Министерство образования и науки РФ

ФГБОУ ВО «Тверской Государственный Университет»

Математический факультет

Специальность «Компьютерная Безопасность»

Специализация «Математические методы защиты информации»

ДИПЛОМНАЯ РАБОТА

Использование искусственных нейронных сетей в робототехнике

Автор:

Беляев Александр Константинович

Научный руководитель:

Зав. кафедрой КБиММУ,

д. ф. –м. н., профессор

Андреева Е.А.

Допущен к защите:

Руководитель ООП:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Заведующий кафедрой Компьютерной безопасности и

математических методов управления (КБиММУ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Тверь, 2017 г.**

**Содержание**

[Введение 4](#_Toc499944272)

[Глава 1. Искусственные нейронные сети 6](#_Toc499944273)

[§1. Математическая модель искусственной нейронной сети. 6](#_Toc499944274)

[§2. Применение ИНС. 11](#_Toc499944275)

[Глава 2. Задачи оптимального управления. 17](#_Toc499944276)

[§1. Формулировка задачи оптимального управления. 17](#_Toc499944277)

[§2. Задачи оптимального управления с запаздыванием. 22](#_Toc499944278)

[§3. Линейно-квадратичная задача оптимального управления с запаздыванием. 27](#_Toc499944279)

[§4. Задача оптимального управления системой с переменным запаздыванием 30](#_Toc499944280)

[§5. Дискретная аппроксимация для ЗОУ с запаздыванием. 31](#_Toc499944281)

[§6. Дискретная аппроксимация для ЛКЗОУ с запаздыванием 32](#_Toc499944282)

[§7. ЗОУ с запаздыванием с нефиксированным временем процесса. 34](#_Toc499944283)

[§8. Дискретная аппроксимация для ЗОУ с запаздыванием с нефиксированным временем процесса. 36](#_Toc499944284)

[§9. Метод градиентного спуска 38](#_Toc499944285)

[Глава 3. Метод динамического программирования 42](#_Toc499944286)

[§1. Принцип оптимальности и уравнение Беллмана 42](#_Toc499944287)

[§2. Уравнение Беллмана для линейно-квадратичной задачи оптимального управления. 44](#_Toc499944288)

[§3. Использование линейно-квадратичного регулятора в системах автоматического регулирования. 47](#_Toc499944289)

[Глава 4. Оптимизация ИНС 50](#_Toc499944290)

[§1. Описание модели ИНС, постановка задачи. 50](#_Toc499944291)

[§2. Дискретная аппроксимация. 54](#_Toc499944292)

[§3. Блок-схема работы алгоритма. 57](#_Toc499944293)

[Глава 5. Программная реализация алгоритма обучения ИНС с запаздыванием. 60](#_Toc499944294)

[§1. Описание разработки программы. 60](#_Toc499944295)

[§2. Описание численных экспериментов. 62](#_Toc499944296)

[Заключение 73](#_Toc499944297)

[Список литературы 75](#_Toc499944298)

[Приложение 1. Листинг программы 81](#_Toc499944299)

### Введение

На протяжении долгого времени люди сталкивались с задачами оптимизации в различных отраслях своей деятельности, начиная от производственных процессов и заканчивая системами принятия решений в сфере информационной безопасности (ИБ). В связи с этим возникло понятие управления.

Цель теории управления — изучение и совершенствование принципов, структур, методов и техники управления. Решению проблем управления сопутствуют значительные трудности, связанные с описанием и моделированием объекта управления, а также с разработкой методов оптимизации.

Теория оптимального управления — это бурно развивающаяся отрасль, значительное расширение методов которой выпало на конец XX века, когда её принципы оказались полезны при решении задач компьютерного моделирования различных систем, процессов и объектов, что позволило существенно увеличить возможности автоматизации человеческого труда.

С середины прошлого века началось бурное развитие теории искусственных нейронных сетей (ИНС) [1]. Деятельность, направленная на изучение структуры и функционирования мозга, открыла огромный простор для применения аппарата нейронных сетей. Задачи, связанные с ИНС, с каждым годом обретают все большую популярность. Нейронным сетям присущи такие свойства, как возможность обучения и обобщение информации, адаптивность к внешним изменениям, гибкость настройки с возможностью учета нелинейных зависимостей, стойкость к внутренним повреждениям благодаря изначально заложенному параллелизму и др.[2] Благодаря этому, ИНС применяют при решении обширного класса прикладных задач, некоторые из которых будут рассмотрены в данной работе.

Ввиду всего вышесказанного, на данный момент актуальна тематика применения теории оптимального управления, решение с помощью её методов прикладных задач, а также вопрос изучения нейронных сетей и их приложения к задачам оптимального управления (ЗОУ).

При рассмотрении большинства реальных управляемых объектов и процессов нельзя обойтись без учета запаздываний, возникающих при передаче сигнала. Этот факт необходимо учитывать при моделировании системы и решении сопутствующих задач. Формулировка необходимых и достаточных условий оптимальности для объектов с запаздываниями, а также вопрос синтеза оптимального управления для линейных систем с последствиями изложены в работах [3-5]

На данный момент, наиболее полно изучены ЗОУ с запаздываниями в фазовой переменной. Решение же ЗОУ с дополнительным запаздыванием еще и в компоненте управления исследованы недостаточно. Также, актуальным вопросом является развитие приближенных и численных методов для решения ЗОУ с запаздываниями.

При написании данной работы, автором преследовались следующие цели:

1. Разработка методов решения различных задач оптимального управления с запаздываниями в аргументах функций состояния и управления.
2. Сравнительный анализ моделей искусственных нейронных сетей и рассмотрение возможности их применения для решения прикладных задач робототехники и информационной безопасности.
3. Применение метода динамического программирования для решения линейно-квадратичной задачи оптимального управления.
4. Применение математической теории оптимального управления для решения задач оптимального управления с запаздыванием и оптимизация искусственной нейронной сети, динамика которой описывается СДУ с запаздываниями по фазовым координатам и по управлению.
5. Разработка программы построения численного решения задачи оптимизации ИНС.

## Глава 1. Искусственные нейронные сети

### §1. Математическая модель искусственной нейронной сети.

**Иску́сственная нейро́нная се́ть** (ИНС) — математическая модель и ее программная или аппаратная реализация, при проектировании которой учитываются особенности строения и функционирования биологических нейронных сетей. Возникновение этого понятия связано с изучением и моделированием процессов, протекающих в мозге, в сетях нервных клеток человека и животных. Первые шаги в данном направлении были сделаны У. Маккалоком и У. Питтсом, которые работали над моделированием искусственных нейронов [1, 6].

В последствии, аппарат ИНС дополнился разнообразными методами и моделями. Появились способы, показавшие потенциал применения новой теории. Важным моментом является то, что нейронные сети не программируются, как это принято делать в стандартных математических методах. К ИНС больше применим такой термин, как обучение. Технически, обучение заключается в нахождении коэффициентов, описывающих связи между отдельными объектами сети, так называемыми нейронами. Это позволяет выявлять сложные зависимости между входными и выходными данными, а также выполнять обобщение и, при необходимости, классификацию. Благодаря этому, после процедуры обучения, система способна предоставить верный результат на основании данных, которые отсутствовали при обучении, либо были утеряны или искажены в процессе передачи. Благодаря этому свойству, нейросетевые методы и алгоритмы обладают значительно большей гибкостью и способностью к адаптации к изменению свойств системы, нежели классические методы оптимизации и регулирования.

Существует несколько основных подходов к классификации искусственных нейронных сетей. Кратко сформулируем их:

**По типу входной информации, ИНС разделяют на:**

* Аналоговые (оперируют информацией в форме действительных чисел);
* Двоичные, либо бинарные (оперируют информацией, представленной в двоичном виде);
* Образные нейронные сети (оперируют информацией, представленной в виде образов: знаков, иероглифов, символов).

**По характеру процедуры обучения:**

* Обучение с учителем — выходное пространство решений нейронной сети известно;
* Обучение без учителя — нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называют самоорганизующимися;
* Обучение с подкреплением — система назначения штрафов и поощрений от среды.

**По характеру настройки синапсов**

* Сети с фиксированными связями (весовые коэффициенты нейронной сети выбираются сразу, исходя из условий задачи);
* Сети с динамическими связями (для них в процессе обучения происходит настройка синаптических связей).

**По характеру связей**

* Сети прямого распространения (Feedforward)
* Рекуррентные нейронные сети
* Радиально-базисные функции
* Самоорганизующиеся карты

Следует заметить, что представлены не все виды ИНС, подробная классификация и обзор моделей ИНС, а также способы их обучения рассмотрены в [2, 7].

**§2. Математическая модель ИНС.**

Центральным понятием во всей тематике нейронных сетей является понятие нейрона. Можно сказать, что это структурная единица всей системы.

**Иску́сственный нейро́н** — элемент ИНС, представляющий из себя условную модель естественного нейрона. С точки зрения математики, искусственный нейрон представляет собой нелинейную функцию, аргументом которой является линейная комбинация входных сигналов. Такие функции получили специальное название - функции активации или срабатывания. Результат, который получен в ходе вычислений подается на выход, который является единственным. Естественными аналогами входов и выхода являются дендриты и аксон – составные части реального нейрона.

Соединяя выходы одних нейронов со входами других, получают искусственные нейронные сети.

Принципиальная структура искусственного нейрона изображена на Рисунке 1.

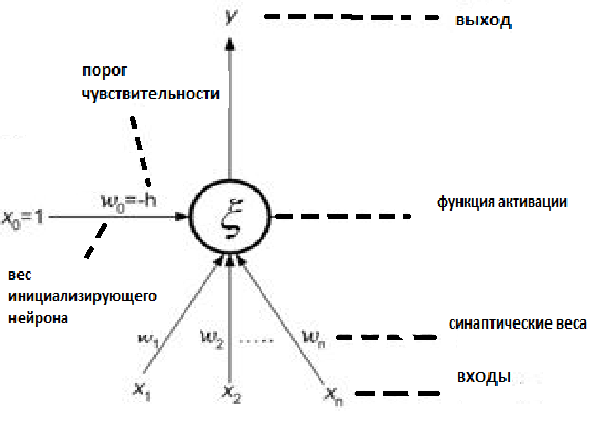


Рисунок 1

Каналы, связывающие входы и выходы разных нейронов принято называть *синапсами* в соответствие с аналогичными связями в биологической нейронной сети. Каждая связь характеризуется своим, так называемым синаптическим или весовым коэффициентом. Связи с положительным весом называются возбуждающими, а с отрицательным — угнетающими или тормозящими. Нейрон имеет один единственный выход, который принято именовать *аксоном* в соответствие с биологическим прототипом. В зависимости от топологии моделируемой нейронной сети, выходной сигнал с одного нейрона может поступать на различное число входов других узлов сети.

Математически нейрон представляет собой взвешенный сумматор, единственный выход которого определяется через его входы и матрицу весов следующим образом:

|  |
| --- |
| , где |

Здесь соответственно сигналы на входах нейрона и веса входов, функция называется индуцированным локальным полем, а – передаточной функцией. Возможные значения сигналов на входах нейрона считают заданными в интервале . Они могут быть либо дискретными (0 или 1), либо аналоговыми (непрерывными). Дополнительный вход и соответствующий ему вес используются для инициализации нейрона. Под инициализацией подразумевается смещение активационной функции нейрона по горизонтальной оси, то есть формирование порога чувствительности нейрона. Кроме того, иногда к выходу нейрона специально добавляют некую случайную величину, называемую сдвигом. Сдвиг можно рассматривать как сигнал на дополнительном, всегда нагруженном, синапсе.

ИНСсостоит из формальных нейронов, которые соединены таким образом, чтобы каждый выходной нейрон далее служил в качестве входного сигнала другого нейрона. Число нейронов и способы, которыми они соединены между собой определяют архитектуру (топологию) нейронной сети.

В основном, нейроны классифицируют на основе их положения в топологии сети. Разделяют:

* Входные нейроны — служат для получения информации, закодированной во входном сигнале. Чаще всего, данные узлы сети не предназначены для выполнения вычислительных операций. В их функции входит передача входного сигнала далее в систему с возможной его модификацией в виде усиления, либо же ослабления;
* Выходные нейроны — служат для предоставления выходной информации. В выходных нейронах иногда производятся дополнительные операции, связанные с преобразованием информации для формирования выходного сигнала;
* Промежуточные нейроны — служат для выполнения основных вычислительных операций.

В зависимости от того, с каким сигналами работает искусственная нейронная сеть различают системы непрерывного времени и системы с дискретным временем.

**Непрерывные по времени** системы работают с сигналами, которые являются функциями непрерывной переменной, которая является непрерывной функцией времени . Такие системы описываются с помощью дифференциальных уравнений. Чтобы моделировать подобный тип систем требуется операция интегрирования.

Динамическая система с непрерывным временем показана на Рисунке 2. Заметим, что данным системам присущ контур обратной связи.

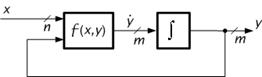


Рисунок 2

Уравнение, соответствующее данным системам, имеет вид:

Однако, в большинстве случаев рассматривается не непрерывные системы, а дискретные. Это проще при обработке и моделировании задачи на компьютере, где изначально все процессы задумывались, как дискретные. К тому же, от любой непрерывной задачи достаточно просто перейти к рассмотрению дискретной. Достаточно осуществить разбиение отрезка времени на конечные промежутки и выбрать отсчетные значения.

**Дискретные по времени** системы работают с сигналами, которые являются функциями дискретной переменной. Подразумевается, что в начале сеть оказывается в момент времени 0 и ее состояние обновляется только в момент времени 1, 2, 3, …

Уравнение, соответствующее данным системам, имеет вид:

Динамическая система с дискретным временем показана на Рисунке 3.

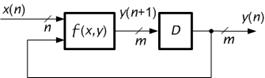


Рисунок 3

Подробно, модели ИНС и системы с наличием обратной связи рассмотрены в [8, 9].

### §2. Применение ИНС.

Как было упомянуто ранее, искусственные нейронные сети используются при решении обширного класса прикладных задач. Рассмотрим некоторые из них.

Из-за свойств нейронных сетей, сближающих их с человеческим интеллектом, известно много примеров их использования при создании искусственного интеллекта. В последнее время наблюдается существенный прогресс в сфере развития нейросетевых технологий. Это связано с несколькими факторами:

1. Развитие вычислительной техники. В частности, появление и распространение параллельных вычислений на графических процессорах, что позволяет достичь большой производительности при относительно небольших затратах.
2. Рост объема данных для обучения. В настоящее время, технологии сбора и хранения данных позволяют без особого труда подобрать необходимый материал для обучения нейронной сети самой разнообразной направленности. К примеру, при создании системы распознавания можно воспользоваться базой изображений ImageNet [10], которая уже на момент написания данной работы содержит более 10 млн. файлов, разбитых на категории.
3. Появление новых эффективных методов обучения, к примеру, методы глубокого обучения [11] сетей с большим количеством скрытых слоев, чье активное развитие началось с началом двадцать первого века.

Примером разработок в области искусственного интеллекта, с которыми можно столкнуться в повседневной жизни, служат программные ассистенты, такие как Алиса от компании Яндекс, Siri от Apple, Assistant от Google, Alexa от Amazon, M от Facebook, Cortana от Microsoft. Алгоритм работы таких систем можно назвать комбинированным, из-за того, что нейронные сети применяются для распознавания речи и графической информации, а для общей работы используются вручную запрограммированные правила.

В области робототехники также можно найти применение искусственным нейронным сетям. К примеру, рассмотрим системы автопилотирования автомобилем, которым в последнее время уделено большое внимание.

Роботизированной системе, которая осуществляет управление автомобилем необходимо наличие специальных датчиков и сенсоров для получения информации об окружающей среде. Такими входными устройствами являются камеры, инерционные датчики движения, датчики оценки положения и т.д. Благодаря им система управления сможет получать информацию о динамических характеристиках объекта, таких как скорость и ускорение, о положении автомобиля на дорожном полотне, кренах кузова и т.д. Помимо этого, важно отслеживать и состояние компонентов самого автомобиля, чтобы исключить возможность движения в случае повреждения важных систем.

Компоненты системы автопилотирования можно условно поделить на несколько функциональных групп:

1. Система наблюдения, отвечающая за собственно зрение машины, определение объектов и позиционирование транспорта.
2. Система, в функции которой входит построение дорожной карты и размещение на ней автомобиля. Здесь подразумевается, что робот самостоятельно строит карту местности, определяя уклон, покрытие, число полос, дорожную разметку и знаки, а также производит анализ погодных условий.
3. Система навигации, отвечающая за построение траектории движения в условиях, определенных модулем, отвечающем за составление карты местности. Сюда же можно отнести и прокладку маршрута, учет расположения объектов рядом с автомобилем, расчет опасных сближений и т.д.
4. Система управления, служащая для манипулирования исполнительными механизмами. В задачи данной системы входит регулировка скорости, контроль торможения и определение оптимального угла поворота колес для задания нужной траектории движения.

Аппарат искусственных нейронных сетей можно применить в каждом из этих компонентов. К примеру, в системе наблюдения не обойтись без точного распознавания образов, которыми могут быть дорожные знаки, разметка и другие участники движения. Учитывая тот факт, что зачастую, движение автомобиля осуществляется в сложных погодных условиях, без гибкости нейросетевых алгоритмов не обойтись.

Примерами передовых разработок в области систем автопилотирования автомобиля можно назвать проект C-Pilot от российской компании Cognitive Technologies, в котором используются нейронные сети глубокого обучения. Сюда же можно отнести и разработки автоконцерна Tesla Motors, в которых важную роль играет нейронная Tesla Neural Net и проект WayMo от Google [12]. В работе [13] рассмотрены вопросы оценки качества обучения «глубоких» нейронных сетей, применяемых в вышеуказанных системах автопилотирования.

Если же решать задачу более узконаправленную, к примеру, движение робота по оптимальной траектории в обход препятствий, то можно применять хорошо известный метод обратного распространения ошибки. В проекте [14] рассмотрен вопрос создания подобного робота, управляемого нейронной сетью.

Нейроуправление можно успешно использовать и при создании роботов-манипуляторов, нашедших применение в сборочных цехах на заводских комплексах, что показано в [15].

Другие случаи применения аппарата ИНС при создании роботов самой различной направленности, к примеру, таких как робот-экскурсовод, а также общие принципы конструирования и программирования мобильных роботов изложены в работах [16-17].

Другим примером использования нейроуправления является система управления гибридным двигателем автомобиля Toyota Prius. В качестве силовой установки данного транспортного средства выступает пара: ДВС и электромотор. Задача состоит в том, чтобы заставить работать эти агрегаты вместе с максимальной эффективностью. За этим кроется то, что автомобиль должен не уступать своим конкурентам в динамике, имея изначально слабый мотор маленького объема, а также обеспечивать приемлемый запас хода. Помимо этого, основной проблемой остается снижение расхода топлива и, как следствие, минимизация вредных выбросов в атмосферу.

Задача усложняется еще и тем, что, имея в распоряжении целых два силовых агрегата (один питается за счет сгорания топливно-воздушной смеси, а другой за счет электроэнергии), а значит существует несколько возможных видов их работы. Движение может осуществляться за счет тяги от ДВС, либо от электромотора, либо от параллельной работы агрегатов. Помимо этого, учитывая особенность конструкции привода автомобиля, при торможении можно восполнять часть электроэнергии, накапливая её в аккумуляторах.

Для того, чтобы качественно управлять всеми протекающими при этом процессами и необходимы методы нейроуправления. В работе [18] рассказывается о применении рекуррентных нейронных сетей и специальных алгоритмах обучения контроллера, которые обеспечивают слаженную работу силовой установки описанного выше автомобиля.

Кроме систем управления наземными мобильными объектами, известны случаи применения ИНС для создания систем управления летательными аппаратами. В работе [19] рассмотрена модель беспилотного летательного аппарата (БПЛА) и предложен способ его стабилизации с учетом внешних воздействий, для чего применяется регулятор на основе нейронных сетей.

Сфера задач, решаемых с использованием робототехники, не ограничивается упомянутыми выше примерами. Практически в любой сфере деятельности, можно найти применение такого подхода. К примеру, в работе [21] уделено внимание разработке специального роботизированного ремонтного комплекса, призванного работать в условиях атомной электростанции (АЭС). При проектировании архитектуры управления системой были использованы методы параллельного гибридного нейроуправления.

Отдельно стоит отметить использование нейросетевых технологий в сфере информационной безопасности.

Большую роль в вопросах защиты информации играет безопасность каналов передачи данных. Как правило, это подразумевает отслеживание состояния сети, которой соединены устройства, обрабатывающие информацию.

В зависимости от физического принципа, на котором базируются сети передачи, различают проводные и беспроводные каналы. Для обеспечения функционирования сети существуют разнообразные сетевые протоколы, призванные обеспечивать надежность, высокую скорость и безопасность. Большую роль при решении вопросов информационной безопасности любого предприятия играет анализ трафика, циркулирующего в информационных каналах.

В работе [22] показана возможность применения ИНС при создании, т. н. средств обнаружения вторжений (СОВ). Данный комплекс направлен на анализ и обнаружение аномальной активности в участках компьютерной сети. Учитывая, что средство должно выделить некоторый шаблон нежелательного поведения, здесь, без сомнения, пригодятся свойства нейронных сетей, связанные с распознаванием.

В работе [23] проводится исследование алгоритмов обучения ИНС в применении к проблеме выявления сетевых атак на компьютерную сеть.

Тематике анализа случаев аномальной активности в каналах передачи данных уделено внимание в статьях [24-25].

Помимо учета рисков, связанных с возможными атаками на компьютерные сети, не стоит исключать из рассмотрения и возможные внутренние нарушения, которые могут быть связаны, в первую очередь с уязвимостями ПО. Так называемые, программные закладки, либо просто недочеты разработчиков могут привести к серьезным последствиям, если нарушителю будут известны подобные слабые места. В связи с этим, на этапе проектирования, разработки и последующего внедрения ПО, следует проводить анализ исходного кода на наличие уязвимостей. Исследования по данной теме проводились в работе [26].

Вышеописанное касается отдельных компонентов, из которых складывается информационная безопасность отдельно взятого предприятия. Зачастую, при проектировании, либо плановом анализе состояния защиты, необходимо производить комплексную оценку. Способы решения данной проблемы предложены в [27].

## Глава 2. Задачи оптимального управления.

### §1. Формулировка задачи оптимального управления.

**Под задачей оптимального управления (ЗОУ)** принято понимать задачу по проектированию системы, которая обеспечивала бы для рассматриваемого объекта, либо процесса управления закон управления, либо последовательность управляющий воздействий, с помощью которых может быть достигнут максимум или минимум введенной совокупности критериев качества.

При решении ЗОУ преследуются цели по расчету, так называемого программного ОУ, либо по синтезу ОУ. При нахождении программы управления используются численные методы нахождения экстремума целевого функционала или способы решения краевой задачи принципа максимума. При синтезе ОУ необходимо решать задачу нелинейного программирования. В этом случае, управление строится поэтапно, при помощи итерационного процесса. Подробнее о синтезе управления будет рассказано далее.

При решении ЗОУ разрабатывается математическая модель управляемой системы, которая будет описывать её поведение с течением времени с учетом влияния внешних, а при наличии и внутренних, управляющих воздействий, а также на основании текущего состояния. Формирование математической модели подразумевает: формулировку цели управления, выраженную через целевой функционал качества управления; определение соотношений, описывающих возможные состояния объекта управления; учет возможных ограничений на компоненты системы, задаваемых в виде равенств или неравенств. Подробнее об этом рассказано в [28].

**Постановка задачи оптимального управления.**

При формулировке ЗОУ необходимо учитывать следующие факторы: математическую модель управляемого объекта, цель управления (именуемую критерием или функционалом качества), различные ограничения на траекторию системы, на множество допустимых управляющих воздействий, на продолжительность во времени процесса управления, и т.д. Чаще всего рассматриваются ограничения, задаваемые в виде равенств или неравенств, а также ограничения смешанного типа. Каждый из этих видов накладывает ограничение на выбор метода решения ЗОУ.

**Математическая модель**

В зависимости от моделируемого объекта или процесса, а также от необходимой степени детализации его изучения, при описании применяются разные типы уравнений: обыкновенные дифференциальные уравнения (ОДУ), уравнения с запаздываниями, стохастические или вероятностные уравнения, интегро-дифференциальные уравнения и др. [29]

Все задачи оптимального управления можно рассматривать как задачи математического программирования и в таком виде решать их численными методами.

В случае, когда объект управления является детерминированным, для его описания используются дифференциальные уравнения. В большинстве случаев применяют ОДУ вида ./. Для моделирования сложных систем, к примеру, с большим числом распределённых параметров для описания используются ДУ в частных производных. В случаях, когда рассматривается стохастический управляемый объект или процесс, то есть состояния системы подчиняются некоторому вероятностному закону, для его описания используются стохастические ДУ.

**Критерий качества (минимизируемый функционал).**

Управление системой необходимо для достижения целей, которые могут формулироваться, к примеру, в виде минимизации или максимизации по управлению некоторого функционала , определяемого управлением и траекторией *.*

Здесь – заданные скалярные функции.

**Ограничения на траекторию, фазовые ограничения**

Чаще всего, на состояния системы накладываются ограничения вида , где – заданная область в . В зависимости от конкретного типа этих ограничений выделяют различные классы задач управления. В задачах с фиксированными концами начальное и конечное состояния системы и заданы вместе с остальными параметрами. Если же хотя бы одно из этих значений не известно заранее, то формулируется ЗОУ со свободным концом. Задача с подвижными концами – это задача, в которой моменты и фиксированы, а векторы и принадлежат соответственно областям и . В ряде случаев ограничения носят интегральный характер и имеют вид . Если в задаче начальное положение и конечное заданы, моменты начала движения и окончания свободны, функция , то ставится задачу о переводе системы из положения в положение за минимально возможное время. Подобного рода задачи именуются задачами быстродействия.

**Ограничения на управляющие воздействия**

Ограничения на управление зависят от того, какая именно информация о системе доступна при выработке управляющего воздействия. Если вектор недоступен измерению, то оптимальное управление ищется в классе функций , зависящих только от. В этом случае оптимальное управление именуется программным. Если же вектор известен точно при, то оптимальное управление ищется в классе функционалов и называется синтезом оптимального управления (или управлением по принципу обратной связи). Здесь означает всю траекторию движения на отрезке . Отметим, что принцип обратной связи является одним из центральных принципов кибернетики.

Все задачи оптимального управления можно разделить на несколько видов в зависимости от того, на каком временном промежутке рассматривается описываемая система. При описании различных классов систем будет использована линейно-квадратичная задача оптимального управления, в которой исследуемая динамическая система описывается линейными дифференциальными уравнениями, а показатель качества представляет собой квадратичный функционал.

**Системы с непрерывным временем, рассматриваемые на фиксированном временном промежутке.**

Состояние системы описывается ОДУ вида:

Временной отрезок фиксирован: .

Квадратичный функционал качества имеет вид:

где называют терминальным слагаемым.

Данное слагаемое отвечает за конечное состояние управляемой системы.

**Системы с непрерывным временем, рассматриваемые на неограниченном временном промежутке**.

Система описывается ОДУ вида:

Временной отрезок в данном случае не ограничивается какими-либо значениями. Квадратичный функционал имеет вид:

**Системы с дискретным временем, рассматриваемые на фиксированном временном промежутке.**

В данном случае система будет иметь иной вид. Задача из непрерывной переходит в дискретную, что влечет за собой применение иной группы методов для её решения. Рассматриваются различных значений времени, в которых может оказаться система. Траектория задается следующим образом:

Функционал примет следующий вид:

**Системы с дискретным временем, рассматриваемые на неограниченном промежутке.**

Отличие от предыдущего типа в том, что рассматриваются счетное множество различных значений времени, в которых может оказаться система. Траектория задается следующим образом:

Функционал примет следующий вид:

В данной работе будут использованы модели систем, рассматриваемые на конечном промежутке времени.

**Управление с обратной связью.**

Важной характеристикой управляемой системы является наличие так называемого контура обратной связи. В этом случае, часть информации с выходов системы подается снова на вход для последующего анализа и учета. Это позволяет, в ряде случаев, корректировать состояния системы в процессе регулирования. Коротко, обратная связь - это процесс, заключающийся в передаче информации о состоянии объекта управления к управляющему , либо регулирующему объекту. Управлению с обратной связью соответствует следующая схема на Рисунке 4:

|  |
| --- |
| http://infolike.narod.ru/img/picter30.gif  Рисунок 4 |
|

Задачу нахождения оптимального управления для системы с обратной связью иногда называют задачей синтеза. По наличию контура обратной связи, все управляемые системы делятся на два вида:

* замкнутые системы с наличием обратной связи
* открытые (разомкнутые) системы без контура обратной связи.

Многие объекты управления достаточно точно описываются линейными динамическими моделями с квадратичным функционалом качества. Путем грамотного подбора квадратичных критериев и ограничений можно синтезировать точные управляющие устройства с линейной обратной связью.

По направленности действия принято различать два вида управляющих устройств:

Терминальное управляющее устройство, в задачи которого входит приведение состояния управляемой системы к условиям, наиболее близким к заданным при постановке цели управления, в конечный момент процесса управления. При этом одновременно должно быть достигнуто приемлемое поведение системы в течение всего процесса управления.

Вторым видом устройств является регулятор, создаваемыйдля поддержания значений отклонения параметров управляемой системы от заданных условий в допустимых пределах путем использования оптимальных значений управляющих воздействий.

В главе 3 данной работы будут затронуты некоторые аспекты проектирования регулирующих устройств.

Описание некоторых особенностей моделирования управляемых систем и применение методов оптимального управления изложено в работах [28-32].

### §2. Задачи оптимального управления с запаздыванием.

Рассмотрим наиболее общую непрерывную задачу оптимального управления с запаздываниями. Динамика системы описывается дифференциальными уравнениями с запаздыванием в аргументе функции состояния и функции управления:

|  |  |
| --- | --- |
|  |  |

где – абсолютно непрерывные функции, – кусочно-непрерывные измеримые функции,

– *n-*мерные непрерывно-дифференцируемые вектор-функции.

Начальные условия для заданы на отрезках запаздывания:

|  |  |
| --- | --- |
|  |  |

где – заданные вектор-функции.

На функцию управления наложены ограничения:

|  |  |
| --- | --- |
|  |  |

Соотношения с компонентами запаздывания по состоянию системы и по управлению соответствуют большинству реальных объектов, где неминуемо возникают всякого рода задержки при распространении выходных и управляющих сигналов, а также при работе движущихся механизмов. Помимо самого наличия таких «лагов» следует учитывать тот факт, что их продолжительность так же может быть переменна, а иногда и зависеть от сопутствующих факторов. В таких случаях на состояние системы оказывает влияние не только действие, оказанное на предыдущем шаге, но и воздействия управляющих сил с предыдущих шагов. Происходит так называемое наложение.

В таких случаях принято рассматривать целых набор задержек, каждая из которых может являться функцией от времени и дополнительных параметров. Без учета этих особенностей невозможно построить модель сложного объекта, который может оказаться крайне чувствителен к качеству управляющего воздействия. Примером неправильно выбранного сигнала, без учета предыдущих корректировок, может являться отклонение движущегося объекта от заданной траектории из-за неправильного распределения тягового усилия. Эта ошибка может произойти и при конструировании простейшего робота на лабораторной работе, а может и при запуске ракеты.

Выбранное нами соотношение может обеспечить довольно высокую степень точности при моделировании различных систем. Все зависит лишь от правильного подбора параметров. Однако и такой, казалось бы, сложной модели может оказаться недостаточно в некоторых случаях. Одно из слабых мест – это запаздывания, которые в нашем случае являются фиксированными величинами, хоть и разными для управлений и состояний. Для реальных объектов требуется провести тщательный анализ с последующим выбором функциональной зависимости для каждого из видов запаздывания.

Помимо этого, часто могут накладываться определенные ограничения, например, минимизация расхода топлива (в объектах с двигателями внутреннего сгорания), минимизация отклонения от заданной траектории (при программировании автопилота), обеспечение удержания температуры объекта в заданных пределах (в химической промышленности), и т.д.

Все это может сделать выбор оптимального управления крайне сложной задачей, решения которой в общем виде может и не существовать. В случае с сложными системами могут применяться самые разнообразные методы приближенных вычислений и другие специальные способы, некоторые из которых будут затронуты в данной работе.

Пару функций будем называть допустимым процессом [5], если она удовлетворяет условиям (2.1)-(2.3). Множество всех допустимых процессов обозначим через Предположим, что . Среди всех допустимых процессов требуется найти процесс, который доставляет минимум функционалу на множестве

|  |  |
| --- | --- |
|  | (2.4) |

где – *n-*мерные непрерывно-дифференцируемые вектор-функции.

Введем следующие обозначения: оптимальный процесс, которому соответствует оптимальные траектории .

Сформулируем краевую задачу ПМП для задачи (2.1) - (2.4).

Выпишем функцию Понтрягина:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Согласно принципу максимума [35] для систем с запаздыванием, описание которого можно найти в работах [5, 36], оптимальное управление удовлетворяет следующему условию:

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

Применяя принцип максимума для задачи (2.1)-(2.4), определим оптимальное управление:

|  |  |
| --- | --- |
|  |  |

где сопряженные функции удовлетворяют системе дифференциальных уравнений с отклоняющимся аргументом:

|  |  |
| --- | --- |
|  |  |

С граничными условиями:

|  |  |
| --- | --- |
|  |  |

Принцип максимума позволяет свести задачу оптимального управления к краевой задаче для системы дифференциальных уравнений с запаздывающим и опережающими аргументами в функциях состояния и управления. Численные методы для решения краевых задач принципа максимума изложены в [3, 4].

Краевая задача принципа максимума для данной задачи будет иметь следующий вид:

|  |  |
| --- | --- |
|  | (2.9) |
|  | (2.10) |
|  | (2.11) |
|  | (2.12) |
|  | (2.13) |

### §3. Линейно-квадратичная задача оптимального управления с запаздыванием.

Рассмотренный выше случай является довольно обобщенным. На практике используются линейные модели в силу того, что ими проще оперировать и чаще всего их достаточно для описания довольно сложных систем. Детерминируем исходную задачу, положив:

|  |  |
| --- | --- |
|  | (3.1) |

Сделав это, мы получили задачу с линейными соотношениями и параллелепипедными ограничениями на управляющие воздействия. Это довольно частый прием, используемый на практике, так как линейные модели проще в расчетах и реализации, чем нелинейные. К тому же, данные предположения позволят более подробно изучить задачу. С учетом сделанных обозначений получим следующие выражения:

|  |  |
| --- | --- |
|  | (3.2) |

Минимизируемый функционал с квадратичной составляющей будет иметь следующий вид:

|  |  |
| --- | --- |
|  | (3.3) |

Принцип максимума Понтрягина будет иметь следующий вид:

|  |  |
| --- | --- |
|  |  |
|  | (3.4) |

Оптимальное управление может быть найдено следующим образом:

В выражение принципа максимума компонента управления входит во второй степени. Точку экстремума следует искать как вершину параболы, уравнение которой образуется слагаемыми для управления.

|  |  |
| --- | --- |
|  |  |

Положим :

|  |  |
| --- | --- |
|  |  |

Учитывая наложенные ограничения, оптимальное управление будет иметь вид:

|  |  |
| --- | --- |
|  | (3.5) |

где

|  |
| --- |
|  |

Выражения для сопряженных переменных будут иметь следующий вид:

|  |  |
| --- | --- |
|  | (3.6) |

Выпишем условие трансверсальности:

|  |  |
| --- | --- |
|  | (3.7) |

Теперь можно составить краевую задачу принципа максимума.

|  |  |
| --- | --- |
|  | (3.9) |
|  | (3.10) |
|  |  |
|  | (3.11) |
|  | (3.12) |
|  | (3.13) |

### §4. Задача оптимального управления системой с переменным запаздыванием

Во многих реальных задачах запаздывания зависят от времени. Рассмотрим один из таких случаев, когда запаздывание присутствует в компоненте состояния. Сформулируем задачу оптимального управления с переменны запаздыванием в аргументе функции состояния.

Состояния системы описываются следующими соотношениями:

|  |  |
| --- | --- |
|  | (4.1) |

где

Здесь – положительная, непрерывно-дифференцируемая функция, причем . Отсюда, в частности, следует, что функция монотонно возрастает и для нее существует обратная функция , являющаяся решением уравнения . Зададим начальные условия для системы:

|  |  |
| --- | --- |
|  | (4.2) |

где – заданное начальное множество.

Граничное условие имеет вид:

Управление является измеримой на функцией и удовлетворяет включению

|  |  |
| --- | --- |
|  | (4.3) |

Требуется минимизировать функционал:

|  |  |
| --- | --- |
|  | (4.4) |

на множестве допустимых процессов. Функции непрерывно дифференцируемы по совокупности аргументов.

Подобным образом можно описывать поведение динамических систем, в которых в зависимости от протекания процесса, характер запаздываний может меняться. Такую ситуацию можно наблюдать при протекании химической реакции, либо при удаленном управлении техникой, в случае если источник сигнала меняет свое удаление от объекта.

### §5. Дискретная аппроксимация для ЗОУ с запаздыванием.

В работе для построения оптимального решения используется метод быстрого автоматического дифференцирования [3]. В соответствии с этим методом, исходная задача аппроксимируется дискретной.

Аппроксимация системы дифференциальных уравнений осуществляется по схеме Эйлера, при этом величина отрезка разбиения , где – целые числа. Интеграл приближается по правилу левых прямоугольников. Далее введем обозначения: .

В этих предположениях, дискретная задача оптимального управления, аппроксимирующая задачу (1)-(4), имеет вид:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Для решения задачи применим метод множителей Лагранжа:

|  |  |
| --- | --- |
|  | (5.5) |

Условия стационарности:

|  |  |
| --- | --- |
|  | (5.6) |
|  | (5.7) |
|  | (5.8) |

Поделим на и осуществим предельный переход при:

|  |  |
| --- | --- |
|  | (5.9) |
|  | (5.10) |
|  | (5.11) |

Помимо этого, не редки случаи, когда управляемому объекту нельзя сопоставить какой-то фиксированный временной промежуток. В таких случаях возникает задача оптимального управления с нефиксированным временем процесса.

### §6. Дискретная аппроксимация для ЛКЗОУ с запаздыванием

Для задачи (3.1) – (3.3) устроим разбиение на отрезке времени , поделив его точками. Выберем разбиение таким образом, чтобы . Аппроксимацию производной осуществим по схеме Эйлера. Интегралы приблизим по правилу левых прямоугольников. Получим следующие соотношения:

|  |  |
| --- | --- |
|  | (6.1) |

где:

|  |
| --- |
|  |

Терминальное слагаемое функционала качества имеет вид:

|  |
| --- |
|  |

Начальные условия для функций состояния и управления, заданные на отрезках запаздывания определяются следующими соотношениями:

|  |  |
| --- | --- |
|  | (6.2) |

Множество задает параллелепипедные ограничения на управляющие воздействия:

|  |
| --- |
|  |

Квадратичный функционал качества для данной задачи имеет вид:

|  |  |
| --- | --- |
|  | (6.3) |

Воспользуемся методом множителей Лагранжа. Запишем функцию Лагранжа:

|  |
| --- |
|  |

Условие стационарности из метода множителей Лагранжа позволят найти рекуррентные соотношения для нахождения сопряженных переменных и выражение для нахождения значения градиента минимизируемой функции:

|  |  |
| --- | --- |
|  | (6.4) |
|  | (6.5) |
|  | (6.6) |

Поделим на и осуществим предельный переход при

|  |
| --- |
|  |
|  |

Выражения, полученные с помощью предельного перехода из соотношений для ДЗОУ должны совпасть с аналогичными из краевой задачи принципа максимума, что, в данной задаче, выполняется.

Полученные соотношения далее можно использовать при нахождении численного решения с помощью метода градиентного спуска. Сам метод и его применение для решения задачи будет описан далее.

### §7. ЗОУ с запаздыванием с нефиксированным временем процесса.

Рассмотрим исходную задачу оптимального управления (2.1)-(2.4) без запаздывания по управлению, как задачу с нефиксированным временем процесса. Это означает, что T – неизвестная величина.

Введем новую величину заменит рассматриваемую переменную t: (). *y*  теперь добавится к задаче как ограничение типа равенства.

С учетом этого получим уже задачу с фиксированным временем, где состояния системы опишут следующие дифференциальные уравнения:

|  |  |
| --- | --- |
|  | (7.1) |

где

с начальными условиями, заданными на промежутке запаздывания:

|  |  |
| --- | --- |
|  | (7.2) |

с ограничениями на управления:

|  |  |
| --- | --- |
|  | (7.3) |

В соотношениях выше:

|  |
| --- |
|  |

Минимизируемый функционал обретет следующий вид:

|  |  |
| --- | --- |
|  | (7.4) |

Составим функцию Понтрягина:

|  |
| --- |
|  |

Сформулируем принцип максимума:

|  |
| --- |
|  |

Оптимальное управление может быть найдено из рассмотрения следующего соотношения:

|  |
| --- |
|  |

Выпишем уравнения для сопряженных переменных:

|  |
| --- |
|  |

Условие трансверсальности:

|  |
| --- |
|  |

Данные соотношения позволяют сформулировать краевую задачу принципа максимума и решать её отдельно.

### §8. Дискретная аппроксимация для ЗОУ с запаздыванием с нефиксированным временем процесса.

Приведем задачу к дискретному виду, используя схему аппроксимации, использованную ранее и применим метод множителей Лагранжа. Задав разбиение на выделенном временном отрезке получим:

|  |  |
| --- | --- |
|  | (8.1) |

Состояния системы описываются следующими соотношениями:

|  |  |
| --- | --- |
|  | (8.2) |

Начальные условия, заданные на отрезке запаздывания:

|  |  |
| --- | --- |
|  | (8.3) |

С ограничениями на управления:

|  |  |
| --- | --- |
|  | (8.4) |

Дополнительная переменная, включенная в задачу как ограничение типа равенства:

|  |  |
| --- | --- |
|  | (8.5) |

Составим функцию Лагранжа:

|  |
| --- |
|  |

Условия стационарности имеют следующий вид:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Поделив , на и осуществив предельный переход при получим следующие соотношения:

|  |
| --- |
|  |
|  |

Как и в случае, где изначально время в задаче было фиксированной величиной, выражения для сопряженных переменных совпали с аналогичными из принципа максимума. Задачи с нефиксированным временем процесса называют задачами быстродействия.

Дальнейшее исследование и разработка методов решения ЗОУ с запаздываниями в компонентах состояния и управления подразумевает рассмотрение разрывных ЗОУ

### §9. Метод градиентного спуска

Для построения численного решения дискретной задачи оптимального управления в работе используется метод градиентного спуска. С применением соотношений, полученных при рассмотрении дискретных аппроксимаций, организуется итерационный процесс, дающий приближенное численное решение. Метод градиентного спуска позволяет получить минимум функции, проведя оптимизацию в направлении антиградиента.

В задачах, рассматриваемых в данной работе, необходимо минимизировать функционал Для этого используется следующее соотношение:

|  |  |
| --- | --- |
|  | (9.1) |

где – шаг спуска, – значение градиента минимизируемой функции на -й итерации.

Для вычисления значения градиента используется следующая формула:

|  |  |
| --- | --- |
|  | (9.2) |

Шаг выбирается таким образом, чтобы вносимая поправка была соразмерна с регулируемыми величинами. Существует несколько основных способов выбора шага градиентного спуска:

1. может быть выбрано постоянным числом, в этом случае сходимость метода не гарантируется,
2. может быть дробным, для этого при переходе на очередную итерацию его делят на некоторое число,
3. может быть найдено с использование метода наискорейшего спуска, в этом случае:

|  |
| --- |
| . |

Одним из важных моментов в данном методе является критерий остановки, влияющий на то, когда стоит прекратить вычисления. Ниже будут приведены одни из основных критериев:

Стоит заметить, что чаще всего используется совокупность критериев, а не какой-то один. Это позволяет добиться лучшего результата. Также стоит упомянуть про такой параметр метода как , называемый точностью. Его стоит выбирать наименьшим из возможных при реализации. То, насколько маленьким будет это число, зависит от вычислительных мощностей и требуемой точности к результату.

Помимо описанных выше критериев, важно также следить за тем, чтобы при осуществлении итерационного процесса шаг был сделан в нужную сторону. К примеру, при поиске минимума функции необходимо проверять разницу между значениями целевой функции на двух последовательных итерациях. Если условие при минимизации функции не будет выполнено, то можно сделать вывод, что шаг градиентного спуска требует корректировки. Необходимо сделать поправки в соответствии с выбранным методом, к примеру, уменьшить шаг, поделив его на некоторое число, и повторить итерацию с новым шагом.

Частичная блок-схема алгоритма метода градиентного спуска с одним критерием остановки предоставлена на Рисунке 5.

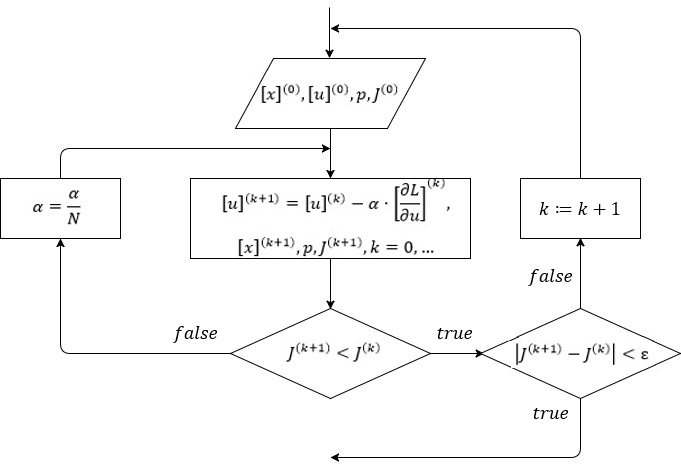


Рисунок 5

Частичная блок-схема алгоритма метода градиентного спуска с несколькими критериями остановки предоставлена на Рисунке 6.

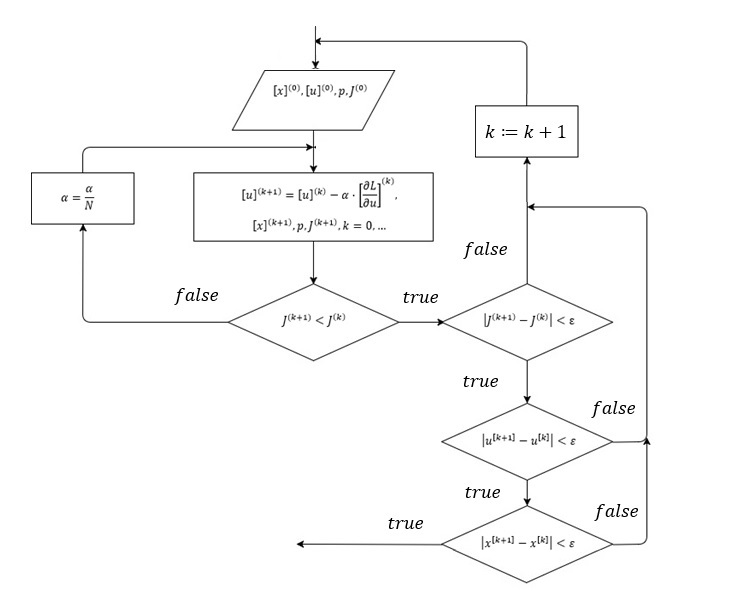


Рисунок 6

## Глава 3. Метод динамического программирования

### §1. Принцип оптимальности и уравнение Беллмана

В середине ХХ века американский ученый-математик Р. Беллман применил к ряду практических задач оптимального управления прием, который в дальнейшем назовут принципом оптимальности. Основной областью его применения являются итерационные процессы с большим количеством шагов, т. н. процессы, развивающиеся во времени. Именно описание систем, изменяющихся во времени, дало основание назвать новый метод оптимизации динамическим. Новый динамический метод отличался от линейного и математического программирования, формулировка задач которых имела подразумевала описание статичного объекта, либо фиксированного состояния системы. Подробное описание метода приведено в монографии [45] и работах [37-39].

Принцип оптимальности, сформулированный Р. Беллманом, гласит следующее:

Каково бы ни было состояние S системы в результате какого-либо числа шагов, на ближайшем шаге управляющее воздействие выбирается таким образом, чтобы оно в совокупности с оптимальным управлением на всех последующих шагах приводило к оптимальному выигрышу на всех оставшихся шагах, включая данный.

Основным приемом для исследования и решения задачи синтеза управления, помимо самого принципа оптимальности, является идея помещения задачи оптимизации в семейство аналогичных задач. Таким образом, можно осуществить разбиение решения ЗОУ на ряд этапов. Еще одной из особенностей, которая выделяет метод Беллмана из числа остальных методов оптимизации, является форма получения конечного результата. С использованием принципа оптимальности, а также погружая исходную задачу в множество подобных ей, но с некоторыми уточнениями, получаются рекуррентные соотношения относительно оптимального значения функционала качества. Получаемые таким образом уравнения позволяют вычислить значения оптимального управления для ЗОУ последовательно, шаг за шагом. Таким образом, исходная ЗОУ разделяется на множество более простых задач, суть решения которых заключается в поиске оптимального управления отдельно на каждом из этапов.

Р. Беллманом также были сформулированы и основные требования к исследуемой системе, при которых выполняется описанный выше принцип.

* Объектом исследования должна быть управляемая сис­тема с заданными ограничениями на состояния и управления*;*
* задача может интерпретироваться как многошаговый процесс, каждый шаг заключается в принятии реше­нияо выборе одного из управлений;
* Процесс управления не должен являться процессом с обратной связью. То есть состояние, в котором оказывается система пос­ле выбора решения на новом шаге, зависит только от данного решения и состояния системы к началу этого шага. Иногда это свойство формулируют, как отсутствие последействий.

Принцип оптимальности утверждает, что для любого процесса без обратной связи оптимальное управление таково, что оно является оптимальным для любого подпроцесса по отношению к исходному состоянию этого подпроцесса. То есть оптимальный процесс, отвечающий требованиям и целям управляемой системы, определяется лишь начальными условиями и конечной целью управления. Поэтому решение на каждом шаге оказывается наилучшим с точки зрения управления в целом. Если изобразить геометрически оптимальную траекторию в виде ломаной линии, то любая часть этой ломаной будет являться оптимальной траекторией относительно начала и конца.

Несмотря на упомянутое выше условие, требующее отсутствия в модели ветви обратной связи, в работе [40] доказано, что формулировка принципа оптимальности останется справедливой и для систем с запаздыванием по управлению, если в рассмотрение состояния системы в текущий момент времени включить предысторию изменения фазовых координат системы на промежутке времени запаздывания. Таким образом, синтез оптимального управления для ЗОУ с запаздыванием возможен, однако требует учета дополнительных ограничений.

Рассмотрение возможности применения принципа Беллмана для синтеза оптимального управления в задаче с запаздыванием в функциях состояния и управления – это тема отдельного исследования. В данной работе ограничимся формулировкой принципа оптимальности для линейно-квадратичной задачи оптимального управления без запаздываний.

Таким образом, обеспечивается следующее: значения управления, являющееся оптимальным для конкретного шага, является также оптимальным для всех последующих шагов в целом.

### §2. Уравнение Беллмана для линейно-квадратичной задачи оптимального управления.

Рассмотрим ЗОУ, в которой требуется найти минимум квадратичного функционала:

|  |  |
| --- | --- |
|  | (2.1) |

при заданных линейных ограничениях типа равенства:

|  |  |
| --- | --- |
|  | (2.2) |

где – фиксированные значения времени, – матрицы с постоянными коэффициентами, – состояние системы.

Для введем следующую функцию:

|  |  |
| --- | --- |
|  | (2.3) |

для которой будет выполнено следующее:

1. доставит минимум функционала при переходе из одного состояния в другое для линейно-квадратичной задачи оптимального управления, если система рассматривается, начиная с состояния системы в момент времени , отличные от начальных.
2. доставляет минимум линейно-квадратичного функционала качества (на состояние в момент времени ).

Можно установить, что:

1. квадратична, то есть, где
2. можно найти рекурсивно, начиная с
3. оптимальное управление для линейно-квадратичной задачи может быть легко выражено при нахождении .

Введенную выше функцию называют функцией Беллмана.

При , то есть в конечный момент времени, значение функции Беллмана совпадает с терминальным слагаемым:

.

Учитывая это, можно получить

Формализуем теперь основной принцип динамического программирования. Сделаем несколько замечаний по поводу значений, которые могут быть известны:

1. Предположим, что известно значение функции на шаге:
2. При выборе оптимального значения управления на шаге : необходимо учитывать следующие факторы:
   1. так называемые, понесенные «расходы» (издержки) или тормозящие воздействия, оказываемые на систему, посредством влияния слагаемого
   2. состояние, в котором находится система в момент :

Последнее подразумевает необходимость того, чтобы воздействие было минимальным при переходе от состояния .

Принцип динамического программирования формально можно записать в следующем виде:

|  |  |
| --- | --- |
|  | (2.4) |

где:

* описывает тормозящие воздействия, учтенные в момент времени при ,
* значение функции Беллмана является минимальным для шага *.*

Заметим, что минимизация функции осуществляется по следующему правилу:

|  |
| --- |
|  |

Таким образом, выполняется основное свойством принципа оптимальности: значения управления, являющееся оптимальным для конкретного шага, является также оптимальным для всех последующих шагов в целом.

Таким образом, уравнение Беллмана**,** определяется следующим соотношением:

|  |  |
| --- | --- |
|  | (2.5) |

Данное уравнение называют уравнением динамического программирования, уравнением Беллмана, или уравнением Гамильтона-Якоби. Значения функции определяются рекурсивно, в обратном порядке, от .

Применим, полученные выше соотношения для описания решения ДЗОУ с использованием метода динамического программирования.

Решение задачи состоит в минимизации следующего квадратичного функционала:

|  |  |
| --- | --- |
|  | (2.6) |

Состояния системы описываются соотношениями:

|  |  |
| --- | --- |
|  | (2.7) |

где

С ограничениями на состояния и управления общего вида:

|  |  |
| --- | --- |
|  | (2.8) |

– заданные матрицы с постоянными коэффициентами.

Начальное состояние системы описывается:

|  |
| --- |
|  |

Функция Беллмана для данной задачи имеет вид:

|  |  |
| --- | --- |
|  | (2.9) |

Используя терминальное слагаемое из квадратичного функционала качества запишем:

|  |  |
| --- | --- |
|  | (2.10) |

Аналогично:

|  |  |
| --- | --- |
|  | (2.11) |

Решив данную задачу до , получается последовательность

|  |
| --- |
|  |

Данную процедуру и называют синтезом оптимального управления. Однако на каждом шаге получается не одно единственное управление, а множество, что подразумевается принципом оптимальности. Теперь, имея данную последовательность и обладая фиксированным начальным значение можно организовать обратный процесс, получая на каждом шаге одно значение управления, удовлетворяющее заданному начальному условию.

### §3. Использование линейно-квадратичного регулятора в системах автоматического регулирования.

Системы автоматического регулирования (САР) применяются для регулирования отдельных параметров в объекте управления. В современных системах автоматического управления (САУ) системы автоматического регулирования являются подсистемами САУ и их применяют для регулирования различных параметров при управлении объектом или процессом. Такими параметрами могут быть самые различные величины, встречающиеся в реальной жизни. Это может быть и температура в контуре системы охлаждения какого-то устройства, и скорость движения транспортного средства, и высота полета летательного аппарата.

Принцип функционирования САР основывается на обнаружении отклонений в значениях регулируемых величин от эталонных, которые характеризуют работу управляемой системы или характер протекания процесса. Задачей регулятора является обеспечение необходимых управляющих воздействий на объект или процесс с целью устранения, либо минимизации этих отклонений.

Для осуществления автоматического управления в систему интегрируется устройство-регулятор, отвечающий за осуществление управляющего воздействия. Это управляющее воздействие вырабатывается регулятором на основании различия между текущим значением регулируемой величины, измеряемой сенсорами системы, и желаемым её значением, устанавливаемым исходя из требований к функционированию системы. Наиболее часто, в качестве регулируемых параметров выступают такие физические величины как температура, давление, уровень жидкости, скорость передвижения и др. Совокупность регулятора и управляемого объекта образует САР.

Основным признаком системы с автоматическим регулированием является наличие канала обратной связи, по которой регулятор обеспечивает контроль значений регулируемых параметров.

Существует множество разновидностей регуляторов, каждый из которых имеет свои плюсы и недостатки. Остановимся подробнее на применении линейно-квадратичного регулятора.

**Линейно-квадратичный регулятор** (англ. Linear quadratic regulator, LQR) — в теории управления один из видов оптимальных регуляторов, использующий квадратичный функционал качества.

Одним из примеров использования ЛКР может служить задача по управлению движением некоторого объекта. В работе [20] рассмотрена возможность применения линейно-квадратичной модели для создания регулятор, управляющего рулевыми механизмами спутников. В работе доказывается, что алгоритм управления на основе линейно-квадратичного регулятора успешно справляется с задачей перевода спутника с одной относительной орбиты на другую. При этом оговаривается условие, что следует внимательно относится к параметрам регулятора, так как при некоторых значениях величина коррекции становится недостижимой.

Весьма популярной сферой применения линейно-квадратичных регуляторов является задача управления полетом квадрокоптера. В работе [42] автор показывает возможность реализации управления летательным аппаратом с помощью пропорционально-интегрально-дифференцирующего и линейно-квадратичного регуляторов. В ходе опытов было произведено сравнение двух способов регулирования в результате чего было установлено, что хоть ПИД регулятор и позволяет совершать более быстрые изменения, но ЛКР выигрывает в точности и стабильности.

В работе [43] рассматривалась аналогичная задача оптимального управления. В ней было продемонстрировано, что LQR-регуляторы применимы к нелинейной математической модели, описывающей движение летательного аппарата.

## Глава 4. Оптимизация ИНС

### §1. Описание модели ИНС, постановка задачи.

Как было сказано выше, искусственные нейронные сети являются эффективным инструментом при решении ряда прикладных задач. Область применения постоянно расширяется, однако, на данный момент, не существует универсального инструмента, основанного на теории нейронных сетей, с помощью которого можно было бы охватить все проблемы сразу.

Часто при решении конкретной прикладной задачи с использованием аппарата искусственных нейронных сетей возникает потребность в моделировании сети нетривиальной топологии. Стандартные методы обучения нейронных сетей могут оказаться в данной ситуации неприменимыми, и поэтому необходимо сформулировать и реализовать индивидуальный алгоритм.

Рассмотрим задачу оптимального управления, моделирующую динамику искусственной нейронной сети с запаздыванием в фазовой компоненте и в компоненте управления. В работе [44] рассматривалась модель ИНС с запаздыванием в фазовой компоненте.

Динамика сети из n нейронов описывается системой обыкновенных дифференциальных уравнений с запаздываниями в компонентах состояния и управления:

|  |  |
| --- | --- |
|  | (1.1) |

где – собственный потенциал нейрона в данный момент времени,

– собственное затухание нейрона, описывающее воздействие на нейрон собственных сил, которые негативно влияют на потенциал.

Суммы и называют телами нейронов. Данная величина показывает идеальный потенциал нейрона, без какого-либо воздействия на него.

Функции управления , описывают аксоны нейронов – электрический или химический импульс который передается от одного нейрона к другим и является важнейшим связующим элементом всей сети, так как отвечает за взаимодействие и работоспособность всей сети.

Функции управления характеризуют внешнее воздействие на нейрон.

Характеристики нейронов в начальный момент времени:

|  |  |
| --- | --- |
|  | (1.2) |

Характеристики внешних управляющий воздействий в начальный момент времени:

|  |  |
| --- | --- |
|  | (1.3) |

где – заданные вектор-функции.

На весовые коэффициенты и внешние управляющие воздействия наложены следующие ограничения:

|  |  |
| --- | --- |
|  | (1.4) |

где – заданные числовые константы.

Целью управления динамикой нейронной сети является обучение сети, которое подразумевает следующие задачи:

1. В конечный момент времени характеристики нейронов должны совпадать с входными данными
2. Минимизация управлений.

Задачи управления можно сформулировать в виде следующего целевого функционала

|  |  |
| --- | --- |
|  | (1.5) |

Основной практической целью является получение оптимальных управлений процесса, при помощи которых достигается минимум целевого функционала.

Для решения задачи оптимального управления (1.1)-(1.5) применим принцип максимума Понтрягина.

Введем функцию Понтрягина:

|  |  |
| --- | --- |
|  | (1.6) |

Применяем принцип максимума:

|  |
| --- |
|  |

Найдем оптимальные управления:

|  |
| --- |
|  |

Графиком функции управления будет парабола, ветви которой направлены вниз. Точкой максимума будет являться вершина:

|  |
| --- |
|  |

Положим Оптимальное управление будет иметь вид:

|  |
| --- |
|  |
|  |

Рассуждая аналогичным образом, получим:

|  |
| --- |
|  |

Оптимальные весовые коэффициенты:

|  |
| --- |
|  |

Получим выражения для сопряженных переменных:

|  |
| --- |
|  |

Условие трансверсальности:

|  |
| --- |
|  |

Все это позволяет сформулировать краевую задачу принципа максимума:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

### §2. Дискретная аппроксимация.

Для построения оптимального решения рассмотрим дискретную аппроксимацию исходной задачи. Аппроксимация системы дифференциальных уравнений осуществляется по схеме Эйлера, при этом , где – целые числа. Интегралы приближаются по правилу левых прямоугольников. Далее введем обозначения: .

В результате мы получим задачу оптимального управления, которая состоит в минимизации функционала:

|  |  |
| --- | --- |
|  | (2.1) |

при фазовых ограничениях:

|  |  |
| --- | --- |
|  | (2.2) |

С начальными условиями, заданными на промежутках запаздывания:

|  |  |
| --- | --- |
|  | (2.3) |

С ограничениями на управляющие воздействия:

|  |  |
| --- | --- |
|  | (2.4) |

Для решения дискретной задачи оптимального управления с запаздыванием воспользуемся методом множителей Лагранжа. Составим функцию Лагранжа:

|  |
| --- |
|  |

Получим рекуррентные соотношения для сопряженных переменных и найдем градиент минимизируемой функции с помощью дифференцирования функции Лагранжа:

|  |  |
| --- | --- |
|  | (2.5) |
|  | (2.6) |
|  | (2.7) |
|  | (2.8) |

Так как в задаче выбраны ограничения на управления параллелепипедного типа (2.4), то при улучшении управляющих воздействий будет использован метод проекции градиента. При использовании других ограничений их следует включать в функционал с использованием метода штрафных функций.

С использованием полученных соотношений составим алгоритм построения приближенного оптимального решения задачи оптимального управления:

1. Задать начальные значения внешних управляющих воздействий и весовых коэффициентов на промежутках запаздывания для нулевой итерации. Также зафиксировать начальные состояния системы на интервале запаздывания. Далее определяет номер итерации.
2. Вычислить состояния системы на *r*-й итерации согласно (2.2).
3. Найти значение целевой функции на *r*-й итерации по формуле (2.1).
4. Вычислить значения сопряженных векторов на *r*-й итерации по формулам (2.5)-(2.6), начиная с компоненты на последнем слое разбиения
5. Используя соотношения (2.7)-(2.8) для вычисления градиента минимизируемой функции определить внешние управляющие воздействия и весовые коэффициенты нейронной сети на -й итерации:

|  |  |
| --- | --- |
|  |  |
|  |  |

где – шаг спуска.

1. Далее, с улучшенными наборами управлений повторяются этапы 2-4 для нахождения
2. Сравниваются значения целевой функции . Если то считается, что шаг был сделан удачно и необходимо проверить, выполнилось ли условие остановки. В данной задаче выбрано условие:

|  |
| --- |
|  |

где число отвечает за точность вычислений. Данную величину следует выбирать наименьшей из возможных и она ограничена лишь вычислительной мощностью используемой платформы.

Если при сравнении значений целевой функции на шаге 7 выполнилось условие , то производится корректировка шагов спуска и повторение шагов 5-7. Данные действия повторяются, пока не будет выполнено условие на шаге 7.

### §3. Блок-схема работы алгоритма.

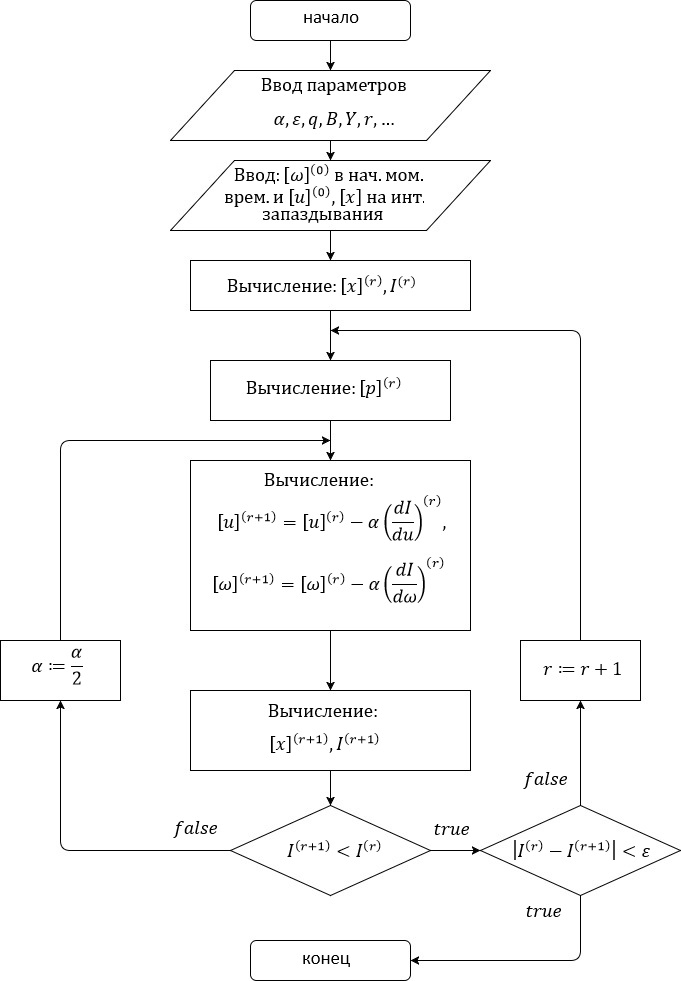


Рисунок 7

## Глава 5. Программная реализация алгоритма обучения ИНС с запаздыванием.

### §1. Описание разработки программы.

При написании программы, реализующей описанный выше алгоритм построения приближенного оптимального решения задачи оптимального управления с запаздыванием был использован язык программирования C++. В качестве среды разработки выбрана платформа Qt 5.9, позволяющая создавать приложения с графическим интерфейсом.

Для построения графиков функций была использована дополнительная библиотека QCustomPlot, подключающаяся к приложению в качестве отдельного модуля.

Пользовательский интерфейс разработанного приложения предоставлен на Рисунке 8:

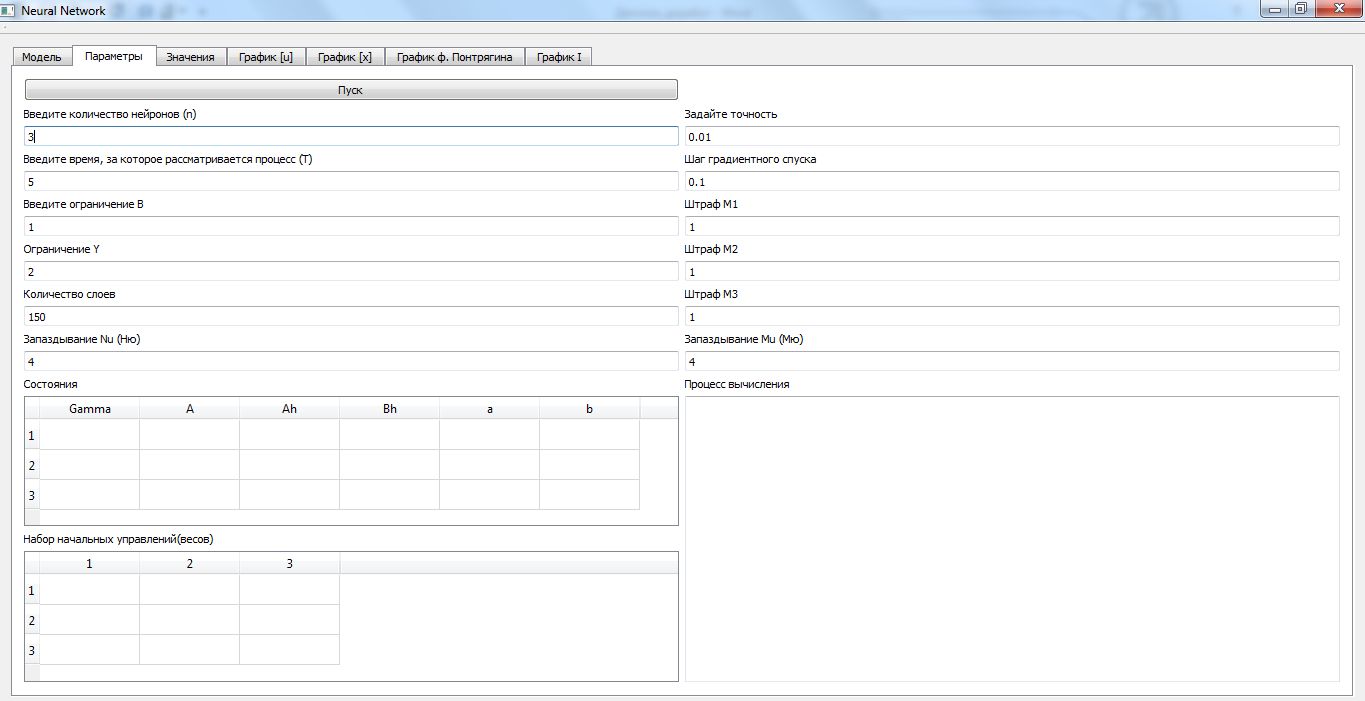


Рисунок 8

Опустим рассмотрение вкладки с названием «Модель», содержащей описание математической модели ИНС с запаздываниями из главы 4. Вкладка «Параметры» предоставляет пользователю возможность ввода параметров метода оптимизации, таких как точность вычислений и шаг градиентого спуска, а также параметров математической модели ИНС, к которым относится количество слоев нейронной сети, ограничения на управляющие воздействия и др. При первом запуске приложения все поля будут проинициализированы начальными значениями, которые можно менять по своему усмотрению. Меняя значения в полях для ввода запаздываний можно рассматривать различные ЗОУ. К примеру, «занулив» одно из значений можно рассматривать модель системы с запаздыванием либо по состоянию, либо по управлению.

Таблица с названием «Состояния», представленная на Рисунке 9 служит для задания величин затуханий , конечных состояний системы , а также числовые коэффициенты благодаря которым можно включать или исключать из рассмотрения слагаемые с запаздываниями.

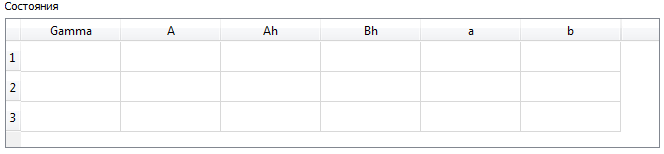


Рисунок 9

Таблица окна программы на Рисунке 10 необходима для ввода начальных значений весовых коэффициентов нейронной сети.

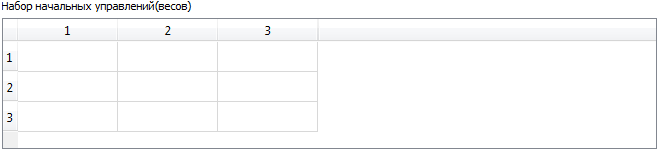


Рисунок 10

Незатронутыми при описании остались лишь начальные значения состояний системы и внешних управляющих воздействий, задаваемые на промежутках запаздываний. Для того, чтобы «облегчить» внешний вид приложения, было принято решение перенести ввод этих значений в отдельные файлы с названиями «init\_x.txt» и «init\_u.txt» для состояний и управлений соответственно. Внешний вид этих файлов представлен на Рисунке 11.

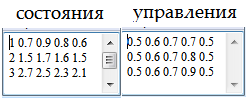


Рисунок 11

В окне с названием «Процесс вычисления» будет отображаться ход вычислений с выводом подсчитанных значений функционала качества, номера итерации, а также затраченного времени.

Для запуска вычислений служит кнопка «Пуск».

После завершения вычислений, результаты можно просмотреть в других вкладках приложения. Так, вкладка «Значения» содержит таблицу, в которой помещены значения траектории системы и внешних управляющих воздействий, полученные на последнем шаге. По этим значениям можно достаточно точно определить результат оптимизации и проследить изменение управления.

Далее идут вкладки с графиками функций управления, состояния, функции Понтрягина и целевого функционала. Каждый из графиков интерактивен, что дает возможность приближать или отдалять необходимый участок. Стоит заметить, что график функции Понтрягина будет особенно полезен при решении ЗОУ без запаздываний. В этом случае, график должен совпадать с графиком константы 0 с возможными минимальными отклонениями, то есть когда амплитуда колебаний не превышает хотя бы значения .

### §2. Описание численных экспериментов.

Рассмотрим задачу оптимизации ИНС, описываемой заданной выше моделью с запаздываниями в компонентах состояния и управления, состоящую из 3 нейронов. Параметры предоставлены на Рисунке 12 Начальные состояния и внешние управляющие воздействия системы, заданные на отрезках запаздывания, взяты из файлов, содержимое которых отражено на Рисунке 11.

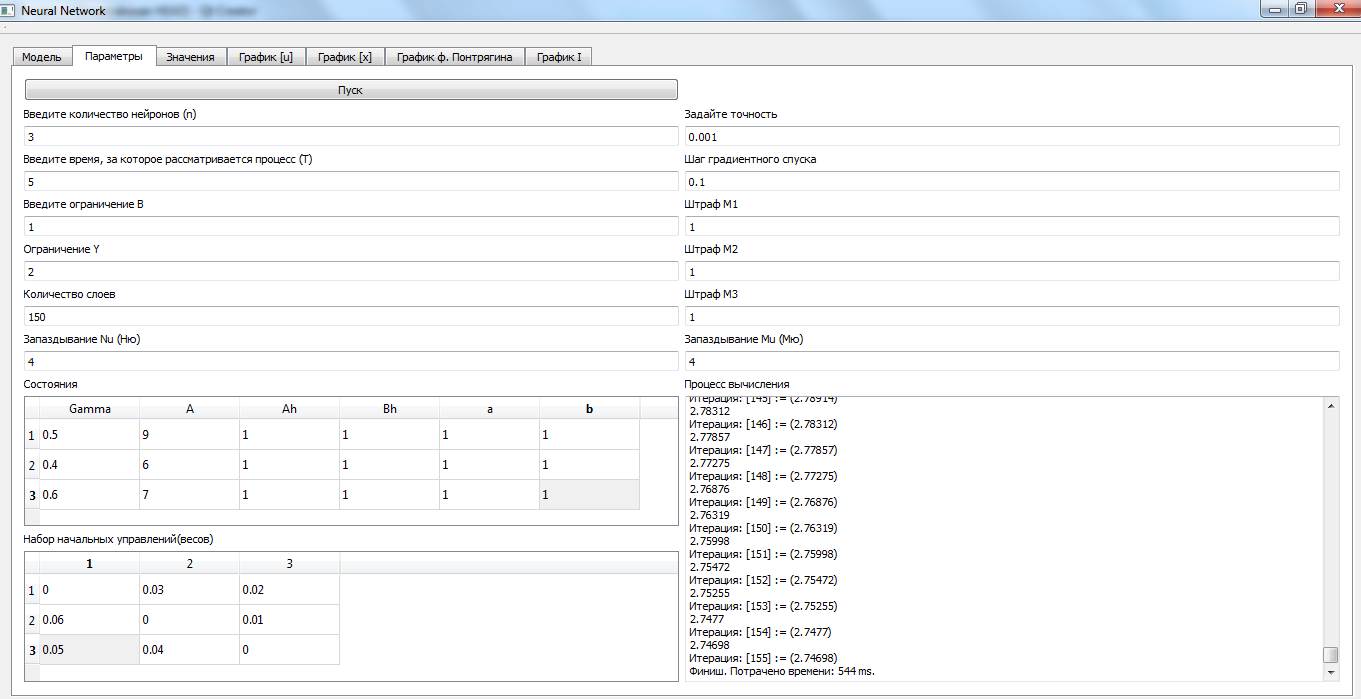


Рисунок 12

Будем считать, что это контрольный замер. Далее будем менять параметры модели и метода, что даст возможность проанализировать, как это сказывается на работе программы.

Видно, что необходимая точность вычислений в ходе работы была достигнута за 155 итераций. Программе на это понадобилось 544 мс.

Значения состояний и управлений на последней итерации предоставлены на Рисунке 13.

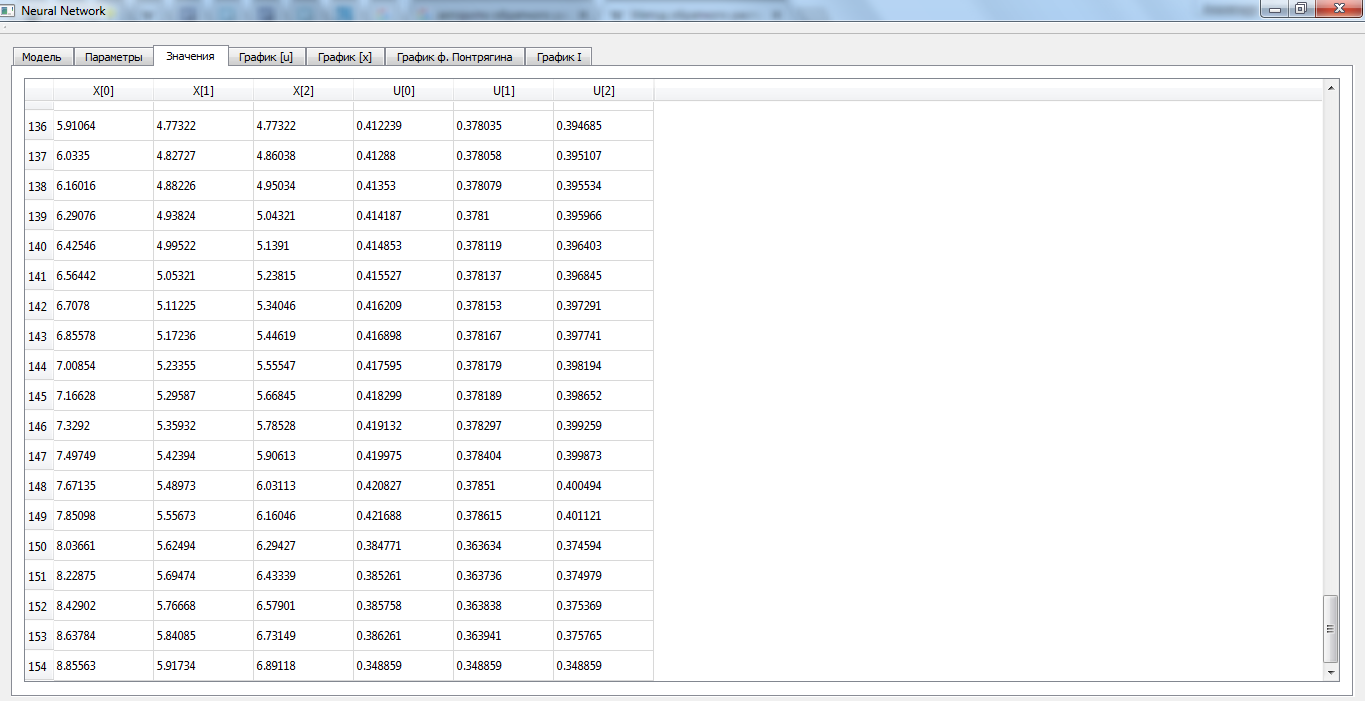


Рисунок 13

Графики функций управления, состояния, Понтрягина, а также целевого функционала предоставлены на Рисунках 14-17.

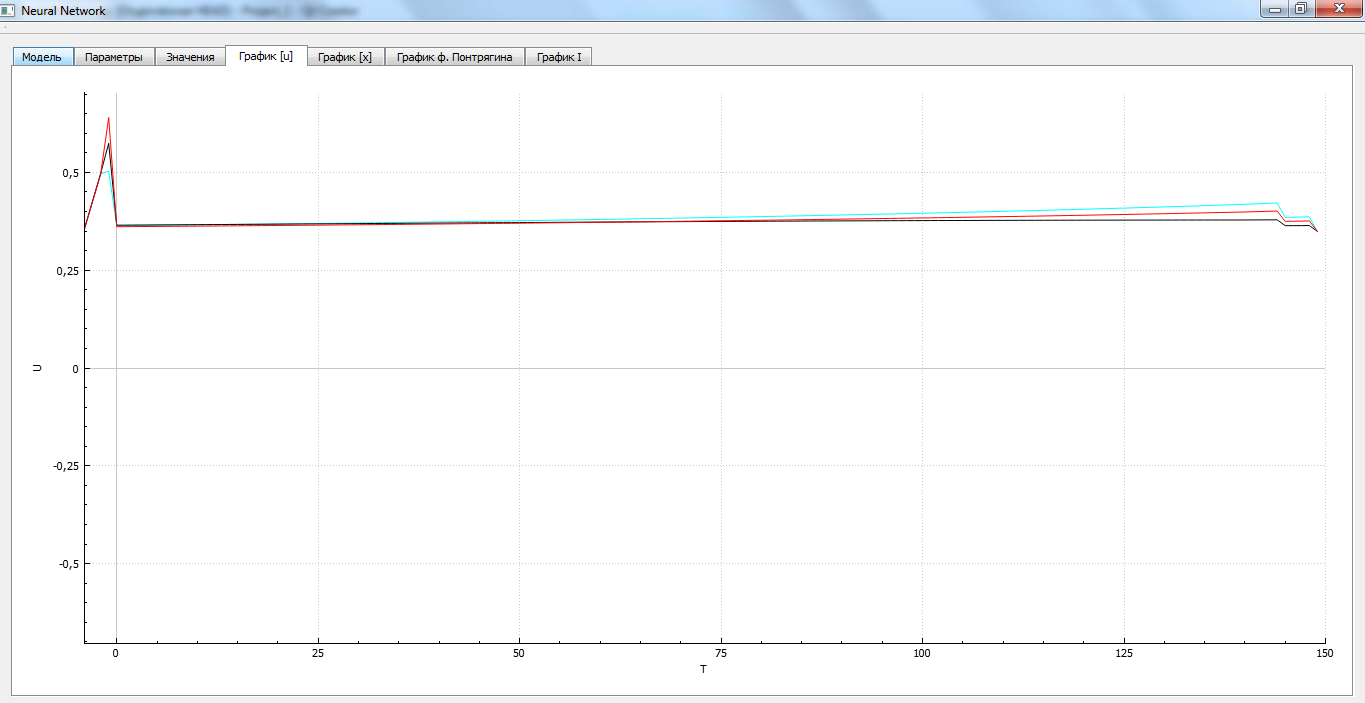


Рисунок 14

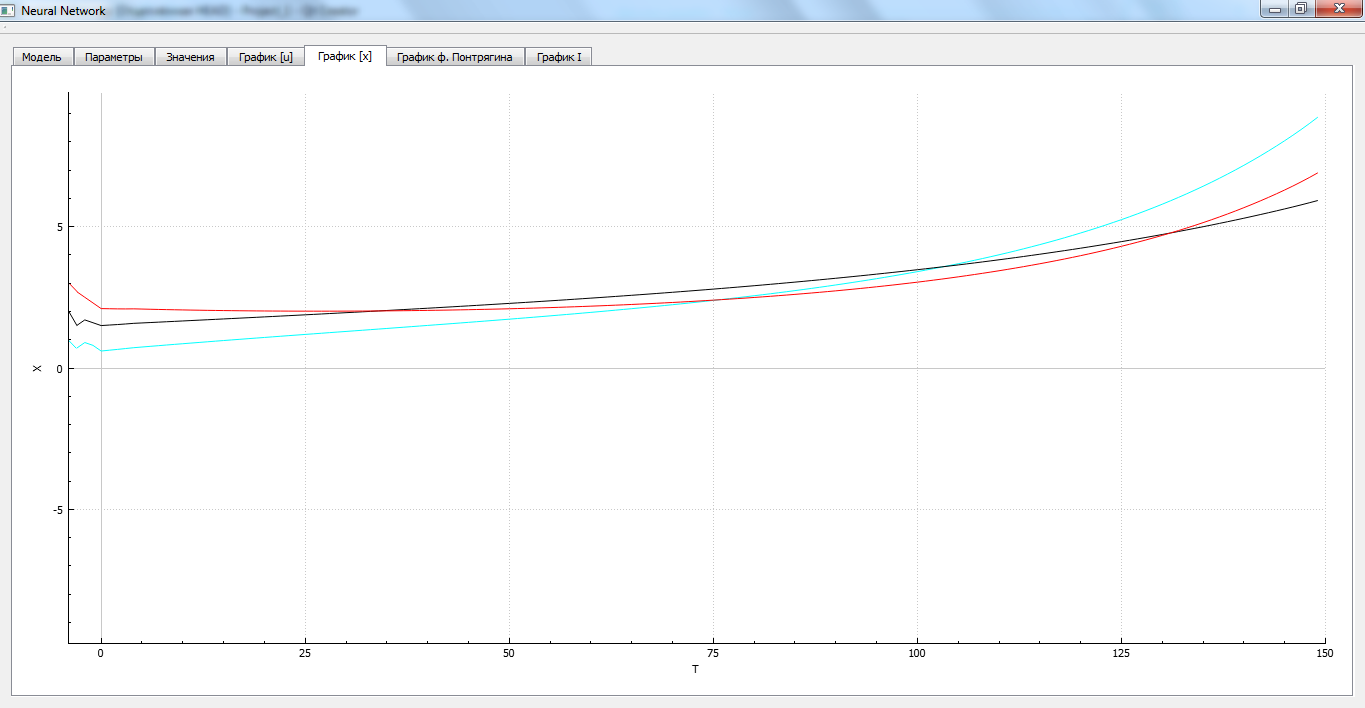


Рисунок 15

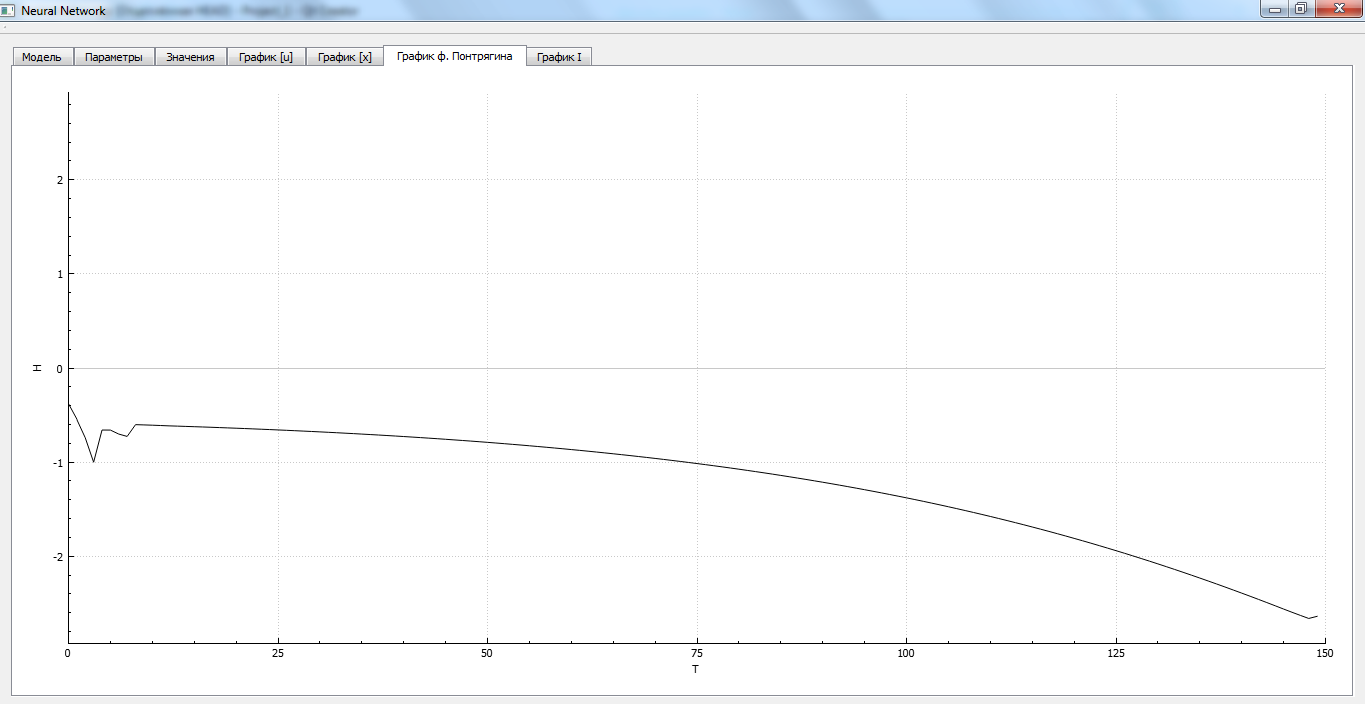


Рисунок 16

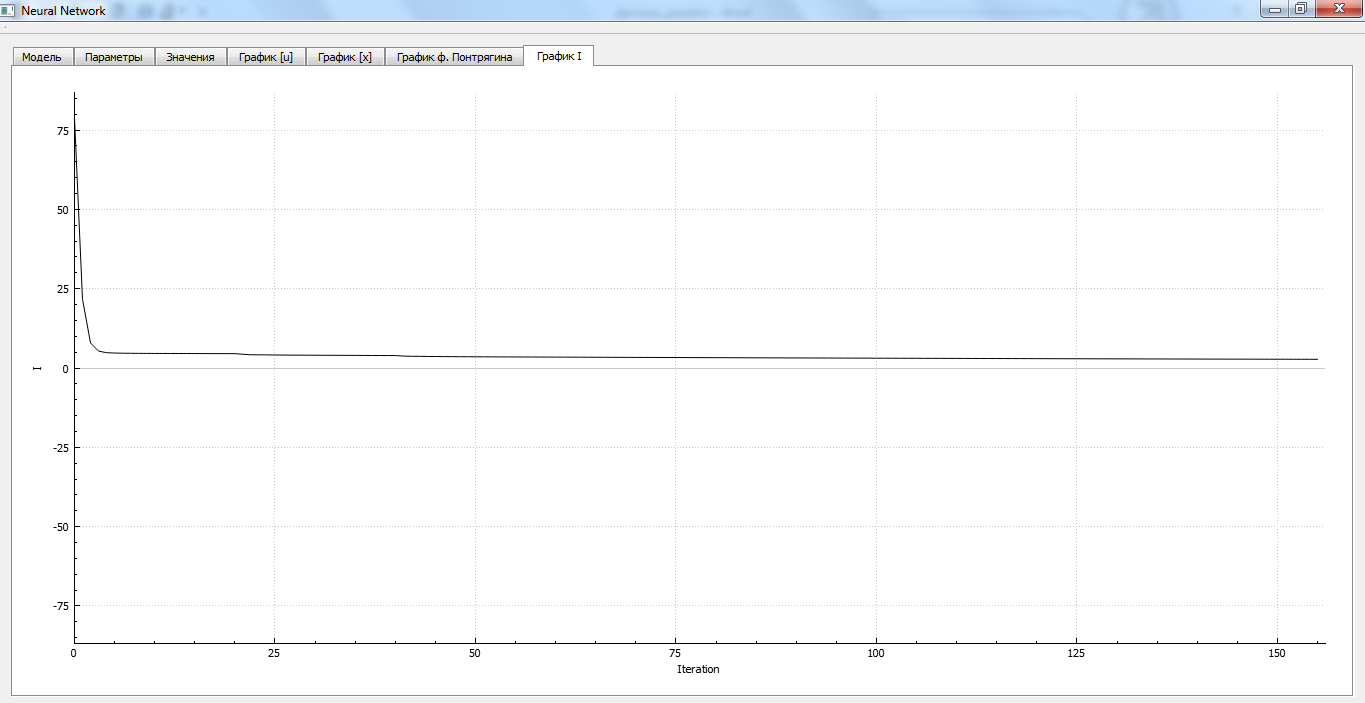


Рисунок 17

Из результатов, предоставленных на таблице Рисунка 13 видно, что в процессе оптимизации, состояния сети не «пришли» в заданную точку. Это объясняется выбором штрафного коэффициента , отвечающего как раз за свойство попадания в конечную точку. На графиках функций состояния и управления явно видны начальные условия, заданные на отрезках запаздывания. Вид графика функции Понтрягина будет полезен при рассмотрении систем без последствий. По графику функционала качества можно наглядно убедиться в том, что в ходе вычислений действительно произошла минимизация функционала. Об этом же говорят и сами значения функционала на вкладке «Параметры».

Увеличим значение параметра , положим его равным 10. В этом случае для завершения потребуется значительно меньше итераций – 15, а программа завершит свою работу за 44 мс. Однако конечные состояния сети практически совпадут с требуемыми, в чем можно убедится, проанализировав значения на Рисунке 18.

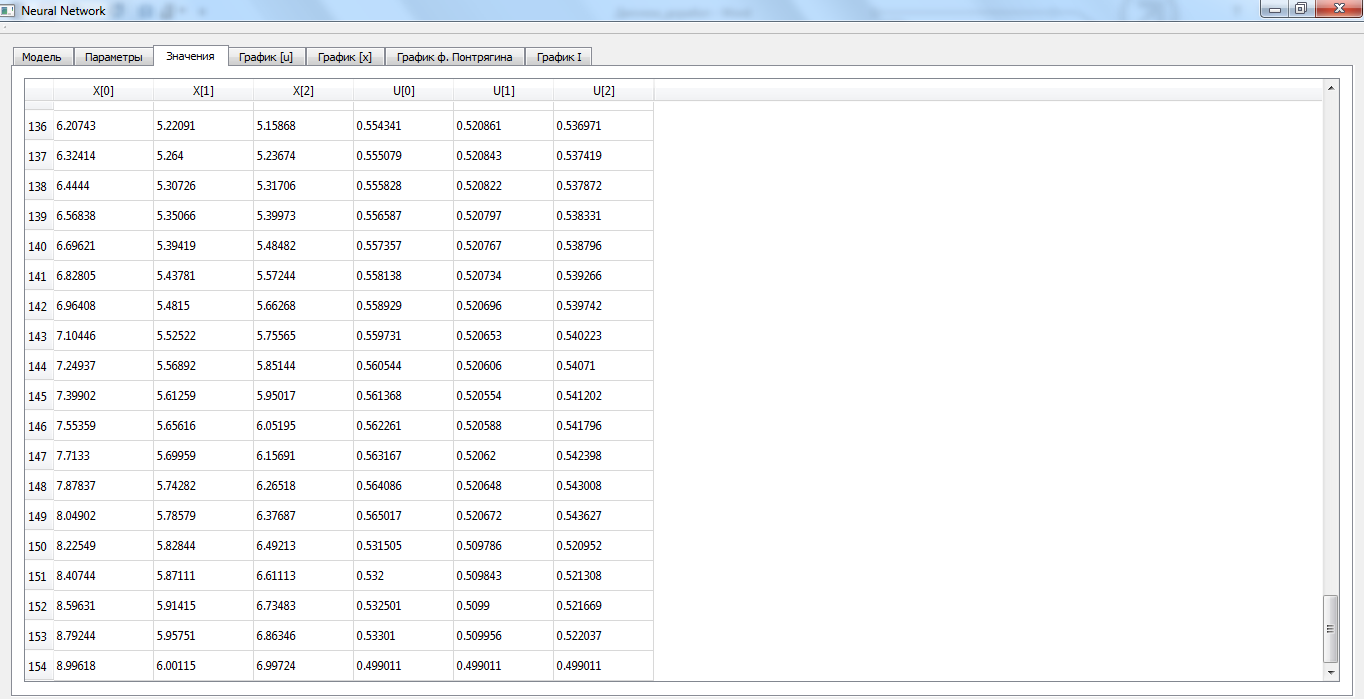


Рисунок 18

Внешний вид графиков функций управления, состояния и целевого функционала кардинально не отличаются от полученных ранее. Изменился график функции Понтрягина. На Рисунке 19 видно, что на некотором протяжении функция близка к константному значению. Подобное наблюдается в задачах без запаздываний.

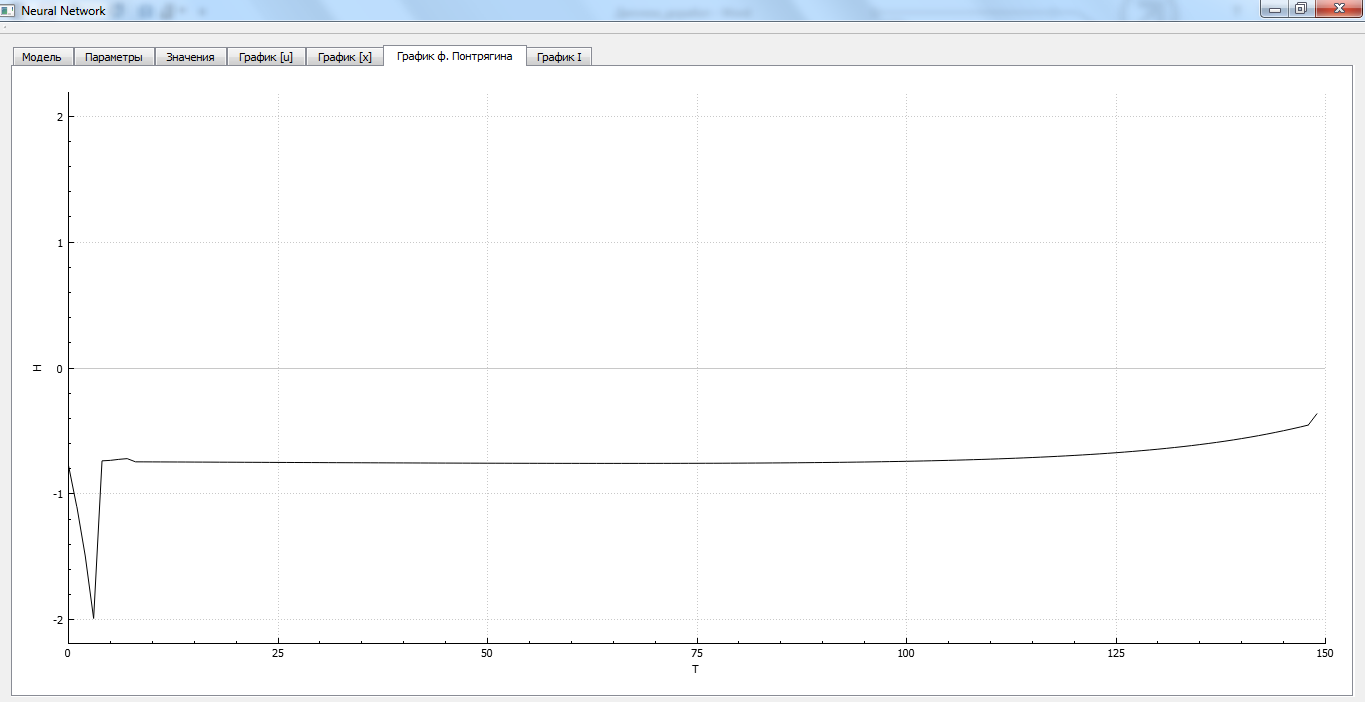


Рисунок 19

Дальнейшее увеличение штрафного коэффициента практически никак не сказывается на процессе вычисления и конечном результате. Для данных параметров достаточно значения 100. Увеличение же точности вычислений приведет к увеличению количества итераций и времени, затраченного на выполнение алгоритма.

На Рисунке 20 предоставлены результаты при точности и штрафном коэффициенте

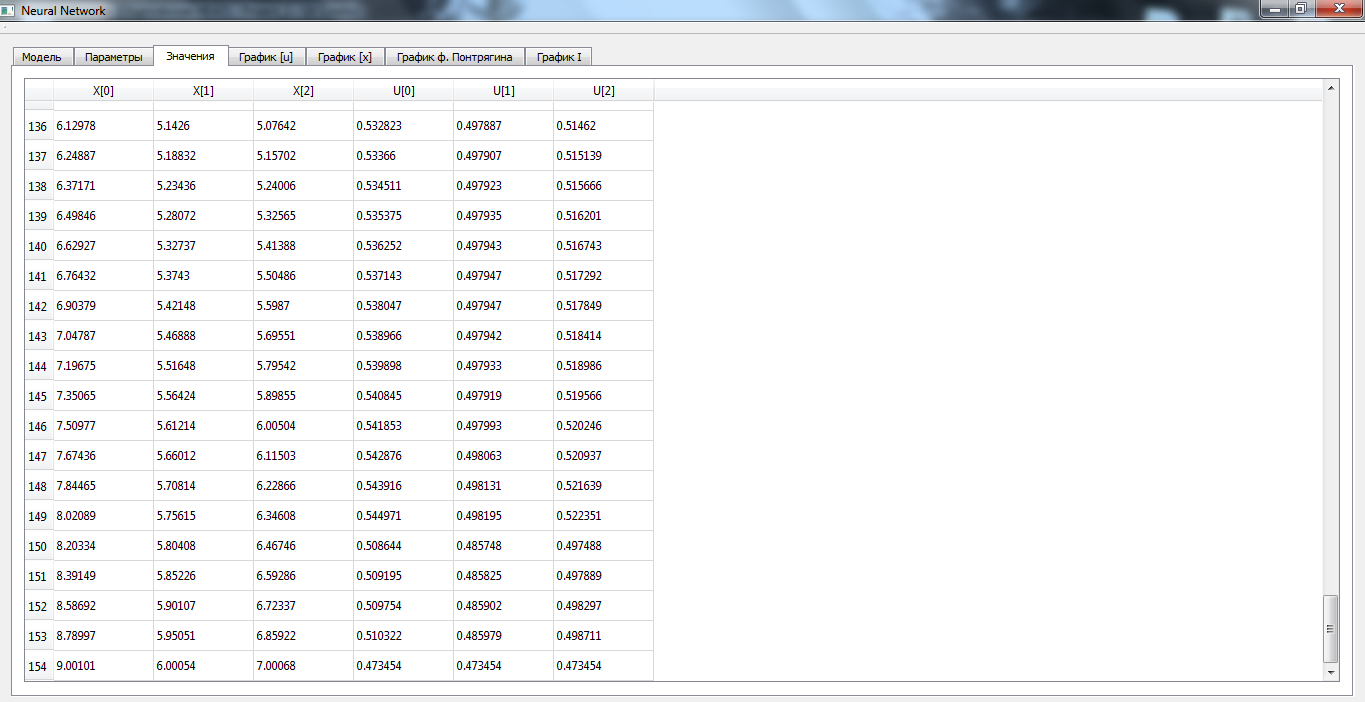


Рисунок 20

Графики функций мало отличаются от полученных ранее, а вот значение целевого функционала, количество итераций и время работы значительно отличаются, как видно из Рисунка 21.

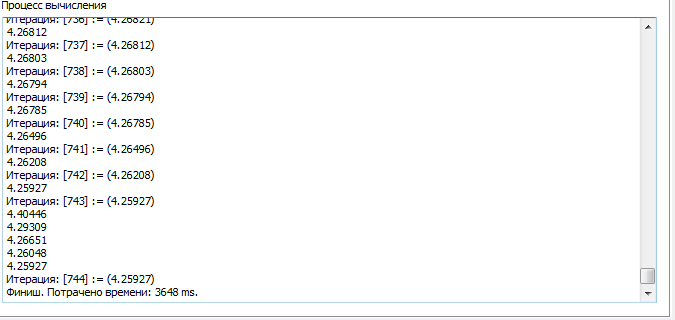


Рисунок 21

Изменение значений времени процесса и количества слоев разбиения также окажут влияние на ход вычислений. При , . Количество итераций увеличится до1785, а время вычислений составит 8270. Конечные значения управлений и состояний системы мало отличаются от аналогичных, из предыдущих экспериментов. Графики функций будут иметь следующий вид:

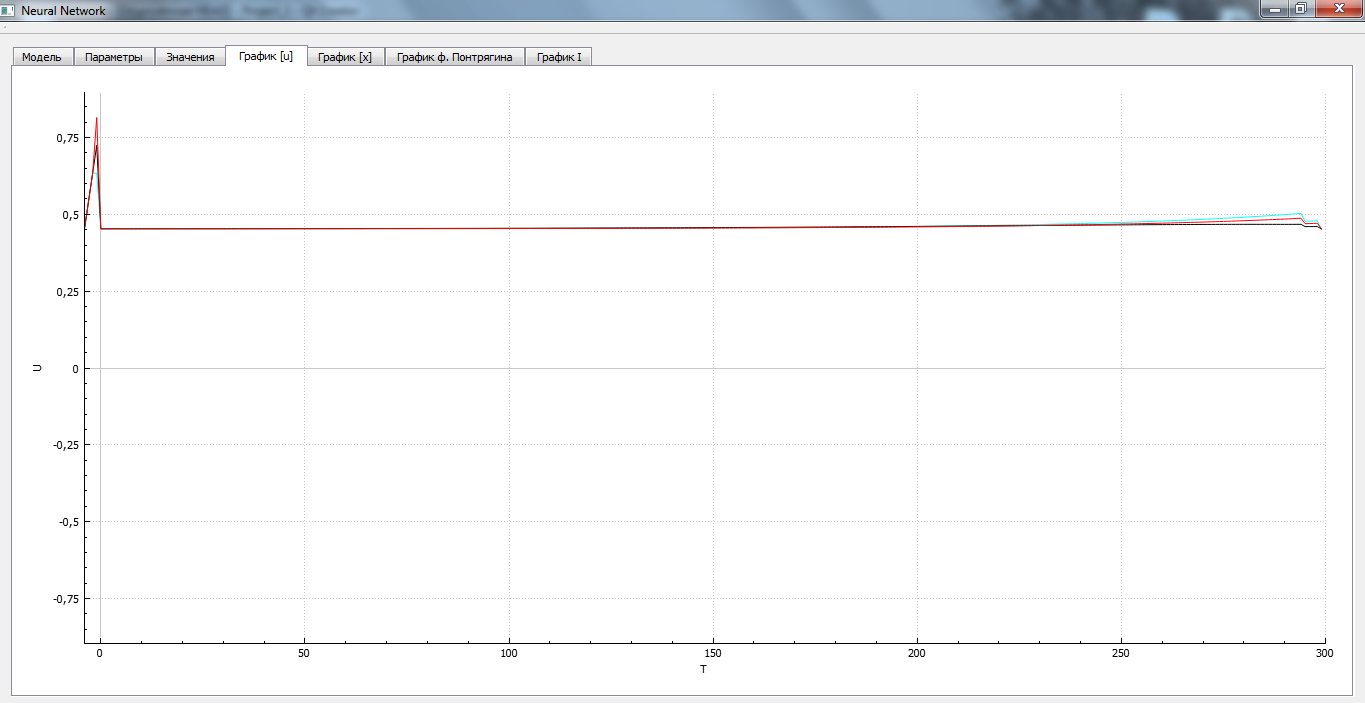


Рисунок 22

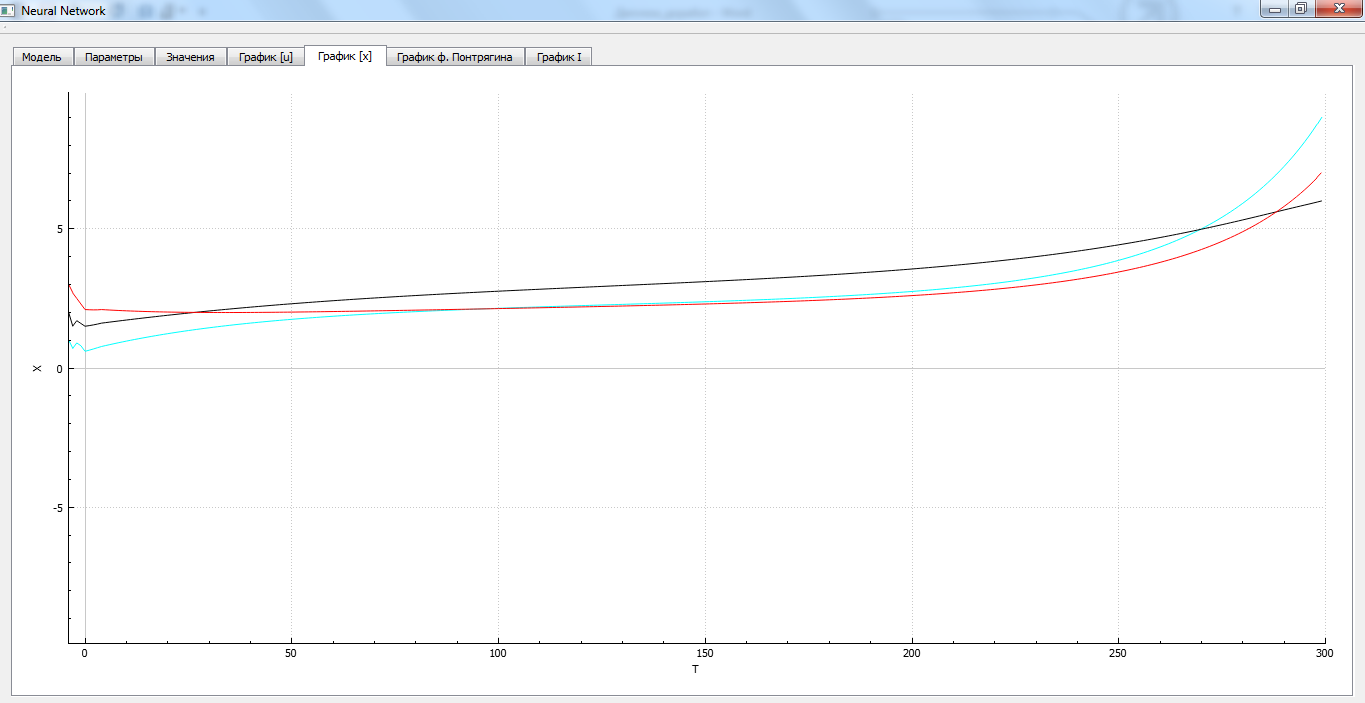


Рисунок 23

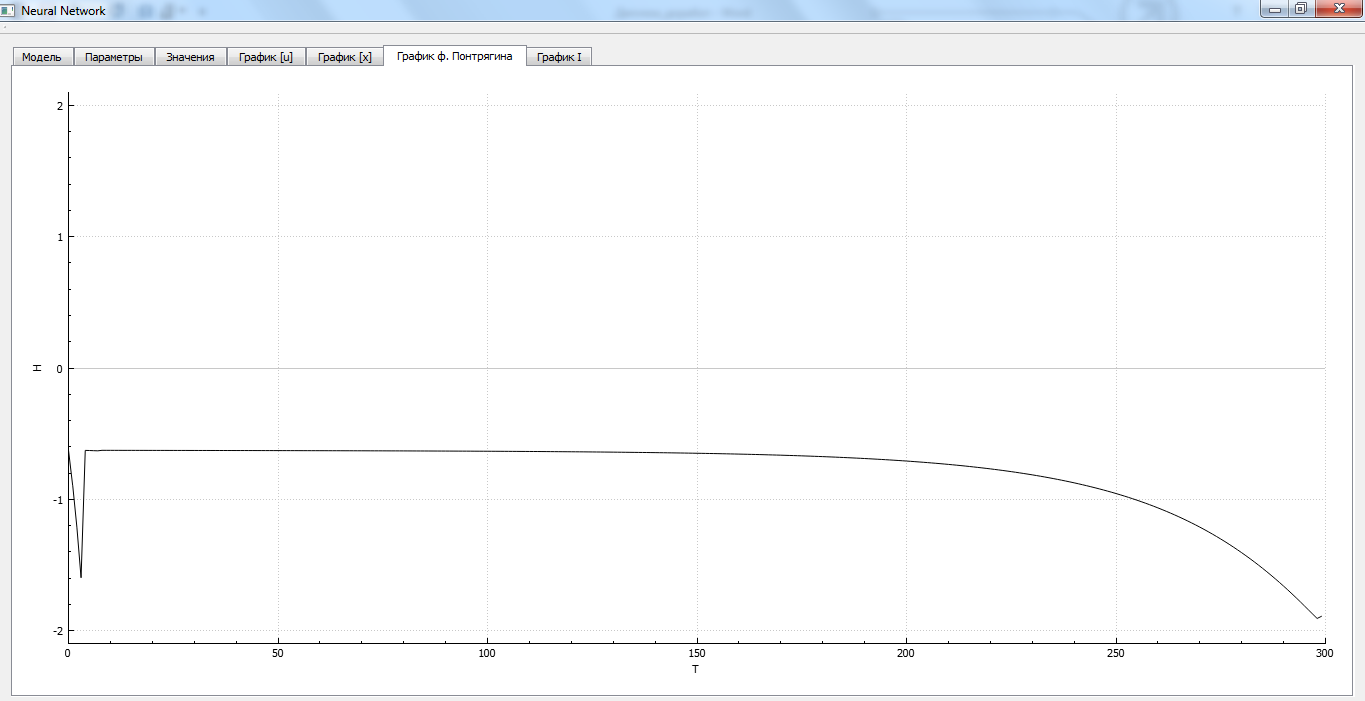


Рисунок 24

График функционала качества не приведен по причине трудной читаемости. Это вызвано большим количеством итераций и большим количеством мало отличающихся друг от друга значений на большом количестве шагов.

В дальнейшем можно рассматривать модели с другими числовыми значениями, исключая из рассмотрения отдельные виды запаздываний, делая их больше или меньше, а также варьируя штрафные коэффициенты, точность вычислений и др. В работе рассмотрен наиболее общий случай с запаздываниями в компонентах состояния и управления.

Отдельно можно рассмотреть динамику сети без последствий. Начальные значения представлены на Рисунке 25.

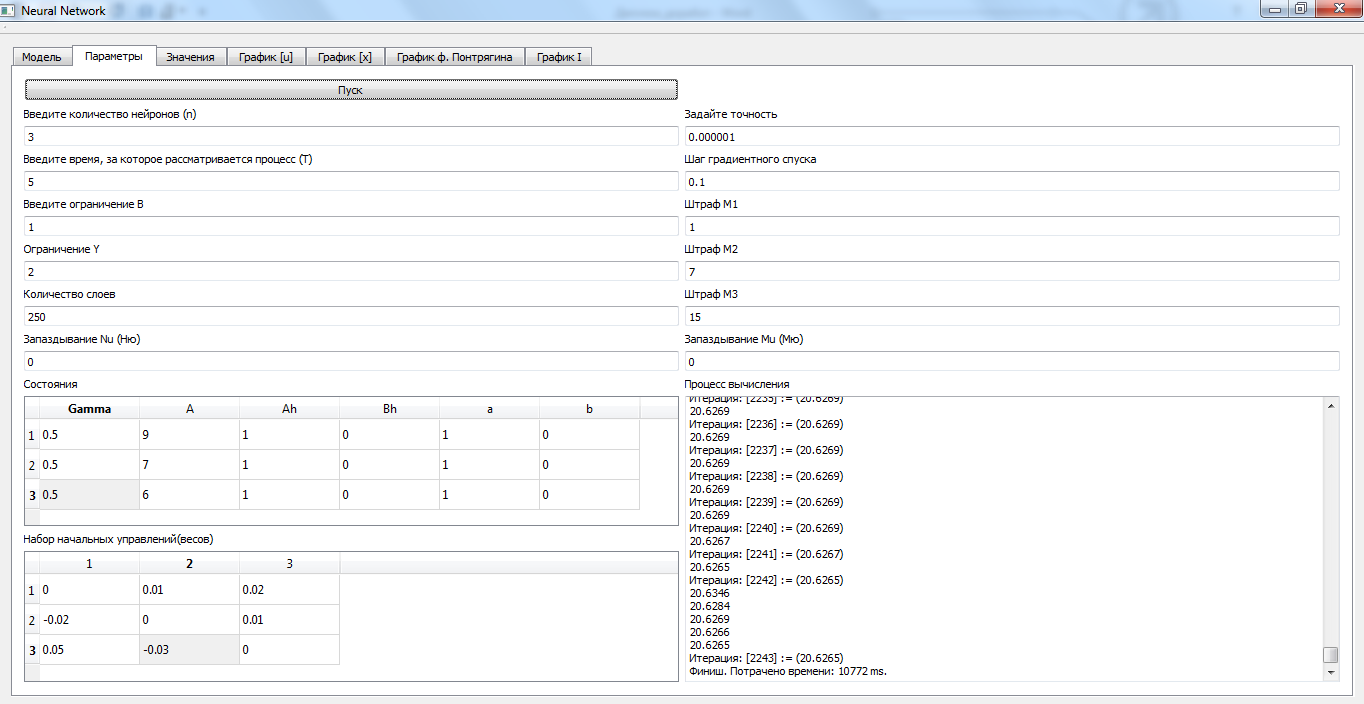


Рисунок 25

Графики функций представлены на Рисунках 26-28.

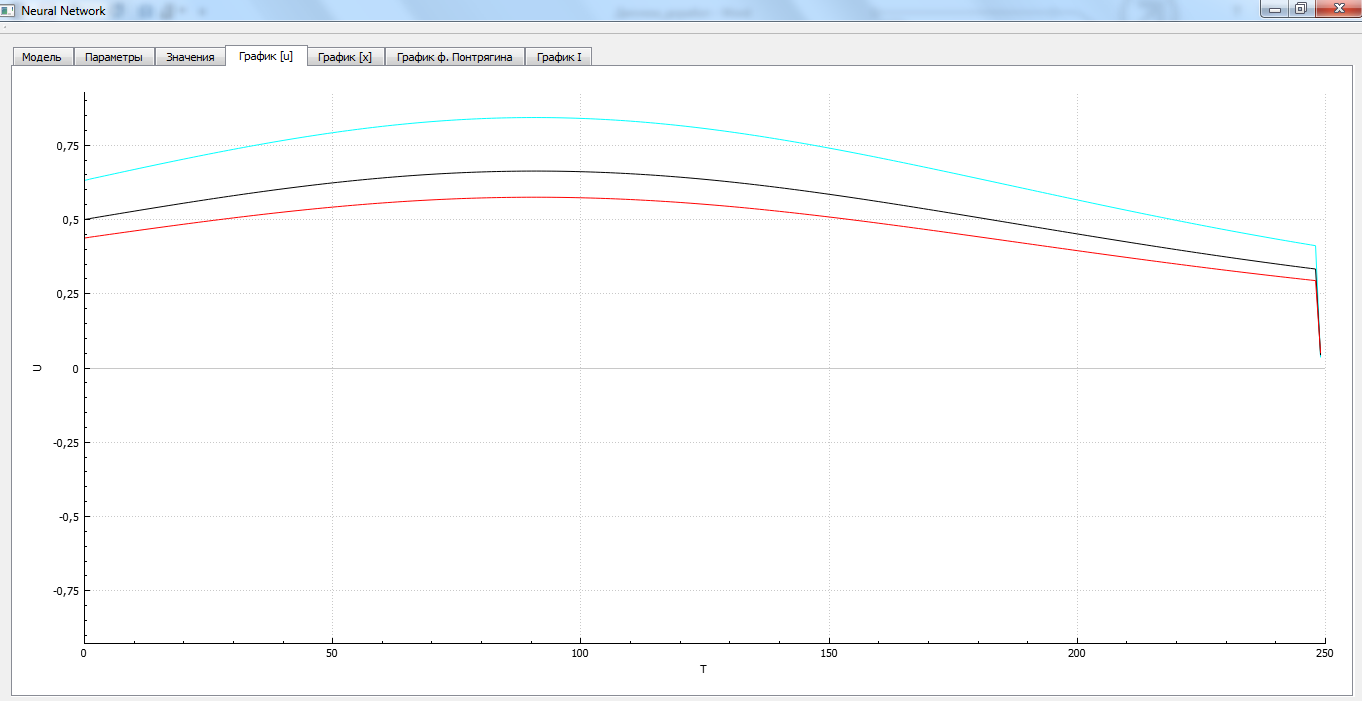


Рисунок 26

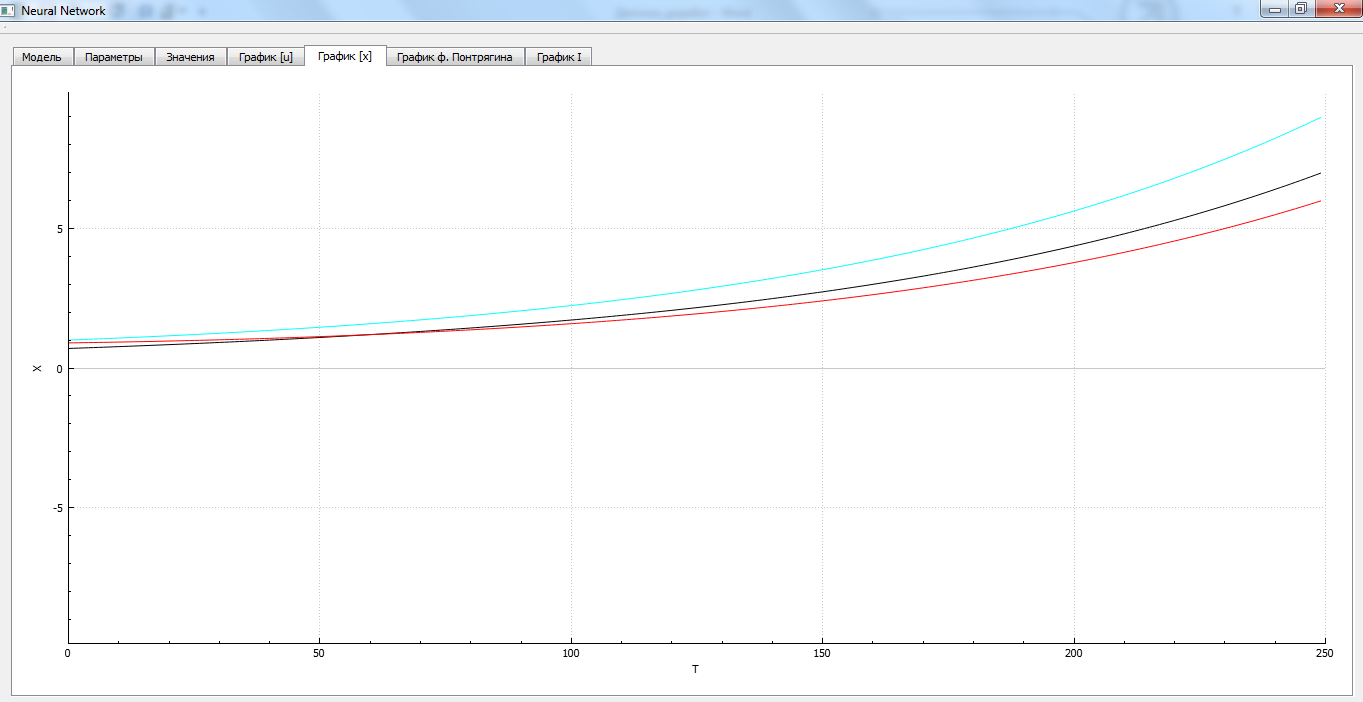


Рисунок 27

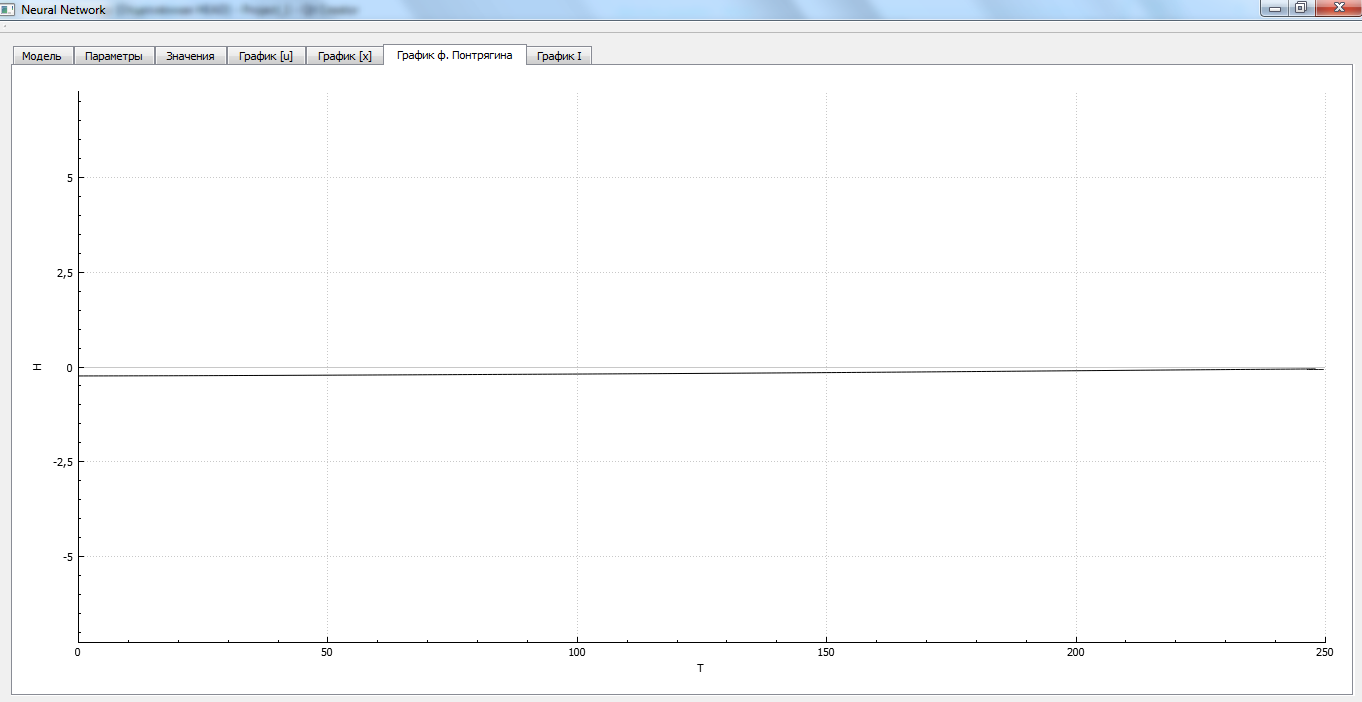


Рисунок 28

График функции Понтрягина для задачи без запаздываний отличается от рассмотренных ранее. Здесь видно, что значения функции близки к нулю.

Дальнейшим улучшением программной реализации алгоритма будет переход к более точным способам выбора шага градиентного спуска, некоторые из которых описаны ранее в работе. Также, функцию, реализующую основные вычисления можно вынести в отдельный поток, так как, на данный момент, запуск вычислений ведет к потере интерактивности интерфейса. Программа условно «замирает» на период вычислений. Данное улучшение больше связано с эстетической составляющей приложения.

Код программы прикреплен в Приложении 1 данной работы.

## Заключение

В данной работе был сделан краткий обзор существующей литературы по вопросам, касающимся теории оптимального. Эта тематика стала очень популярной за последние десятилетия и несомненно, во многом определила дальнейшее развитие человечества. Применение методов, изучаемых в данной теории нашли свое отражение в огромном множестве отраслей деятельности человека. Центральное место в работе было уделено исследованию различных ЗОУ с запаздываниями в компонентах состояния и управления. Были предложены способы решения как непрерывных задач общего вида, так и их линейно-квадратичных вариантов. Затронута тематика решения ЗОУ с запаздываниями с нефиксированным временем процесса, а также продемонстрирована модель системы с переменными запаздываниями. Отдельное внимание уделено решению ДЗОУ, полученных из исходных непрерывных с помощью аппроксимации по предложенной схеме. Для полноты картины, необходимо исследовать решение разрывных ЗОУ с запаздываниями и возможность применения полученного метода для оптимизации ИНС, описываемой СДУ с разрывной правой частью и запаздываниями в аргументах функций состояния и управления. Данная задача сопряжена с некоторыми трудностями и её рассмотрению можно уделить внимание в новой работе.

В качестве альтернативного метода решения ЗОУ был использован метод динамического программирования. На примере линейно-квадратичной ЗОУ без запаздываний была описана процедура синтеза оптимального управления с использованием уравнения Беллмана. Дальнейшим развитием данной темы может стать рассмотрение возможности применения данного метода для решения ЗОУ с запаздываниями, так как на данный момент рассмотрена процедура синтеза управления для задачи с запаздыванием по управлению, на что сделана ссылка в работе.

Большое внимание в работе было уделено изучению искусственных нейронных сетей и нейросетевых методов управления, нашедших широкое применение в теории оптимального управления. Были показаны области и способы применения аппарата ИНС для решения прикладных задач робототехники, информационной безопасности и др. Благодаря гибкости, устойчивости и адаптивности к новым условиям, нейронные сети позволяют с относительной легкостью решать сложные задачи.

Кульминацией работы стала разработка программы, реализующей оптимизацию ИНС по материалам, разработанным в Главе 4. Предоставлены результаты численных экспериментов, показывающих работоспособность приложения и правильную работу алгоритма нахождения оптимального решения. Полученное приложение можно применять при моделировании ИНС, динамика которых описывается соотношениями с последствиями, а также и без них.

## Список литературы

1. Логическое исчисление идей, относящихся к нервной активности. [Электронный ресурс]. - режим доступа: http://www.raai.orgwww.raai.org/library/books/mcculloch/mcculloch.pdf, свободный - Заглавие с экрана. (Дата обращения: 20.10.2017).
2. Хайкин, С. Нейронные сети: полный курс, 2-е издание. [Текст]: Пер. с Англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.
3. Евтушенко, Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации [Текст]: уч. пособ. - М.: Наука, 1982 – 432 с.
4. Кротов, В.Ф. Методы и задачи оптимального управления [Текст]: уч. пособ. / Кротов В.Ф., Гурман В.И. / М.: Наука, 1973 – 448 с.
5. Андреева Е.А. Оптимальное управление динамическими системами [Текст]: учеб. пособие: в 2 ч. – Тверь: Изд-во Тверсоко гос. ун-та, 2016. ч.2 – 188с.
6. Искусственная нейронная сеть. [Электронный ресурс]. - режим доступа:  http://ru.wikipedia.org/?oldid=82538334, свободный - Заглавие с экрана. (Дата обращения: 12.09.2017).
7. Сигеру Омату.  Нейроуправление и его приложения. Neuro-Control and its Applications. [Текст]: монография: 2-е изд. / Сигеру Омату, Марзуки Халид, Рубия Юсоф /— М.: ИПРЖР, 2000. — 272 с.
8. Искусственный нейрон. [Электронный ресурс]. - режим доступа:  http://ru.wikipedia.org/?oldid=79084082, свободный - Заглавие с экрана. (Дата обращения: 12.09.2017).
9. Neural network. [Электронный ресурс]. – режим доступа: http://www.byclb.com/TR/Tutorials/neural\_networks/ch7\_1.htm, свободный – Заглавие с экрана. – (Дата обращения: 11.09.2017).
10. About ImageNet. [Электронный ресурс]. - режим доступа: http://image-net.org/about-overview, свободный - Заглавие с экрана. (Дата обращения: 7.10.2017).
11. Deep Learning. [Электронный ресурс]. - режим доступа: https://en.wikipedia.org/wiki/Deep\_learning, свободный - Заглавие с экрана. (Дата обращения: 7.10.2017).
12. Waymo safety report. On the Road to Fully Self-Driving. [Электронный ресурс]. – режим доступа: https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017.pdf, свободный – Заглавие с экрана. – (Дата обращения: 11.10.2017)
13. Deep Test: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. [Электронный ресурс]. – режим доступа: https://arxiv.org/pdf/1708.08559.pdf, свободный – Заглавие с экрана. – (Дата обращения: 5.10.2017)
14. Робот, управляемый нейронной сетью. [Электронный ресурс]. – режим доступа: http://robocraft.ru/blog/3298.html, свободный – Заглавие с экрана. – (Дата обращения: 28.09.2017)
15. Deep Learning for Robots: Learning from Large-Scale Interaction. [Электронный ресурс]. – режим доступа: https://research.googleblog.com/2016/03/deep-learning-for-robots-learning-from.html, свободный – Заглавие с экрана. – (Дата обращения: 28.09.2017)
16. Системы управления интеллектуальных мобильных роботов в среде Dyn-Soft RobSim 5. [Электронный ресурс]. – режим доступа: http://robsim.dynsoft.ru/design3.pdf, свободный – Заглавие с экрана. – (Дата обращения: 7.10.2017)
17. Динамическое планирование поведения робота на основе сети «интеллектуальных» нейронов. [Электронный ресурс]. – режим доступа: http://www.raai.org/about/persons/karpov/pages/58-69.pdf, свободный – Заглавие с экрана. – (Дата обращения: 7.10.2017)
18. Прохоров, Д.В. Toyota Prius HEV Neurocontrol and Diagnostics. [Текст] / Neural Networks. — 2008. — №. 21. — P. 458—465
19. Стабилизация БПЛА на основе нейросетевого регулятора [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/stabilizatsiya-bespilotnogo-letatelnogo-apparata-na-osnove-neyrosetevogo-regulyatora, свободный – Заглавие с экрана. – (Дата обращения: 11.10.2017).
20. Кушнирук М.С. Исследование алгоритмов управления движением группы спутников с помощью аэродинамической силы сопротивления. [Текст]/ Кушнирук М.С., Иванов Д. С. / Препринты ИПМ им. М.В.Келдыша. -2015. -№ -28. 30 с.
21. Система нейроуправления на основе 3-D сети в условиях робототехнического комплекса специального назначения. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/sistema-neyroupravleniya-na-osnove-3d-seti-v-usloviyah-robototehnicheskogo-kompleksa-spetsialnogo-naznacheniya, свободный – Заглавие с экрана. – (Дата обращения: 11.10.2017)
22. Применение нейронных сетей для обнаружения вторжений. [Электронный ресурс]. – режим доступа: http://dom8a.ru/cupnewyear2014/prezentation/evlanenkova\_paper.pdf, свободный – Заглавие с экрана. – (Дата обращения: 16.10.2017)
23. Сравнительный анализ некоторых алгоритмов роевого интеллекта при обнаружении сетевых атак нейросетевыми методами. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-nekotoryh-algoritmov-roevogo-intellekta-pri-obnaruzhenii-setevyh-atak-neyrosetevymi-metodami, свободный – Заглавие с экрана. – (Дата обращения: 16.10.2017)
24. Нейросетевая диагностика аномальной сетевой активности. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/neyrosetevaya-diagnostika-anomalnoy-setevoy-aktivnosti, свободный – Заглавие с экрана. – (Дата обращения: 18.10.2017)
25. Применение нейронных сетей для мониторинга безопасности информационных систем. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/primenenie-neyronnyh-setey-dlya-monitoringa-bezopasnosti-informatsionnyh-sistem, свободный – Заглавие с экрана. – (Дата обращения: 18.10.2017)
26. Применение ИНС при поиске уязвимостей в исходных текстах ПО. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/primenenie-iskusstvennyh-neyronnyh-setey-pri-poiske-uyazvimostey-v-ishodnyh-tekstah-programmnogo-obespecheniya, свободный – Заглавие с экрана. – (Дата обращения: 18.10.2017)
27. Нейросетевое прогнозирование инцидентов информационной безопасности предприятия. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/neyrosetevoe-prognozirovanie-intsidentov-informatsionnoy-bezopasnosti-predpriyatiya, свободный – Заглавие с экрана. – (Дата обращения: 25.10.2017)
28. Оптимальное управление. [Электронный ресурс]. – режим доступа: http://ru.wikipedia.org/?oldid=, свободный – Заглавие с экрана. – (Дата обращения: 28.09.2017)
29. Задачи оптимального управления. [Электронный ресурс]. – режим доступа: http://www.pereplet.ru/nauka/Soros/pdf/9706\_121.pdf, свободный – Заглавие с экрана. – (Дата обращения: 28.09.2017).
30. Линейно-квадратичный регулятор. [Электронный ресурс]. – режим доступа: http://ru.wikipedia.org/?oldid=80066743, свободный – Заглавие с экрана. – (Дата обращения: 25.10.2017)
31. Замкнутые и разомкнутые системы управления. [Электронный ресурс]. – режим доступа: http://infolike.narod.ru/algoritm7.html#top, свободный – Заглавие с экрана. – (Дата обращения: 25.10.2017).
32. Брайсон А., Прикладная теория оптимального управления. [Текст] / Брайсон А., Хо Ю-Ши / -М.: Мир, -1972г. -544с.
33. Веремей Е.И. Линейные системы с обратной связью. [Текст]: учебное пособие. – СПб.: Издательство «Лань», 2013. – 448 с.
34. Нестационарные линейные системы. [Электронный ресурс]. – режим доступа: http://edu.alnam.ru/book\_v\_tau1.php?id=38, свободный – Заглавие с экрана. – (Дата обращения: 11.11.2017).
35. Математическая теория оптимальных процессов [Текст]: монография / Понтрягин Л.С. [и др.]. - М.: Физматгиз, 1961 – 392 с.
36. Боков, Г.В. Принцип максимума Понтрягина в задаче с временным запаздыванием [Текст] / Г.В. Боков // Фундамент. и прикл. матем.-2009 - № 15 – С. 3–19.
37. Динамическое программирование. [Электронный ресурс]. – режим доступа: http://www.math.mrsu.ru/text/courses/method/dinamicheskoe\_programmirovanie1.htm, свободный – Заглавие с экрана. – (Дата обращения: 10.11.2017).
38. Принцип оптимальности Беллмана и алгоритм решения задач динамического программирования. [Электронный ресурс]. – режим доступа: http://studopedia.ru/2\_23588\_printsip-optimalnosti-bellmana-i-algoritm-resheniya-zadach-dinamicheskogo-programmirovaniya.html, свободный – Заглавие с экрана. – (Дата обращения: 11.11.2017).
39. Принцип оптимальности и уравнения Беллмана. [Электронный ресурс]. – режим доступа: http://finlit.online/metodyi-modelirovanie-matematicheskie/162-printsip-optimalnosti-uravneniya-11155.html, свободный – Заглавие с экрана. – (Дата обращения: 11.11.2017).
40. Робастная стабилизация многомерного линейного объекта с запаздываниями по управлениям. [Электронный ресурс]. – режим доступа: https://cyberleninka.ru/article/n/optimalnyy-sintez-dlya-dinamicheskih-sistem-s-zapazdyvaniem-po-upravleniyu, свободный – Заглавие с экрана. – (Дата обращения: 5.11.2017)
41. Андреева, Е.А. Вариационное исчисление и методы оптимизации. [Текст]: учебн. пособие. / Андреева Е.А., Цирулёва В.М. / – Тверь: Твер. гос. ун-т, -2001. – 576 с.
42. Modelling and control of quadcopter. [Электронный ресурс]. – режим доступа: http://sal.aalto.fi/publications/pdf-files/eluu11\_public.pdf, свободный – Заглавие с экрана. – (Дата обращения: 27.10.2017)
43. Колотов, М. Е. Синтез LQR-регуляторов для управления квадрокоптером и их сравнительный анализ на основе имитационного моделирования при помощи пакета прикладных программ MATLAB & Simulink. [Текст] / Колотов М. Е., Смирнова Т. А. / Молодой ученый. — 2016. — №13. — С. 172-177.
44. Обучение нейронной сети с запаздыванием. [Электронный ресурс]. – режим доступа: http://www.swsys.ru/index.php?page=article&id=2757, свободный – Заглавие с экрана. – (Дата обращения: 15.11.2017)
45. Беллман Р. Динамическое программирование. [Текст]: монография. –М.: Изд-во иностранной литературы, -1960. – 400 с.

### Приложение 1. Листинг программы

#include "mainwindow.h"

#include "time.h" // чтобы измерять время

#include "ui\_mainwindow.h"

#include "qmath.h" // чтобы использовать некторые мат. функции

/\* нужно для многопоточности, на будущее \*/

#include "QtConcurrent/qtconcurrentrun.h"

#include "QThread"

#include <algorithm> // чтобы можно было искать минимум и максимум в векторе.

#include <fstream>

#define COLOR\_ARRAY\_SIZE (14)

int launch = 0; // количество запусков задачи (нажатие кнопки "Делать")

int n; // neur\_amount

int q; // layout amount

double dt; // dt = T/q

double alpha; // depth of gradient fall

double epsilon; // accurasy of calculating

double B; // ограничения на весовые коэффициенты

double Y; // ограничение на управление U

/\* Переменные для запаздываний. \*/

int mu;

int nu;

int max\_mu\_nu;

/\*Функция для улучшения весов\*/

int calc\_weights(double \*\*\*w\_next, double \*Alpha, double \*Bheta, double \*\*\*w\_previous, double \*\*p, double \*\*X, double M2, double B){

for (int k = 0; k < nu+1; k++)

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++){

// пересчитываем веса

w\_next[k][i][j] = w\_previous[k][i][j] - alpha\*(2\*M2\*dt\*w\_previous[k][i][j] - Alpha[i]\*dt\*p[k+1][i]\*X[k][j]); //p[k+1] должно быть

// сравниваем с ограничениям сверху и снизу

if (w\_next[k][i][j] > B)

w\_next[k][i][j] = B;

if (w\_next[k][i][j] < -B)

w\_next[k][i][j] = -B;

}

for (int k = nu; k < q+nu; k++)

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++){

// пересчитываем веса

w\_next[k][i][j] = w\_previous[k][i][j] - alpha\*(2\*M2\*dt\*w\_previous[k][i][j] - Alpha[i]\*dt\*p[k+1][i]\*X[k][j]-Bheta[i]\*dt\*p[k+1][i]\*X[k-nu][j]); //p[k+1] должно быть

// сравниваем с ограничениям сверху и снизу

if (w\_next[k][i][j] > B)

w\_next[k][i][j] = B;

if (w\_next[k][i][j] < -B)

w\_next[k][i][j] = -B;

}

return 0;

}

/\* Функция для улучшения управлений \*/

int calc\_U(double \*\*U\_next, double \*\*U\_prev, double \*a, double \*b, double \*\*p, double Y, double M1){

for(int k = 0; k < q+mu; k++)

for(int i = 0; i < n; i++){

// улучшаем управления

U\_next[k][i] = U\_prev[k][i] - alpha\*(M1\*2\*U\_prev[k][i]\*dt - dt\*a[i]\*p[k+1][i]-dt\*b[i]\*p[k+mu+1][i]);

// сравниваем с ограничениями

if (U\_next[k][i] > Y)

U\_next[k][i] = Y;

if (U\_next[k][i] < -Y)

U\_next[k][i] = -Y;

}

return 0;

}

/\* Функция для подсчета значений сопряженных переменных P \*/

int calc\_p(double \*\*X, double \*Alpha, double \*Bheta, double \*gamma, double \*A, double \*\*\*weight, double \*\*p, double M3){

double sum; // переменная для промежуточных расчетов

for (int k = q+3\*(nu+1)-1; k >= 0; k--)

for (int i = 0; i < n; i++){

p[k][i] = 0;

}

// считаем P на последнем слое

for (int i = 0; i < n; i++)

p[q+nu-1][i] = -2\*M3\*(X[q+nu-1][i]-A[i]);

for (int k = q+nu-2; k >= nu; k--)

for (int i = 0; i < n; i++){

// обнуляем буфер

sum = 0;

for (int j = 0; j < n; j++){

// считаем "добавку" на этом шаге

sum += p[k+1][j]\*weight[k][i][j]\*Alpha[j] + p[k+1+nu][j]\*weight[k+nu][i][j]\*Bheta[j]; // j - это строка должна быть в матрице весов или столбец

}

sum \*= dt;

p[k][i] = p[k+1][i] - dt\*gamma[i]\*p[k+1][i] + sum;

}

return 0;

}

/\* Функция для подсчета значения целевой функции \*/

double calc\_I(double \*\*\*weight, double \*X\_q, double \*A, double M1, double M2, double M3, double \*\*U){

double I = 0; // переменная для хранения значений целевой функции

double sum = 0; // буфер для промежуточных вычислений

for (int k = 0; k < q+mu; k++)

for(int i = 0; i < n; i++)

I += U[k][i]\*U[k][i];

I \*= M1\*dt;

for (int k = 0; k < q+nu; k++)

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

sum += weight[k][i][j]\*weight[k][i][j];

sum \*= M2\*dt;

I += sum;

sum = 0;

for (int i = 0; i < n; i++)

sum += (X\_q[i] - A[i])\*(X\_q[i] - A[i]);

sum \*= M3;

I += sum;

return I;

}

/\* Функция для подсчета значений траектории системы (X)

Alpha - коэффициенты при x без запазд.

Bheta - коэффициенты при x с запазд.

a - коэффициенты при u без запазд.

b - коэффициенты при u с запазд.

\*/

int calc\_X(double \*gamma, double \*Alpha, double \*Bheta, double \*a, double \*b, double \*\*\*weight, double \*\*X\_0, double \*\*X, double \*\*U){

int k, i, j; // переменные для циклов

double Sum\_x; // буфер для промежуточных расчетов компонент x

double Sum\_u; // буфер для промежуточных расчетов компонент u

double argument; // аргумент для функции активации

/\* Начальные условия. \*/

for (k = 0; k < nu+1; k++)

for (i = 0; i < n; i++)

X[k][i] = X\_0[k][i];

/\* \*/

for (k = nu; k < q+nu-1; k++)

for (i = 0; i < n; i++){

Sum\_x = 0;

Sum\_u = 0;

for (j = 0; j < n; j++){

/\* Считаем сумму с x \*/

argument = 0;

argument = Alpha[i]\*X[k][j] + Bheta[i]\*X[k-nu][j];

Sum\_x += weight[k][i][j]\*argument;

}

/\* Считаем сумму с u \*/

argument = 0;

argument = a[i]\*U[k][i] + b[i]\*U[k-mu][i];

Sum\_u += argument;

X[k+1][i] = X[k][i] + dt\*(-gamma[i]\*X[k][i] + Sum\_x + Sum\_u);

}

return 0;

}

/\* Функция для подсчета значений функции Понтрягина \*/

int calc\_H(double \*H, double \*Alpha, double \*Bheta, double \*a, double \*b, double \*\*X, double \*\*U, double \*\*\*weight, double \*\*p, double \*gamma, double M1, double M2, double M3){

double Sum, sum;

for(int k = 0; k < q; k++){

H[k] = 0;

sum = 0;

for(int i = 0; i < n; i++)

sum += U[k][i]\*U[k][i];

sum \*= M1;

H[k] -=sum;

sum = 0;

for(int i = 0; i < n; i++)

for(int j = 0; j < n; j++)

sum += weight[k][i][j]\*weight[k][i][j];

sum \*= M2;

H[k] -= sum;

Sum = 0;

for(int i = 0; i < n; i++){

sum = 0;

for(int j = 0; j < n; j++){

sum += Alpha[i]\*weight[k][i][j]\*X[k][j];

if (k >= nu)

sum += Bheta[i]\*weight[k][i][j]\*X[k-nu][j];

}

sum += a[i]\*U[k][i];

if (k >= mu)

sum += b[i]\*U[k-mu][i];

sum -= X[k][i]\*gamma[i];

sum \*= p[k][i];

Sum += sum;

}

H[k] += Sum;

}

return 0;

}

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

// QPixmap myPixmap( "C:/QT\_Projects/Project\_1/giphy\_s.gif" );

QPixmap myPixmap( "C:/QT\_Projects/Project\_2/Project\_1/model\_image.gif" );

ui->label->setPixmap(myPixmap);

ui->tab->adjustSize();

ui->tab\_2->adjustSize();

ui->label\_2->setText("Введите количество нейронов (n)");

ui->label\_2->adjustSize();

ui->label\_3->setText("Введите время, за которое рассматривается процесс (T)");

ui->label\_3->adjustSize();

ui->label\_4->setText("Введите ограничение B");

ui->label\_4->adjustSize();

ui->label\_5->setText("Задайте точность");

ui->label\_5->adjustSize();

ui->label\_7->setText("Шаг градиентного спуска");

ui->label\_7->adjustSize();

ui->label\_8->setText("Набор начальных управлений(весов)");

ui->label\_8->adjustSize();

ui->label\_10->setText("Штраф M1");

ui->label\_10->adjustSize();

ui->label\_11->setText("Штраф M2");

ui->label\_11->adjustSize();

ui->label\_12->setText("Количество слоев");

ui->label\_12->adjustSize();

ui->label\_13->setText("Штраф M3");

ui->label\_13->adjustSize();

ui->pushButton->setText("Пуск");

ui->lineEdit->setText("3"); // neuron amount

ui->lineEdit\_2->setText("5"); // time T

ui->lineEdit\_9->setText("150"); // amount of layers

ui->lineEdit\_3->setText("1"); // Ограничение B

ui->lineEdit\_4->setText("0.01"); // accuracy

ui->lineEdit\_6->setText("0.1"); // depth of gradient fall

ui->lineEdit\_7->setText("1"); // M1

ui->lineEdit\_8->setText("1"); // M2

ui->lineEdit\_5->setText("1"); // M3

ui->lineEdit\_10->setText("2"); // ограничение Y

ui->lineEdit\_11->setText("4"); // запаздывание mu

ui->lineEdit\_12->setText("4"); // запаздывание nu

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

launch++; // прибавляем количество запусков

srand((unsigned) time(NULL));

QTime t;

t.start();

int Neur\_amount;

double M1, M2, M3;

M1 = ui->lineEdit\_7->text().toDouble();

M2 = ui->lineEdit\_8->text().toDouble();

M3 = ui->lineEdit\_5->text().toDouble();

Neur\_amount = ui->lineEdit->text().toInt();

// Ввод параметров функции активации

nu = ui->lineEdit\_11->text().toDouble(); // МЮ (MU, mu)

mu = ui->lineEdit\_12->text().toDouble(); // НЮ (NU, nu)

if (nu < mu)

max\_mu\_nu = mu;

else

max\_mu\_nu = nu;

n = Neur\_amount;

q = ui->lineEdit\_9->text().toInt();

// int T = ui->lineEdit\_2->text().toInt();

double T = ui->lineEdit\_2->text().toDouble(); //если T integer, то берется целая часть от деления. Если double, то выполняется деление с остатком.

dt = T/q;

// ui->lineEdit->setText(QString::number(dt)); //DEBUG

double \*\*\*w\_old = new double \*\*[q+3\*(nu+1)];

double \*\*\*w\_new = new double \*\*[q+3\*(nu+1)];

for (int j = 0; j < q+3\*(nu+1); j++){

w\_new[j] = new double \*[Neur\_amount];

w\_old[j] = new double \*[Neur\_amount];

for (int i = 0; i < Neur\_amount; i++){

w\_new[j][i] = new double [Neur\_amount];

w\_old[j][i] = new double [Neur\_amount];

}

}

/\* начальные \*/

for (int i = 0; i < Neur\_amount; i++)

for(int j = 0; j < Neur\_amount; j++)

w\_old[0][i][j] = ui->tableWidget->item(i, j)->text().toDouble();

// заполняем веса (делаем такими же, как на в нулевой момент)

for (int k = 1; k < q+3\*nu; k++)

for (int i = 0; i < Neur\_amount; i++)

for (int j = 0; j < Neur\_amount; j++)

w\_old[k][i][j] = w\_old[k-1][i][j];

// объявляем массивы

double \*gamma = new double[Neur\_amount]; // гамма (затухания)

double \*A = new double[Neur\_amount]; // конечная точка

double \*Alpha = new double[Neur\_amount];

double \*Bheta = new double[Neur\_amount];

double \*a = new double[Neur\_amount];

double \*b = new double[Neur\_amount];

double \*\*X\_0 = new double \*[nu+1]; // начальные иксы

double \*\*X = new double \*[q+nu]; // иксы

double \*\*X\_new = new double \*[q+nu]; // новые иксы для пересчета целевой функции

double \*\*p = new double \*[q+3\*(nu+1)]; // коэффициенты P

double \*\*U\_old = new double \*[q+max\_mu\_nu]; // управления U

double \*\*U\_new = new double \*[q+max\_mu\_nu];

std::ifstream init\_x("C:/QT\_projects/Project\_2/Project\_1/init\_x.txt");

std::ifstream init\_u("C:/QT\_projects/Project\_2/Project\_1/init\_u.txt");

for (int i = 0; i < nu+1; i++)

X\_0[i] = new double[Neur\_amount];

for (int i = 0; i < q+nu; i++){

X[i] = new double[Neur\_amount];

X\_new[i] = new double[Neur\_amount];

}

for (int i = 0; i < q+3\*(nu+1); i++)

p[i] = new double[Neur\_amount];

for (int i = 0; i < q+max\_mu\_nu; i++){

U\_old[i] = new double[Neur\_amount];

U\_new[i] = new double[Neur\_amount];

}

// заполнение начальных значений, коэф затухания, конечные величины и управления U

for (int i = 0; i < Neur\_amount; i++){

gamma[i] = ui->tableWidget\_2->item(i,0)->text().toDouble(); // затухания

A[i] = ui->tableWidget\_2->item(i,1)->text().toDouble(); // концы для иксов

Alpha[i] = ui->tableWidget\_2->item(i,2)->text().toDouble(); // альфы

Bheta[i] = ui->tableWidget\_2->item(i,3)->text().toDouble(); // беты

a[i] = ui->tableWidget\_2->item(i,4)->text().toDouble(); // а малые

b[i] = ui->tableWidget\_2->item(i,5)->text().toDouble(); // б малые

}

for (int i = 0; i < Neur\_amount; i++)

for (int k = 0; k < (nu + 1); k ++)

init\_x >> X\_0[k][i];

for (int i = 0; i < Neur\_amount; i++)

for (int k = 0; k < (max\_mu\_nu + 1); k++)

init\_u >> U\_old[k][i];

// добиваем оставшиеся слои управления начальными значениями, потому что улучшать пока нечем

for(int k = max\_mu\_nu+1; k < q+max\_mu\_nu; k++)

for(int i = 0; i < Neur\_amount; i++)

U\_old[k][i] = U\_old[k-1][i];

// считаем X для нулевой итерации

calc\_X(gamma, Alpha, Bheta, a, b, w\_old, X\_0, X, U\_old);

double I\_new = 0, I\_old = 0; // переменные для хранения значений целевой функции

epsilon = ui->lineEdit\_4->text().toDouble(); // считали точность

alpha = ui->lineEdit\_6->text().toDouble(); // считали шаг градиентного спуска (сейчас он один для весов и для управлений) // TODO: СДелать два разных шага спуска.

B = ui->lineEdit\_3->text().toDouble(); // считали ограничение на веса W

Y = ui->lineEdit\_10->text().toDouble(); // считали ограничение на управление U

double dI = 0; // переменная для хранения разности значений целевой функции на разных итерациях

int iter = 0; // переменная для хранения номера итерации

QVector<double> I\_y\_graph; // Значения целевой функции для графика

// считаем значение целевой функции на нулевой итерации

I\_old = calc\_I(w\_old, X[q+nu-1], A, M1, M2, M3, U\_old);

I\_y\_graph.push\_back(I\_old);

// ui->lineEdit->setText(QString::number(I\_old));

// выводим значение целевой функции и номер итерации

ui->textBrowser->append("|\*\*\*\*\*\*\*\* Запуск [" + QString::number(launch) + "] \*\*\*\*\*\*\*\*|");

ui->textBrowser->append("Итерация: [" + QString::number(iter) + "] := (" + QString::number(I\_old) + ")");

// основной цикл для оптимизации нашего критерия

do{

if(iter % 20 == 0 && iter != 0) // если было сделано 20 итераций, то увеличиваем шаг спуска (должно работать быстрее благодаря этому)

alpha = alpha\*pow(2,5);

// считаем значения сопряженных переменных

calc\_p(X, Alpha, Bheta, gamma, A, w\_old, p, M3);

do{

calc\_weights(w\_new, Alpha, Bheta, w\_old, p, X, M2, B); // пересчитываем веса

calc\_U(U\_new, U\_old, a, b, p, Y, M1); // пересчитываем управления

calc\_X(gamma, Alpha, Bheta, a, b, w\_new, X\_0, X\_new, U\_new); // пересчитываем X

I\_new = calc\_I(w\_new, X\_new[q+nu-1], A, M1, M2, M3, U\_new); // пересчитываем значение целевой функции

ui->textBrowser->append(QString::number(I\_new)); // выводим получившееся значение целевой функции

// корректируем шаг градиентного спуска, закомментированные условия нужны для учета доп. критериев.

if (I\_new > I\_old/\* || X\_new[q-1][0] > A[0] || X\_new[q-1][1] > A[1] || X\_new[q-1][2] > A[2]\*/){

alpha = alpha / 2;

}

}while(I\_new > I\_old && fabs(I\_new - I\_old) > epsilon/\*&& ((I\_new - I\_old < 0) && (-(I\_new - I\_old) > epsilon)) || ((I\_new - I\_old > 0) && (I\_new - I\_old > epsilon))\*/);

dI = I\_old - I\_new;

if (dI > -epsilon || dI < epsilon){

I\_old = I\_new;

I\_y\_graph.push\_back(I\_old);

iter++;

ui->textBrowser->append("Итерация: [" + QString::number(iter) + "] := (" + QString::number(I\_old) + ")");

for (int k = 0; k < q+nu; k++)

for (int i = 0; i < Neur\_amount; i++)

for (int j = 0; j < Neur\_amount; j++)

w\_old[k][i][j] = w\_new[k][i][j];

for(int k = 0; k < q+mu; k++)

for(int i = 0; i < Neur\_amount; i++){

X[k][i] = X\_new[k][i];

U\_old[k][i] = U\_new[k][i];

}

}

// else

// break;

}while(dI < -epsilon || dI > epsilon);

ui->textBrowser->append("Финиш. Потрачено времени: " + QString::number(t.elapsed()) + " ms.");

// вывод X, U, посчитанных на конечной итерации

ui->tableWidget\_3->adjustSize();

ui->tableWidget\_3->setRowCount(q+max\_mu\_nu);

ui->tableWidget\_3->setColumnCount(2\*Neur\_amount);

/\* сформируем подписи к таблице со значениями X и U \*/

char header[20]; // переменная под заголовок столбца

QStringList horzHeaders; // массик с заголовками

// заполняем заголовки

for (int i = 0; i < Neur\_amount; i++){

sprintf(header, "X[%d]", i);

horzHeaders << header;

}

for (int i = 0; i < Neur\_amount; i++){

sprintf(header, "U[%d]", i);

horzHeaders << header;

}

// назначаем заголовки нашей таблице

ui->tableWidget\_3->setHorizontalHeaderLabels(horzHeaders);

// вывод значений X и U в таблицу

for (int k = 0; k < q+nu; k++){

for (int i = 0; i < Neur\_amount; i++){

QTableWidgetItem \*tmp1 = new QTableWidgetItem;

tmp1->setText(QString::number(X\_new[k][i], 'g', 6));

ui->tableWidget\_3->setItem(k, i, tmp1);

}

}

for (int k = 0; k < q+mu; k++){

for (int i = 0; i < Neur\_amount; i++){

QTableWidgetItem \*tmp2 = new QTableWidgetItem;

tmp2->setText(QString::number(U\_new[k][i], 'g', 6));

ui->tableWidget\_3->setItem(k, i+Neur\_amount, tmp2);

}

}

// QVector<double> X\_graph(q), Y\_graph(q);

QVector<QVector<double> > X\_graph(Neur\_amount);

QVector<QVector<double> > Y\_graph(Neur\_amount);

double max\_Y\_old = 10, max\_Y\_new = 10;

for(int i = 0; i < Neur\_amount; i++)

for (int j = -nu; j < q; j++){

X\_graph[i].push\_back(j);

}

for(int i = 0; i < Neur\_amount; i++)

for (int j = 0; j < q+nu; j++){

Y\_graph[i].push\_back(X\_new[j][i]);

}

ui->widget->clearGraphs();

// int color[5] = {Qt::black, Qt::cyan, Qt::red, Qt::blue, Qt::green};

/\* Создадим массив цветов, чтобы линии рисовались разными цветами \*/

QColor color[COLOR\_ARRAY\_SIZE] = {Qt::cyan, Qt::black , Qt::red, Qt::blue, Qt::green, Qt::magenta, Qt::yellow, Qt::gray, Qt::darkBlue, Qt::darkCyan, Qt::darkGray, Qt::darkGreen, Qt::darkMagenta, Qt::darkRed};

for(int i = 0; i < Neur\_amount; i++){

/\* добавляем линию на полотно \*/

ui->widget->addGraph();

/\* назначаем цвет для линии, а так как их количество у меня ограничено, то берем остаток от деления (чтобы повторялись, но хоть какое-то разнообразие было) \*/

ui->widget->graph(i)->setPen(QPen(color[i%COLOR\_ARRAY\_SIZE]));

ui->widget->graph(i)->setData(X\_graph[i], Y\_graph[i]);

}

ui->widget->replot();

/\* Находим максимально значение по оси Y, которое надо будет отобразить, это надо для выбора границ отображения

Так как у нас вектор векторов, то надо искать максимум в каждом, а потом среди них выбрать максимум.

Я это делаю последовательно. Посчитал максимум, сравнил с предыдущим и так далее.

\*/

max\_Y\_old = fabs(\*std::max\_element(Y\_graph[0].begin(),Y\_graph[0].end()));

for(int i = 1; i < Neur\_amount; i++){

max\_Y\_new = fabs(\*std::max\_element(Y\_graph[i].begin(),Y\_graph[i].end()));

if (max\_Y\_new > max\_Y\_old)

max\_Y\_old = max\_Y\_new;

}

ui->widget->yAxis->setRange(-max\_Y\_old - 0.1\*max\_Y\_old, max\_Y\_old + 0.1\*max\_Y\_old);

ui->widget->xAxis->setRange(-nu, q);

ui->widget->replot(); // FIXME: там выше есть уже реплот

/\* Графикки функций управления (U). \*/

QVector<QVector<double> > U\_x\_graph(Neur\_amount);

QVector<QVector<double> > U\_y\_graph(Neur\_amount);

for(int i = 0; i < Neur\_amount; i++)

for (int j = -mu; j < q; j++){

U\_x\_graph[i].push\_back(j);

}

for(int i = 0; i < Neur\_amount; i++)

for (int j = 0; j < q+mu; j++){

U\_y\_graph[i].push\_back(U\_new[j][i]);

}

ui->widget\_2->clearGraphs();

for(int i = 0; i < Neur\_amount; i++){

ui->widget\_2->addGraph();

ui->widget\_2->graph(i)->setPen(QPen(color[i%COLOR\_ARRAY\_SIZE]));

ui->widget\_2->graph(i)->setData(U\_x\_graph[i], U\_y\_graph[i]);

}

/\* Подгоняем размер осей. \*/

max\_Y\_old = fabs(\*std::max\_element(U\_y\_graph[0].begin(),U\_y\_graph[0].end()));

for(int i = 1; i < Neur\_amount; i++){

max\_Y\_new = fabs(\*std::max\_element(U\_y\_graph[i].begin(),U\_y\_graph[i].end()));

if (max\_Y\_new > max\_Y\_old)

max\_Y\_old = max\_Y\_new;

}

ui->widget\_2->yAxis->setRange(-max\_Y\_old - 0.1\*max\_Y\_old, max\_Y\_old + 0.1\*max\_Y\_old);

ui->widget\_2->xAxis->setRange(-mu, q);

ui->widget\_2->replot();

/\* Построение графика функции Понтрягина \*/

double \*H = new double[q];

calc\_H(H, Alpha, Bheta, a, b, X\_new, U\_new, w\_new, p, gamma, M1, M2, M3);

ui->widget\_3->clearGraphs();

QVector<double> H\_x\_graph;

QVector<double> H\_y\_graph;

for (int j = 0; j < q; j++){

H\_x\_graph.push\_back(j);

H\_y\_graph.push\_back(H[j]);

}

ui->widget\_3->addGraph();

ui->widget\_3->graph(0)->setPen(QPen(color[1]));

ui->widget\_3->graph(0)->setData(H\_x\_graph, H\_y\_graph);

for (int i = 0; i < q; i++)

H\_y\_graph[i] = fabs(H\_y\_graph[i]);

max\_Y\_old = fabs(\*std::max\_element(H\_y\_graph.begin(), H\_y\_graph.end()));

ui->widget\_3->yAxis->setRange(-max\_Y\_old - 0.1\*max\_Y\_old, max\_Y\_old + 0.1\*max\_Y\_old);

ui->widget\_3->xAxis->setRange(0, q);

ui->widget\_3->replot();

/\* Построение графика целевой функции. \*/

ui->widget\_4->clearGraphs();

QVector<double> I\_x\_graph;

for (int j = 0; j < I\_y\_graph.size(); j++){

I\_x\_graph.push\_back(j);

}

ui->widget\_4->addGraph();

ui->widget\_4->graph(0)->setPen(QPen(color[1]));

ui->widget\_4->graph(0)->setData(I\_x\_graph, I\_y\_graph);

for (int i = 0; i < I\_y\_graph.size(); i++)

I\_y\_graph[i] = fabs(I\_y\_graph[i]);

max\_Y\_old = fabs(\*std::max\_element(I\_y\_graph.begin(), I\_y\_graph.end()));

ui->widget\_4->yAxis->setRange(-max\_Y\_old - 0.1\*max\_Y\_old, max\_Y\_old + 0.1\*max\_Y\_old);

ui->widget\_4->xAxis->setRange(0, I\_y\_graph.size());

ui->widget\_4->replot();

/\* Разрешаем масштабирование и "двигание" графиков \*/

ui->widget->setInteraction(QCP::iRangeZoom, true);

ui->widget->setInteraction(QCP::iRangeDrag, true);

ui->widget->xAxis->setLabel("T");

ui->widget->yAxis->setLabel("X");

ui->widget\_2->setInteraction(QCP::iRangeZoom, true);

ui->widget\_2->setInteraction(QCP::iRangeDrag, true);

ui->widget\_2->xAxis->setLabel("T");

ui->widget\_2->yAxis->setLabel("U");

ui->widget\_3->setInteraction(QCP::iRangeZoom, true);

ui->widget\_3->setInteraction(QCP::iRangeDrag, true);

ui->widget\_3->xAxis->setLabel("T");

ui->widget\_3->yAxis->setLabel("H");

ui->widget\_4->setInteraction(QCP::iRangeZoom, true);

ui->widget\_4->setInteraction(QCP::iRangeDrag, true);

ui->widget\_4->xAxis->setLabel("Iteration");

ui->widget\_4->yAxis->setLabel("I");

init\_u.close();

init\_x.close();

/\* Освобождаем память \*/

free(X);

free(X\_new);

free(w\_old);

free(w\_new);

free(U\_old);

free(U\_new);

}

void MainWindow::on\_lineEdit\_textEdited(const QString &arg1)

{

int Neur\_amount;

QStringList horzHeaders;

Neur\_amount = ui->lineEdit->text().toInt();

ui->tableWidget\_2->setRowCount(Neur\_amount);

ui->tableWidget\_2->setColumnCount(6);

horzHeaders << "Gamma" << "A" << "Ah" << "Bh" << "a" << "b";

ui->tableWidget\_2->setHorizontalHeaderLabels(horzHeaders);

ui->tableWidget->setRowCount(Neur\_amount);

ui->tableWidget->setColumnCount(Neur\_amount);

}