**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра АПУ**

**отчет**

**по исследовательской работе**

**по дисциплине «Моделирование Искусственного Интеллекта»**

**Тема: Нейросетевые регуляторы на основе статических и динамических нейронных сетей**

|  |  |  |
| --- | --- | --- |
| Студенты |  | Ларионов. Н. М. Павлов А. А. |
| Преподаватель |  | Грудяева Е.С. |

Санкт-Петербург

2024

Таблица изменений

|  |  |  |
| --- | --- | --- |
| Студент | Дата | Изменения |
| Ларионов Н.М. | 24.10.2024-29.10.2024 | Переработано содержание работы. Расписано введение. Добавлена таблица изменений, добавлено автособираемое оглавление, добавлена нумерация страниц, исправлена структура отчета исследования. |
| Павлов А.А. | 24.10.2024-29.10.2024 | Полностью переработано содержание раздела ПИД-регуляторы, изменена структура, изменено содержание определения терминов, добавлена классификация регуляторов. |
| Ларионов Н.М. | 29.10.2024-06.11.2024 | Исправлено введение. Переработано содержание пункта 2.3. Переработано содержание пункта 3.2. |
| Павлов А.А. | 29.10.2024-06.11.2024 | Поиск источников и подборка подходящих иллюстраций. Добавлен пункт 2.3 и соответствующие подпункты. Переработано содержание пункта 3.1. |
| Ларионов Н.М. | 06.11.2024- 13.11.2024 | Переработано введение. Изменена структура работы. Добавлены плюсы и минусы использования нейронных сетей в пункте 5.1 |
| Павлов А.А. | 06.11.2024- 13.11.2024 | Переработан глобальны пункт 3. Добавлено описание различия понятий динамических сетей. Поиск источников и добавление примеров для рекуррентных нейронных сетей. |
| Ларионов Н.М. | 13.11.2024- 20.11.2024 | Анализ датасета. Работа с табличными значениями и подготовка изображений. |
| Павлов А.А. | 13.11.2024- 20.11.2024 | Поиск датасетов и их описания. Изменение структуры пункта 7 и переработка пункта 7.1. |
| Ларионов Н.М. | 20.11.2024- 27.11.2024 | Исправление ошибок в коде. Работа с данными и подготовка изображений. Написание и оформление пункта 7.2 |
| Павлов А.А. | 20.11.2024- 27.11.2024 | Написание программы для обучения машинной модели, используемой для моделирования поведения легкого. |
| Ларионов Н.М. | 27.11.2024 – 04.12.2024 | Написание интерфейса программы и базовой общей модели взаимодействия легкого и регулятора. Оформление пункта 7.3. |
| Павлов А.А. | 27.11.2024 – 04.12.2024 | Настройка входных данных и конкретного поведения модели контроллера и входных данных. Исправление ошибок визуализации у временных диаграмм. Настройка коэффициентов регулятора и модели легких. |

Оглавление

[Введение 10](#_Toc184298385)

[1. Регуляторы 11](#_Toc184298386)

[1.1. Список терминов 11](#_Toc184298387)

[1.2 Устройство регуляторов в общем виде 13](#_Toc184298388)

[1.3 Основные типы регуляторов 14](#_Toc184298389)

[*1.3.1* *Пропорциональный регулятор (П-регулятор)* 14](#_Toc184298390)

[*1.3.2* *Пропорционально-интегральный регулятор (ПИ-регулятор)* 14](#_Toc184298391)

[*1.3.3* *Пропорционально-дифференциальный регулятор (ПД-регулятор)* 14](#_Toc184298392)

[*1.3.4* *Пропорционально-интегрально-дифференциальный регулятор (ПД-регулятор)* 15](#_Toc184298393)

[2. Нейронные сети 15](#_Toc184298394)

[2.1 Список терминов 15](#_Toc184298395)

[2.2 Устройство нейронных сетей в общем виде 15](#_Toc184298396)

[2.3 Различие статических и динамических нейронных сетей 20](#_Toc184298397)

[3. Статические нейронные сети 24](#_Toc184298398)

[4. Динамические нейронные сети 24](#_Toc184298399)

[4.1 Определение понятия динамической нейронной сети 24](#_Toc184298400)

[4.2 Рекуррентные нейронные сети 25](#_Toc184298401)

[4.2.1 Релаксационные нейронные сети 26](#_Toc184298402)

[4.2.2 Нейронная сеть Хопфилда 27](#_Toc184298403)

[4.2.3 Нейронная сеть Хемминга 30](#_Toc184298404)

[5. Интеграция нейронных сетей в ПИД-подобные регуляторы 33](#_Toc184298405)

[5.1 Преимущества и недостатки применения статических и динамических нейронных сетей 33](#_Toc184298406)

[5.2 ПИД-регулятор с блоком автонастройки на основе нейронной сети 38](#_Toc184298407)

[5.3 ПИД-регулятор с блоком автонастройки на основе нейронной сети 41](#_Toc184298408)

[6. Прикладное применение нейронных сетей в ПИД подобных регуляторах 41](#_Toc184298409)

[6.1 Настройка ПИД-контроллера для модели квадрокоптера 41](#_Toc184298410)

[7. Прикладное применение нейронных сетей в регуляторах 46](#_Toc184298411)

[7.1 Подготовка предметной области 46](#_Toc184298412)

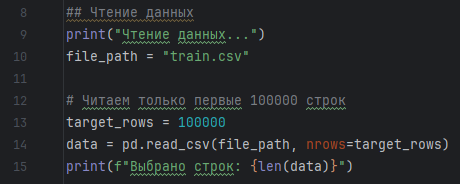
[7.1.1 Поиск предметной области 46](#_Toc184298413)

[7.1.2 Описание предметной области 46](#_Toc184298414)

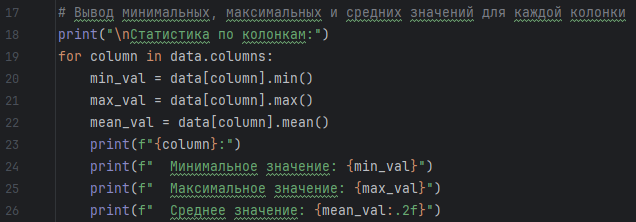
[7.1.3 Описание датасета 47](#_Toc184298415)

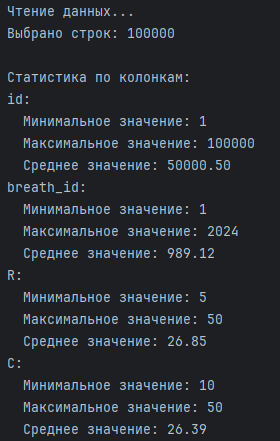
[7.1.4 Анализ данных 49](#_Toc184298416)

[7.2 Формирование модели легкого 52](#_Toc184298417)

[ 53](#_Toc184298418)

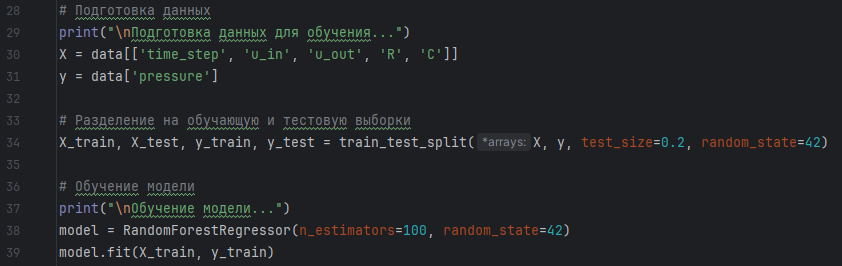
[Чтение данных производится при помощи библиотеки pandas. В память компьютера загружается выборка из 100 000 строчек, после чего проводится анализ данных. 53](#_Toc184298419)

[ 54](#_Toc184298420)

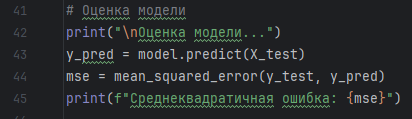
[ 54](#_Toc184298421)

[В терминал выводится данные по выборке для каждого параметра в столбце. Несмотря на то, что выводятся сразу все характеристики, важны только те, которые были указаны во время анализа данных. 54](#_Toc184298422)

[Далее производится подготовка данных для дальнейшего их использования при обучении. После завершения подготовки данных, они сохраняются в памяти компьютера и производится настройка и дальнейшее обучение модели случайного леса на основе обучающей выборки. 55](#_Toc184298423)

[ 55](#_Toc184298424)

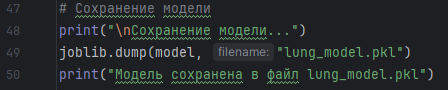
[После построения модели производится проверка ее точности предсказания при помощи тестирования на основе тестовой выборки. 55](#_Toc184298425)

[ 55](#_Toc184298426)

[ 55](#_Toc184298427)

[Можно увидеть, что точность построенной модели определяется по среднеквадратичной ошибке. Для построенной модели данное число равняется 17,84. При увеличении обучающей выборки с 100 000 до 500 000 удалось повысить точность модели до 17,0. Такую модель нельзя назвать достаточно точной для ее использования в предсказаниях. Однако такая точность полностью удовлетворяет базовым условиям для моделирования поведения легких у «больного пациента». 56](#_Toc184298428)

[После тестирования модели, она сохраняется в файловой системе компьютера и может быть в дальнейшем использована для интеграции с моделью ПИД-контроллера без необходимости заново обучать модель при каждом запуске программы. 56](#_Toc184298429)

[ 56](#_Toc184298430)

[Вывод. 56](#_Toc184298431)

[Источники. 56](#_Toc184298432)

[Приложение 1. 60](#_Toc184298433)

# Введение

Данная работа посвящена исследованию вопроса применения нейросетевых регуляторов и изучением проблем, связанных с их практической реализацией.

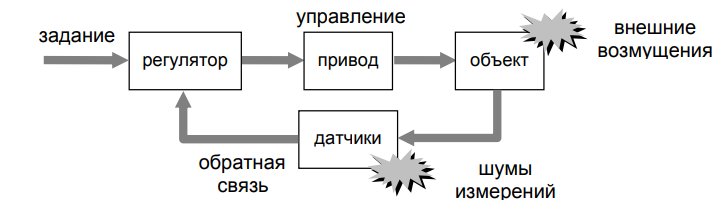
Современные технологии в области электроники и компьютерной техники позволяют реализовать в небольшом и относительно недорогом устройстве крайне мощный и универсальный инструмент – нейронную сеть. Нейросети могут быть интегрированы в промышленные регуляторы на различных производственных линиях условного предприятия, тем самым повышая точность работы рабочих механизмов и удешевляя стоимость их эксплуатации.

Целью данной работы является исследование возможностей разработки и применения нейросетевых регуляторов. Для этого необходимо изучить принципы работы нейросетей и нейросетевых регуляторов, детально исследовать область их применения и рассмотреть возможности их реализации и использования в прикладных задачи.

# Регуляторы

## Список терминов

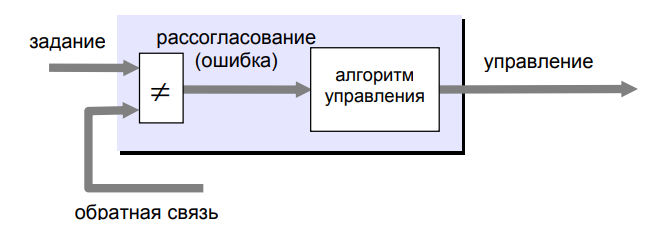
1. **Система управления** состоит из управляемого объекта (например, двигателя) и регулятора. Регулятор — это управляющий элемент, который изменяет состояние объекта, чтобы поддерживать желаемый режим работы, управляя через исполнительные механизмы и опираясь на данные от датчиков.
2. **Структура системы управления**:



**рис. 1**

* **Объект управления** — система, которая нуждается в управлении (например, двигатель или корабль).
* **Регулятор** — устройство, которое контролирует объект и стремится свести к нулю отклонения от заданного режима.
* **Датчики** — устройства, фиксирующие состояние объекта и передающие эту информацию регулятору.
* **Исполнительные механизмы** — элементы, которые непосредственно воздействуют на объект (например, приводы или клапаны).

1. **Обратная связь** — ключевой элемент системы, благодаря которому регулятор получает информацию о текущем состоянии объекта, сравнивая его с заданным значением, и корректирует управление. Системы с обратной связью называют замкнутыми, тогда как разомкнутые системы работают без корректировки в реальном времени.
2. **Рассогласование** — разница между текущим состоянием объекта и заданным значением. Регулятор стремится минимизировать эту разницу.



**рис. 2**

1. **Типы задач в управлении**:

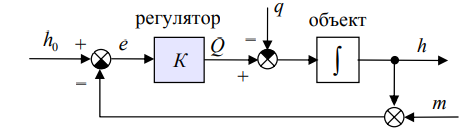
* **Стабилизация** — поддержание неизменного состояния системы.
* **Программное управление** — управление системой по заранее установленной программе.
* **Слежение** — адаптация к изменяющемуся внешнему воздействию.

1. **Виды систем**:

* **Линейные и нелинейные**: Линейные системы проще в управлении и моделировании, в них параметры изменяются пропорционально входным воздействиям. В нелинейных системах, встречающихся в реальности, такие зависимости более сложны.
* **Одномерные и многомерные**: одномерные системы имеют один вход и выход, а многомерные — несколько входов и выходов.
* **Стационарные и нестационарные**: параметры стационарных систем постоянны, а в нестационарных изменяются с течением времени.
* **Детерминированные и стохастические**: в детерминированных системах параметры известны точно, а в стохастических — вероятностны.

## Устройство регуляторов в общем виде

Регулятор отслеживает разницу между текущим и желаемым состоянием системы (ошибку), преобразуя её в управляющий сигнал. Это обычно достигается через обратную связь: регулятор получает данные от датчиков, сравнивает их с заданием и регулирует воздействие на объект. Системы с обратной связью называют замкнутыми [[3](#_Источники.)].



**рис. 3**

## Основные типы регуляторов

### *Пропорциональный регулятор (П-регулятор)*

Управляющий сигнал *q*(*t*) пропорционален ошибке

*q*(*t*)  *K*  *e*(*t*)  *K*  *h*0 (*t*)  *h*(*t*).

где K ​ — коэффициент усиления регулятора.

П-регулятор эффективен для устойчивых объектов, но оставляет статическую ошибку в установившемся режиме, особенно при возмущениях

### *Пропорционально-интегральный регулятор (ПИ-регулятор)*

Содержит интегральный канал, что позволяет компенсировать статическую ошибку. Интегральная часть постепенно накапливает ошибку и на основе этого корректирует систему



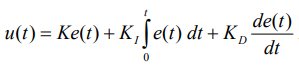
### *Пропорционально-дифференциальный регулятор (ПД-регулятор)*

Добавляет к пропорциональной части дифференциальный канал, что помогает предугадывать и смягчать изменения, реагируя на скорость изменения ошибки​

*u*(*t*)  *K*  *e*(*t*) +

### *Пропорционально-интегрально-дифференциальный регулятор (ПД-регулятор)*

Сочетает все три элемента, что позволяет управлять ошибкой точнее, улучшая реакцию системы и снижая статическую ошибку. Он особенно эффективен в сложных задачах управления



# Нейронные сети

## Список терминов

* **Искусственная нейронная сеть** (*ИНС*) - математическая модель, а также её программное или аппаратное воплощение, построенное по принципу организации и функционирования биологических нейронных сетей. Представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов), каждый из которых функционирует с сигналами, поступающими на ему на вход, и с сигналами, которые он посылает другим процессорам

## Устройство нейронных сетей в общем виде

В целях ознакомления с вариантами применения статических и динамических нейронных сетей, было изучено их устройство и принципы их работы [[4](#_Источники.)][[5](#_Источники.)].

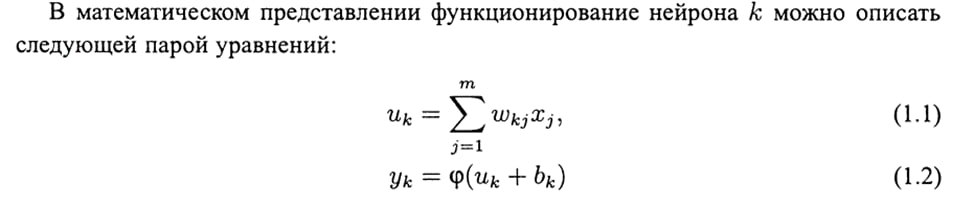
Нейронная сеть в общем случае представляет собой математическую модель, реализованную на программном уровне, построенную по принципу организации и функционирования биологических нейронных сетей. Представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов), каждый из которых функционирует с сигналами, поступающими на ему на вход, и с сигналами, которые он посылает другим процессорам.

В общем виде один процессор внутри сети можно изобразить следующим образом:



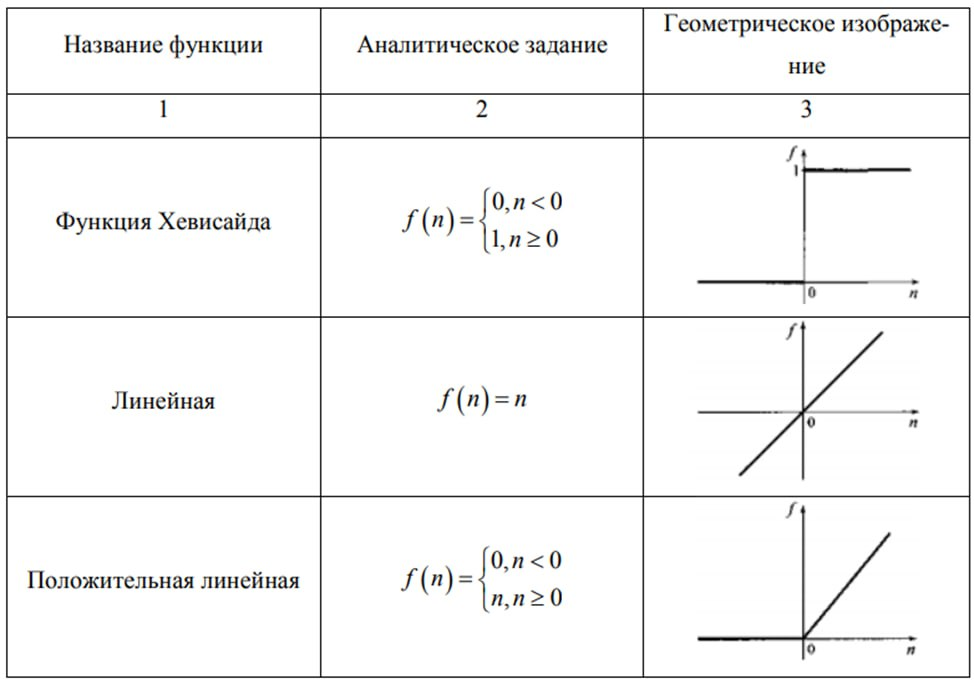
**рис. 4**

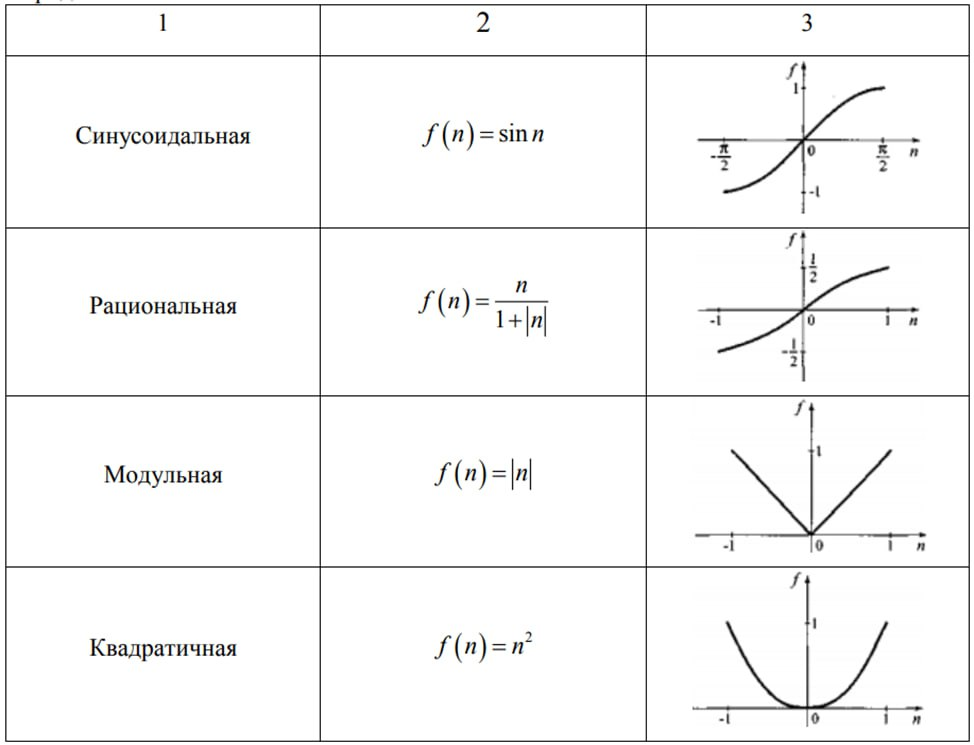
На схеме процессор (нейрон) получает различные сигналы, каждый из которых имеет разные веса и складывает их. При достижении порогового значения, процессор начинает генерировать свой сигнал и передает его следующему процессору.

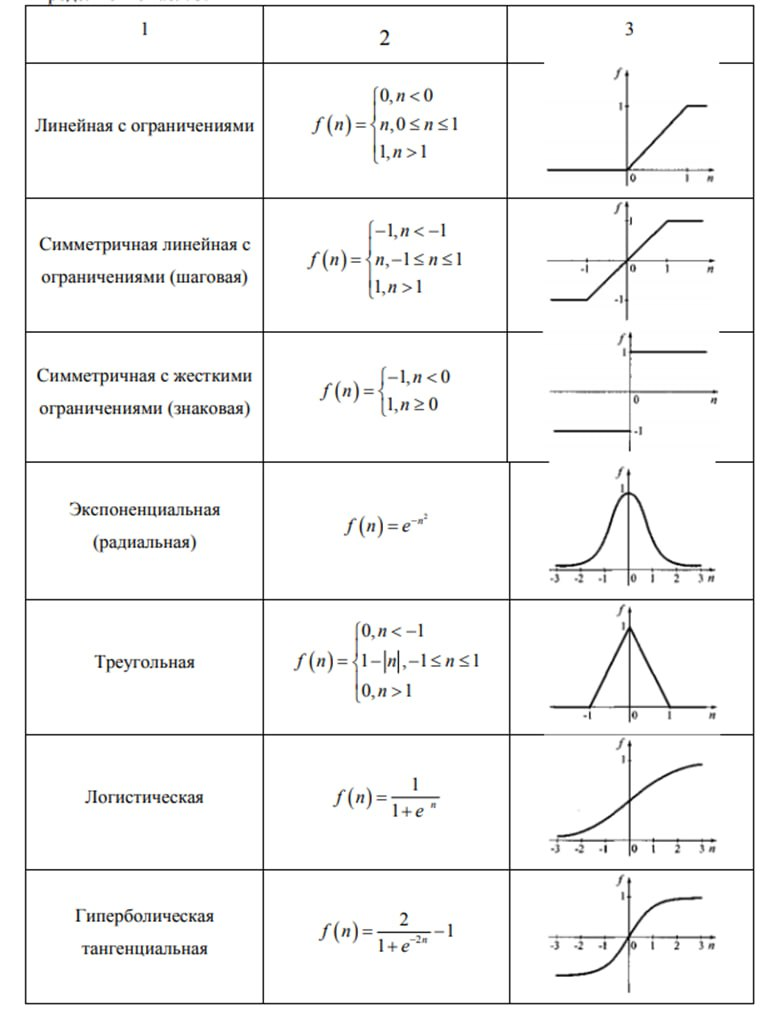


Передаваемые сигналы могут различаться в зависимости от суммы значений получаемых сигналов. Зависимость выходных сигналов от входных называется активационной функцией, которая зачастую будет нелинейной.

Ниже представлены варианты наиболее часто применяемых активационных функций:







Множество процессоров-нейронов формируют нейронные слои, каждый из которых представляет специальный уровень обработки. Входные слои описывают получаемую от мира информацию, когда как выходные слои принимают решения, как на нее реагировать. В искусственных нейронных сетях все нейроны в слое делят те же входы и активационную функцию, как показано на изображении:

В общем виде нейронные слои можно представить следующим образом:



**рис. 5**

Нейронные сети могут быть составлены из нескольких соединенных слоев, которые называются многослойными сетями. Обычные нейронные слои могут быть разделены на 3 класса:

1. Входной слой

2. Слой скрытых нейронов

3. Выходной слой нейронов

На практике, дополнительный нейронный слой добавляет другой уровень абстракции внешней стимуляции, тем самым повышая способность нейронных сетей представлять больше комплексных данных.

Каждая нейросеть имеет как минимум входной/выходной слой независимо от количества слоев. В случае с многослойной сетью, слои между входом и выходом названы скрытыми.

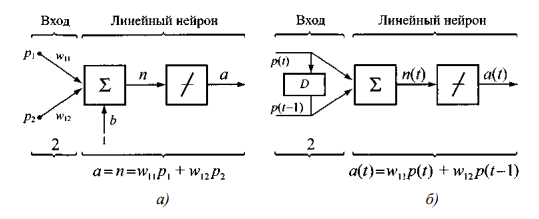
## Различие статических и динамических нейронных сетей

Устройство статических и динамических нейронных сетей, включая описание их различия, хорошо описаны в учебно-методическом пособии [[7](#_Источники.)]

Пусть имеется последовательность m входных векторов. Будем говорить, что вход ИНС статический (неподвижный), если входная информация – сигналы – подается на сеть в виде последовательности уже сформированных заранее векторов входа, которые обрабатываются ИНС одновременно (от формирования взвешенного входа до получения выходного вектора). ИНС со статическим входом носит соответствующее название – *статическая ИНС*. Работа такой ИНС равносильна одновременному функционированию одинаковых сетей, на каждую из которых подавался бы один из векторов входа и генерировался один из выходов. Примером таких нейросетей могут быть многослойные персептроны и радиальные базисные функции.

Пусть теперь имеется последовательность m отдельных сигналов (импульсов), не являющихся векторами входа. Будем говорить, что вход ИНС динамический (подвижный), если входная информация подается на сеть в виде последовательности объектов, не являющихся входными векторами, на основании которой ИНС создает по некоторому правилу последовательно («пошагово») входные векторы, формирует взвешенный вход и генерирует выход. Сеть с динамическим входом носит одноименное название – динамическая ИНС. Составные части динамических сетей, которые обеспечивают формирование входных векторов, называются линиями (элементами) задержки (запаздывания).

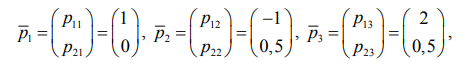
Для лучшего понимания статических и динамических сетей рассмотрим конкретный пример. Построим наипростейшие статическую и динамическую сети ИНС-1, состоящие из одного нейрона с прямыми связями и линейной функции активации f(n)=n – линейного нейтрона.



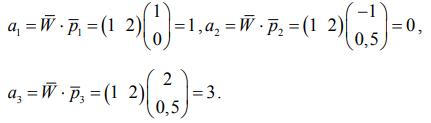
**рис. 6**

Как видно из рисунка а, у статической сети векторы входа и весов двухэлементные: , а значит, сетевая функция имеет вид  и одновременно дает выход сети (согласно выбранной функции активации). Присваиваем значения весов и смещения: W = (1,2), b=0 (нейрон без смещения). Сеть готова к работе, так как определились все ее характеристики-параметры.

Пусть на сеть подается последовательность из трех векторов входа:



тогда соответствующие выходы таковы:



Статическая сеть обрабатывает сразу все векторы и выдает результат в виде трехкомпонентного вектора . Объясняется это тем, что векторы  можно рассматривать как два трехэлементных вектора:



и тогда их взвешенная сумма даст вектор :

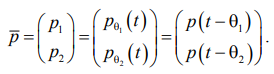


То есть сеть взвешивает одновременно все первые и все вторые координаты заданных векторов входа, а потом находит сумму полученных трехмерных векторов и выдает вектор .

Динамическая сеть (рис. б) так действовать не может, так как для нее входная информация представляется объектами, которые не являются входными векторами. Для обеспечения возможности формировать векторы входа статическую сеть дополняют элементом задержки D (delay - задержка), который характеризуется вектором задержки . Значения его компонент определяют выбор правила формирования входных векторов. Возвращаясь, к примеру, положим , а последовательность входных сигналов, из которых будут формироваться входные векторы, выберем такой:



Координаты входных векторов р1, р2 рассматриваются как функции параметра τ – абстрактного времени – с учетом вектора задержки:



В нашем случае , значит



где t = 0, 1, 2, 3, 4 – моменты абстрактного времени. Соотношения определяют правило построения входных векторов. Для t = 0 – начального момента времени – обычно полагают р (0) =0.

Придавая t значения 1, 2, 3, 4, получаем последовательность входных векторов:



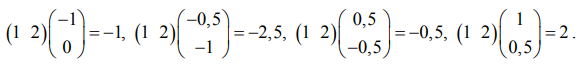
Базовая функция, а у нас и выход, формируются согласно уравнению:



В результате работы сеть сгенерирует выход



Действительно:



Подводя итоги вышеизложенного, отметим, что в отличие от биологических нейронных сетей – реально существующих объектов, ИНС является вычислительной схемой (процедурой), которая по заданной (входной) числовой информации (обычно это числовые массивы) формирует выходные массивы чисел.

# Статические нейронные сети

# Динамические нейронные сети

## Определение понятия динамической нейронной сети

У динамических нейронных сетей нет устоявшегося определения. Разные исследователи и авторы подходят к описанию динамических нейронных сетей с различных сторон, акцентируя внимание на разных аспектах этих нейронных сетей.

Были рассмотрены и проанализированы следующие определения:

1. Наличие обратных связей в сетях оказывает непосредственное влияние на способность таких сетей к обучению и на их производительность. Более того, обратная связь подразумевает использование элементов единичной задержки, что приводит к нелинейному динамическому поведению [[9](#_Источники.)].

2. Следующий класс нейронных сетей, который мы рассмотрим, - динамические или рекуррентные, сети. Они построены из динамических нейронов, чье поведение описывается дифференциальными или разностными уравнениями, как правило, первого порядка. Сеть организована так, что каждый нейрон получает входную информацию от других нейронов (возможно, и от себя самого) и из окружающее среды. этот тип сетей имеет важное значение, так как с его помощью можно моделировать нелинейные динамические системы. это – весьма общая модель, которую потенциально можно использовать в самых разных приложениях, например: ассоциативная память, нелинейная обработка сигналов, моделирование конечных автоматов, идентификация систем, задачи управления [[4](#_Источники.)].

3. Динамические нейронные сети представляют собой сети, построенные из нейронов, состояние которых описывается системой дифференциальных уравнений. Организация сети такова, что каждый нейрон получает сигналы от ансамбля нейронов. С помощью искусственных нейронных сетей можно моделировать работу нелинейных динамических систем. Такие сети можно использовать при моделировании ассоциативной памяти, внимания, задач управления сложными системами, распознавании и классификации образов, нелинейной обработке сигналов и др. Следует отметить, что вследствие широкого спектра различных применений невозможно создать универсальную модель нейронной сети. Каждая такая модель имеет кроме ряда преимуществ и очевидные ограничения, которые необходимо учитывать в практических реализациях [[11](#_Источники.)].

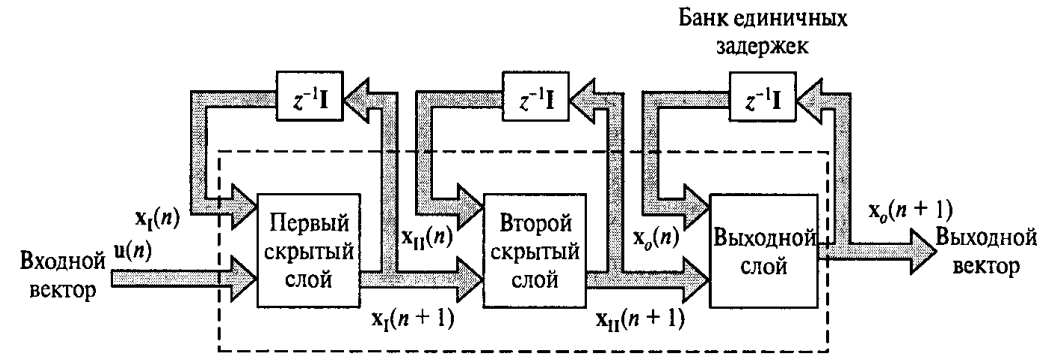
4. Пусть теперь имеется последовательность m отдельных сигналов (импульсов), не являющихся векторами входа. Будем говорить, что вход ИНС динамический (подвижный), если входная информация подается на сеть в виде последовательности объектов, не являющихся входными векторами, на основании которой ИНС создает по некоторому правилу последовательно («пошагово») входные векторы, формирует взвешенный вход и генерирует выход. Сеть с динамическим входом носит одноименное название – динамическая ИНС. Составные части динамических сетей, которые обеспечивают формирование входных векторов, называются линиями (элементами) задержки (запаздывания) [[7](#_Источники.)].

В ходе исследования было принято решение придерживаться последнего итогового определения [[7](#_Источники.)], так как данное определение является наиболее широким и универсальным. Данное определение охватывает все основные аспекты динамических нейронных сетей данные другими определениями. Таким образом, всё, что соответствует другим определениям, будет также удовлетворять критериям, изложенным в последнем определении, что делает его наиболее обобщающим и удобным для дальнейшего анализа и применения.

## Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) – это сети с обратной связью между различными слоями нейронов [[10](#_Источники.)].

Для сохранения обратного сигнала во времени, используются операторы (Банк) единичной задержки.



**рис. 7**

Главная особенность, выделяющая эти сети среди других нейронных сетей, – динамические зависимости на каждом этапе функционирования. Изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа "один ко многим". В сети возникает некоторый переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего.

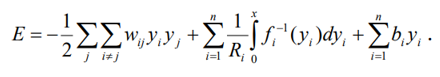
Рекуррентные НС характеризуются прямым и обратным распространением информации между слоями нейронной сети. Обратная связь может присутствовать в нейронных сетях в виде локальной обратной связи (то есть на уровне одного нейрона) или глобальной ОС на уровне всей НС. На практике в основном используются два класса рекуррентных сетей:

* ассоциативная память
* сети отображения вход-выход (RMLP, RTRN, НС Элмана)

### Релаксационные нейронные сети

Подклассом рекуррентных НС являются релаксационные НС (РНС) [[8](#_Источники.)]. В основе функционирования таких сетей лежит итеративный принцип работы. На каждой итерации процесса происходит обработка данных, полученных на предыдущем шаге. Такая циркуляция информации продолжается до тех пор, пока не установится состояние равновесия (релаксация). При этом состояния нейронной сети перестают изменяться и характеризуются стационарными значениями.

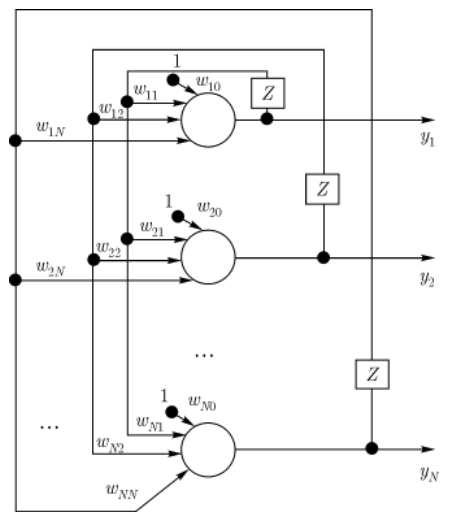
Если функцию активации нейрона обозначить, где u – взвешенная сумма его возбуждений, то состояние нейрона можно определить выходным сигналом, где n – размерность входного вектора и количество нейронов в первом слое. Изменение состояния i-го нейрона можно описать системой дифференциальных уравнений для i 1,n , где i b – пороговое значение. Рекуррентной сети можно поставить в соответствие энергетическую функцию Ляпунова:



Классическими примерами РНС являются нейронные сети Хопфилда, Хемминга, двунаправленная ассоциативная память (ДАП) и машина Больцмана (МБ)

### Нейронная сеть Хопфилда

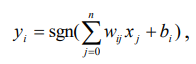
В 1982 году американский биофизик Д. Хопфилд представил математический анализ релаксационных сетей с обратными связями [[8](#_Источники.)]. Поэтому такие НС получили название сетей Хопфилда. НС Хопфилда реализует существенное свойство авто ассоциативной памяти – восстановление по искаженному (зашумленному) образцу ближайшего к нему эталонного. В этом случае входной вектор используется как начальное состояние сети, и далее сеть эволюционирует согласно своей динамике. Выходной восстановленный образец формируется, когда сеть достигает равновесия.

****

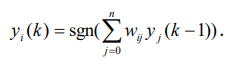
**рис. 8**

Структура сети Хопфилда представляется в виде системы с непосредственной обратной связью выхода со входом. Выходные сигналы нейронов являются одновременно входными сигналами сети:

Наиболее часто используется в качестве функции активации используется биполярная ступенчатая функция активации со значениями, то есть выходной сигнал i-го нейрона определяется функцией:

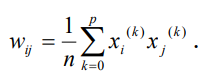


Если порог срабатывания функции (7.2) является компонентой вектора X. Тогда основную зависимость, определяющую сеть Хопфилда, можно представить в виде:

****

Три наиболее часто используемых метода обучения сети Хопфилда: правило Хебба, метод проекций, метод дельта проекций.

Для одного обучающего вектора значения весов могут быть вычислены по правилу Хебба: так как вследствие биполярных значений элементов вектора X, то есть всегда. При вводе большего количества обучающих векторов веса подбираются согласно обобщенному правилу Хебба:

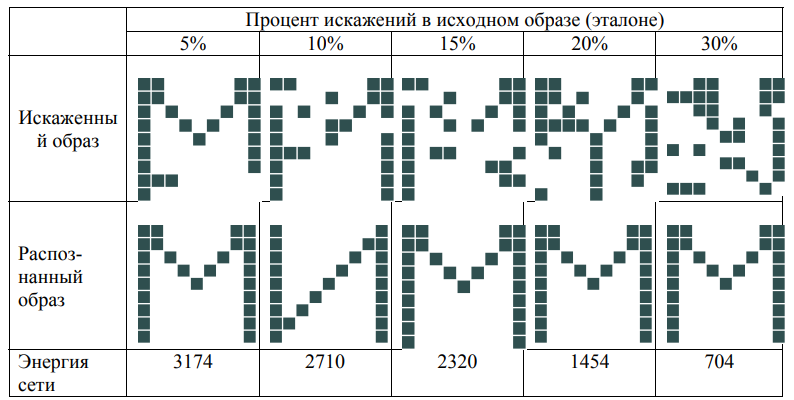


Показано, что при использовании для обучения правила Хебба максимальная емкость памяти составит всего лишь около 13,8% от количества нейронов, образующих ассоциативную память. Столь малая емкость обусловлена тем, что сеть Хебба хорошо запоминает только взаимно ортогональные векторы или близкие к ним.

Одним из примеров применения нейронной сети Хопфилда является ее использование в качестве инструмента распознавания образов [[12](#_Источники.)]. Нейронная сеть, обученная на основе сети Хопфилда позволяет распознавать образы, искаженные вплоть до 30%, что можно увидеть на рисунках ниже.



**рис. 9. Примеры распознавания буквы «E».**



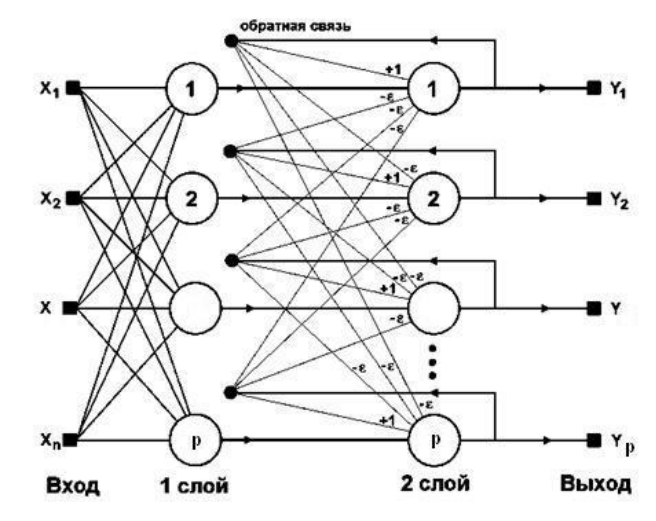
**рис. 10. Примеры распознавания буквы «М».**

### Нейронная сеть Хемминга

Когда нет необходимости, чтобы сеть в явном виде выдавала образец, то есть достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга [[5](#_Источники.)]. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений.

Достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Многочисленные эксперименты доказали, что сеть Хемминга дает лучшие результаты, чем сеть Хопфилда.

Единственная проблема, связанная с сетью Хемминга, проявляется в случае, когда зашумленные образы находятся на одинаковом (в смысле Хемминга) расстоянии от двух или более эталонов. В этом случае выбор сетью Хемминга одного из эталонов становится случайным.

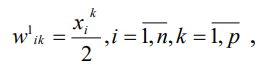
****

**рис. 11**

Сеть Хемминга включает в себя два слоя. Первый слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой состоит из нейронов, связанных обратными связями по принципу "каждый с каждым", при этом в каждом нейроне слоя существует авто связь (связь входа нейрона со своим собственным выходом). Количество нейронов в каждом слое сети равно количеству запоминаемых векторов.

Разные нейроны во втором слое связаны отрицательной (тормозящей) обратной связью с весом, при этом величина обычно обратно пропорциональна количеству образов. С собственным входом нейрон связан положительной (возбуждающей) обратной связью с весом, равным +1. Пороговые веса нейронов приняты равными нулю. Нейроны этого слоя функционируют в режиме WTA (англ.: Winner Takes All - "победитель получает все"), при котором в каждой фиксированной ситуации активизируется только один нейрон, а остальные пребывают в состоянии покоя.

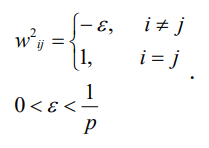
Обучение сети Хемминга заключается в предварительном расчете значений весовых коэффициентов и порогов активации нейронов по следующим правилам. Весам первого слоя присваиваются значения, рассчитываемые по выражению:

****

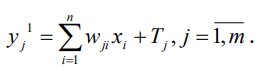
при этом пороги активационных функций равны:



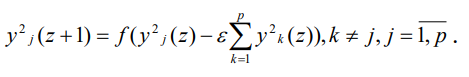
Как отмечалось ранее, веса связи i-го и j-го нейронов второго слоя равны:



После обучения сети сеть способна распознавать вектора, подаваемые на её вход. На входы сети подается неизвестный вектор X, исходя из которого, рассчитываются состояния нейронов первого слоя. Значения выходных сигналов нейронов определяются по формуле:



Эти сигналы становятся начальными состояниями нейронов второго слоя. Этот слой определяет "победителя", то есть нейрон, выходной сигнал которого близок к 1. Такой нейрон указывает на вектор образа с минимальным расстоянием Хемминга до входного вектора X. Нейрон победитель определяется итерационным процессом расчета состояний нейронов второго слоя за счет ослабления весами входных сигналов слоя:



Итерационный процесс (13) во втором слое завершается, когда активным остается только один нейрон (его выход имеет значение близкое к 1), тогда как остальные нейроны пребывают в близком к нулю состоянии.

Примером использование нейронной сети Хэмминга является идентификация качества продукции [[13](#_Источники.)]. Нейронная сеть успешно используется для контроля качества металлургической продукции на производственной линии.

# Интеграция нейронных сетей в ПИД-подобные регуляторы

## Преимущества и недостатки применения статических и динамических нейронных сетей

Нейронные сети эффективны за счет распараллеливания обработки информации и из-за способности самообучаться, т.е. создавать обобщения [[7](#_Источники.)]. Под термином обобщение понимается способность получать обоснованный результат на основании данных, которые не встречались в процессе обучения. Эти свойства позволяют нейронным сетям решать сложные задачи, которые на сегодняшний день считаются трудноразрешимыми. Однако на практике при автономной работе нейронные сети не могут обеспечить готовые решения. Их необходимо интегрировать в сложные системы. В частности, комплексную задачу можно разбить на последовательность относительно простых, часть из которых может решаться нейронными сетями.

Использование нейронных сетей придает информационным системам, в которых они используются, различные особенности.

1. Нелинейность — Искусственные нейроны могут быть линейными и нелинейными. Нейронные сети, построенные из соединений нелинейных нейронов т.е. использующих нелинейные функции преобразования, сами являются нелинейными. Нелинейность является чрезвычайно важным свойством, особенно если сам физический механизм, отвечающий за формирование входного сигнала, тоже является нелинейным.

Плюсы:

* Нелинейность позволяет эффективно моделировать сложные системы, где нелинейные зависимости играют ключевую роль.

Минусы:

* Нелинейные системы нуждаются в сложной настройке и могут некорректно работать в случае, если настроены неправильно.
* Усложняется интерпретация и отладка нелинейной системы.
* Нелинейные нейронные сети более ресурсозатратны при обучении, чем линейные нейронные сети.

1. Отображение входной информации в выходную — Сети обучаются на примерах, сопоставляя входные и выходные данные, что напоминает непараметрическое обучение без строгих статистических моделей.

Плюсы:

* Позволяет нейронной сети обучаться на данных без необходимости сложного математического моделирования процесса.
* Ускоряет разработку, так как исключает этапы проектирования аналитических моделей.
* Способствует повышению точности в сложных системах за счет обучения на реальных данных.

Минусы:

* Высокая чувствительность к качеству данных, так как ошибки в обучающих данных приводят к ошибкам на выходных сигналах.

1. Адаптивность — Нейронные сети обладают способностью адаптировать свои веса к изменениям среды. В частности, нейронные сети, обученные действовать в определенной среде, могут быть легко переучены для работы в условиях колебаний параметров среды. Более того, для работы в нестационарной среде (где статистика изменяется с течением времени) могут быть созданы нейронные сети, изменяющие синаптические веса в реальном времени. Чем выше адаптивные способности системы, тем более устойчивой будет ее работа в нестационарной среде. При этом хотелось бы заметить, что адаптивность не всегда ведет к устойчивости; иногда она приводит к совершенно противоположному результату. Адаптивная система с параметрами, быстро изменяющимися во времени, может также быстро реагировать и на посторонние возбуждения, что вызовет потерю производительности. Для того чтобы использовать все достоинства адаптивности, параметры системы должны быть достаточно стабильными, чтобы можно было не учитывать помехи, и достаточно гибкими, чтобы обеспечить реакцию на существенные изменения среды. Эта задача обычно называется дилеммой стабильности-пластичности.

Плюсы:

* Способность подстраиваться к динамически меняющимся характеристикам объекта управления.
* Эффективность в условиях, где свойства окружающей среды имеют динамические свойства, изменяющиеся во времени.

Минусы:

* Недостаточная скорость адаптации будет приводить к некорректной работе, так как система не будет успевать адаптироваться к изменениям окружающей среды.
* Слишком высокая скорость адаптации будет приводить к слишком высоким изменениям параметров системы относительно среды, что будет приводить к колебаниям системы.

1. Очевидность ответа — В задачах классификации сети могут накапливать информацию, чтобы повышать достоверность результата и избегать ошибок.

Плюсы:

* Упрощает отладку и работу с регулятором благодаря легкости интерпретации и предсказуемости поведения во время реагирования на условия.

Минусы:

* В общем случае нейронная сеть рассматривается как черный ящик – проблематично рассмотреть процесс принятия решения нейронной сетью.

1. Контекстная информация — Знания хранятся в структуре сети, а взаимодействие нейронов обеспечивает работу с контекстом данных.

Плюсы:

* Учет предыдущих состояний системы позволяет повысить точность управления.
* Позволяет учитывать инерционные свойства системы или окружающей среды.
* Делает поведение системы более стабильным в условиях внезапных изменений.

Минусы:

* Усложняет модель и требует ресурсов для хранения и обработки данных.
* Слишком большое количество данных будет делать каждое изменение нейронной сети все менее и менее значимым.

1. Отказоустойчивость — Нейронные сети устойчивы к повреждениям, поскольку информация хранится распределённо и незначительные повреждения не приводят к критическим сбоям.

Плюсы:

* Система остается работоспособной даже при частичном выходе из строя различных компонентов.

Минусы:

* Требуются дополнительные ресурсы для архитектуры, что увеличивает сложность и стоимость реализации.
* Диагностика неисправностей становится сложнее.

1. Масштабируемость — Сети легко расширяются, особенно с использованием технологий интегральных схем (VLSI), и могут быстро решать задачи за счёт параллельной структуры.

Плюсы:

* Возможность применения к более крупным системам или сложным задачам, увеличивая размер сети и объем данных.
* Возможность интеграции с другими системами и независимой замене задействованных компонентов.

Минусы:

* Риск ухудшения производительности при масштабируемости в случае, если архитектура не оптимизирована.
* Увеличение объема вычислений требует более мощного оборудования.

1. Единообразие и гибкость — Нейронные сети являются универсальным инструментом, подходящим для различных областей и задач благодаря общей структуре и стандартным алгоритмам обучения.

Плюсы:

* Одна архитектура может быть использована для различных объектов управления, что упрощает разработку.
* Удобство изменения структуры сети или ее параметров для адаптации к новым задачам.

Минусы:

* Упрощение системы может снизить универсальность и производительность системы.
* Требуется тщательная настройка, что бы гибкость не привела к избыточной сложности.

## ПИД-регулятор с блоком автонастройки на основе нейронной сети

Типовая структура системы автоматического регулирования с ПИД-регулятором и нейронной сетью в качестве блока автонастройки[[2](#_Источники.)] показана на рисунке.

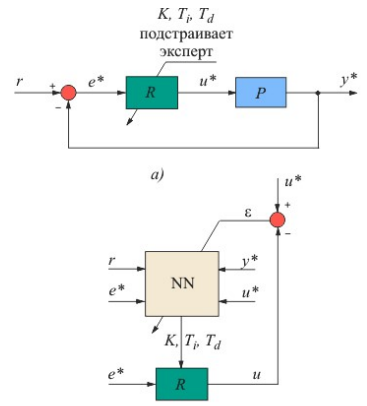


**рис. 12**

Рис. - Структура ПИД-регулятора с блоком автонастройки на основе нейронной сети NN

Нейронная сеть NN в данной структуре выполняет роль функционального преобразователя, который для каждого набора сигналов r, e, u, y вырабатывает коэффициенты ПИД-регулятора , стремясь минимизировать ошибку выходного сигнала от сигнала задатчика. Наиболее сложной в проектировании регуляторов с нейронной сетью является процедура обучения. «Обучение» состоит в идентификации неизвестных параметров нейронов  Для обучения нейронной сети, в системах управления электроприводом, обычно используют методы градиентного поиска минимума критериальной функции , зависящей от параметров нейронов. Процесс поиска является итерационным, на каждой итерации находят все коэффициенты сети, сначала для выходного слоя нейронов, затем предыдущего, и так до первого слоя (метод обратного распространения ошибки). Помимо этого, используются и другие методы поиска минимума, в том числе генетические алгоритмы, метод моделирования отжига, метод наименьших квадратов.

Процесс обучения нейронной сети показан на рисунке.



**рис. 13**

Эксперту предоставляют возможность подстраивать параметры регулятора K Ti Td в замкнутой системе автоматического регулирования при различных входных воздействиях r(t). Предполагается, что эксперт умеет это делать с достаточным для практики качеством. Временные диаграммы (осциллограммы) переменных , полученные в системе, подстраиваемой экспертом, записываются в архив и затем подаются на нейронную сеть, подключенную к ПИД-регулятору.

Нейронная сеть настраивается таким образом, чтобы минимизировать погрешность ε между сигналом u\*, полученным с участием эксперта, и сигналом u, полученным в процессе обучения нейронной сети. После выполнения процедуры обучения параметры нейронной сети заносятся в блок автонастройки. В соответствии с теорией нейронных сетей, обученная нейронная сеть должна вести себя так же, как и эксперт, причем даже при тех входных воздействиях, которые не были включены в набор сигналов, использованных при обучении. Для этого работоспособность нейронной сети проверяют новым набором данных, не используемых для ее обучения. Если ошибка на проверочных данных не превышает допустимую, нейронную сеть считают обученной. Длительность процесса обучения и его качество являются основными сложностями, которые не позволяет широко использовать метод нейронных сетей в ПИД-регуляторах. В то же время, настроить нейронную сеть и проверить ее на отсутствие ошибок часто бывает намного быстрее и легче, чем составить адекватную модель электропривода и объекта управления и настроить на ее основе ПИД-регулятор. Причем далеко не всегда настройка регулятора на основе положений теории систем автоматического управления может быть оптимальной. К другим недостаткам нейронных сетей относится невозможность предсказания погрешности регулирования для входных воздействий, входящие в набор обучающих сигналов; отсутствие критериев выбора количества нейронов в сети, длительности обучения, диапазона и количества обучающих воздействий.

## ПИД-регулятор с блоком автонастройки на основе нейронной сети

# Прикладное применение нейронных сетей в ПИД подобных регуляторах

## Настройка ПИД-контроллера для модели квадрокоптера

Для постройки регулятора, в котором используется нейронная сеть, для подбора его коэффициентов не требуется формулировать четкие правила, достаточно, чтобы человек определенное количество раз настроил регулятор в процессе обучения нейросети.

Статья описывает интеграцию ПИД-подобных контроллеров в схему управления квадрокоптеров, используемых в задачах транспортировки и наблюдения. Основной задачей при разработке систем управления квадрокоптеров является решение проблемы стабильности их полета без постоянного участия человека. Основные проблемы перечислены ниже:

1. Пере- и недорегулирование:
   * Недорегулирование происходит, когда скорость исправления ошибки контроллером слишком мала по сравнению с непосредственной ошибкой. В результате квадрокоптер очень медленно достигает устойчивого состояния, что может привести к затяжной корректировке и замедленной реакции на команды управления.
   * Перерегулирование возникает в том случае, когда скорость исправления ошибки слишком велика по сравнению с ошибкой. В этом случае квадрокоптер может быть нестабилен, система демонстрирует сильные колебания, что делает управление менее точным и может даже привести к потере контроля управлению.
2. Статические ошибки: в некоторых случаях квадрокоптер не способен достичь устойчивого состояния без осцилляций, поскольку учитывает только текущую ошибку без учета накопленных отклонений. В результате квадрокоптер может испытывать постоянное отклонение от желаемого состояния (например, по высоте или углу наклона).

Решение данных проблем может достигаться при помощи непосредственного использования ПИД-подобных контроллеров.

В статье подробно рассматривается влияние P, PD и PID контроллеров на стабильность полета квадрокоптера. Исследование направлено на сравнение этих трех типов контроллеров с точки зрения их влияния на динамику и стабильность управления квадрокоптером, а также на поиск оптимальных настроек с помощью алгоритма роя частиц (PSO).

P-контроллер (пропорциональный) регулирует систему, основываясь только на текущей ошибке — разнице между желаемым и фактическим значением параметра. Он использует пропорциональный коэффициент Kp​, который определяет, насколько сильно система будет реагировать на ошибку.

Из преимуществ такого контроллера следует отметить:

* Простоту реализации
* Малое время реакции на ошибки

К недостаткам следует отнести:

* Отсутствие возможности устранить статические ошибки: P-контроллер не может устранить постоянные смещения, так как его действие зависит только от текущей ошибки. Это приводит к тому, что квадрокоптер может испытывать остаточные ошибки (например, отклонения по высоте или углам).
* Перерегулирование и колебания: P-контроллер может вызвать значительные колебания вокруг заданного значения, поскольку он реагирует только на текущую ошибку без учета скорости изменения или накопленной ошибки.

Исследования показали, что P-контроллер не способен обеспечить устойчивое состояние квадрокоптера. На практике это приводит к тому, что система либо не достигает требуемого состояния, либо это происходит медленно и с большими отклонениями. В статье отмечается, что траектория полета квадрокоптера с P-контроллером была нестабильной, что подтверждается тем, что ошибки во времени сильно колебались, и система не могла достичь устойчивого состояния в течение всего эксперимента.

PD-контроллер добавляет к пропорциональной составляющей дополнительный дифференциальный элемент, который учитывает скорость изменения ошибки. Он включает в себя коэффициенты Kp ​ (пропорциональный) и Kd (дифференциальный), где последний отвечает за учет скорости изменения ошибки во времени.

Из преимуществ:

* Уменьшение колебаний: добавление дифференциальной составляющей (D) помогает сглаживать колебания и снижает вероятность перерегулирования, что улучшает поведение системы при быстрых изменениях.
* Более быстрое приближение к устойчивому состоянию, чем у P-контроллера.

Из недостатков:

* Все еще не устраняет **остаточные ошибки**: PD-контроллер также не учитывает накопленную ошибку, что может привести к тому, что система не достигнет точно заданного значения.
* Более сложная настройка: необходимо подобрать оба коэффициента Kp​ и Kd​, что усложняет процесс оптимизации системы.

Использование PD-контроллера значительно улучшает устойчивость квадрокоптера по сравнению с P-контроллером. Система быстрее реагировала на отклонения и демонстрировала меньшие колебания. Однако на долгосрочных временных промежутках система все равно не была полностью стабильной и не всегда могла достигать идеально устойчивого состояния, особенно по таким осям, как высота (Z) и угол рыскания (ψ).

PID-контроллер является наиболее сложным из трех, так как он объединяет в себе пропорциональную, интегральную и дифференциальную составляющие. Интегральная составляющая (I) учитывает накопленную ошибку за все время и добавляет её к управлению, что позволяет системе компенсировать постоянные отклонения, с которыми не справляются P и PD контроллеры.

Из преимуществ:

* Полная компенсация ошибок: за счет интегральной составляющей PID-контроллер устраняет статические отклонения, что обеспечивает квадрокоптеру достижение точного положения и стабильного состояния.
* Снижение колебаний и перерегулирования: дифференциальная составляющая сглаживает изменения и предотвращает колебания, а пропорциональная часть обеспечивает быструю реакцию на отклонения.
* Высокая точность: за счет сочетания всех трех составляющих, PID-контроллер способен обеспечить квадрокоптеру максимально устойчивый полет.

Из недостатков:

* Сложность настройки: PID-контроллер требует настройки трех коэффициентов Kp​, Ki​, Kd​, что значительно увеличивает сложность процесса калибровки системы.
* Возможность чрезмерной коррекции: если коэффициенты настроены неправильно, могут возникать ситуации перерегулирования или, наоборот, чрезмерного затухания, что отрицательно скажется на времени достижения стабильности.

В результате экспериментов PID-контроллер показал лучшие результаты среди всех рассмотренных. Он успешно стабилизировал квадрокоптер по всем осям (ролл, тангаж, рыскание и высота). Ошибки в управлении квадрокоптером значительно снизились, и система быстро достигала устойчивого состояния. PID-контроллер смог компенсировать все статические ошибки и минимизировать колебания при изменении положения квадрокоптера, что было подтверждено стабильной траекторией полета в течение всего времени эксперимента.

В заключение в статье говориться, что:

• P-контроллер недостаточен для поддержания стабильного полета квадрокоптера, поскольку не может справиться с остаточными ошибками и вызывает значительные колебания.

• PD-контроллер улучшает стабильность за счет учета скорости изменения ошибки, но также не устраняет остаточные ошибки и требует точной настройки двух коэффициентов.

• PID-контроллер является наиболее эффективным для стабилизации квадрокоптера. Он компенсирует как мгновенные, так и накопленные ошибки, что делает его оптимальным для использования в критических задачах, требующих высокой точности и стабильности.

# Прикладное применение нейронных сетей в регуляторах

## Подготовка предметной области

### Поиск предметной области

Для демонстрации прикладного применения нейросетевых регуляторов и сравнения их работы с классическими регуляторами была выбрана область разработки медицинского оборудования, а точнее была выбрана модель работы аппарата искусственной вентиляции легких. Данная модель позволяет продемонстрировать особенности того, с чем приходится сталкиваться при разработке классических и нейросетевых регуляторов.

### Описание предметной области

Как уже было указано, аппарат искусственной вентиляции легких подходит для демонстрации работы регуляторов. Поведение легких представляет собой систему, которая изменяется динамически во времени. Кроме этого, каждая итерация вдоха легких уникальна и отличается от других. Дыхание также зависит от некоторых внешних параметров, которые также случайно могут изменяться во времени.

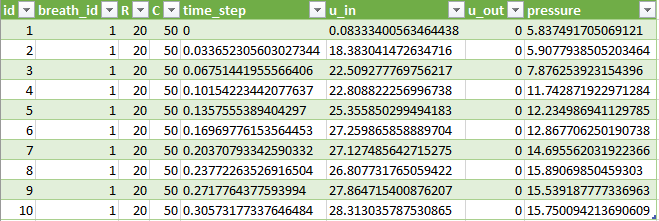
В качестве основы был взят датасет из соревнования «Google Brain - Ventilator Pressure Prediction» [[14](#_Источники.)]. Соревнование направлено на разработку наиболее точной и эффективной нейросетевой модели, которая должна предсказывать давление в легких пациента исходя из входных параметров. В контексте исследования упрощенная модель машинного обучения, обученная на данных из датасета, будет использоваться не как инструмент для предсказания, а как модель, имитирующая поведение легких.

### Описание датасета

Датасет представляет собой набор из файлов в формате csv. Представлена обучающая выборка, тестовая выборка и пример итогового файла с предсказаниями, который в нашем контексте не пригодится.

Выборка представляет собой таблицу, состоящую из восьми столбцов. Обучающая выборка состоит из 6 000 000 строк, тестовая выборка состоит из 4 000 000 строк.

Данные представлены в следующем формате.



**рис. 14**

* id – целочисленная величина, отображающая номер записи/строки
* breath\_id – целочисленная величина, отображающая номер вдоха. Каждый вдох зафиксирован несколькими записями.
* R – численная константа, которая обозначает атрибут легкого, указывающий, насколько ограничены дыхательные пути (cmH2O/L/S). Физически это изменение давления на изменение потока (объема воздуха за время). Интуитивно можно представить себе надувание воздушного шара через трубочку. Величина может измениться, уменьшение диаметра трубочки, увеличивает сопротивление, которое будет испытывать воздух при надувании шарика.
* C – численная константа, которая обозначает атрибут легкого, указывающий, насколько податливо легкое (mL/cmH2O). Физически это изменение объема на изменение давления. Интуитивно можно представить тот же пример с воздушным шаром. Можно изменять податливость легкого, изменяя толщину материала, из которого состоит воздушный шарик. Мы можем изменить, Cизменив толщину латекса воздушного шара. Чем выше податливость шарика, тем проще его надувать.
* time\_step – временная величина, которая указывает на то, сколько времени прошло с начала нового вдоха. Каждый вдох начинается с нулевого временного значения.
* u\_in – численная величиная, отображающая входное воздействие устройства искусственной вентиляции легких. Величина представлена в виде некоторого электронного значенияБ и обозначает мощность, с которой работает прибор, а не поток воздуха.
* u\_out – бинарная величина, отображающая работу выходного клапана устройства. 0 обозначает закрытый выходной клапан, 1 обозначает открытый выходной клапан. Открытие клапана регулирует выдох легких.
* pressure – численная величина, которая отображает давление внутри легких.

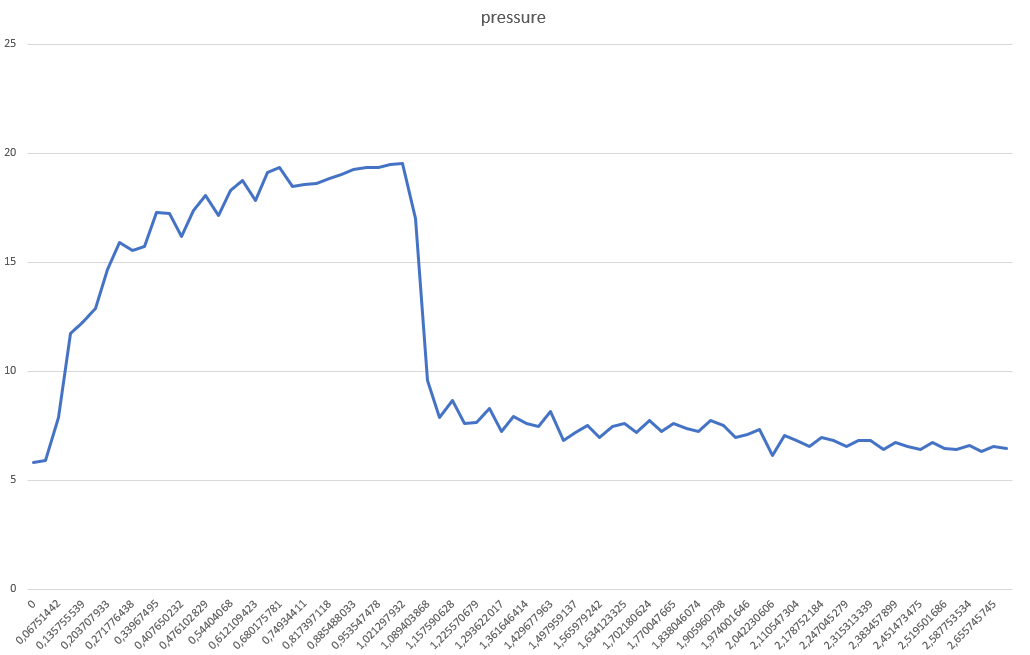
### Анализ данных

Был проведен анализ данных, содержащихся в датасете. Были подсчитаны минимальные, максимальные и средние значения каждой величины. Было установлено, что

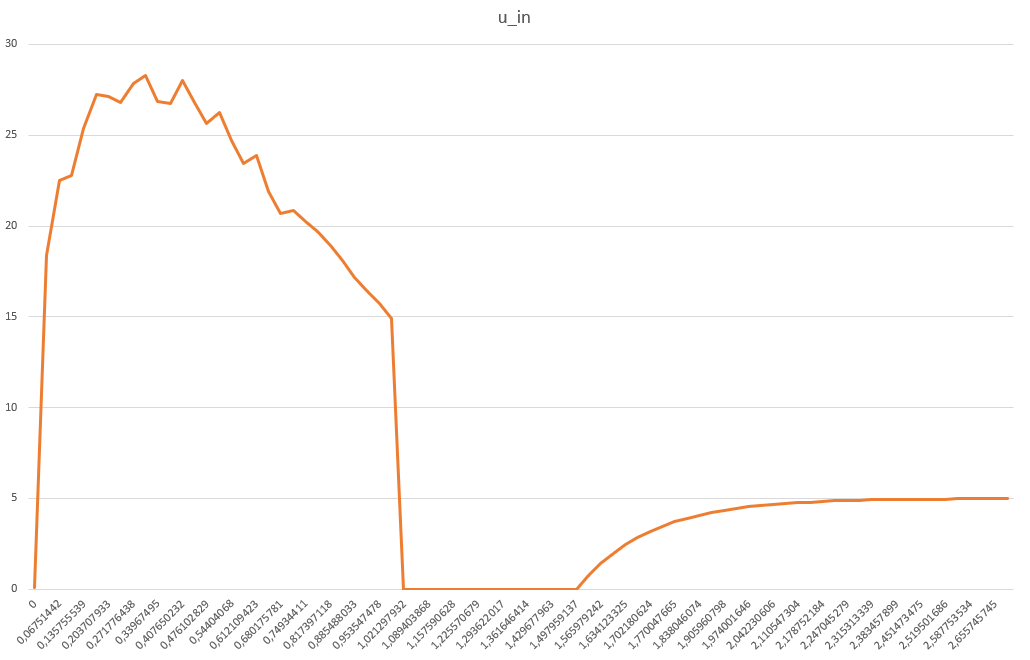
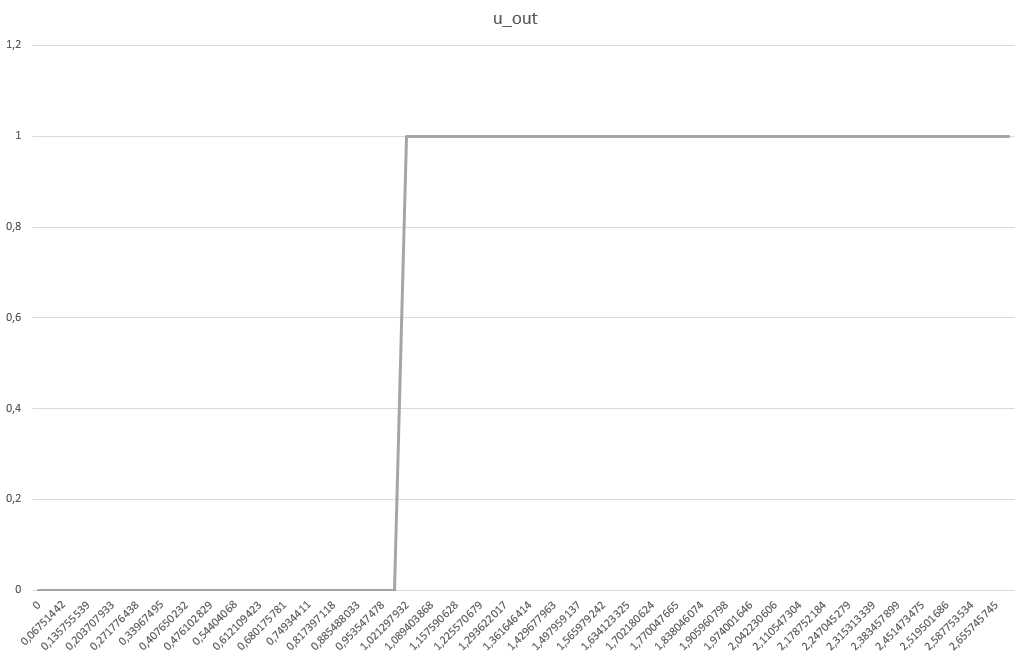
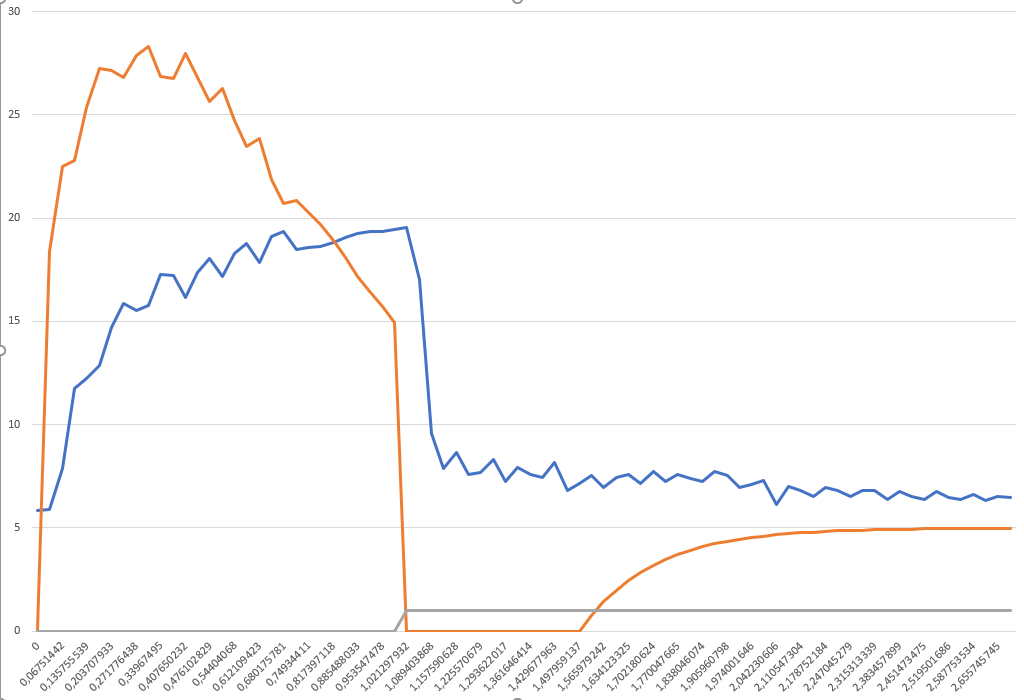
* R принимает значения в диапазоне от 5 до 50.
* R имеет среднее значение, равное 26,85
* C принимает значения в диапазоне от 10 до 50.
* C имеет среднее значение, равное 26,39
* time\_step принимает максимальное значение, равное 2.7387404441833496
* u\_in принимает значения в диапазоне от 0 до 100.
* u\_in имеет среднее значение, равное 7.46
* u\_out имеет среднее значение, равное 0,62
* pressure принимает значения в диапазоне от 1.895744294564641 до 64.8209917386395.
* pressure имеет среднее значение 11.34.

Также, расположив данные давления pressure и управляющих воздействий u\_in и u\_out на временной шкале, мы сможем проанализировать динамику одной итерации вдоха-выдоха.

Динамика давления:

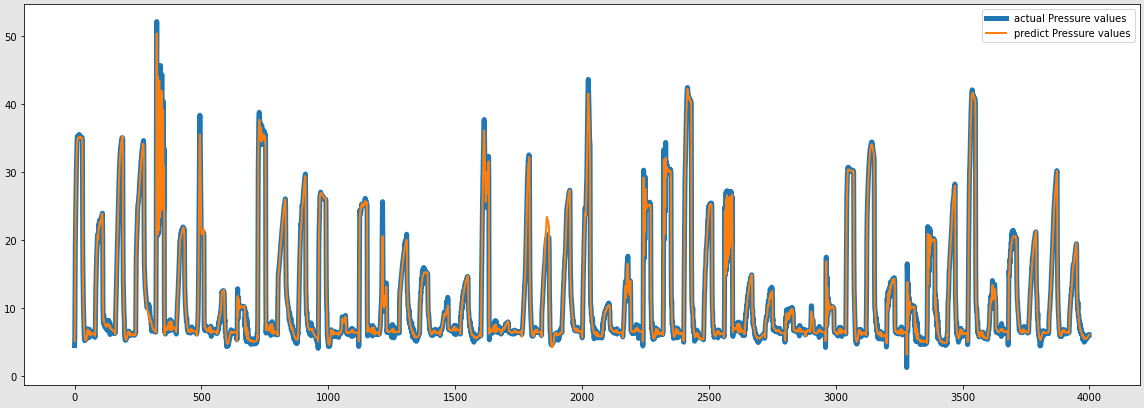
**рис. 15**

Динамика входного воздействия:

**рис. 14**Динамика выходного воздействия:****  
**рис. 15**Динамика одной итерации:  


**рис. 16**

Как мы видим, исходя из двух диаграмм, каждая итерация состоит из двух частей – вдоха и выдоха. Во время вдоха устройство нагнетает воздух, стремясь к некоторому значению, а выходной клапан закрыт и давление в легких растет. Во время выдоха выходной клапан открывается, давление в легких падает. Устройство в начале выдоха выключается, но потом начинает нагнетать некоторое количество воздуха для его замены внутри легких. Стоит отметить, что этапы вдоха-выдоха довольно жестко зафиксированы по временной шкале.

Динамика нескольких итераций:  


**рис. 17**

Если посмотреть на динамику нескольких вздохов на временной шкале, то можно увидеть, что большинство вдохов относительно сильно отличаются друг от друга. Однако, определяя упрощенную модель легких можно сказать, что большинство «здоровых» вздохов стремятся к значению давления в 30-35 единиц давления.

Соответственно мы разделим нашу выборку на два типа вдохов – «здоровые», у которых амплитуда достигает 30-35 единиц и «нездоровые» вдохи, значения которых отличаются в ту или иную сторону.

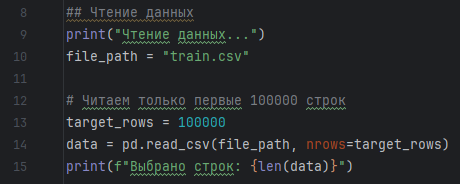
## Формирование модели легкого

Как уже было указано, моделирование поведения легкого будет выполнять модель машинного обучения. Для обработки данных будут использоваться библиотека pandas для анализа данных и работы с данными в формате таблиц и библиотека numpy, которая позволяет обрабатывать многомерные массивы, заполненные большим количеством данных. В качестве модели машинного обучения была выбрана регрессионная модель случайного леса (Random Forest Regression) из инструментария библиотеки sklearn.

Прежде чем идти дальше, стоит сказать, что использование термина модели имеет два случая и два определения. В одном случае подразумевается модель машинного обучения – т.е. некоторая структура нейронной сети, которая способна обрабатывать определенные данные Во втором случае подразумевается модель контроллера, модель легких, модель Х, т.е. модель упрощенного поведения некоторого объекта или явления.

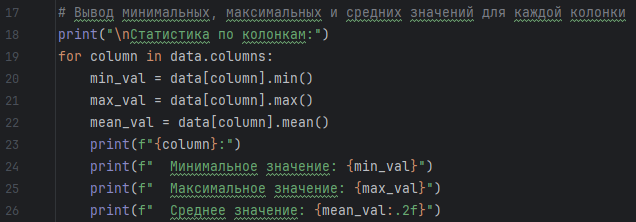
Случайный лес представляет собой модель машинного обучения, базирующийся на ансамблевом алгоритме, использующем деревья решений, каждое из которых строится на случайной выборке данных. Такая модель достаточно проста для базовой реализации модели легкого и обладает такими полезными свойствами, как гибкость к различным данным и устойчивость к переобучению. Кроме этого, модель случайного леса довольно просто масштабируется и позволяет пользователю доступно настраивать параметры весов.

Далее представлены фрагменты работы программы для обучения модели случайного леса [[Приложение 1](#_Приложение_1.)].

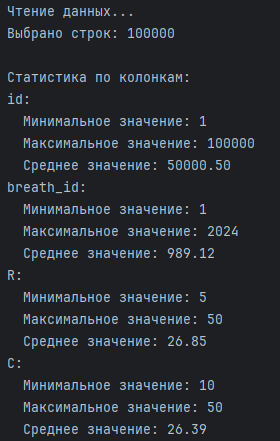


**рис. 18**

Чтение данных производится при помощи библиотеки pandas. В память компьютера загружается выборка из 100 000 строчек, после чего проводится анализ данных.



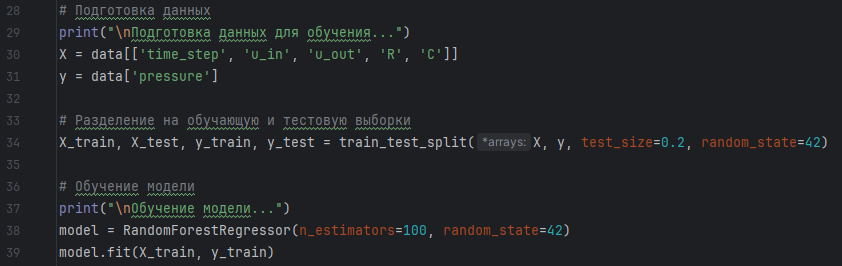
**рис. 19**



**рис. 20**

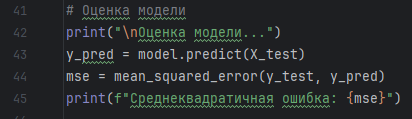
В терминал выводится данные по выборке для каждого параметра в столбце. Несмотря на то, что выводятся сразу все характеристики, важны только те, которые были указаны во время анализа данных.

Далее производится подготовка данных для дальнейшего их использования при обучении. После завершения подготовки данных, они сохраняются в памяти компьютера и производится настройка и дальнейшее обучение модели случайного леса на основе обучающей выборки.



**рис. 21**

После построения модели производится проверка ее точности предсказания при помощи тестирования на основе тестовой выборки.



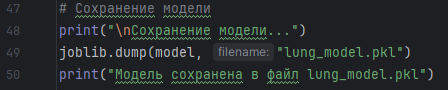
**рис. 22**



**рис. 23**

Можно увидеть, что точность построенной модели определяется по среднеквадратичной ошибке. Для построенной модели данное число равняется 17,84. При увеличении обучающей выборки с 100 000 до 500 000 удалось повысить точность модели до 17,0. Такую модель нельзя назвать достаточно точной для ее использования в предсказаниях. Однако такая точность полностью удовлетворяет базовым условиям для моделирования поведения легких у «больного пациента».

После тестирования модели, она сохраняется в файловой системе компьютера и может быть в дальнейшем использована для интеграции с моделью ПИД-контроллера без необходимости заново обучать модель при каждом запуске программы.



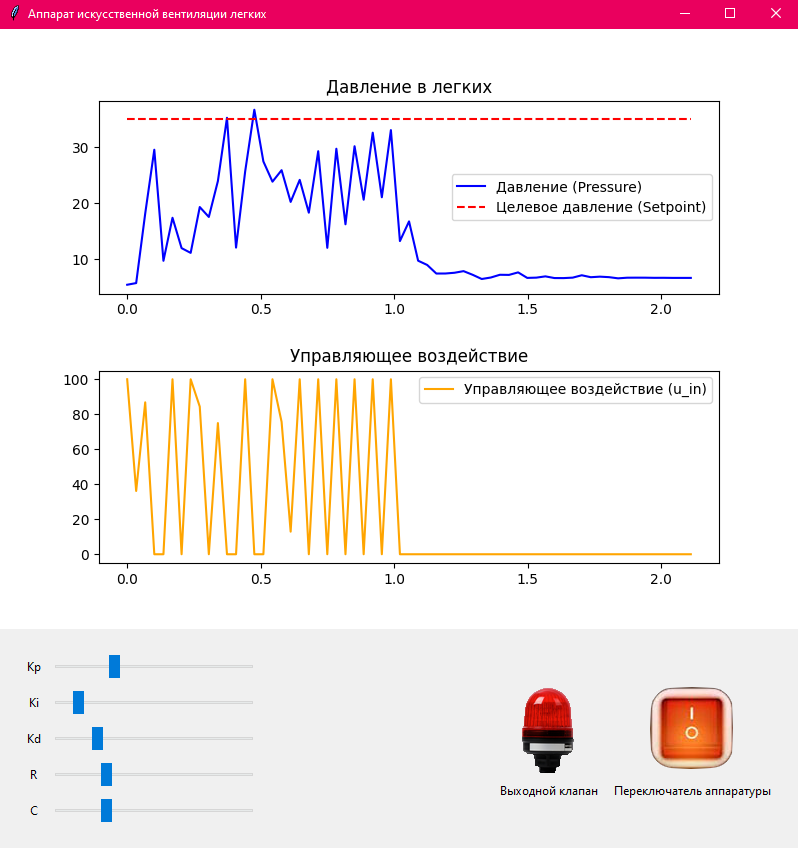
**рис. 24**

## Формирование модели ПИД-контроллера

Для разработки модели пид контроллера также использовались библиотека pandas для работы с табличными данными и файлами и библиотека numpy для работы с многомерными массивами.

Также, для визуализации модели использовались графические библиотеки. При помощи библиотеки matplotlib была реализована визуализация графиков изменения состояния модели. При помощи библиотеки tkinter были реализованы переключатели и кнопки настройки поведения модели.

### Описание интерфейса программы



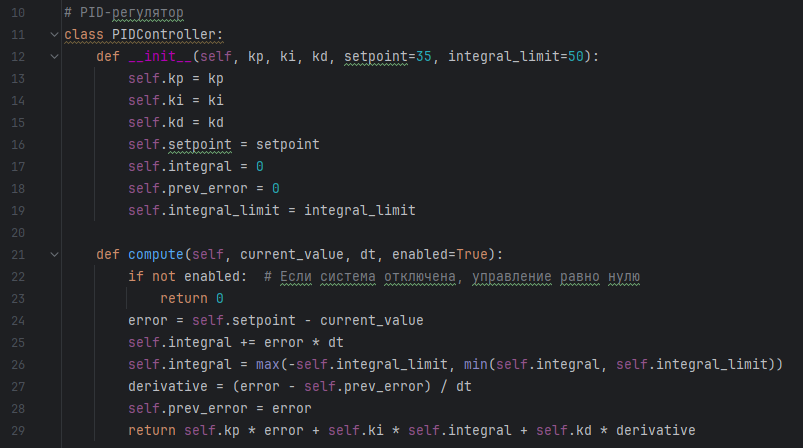
**рис. 25**

В верхней части окна представлены динамические временные диаграммы. Верхняя диаграмма отображает зависимость давления в легких от времени и целевое давление. Нижняя диаграмма отображает зависимость управляющего воздействия модели ПИД-контроллера от времени. В нижней части окна представлены ползунки, позволяющие регулировать коэффициенты параметров модели ПИД-регулятора и константы R и C, влияющие на поведение легких. Загорающаяся лампочка отображает работу выходного клапана и загорается в тот момент, когда клапан открывается. Переключатель управляет соединением модели ПИД-регулятора с моделью легкой и в случае выключение, входное воздействие на легкие приравнивается к нулю.

### Описание модели ПИД-контроллера.

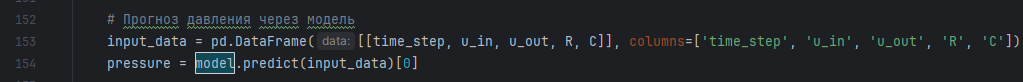
ПИД-контроллер имеет три параметра:

* **Пропорциональная составляющая** вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемого в данный момент времени. Он тем больше, чем больше это отклонение. Если входной сигнал равен заданному значению, то выходной равен нулю.
* **Интегрирующая составляющая** пропорциональна интегралу по времени от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку.
* **Дифференцирующая составляющая** пропорциональна темпу изменения отклонения регулируемой величины и предназначена для противодействия отклонениям от целевого значения, которые прогнозируются в будущем. Отклонения могут быть вызваны внешними возмущениями или запаздыванием воздействия регулятора на систему.

Каждый из параметров перед подачей выходного воздействия также умножается на соответствующий ему коэффициент, каждый из которых позволяет регулировать поведение ПИД-регулятор. В дальнейшем в структуру ПИД-регулятора может быть встроена нейросетевая модель, которая позволит имитировать поведение нейросетевого регулятора. Ниже представлен фрагмент кода, отвечающий за поведение модели ПИД-контроллера. 

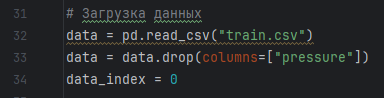
**рис. 26**

Управляющее воздействие контроллера подается на уже готовую модель поведения легких. Исходя из входного воздействия, модель легких прогнозирует результирующее давление.

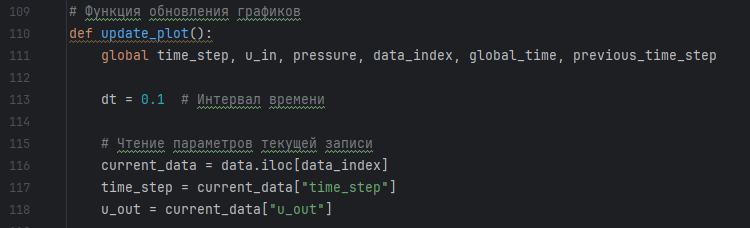


**рис. 27**

Непосредственно поведение легких моделируется путем зацикленной прокрутки содержимого датасета с тестовой выборкой. Так как в выборке находится 2000 итераций вдоха-выдоха, то для временной демонстрации работы модели каждая итерация будет уникальной. Данные, поступающие из построчного чтения датасета поступают в модель поведения легких вместе с управляющим воздействием контроллера.



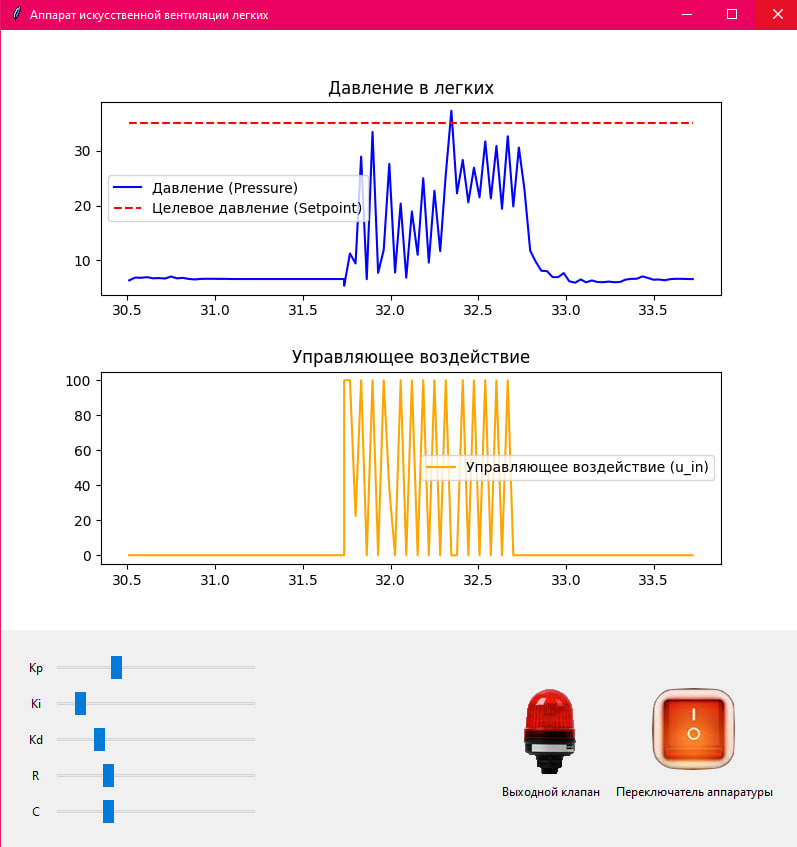
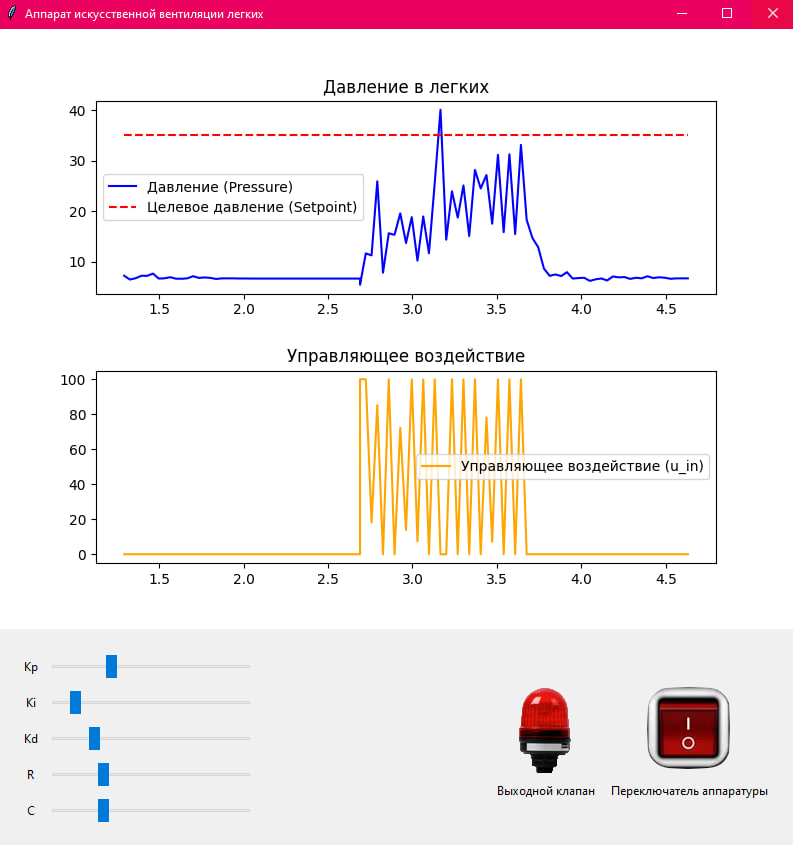
**рис. 28**



**рис. 29**

Управляющее воздействие контроллера подается на уже готовую модель поведения легких. Исходя из входного воздействия, модель легких прогнозирует результирующее давление.

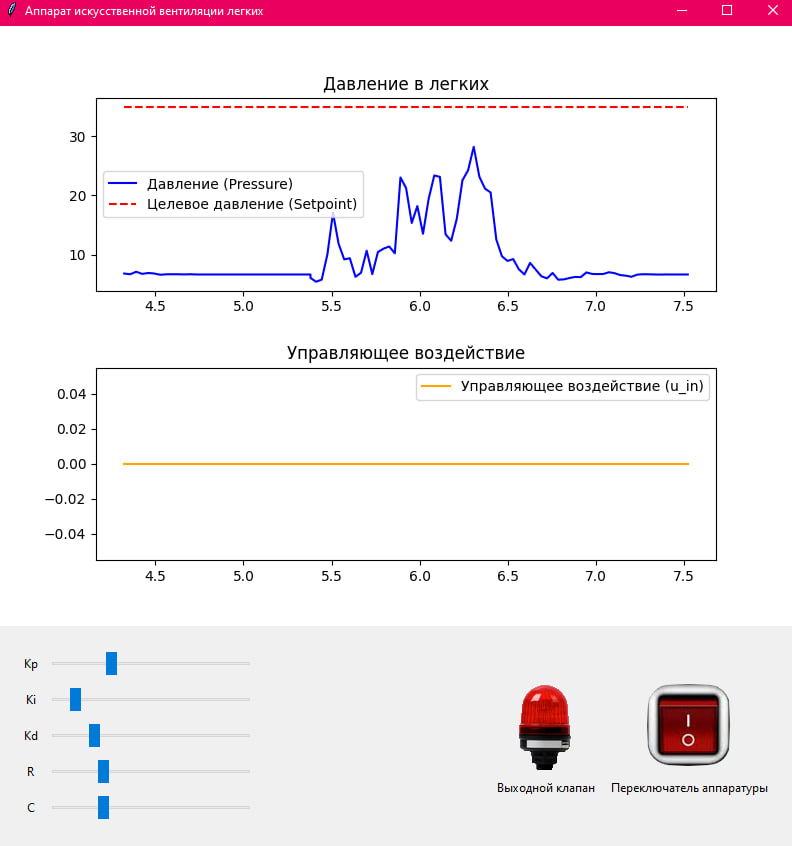
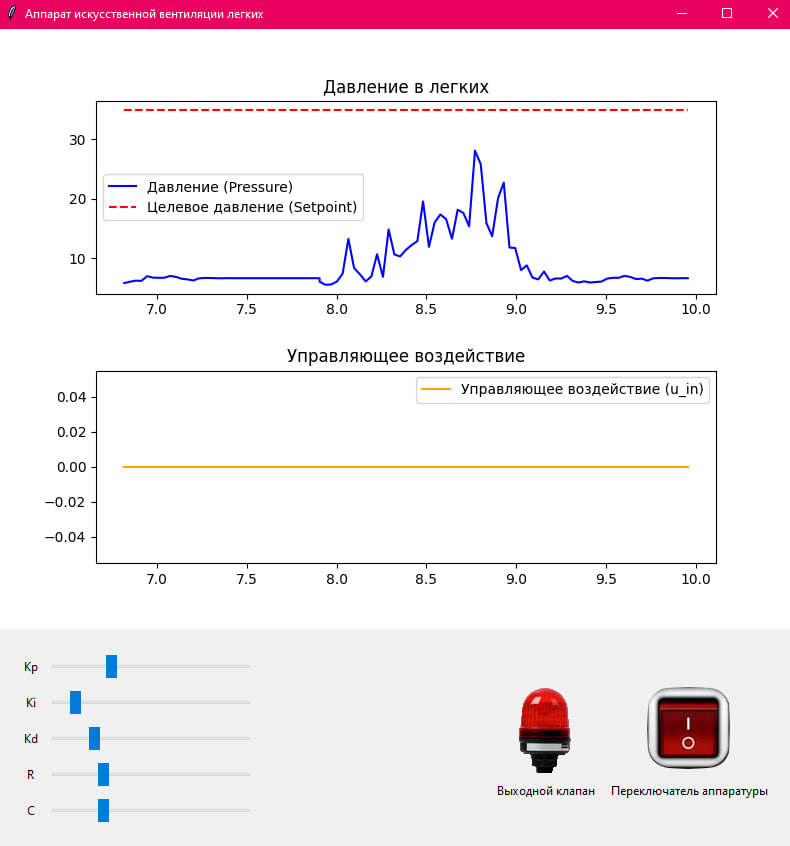
### Демонстрация работы общей модели.



**рис. 30 – рис. 31**

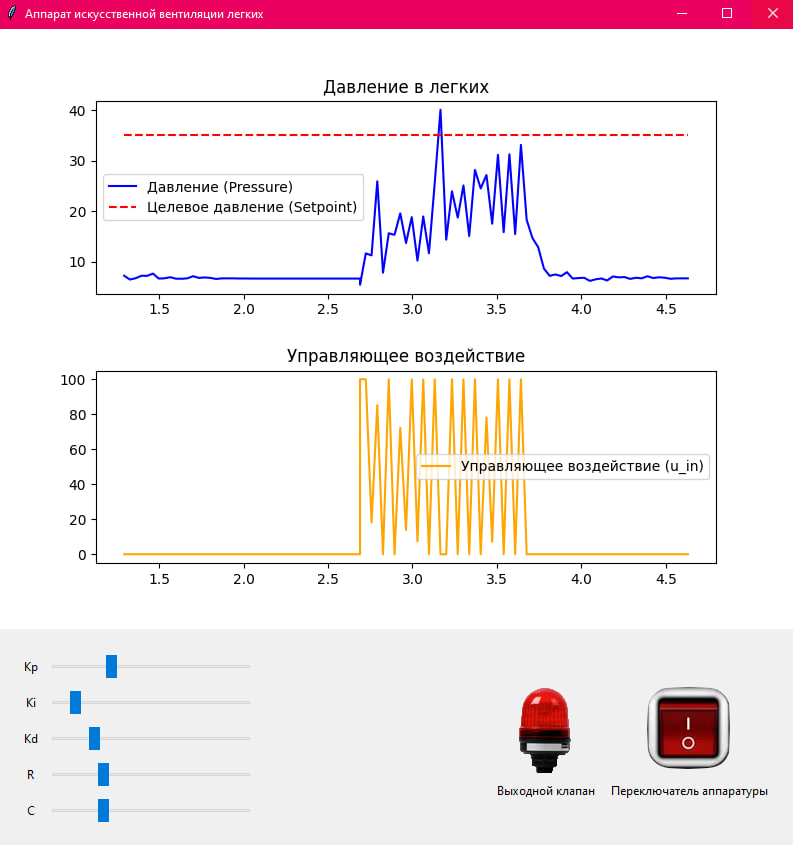
На изображениях представлена итерация вдоха-выдоха при воздействии на модель легкого управляющего воздействия модели ПИД-регулятора. ПИД-регулятор стремится выровнять давление при вдохе до эталонных 35 единиц. Стоит отметить, что при достижении требуемой отметки, управляющее воздействие становится нулевым, что можно наблюдать в виде «зубчатой» линии на обеих временных диаграммах.

В случае, если модель регулятора отключить от модели легкого пацента, у пациента существенная часть вдохов станет «болезнеными» и не будет достигать установленного порога.

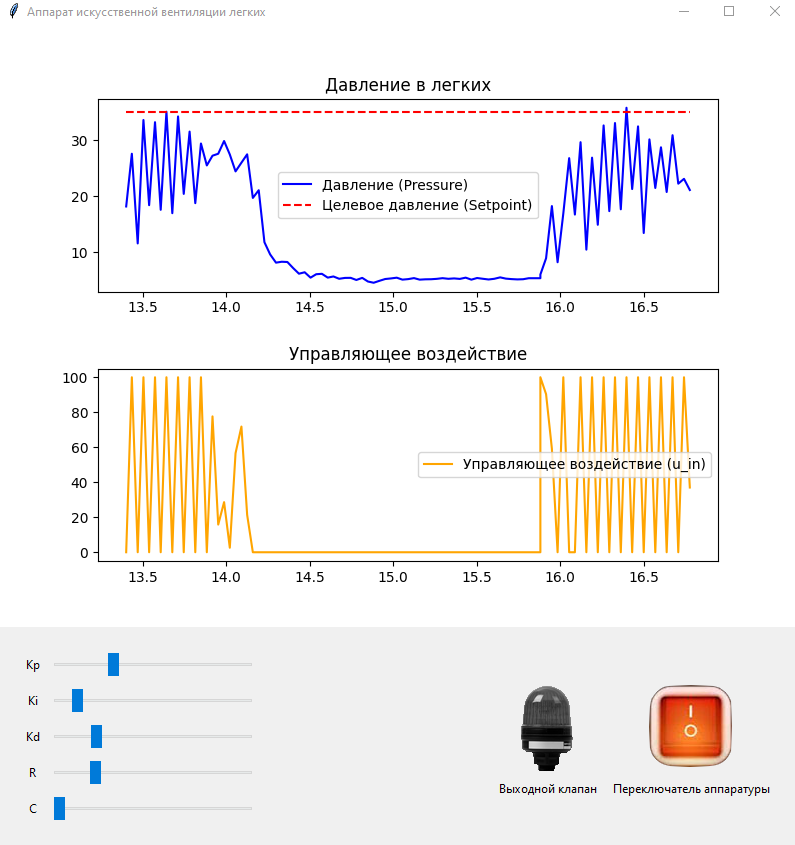
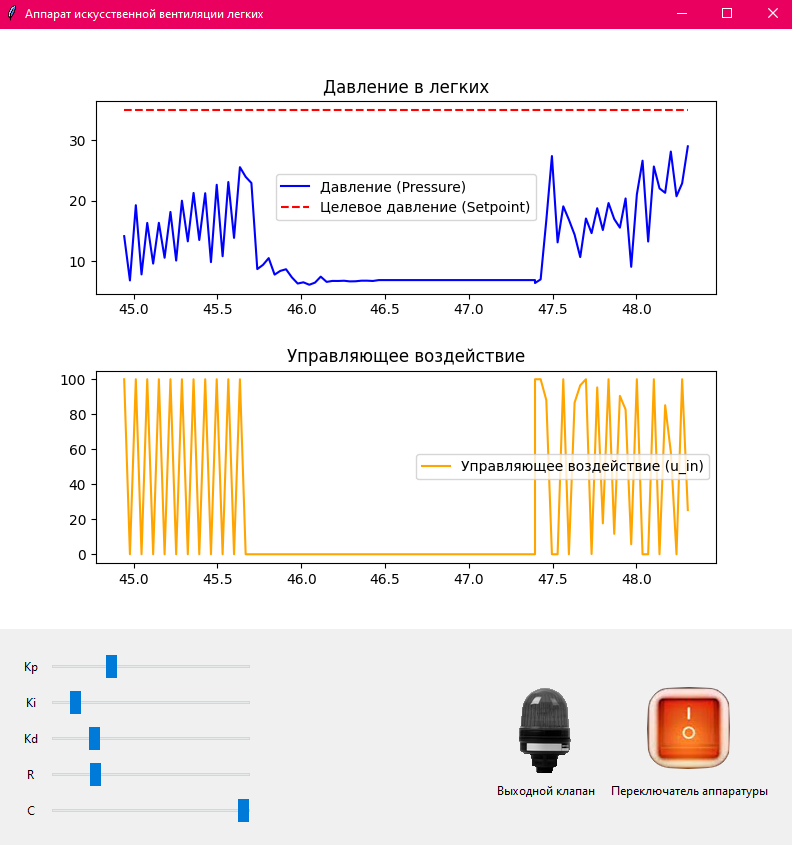
**рис. 32 – рис. 33**

На рисунках отмечены «болезненые вдохи», где управляющее воздействие регулятора приравнено к нулю. Стоит учесть, что в некоторых случаях пациент сможет сделать «здоровый» вдох.



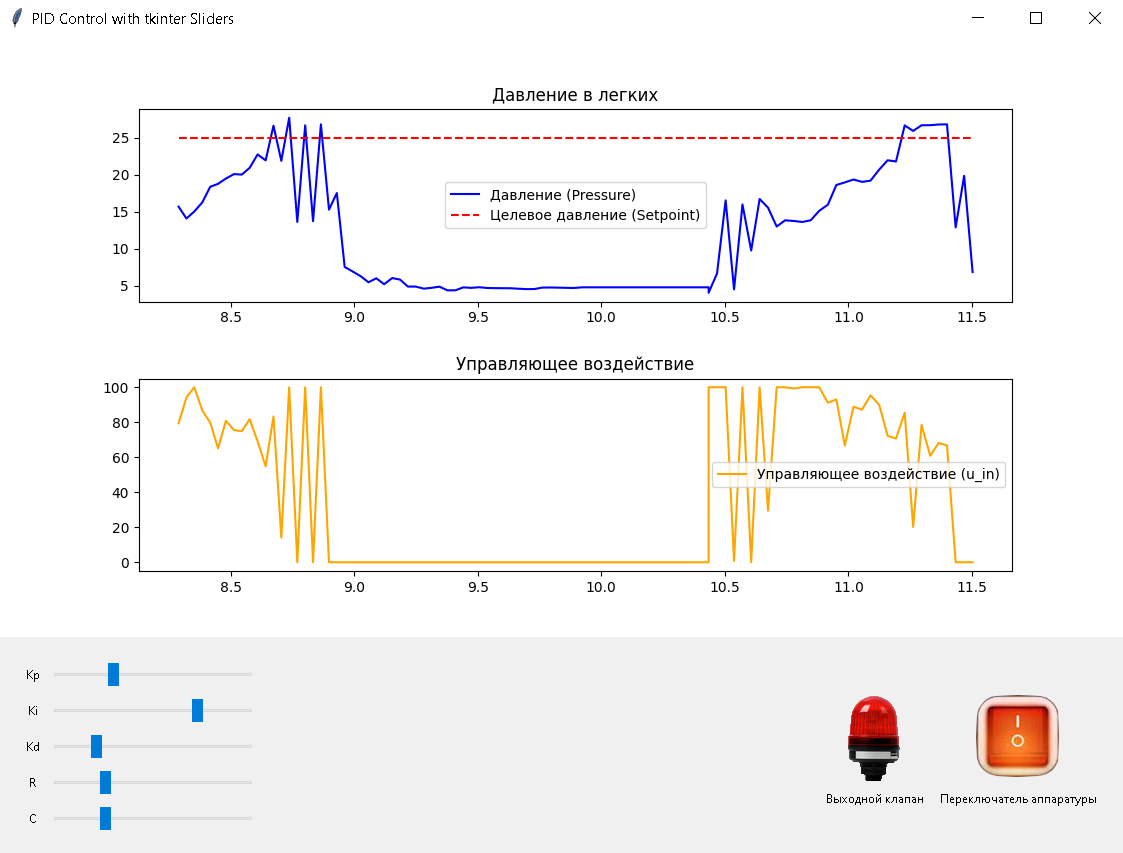
**рис. 34**

Изменение констант R и C также влияет на работу модели.

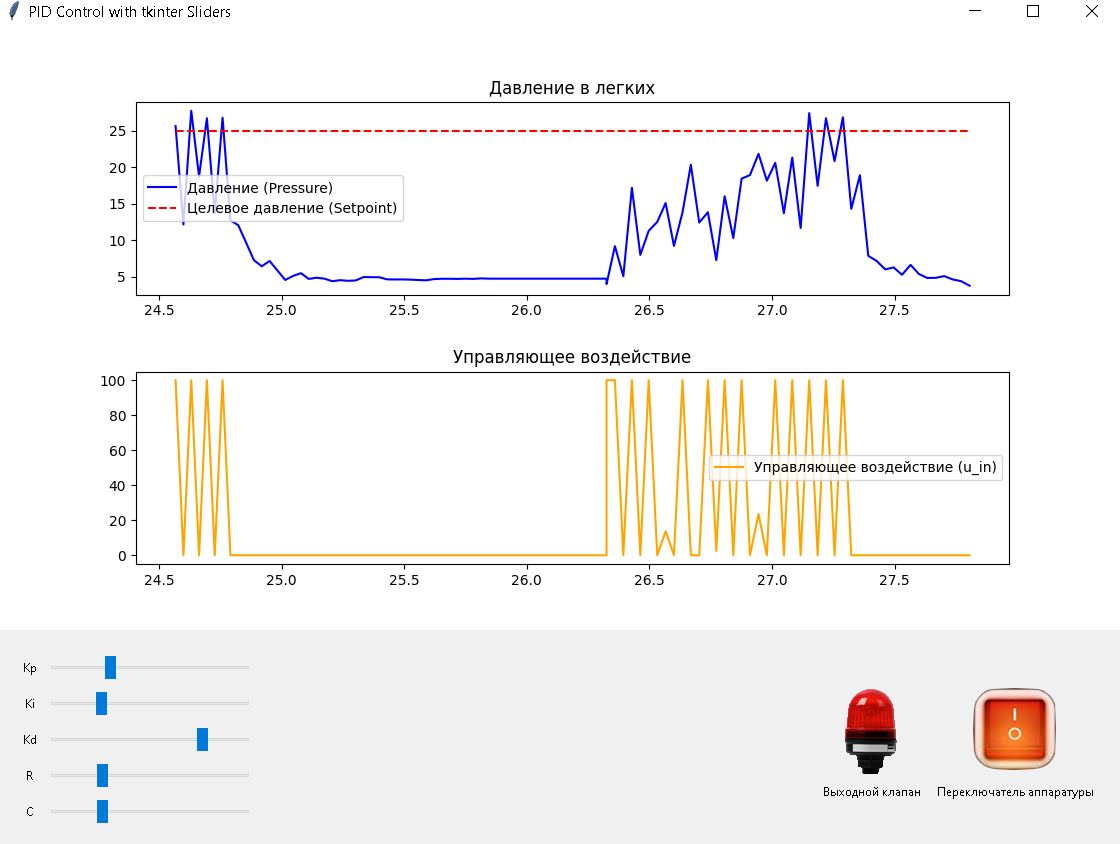
**рис. 35 – рис. 36**

Изменение коэффициентов контроллера также меняет поведение модели взаимодействия легких и контроллера. Так, повышение значений интегрального коэффициента позволяет значительно сгладить «зубья» на значениях, близких к целевым:



**рис. 37**

Увеличение дифференциальной характеристики заметно выравнивает итерацию вдоха, делая «шапку» более линейной:



**рис. 38**

Увеличение пропорциональной характеристики заметно выравнивает рост давления в легких на значениях от нуля до половины от целевых значений давления, выравнивая «зубья» на начале вдоха.



**рис. 39**

Важным уточнением будет то, что поведение общей модели зависит от нескольких факторов – управляющего воздействия контроллера, констант, влияющих на поведение легкого и характеристик отдельно взятой итерации вдоха выдоха. Для более детального рассмотрения можно, например, определить одну конкретную, «эталонную» итерацию вдоха-выдоха и зациклить на входе модели ее значения. Также, нужно учитывать тот факт, что сила входного воздействия регулятора ограничена. Работу модели можно сделать более подробной, сделав величину, к которой стремится регулятор при вдохе динамической и зависящей от динамики вдоха. Можно также повысить точность поведения модели легких и настроить обучающие метрики для того, что бы работа регулятора была более ощутимой.

## Формирование модели нейросетевого контроллера

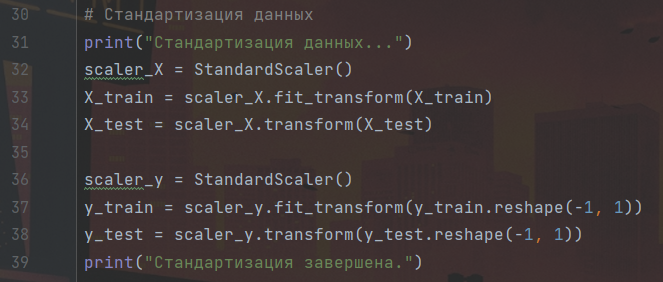
Для разработки модели нейросетевого контроллера также использовались библиотека pandas для работы с табличными данными и файлами. Для построения модели нейросетевого контроллера была использована библиотека PyTorch.

Библиотека машинного обучения PyTorch обладает следующими полезными свойствами:

1. Динамическое построение вычислительного графа: PyTorch позволяет создавать граф вычислений "на лету", обеспечивая гибкость в разработке моделей машинного обучения и упрощая процесс отладки.
2. Высокопроизводительные вычисления: Библиотека поддерживает ускорение с использованием графических процессоров (GPU) через интеграцию с CUDA, что существенно повышает скорость выполнения операций над большими объемами данных.
3. Интуитивный интерфейс и модульность: PyTorch предлагает понятный синтаксис и структуру, схожую с Python и NumPy, что облегчает разработку и применение нейронных сетей.

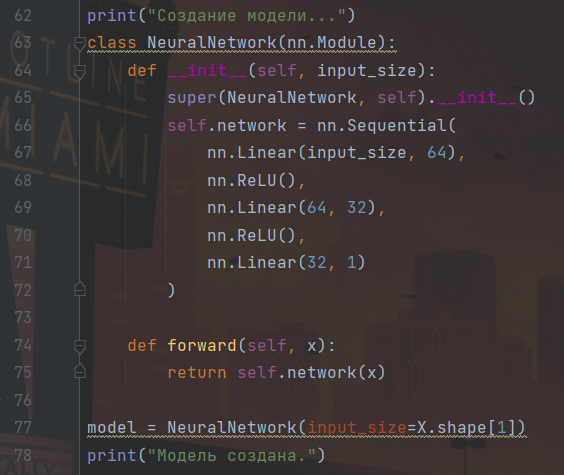
Для реализации модели нейросетевого контроллера была выбрана статическая нейронная сеть, так как применение динамической нейронной сети потребовало бы обучение на основе датасета, непрерывно составляющегося в процессе работы модели на основе получающихся данных. Кроме этого, модель должна обучаться на «правильных», «истинных» значениях и под этим подразумевается, что у модели нейросетевого контроллера был бы некоторый динамический эталон, который выводил бы данные, пригодные для дообучения модели нейросетевого контроллера. Все это кардинально усложнило бы разработку динамической модели контроллера вместо статической модели.

Ниже представлен фрагмент подготовки данных для обучения модели машинного обучения, а именно предварительное преобразование обучающих данных в единый эквивалентный диапазон:



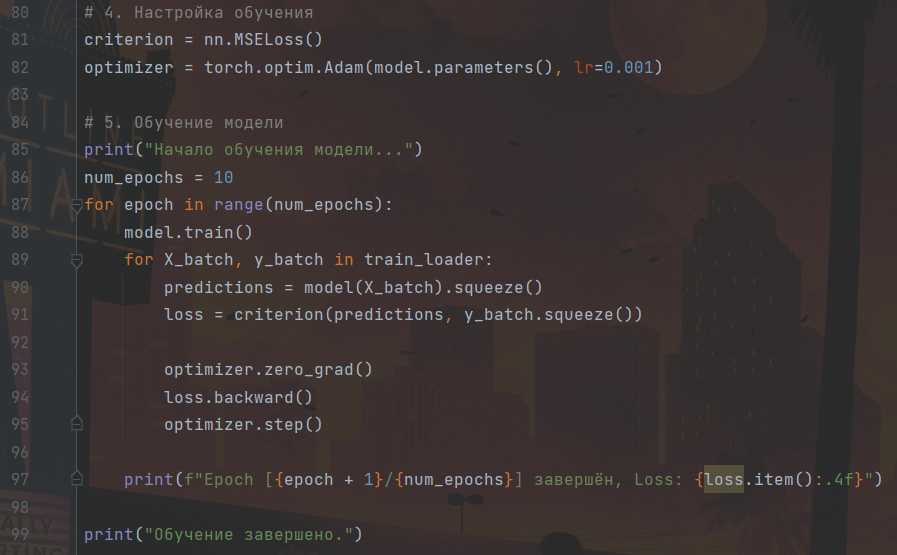
**рис. 40**

Далее приведен фрагмент инициализации модели машинного обучения.

****

**рис. 40**

И фрагмент обучения инициализированной модели нейронной сети на подготовленных для этого данных

****

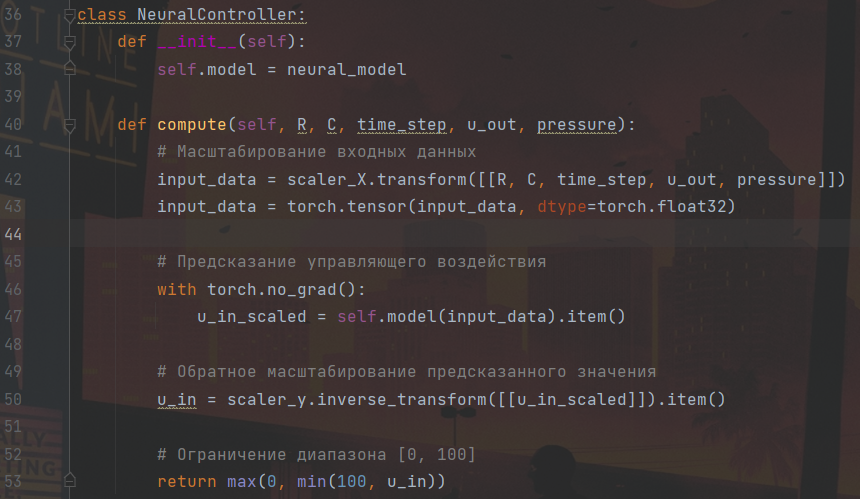
**рис. 41**

Обученная нейронная сеть была сохранена и будет использоваться в структуре нейросетевого контроллера.

## Интеграция нейросети в модель контроллера

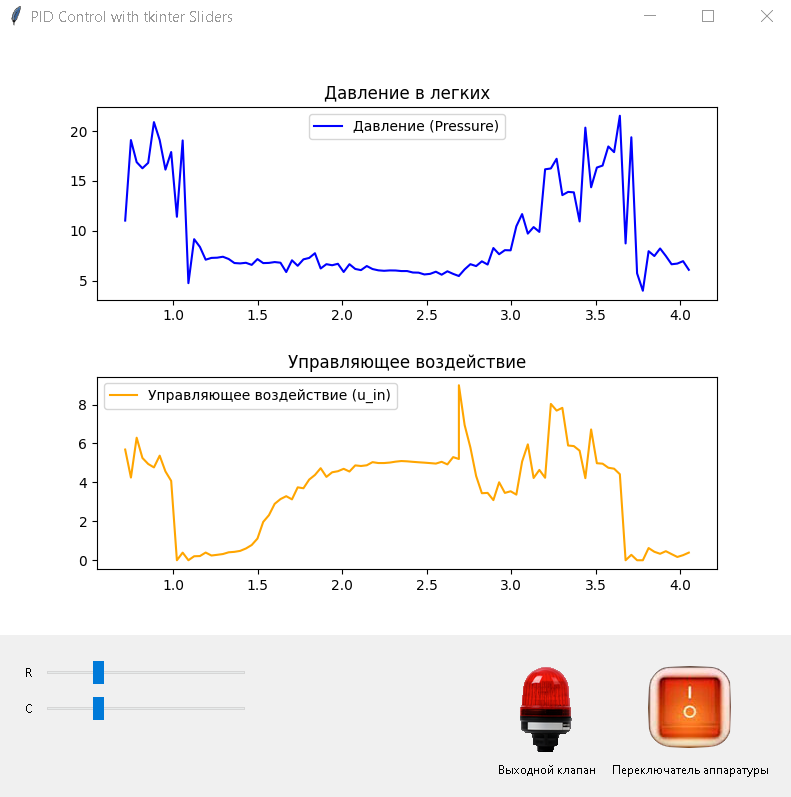
Завершающим шагом необходимо заменить ПИД-контроллер на нейросетевой контроллер и сравнить их работу.

Ниже представлен класс нейросетевого контроллера, который получает значение давления в легких и регулирует уровень входного воздействия:

****

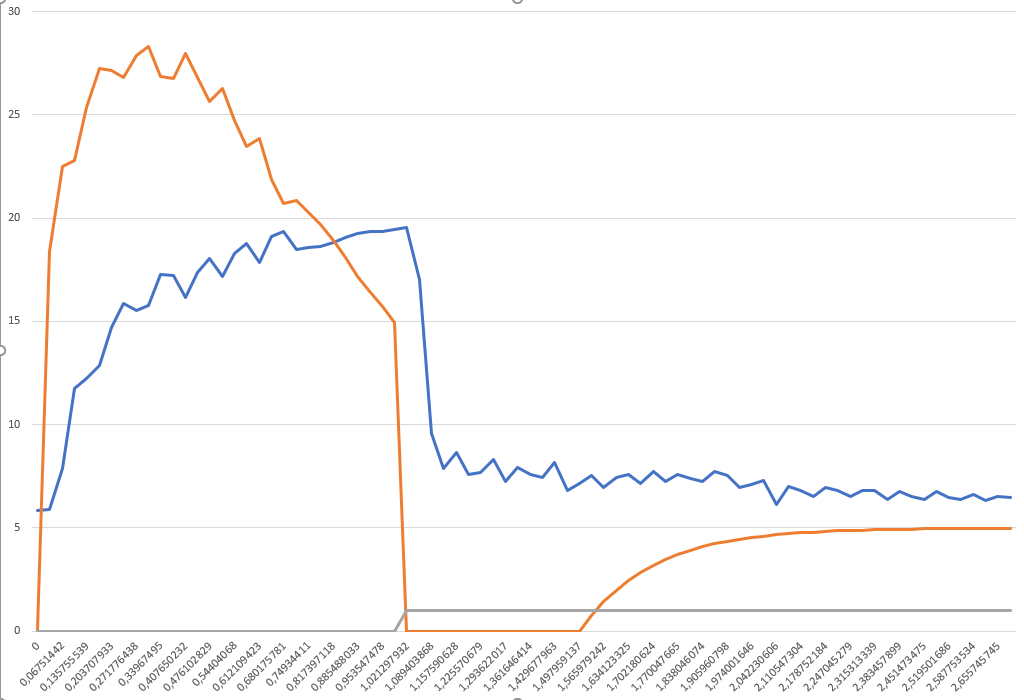
**рис. 42**

Далее демонстрируется работа нейросетевого контроллера с моделью легких:



**рис. 43**

Как видно из графиков давления и управляющего воздействия, использование нейросетевого контроллера позволило стабилизировать изменение уровня давления в легких, снизив количество «зубьев». Кроме этого, удалось снизить запрашиваемую мощность управляемого прибора более чем в 10 раз. В дополнение к этому, удалось реализовать механизм «вентиляции» - продувки легких, которая происходит в момент выдоха, что совпадает с эталонной итерацией вдоха-выдоха на следующем рисунке (оранжевый график)



**рис. 44**

# Вывод.

В ходе исследования изучены особенности работы различных типов регуляторов, таких как ПИД-регуляторы, их интеграция с нейронными сетями и влияние на управление динамическими системами.

* Нейросетевые регуляторы обладают преимуществами в адаптации и гибкости, что делает их эффективными для сложных и нелинейных систем управления.
* Статические нейронные сети предоставляют высокую стабильность в обработке заранее известных данных, в то время как динамические сети позволяют моделировать процессы, зависящие от времени и последовательности событий.
* Экспериментальная часть исследования включала моделирование аппарата искусственной вентиляции легких. Созданные модели подтвердили эффективность интеграции нейронных сетей в регуляторы, хотя точность модели нуждается в улучшении для сложных задач.

Источники.

1. Нейросетевые регуляторы в системах управления с прогнозирующей моделью: магистерская диссертация / Мария Евгеньевна Павловец; БГУ, Факультет прикладной математики и информатики, Кафедра дискретной математики и алгоритмики; науч. рук. Дмитрук Н. М.:  
   сайт: некоммерч, интернет-версия. — URL: [https://elib.bsu.by/handle/123456789/221836](https://elib.bsu.by/handle/123456789/221836%20)  
   (дата обращения: 20.09.2024).
2. Применение нейросетевых регуляторов в системах управления электроприводами. Текст научной статьи по специальности «Электротехника, электронная техника, информационные технологии» / Цвенгер И.Г., Низамов И.Р.:  
   сайт: некоммерч, интернет-версия. — URL: <https://cyberleninka.ru/article/n/primenenie-neyrosetevyh-regulyatorov-v-sistemah-upravleniya-elektroprivodami/viewer>  
   (дата обращения: 20.09.2024).
3. Теория автоматического управления для «чайников». Учебное пособие / Поляков К. Ю.:  
   сайт: некоммерч, интернет-версия. — URL: <https://kpolyakov.spb.ru/uni/teapot.htm>  
   (дата обращения: 24.10.2024).
4. Интеллектуальные информационные системы в экономике. Учебное пособие / Романов В.П.:  
   сайт: некоммерч, интернет-версия. — URL: [https://studfile.net/preview/986741](https://studfile.net/preview/986741%20)  
   (дата обращения: 10.10.2024).
5. Нейронные сети для обработки информации. Учебное пособие / Станислав Осовский  
   сайт: некоммерч, интернет-версия. — URL: <https://vk.com/doc232492587_453443734?hash=416GjQDe07OPaLsdsmLw49kixUGZCJPCBw3S3CQDFb0&dl=9ZMSztxaQd4UAoHFuwnbu7qmlmlmN92XS2KqJvSfmgH>  
   (дата обращения: 26.09.2024).
6. Постройка ПИД-регулятора с использованием нейронных сетей. / Р. А. Жилов   
   <https://cyberleninka.ru/article/n/postroyka-pid-regulyatora-s-ispolzovaniem-neyronnyh-setey>  
   (дата обращения: 03.10.2024).
7. Нейронные сети обработки информации / C.В Макартичян, Н.С. Кузнецова, С.С. Дементьев  
   <https://swsu.ru/sveden/files/MU_LZ_No_2_Mashinnoe_obuchenie_i_neyrosetevye_modeli.pdf>  
   (дата обращения: 06.10.2024).
8. Искусственные нейронные сети. Учебник. / В.С. Ростовцев <https://iweb.vyatsu.ru/document/material/41/_Учебник%20ИНС_2014_Э4743.pdf>  
   (дата обращения: 06.10.2024).
9. Нейронные сети. Полный курс / Саймон Хайкин  
   <https://vk.com/doc10903696_282205957?hash=qb4YIOWwover1wtXpnhQXWj8rOG4ishosg48PzOJEbH&dl=NvObVSzh3yGfhfOWXPvvLpd7r4Jj0BgdYIIQdKBvtrs>  
   (дата обращения: 06.10.2024).
10. Рекуррентные нейронные сети: моделирование последовательностей  
    <https://cchgeu.ru/upload/iblock/e53/3t2mob927sowc9woron5wvtp8t8rblxg/MU_GOsR_Prakt_MAG_TII_2022_1.pdf>

(дата обращения: 06.10.2024).

1. Математическое моделирование управления динамической нейронной сетью с запаздыванием / Е.А. Андреева, В.М. Цирулева.  
   <https://moitvivt.ru/ru/journal/pdf?id=429>

(дата обращения: 13.11.2024).

1. Применение сети Хопфилда в качестве инструмента распознавания образов. / Б.А. Анатольевна, П.А. Илларионовна. Новосибирский государственный университет экономики и управления.  
   <https://cyberleninka.ru/article/n/primenenie-seti-hopfilda-v-kachestve-instrumenta-raspoznavaniya-obrazov/viewer>

(дата обращения: 13.11.2024).

1. Применение сети Хэмминга для идентификации качества продукции. / А.Ю. Постыляков, М.А. Волков. Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, г. Екатеринбург  
   <https://elar.urfu.ru/bitstream/10995/74543/1/usmm_2018_115.pdf>

(дата обращения: 13.11.2024).

1. Google Brain - Ventilator Pressure Prediction - <https://www.kaggle.com/competitions/ventilator-pressure-prediction/data>

(дата обращения: 20.11.2024)

# Приложение 1.

Здесь представлен полный код программы, выполняющий подготовку модели поведения легкого.

import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
import joblib  
  
## Чтение данных  
print("Чтение данных...")  
file\_path = "train.csv"  
  
# Читаем только первые 100000 строк  
target\_rows = 100000  
data = pd.read\_csv(file\_path, nrows=target\_rows)  
print(f"Выбрано строк: {len(data)}")  
  
# Вывод минимальных, максимальных и средних значений для каждой колонки  
print("\nСтатистика по колонкам:")  
for column in data.columns:  
 min\_val = data[column].min()  
 max\_val = data[column].max()  
 mean\_val = data[column].mean()  
 print(f"{column}:")  
 print(f" Минимальное значение: {min\_val}")  
 print(f" Максимальное значение: {max\_val}")  
 print(f" Среднее значение: {mean\_val:.2f}")  
  
# Подготовка данных  
print("\nПодготовка данных для обучения...")  
X = data[['time\_step', 'u\_in', 'u\_out', 'R', 'C']]  
y = data['pressure']  
  
# Разделение на обучающую и тестовую выборки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Обучение модели  
print("\nОбучение модели...")  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# Оценка модели  
print("\nОценка модели...")  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Среднеквадратичная ошибка: {mse}")  
  
# Сохранение модели  
print("\nСохранение модели...")  
joblib.dump(model, "lung\_model.pkl")  
print("Модель сохранена в файл lung\_model.pkl")

# Приложение 2.

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
from PIL import Image, ImageTk  
import joblib  
  
# PID-регулятор  
class PIDController:  
 def \_\_init\_\_(self, kp, ki, kd, setpoint=35, integral\_limit=50):  
 self.kp = kp  
 self.ki = ki  
 self.kd = kd  
 self.setpoint = setpoint  
 self.integral = 0  
 self.prev\_error = 0  
 self.integral\_limit = integral\_limit  
  
 def compute(self, current\_value, dt, enabled=True):  
 if not enabled: # Если система отключена, управление равно нулю  
 return 0  
 error = self.setpoint - current\_value  
 self.integral += error \* dt  
 self.integral = max(-self.integral\_limit, min(self.integral, self.integral\_limit))  
 derivative = (error - self.prev\_error) / dt  
 self.prev\_error = error  
 return self.kp \* error + self.ki \* self.integral + self.kd \* derivative  
  
# Загрузка данных  
data = pd.read\_csv("train.csv")  
data = data.drop(columns=["pressure"])  
data\_index = 0  
  
# Загрузка модели  
model = joblib.load("lung\_model1.pkl")  
  
# Инициализация параметров  
time\_step = 0.0  
pressure = 0  
u\_in = 0  
pid = PIDController(kp=3.0, ki=0.2, kd=1.0, setpoint=35)  
system\_enabled = True # Переключатель состояния системы  
  
# Подготовка данных для графиков  
global\_time = 0  
previous\_time\_step = 0  
time\_data = []  
pressure\_data = []  
control\_data = []  
setpoint\_data = []  
  
# Создание окна tkinter  
root = tk.Tk()  
root.title("PID Control with tkinter Sliders")  
  
# Создание графиков  
fig, ax = plt.subplots(2, 1, figsize=(8, 6))  
plt.subplots\_adjust(hspace=0.4)  
  
canvas = FigureCanvasTkAgg(fig, master=root)  
canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)  
  
# Загрузка изображений переключателя и лампочки  
turn\_on\_image = ImageTk.PhotoImage(Image.open("turn\_on.png"))  
turn\_off\_image = ImageTk.PhotoImage(Image.open("turn\_off.png"))  
light\_image = ImageTk.PhotoImage(Image.open("light.png"))  
dark\_image = ImageTk.PhotoImage(Image.open("dark.png"))  
  
# Добавление нижней рамки для элементов управления  
bottom\_frame = tk.Frame(root)  
bottom\_frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=10)  
  
# Рамка для ползунков (левая часть)  
slider\_frame = tk.Frame(bottom\_frame)  
slider\_frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)  
  
# Рамка для лампочки и переключателя (правая часть)  
control\_frame = tk.Frame(bottom\_frame)  
control\_frame.pack(side=tk.RIGHT, fill=tk.Y, padx=10, pady=10)  
  
# Лампочка  
lamp\_frame = tk.Frame(control\_frame)  
lamp\_frame.pack(side=tk.LEFT, padx=5)  
lamp\_label = tk.Label(lamp\_frame, image=light\_image)  
lamp\_label.pack()  
tk.Label(lamp\_frame, text="Выходной клапан").pack()  
  
# Переключатель  
switch\_frame = tk.Frame(control\_frame)  
switch\_frame.pack(side=tk.LEFT, padx=5)  
switch\_label = tk.Label(switch\_frame, image=turn\_on\_image)  
switch\_label.pack()  
tk.Label(switch\_frame, text="Переключатель аппаратуры").pack()  
  
# Функция переключения состояния  
def toggle\_system():  
 global system\_enabled  
 system\_enabled = not system\_enabled  
 if system\_enabled:  
 switch\_label.config(image=turn\_on\_image)  
 else:  
 switch\_label.config(image=turn\_off\_image)  
  
# Обработчик клика по переключателю  
switch\_label.bind("<Button-1>", lambda event: toggle\_system())  
  
# Функция обновления графиков  
def update\_plot():  
 global time\_step, u\_in, pressure, data\_index, global\_time, previous\_time\_step  
  
 dt = 0.1 # Интервал времени  
  
 # Чтение параметров текущей записи  
 current\_data = data.iloc[data\_index]  
 time\_step = current\_data["time\_step"]  
 u\_out = current\_data["u\_out"]  
  
 # Обновление состояния лампочки  
 if u\_out == 1:  
 lamp\_label.config(image=light\_image)  
 else:  
 lamp\_label.config(image=dark\_image)  
  
 # Получение значений R и C из ползунков  
 R = slider\_R.get()  
 C = slider\_C.get()  
  
 # Обновление глобального времени  
 if time\_step < previous\_time\_step: # Новый цикл вдоха  
 global\_time += time\_step  
 else: # Текущий цикл  
 global\_time += time\_step - previous\_time\_step  
 previous\_time\_step = time\_step  
  
 # Цикличность данных  
 data\_index = (data\_index + 1) % len(data)  
  
 # Обновление PID  
 pid.kp = slider\_kp.get()  
 pid.ki = slider\_ki.get()  
 pid.kd = slider\_kd.get()  
  
 # Управляющее воздействие через PID-регулятор  
 if u\_out == 1: # Если выходной клапан открыт  
 u\_in = 0  
 else:  
 u\_in = pid.compute(pressure, dt, enabled=system\_enabled)  
 u\_in = max(0, min(100, u\_in)) if system\_enabled else 0 # Ограничение по диапазону  
  
 # Прогноз давления через модель  
 input\_data = pd.DataFrame([[time\_step, u\_in, u\_out, R, C]], columns=['time\_step', 'u\_in', 'u\_out', 'R', 'C'])  
 pressure = model.predict(input\_data)[0]  
  
 # Обновление данных для графиков  
 time\_data.append(global\_time)  
 pressure\_data.append(pressure)  
 control\_data.append(u\_in)  
 setpoint\_data.append(pid.setpoint)  
  
 # Ограничение числа точек на графике  
 max\_points = 100  
 if len(time\_data) > max\_points:  
 time\_data.pop(0)  
 pressure\_data.pop(0)  
 control\_data.pop(0)  
 setpoint\_data.pop(0)  
  
 # Обновление графиков  
 ax[0].clear()  
 ax[1].clear()  
 ax[0].plot(time\_data, pressure\_data, label="Давление (Pressure)", color="blue")  
 ax[0].plot(time\_data, setpoint\_data, label="Целевое давление (Setpoint)", linestyle="--", color="red")  
 ax[1].plot(time\_data, control\_data, label="Управляющее воздействие (u\_in)", color="orange")  
 ax[0].set\_title("Давление в легких")  
 ax[1].set\_title("Управляющее воздействие")  
 ax[0].legend()  
 ax[1].legend()  
  
 canvas.draw()  
 root.after(100, update\_plot)  
  
# Ползунки  
slider\_kp = ttk.Scale(slider\_frame, from\_=0.1, to=10.0, value=pid.kp, orient=tk.HORIZONTAL, length=200)  
slider\_ki = ttk.Scale(slider\_frame, from\_=0.0, to=2.0, value=pid.ki, orient=tk.HORIZONTAL, length=200)  
slider\_kd = ttk.Scale(slider\_frame, from\_=0.0, to=5.0, value=pid.kd, orient=tk.HORIZONTAL, length=200)  
slider\_R = ttk.Scale(slider\_frame, from\_=10, to=50, value=20, orient=tk.HORIZONTAL, length=200)  
slider\_C = ttk.Scale(slider\_frame, from\_=10, to=50, value=20, orient=tk.HORIZONTAL, length=200)  
  
# Метки для ползунков  
tk.Label(slider\_frame, text="Kp").grid(row=0, column=0, padx=5, pady=5)  
slider\_kp.grid(row=0, column=1, padx=5, pady=5)  
  
tk.Label(slider\_frame, text="Ki").grid(row=1, column=0, padx=5, pady=5)  
slider\_ki.grid(row=1, column=1, padx=5, pady=5)  
  
tk.Label(slider\_frame, text="Kd").grid(row=2, column=0, padx=5, pady=5)  
slider\_kd.grid(row=2, column=1, padx=5, pady=5)  
  
tk.Label(slider\_frame, text="R").grid(row=3, column=0, padx=5, pady=5)  
slider\_R.grid(row=3, column=1, padx=5, pady=5)  
  
tk.Label(slider\_frame, text="C").grid(row=4, column=0, padx=5, pady=5)  
slider\_C.grid(row=4, column=1, padx=5, pady=5)  
  
# Запуск обновления графика  
root.after(100, update\_plot)  
root.mainloop()