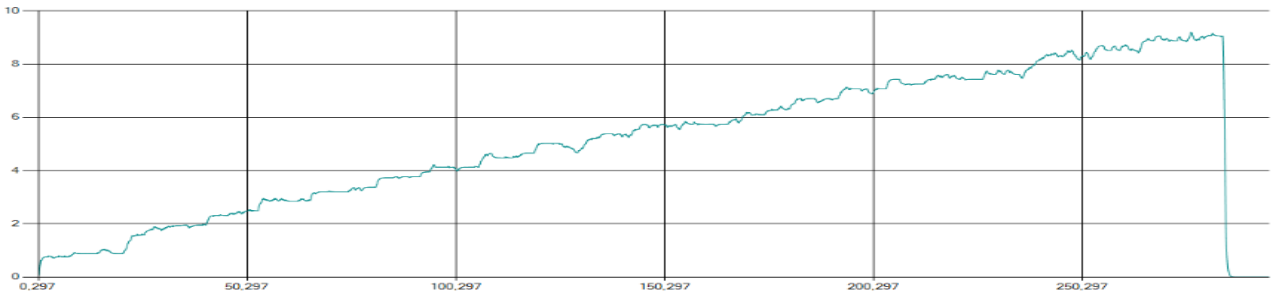


Рисунок 56 – Обучающая выборка инверсного нейроэмулятора для состояния m5

Для каждого из прямых и инверсных нейроэмуляторов было произведено обучение продолжительность 2000 эпох, что обеспечило приемлемый уровень итогового СКО в 0.001. После обучения было произведено испытание идентификации состояния объекта посредством нейроэмуляторов. Для этого была написана программа, рассчитывающая рассогласование между текущим значением оборотов двигателя и значением, выдаваемым нейроэмулятором. На основе полученных данных алгоритм определяет принадлежность текущего состояния одному из пяти возможных. Произведена проверка работы программы путем изменения нагрузки при фиксированном задании в 20. Результаты проверки предоставлены на рисунке 57. Цветами выделены выходы нейроэмуляторов. Черная линия – текущее значение с энкодера. На нижнем графике отображается идентифицированное состояние объекта.

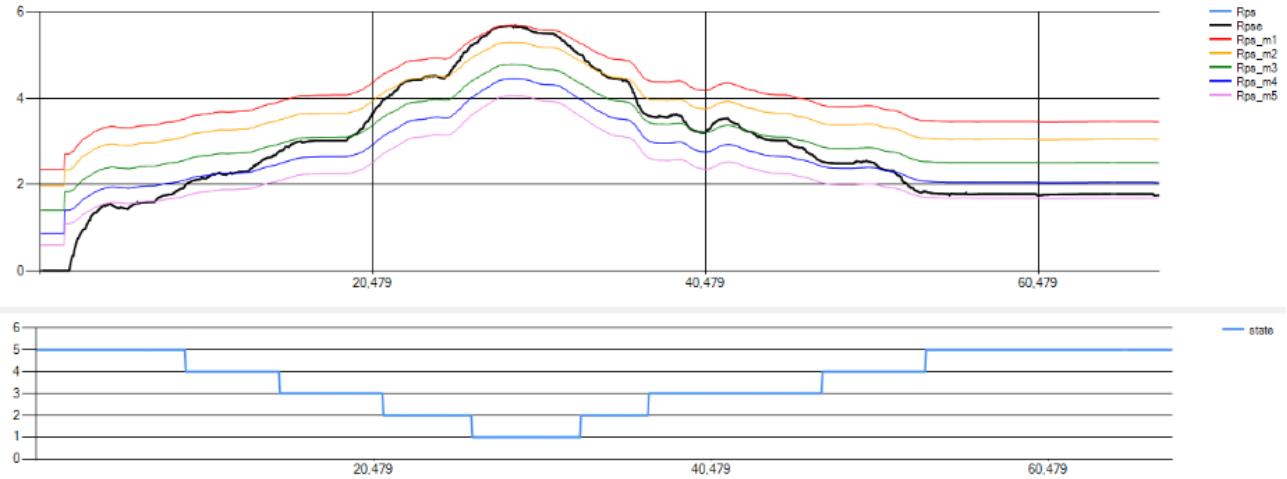


Рисунок 57 – Идентификация состояния объекта

По результатом исследования можно установить, что группа прямых нейроэмуляторов позволяет идентифицировать состояние объекта. Так же было выявлено несовершенство разработанного стенда. В процессе испытаний периодически менялась нагрузка при одинаковых значениях положения винта.

Для финальной реализации многомодульного нейроконтроллера была написана программа, передающая управление инверсному нейроэмулятору, соответствующему текущему состоянию. Полученный нейроконтроллер был

испытан посредством изменения нагрузки на валу двигателя при фиксированной уставке в 6 оборотов в секунду. Результаты испытания предоставлены на рисунке 58.

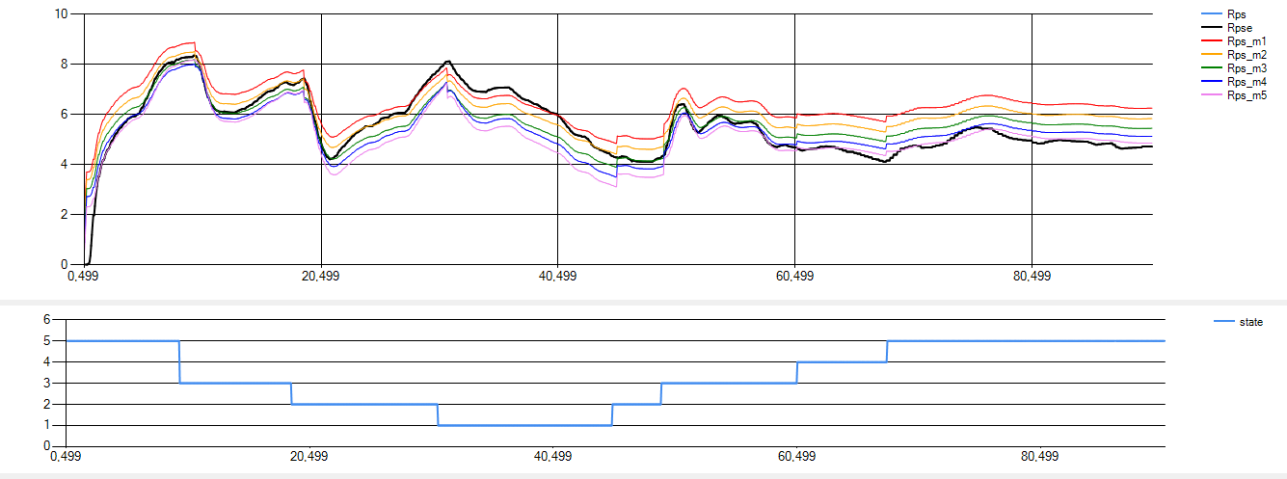


Рисунок 58 – Результаты работы многомодульного нейроконтроллера

По результатам испытания было выявлено низкое качество переходного процесса, обусловленное скачками задания при смене состояния объекта. Так же в интервале между состояниями наблюдалось сильное влияние момента на итоговое значение с энкодера. Данная проблема может быть решена путем увеличения количества пар модулей прямой–инверсный нейроэмуляторор, или использования дополнительно контроллера для минимизации статической ошибки.

4.6. Синтез многомодульного нейроконтроллера с использованием ПИ-регуляторов

В процессе испытаний было выявлена непригодность многомодульной архитектуры типа прямой–инверсный нейроэмуляторор для применяемого учебного стенда. В процессе исследований менялись параметры устройства управления моментом, вследствие чего заданный для прямого нейроэмулятора инверсный нейроэмулятор не мог считаться адекватным. Исходя из этого факта было решено применить ПИ-регуляторы вместо инверсных нейроэмуляторов.

Выбор данного регулятора обусловлен простотой и высокой пригодностью для типов объектов, сходных с исследуемым. Формула дискретного ПИ-регулятора выглядит как (19):

$$u(N) = K_p \cdot \mathrm{e(N)} + T \cdot K_i \cdot \sum_{i=1}^{n} e(N-i). \qquad (20)$$

В соответствие с (20) был реализован класс PI, описывающий дискретный ПИ-регулятор. Для каждого из состояний объекта была произведена настройка ПИ регулятора. Результаты настройки представлены на рисунке 59. Первый параметр конструктора класса PI определяет пропорциональный коэффициент регулятора, второй параметр определяет интегральный коэффициент.

```
PI_m1 = new PI(4, 0.2);
PI_m2 = new PI(3.5, 0.28);
PI_m3 = new PI(2.5, 0.32);
PI_m4 = new PI(2, 0.4);
PI_m5 = new PI(2, 0.45);
```

Рисунок 59 – Инициализация программных ПИ-регуляторов

Полученные ПИ-регуляторы были использованы вместо инверсных нейроэмуляторов и испытаны аналогичным образом. Результаты работы многомодульного нейроконтроллера с ПИ-регуляторами при уставке 6 оборотов в секунду и меняющейся нагрузкой приведены на рисунке 60.

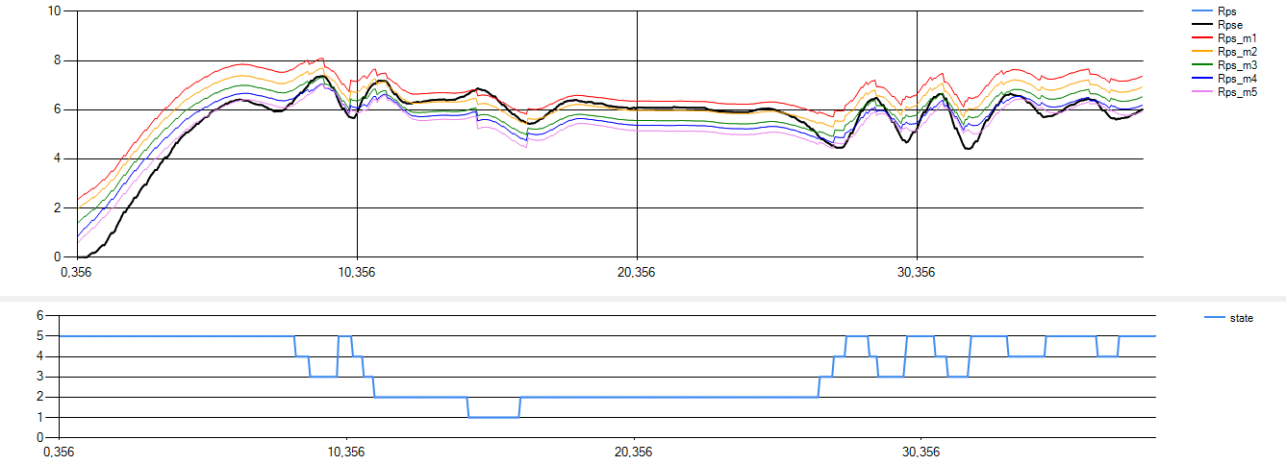


Рисунок 60 – Результаты работы многомодульного нейроконтроллера с ПИ-регуляторами

В ходе испытаний были выявленны недостатки разработанного метода, связанные сразличными значениями уставок. В процессе работы регулятора, при смене состояний происходил резкий скачок уставки, связанный с различной накомленной интегральной состовляющей ПИ-регуляторов. Данный скачок приводил к ошибочной идентификации состояния, что в сою очередь порождало новые переключения регуляторов.

Для исправления данного недостатка интегральные части всех ПИ-регуляторов были сведены в одну глобальную переменную IntegralSum. Полученное решение можно рассматривать как один ПИ-регулятор с возможность изменения параметров. Результаты работы модефицированного регулятора представлены на рисунке 61.

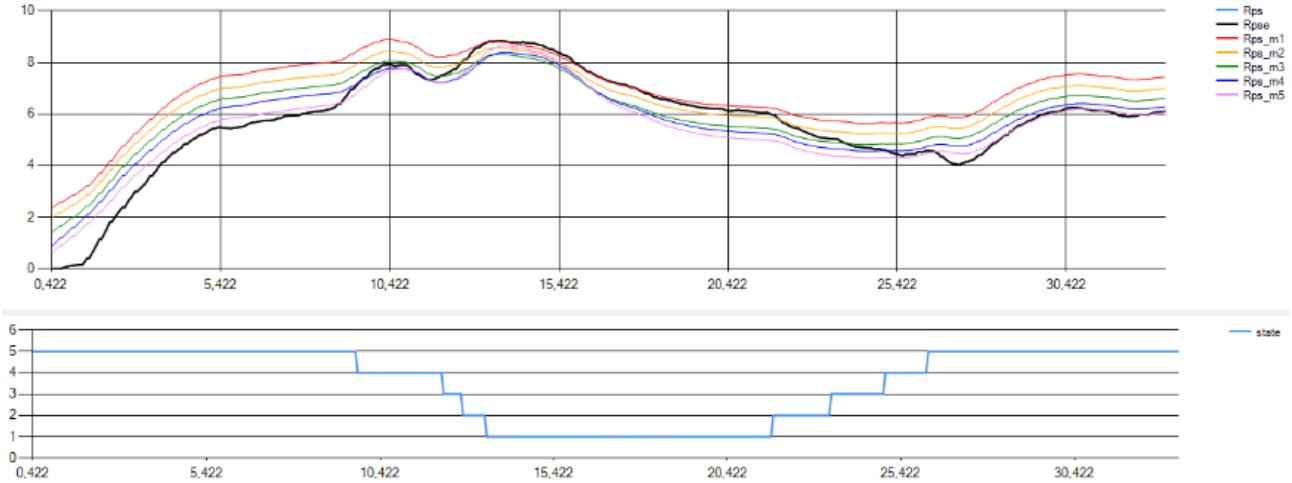


Рисунок 61 – Результаты работы модифицированного многомодульного нейроконтроллера с ПИ-регуляторами

Полученный контроллер сохранял стабильность на всем интервале регулирования. Для сравнения с типовым решением было произведено исследование ПИД-регулятора при аналогичных условиях. Настройка ПИД регулятора была произведена для состояния m3, которое совпадает с состоянием объекта, используемым ранее для идентификации системы в MatLAB и соответствующем синтезе ПИД-регулятора. Результаты работы ПИД регулятора представлены на рисунке 62.

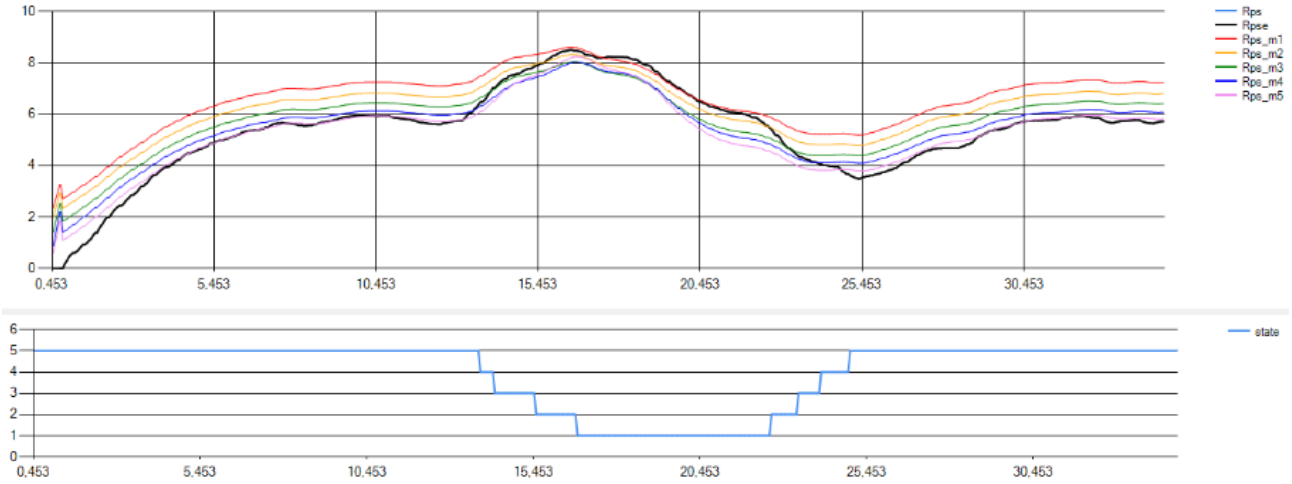


Рисунок 62 – Результаты работы ПИД-регулятора

Полученному нейроконтроллеру удалось достичь более высоких, в сравнении с ПИД-регулятором, качеств переходного процесса. Обеспечено меньшее время переходного процесса и меньшее перерегулирование при изменении нагрузки на валу.

Полученные преимущества не оправдываются сложной процедурой синтеза и необходимостью высоких аппаратных мощностей при обучении нейронных сетей. Для исследуемого учебного стенда рационально использовать типовой ПИД-регулятор, так как состояния системы не постоянны и могут меняться в процессе исследования вследствие износа частей устройства, обеспечивающего изменение момента. В тоже время, полученное решение может быть с успехом применено в учебных целях для исследования работы нейронных сетей и синтеза собственных вариантов контроллеров с их использованием.

5. ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

Введение

В данном разделе рассмотрен анализ экономической эффективности разработанной системы автоматизированного управления двигателем постоянного тока на основе нейронного регулятора и перспективности ведения научно-исследовательских работ в данном направлении.

Разрабатываемый учебный стенд представляет собой двигатель постоянного тока с возможностью динамического изменения момента на вал двигателя. Управление двигателем осуществляется посредством аппаратно-вычислительной платформы RobotDyn NANO V3 совместно с контроллером Skywalker 40A. Питание контролера осуществляется от источника постоянного тока напряжением 12В. Нейронный регулятор реализован в виде Windows-приложения на языке C#. Обмен данными между микроконтроллером и ПК производится посредством интерфейса UART.

Целью научно-исследовательской работы является синтез нейронного регулятора для управления двигателем постоянного тока и сравнение полученного решения с аналогичными стандартными решениями.

Для достижения данной цель необходимо собрать учебный стенд, состоящий из двигателя постоянного тока и управляющей электронике на основе микроконтроллера. Реализовать обмен данными между ПК и микроконтроллером. Выбрать архитектуру нейронного регулятора и разработать методику обучения. Произвести синтез и настройку нейронного регулятора. Оценить качества переходного процесса и сравнить с аналогичными типовыми решениями в автоматике

Результаты работы могут быть использованы в качестве учебного стенда для демонстрации работы нейронного регулятора, а так же применены в производстве для повышения качества управления двигателем постоянного тока.

5.1. Предпроектный анализ

5.1.1. Потенциальные потребители результатов исследования

Для анализа потребителей результатов исследования необходимо рассмотреть целевой рынок и провести его сегментирование. Сегментирование – это разделение покупателей на однородные группы, для каждой из которых может потребоваться определенный товар (услуга).

Потенциальными потребителями НИР могут являться как физические, юридические лица, так и коммерческие организации различных размеров, ориентирующиеся на широкий спектр промышленности и науки. Выберем рынок услуг по разработке регуляторов и сегментируем его по областям применения регуляторов и размеру компаний-потребителей. Полученная карта сегментирования рынка представлена на рисунке 63.

		Область применения			
		Образование	Производство	Военная промышленность	Потребительские товары
Размер компании	крупные		К	К	
	средние		К		
	мелкие		К		

Рисунок 63 – Карта сегментирования рынка услуг по разработке регуляторов

На карте отмечены заинтересованные в данных услугах компании-потребители. Сегментирование произведено по размеру потребителей услуг и по области применения регуляторов. Буквой «К» отмечены те сегменты, на которых наблюдается высокая конкуренция. Конкуренция представлена как иностранными (Siemens, Schneider Electric, Allen Bradley, Yokogawa) так и отечественными (Овен, Метран, Элеси) компаниями.

Из приведенной карты сегментирования можно сделать вывод, что основной областью применения регуляторов является производство, там же

наблюдается самая высокая конкуренция. Остальные области представлены крупными компаниями-потребителями.

В виду высокого порога вхождения на рынок услуг по разработке регуляторов для производственных целей и военной промышленности, потенциальным потребителем решено выбрать образовательные учреждения.

5.1.2. Анализ конкурентных технических решений

Оценочная карта конкурентных технических решений приведена в таблице 8.

Таблица 8 – Оценочная карта конкурентных технических решений

Критерии оценки	**Вес крите-рия**	**Баллы**			**Конкуренто-способность**		
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$К_ф$	$К_{к1}$	$К_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1.Простота настройки	0.2	4	1	3	0.8	0.2	0.6
2.Потребность в аппаратных ресурсах	0.05	1	3	5	0.05	0.15	0.25
3.Функциональное исполнение системы	0.1	2	3	5	0.2	0.3	0.5
4.Качество управления	0.2	5	3	1	1	0.6	0.2
5.Уровень унификации	0.05	2	5	5	0.1	0.25	0.25
Экономические критерии оценки эффективности							
1.Конкурентоспособность	0.1	4	5	3	0.4	0.5	0.3
2.Уровень востребованности среди потребителей	0.1	2	5	3	0.2	0.5	0.3
3.Цена	0.1	2	4	5	0.2	0.4	0.5
4.Финансирование разработки	0.05	4	2	2	0.2	0.1	0.1
5.Срок исполнения системы	0.05	2	3	3	0.1	0.15	0.15
Итого	**1**	**28**	**34**	**35**	**3.25**	**3.15**	**3.15**

В таблице буквой $Б_Ф$ обозначено разрабатываемое решение, в качестве конкурентных технических решений выбраны трехпозиционный регулятор, обозначенный как $Б_{К1}$, и ПИД-регулятор, обозначенный как $Б_{К2}$. Соответствующими буквами $К_Ф$, $К_{К1}$ и $К_{К2}$ обозначены коэффициенты конкурентоспособности, равные произведению бальных оценок категорий и соответствующих весовых коэффициентов. Исходя из расчетов, сделанных выше, разрабатываемое техническое решение имеет высокий уровень

конкурентоспособности. Конкурентное преимущество обусловлено простотой настройки и высоким качеством управления.

5.1.3. Диаграмма Исикавы

Диаграмма причины-следствия Исикавы – это графический метод анализа и формирования причинно-следственных связей. Диаграмма представлена на рисунке 64.

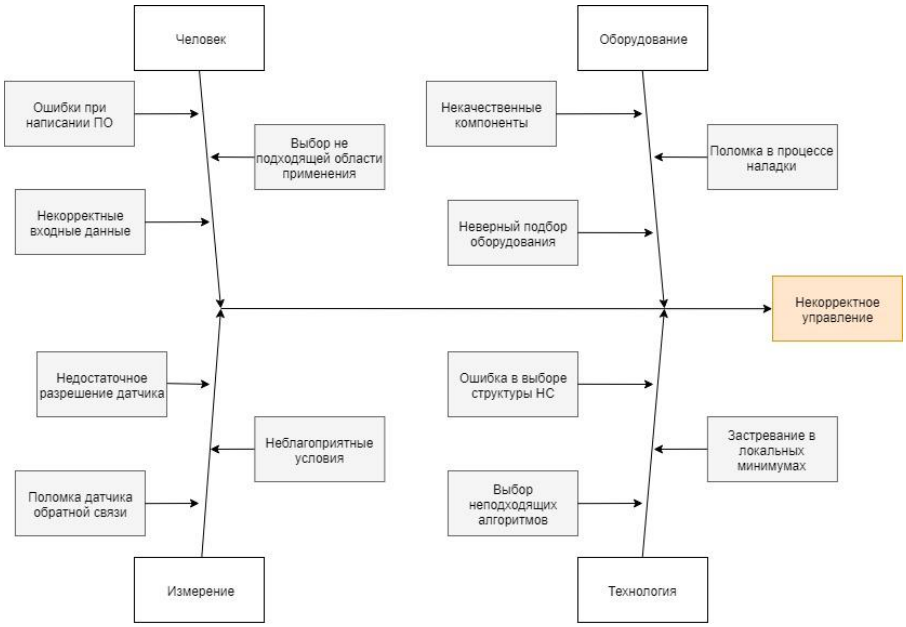


Рисунок 64 – Диаграмма Исикавы

В качестве проблемной области было выбрано качество управления. На качество влияют такие группы факторов, как человек, измерение, оборудование и технология. Ошибки разработчика могут привести к некорректной структуре регулятора и неправильному обучению. Неверное измерение приведет к ложному представлению регулятора об объекте управления. Ошибка в выборе компонентов и риски, связанные с качеством компонентов, могут привести к некорректной работе объекта управления. Неправильная технология может привести к невозможности синтеза необходимого регулятора. Таким образом, каждый из выявленных факторов влияет на снижение качества управления.

5.1.4. SWOT-анализ

SWOT анализ представляет собой комплексный анализ научно-исследовательского проекта и применяется для выявления сильных и слабых сторон проекта, возможностей развития и угроз существованию. Матрица SWOT представлена в таблице 9.

Таблица 9 – SWOT-анализ

	Сильные стороны научно-исследовательского проекта: С1. Простота настройки и эксплуатации системы. С2. Возможность применения в учебном процессе. С3. Возможность простой отладки ПО на персональном компьютере.	**Слабые стороны научно-исследовательского проекта:** Сл1. Недостаточное количество отработанных методов получения данных для обучения. Сл2. Необходимость использования персонально компьютера для обучения НС.
Возможности: В1. Использование инфраструктуры ТПУ для распространения предложенного проекта. В2. Получение финансирования для дальнейшего более глубокого исследования. В3. Возможность применить результаты исследования на других системах в университете.	Простота настройки позволит легко опробовать результаты исследования на объектах ТПУ, что позволит в дальнейшем, с легкостью внедрять и на другие объекты. Относительная дешевизна позволит получить финансирование для дальнейших разработок по данной теме. Наглядность и возможность простой отладки позволит применять разработку в учебных целях.	Недостаточное количество исследованных методов и не самое может привести к неудовлетворительным результатам на других объектах и системах. Необходимость использования персонального компьютера может ограничить область применения разработки.
Угрозы: У1. Развитая конкуренция. У2. Остановка исследования из-за необходимости использования качественно новых методов и подходов. У3. Отсутствие спроса на новые технологии производства.	Отсутствие спроса на новые технологии производства может замедлить срок выхода разработки на рынок и понизить квалификацию научного труда. Развитая конкуренция на рынке регуляторов и новые методы могут привести к снижению конкурентоспособности	В силу необходимости использовать персональный компьютер резко понижается показатель конкурентоспособности на рынке производственных регуляторов. Дальнейшая разработка может потребовать использования дорогостоящего оборудования и принципиально новые методы.

5.2. Инициализация проекта

Планирование комплекса предполагаемых работ по разработке системы управления двигателем постоянного тока организовано в следующем порядке:

- определение структуры работ в рамках научного исследования;
- определение участников каждой работы;
- установление продолжительности работ;
- построение графика проведения научных исследований.

Для выполнения данного научного исследования была сформирована рабочая группа, в состав которой входит научный сотрудник и студент – дипломник.

Перечень работ и этапов в рамках проведения научного проекта и распределение исполнителей по видам работ представлены в таблице 10.

Таблица 10 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№ раб	Содержание работ	Должность исполнителя
Разработка технического задания	1	Составление и утверждение технического задания	Руководитель
Выбор направления исследований	2	Подбор и изучение материалов по теме	Студент
	3	Выбор способа решения задачи	Руководитель, студент
	4	Календарное планирование работ по теме	Руководитель
Теоретические и экспериментальные исследования	5	Разработка структурной (принципиальной) схемы устройства	Руководитель, студент
	6	Выбор компонентов устройства	Руководитель, студент
	7	Выбор и изучение ПО	Студент
	8	Сборка устройства	Студент
	9	Оптимизация аппаратной части	Студент
	10	Оптимизация программной части	Студент
	11	Тестирование устройства	Руководитель, студент
Обобщение и оценка результатов	12	Оценка эффективности полученных результатов	Руководитель
Проведение ОКР			
Разработка технической документации и проектирование	13	Технико-экономические расчеты	Студент
	14	Вопросы безопасности и экологичности проекта	Студент
	15	Составление пояснительной записки (эксплуатационно-технической документации)	Студент

5.3. Планирование управления научно-техническим проектом

Определение трудоемкости работ каждого из участников исследования может считаться крайне важным моментом, так как трудовые затраты по большей части образуют основную часть стоимости разработки.

Трудоемкость выполнения научного исследования оценивается экспертным путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости $t_{ожi}$ используется формула (21).

$$t_{ожi} = \frac{3t_{\min i} + 2t_{\max i}}{5}, \qquad (21)$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения *i*-ой работы чел.-дн.;

$t_{\min i}$ – минимально возможная трудоемкость выполнения заданной *i*-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн.;

$t_{\max i}$ – максимально возможная трудоемкость выполнения заданной *i*-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Для выполнения перечисленных в таблице работ требуются специалисты: студент, научный руководитель. Исходя из ожидаемой трудоемкости работ, по формуле (22) определяется продолжительность каждой работы в рабочих днях $T_р$, учитывающая параллельность выполнения работ несколькими исполнителями.

$$T_{рi} = \frac{t_{ожi}}{Ч_i}, \qquad (22)$$

где $T_{рi}$ – продолжительность одной работы, раб. дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

5.3.1. Разработка графика проведения научного исследования

Для удобства построения графика, длительность каждого из этапов работ из рабочих дней следует перевести в календарные дни. Для этого необходимо воспользоваться формулой (23).

$$T_{\text{к}i} = T_{\text{р}i} \cdot k_{\text{кал}}, \tag{23}$$

где $T_{\text{к}i}$ – продолжительность выполнения *i*-й работы в календарных днях;

$T_{\text{р}i}$ – продолжительность выполнения *i*-й работы в рабочих днях;

$k_{\text{кал}}$ – коэффициент календарности.

Коэффициент календарности определяется по формуле (24).

$$k_{\text{кал}} = \frac{T_{\text{кал}}}{T_{\text{кал}} - T_{\text{вых}} - T_{\text{пр}}}, \tag{24}$$

где $T_{\text{кал}}$ – календарные дни ($T_{\text{кал}}$ = 365);

$T_{\text{вых}}$ – выходные дни ($T_{\text{вых}}$ = 116);

$T_{\text{пр}}$ – праздничные дни ($T_{\text{пр}}$ = 14).

Рассчитанные значения в календарных днях по каждой работе $T_{\text{к}i}$ необходимо округлить до целого числа. Все рассчитанные значения сведены в таблицу 11.

Пример расчета (составление и утверждение технического задания) предоставлен в формулах (25), (26), (27) и (28).

$$t_{ожс} = \frac{3 \cdot t_{\min} + 2 \cdot t_{\max}}{5} = \frac{3 \cdot 1 + 2 \cdot 2}{5} = 1,4 \approx 2 \; чел - дней; \tag{25}$$

$$T_p = \frac{t_{ожс}}{Ч} = \frac{2}{1} = 2 \; дня; \tag{26}$$

$$k_{кал} = \frac{T_{кал}}{T_{кал} - T_{вых} - T_{пр}} = \frac{365}{365 - 116 - 14} = 1,553; \tag{27}$$

$$T_к = T_p \cdot k_{кал} = 2 \cdot 1,553 = 3,106 \approx 4 \; дня. \tag{28}$$

Таблица 11 – Временные показатели проведения научного исследования

Название работы	Трудоёмкость работ						Длительность работ в рабочих днях $T_{рi}$		Длительность работ в календарных днях $T_{кi}$	
	t_{min}, чел-дни		t_{max}, чел-дни		$t_{ожi}$, чел-дни					
	Руководитель	Студент	Руководитель	Студент	Руководитель	Студент	Руководитель	Студент	Руководитель	Студент
Составление и утверждение технического задания	1		2		2		2		4	
Подбор и изучение материалов по теме		5		8		7		7		11
Выбор способа решения задачи	3	3	4	4	4	4	4	4	7	7
Календарное планирование работ по теме	3		5		4		4		7	
Разработка структурной (принципиальной) схемы устройства	4	4	9	9	6	6	6	6	10	10
Выбор компонентов устройства	4	4	9	6	6	5	6	5	10	8
Выбор и изучение ПО		2		4		8		8		12
Сборка устройства		2		4		3		3		5
Оптимизация аппаратной части	3		6		5		5		8	
Оптимизация программной части		4		8		6		6		10
Тестирование устройства	5	5	8	8	7	7	7	7	11	11
Оценка эффективности полученных результатов	2		3		3		3		5	
Технико-экономические расчеты		3		7		5		5		8
Вопросы безопасности и экологичности проекта		3		7		5		5		8
Составление пояснительной записки		1		3		2		2		4

Следующим этапом является построение календарного плана-графика на основании данных в таблице 11. График строится для максимального по длительности исполнения работ в рамках научно-исследовательского проекта с разбивкой по месяцам и декадам (10 дней) за период времени выполнения научно-технического исследования. Штриховкой обозначен научный

руководитель, а сплошным цветом – студент. Календарный план-график приведен в таблице 12.

Таблица 12 – Календарный план-график проведения НИОКР

№ работ	Вид работ	Исполнители	$T_{кi}$, кал. дн.	Продолжительность выполнения работ												
				фев.		март			апрель			май			июнь	
				2	3	1	2	3	1	2	3	1	2	3	1	2
1	Составление и утверждение технического задания	Руководитель	4													
2	Подбор и изучение материалов по теме	Студент	11													
3	Выбор способа решения задачи	Руководитель, студент	7													
4	Календарное планирование работ по теме	Руководитель	7													
5	Разработка структурной (принципиальной) схемы устройства	Руководитель, студент	10													
6	Выбор компонентов устройства	Руководитель, студент	8													
7	Выбор и изучение ПО	Студент	12													
8	Сборка устройства	Студент	5													
9	Оптимизация аппаратной части	Студент	3													
10	Оптимизация программной части	Студент	10													
11	Тестирование устройства	Руководитель, студент	11													

Продолжение таблицы 12

12	Оценка эффективности полученных результатов	Руководитель	2	
13	Технико-экономические расчеты	Студент	8	
14	Вопросы безопасности и экологичности проекта	Студент	8	
15	Составление пояснительной записки	Студент	4	
	Общее время на проект		111	
	Время работы студента		98	
	Время работы руководителя		49	

5.3.2. Бюджет научно-технического исследования (НТИ)

При планировании бюджета НТИ должно быть обеспечено полное и достоверное отражение всех видов расходов, связанных с его выполнением. В процессе формирования бюджета НТИ используется следующая группировка затрат по статьям:

- материальные затраты НТИ;
- затраты на специальное оборудование для научных (экспериментальных) работ;
- основная заработная плата исполнителей темы;
- дополнительная заработная плата исполнителей темы;
- отчисления во внебюджетные фонды (страховые отчисления);
- затраты научные и производственные командировки;
- контрагентные расходы;
- накладные расходы.

5.3.4.1. Расчет материальных затрат НТИ

Данная статья включает стоимость всех материалов, используемых при разработке проекта:

- приобретаемые со стороны сырье и материалы, необходимые для создания научно-технической продукции;
- покупные материалы, используемые в процессе создания научно-технической продукции для обеспечения нормального технологического процесса и для упаковки продукции или расходуемых на другие производственные и хозяйственные нужды (проведение испытаний, контроль, содержание, ремонт и эксплуатация оборудования, зданий, сооружений, других основных средств и прочее), а также запасные части для ремонта оборудования, износа инструментов, приспособлений, инвентаря, приборов, лабораторного оборудования и других средств труда, не относимых к основным средствам, износ спецодежды и других малоценных и быстроизнашивающихся предметов;
- покупные комплектующие изделия и полуфабрикаты, подвергающиеся в дальнейшем монтажу или дополнительной обработке;
- сырье и материалы, покупные комплектующие изделия и полуфабрикаты, используемые в качестве объектов исследований (испытаний) и для эксплуатации, технического обслуживания и ремонта изделий – объектов испытаний (исследований).

В материальные затраты, помимо вышеуказанных, включаются дополнительно затраты на канцелярские принадлежности, диски, картриджи и т.п. Однако их учет ведется в данной статье только в том случае, если в научной организации их не включают в расходы на использование оборудования или накладные расходы. В первом случае на них определяются соответствующие нормы расхода от установленной базы. Во втором случае их величина учитывается как некая доля в коэффициенте накладных расходов. Расчет материальных затрат осуществляется по формуле (29).

$$З_м = (1 + k_T) \times \sum_{i=1}^{m} Ц_i \times N_{расхi}, \quad (29)$$

где m – количество видов материальных ресурсов, потребляемых при выполнении научного исследования;

$N_{расхi}$ – количество материальных ресурсов i-го вида, планируемых к использованию при выполнении научного исследования (шт., кг, м, м^2 и т.д.);

$Ц_i$ – цена приобретения единицы i-го вида потребляемых материальных ресурсов (руб./шт., руб./кг, руб./м, руб./м^2 и т.д.);

k_T – коэффициент, учитывающий транспортно-заготовительные расходы.

Величина коэффициента k_T, отражающего соотношение затрат по доставке материальных ресурсов и цен на их приобретение, зависит от условий договоров поставки, видов материальных ресурсов, территориальной удаленности поставщиков и т.д. Транспортные расходы принимаются в пределах 15-25% от стоимости материалов.

В таблице 13 представлены данные о материальных затратах:

Таблица 13 – Материальные затраты

Наименование материалов	**Цена за ед., руб.**	**Количество**	**Сумма, руб.**
Платформа Robodyn NANO	1000	1	1000
Контроллер Ardumoto L298P	900	1	900
Энкодер	200	1	200
Двигатель постоянного тока	800	1	800
Плата расширения	100	1	100
Набор гибких проводников	200	1	200
Блок питания 9В	300	1	300
Итого:			**4500**

Расчет общих материальных затрат представлен в формуле (30).

$$З_м = 1.2 \cdot 4500 = 5400. \quad (30)$$

5.3.4.2. Расчет затрат на специальное оборудование для научных (экспериментальных) работ

В данную статью включаются все затраты, связанные с приобретением специального оборудования (приборов, контрольно-измерительной аппаратуры, стендов, устройств и механизмов), необходимого для проведения работ. Определение стоимости спецоборудования производится по действующим прейскурантам, а в ряде случаев по договорной цене.

При приобретении спецоборудования необходимо учесть затраты по его доставке и монтажу в размере 15% от его цены. Стоимость оборудования, используемого при выполнении конкретного НТИ и имеющегося в данной научно-технической организации, учитывается в калькуляции в виде амортизационных отчислений. Коэффициент амортизации составил 0.025, так как длительность НТИ составляет три месяца.

Все расчеты по приобретению спецоборудования и оборудования, имеющегося в организации, но используемого в данной работе, приведены в таблице 14.

Таблица 14 – Расчет бюджета затрат на приобретение спецоборудования для научных работ

Наименование	**Цена за ед., руб.**	**Количество**	**Сумма, руб.**
Паяльная станция	8000	1	8000
Персональный компьютер	15000	1	15000
Мультиметр	1500	1	1500
Итого:			**25500**

Расчет затрат на специальное оборудование представлен в формуле (31).

$$З_{со} = 1.15 \cdot 25500 \cdot 0.025 = 733.13\ . \qquad (31)$$

5.3.4.3. Основная заработная плата исполнителей

В настоящую статью включается основная заработная плата научных и инженерно-технических работников, участвующих в выполнении работ по данной теме. Величина расходов по заработной плате определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и

тарифных ставок. В состав основной заработной платы включается премия, выплачиваемая ежемесячно из фонда заработной платы в размере 20 –30 % от тарифа или оклада.

Среднедневная заработная плата рассчитывается по формуле (32).

$$\text{Дневная з/плата} = \frac{\text{Месячный оклад}}{25{,}17 \text{ дней}}, \quad (32)$$

Расчеты затрат на основную заработную плату приведены в таблице 8. При расчете учитывалось, что в году 302 рабочих дня и, следовательно, в месяце 25,17 рабочих дня. Также необходимо принять во внимание районный коэффициент $K_{РК} = 0{,}3$.

Таблица 15 – Затраты на основную заработную плату

Исполнитель	**Оклад руб/мес.**	**Среднедневная ставка, руб./день**	**Затраты времени, дни**	**Фонд з/п без К, руб.**
Научный руководитель	33664	1337.47	49	65536.03
Студент	2421	96.19	98	9426.22
Итого:				**74962.25**

Расчет фонда заработной платы представлен в формуле (33).

$$З_{зп} = 1.3 \cdot 74692.25 = 97450.93 . \quad (33)$$

5.3.4.4. Отчисления во внебюджетные фонды

Величина отчислений во внебюджетные фонды определяется исходя из формулы (34).

$$З_{внеб} = k_{внеб} \times (З_{осн} + З_{доп}), \quad (34)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.).

На 2014 г. в соответствии с Федерального закона от 24.07.2009 №212-ФЗ установлен размер страховых взносов равный 30%. На основании пункта 1

ст.58 закона №212-ФЗ для учреждений осуществляющих образовательную и научную деятельность в 2014 году водится пониженная ставка – 27.1%.

В таблице 16 представлены данные об отчислениях во внебюджетные фонды.

Таблица 16 – Отчисления во внебюджетные фонды

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**
Руководитель	85196.83	–
Студент	12254.08	–
Коэффициент отчислений во внебюджетные фонды	$k_{внеб}$=30.2%	
Итого:	**29430.17**	

Расчет отчислений во внебюджетные фонды представлен в формуле (35).

$$З_{внеб} = 0.302 \cdot 97450.93 \ = 29430.17\ . \qquad (35)$$

5.3.4.5. Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергии, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по формуле (36).

$$З_{накл} = (\text{сумма статей } 1 \div 7) \times k_{нр}, \qquad (36)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы.

Величину коэффициента накладных расходов можно взять в размере 16%. Расчет накладных расходов предоставлен в формуле (37).

$$З_{накл} = (5400 + 733.13 + 97450.93 + 29430.17\ ) \cdot 0.16 = 21282.28. \quad (37)$$

5.3.4.6. Формирование бюджета затрат научно-исследовательского проекта

Рассчитанная величина затрат научно-исследовательской работы является основой для формирования бюджета затрат проекта, который при формировании договора с заказчиком защищается научной организацией в качестве нижнего предела затрат на разработку научно-технической продукции.

Определение бюджета затрат на научно-исследовательский проект по каждому варианту исполнения приведен в таблице 17.

Таблица 17 – Расчёт бюджета затрат НТИ

Наименование статьи	**Сумма, руб.**
Материальные затраты НТИ	5400
Затраты на специальное оборудование для научных (экспериментальных) работ	733.13
Затраты по основной заработной плате исполнителей темы	97450.93
Отчисления во внебюджетные фонды	29430.17
Накладные расходы	21282.28
Бюджет затрат НТИ	**154296.51**

Таким образом, суммарный бюджет затрат НТИ составил 154267 рублей.

5.3.3. Реестр рисков проекта

Идентифицированные риски проекта включают в себя возможные неопределенные события, которые могут возникнуть в проекте и вызвать последствия, которые повлекут за собой нежелательные эффекты.

Уровень риска может быть: высокий, средний или низкий в зависимости от вероятности наступления и степени влияния риска. Риски с наибольшей вероятностью наступления и высокой степенью влияния будут иметь высокий уровень.

Реестр рисков проекта предоставлен в таблице 11.

Таблица 18 – Реестр рисков проекта

№	Риск	Потенциальное воздействие	Вероятность наступления (1 - 5)	Влияние риска (1 - 5)	Уровень риска	Способы смягчения
1	Застревание в локальном минимуме при обучении	Некорректное упавление	3	5	высокий	Разработка новых методов и подстройка параметров
2	Нехватка времени	Не выполнение всех задач	3	5	высокий	Планирование задач
3	Неверный подбор оборудования	Некорректная работа объекта управления	2	3	средний	Разработка универсальных алгоритмов
4	Ошибки в написании ПО	Некорректная работа регулятора	3	3	средний	Постоянная отладка
5	Зашумленность датчика	Низкое качество регулирования	2	4	средний	Применение алгоритмов фильтрации
6	Недостаточная вычислительная мощность ПК	Невозможность обеспечения качественного управления	2	5	средний	Оптимизация алгоритмов
7	Поломка в процессе наладки	Невозможность дальнейшего исследования	1	5	низкий	Закупка резервных компонентов

Таким образом было выявлено два риска высокого уровня, четыре риска среднего уровня и один риск низкого уровня. Для каждого риска были разработаны способы смягчения отрицательного влияния на проект.

5.4. Определение ресурсосберегающей, финансовой, бюджетной, социальной и экономической эффективности исследования

Результатом проведенного исследования является система управления двигателем постоянного тока, основанная на нейроном регуляторе. Потенциальные потребителя ми разработки являются образовательные учреждения и, в меньшей степени, компании ориентированные на производство материальных благ.

Для получения точной количественной оценки экономической эффективности проекта необходим комплекс исследований, которые выходят далеко за рамки представленной работы.

5.4.1. Бюджет научно-технического исследования (НТИ)

Для оценки научной ценности, технической значимости и эффективности, планируемых и выполняемых НИР, используется метод бальных оценок. Бальная оценка заключается в том, что каждому фактору по принятой шкале присваивается определенное количество баллов. Обобщенную оценку рассчитывают по формуле (38).

$$K_{\text{НТУ}} = \sum_{i=1}^{3} R_i \cdot n_i, \quad (38)$$

где $K_{НТУ}$ – коэффициент научно-технического уровня; R_i – весовой коэффициент i-го признака научно-технического эффекта;

n_i – количественная оценка i-го признака научно-технического эффекта, в баллах.

Выбор весовых коэффициентов и балов приведен в таблице 19.

Таблица 19 – Оценка признаков научно технического эффекта НИР

Признак научно-технического эффекта НИР	**Характеристика признака НИОКР**	**R_i**	**Баллы**
Уровень новизны	Систематизируются и обобщаются сведения, определяются пути дальнейших исследований	0.3	6
Теоретический уровень	Разработка способа (алгоритм, программа мероприятий, устройство, вещество и т.п.)	0.4	6
Возможность реализации	Время реализации в течение первых лет	0.3	10

Расчет коэффициента научно-технического уровня представлен в формуле (39).

$$K_{\text{НТУ}} = 0.3 \cdot 6 + 0.4 \cdot 6 + 0.3 \cdot 10 = 7.2. \quad (39)$$

Таким образом, проект имеет средний уровень научно-технического эффекта, который обусловлен тем, что в ходе разработки обобщаются известные сведения, но применяется новые методы и алгоритмы.

6. СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

Аннотация

Определение понятия «Социальная ответственность» приведено в международном стандарте IC CSR-08260008000 «Социальная ответственность организации. Требования».

В соответствии с данным документом, социальная ответственность - ответственность организации за воздействие ее решений и деятельности на общество и окружающую среду через прозрачное и этичное поведение, которое:

- содействует устойчивому развитию, включая здоровье и благосостояние общества;
- учитывает ожидания заинтересованных сторон;
- соответствует применяемому законодательству и согласуется с международными нормами поведения;
- интегрировано в деятельность всей организации и применяется в ее взаимоотношениях [19].

Введение

В данной работе представлена разработка нейронного регулятора для управления двигателем постоянного тока. Учебный стенд представляет собой двигатель постоянного тока с возможностью динамического изменения момента на вал двигателя. Управление двигателем осуществляется посредством аппаратно-вычислительной платформы RobotDyn NANO V3 совместно с контроллером Ardumoto L298P. Питание контролера осуществляется от источника постоянного тока напряжением 9В. Нейронный регулятор реализован в виде Windows-приложения на языке C#. Обмен данными между микроконтроллером и ПК производится посредством интерфейса UART.

Разработка программной и аппаратной частей устройства проходит в аудитории 117-А 10-го корпуса ТПУ.

Результаты работы могут быть использованы в качестве учебного стенда для демонстрации работы нейронного регулятора, а так же применены в производстве для повышения качества управления двигателем постоянного тока.

6.1. Производственная безопасность

При разработке и наладке устройства могут возникнуть вредные и опасные факторы. Используя ГОСТ 12.0.003-2015 [20], можно выделить ряд факторов, приведенных в таблице 20. Так же выделены источники вредных и опасных факторов и нормативные документы, регламентирующие действие каждого фактора.

Таблица 20 – Опасные и вредные факторы при разработке устройства

Источник фактора	**Факторы**		**Нормативные документы**
	Вредные	**Опасные**	
1. Сборка и наладка аппаратной части 2. Разработка и тестирование программной части за ПК	1. Повышенный уровень шума 2. Повышенный уровень электромагнитных излучений 3. Отклонение показателей микроклимата 4. Недостаточная освещенность 5. Психофизиологические нагрузки	1. Поражение электрическим током 2. Статическое электричество 3. Короткое замыкание	1. СН 2.2.4/2.1.8.562–96 [21] 2. СанПиН 2.2.2/2.4.1340-03 [22] 3. СанПиН 2.2.4/2.1.8.055-96 [23] 4. СанПиН 2.2.4.548–96 [24] 5. СНиП 23-05-95 [25] 6. СанПиН 2.2.1/2.1.1.1278–03 [26] 7. ГОСТ Р 12.1.019-2009 ССБТ [27]

Для каждого фактора были спроектированы мероприятия по защите исследователя от их влияния.

6.1.1. Повышенный уровень шума

Объектом управления является коллекторный двигатель постоянного тока, который имеет высокий уровень шума. Так же на увеличение фонового шума влияет активная система охлаждения персонального компьютера и высокочастотный писк дросселей источников питания постоянного тока.

Шум является важным фактором, влияющим на организм человека, на его психическое состояние и, как следствие, на качество выполняемой им работы. Согласно СН 2.2.4/2.1.8.562–96 [21] уровень шума на рабочем месте, оборудованном персональным компьютером, не должен превышать 50 дБ. Уровень шума активной системы охлаждения используемого персонального компьютера в соответствии с руководством по эксплуатации составляет 25 дБ, что полностью соответствует нормам. Уровень шума используемого коллекторного двигателя постоянного тока доходит до 60 дБ, что превышает установленные нормативные значения.

Для минимизации влияния рассматриваемого вредного фактора можно использовать средства индивидуальной защиты такие как, пассивные или активные накладные наушники или беруши. Уровень шума в помещении может быть снижен путем использования звукопоглощающих панелей и потолков. Так же можно изолировать объект исследования от исследователя, так как не требуется прямой визуальный контакт.

6.1.2. Повышенный уровень электромагнитных излучений

Основным источником электромагнитных излучений при разработке устройства является персональный компьютер. Электромагнитное поле, которое создается персональным компьютером, имеет сложный спектральный состав в диапазоне частот от 0 Гц до 1000 МГц.

Требования к допустимым уровням электромагнитных излучений и времени воздействия на человека, изложены в СанПиН 2.2.2/2.4.1340-03 [4] и СанПиН 2.2.4/2.1.8.055-96 [13].

Допустимые уровни напряженности электромагнитного поля персонального компьютера в соответствии с СанПиН 2.2.2/2.4.1340-03 приведены в таблице 21.

Таблица 21 – Допустимые уровни напряженности электромагнитных полей по СанПиН 2.2.2/2.4.1340-03

Параметры воздействия, частота излучения	**Допустимые значения**
Статическое поле	20 000 В/м
На расстоянии 50 см вокруг - диапазон частот 5Гц – 2кГц - диапазон частот 2 – 400 кГц	 25 В/м 2,5 В/м
Переменное поле на расстоянии 50 см вокруг	0,25 А/м
Магнитная индукция не более - диапазон частот 5 Гц – 2кГц - диапазон частот 2 – 400 кГц	 250 нТл 25 нТл
Поверхностный электростатический потенциал не более	500 В

Согласно СанПиН 2.2.4/2.1.8.055-96 энергетическая экспозиция электромагнитного излучения в диапазоне частот 30 кГц - 300 МГц определяется как произведение квадрата напряженности электрического или магнитного поля на время воздействия на человека. Энергетическая экспозиция за рабочий день не должна превышать значений, указанных в таблице 22.

Таблица 22 – предельно допустимые значения энергетической экспозиции по СанПиН 2.2.4/2.1.8.055-96

Диапазоны частот	**Предельно допустимая энергетическая экспозиция**	
	По электрической составляющей, $(В/м)^2 \times ч$	**По магнитной составляющей, $(А/м)^2 \times ч$**
30 кГц - 3 МГц	20000,0	200,0
3 - 30 МГц	7000,0	Не разработаны
30 - 50 МГц	800,0	0,72
50 - 300 МГц	800,0	Не разработаны

Длительное воздействие электромагнитного поля на организм человека может привести к дыхательной, нервной и сердечнососудистой систем, головным болям, утомляемости. Для обеспечения меньшего уровня электромагнитного излучение использован жидкокристаллический монитор. Необходимо чтобы компьютер был заземлен, а так же необходимо по возможности сокращать время работы за компьютером.

Рабочие места аудитории 117-А, удовлетворяют всем перечисленным требованиям: используются современные жидкокристаллические мониторы, расположение ПК соответствуют СанПиН 2.2.2/2.4.1340-03, в рабочем графике предусмотрены кратковременные перерывы.

6.1.3. Отклонение показателей микроклимата

Разработка программного обеспечения устройства производится за персональным компьютером. Согласно СанПиН 2.2.4.548–96 [23] работа инженера-программиста относится к категории легких работ (А1). Категория А1 относится к работам с интенсивностью энергозатрат до 120 ккал/ч производимые сидя и сопровождающиеся незначительным физическим напряжением.

В соответствии с СанПиН 2.2.4.548–96, показателями, характеризующими микроклимат в производственных помещениях, являются:

- температура воздуха;
- температура поверхностей;
- относительная влажность воздуха;
- скорость движения воздуха;
- интенсивность теплового облучения.

Оптимальный микроклимат на рабочем месте обеспечивает ощущение теплового комфорта в течение работы при минимальном напряжении механизмов терморегуляции человека, не вызывает отклонений состояния здоровья, обеспечивает условия для высокого уровня работоспособности и является предпочтительным на рабочем месте.

Допустимые значения показателей микроклимата для категории А1 приведены в таблице 23.

Таблица 23 – Допустимые значения показателей микроклимата по СанПиН 2.2.4.548–96

Период года	**Температура воздуха, °С**	**Температура поверхностей, °С**	**Относительная влажность воздуха, %**	**Скорость движения воздуха, м/с**
Холодный	22-24	21-25	60-40	0,1
Теплый	23-25	22-26	60-40	0,1

Допустимые значения показателей обеспечиваются с помощью систем отопления, вентиляции и кондиционирования.

Вентиляция может осуществляться естественным и механическим путем. В помещения, оснащенные персональными компьютерами, должны подаваться достаточные объемы свежего воздуха, нормы которых приведены в таблице 24.

Таблица 24 – Нормы подачи свежего воздуха по СанПиН 2.2.4.548–96

Характеристика помещения	**Объемный расход подаваемого в помещение, свежего воздуха $м^3$ /на одного человека в час**
объем до 20 $м^3$ на человека	Не менее 30
объем 20…40 $м^3$ на человека	Не менее 20
более 40 $м^3$ на человека	Естественная вентиляция

Разработка устройства происходит в помещении, в котором имеется естественная вентиляция, при которой воздух поступает и удаляется через окна, двери и щели. При таком типе вентиляции воздух, поступающий в помещение, не проходит предварительную очистку и нагрев. В рассматриваемом помещении не выполняется требование относительно объема воздуха на одного человека, поэтому необходимо применение механической вентиляции.

6.1.4. Недостаточная освещенность

Требования к освещению рабочих мест, оборудованных персональными компьютерами, определяются характером зрительной работы сотрудников [9]. Особенность таких рабочих мест заключается в необходимости одновременной работы с разными информационными носителями: на бумаге и на экране монитора. Экранное изображение в отличие от бумажного является светящимся, что оказывает воздействие на зрительную работоспособность и утомляемость. Дополнительной нагрузкой на органы зрения служит необходимость постоянной адаптации при переносе взгляда с экрана монитора на бумажный носитель.

Сложные зрительные задачи часто сочетаются с необходимостью анализа поступающей информации, принятием решением в условиях с ограничением по времени и недопустимости ошибок, что приводит к психофизическому и эмоциональному напряжению человека. Основной причиной физического дискомфорта у сотрудников, работающих за персональным компьютером, являются неоптимальные условия рабочего места, значительную роль в этом играет освещение.

Освещение помещений с персональными компьютерами характеризуется следующими требованиями:

- обеспечение необходимых уровней освещенности в горизонтальной плоскости в зоне бумажного носителя и клавиатуры;
- исключение засветки изображение на экране монитора путем ограничения освещенности вертикальной плоскости экрана монитора;
- обеспечение надлежащего распределения яркости в центральном поле зрения пользователя;
- снижение прямой и отраженной блескости;
- ограничение глубины пульсации освещенности.

Требования к освещению рабочих мест, оборудованных персональными компьютерами, изложены в документах: СанПиН 2.2.2/2.4.1340-03 [4], СНиП 23-05-95 [9] и СанПиН 2.2.1/2.1.1.1278–03 [26].

Для общего освещения помещений следует использовать лампы со световой отдачей не менее 55 лм/Вт. Для освещения помещений, оборудованных персональными компьютерами, следует применять систему общего освещения. Также допускается применение комбинированного освещения с целью дополнительного освещения бумажного носителя при исключении засветки от экрана монитора.

Произведем расчет искусственного освещения рабочей аудитории. Размеры аудитории: длина $a = 9$ м, ширина $b = 6$ м, $h = 3{,}5$ м. Целью данных расчетов является оценка соответствия искусственного освещения существующим рекомендациям. В аудитории предусмотрено общее равномерное освещение, для которого применяются люминесцентные лампы типа ЛБ (лампы белого цвета), для которых используются светильники типа ОД - 2-30 (длинной 933 мм, шириной 204 мм).

Согласно рекомендациям [27], оптимальное размещение светильников в помещении определяется следующими геометрическими параметрами помещения :

H – высота помещения (3,5 м);

$h_c = H - h_n$- расстояние светильников от перекрытия ;

h_n – высота светильника над полом, высота подвеса;

h_p – высота рабочей поверхности над полом;

$h = h_n - h_p$ – расчетная высота, высота светильника над рабочей поверхностью;

L – расстояние между соседними светильниками или рядами;

l – расстояние от крайних светильников или рядов до стены (согласно рекомендациям, данным в [26] принимается равенство $l = L/3$).

Также учитывается критерий интегральный критерий оптимальности расположения светильников в помещении λ (оптимальность заключается в компромиссе стоимости оборудования и его обслуживания и равномерности производимого света). Согласно [27], для обозначенного типа светильников $\lambda = 1{,}4$.

Для светильников типа ОД - 2-30 и заданного размера помещения характерны следующие значения обозначенных параметров: $h_n = 3{,}5$м; $h_\text{с} = 0$; $h_\text{р} = 1{,}5$м; $h = 2$м; $L = \lambda \cdot h = 2{,}8$м.

Так как помещение прямоугольное, расстояние до крайних рядов светильников различны: $l_a = L/3 = 0{,}93$м; $l_b = (b - L - w)/2 = 1{,}39$м, где w – суммарная ширина предполагаемых рядов светильников (0,408м).

На основе данных расчетов рекомендуется разместить светильники в два ряда, в каждом из которых нужно установить 6 светильников типа ОД - 2 мощностью 30 Вт. План размещения светильников для рассматриваемого помещения представлен на рисунке 65. Разрывы между светильниками в ряду составят 0,30 м. Учитывая, что в каждом светильнике установлено по две лампы, общее число ламп в помещении составит 24.

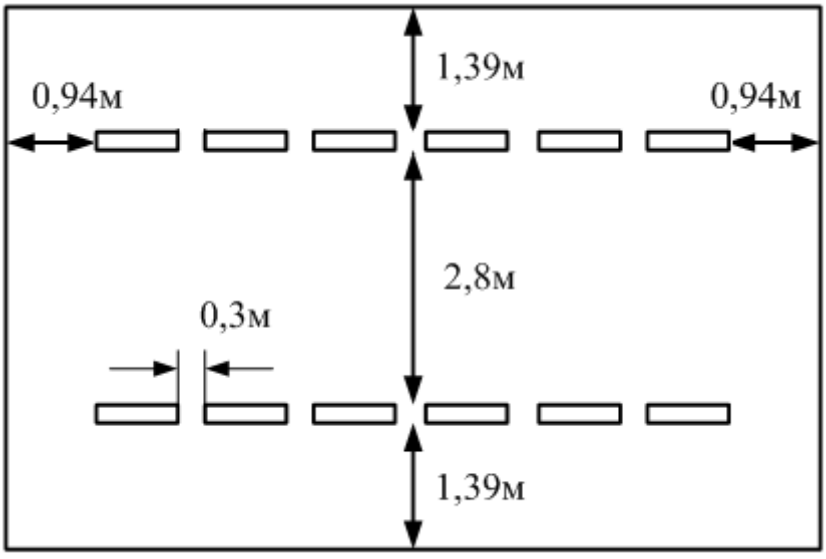


Рисунок 65 – План размещения светильников в помещении

Расчет общего равномерного освещения выполняется методом коэффициента светового потока, учитывающим световой поток, отраженный от потолка и стен [9]. Световой поток группы люминесцентных ламп светильника определяется по формуле (40).

$$F = \frac{E_{\text{н}} * S * K_{\text{З}} * Z}{n * \eta}, \quad (40)$$

где $E_{\text{н}}$ - нормируемая минимальная освещенность (для данной категории помещений $E_{\text{н}}$ - 300 лк);

S - площадь освещаемого помещения (54 м^2);

$K_{\text{З}}$ - коэффициент запаса, учитывающий загрязнение светильника, запыленность ($К_З$ = 1,5 - для помещений с малым выделением пыли);

Z - коэффициент неравномерности освещения (для люминесцентных ламп берется равным 1,1);

n - число ламп ($n = 24$);

η - коэффициент использования светового потока - показывает, какая часть светового потока ламп попадает на рабочую поверхность.

Степень использования светового потока ламп в помещении зависит как от геометрии помещения (площади (S), линейных размеров (a, b), высоты уровня светильников над рабочей поверхностью (h)), так и от коэффициентов отражения стен p_c и потолка $p_{\text{п}}$. Учет геометрии помещения осуществляется, с помощью индекса помещения, вычисляемого по формуле (41)

$$i = \frac{S}{h * (a + b)}. \quad (41)$$

Учитывая, параметры помещения, приведенные выше - $i = 1{,}8$м. Коэффициенты отражения p_c и p_n имеют следующие значения [27]: p_c = 70% (стены свежепобеленные с окнами закрытыми шторами); $p_{\text{п}}$. = 70% (потолок свежепопобеленный). Согласно таблице коэффициентов использования светового потока с люминесцентными лампами приведенной в [9] имеем η = 53%.

Используя формулу (1) получим $F = 2101$ лк. Таким образом, исходя из известного типа лампы, а также вычисленного необходимого светового потока, может быть подобрана стандартная лампа, удовлетворяющая требованиям освещенности помещения. Согласно [27], ближайшей подходящей стандартной лампой является - ЛБ 30 Вт с потоком 2180 лк. Напряжение сети 220 В, напряжение на лампе 104 В

По результатам вышеприведенного анализа, может быть сделан вывод, что интенсивность искусственного освещения удовлетворяет требованиям, предъявляемым к помещениям данного типа. Фактическое расположение светильников, а также световая характеристика ламп и их номинальная мощность соответствуют приводимым в [27] рекомендациям по подбору, комплектации и расположению светильников. Рассматриваемая аудитория имеет крупное окно, занимающее значительную часть площади стены, таким образом, освещение может считаться комбинированным. Для улучшения использования светового потока, окно снабжено занавесками в качестве которых выступают белые жалюзи. В целом, организация освещения в рабочей аудитории является удовлетворительной.

6.1.5. Психофизиологические вредные факторы

Большое значение в обеспечении комфортных условий труда имеет устранение влияние психофизиологических вредных факторов. В общем, под психофизиологическими вредными производственными факторами понимают все возможные физические и нервно-психические перегрузки, которым подвержен работник. В работе инженера-программиста к таковым могут быть отнесены, прежде всего, нервно-психические перегрузки: умственное перенапряжение, перенапряжение зрительных анализаторов, эмоциональные перегрузки.

Снижение влияние данных факторов осуществляется путем создания комфортных условий труда, регламентируется в СанПиН 2.2.2/2.4.1340-03, и предполагает выполнение следующих условий:

- оптимальная организация рабочего места;
- правильная организация освещения;
- удовлетворительные с точки зрения производственной санитарии условия.

Что в свою очередь предполагает проведение следующих мероприятий:

- мероприятия по защите от вредного воздействия излучений;
- мероприятия по снижению шума;
- мероприятия по обеспечению достаточной освещенности рабочего помещения, в том числе рабочего места;
- мероприятия по организации рабочего места исполнителя.

Таким образом, меры, принятые в рассмотренных выше разделах, позволяют констатировать наличие созданной благоприятной рабочей среды.

6.1.6. Электробезопасность

К числу опасных производственных факторов, характерных для рабочего места инженера-программиста, относится электрический ток. Степень опасного воздействия на человека электрического тока зависит от рода и величины напряжения и тока, а также от частоты электрического тока, пути тока через тело человека, продолжительности его воздействия на организм человека, условий внешней среды. В частности, электрический ток, протекая через тело человека, производит термическое, механическое и световое воздействие (электролитическое разложение жидкости, в том числе и крови, судорожное сокращение мышц, разрыв тканей и поражение глаз).

Общие требования к электробезопасности для рабочего места инженера-программиста регламентируются в ГОСТ Р 12.1.019-2009 ССБТ [28], лаборатория 117-А 10-го корпуса ТПУ относится к категории помещений без повышенной опасности, т.к. ее можно охарактеризовать как сухое

(относительная влажность воздуха не превышает 60%), беспыльное помещение с нормальной температурой воздуха (+24^{0}C), с изолирующими сухими бетонными полами, покрытыми линолеумом. Единственным типом технологического электрооборудования присутствующим в рабочем помещении является вычислительная техника (персональные компьютеры). Таким образом, все применяемое для работы электрооборудование имеет напряжение питания до 1000 В и малое выходное напряжение преобразовательных элементов (например, блока питания ПК - 5-25 В). В связи свыше обозначенными обстоятельствами, в целом, уровень электрической опасности рабочего помещения можно расценивать как низкий.

Во время нормального режима работы оборудования опасность электропоражения крайне мала, однако, возможны аварийные режимы работы, когда происходит случайное электрическое соединение частей оборудования, находящегося под напряжением с заземленными конструкциями.

Поражение человека электрическим током может произойти в следующих случаях:

- при прикосновении к токоведущим частям во время ремонта ПЭВМ;
- при однофазном (однополюсном) прикосновении неизолированного от земли человека к неизолированным токоведущим частям электроустановок, находящихся под напряжением;
- при прикосновении к нетоковедущим частям, находящимся под напряжением, то есть в случае нарушения изоляции;
- при соприкосновении с полом и стенами, оказавшимися под напряжением;
- при возможном коротком замыкании в высоковольтных блоках: блоке питания, блоке развертки монитора.

Так же в ходе эксплуатации возникает опасность возникновения короткого замыкания, обусловленная нарушением изоляции или неправильной коммутацией электрических цепей. Короткое замыкание может привести к

возникновению пожара а так же выделению токсичных веществ при горении изоляции.

Стоит учесть возможность поражения статическим электричеством. Статическое электричество образуется в результате трения двух диэлектриков друг о друга или диэлектриков о металлы. При этом на трущихся веществах могут накапливаться электрические заряды, которые легко стекают на землю, если тело является проводником электричества и оно заземлено. В рассматриваемых условиях не накапливаются значимые статические заряды.

Мероприятия по обеспечению электробезопасности сводятся к правильному размещению оборудования и применению технических средств защиты. К основным техническим средствам защиты от поражения электрическим током относятся:

- изоляция токопроводящих частей;
- защитное заземление;
- зануление;
- защитное отключение;
- предупредительная сигнализация и блокировки.

Аудитория, в котором проводятся работы, полностью соответствует требованиям электробезопасности, приведенным в ГОСТ Р 12.1.019-2009 ССБТ.

6.2. Экологическая безопасность

Согласно ГОСТ 30772-2001 [29] к отходам относятся остатки продуктов или дополнительный продукт, образующиеся в процессе или по завершении определенной деятельности и не используемые в непосредственной связи с этой деятельностью.

В данной работе выявлены следующие источники загрязнения окружающей среды:

- коллекторные двигатели постоянного тока;
- комплектующие персонального компьютера;

- лампы осветительных приборов.

Их составные части требуют специальной утилизации, поэтому эти источники загрязнения окружающей среды необходимо утилизировать по истечении срока службы.

Под утилизацией отходов понимается деятельность, связанная с использованием отходов на этапах их технологического цикла, и/или обеспечение повторного (вторичного) использования или переработки списанных изделий [28].

Перед утилизацией металлические составные части необходимо отсортировать по видам металла, удалить неметаллические части. Утилизация ламп осветительных приборов производится в специальных пунктах приема на утилизацию. Сложность утилизации ламп обусловлено содержанием в них паров ртути. Лампы, подлежащие утилизации, должны храниться в герметичных тарах во избежание отравления тяжелыми металлами в случае повреждения лампы.

6.3. Безопасность в чрезвычайных ситуациях

Наиболее вероятным чрезвычайными ситуациями при разработке устройства являются пожар на рабочем месте. Потенциальное возникновение пожара связано с возможным накоплением токоведущей пыли внутри компьютера, что может привести к короткому замыканию, возгоранию пыли и, если не будет не принято никаких мер, распространению пожара. В связи с возможной угрозой возникновения пожара был разработан план действий согласно с ГОСТ 12.1.004–91 ССБТ [30]:

- в случае обнаружения возгорания необходимо сообщить руководителю и попытаться потушить очаг возгорания своими силами с помощью средств первичного пожаротушения такими как: огнетушитель (порошковый, углекислотный);
- в случае если потушить очаг возгорания не удается, привести в действие ручной пожарный извещатель;

- немедленно сообщить о чрезвычайной ситуации в пожарную охрану по телефону 01 (сотовый 010), назвать адрес объекта, место и причины возникновения пожара;
- принять меры по эвакуации людей, материальных ценностей;
- приступить к тушению пожара, отключив электроэнергию;
- встретить подразделения пожарной охраны и, при необходимости, оказать помощь при выборе наилучшего пути для подхода к очагу пожара.

План эвакуации предоставлен на рисунке 66.

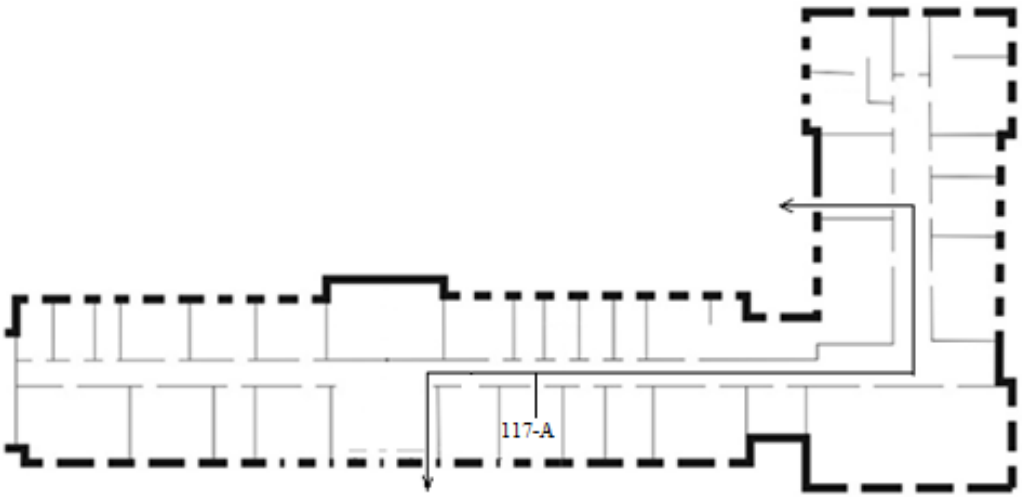


Рисунок 66 – План эвакуации при пожаре и других ЧС из помещений учебного корпуса №10, пр. Ленина, 2, 1-й этаж

6.4. Организационные вопросы обеспечения безопасности

6.4.1. Охрана труда

Согласно ФЗ-197 от 30.12.2001 [31], охрана труда - система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия.

Правовые мероприятия по охране труда заключаются в создании системы правовых норм, устанавливающих стандарты безопасных и здоровых

условий труда и правовых средств по обеспечению их соблюдения. Эта система правовых норм основывается на Конституции РФ и включает законы, подзаконные нормативные акты, а также локальные нормативные акты, принимаемые в конкретных организациях.

Социально-экономические мероприятия по охране труда включают меры государственного стимулирования работодателей по повышению уровня охраны труда, установление компенсаций и льгот при выполнении работ во вредных и опасных условиях труда, защиту отдельных, наименее социально защищенных категорий работников, обязательное социальное страхование и выплату компенсаций при возникновении профессиональных заболеваний и производственных травмах.

Организационно-технические мероприятия по охране труда заключаются в создании системы управления охраной труда – единого комплекса взаимосвязанных и взаимодействующих между собой элементов, устанавливающих политику и цели в области охраны труда в конкретной организации и процедуры по достижению этих целей.

Санитарно-гигиенические мероприятия по охране труда заключаются в проведении работ, направленных на снижение уровня воздействия на работников вредных и опасных производственных факторов с целью обеспечения благоприятных условий труда и предотвращения профессиональных заболеваний.

Лечебно-профилактические мероприятия по охране труда включают организацию предварительных, периодических и внеочередных медицинских осмотров, обязательных психиатрических освидетельствований работников, выдачу молока и лечебно-профилактического питания.

Реабилитационные мероприятия по охране труда заключаются в осуществлении комплекса мер, направленных на восстановление здоровья и трудоспособности работников, пострадавших в результате несчастного случая на производстве и профессиональных заболеваний.

Условия труда в НИИ ТПУ полностью соответствуют трудовому кодексу Российской Федерации [31].

6.4.2. Эргономические требования к рабочему месту инженера-программиста

Разработка программного обеспечения устройства производится за персональным компьютером.

Общие эргономические требования для рабочих мест при выполнении работ сидя регламентируются ГОСТ 12.2.032-78 [32]. 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. Зоны выполнения ручных операций и размещения органов управления представлены на рисунке 67, где:

1 - зона для размещения наиболее важных и очень часто используемых органов управления (оптимальная зона моторного поля);

2 - зона для размещения часто используемых органов управления (зона легкой досягаемости моторного поля);

3 - зона для размещения редко используемых органов управления (зона досягаемости моторного поля).

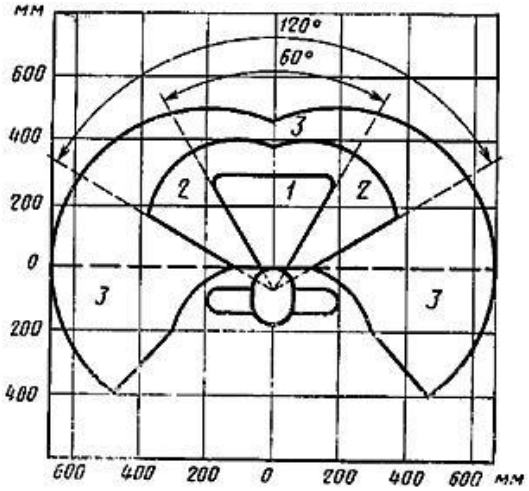


Рисунок 67 – Зоны для выполнения ручных операций и размещения органов управления

Рабочие места, оборудованные персональными компьютерами, должны располагаться по отношению к световым проемам таким образом, чтобы естественный свет падал с боковой стороны, преимущественно слева.

Расстояние между боковыми поверхностями мониторов должно составлять не менее 1,2 м, расстояние между экраном монитора и задней частью другого монитора – не менее 2 м.

Рабочий стол может быть любой конструкции, которая отвечает современным требованиям эргономики и позволяет удобно разместить на рабочей поверхности оборудование с учетом его количества, размеров и характера выполняемой работы. Целесообразно применение столов, имеющих отдельную от основной столешницы специальную рабочую поверхность для размещения клавиатуры. В случае, когда используется стол с нерегулируемой высотой рабочей поверхности, его высота должна быть в пределах от 680 до 800 мм. Глубина рабочей поверхности стола должна составлять 800 мм, ширина – 1600 мм. 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, на уровне вытянутых ног – не менее 650 мм.

Конструкция рабочего стула или кресла должна обеспечивать поддержание рациональной рабочей позы работника и позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины. Рабочий стул или кресло должны быть подъемно-поворотным, 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.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю, или на специальной поверхности, отделенной от основной столешницы.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм [22].

Рабочие места аудитории 117-А 10-го корпуса ТПУ полностью соответствуют представленным рекомендациям.

6.5. Заключение

В ходе исследования выявлены вредные и опасные факторы, на основы которых составлен перечень необходимых мер противодействия и минимизации влияния с учетом нормативной документации. Так же реализован комплекс мер по противодействию загрязнению окружающей среды, который состоит в правильной и своевременной утилизации отходов. Приведен перечень действий, необходимых при возникновении возможной чрезвычайной ситуации – пожара. Рассмотрены организационные аспекты обеспечения безопасности.

По итогам исследования рабочее место и объект исследования полностью соответствуют понятиям, изложенным в стандарте IC CSR-08260008000.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был разработан и собран учебный стенд по управлению бесколлекторным двигателем постоянного тока с возможностью динамического изменения нагрузки на валу. Так же разработана программа, реализующая генерацию обучающей выборки, обучение нейронной сети и синтез нейронного регулятора.

В ходе работы была изучена возможность применения нейронных сетей для управления динамическими объектами, в частности, для бесколлекторного двигателя постоянного тока. Лучшие результаты показал многомодульный нейрорегулятор с использованием ПИ-регуляторов. В тоже время, в сравнении с классическим ПИД-регулятором, изменения в качестве управления не существенны. Была выявлена высокая сложность процедуры синтеза многомодульного нейрорегулятора, заключающаяся в подробном испытании объекта во всех возможных состояниях, что не всегда возможно. Программа нейронного регулятора выполняется на персональном компьютере, что значительно сокращает число ее применений в практических целях.

В тоже время разработанный стенд и простота программного обеспечения могут служить наглядным пособием для изучения структуры и принципов работы нейронных сетей. Реализованные методы записи и считывания обученных нейронных сетей в файлы, а также представление процесса обучения в виде наглядного графика ошибок в реальном времени позволяют использовать полученную программу как инструмент по разработке и исследованию собственных структур нейронных регуляторов. Стоит отметить, что разработанные алгоритмы идентификации состояния объекта могут быть использованы и для других объектов, что потенциально позволяет применять разработанную программу как часть системы управления реальным производственным объектом или как часть SCADA–системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кулянин Е. М. Нейронные сети: история развития и перспективы применения // Научно-методический электронный журнал «Концепт». – 2015. – Т. 13. – С. 2646–2650. – [Электронный ресурс]: http://e-koncept.ru/2015/85530.htm
2. Нейронная сеть Хопфилда – [Электронный ресурс]: https://habr.com/post/301406/
3. Нейросеть для определения лиц, встроенная в смартфон – [Электронный ресурс]: https://apptractor.ru/develop/neyroset-dlya-opredeleniya-lits-vstroennaya-v-smartfon.html
4. Фёдорова Е.А., Линкова М.А. Прогнозирование курса валюты с помощью нейронных сетей // Финансовая аналитика: проблемы и решения. 2013. №11.
5. W. Yonghui, M. Schuster Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation – [Электронный ресурс]: https://ai.google/research/pubs/pub45610
6. B. Widrow Pattern Recognition and Adaptive Control // Proceedings of the IRE-AIEE Joint Automatic Control Conference, pp.19-26, August 1962.
7. Чернодуб А. Н. , Дзюба Д. А. Обзор методов нейроуправления // Проблемы программирования. — 2011. — No 2. — С. 79-94.
8. Цвенгер И.Г., Низамов И.Р. Применение нейросетевых регуляторов в системах управления электроприводами // Вестник Казанского технологического университета. 2017. №8.
9. Зубков Е. В., Галиуллин Л. А. Гибридная нейронная сеть для моделирования режимов испытаний двигателей внутреннего сгорания // Научно-технические ведомости СПбГПУ. 2011. №1 (117).
10. Есилевский В. С., Кузнецов В. Н., Панов В. П. Управление насосными агрегатами К. Н. С. С помощью систем нечетко-нейронного управления // Вестник ВГТУ. 2012. №9.
11. Бураков М. В., Шишлаков В. Ф. Адаптивное управление двигателем постоянного тока // Известия Самарского научного центра РАН. 2016. №4-3.

12. Коротков А. В Регулирование скорости двигателя постоянного тока с использованием нейросети // Публикации сборников кафедры ЕАПУ. 2010.
13. Физиология нервной деятельности человека – [Электронный ресурс]: http://hi-intel.ru/302/107.html
14. Хайкин С. Нейронные сети: полный курс / Хайкин С.; пер. с англ. – [2-е изд., испр.]. – М.: Вильямс, 2006. – 1102 с.
15. Управление бесколлекторным двигателем по сигналам обратной ЭДС – [Электронный ресурс]: https://habr.com/post/390469/ IC CSR-08260008000 «Социальная ответственность организации. Требования»
16. RobotDyn Nano: схема, описание – [Электронный ресурс]: https://robotdyn.com/nano-v3-atmega-328-usb-ttl-ch340g-micro-usb.html
17. SunnySky X2216 KV880 Brushless Motor – [Электронный ресурс]: https://www.fpvmodel.com/sunnysky-x2216-kv880-motor_g280.html
18. Регулятор хода (ESC) HobbyWing Skywalker 40A-UBEC – [Электронный ресурс]: https://www.flexinnovations.com/product-p/ftvhwbq8004b.htm
19. ГОСТ 12.0.003-2015 «Система стандартов по безопасности труда. Опасные и вредные производственные факторы. Классификация».
20. СН 2.2.4/2.1.8.562–96. «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки».
21. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».
22. СанПиН 2.2.4/2.1.8.055-96 . «Электромагнитные излучения радиочастотного диапазона (ЭМИ РЧ)»
23. СанПиН 2.2.4.548–96. «Гигиенические требовния к микроклимату производственных помещений».
24. СНиП 23-05-95 «Естественное и искусственное освещение».
25. СанПиН 2.2.1/2.1.1.1278–03. «Гигиенические требования к естественному, искусственному и совмещённому освещению жилых и общественных зданий».

26. Расчет искусственного освещения. Методические указания к выполнению индивидуальных заданий для студентов дневного и заочного обучения всех специальностей. - Томск: Изд. ТПУ, 2008. - 12 с
27. ГОСТ Р 12.1.019-2009 ССБТ. «Электробезопасность. Общие требования и номенклатура видов защиты».
28. ГОСТ 30772-2001 «Ресурсосбережение. Обращение с отходами. Термины и определения».
29. ГОСТ 12.1.004–91 ССБТ. «Пожарная безопасность. Общие требования».
30. «Трудовой кодекс Российской Федерации» от 30.12.2001 N 197-ФЗ (ред. от 05.02.2018)
31. ГОСТ 12.2.032-78. «Рабочее место при выполнении работ сидя. Общие эргономические требования».

ПРИЛОЖЕНИЕ А

(обязательное)

1. LITERATURE REVIEW

2. OBJECTS AND METHODS OF RESEARCH

Студент:

Группа	ФИО	Подпись	Дата
8АМ61	Усольцев Денис Вячеславович		

Консультант отделения автоматизации и робототехники

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Яковлева Елена Максимовна	к.т.н.		

Консультант – лингвист отделения иностранных языков:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Шепетовский Денис Владимирович			

1. LITERATURE REVIEW

First artificial neural network works are dated by 1940 and connected with exploring in the field of human brain studies. First known math model of artificial neural network was created by neurolinguist Walter Pitts and neuropsychologist Warren McCulloch. Their network was working with binary numbers and was able to train on examples. First hardware network was made by Frank Rosenblatt in 1958. Neurocomputer «Mark-I» was aimed to perform the sign-recognition task. It used photo elements to read letters on special cards [1].

But neural networks models of early time had significant limits. They were not able to operate with non-line-separabable functions. Such problem could be solved with using of multi-layer perceptron, but there weren't known methods to train such a net during that time. That fact slowed down neural network research until 1982, when John Hopfield found a way to exchange information between neurons [2].

Nowadays neural networks are widespread because of information technology breakthrough. Main application of neural networks is data analysis and processing. The most popular using case of neural networks is image recognition. For example neural nets are used for face identification on smart phones [3]. Neural nets also are used for search of time-based patterns, for example, it predicts price currencies [4]. Much known companies are using neural nets on their products. Google found application of neural nets on their machine translator Google Translator. That system is capable to translate from over a hundred languages to English and back again [5].

Neural nets also found their application in tasks of automate control. It's used as a neural controller for dynamic objects. First known idea of neural controller are connected with Bernard Widrow and it is dated by 1964. Adaptive controller, designed by Widrow, corrects its parameters depending on object behavior patterns that are recognized by neural net.

Main task to solve with dynamic objects control is synthesis of inverse object model. Difficulties are connected with compete cause-and-effect relationships at

behavior and common non-linear nature of dynamic objects. Neural networks make possible to partially solve this problem by training on real object behavior examples.

There are different neural networks were used for automate control. For example single-layer perceptron or Kohonen networks. Best-known results were obtained with use of multi-layer perceptron with tapped delay line [7]. There are different architectures was designed for neural control. They could be classified into direct methods, when neural network directly influences on object, and indirect methods, when neural net is used to solve one partial task of control. Most interesting methods are common inverse neural control and multi-module neural control. First method is using neural network as an inverse model of object to obtain desired control quality. Second one is using a set of neural networks that consists of pairs of direct and inverted neuroemulators. Every pair is designed to control object on its define state.

There are different applications that are designed on scientific papers. The article [8] reads the opportunity of controlling an electrodrive with help of adaptive PID-controller. The coefficients of controller are tuned by neural network. The designed system has high control quality and could adapt to continuous object parameters. On the other hand, procedure of synthesis is quite complete and trained controller is not useful for another object.

The article [9] presents application of neural network for the creation of explosion engine model. Author notices high validity of designed model and opportunity of neural networks for generalization.

The article [10] contains description of neural net for waterpump station control. The system minimizes power consumption. Author notices ability of neural nets to find complete rules and patterns in object behavior.

The article [11] presents implementation of adaptive controller for direct current motor. Neural network is seeking for optimal adaptation speed. Author notices significant improvement of control quality.

The article [12] describes direct current motor speed control system. Neural nets are used for rotor current identification. Author notices high complexity of trainset generation.

Reviewed articles contain description of neural network application for control quality improvement, but they're not researching for direct application of neural networks for dynamic control.

With using of [7], multi-layer perceptron was chosen as a neural network. That selection is defined by simple perceptron's math model and effective training methods. Multi-module neural network model was chosen as architecture of controller. Selection is defined by non stationary state of control object for investigated control range. Brushless direct current motor is considered as black box model due to chosen methods.

2. OBJECTS AND METHODS OF RESEARCH

2.5. Artificial neural network

Artificial neural network (ANN) makes a research trend in cybernetics that describes modeling of biological neural networks for tasks of artificial intelligent and automated control. Neural network consists of cross-connected set of neurons.

Neuron is a basic element of neural network and it is capable to store, process and transport information. Biological neuron consists of cell body, a large set of dendrites and a single axon. Dendrites receive information from other neurons. Axon sends signals to other neurons or organs. Location between axon and dendrites are called synapse [13]. Structure of neuron is presented in Figure 1.

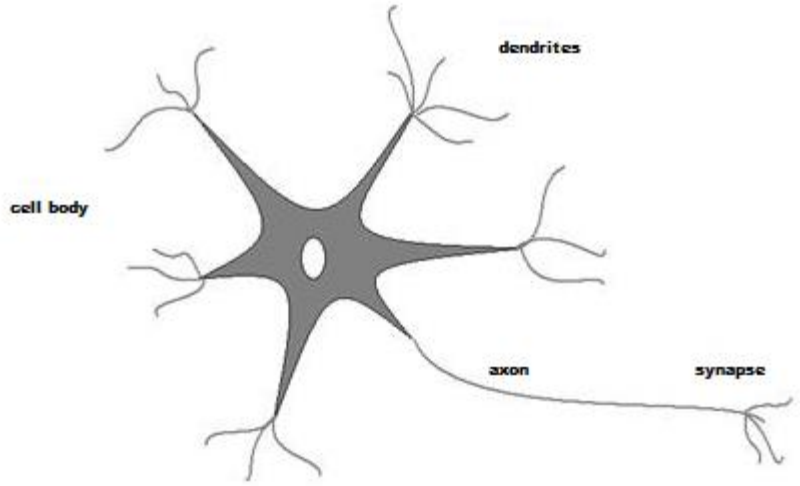


Figure 1 – Biological neuron

Neuron model uses weighted inputs as dendrites. Input of activation function is presented by weighted sum of inputs and bias. Activation function simulates neuron excitation. Output of activation function represents axon. Structure of artificial neuron is presented in Figure 2.

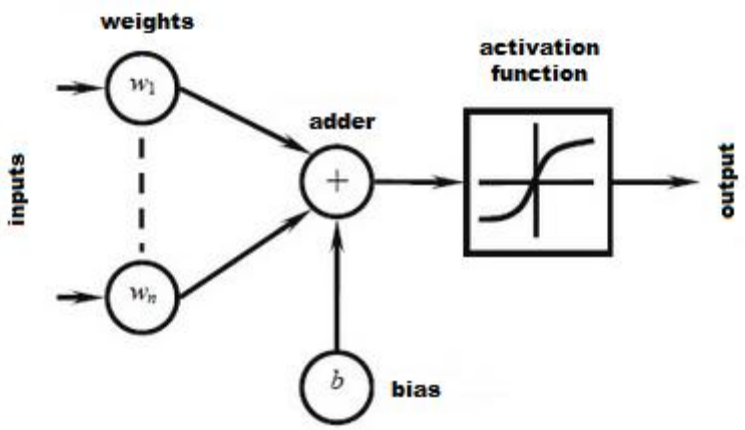


Figure 2 – Neuron model

Mathematical description of such a model could be represented as (1):

$$y = F\left(b + \sum_{i=0}^{n} w_i \cdot x_i\right), \quad (1)$$

where F(x) is an activation function;

b – start bias;

n – number of inputs;

w_i – weight of i-th neuron.

Activation function is usually a non-linear function with saturation. Different functions are presented in Figure 3.

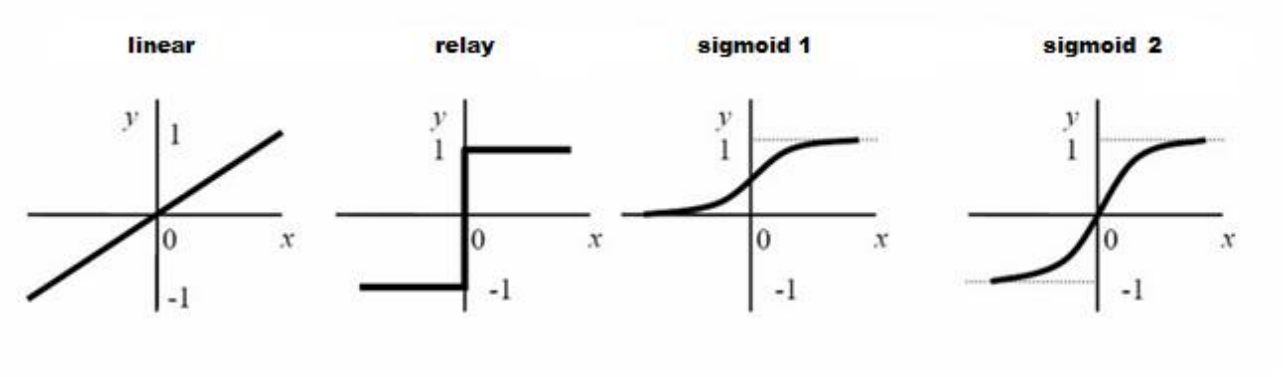


Figure 3 – Activation functions

Sigmoid function is more common to biological model, because it has different sensitivity to the signals of different level [13]. Function is more sensitive to

signals until saturation level. But it's not sensitive to signals that are above the level of saturation.

One of the first neural networks models was Rosenblatt perceptron. Basic perceptron consists of elements of three different types. Input sensors elements receive and transform signals and send them to the analyzing elements. Analyzing element are the formal models of neuron. After non-linear transformation signals proceed to the receptors. Receptors forming feedback signal of neural net. Rosenblatt perceptron is shown in Figure 4.

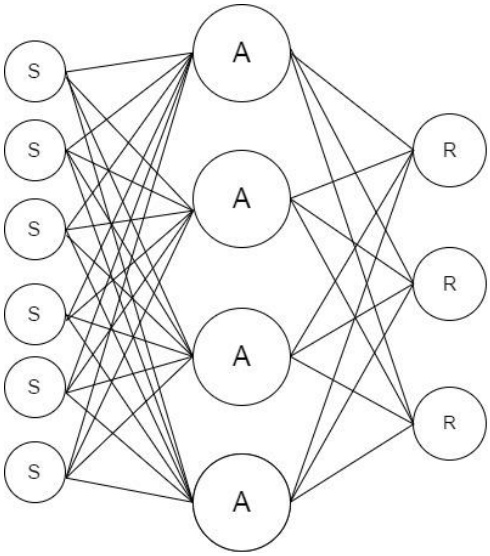


Figure 4 – Rosenblatt perceptron model

Rosenblatt perceptron has significant limit. It couldn't operate with non-linear-separable functions. This fact makes not possible to operate dynamic object with using of Rosenblatt perceptron [1]. Example of non-line-separable function could be presented by exclusive OR function (2):

$$Y = A \oplus B. \tag{2}$$

Graphical representation of exclusive OR is shown on picture 5.

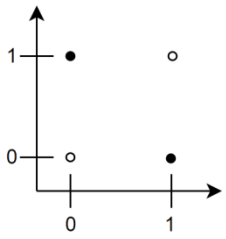


Figure 5 –Graphical representation of exclusive OR

It’s not possible to draw such a line that could separate positive and negative functions values.

Multi-layer Rosenblatt perceptron is used to solve this problem. On base levels of such network signals transforms to organize line-separable sets that could be processed by next layers. Structure of multi-layer Rosenblatt perceptron is presented in Figure 6.

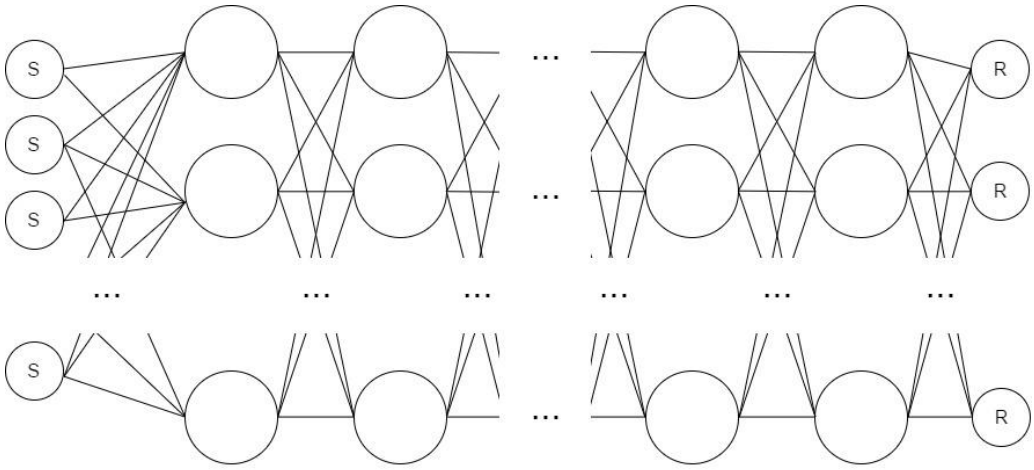


Figure 6 – Multi-layer Rosenblatt perceptron

Sensor neurons are called input layer. Reacting neurons are called output layer. Neurons between them form hidden layers.

There are many different types of neural networks such as Kohonen networks or convolutional networks [1], but they are used commonly under the tasks of classification and clusterization and not usable for the task of automated control.

2.6. Training

To make neural network work it should be trained under defined trainset. Training process of neural network consists of changing connections weights to achieve defined criteria. Training algorithms could be divided into algorithms with and without the teacher.

In training algorithms without teacher trainset only consists of input signals. Neural network learning to group same signals. In other worlds neural network solves task of clusterization.

Secound group of algorithms is using some abstract teacher who knows the right answers. Neural network processes input signals and if the output reaction is incorrect, the network changes its weights.

Most common algorithm of training of multi-layer Rosenblatt is backpropagation algorithm. It's training with the teacher algorithm [14]. Algorithm solves optimization task of global network error minimum search in the dimensions of weights by method of gradient descent.

Examine this algorithm for the multi-layer perceptron with one hidden layer. Representation of such perceptron is shown in Figure 7.

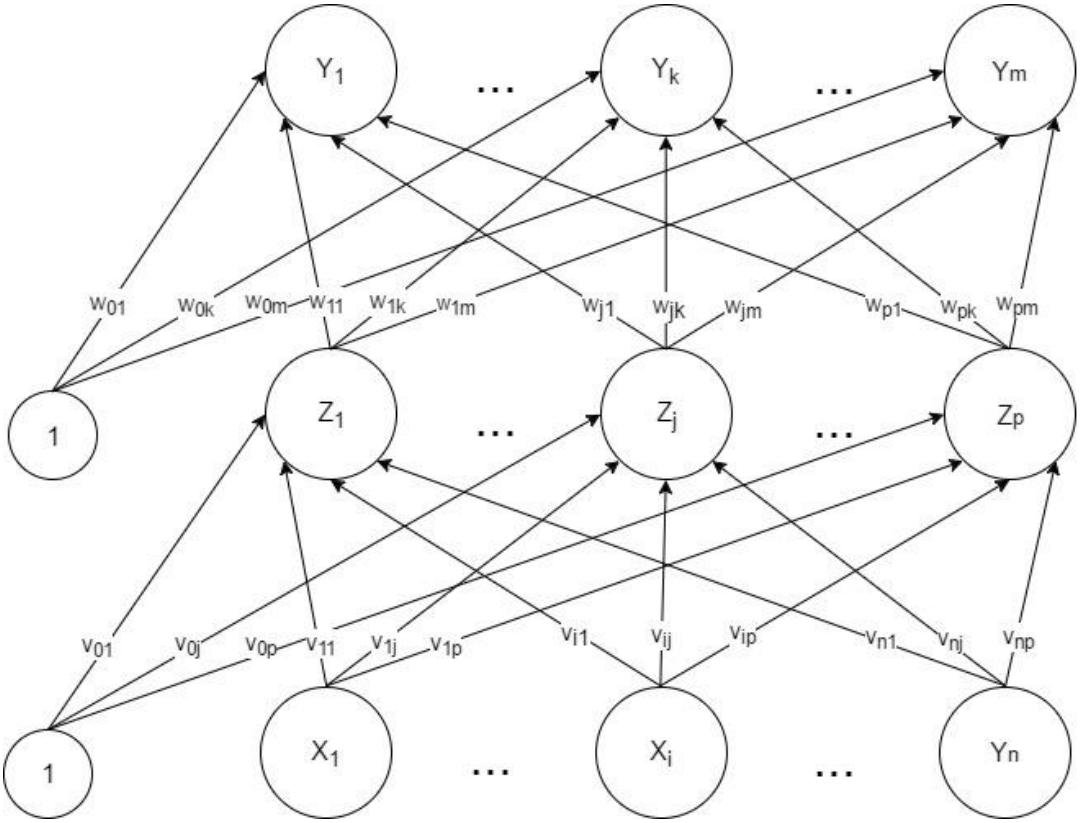


Figure 7 – Three-layer Rosenblatt perceptron

Let's introduce the following notation:

x – input vector of trainset with dimension of n;

y – output vector of desired reactions with dimension of m;

X_i – neuron at input layer with index i;

Z_j – neuron at hidden layer with index j;

Y_k – neuron at output layer with index k;

n – number of neurons at input layer;

p – number of neurons at hidden layer;

m – number of neurons at output layer;

v_{0j} – initial bias of neuron Z_j;

w_{0k} – initial bias of neuron Y_k;

v_{ij} – weight of the link between X_i и Z_j;

w_{jk} – weight of the link between Z_j и Y_k;

α – learning rate coefficient.

Activation function represented by binary sigmoid (3):

$$F(x) = \frac{1}{1 + e^{-x}}. \quad (3)$$

Binary sigmoid acts quite the same as biological neuron and has simple derivative calculation (4):

$$F'^{(x)} = F(x) \cdot (\, 1 - F(x)\,). \quad (4)$$

Firstly all weights of network should be initialized by random values between -1 and 1.

Every neuron of input layer receives input vector and sends it to the hidden layer (5). Every neuron of hidden layer sums weighted inputs with bias and apply activation function (6).

$$X_{j\,out} = x_j. \quad (5)$$

$$Z_{j\,out} = F(\, v_{0j} + \sum_{i=0}^{n} X_{i\,out} \cdot v_{ij}\,). \quad (6)$$

Every neuron of output layer calculate output in the same way (7):

$$Y_{k\,out} = F(\, w_{0k} + \sum_{j=0}^{p} Z_{j\,out} \cdot w_{jk}\,). \quad (7)$$

Mean square error function could be used for the estimation of neural network (8):

$$E = \frac{1}{2}\sum_{k=1}^{m} (Y_{k\,out} - y_k)^2. \quad (8)$$

To minimize error of neural network, optimization task of minimum search should be solved. Partial derivative of network error for the network output could be found as (9):

$$\frac{\partial E}{\partial Y_{k\,out}} = Y_{k\,out} - y_k = E_Y. \tag{9}$$

Introduce notation E_Y – output layer error. Partial derivative of network error for the hidden layer output could be found as (10). Introduce notation E_Z – hidden layer error:

$$\frac{\partial E}{\partial Z_{j\,out}} = \frac{1}{2}\sum_{k=1}^{m}(F(\,w_{0k} + \sum_{j=1}^{p} Z_{j\,out}\cdot w_{jk}\,) - y_k)^2 =$$

$$= \sum_{k=1}^{m}\left(E_Y \cdot F'\left(\,w_{0k} + \sum_{j=1}^{p} Z_{j\,out}\cdot w_{jk}\,\right)\cdot\, w_{jk}\right) = E_Z \tag{10}$$

Partial derivative of network error for the every weight of output layer could be found as (11). Introduce notation σ_{jk} – weight error from j-th neuron of hidden layer to k-th neuron of output layer:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial Y_{k\,out}}\cdot\frac{\partial Y_{k\,out}}{\partial w_{jk}} = E_Y \cdot F'\left(\,w_{0k} + \sum_{j=1}^{p} Z_{j\,out}\cdot w_{jk}\,\right)\cdot\, Z_{j\,out} = \sigma_{jk} \tag{11}$$

Partial derivative of network error for the every weight of hidden layer could be found as (12). Introduce notation δ_{ij} – weight error from i-th neuron of input layer to j-th neuron of hidden layer:

$$\frac{\partial E}{v_{ij}} = \frac{\partial E}{\partial Z_{j\,out}}\cdot\frac{\partial Z_{j\,out}}{v_{ij}} = E_Z \cdot F'\left(v_{0j} + \sum_{i=1}^{n} X_{i\,out}\cdot v_{ij}\,\right)\cdot\, X_{i\,out} = \delta_{ij} \tag{12}$$

In such way, for error minimization gradient step should be done to the direction of gradient descent for the every partial derivative by changing weights between neurons.

Correct all weights between neurons (13, 14):

$$v_{ij} = v_{ij} + \alpha\cdot\delta_{ij} \tag{13}$$

$$w_{jk} = w_{jk} + \alpha\cdot\sigma_{jk} \tag{14}$$

Steps of algorithm are applying for the every pair of trainset elements. One iteration of processing all trainset is called epoch. Total error for the epoch could be find as total mean square error (15):

$$E_{epoch} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{2}\sum_{k=1}^{m}(Y_{k\,out} - y_k)^2\right)}, \quad (15)$$

where M – number of trainset elements. Training continues until reaching one of criteria. For example learning time had been up, lasted for a defined number of iteration or total error for epoch crossed defined threshold.

At the process of training error of networks spreads backwards through the weights, such a description that is contained in the name of algorithm. Algorithm has low convergence speed and could stack in the local minimum. During the training weight could reach high levels. In such situation derivative of activation function become equal to zero that paralyze learning [14]. There are no direct methods to counter act this problems. In real examples of neural networks sets of approximate heuristics are under use, but it's effective only for defined tasks.

2.7. Neural controllers structures

ANN should be used when identification of dynamic object is impossible or connected with serious complicities. In such way controlled object represented by a model of black box, for which set of input signals and reactions for limited range is known. Neural network has benefits in that class of tasks, because it could interpolate, extrapolate and find common patterns.

In design process of neural controllers capability to reproduce learned reactions and adapt to continuous parameters of object are used. Non-linear activation functions in structure of ANN could be used as linearization of control object or as a non-linear controller.

ANN couldn't be used with trainsets where elements not presented in time. Neural network lost connections and couldn't achieve training criteria. For

representation of dynamic parameters of control object autoregressive model could be used. It could be represented by tapped delay line (TDL) (14):

$$TDL = \begin{pmatrix} y[n] \\ y[n-1] \\ y[n-2] \\ \dots \\ y[n-N] \end{pmatrix}. \quad (16)$$

All neural control methods could be divided into direct and indirect methods. In direct method neural network directly influences at control object. In indirect methods neural network solves one defined task for the part of control system. Neural controller also could be divided into single-module and multi-module. Single-module controllers uses only one neural network. Multi-module controllers are represented by combination of neural networks [7].

4.1.5. Imitation neural control

Imitation neural controller emulates behavior of a common controller. During the training of ANN input of trainset is input control signal and delayed outputs of control object. Output of trainset is output of common controller. Neural controller is training for different states of control object. Structure scheme of imitation controller training is represented in Figure 6.

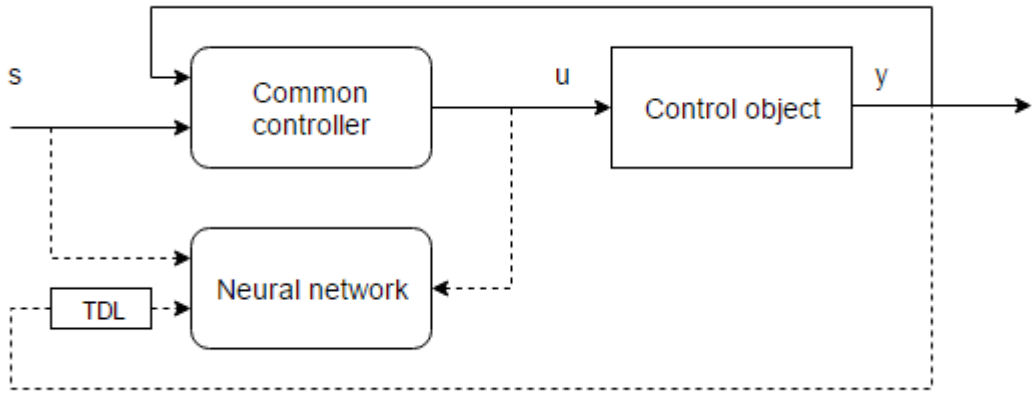
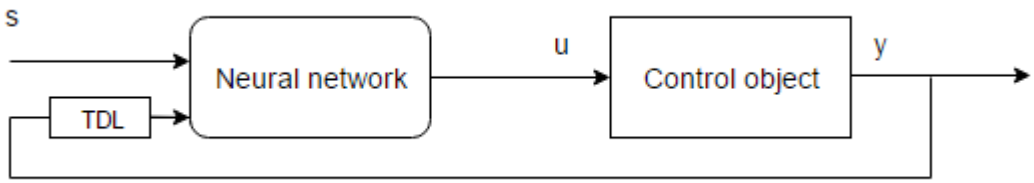


Figure 6 – Training of the imitation neural controller

Neural controller substitutes common controller after the training. Such a controller could learn different states of controlled object and continue controlling the

object without re-learning. Structure scheme of imitation controller in process of control is shown on picture 7.



Picture 7 –Imitation neural controller

Such a controller could not achieve better control quality than common controller at a defined state. Imitation neural controller could be used with dynamic object with continuous parameters.

4.1.6. Extended inverse neural control

In method of inverse neural control ANN represented as inverse model of control object. Such neural net called inverse neural emulator. During the training ANN receive current and delayed output of controlled object. Input of controlled object is output of trainset. Structure scheme of inverse neural controller training is shown in Figure 8.

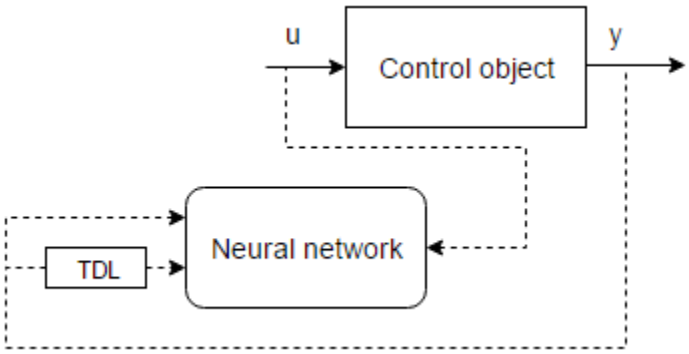


Figure 8 – Training of inverse neural controller

During the control inverse neural controller receives desired value of object output and delayed object output. Output of inverse neural controller proceeds to the object input. Structure scheme of inverse neural controller in process of control is shown in Figure 9.

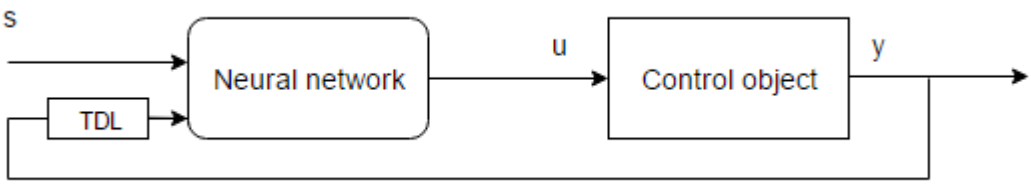


Figure 9 – Inverse neural control

According to the stabilization affection of feedback, controller has high level of control quality. But if input of object is changing discretely, training of neural network could be impossible.

4.1.7. Multi-module neural control with using of pairs direct and inverse neural emulators

In that method neural controller consists of pairs of direct and inverse neural emulators. Every pair is aimed to control at one defined state of controlled object.

Controlled object is fixed at one defined state and examined by random signals for direct neural emulator training. Direct neural emulator is learning to predict behavior of dynamic object by input signal and delayed outputs of object. Structure scheme of direct neural emulator training is shown in Figure 10.

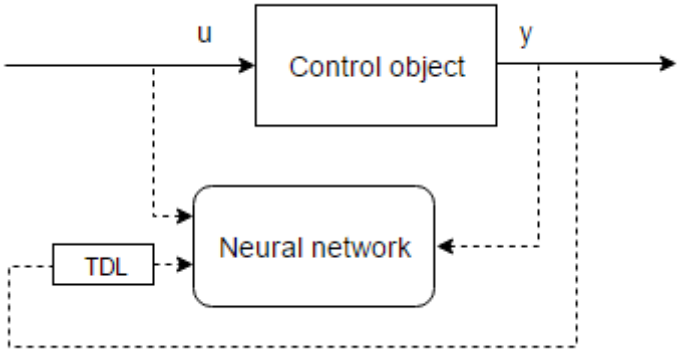


Figure 10 – Direct neural emulator training

Direct neural emulator calculates current output of dynamic object by input signal and delayed outputs of object in process of control. It's in contemplation that each pair of neural emulators is valid for defined state of dynamic object. If direct neural emulator correctly predicts dynamic of controlled object, that assigned inverse

emulator has high control quality. In addition, it is in contemplation that states of object is quite different. In process of control, system estimates validity of each pair of emulators and choose one optimal pair. Structure scheme of multi-module neural controller in process of control is shown in Figure 11.

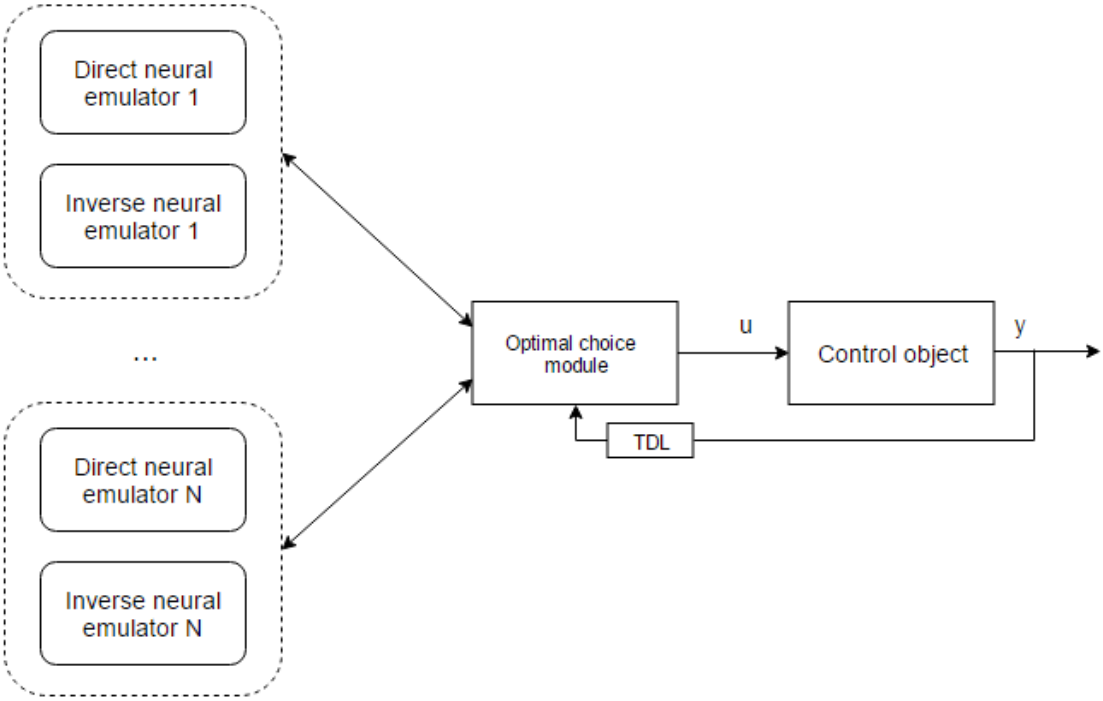


Figure 11 – Multi-module neural controller

Significant disadvantage of this system is not formalized procedure of trainset separation for different states of controlled object. Also at the moment of changing control module there is could be step changing of control signal.

2.8. Brushless direct current motor

Brushless direct current motor is a type of direct current motors. Example of brushless direct current motor is shown in Figure 12.

Figure 12 – Brushless direct current motor

Main difference between brushless and brush direct current motor is in mechanic commutator – part, that passes voltage to the rotor of brush direct current motor with help of brushes. Brush motor coils switches mechanically. It significantly decreases reliability of the system.

Brushless motor coils switches electronically with help of specialized controller. Complexity of brushless motor control is in the necessity of rotor position identification to know exactly which coils should be switched to voltage supplier. Brushless motor is synchronic motor and if frequency of rotor and stator magnetic field is not the same, motor could just stop. Problem of identification could be solved with the help of sensors, but it complicates structure of the motor. Also there are sensorless methods depending on measurement of electromotive force of coils self induction [13]. Commonly electronic speed controllers are used to implement brushless motors rotor position identification and control.

ПРИЛОЖЕНИЕ Б

(справочное)

Листинг программы микроконтроллера

```
#include <Servo.h>
Servo bldc;
const byte pinSW = 6;  // номер вывода, подключенный к SW
const byte pinDT = 7;  // номер вывода контроллера, подключенный к DT енкодера
```

```
unsigned long i;
int j;
unsigned long k;
bool curSW;
bool curDT;
bool prewSW;
bool prewDT;
unsigned long prev_time;
unsigned long sum_time;
unsigned long mes_time;
double rps; //обороты в секунду
String readStr; //последнее считаное значение
char readCh;// считываемый символ
String readBufer; //буфер получаемого значения
boolean bf; //флаг начала получаемого значения
boolean ef; //флаг конца получаемого значения
int max_bldc = 1500; //Максимальное значение ШИМ
int min_bldc = 1206;  //Минимальное значени ШИМ
bool ff=false; // флаг сработавшего фильтра
bool sf=false; //флаг схожести значений в выборке
double tf; // значение текущего такта
int cf; // количество отчетов в прошлом такте
int ik;//

void setup()
{
  pinMode (6,INPUT);
  pinMode (7,INPUT);
  //pinMode(11, OUTPUT);
  //digitalWrite(11, LOW);
  bldc.attach(9);    //Инициальзация
  Serial.begin (2000000);
}


void loop()
{
  // ! считывание в начале цикла
  // /*
  diRead(); //считывание дискретных входов
  portRead(); //считывание значений с последовательного порта
  // ! обороты в секунду с энкодера
  // /*
  if ((prewDT!=curDT)and(prewDT==false)) {i++;}
  if ((prewDT!=curDT)and(prewDT==true)) {i++;}
  unsigned long _time = micros();
  unsigned long _period = _time - prev_time;
  prev_time = _time;
  sum_time=sum_time+_period;
  j++;
  prewSW=curSW;
  prewDT=curDT;
  if (j==1000)
  {
    tf=((double)i/74)/((double)sum_time/1000000);
    mes_time=sum_time;  sum_time=0;
    if (cf==i) {ff=false; sf=true;} else {sf=false;}
    if ((cf!=i) and (!ff)) {ff=true;} else {rps=tf;}
    cf=i;
    i=0; j=0;
  }

  // ! управление bldc
```

```
  // /*
  if   (!curSW)   {bldc.write(map(readStr.toFloat(),   0,   (max_bldc-min_bldc),
min_bldc, max_bldc));} else {bldc.write(min_bldc);}
  //bldc.write(map(readStr.toFloat(),    0,    (max_bldc-min_bldc),    min_bldc,
max_bldc));
  // */

}

void diRead()
{
  curSW=digitalRead(6);
  curDT=digitalRead(7);
}

void portRead()
{
  if (Serial.available())
    {
      readCh=Serial.read();
      if (readCh=='{') {bf=true;}
      if (readCh=='}') {ef=true;}
      if ((bf)&&(!ef)&&(readCh!='{')) {readBufer+=readCh;}
    }
  if ((bf)&&(ef)) { readStr=readBufer; readBufer=""; bf=false; ef=false;}
}

void portWrite(String s)
{
  Serial.print('{');
  Serial.print(s);
  Serial.println('}');
}
```

ПРИЛОЖЕНИЕ В

(справочное)

Листинг программы Neural

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.IO.Ports;
using System.IO;
using System.Diagnostics;
using System.Runtime.Serialization.Formatters.Binary;


namespace TryWF
{

    public partial class Form1 : Form
    {
        //! определение переменных
        //
        public SerialPort Port;
        public Stopwatch StopWatch = new Stopwatch();
        public Stopwatch DbStopWatch = new Stopwatch();
        public Series xySet; // данные для графика переходных процессов не
отфильтрованные
        public Series xySete; // данные для графика переходных процессов
отфитрованные
        public Series xySet_m1; // данные для графика переходных процессов
эмулятор m1
        public Series xySet_m2; // данные для графика переходных процессов
эмулятор m2
        public Series xySet_m3; // данные для графика переходных процессов
эмулятор m3
        public Series xySet_m4; // данные для графика переходных процессов
эмулятор m4
        public Series xySet_m5; // данные для графика переходных процессов
эмулятор m5
        public Series errSet; // данные для графика ошибок обучения
        public Series emuSet; // даные для графика проверки обучения
восстановленного эмулятором сигнала
        public Series realSet; // данные для графика проверки обучения иходного
сигнала
        public Series stSet; // данные для графика состояния системы
        public int LoopPeriod = 50; // период выполнение Loop
        bool PermissionDraw = true; // разрешение на перерисовку графика
        public string ReadStr = "0"; // строка данных с COM-порта
        public double Time = 0; // время с начала работы программы
        public double Rps; // обороты в секунду с энкодера
        public double Re; // обороты в секунду, отфильтрованные экспоненциальным
фильтром
        public double R_m1; // обороты в секунды, нейроэмулятор m1
        public double R_m2; // обороты в секунды, нейроэмулятор m2
        public double R_m3; // обороты в секунды, нейроэмулятор m3
```

```
        public double R_m4; // обороты в секунды, нейроэмулятор m4
        public double R_m5; // обороты в секунды, нейроэмулятор m5
        public double Error_m1; // ошибка нейроэмулятора m1
        public double Error_m2; // ошибка нейроэмулятора m2
        public double Error_m3; // ошибка нейроэмулятора m3
        public double Error_m4; // ошибка нейроэмулятора m4
        public double Error_m5; // ошибка нейроэмулятора m5
        public double[] Error_m = new double[5]; // ошибка нейроэмуляторов
        public double[][] DelayedError; //массив ошибок нейроэмуляторов
        public int DelayErrDeep = 15; // глубина линии задержки ошибок
        public int SystemState = 1; // текущее состояние нагрузки
        public double ComTask = 4; // текущее задание для ДПТ
        double ActualSetpoint = 0; // уставка переходного процесса
        public double[] X; // обучающая выборка не преобразованная X
        public double[] Y; // обучающая выборка не преобразованная Y
        public  double[][]  Xe;  //  обучающая  выборка  преобразованная  X  для
нейроэмулятора
        public  double[][]  Ye;  //  обучающая  выборка  преобразованная  Y  для
нейроэмулятора
        public  double[][]  Xi;  //  обучающая  выборка  преобразованная  X  для
инверсного нейроэмулятора
        public  double[][]  Yi;  //  обучающая  выборка  преобразованная  Y  для
инверсного нейроэмулятора
        public Network EmuNet; // нейроэмулятор
        public Network InvNet; // инверсный нейроэмулятор
        public Network EmuNet_m1; // нейроэмулятор m1
        public Network EmuNet_m2; // нейроэмулятор m2
        public Network EmuNet_m3; // нейроэмулятор m3
        public Network EmuNet_m4; // нейроэмулятор m4
        public Network EmuNet_m5; // нейроэмулятор m5
        public Network InvNet_m0; // инверсный нейроэмулятор m0
        public Network InvNet_m1; // инверсный нейроэмулятор m1
        public Network InvNet_m2; // инверсный нейроэмулятор m2
        public Network InvNet_m3; // инверсный нейроэмулятор m3
        public Network InvNet_m4; // инверсный нейроэмулятор m4
        public Network InvNet_m5; // инверсный нейроэмулятор m5
        public PI PI_m1; // ПИ-регулятор модуля m1
        public PI PI_m2; // ПИ-регулятор модуля m2
        public PI PI_m3; // ПИ-регулятор модуля m3
        public PI PI_m4; // ПИ-регулятор модуля m4
        public PI PI_m5; // ПИ-регулятор модуля m5
        public  bool  NewAvailable  =  false;  //  наличие  данных,  доступных  для
чтения
        public bool WriteToFile = false; // разрешение записи в файл
        public double[] LearningErrors; // ошибки обучения в эпохе
        public double[] EmulatedTrainset; // эмулированная обучающая выборка
        public bool DoneLearning = false; // флаг завершения процесса обучения
        public bool Emulated = false; // флаг завершения проверки обучения
        public double[] DelayedInput; // вход для нейросетей
        public bool Start = true;
        public double IntegralSum=0;
        // ПИД-регулятор
        public double ProportionalCoef = 3;
        public double IntegralCoef = 0.3;
        public double DerivativeCoef = 0.31;
        public double SumIntegral = 0;
        public double PrevError = 0;
        public double CurrentError = 0;
        public double PIDOut = 0;
        public double ProportionalOut = 0;
        public double IntegralOut = 0;
        public double DerivativeOut = 0;
        // пути к файлам
```

```
        string FilePath = @"C:\Users\Nick\Desktop\Y(N)+X+X(N)=Y\Logs"; //путь к
файлу с логами
        string FileName = "ident_m1.txt"; //название файла
        //string FileName = "ident_inv_m0.txt"; //название файла
        string EmulatorName = "Emulator_m0"; // файл нейроэмулятора
        string InvertedName = "Inverted_m0"; // файл инверсного нейроэмулятора
        // настройки обучения
        public double LearningThreshold = 0.001; // порог ошибки при обучении
        public double EpochTreshold = 2000; // порог ошибки при обучении
        public double LearningRate = 0.7; // коэффициент скорости обучения
        public int DelayLineDeep = 10; // глубина линии задержки
        public int HiddenNeurons = 10; // колличество нейронов в скрытом слое
        public int Waiting = 40; // чувствительность стабилизации переходного
процесса
        public double ExpAlpha = 0.2; // постоянная экспоненциального фильтра
        // конфигурация запуска программы
        public bool Emulator = false; // флаг обучения нейроэмулятора
        public bool Inverse = false; // флаг обучения инверсного нейроэмулятора
        public bool DirectGen = false; // флаг генерации обучающей выборки для
нейроэмулятора
        public bool InverseGen = false; // флаг генерации обучающей выборки для
инверсного нейроэмулятора
        public bool NetControl = false; // флаг управления нейроконтроллером
        public bool PIDControl = false; // флаг управления ПИД-регулятором
        public bool CombinedControl = true; // флаг управления комбинированным
нейроконтроллером
        //

        public Form1() //! инициализация компонентов
        {
            //
            InitializePort();
            InitializeComponent();
            InitializeLoopTimer();
            InitializePlot();
            StopWatch.Start();
            DbStopWatch.Start();
            if (!(DirectGen ^ InverseGen)) ReadFromFile();
            if (!(DirectGen ^ InverseGen)) ToEmulatorTrainsetTransform();
            if (!(DirectGen ^ InverseGen)) ToInvertedTrainsetTransform();
            DelayedInput = new double[DelayLineDeep + 1];
            for (int i = 0; i < DelayLineDeep + 1; i++) DelayedInput[i] = 0;
            //
            DelayedError = new double[5][];
            for (int i = 0; i < 5; i++)
            {
                DelayedError[i] = new double[DelayErrDeep];
                for (int j= 0; j < DelayErrDeep; j++)
                {
                    DelayedError[i][j] = 0;
                }
            }
            //
            PI_m1 = new PI(4, 0.2);
            PI_m2 = new PI(3.5, 0.28);
            PI_m3 = new PI(2.5, 0.32);
            PI_m4 = new PI(2, 0.4);
            PI_m5 = new PI(2, 0.45);
            //
            // инициализация потоков
            System.Threading.Thread ThreadNet = new System.Threading.Thread(new
System.Threading.ParameterizedThreadStart(Net));
```

```
            System.Threading.Thread ThreadRead = new System.Threading.Thread(new
System.Threading.ParameterizedThreadStart(ComRead));
            System.Threading.Thread      ThreadTrainsetGeneration      =      new
System.Threading.Thread(new
System.Threading.ParameterizedThreadStart(TrainsetGen));
            if (!(DirectGen ^ InverseGen)) ThreadNet.Start();
            ThreadRead.Start();
            if ((DirectGen ^ InverseGen)) ThreadTrainsetGeneration.Start();
            //
        }

        public void Loop(object Sender, EventArgs e) //! основной цикл программы
        {
            // данные для графика
            Time += Timer();
            if (PermissionDraw)
            {
                if (DoneLearning)
                {
                    xySet_m1.Points.AddXY(Time, R_m1);
                    xySet_m2.Points.AddXY(Time, R_m2);
                    xySet_m3.Points.AddXY(Time, R_m3);
                    xySet_m4.Points.AddXY(Time, R_m4);
                    xySet_m5.Points.AddXY(Time, R_m5);
                    stSet.Points.AddXY(Time, SystemState);
                }
                //xySet.Points.AddXY(Time, Rps);
            }
            xySete.Points.AddXY(Time, Re);
            //
            if   ((!DoneLearning)&(!(DirectGen   ^   InverseGen))   &(Emulator   ^
Inverse))
            {
                errPlot.Series.Clear();
                errSet.Points.Clear();
                for (int i = 0; i < LearningErrors.Length; i++)
                {
                    if (Inverse) { errSet.Points.AddXY(i, LearningErrors[i]); }
                    else { errSet.Points.AddXY(i, LearningErrors[i] * 100); }
                }
                errPlot.Series.Add(errSet);
            }

            //
            if ((DoneLearning)&(!Emulated)&(!Inverse))
            {
                emuPlot.Series.Clear();
                emuSet.Points.Clear();
                for (int i = 0; i < Xe.Length; i++)
                    emuSet.Points.AddXY(i*0.065, EmulatedTrainset[i]*100);
                for (int i = 1; i < Xe.Length; i++)
                    realSet.Points.AddXY(i*0.065, Ye[i][0]*100);
                emuPlot.Series.Add(emuSet);
                emuPlot.Series.Add(realSet);
                Emulated = true;
            }
            //

            //

        }

        public void Net(object obj) //! инициализация и обучение нейросетей
```

```
        {
            //
            //обучение нейроэмулятора
            if (Emulator)
            {
                LearningErrors = new double[Xe.Length];
                EmuNet = new Network(Xe, Ye, HiddenNeurons, LearningRate);
                System.Threading.Thread.Sleep(1000);
                //while (EmuNet.SqrtError > LearningThreashold)
                while (EmuNet.Epochs < EpochTreshold)
                {
                    EmuNet.Train();
                    LearningErrors = EmuNet.LocalErrors;

                    if  (LearningError.InvokeRequired)  LearningError.Invoke(new
Action<string>((s) => LearningError.Text = s), EmuNet.SqrtError.ToString());
                    else LearningError.Text = EmuNet.SqrtError.ToString();

                    if          (Epoch.InvokeRequired)          Epoch.Invoke(new
Action<string>((s) => Epoch.Text = s), EmuNet.Epochs.ToString());
                    else Epoch.Text = EmuNet.Epochs.ToString();
                }
                NetToFile(EmuNet, EmulatorName);
            }
            //

            // обучение инверсного нейроэмулятора
            if (Inverse)
            {
                InvNet = new Network(Xi, Yi, HiddenNeurons, LearningRate);
                LearningErrors = new double[Xi.Length];
                System.Threading.Thread.Sleep(1000);
                LearningRate = 0.1;
                //while (InvNet.SqrtError > LearningThreashold)
                while (InvNet.Epochs < 60000)
                {
                    InvNet.Train();
                    LearningErrors = InvNet.LocalErrors;

                    if  (LearningError.InvokeRequired)  LearningError.Invoke(new
Action<string>((s) => LearningError.Text = s), InvNet.SqrtError.ToString());
                    else LearningError.Text = InvNet.SqrtError.ToString();

                    if          (Epoch.InvokeRequired)          Epoch.Invoke(new
Action<string>((s) => Epoch.Text = s), InvNet.Epochs.ToString());
                    else Epoch.Text = InvNet.Epochs.ToString();
                }
                NetToFile(InvNet, InvertedName);
            }
            //

            // извление обученых нейросетей из файлов
            EmuNet = NetFromFile(EmulatorName);
            InvNet = NetFromFile(InvertedName);
            EmuNet_m1 = NetFromFile("Emulator_m1");
            EmuNet_m2 = NetFromFile("Emulator_m2");
            EmuNet_m3 = NetFromFile("Emulator_m3");
            EmuNet_m4 = NetFromFile("Emulator_m4");
            EmuNet_m5 = NetFromFile("Emulator_m5");
            InvNet_m1 = NetFromFile("Inverted_m1");
            InvNet_m2 = NetFromFile("Inverted_m2");
            InvNet_m3 = NetFromFile("Inverted_m3");
            InvNet_m4 = NetFromFile("Inverted_m4");
```

```
            InvNet_m5 = NetFromFile("Inverted_m5");
            InvNet_m0 = NetFromFile("Inverted_m0");



                if (Time > 40)
                { PortWrite(0.ToString()); }

            }

            while (false)
            {
                if (Time < 10)
                {
                    ActualSetpoint = 7;
                }

                if ((Time > 10) & (Time < 20))
                {
                    ActualSetpoint = 11;
                }

                if ((Time > 20) & (Time < 30))
                {
                    ActualSetpoint = 6;
                }

                if ((Time > 30) & (Time < 40))
                {
                    ActualSetpoint = 12;
                }

                CurrentError = ActualSetpoint - Re;
                ProportionalOut = ProportionalCoef * CurrentError;
                IntegralOut = SumIntegral + IntegralCoef * CurrentError * 0.065;
                DerivativeOut = DerivativeCoef * (CurrentError - PrevError) /
0.065;
                PIDOut = ProportionalOut + IntegralOut + DerivativeOut;
                if (PIDOut >= 100) PIDOut = 100;
                if (PIDOut <= 0) PIDOut = 0;
                ComTask = PIDOut;
                PortWrite(ComTask.ToString());
                SumIntegral += IntegralCoef * CurrentError;
                PrevError = CurrentError;
                DelayedInput[DelayLineDeep] = ComTask / 50;

                DoubleToTextBox(ProportionalOut, Pt);
                DoubleToTextBox(IntegralOut, It);
                DoubleToTextBox(DerivativeOut, Dt);
                DoubleToTextBox(ComTask, ActualTask);

                using (StreamWriter outputFile = new
StreamWriter(Path.Combine(FilePath, "etp.txt"), true))
                {
                    outputFile.WriteLine(Time.ToString());
                }

                using (StreamWriter outputFile = new
StreamWriter(Path.Combine(FilePath, "eyp.txt"), true))
                {
                    outputFile.WriteLine(Re.ToString());
                }
```

```
                if (Time >40 )
                { PortWrite(0.ToString()); }

            }



            // восстановление данных по нейроэмулятору
            if (!Inverse)
            {
                EmulatedTrainset = new double[Xe.Length];
                for (int i = 0; i < DelayLineDeep; i++)
                {
                    EmuNet.Calculate(Xe[i]);
                    EmulatedTrainset[i] = EmuNet.OutputLayer.Outputs[0];
                }
                double[] cin = Xe[0];
                for (int i = DelayLineDeep; i < Xe.Length; i++)
                {
                    cin = Xe[i];
                    for (int j = 1; i < DelayLineDeep + 1; i++)
                    {
                        cin[j - 1] = EmulatedTrainset[i + DelayLineDeep - j];
                    }
                    EmuNet.Calculate(cin);
                    EmulatedTrainset[i] = EmuNet.OutputLayer.Outputs[0];
                }
            }

            //
            DoneLearning = true;
            //
        }

        public void TrainsetGen(object obj) //! генерация обучающей выборки
        {
            //
            bool Stabilized = false;
            int k = 0;
            double R;
            double Rm = 0;
            double[] DelayRps = new double[Waiting];
            void ClrDelay()
            {
                for (int q = 0; q < Waiting; q++)
                { DelayRps[q] = -99; }
            }
            void WaitForStabilization()
            {
                k = 0;

                Stabilized = false;
                ClrDelay();
                while (!Stabilized)
                {
                    string ReadSt = ComTask.ToString()+' '+k.ToString();
                    if          (toPLC.InvokeRequired)          toPLC.Invoke(new
Action<string>((s) => toPLC.Text = s), ReadSt);
                    else toPLC.Text = ReadSt;

                    if (k == Waiting - 1) { Stabilized = true; }
```

```
                    else
                    {
                        if (NewAvailable)
                        {
                            NewAvailable = false;
                            R = Re;
                            if (DirectGen)
                            {
                                using    (StreamWriter    outputFile    =    new
StreamWriter(Path.Combine(FilePath, FileName), true))
                                {
                                    outputFile.WriteLine(ComTask.ToString()  +  '
' + R.ToString());
                                }
                            }



                            for (int p = 0; p < k + 1; p++)
                            {
                                if (DelayRps[p] == -99) { DelayRps[p] = R; ++k;
p = 100; }
                                else
                                {
                                    if ((R >= 0.9 * DelayRps[p]) && (R <= 1.1 *
DelayRps[p])) { }
                                    else { k = 0; ClrDelay(); p = 100; }
                                }
                            }
                        }
                    }
                }
            }
            //
            if (InverseGen)
            {
                double prewVal = 0;


                ComTask = 9;
                PortWrite(ComTask.ToString());
                WaitForStabilization();

                for (int i = 0; i < 40; i++)
                {
                    PortWrite(10.ToString());
                    System.Threading.Thread.Sleep(2000);
                    ++ComTask;
                    PortWrite(ComTask.ToString());
                    WaitForStabilization();

                    Rm = 0;
                    for (int j = 0; j < Waiting/2-1; j++)
                    {
                        Rm += DelayRps[Waiting / 2 + j];
                    }
                    Rm = Rm / (Waiting/2 -1);

                        if (((prewVal * 1.02) < Rm) ^ ((prewVal * 0.98) > Rm))
                    {
                        using      (StreamWriter      outputFile      =      new
StreamWriter(Path.Combine(FilePath, FileName), true))
                        {
```

```
                            outputFile.WriteLine(ComTask.ToString()  +  '  '  +
Rm.ToString());
                        }
                        prewVal = Rm;
                    }

                }
                PortWrite((0).ToString());
            }

            if (DirectGen)
            {
                ComTask = 10;
                PortWrite(ComTask.ToString());
                WaitForStabilization();
                for (int i = 0; i < 8; i++)
                    for (int j = i+1; j < 9; j++)
                    {
                        ComTask =10 +  i * 10 + j;
                        PortWrite(ComTask.ToString());
                        WaitForStabilization();

                        ComTask =10 + (j * 10)+i;
                        PortWrite(ComTask.ToString());
                        WaitForStabilization();

                    }
                PortWrite((0).ToString());
            }
            //
        }

        public void ToEmulatorTrainsetTransform() //! трансформация обучающей
выборки для нейроэмулятора
        {
            //
            int k = X.Length - DelayLineDeep;
            Xe = new double[k][];
            Ye = new double[k][];
            for (int i = 0; i < k; i++)
            {
                Xe[i] = new double[DelayLineDeep + 1];
                Ye[i] = new double[1];
            }
            for (int i = 0; i < k; i++)
            {
                Ye[i][0] =  Y[i + DelayLineDeep] / 100;
                Xe[i][DelayLineDeep] = X[i + DelayLineDeep]/50;
                for (int j = 1; j < DelayLineDeep+1; j++)
                {Xe[i][j-1] = Y[i + DelayLineDeep - j]/100;}
            }
            //
        }

        public void ToInvertedTrainsetTransform() //! трансформация обучающей
выборки для инверсного нейроэмулятора
        {
            //
            int k = X.Length;
            Xi = new double[k][];
            Yi = new double[k][];
            for (int i = 0; i < k; i++)
            {
```

```
                Xi[i] = new double[1];
                Yi[i] = new double[1];
                Xi[i][0] = Y[i];
                Yi[i][0] = X[i]/100;
            }
            //
        }

        public void ComRead(object obj) //! считывание и реакция на данные с
последовательного порта
        {
            //
            while (true)
            {
                ReadStr = PortRead();
                if          (fromPLC.InvokeRequired)          fromPLC.Invoke(new
Action<string>((s) => fromPLC.Text = s), ReadStr);
                else fromPLC.Text = ReadStr;
                Rps                   =                   Double.Parse(ReadStr,
System.Globalization.CultureInfo.InvariantCulture);
                Re = Re * (1 - ExpAlpha) + ExpAlpha * Double.Parse(ReadStr,
System.Globalization.CultureInfo.InvariantCulture);
                NewAvailable = true;



                if (DoneLearning)
                {
                    for (int j = 0; j < DelayLineDeep-1; j++)
                    {
                        DelayedInput[DelayLineDeep        -        j-1]        =
DelayedInput[DelayLineDeep - j - 2];
                    }
                    DelayedInput[0] = Re/100;
                    EmuNet_m1.Calculate(DelayedInput);
                    EmuNet_m2.Calculate(DelayedInput);
                    EmuNet_m3.Calculate(DelayedInput);
                    EmuNet_m4.Calculate(DelayedInput);
                    EmuNet_m5.Calculate(DelayedInput);
                    R_m1 = EmuNet_m1.OutputLayer.Outputs[0] * 100;
                    R_m2 = EmuNet_m2.OutputLayer.Outputs[0] * 100;
                    R_m3 = EmuNet_m3.OutputLayer.Outputs[0] * 100;
                    R_m4 = EmuNet_m4.OutputLayer.Outputs[0] * 100;
                    R_m5 = EmuNet_m5.OutputLayer.Outputs[0] * 100;

                    Error_m1 = Math.Sqrt(Math.Pow(Re - R_m1, 2));
                    Error_m2 = Math.Sqrt(Math.Pow(Re - R_m2, 2));
                    Error_m3 = Math.Sqrt(Math.Pow(Re - R_m3, 2));
                    Error_m4 = Math.Sqrt(Math.Pow(Re - R_m4, 2));
                    Error_m5 = Math.Sqrt(Math.Pow(Re - R_m5, 2));

                    for (int i = 0; i < 5; i++)
                    {
                        for (int j = 0; j < DelayErrDeep - 1; j++)
                        {
                            DelayedError[i][DelayErrDeep    -    j    -    1]    =
DelayedError[i][DelayErrDeep - j - 2];
                        }
                    }
                    DelayedError[0][0] = Error_m1;
                    DelayedError[1][0] = Error_m2;
                    DelayedError[2][0] = Error_m3;
```

```
                    DelayedError[3][0] = Error_m4;
                    DelayedError[4][0] = Error_m5;
                    for (int i = 0; i < 5; i++)
                    {
                        Error_m[i] = 0;
                        for (int j = 0; j < DelayErrDeep - 1; j++)
                        {
                            Error_m[i] += DelayedError[i][j];
                        }
                    }

                    if ((Error_m[0] < Error_m[1]) & (Error_m[0] < Error_m[2]) &
(Error_m[0] < Error_m[3]) & (Error_m[0] < Error_m[4]))
                    {
                        if       (CurEmu.InvokeRequired)       CurEmu.Invoke(new
Action<string>((s) => CurEmu.Text = s), "m1");
                        else CurEmu.Text = "m1";
                        SystemState = 1;
                    }
                    if ((Error_m[1] < Error_m[0]) & (Error_m[1] < Error_m[2]) &
(Error_m[1] < Error_m[3]) & (Error_m[1] < Error_m[4]))
                    {
                        if       (CurEmu.InvokeRequired)       CurEmu.Invoke(new
Action<string>((s) => CurEmu.Text = s), "m2");
                        else CurEmu.Text = "m2";
                        SystemState = 2;
                    }
                    if ((Error_m[2] < Error_m[0]) & (Error_m[2] < Error_m[1]) &
(Error_m[2] < Error_m[3]) & (Error_m[2] < Error_m[4]))
                    {
                        if       (CurEmu.InvokeRequired)       CurEmu.Invoke(new
Action<string>((s) => CurEmu.Text = s), "m3");
                        else CurEmu.Text = "m3";
                        SystemState = 3;
                    }
                    if ((Error_m[3] < Error_m[0]) & (Error_m[3] < Error_m[1]) &
(Error_m[3] < Error_m[2]) & (Error_m[3] < Error_m[4]))
                    {
                        if       (CurEmu.InvokeRequired)       CurEmu.Invoke(new
Action<string>((s) => CurEmu.Text = s), "m4");
                        else CurEmu.Text = "m4";
                        SystemState = 4;
                    }
                    if ((Error_m[4] < Error_m[0]) & (Error_m[4] < Error_m[1]) &
(Error_m[4] < Error_m[2]) & (Error_m[4] < Error_m[3]))
                    {
                        if       (CurEmu.InvokeRequired)       CurEmu.Invoke(new
Action<string>((s) => CurEmu.Text = s), "m5");
                        else CurEmu.Text = "m5";
                        SystemState = 5;
                    }

                    if (Start) { ActualSetpoint = 6; Start = false; }


                    if (NetControl)
                    {
                        if (SystemState == 1)
                        {
                            InvNet_m1.Calculate(new  double[1]  {  ActualSetpoint
});
                            ComTask = InvNet_m1.OutputLayer.Outputs[0] * 100;
                            PortWrite(ComTask.ToString());
```

```
                        DelayedInput[DelayLineDeep] = ComTask / 50;
                    }
                    if (SystemState == 2)
                    {
                        InvNet_m2.Calculate(new  double[1]  {  ActualSetpoint
});
                        ComTask = InvNet_m2.OutputLayer.Outputs[0] * 100;
                        ComTask = InvNet_m2.OutputLayer.Outputs[0] * 100;
                        PortWrite(ComTask.ToString());
                        DelayedInput[DelayLineDeep] = ComTask / 50;
                    }
                    if (SystemState == 3)
                    {
                        InvNet_m3.Calculate(new  double[1]  {  ActualSetpoint
});
                        ComTask = InvNet_m3.OutputLayer.Outputs[0] * 100;
                        PortWrite(ComTask.ToString());
                        DelayedInput[DelayLineDeep] = ComTask / 50;
                    }
                    if (SystemState == 4)
                    {
                        InvNet_m4.Calculate(new  double[1]  {  ActualSetpoint
});
                        ComTask = InvNet_m4.OutputLayer.Outputs[0] * 100;
                        PortWrite(ComTask.ToString());
                        DelayedInput[DelayLineDeep] = ComTask / 50;
                    }
                    if (SystemState == 5)
                    {
                        InvNet_m5.Calculate(new  double[1]  {  ActualSetpoint
});
                        ComTask = InvNet_m5.OutputLayer.Outputs[0] * 100;
                        PortWrite(ComTask.ToString());
                        DelayedInput[DelayLineDeep] = ComTask / 50;
                    }

                }

                if (PIDControl)
                {
                    CurrentError = ActualSetpoint - Re;
                    ProportionalOut = ProportionalCoef * CurrentError;
                    IntegralOut = SumIntegral + IntegralCoef * CurrentError
* 0.065;
                    DerivativeOut   =   DerivativeCoef   *   (CurrentError   -
PrevError) / 0.065;
                    PIDOut = ProportionalOut + IntegralOut + DerivativeOut;
                    if (PIDOut >= 100) PIDOut = 100;
                    if (PIDOut <= 0) PIDOut = 0;
                    ComTask = PIDOut;
                    PortWrite(ComTask.ToString());
                    SumIntegral += IntegralCoef * CurrentError;
                    PrevError = CurrentError;
                    DelayedInput[DelayLineDeep] = ComTask / 50;

                    DoubleToTextBox(ProportionalOut, Pt);
                    DoubleToTextBox(IntegralOut, It);
                    DoubleToTextBox(DerivativeOut, Dt);
                }

                if (CombinedControl)
                {
                    PI_m1.Calculate(Re, ActualSetpoint);
```

```
        PI_m2.Calculate(Re, ActualSetpoint);
        PI_m3.Calculate(Re, ActualSetpoint);
        PI_m4.Calculate(Re, ActualSetpoint);
        PI_m5.Calculate(Re, ActualSetpoint);

        if (SystemState == 1)
        {
            //PI_m1.SumIntegral = IntegralSum;
            //PI_m1.Calculate(Re, ActualSetpoint);
            //IntegralSum = PI_m1.SumIntegral;
            ComTask = PI_m1.Out;
            PortWrite(ComTask.ToString());
            DelayedInput[DelayLineDeep] = ComTask / 50;
            DoubleToTextBox(PI_m1.ProportionalOut, Pt);
            DoubleToTextBox(PI_m1.IntegralOut, It);
        }
        if (SystemState == 2)
        {
            //PI_m2.SumIntegral = IntegralSum;
            //PI_m2.Calculate(Re, ActualSetpoint);
            //IntegralSum = PI_m2.SumIntegral;
            ComTask = PI_m2.Out;
            PortWrite(ComTask.ToString());
            DelayedInput[DelayLineDeep] = ComTask / 50;
            DoubleToTextBox(PI_m2.ProportionalOut, Pt);
            DoubleToTextBox(PI_m2.IntegralOut, It);
        }
        if (SystemState == 3)
        {
            //PI_m3.SumIntegral = IntegralSum;
            //PI_m3.Calculate(Re, ActualSetpoint);
            //IntegralSum = PI_m3.SumIntegral;
            ComTask = PI_m3.Out;
            PortWrite(ComTask.ToString());
            DelayedInput[DelayLineDeep] = ComTask / 50;
            DoubleToTextBox(PI_m3.ProportionalOut, Pt);
            DoubleToTextBox(PI_m3.IntegralOut, It);
        }
        if (SystemState == 4)
        {
            //PI_m4.SumIntegral = IntegralSum;
            //PI_m4.Calculate(Re, ActualSetpoint);
            //IntegralSum = PI_m4.SumIntegral;
            ComTask = PI_m4.Out;
            PortWrite(ComTask.ToString());
            DelayedInput[DelayLineDeep] = ComTask / 50;
            DoubleToTextBox(PI_m4.ProportionalOut, Pt);
            DoubleToTextBox(PI_m4.IntegralOut, It);
        }
        if (SystemState == 5)
        {
            //PI_m5.SumIntegral = IntegralSum;
            //PI_m5.Calculate(Re, ActualSetpoint);
            //IntegralSum = PI_m5.SumIntegral;
            ComTask = PI_m5.Out;
            PortWrite(ComTask.ToString());
            DelayedInput[DelayLineDeep] = ComTask / 50;
            DoubleToTextBox(PI_m5.ProportionalOut, Pt);
            DoubleToTextBox(PI_m5.IntegralOut, It);
        }
    }

    DoubleToTextBox(ComTask, ActualTask);
```

```
                }

                DbTimer();
            };
            //
        }

        public void DoubleToTextBox(double val, TextBox tb) //! запись числа
double в TextBox
        {
            //
            if (tb.InvokeRequired) tb.Invoke(new Action<string>((s) => tb.Text =
s), val.ToString());
            else tb.Text = val.ToString();
            //
        }

        public double Timer() //! таймер для графика
        {
            //
            StopWatch.Stop();
            int     temp=     StopWatch.Elapsed.Milliseconds     +     (1000     *
StopWatch.Elapsed.Seconds);
            StopWatch.Restart();
            return (double)temp / 1000;
            //
        }

        public void DbTimer() //! отладочный таймер цикла
        {
            //
            DbStopWatch.Stop();
            int    temp    =    DbStopWatch.Elapsed.Milliseconds    +    (1000    *
DbStopWatch.Elapsed.Seconds);
            try
            {
                Debug.Text = temp.ToString();
            }
            catch
            {
                if (Debug.InvokeRequired) Debug.Invoke(new Action<string>((s) =>
Debug.Text = s), temp.ToString());
                else Debug.Text = temp.ToString();
            }
            DbStopWatch.Restart();
            //
        }


        public void ReadFromFile() //! чтение обучающей выборки с файла
        {
            //
            int i = 0;
            int k = 0;
            string line;
            using  (StreamReader  sr  =  new  StreamReader(FilePath  +  '\\'  +
FileName))
            { while (sr.ReadLine() != null) { i++; } }
            X = new double[i];
            Y = new double[i];
            i = 0;
            using  (StreamReader  sr  =  new  StreamReader(FilePath  +  '\\'  +
FileName))
```

```
            {
                while ((line = sr.ReadLine()) != null)
                {
                    k = line.IndexOf(' ');
                    X[i] = Double.Parse(line.Substring(0, k));
                    Y[i] = Double.Parse(line.Substring(k + 1, line.Length - k -
1));
                    i++;
                }
            }
            //
        }

        public void InitializePlot() //! инициализация графика
        {
            //
            Plot.Parent = this;
            xySet = new Series("Rps");
            xySet.ChartType = SeriesChartType.Line;
            Plot.Series.Add(xySet);
            xySete = new Series("Rpse");
            xySete.ChartType = SeriesChartType.Line;
             xySete.BorderWidth = 2;
            xySete.Color = Color.Black;
            Plot.Series.Add(xySete);
            //
            xySet_m1 = new Series("Rps_m1");
            xySet_m1.ChartType = SeriesChartType.Line;
            xySet_m1.Color = Color.Red;
            Plot.Series.Add(xySet_m1);
            xySet_m2 = new Series("Rps_m2");
            xySet_m2.ChartType = SeriesChartType.Line;
            xySet_m2.Color = Color.Orange;
            Plot.Series.Add(xySet_m2);
            xySet_m3 = new Series("Rps_m3");
            xySet_m3.ChartType = SeriesChartType.Line;
            xySet_m3.Color = Color.Green;
            Plot.Series.Add(xySet_m3);
            xySet_m4 = new Series("Rps_m4");
            xySet_m4.ChartType = SeriesChartType.Line;
            xySet_m4.Color = Color.Blue;
            Plot.Series.Add(xySet_m4);
            xySet_m5 = new Series("Rps_m5");
            xySet_m5.ChartType = SeriesChartType.Line;
            xySet_m5.Color = Color.Violet;
            Plot.Series.Add(xySet_m5);
            //
            stPlot.Parent = this;
            stSet = new Series("state");
            stSet.ChartType = SeriesChartType.Line;
            stPlot.Series.Add(stSet);
            stSet.BorderWidth = 2;
            stPlot.ChartAreas[0].AxisX.MajorGrid.Enabled = false;
            //stPlot.ChartAreas[0].AxisY.MajorGrid.Enabled = false;
            //
            errPlot.Parent = this;
            errSet = new Series("err");
            errSet.ChartType = SeriesChartType.Column;
            errPlot.Series.Add(errSet);
            //
            emuPlot.Parent = this;
            emuSet = new Series("emu");
            emuSet.ChartType = SeriesChartType.Line;
```

```
    emuSet.Color = Color.Red;
    emuPlot.Series.Add(emuSet);
    realSet = new Series("real");
    realSet.Color = Color.DarkCyan;
    realSet.ChartType = SeriesChartType.Line;
    emuPlot.Series.Add(realSet);
    //
}

public void InitializePort()  //! инициализация последовательного порта
{
    //
    //string[] s=SerialPort.GetPortNames();
    //foreach (string ss in s) { Console.WriteLine(ss); }
    Port = new SerialPort("COM3", 2000000);
    if (!Port.IsOpen)
    {
        try
        {
            Port.Open();
            System.Threading.Thread.Sleep(1000);
        }
        catch { }
    }
    else
    {
        Console.WriteLine("Port is closed.");
    }
    //
}

public string PortRead() //! чтение последовательного порта
{
    //
    try
    {
        Port.DiscardInBuffer();
        System.Threading.Thread.Sleep(50);
        //Port.ReadLine();
        return Port.ReadLine();
    }
    catch { return "error"; }
    //
}

public void PortWrite(string str) //! запись в последовательный порт
{
    //
    try
    {
        Port.Write('{' + str + '}');
    }
    catch { }
    //
}

private void InitializeLoopTimer() //! инициализация таймера цикла Loop
{
    //
    Timer LoopTimer = new Timer();
    LoopTimer.Interval = LoopPeriod;
    LoopTimer.Tick += new EventHandler(Loop);
    LoopTimer.Enabled = true;
```

```
            //
        }

        private void ToPLC_PreviewKeyDown(object sender, PreviewKeyDownEventArgs
e) //! событие ввода задания
        {
            //
            if (e.KeyCode == Keys.Enter)
            {
                if (PIDControl ^ NetControl ^ CombinedControl)
                {
                    ActualSetpoint          =          Double.Parse(toPLC.Text,
System.Globalization.CultureInfo.InvariantCulture);
                }
                else
                {          ComTask          =          Double.Parse(toPLC.Text,
System.Globalization.CultureInfo.InvariantCulture);       PortWrite(toPLC.Text);
DelayedInput[DelayLineDeep] = ComTask /50; }
                toPLC.SelectionStart = 0;
                toPLC.SelectionLength = toPLC.Text.Length;
            }
            //
        }

        private void EnableDrawing_Click(object sender, EventArgs e) //! событие
нажатия на кнопку Отрисовать
        {
            PermissionDraw = !PermissionDraw;
        }

        public void Form1_FormClosing(object sender, FormClosingEventArgs e) //!
событие закрытия программы
        {
            PortWrite("0");
            Port.Close();
            Environment.Exit(0);
        }

        public Network NetFromFile(string FileName) //! считывание нейросети из
файла
        {
            //
            Network Temp = null;
            NetSerialize             Serializer             =             new
NetSerialize(FilePath+'\\'+FileName+".txt", Temp);
            Temp = (Network)Serializer.Deserialize();
            Serializer.Close();
            return Temp;
            //
        }

        public void NetToFile(Network Net, string FileName) //! запись нейросети
в файл
        {
            //
            NetSerialize  Serializer  =  new  NetSerialize(FilePath  +  '\\'  +
FileName + ".txt", Net);
            Serializer.Serialize();
            Serializer.Close();
            //
        }

        [Serializable]
```

```
        public class Neuron //! нейрон
        {
            //
            public Neuron(double[] weights_init, double[] inputs_init)
            {
                Weights = new double[inputs_init.Length + 1];
                WeightsErrors = new double[inputs_init.Length + 1];
                Inputs = new double[inputs_init.Length];
                Weights = weights_init;
                Inputs = inputs_init;
            }

            public double[] Weights;
            public double[] WeightsErrors;
            public double[] Inputs;
            public double Sum;
            public double Output;


            public void Calculate()
            {
                Sum = Weights[Inputs.Length] * 1;
                for (int i = 0; i < Inputs.Length; i++)
                {
                    Sum += Weights[i] * Inputs[i];
                }
                Output = 1/(1 + Math.Exp(-Sum));
            }

            public double Gradient()
            {
                return Output * (1 - Output);
            }
            //
        }

        [Serializable]
        public class NetInput //! вход нейронной сети
        {
            //кол-во входов определяется обучаещей выборкой; InitVals -
начальные значения
            public NetInput(double[] InitVals)
            {
                Outputs = new double[InitVals.Length];
                Calculate(InitVals);
            }

            public double[] Outputs;

            public void Calculate(double[] Trainset)
            {
                for (int i = 0; i < Trainset.Length; i++) { Outputs[i] =
Trainset[i]; }
            }
            //
        }

        [Serializable]
        public class NetLayer //! слой нейронной сети
        {
            // n - кол-во нейронов
            public NetLayer(int n, double[] InitInputs)
            {
```

```
                Neurons = new Neuron[n];
                Outputs = new double[n];
                Errors = new double[n];
                GradientSum = new double[n+1];
                Inputs = new double[InitInputs.Length];
                Rnd = new Random(n + InitInputs.Length);
                double Scale = 0.7 * Math.Pow(n, 1 / InitInputs.Length); //
коэффициент масштабирования при инициализации весов
                double wSum = 0; // среднеквадратичная сумма весов слоя
                for (int i = 0; i < n; i++)
                {
                    double[] InitWeights = new double[InitInputs.Length+1];
                    InitWeights[InitInputs.Length] = (((double)Rnd.Next(0, 2000)
/ 1000) - 1) * Scale;
                    double[] Smth = new double[InitInputs.Length]; // ???
                    for (int j = 0; j < InitInputs.Length; j++)
                    { InitWeights[j] = (double)Rnd.Next(0, 1000) / 1000 - 0.5;
wSum += InitWeights[j] * InitWeights[j]; Smth[j] = i; }
                    Neurons[i] = new Neuron(InitWeights, InitInputs);
                    Neurons[i].Calculate();
                    Outputs[i] = Neurons[i].Output;
                }
                wSum = Math.Sqrt(wSum);
                for (int i = 0; i < n; i++)
                {
                    for (int j = 0; j < InitInputs.Length; j++) {
Neurons[i].Weights[j] = Scale * Neurons[i].Weights[j] / wSum; }
                }
            }


            public Neuron[] Neurons;
            public double[] Inputs;
            public double[] Outputs;
            public double[] Errors;
            public double[] GradientSum;
            public double InnerTreshold = 0.0000000000001;
            Random Rnd;


            public void Calculate(double[] Inputs)
            {
                for (int i = 0; i < Neurons.Length; i++) { Neurons[i].Inputs =
Inputs; Neurons[i].Calculate(); Outputs[i] = Neurons[i].Output;}
                this.Inputs = Inputs;
            }


            public void GetOutputErrors(double[] ActOutputs)
            {
                for (int i = 0; i < ActOutputs.Length; i++)
                {
                    Errors[i] = ActOutputs[i] - Outputs[i];
                    for (int j = 0; j < Neurons[0].Weights.Length-1; j++) {
Neurons[i].WeightsErrors[j] = Errors[i] * Neurons[i].Gradient() *
Neurons[i].Inputs[j]; }
                    Neurons[i].WeightsErrors[Neurons[0].Weights.Length-1] =
Errors[i] * Neurons[i].Gradient() * 1;
                }
            }

            public void GetHiddenErrors(NetLayer NextLayer)
            {
```

```
                for (int i = 0; i < Neurons.Length; i++)
                {
                    GradientSum[i] = 0;
                    for (int j = 0; j < NextLayer.Neurons.Length; j++) {
GradientSum[i] += NextLayer.Errors[j] * NextLayer.Neurons[j].Gradient() *
NextLayer.Neurons[j].Weights[i]; }
                    Errors[i] = GradientSum[i];
                    for (int j = 0; j < Neurons[0].Weights.Length-1; j++) {
Neurons[i].WeightsErrors[j] = Errors[i] * Neurons[i].Gradient() *
Neurons[i].Inputs[j]; }
                    Neurons[i].WeightsErrors[Neurons[0].Weights.Length-1] =
Errors[i] * Neurons[i].Gradient() * 1;
                }
            }

            public void ChangeWeights(double LearningRate)
            {
                for (int i = 0; i < Neurons.Length; i++)
                {
                    for (int j = 0; j < Neurons[0].Weights.Length; j++)
Neurons[i].Weights[j] += LearningRate * Neurons[i].WeightsErrors[j];
                }
            }
            //
        }

        [Serializable]
        public class Network //! нейронная сеть
        {
            //
            public Network(double[][] TrainInputs, double[][] TrainOutputs, int
HiddenNeurons, double LearnRate)
            {
                Xt = TrainInputs;
                Yt = TrainOutputs;
                LearningRate = LearnRate;
                LocalErrors = new double[Xt.Length];
                Epochs = 0;
                SqrtError = 99;
                InputLayer = new NetInput(Xt[0]);
                HiddenLayer = new NetLayer(HiddenNeurons, InputLayer.Outputs);
                OutputLayer = new NetLayer(1, HiddenLayer.Outputs);
            }

            public NetInput InputLayer;
            public NetLayer HiddenLayer;
            public NetLayer OutputLayer;
            public double[][] Xt;
            public double[][] Yt;
            public double LearningRate;
            public double SqrtError; //квадрат СКО за эпоху
            public double[] LocalErrors; //ошибки в эпохе
            public int Epochs; //колличество эпох


            public void Calculate(double[] Input)
            {
                InputLayer.Calculate(Input);
                HiddenLayer.Calculate(InputLayer.Outputs);
                OutputLayer.Calculate(HiddenLayer.Outputs);
            }

            public void GetErrors(double[] Output)
```

```
            {
                OutputLayer.GetOutputErrors(Output);
                HiddenLayer.GetHiddenErrors(OutputLayer);
            }

            public void ChangeWeights()
            {
                HiddenLayer.ChangeWeights(LearningRate);
                OutputLayer.ChangeWeights(LearningRate);
            }

            public void Train()
            {
                Epochs++;
                SqrtError = 0;
                for (int i = 0; i < Xt.Length; i++)
                {
                    Calculate(Xt[i]);
                    GetErrors(Yt[i]);
                    ChangeWeights();
                    SqrtError    +=    0.5    *    OutputLayer.Errors[0]    *
OutputLayer.Errors[0];
                    LocalErrors[i] = OutputLayer.Errors[0];
                }
                SqrtError = Math.Sqrt(SqrtError/ Xt.Length);
            }
            //
        }

        public class PI //! ПИ-регулятор
        {
            //
            public PI(double P, double I)
            {
                ProportionalCoef = P;
                IntegralCoef = I;
                SumIntegral = 0;
                PrevError = 0;
            }

            public double ProportionalCoef;
            public double IntegralCoef;
            public double SumIntegral;
            public double PrevError;
            public double CurrentError;
            public double Out;
            public double ProportionalOut;
            public double IntegralOut;
            public double ActualSetpoint;

            public void Calculate(double Value, double Setpoint)
            {
                ActualSetpoint = Setpoint;
                CurrentError = ActualSetpoint - Value;
                ProportionalOut = ProportionalCoef * CurrentError;
                IntegralOut = SumIntegral + IntegralCoef * CurrentError * 0.065;
                Out = ProportionalOut + IntegralOut;
                if (Out >= 100) Out = 100;
                if (Out <= 0) Out = 0;
                SumIntegral += IntegralCoef * CurrentError;
                PrevError = CurrentError;
            }
            //
```

```
        }

        public class NetSerialize //! класс сериализации
        {
            //
            private object Type = null;
            private string File = "";
            private FileStream fStream;
            public NetSerialize(string Path, object Obj)
            {
                Type = Obj;
                File = Path;
            }

            public void Serialize()
            {
                fStream     =     new     FileStream(File,     FileMode.Create,
FileAccess.Write, FileShare.ReadWrite);
                BinaryFormatter bFormatter = new BinaryFormatter();
                bFormatter.Serialize(fStream, Type);
                fStream.Close();
            }

            public object Deserialize()
            {
                object a1;
                fStream = new FileStream(File, FileMode.Open, FileAccess.Read,
FileShare.Read);
                BinaryFormatter bFormatter = new BinaryFormatter();
                a1 = bFormatter.Deserialize(fStream);
                fStream.Close();
                return a1;
            }

            public void Close()
            {
                Type = null;
                File = "";
            }
            //
        }

    }
}
```