Министерство образования и науки РФ

ФГБОУ ВПО «Тверской Государственный Университет»

Математический факультет

Кафедра компьютерной безопасности и математических методов управления

Направление «Компьютерная безопасность»

ДИПЛОМНАЯ РАБОТА

Исследование динамических нейронных сетей, применяемых при изучении ассоциативной памяти и внимания

|  |  |  |
| --- | --- | --- |
| Допущен к защите: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_г.  Заведующий кафедрой:  (подпись) |  | Автор:  Перова Елена Андреевна  6 курс, 65 группа  Научный руководитель:  доктор физико-математических наук, профессор  Андреева Елена Аркадьевна |

Тверь, 2016 г.

TOC \t "heading 1, 1,heading 2, 2,heading 3, 3"

Введение PAGEREF \_Toc \h 3

Глава 2. Динамические модели нейронных сетей. PAGEREF \_Toc1 \h 7

§2. Модель Ходжкина – Хаксли. PAGEREF \_Toc2 \h 9

§3. Модель Курамото. PAGEREF \_Toc3 \h 10

§4. Модификация модели Курамото. PAGEREF \_Toc4 \h 11

Глава 3. Постановка задачи оптимального управления PAGEREF \_Toc5 \h 14

Глава 4. Методы сведения многокритериальных задач к задаче с одним критерием. PAGEREF \_Toc6 \h 16

. Применение методов оптимизации при исследовании динамической нейронной сети PAGEREF \_Toc7 \h 20

§2. Программная реализация алгоритма поиска численного решения PAGEREF \_Toc8 \h 29

§3.Результаты численного эксперимента. PAGEREF \_Toc9 \h 30

Список литературы. PAGEREF \_Toc10 \h 35

Приложение 1. Код программы PAGEREF \_Toc11 \h 38

# Введение

Интерес к теории нейронных сетей в последние годы привлекает большое внимание многих учёных различных научных сфер: биология, медицина, физика, математика и др. Это обусловлено желанием объяснить и понять принцип работы нервной системы человека.

Активность мозга сложна и изменчива, но, несмотря на это, имеется ряд экспериментальных данных, опираясь на которые можно установить, что между активностью одиночных нейронов, нейронных ансамблей и структур мозга существуют устойчивые соотношения в пространстве и времени.

Теория нейронных сетей в настоящее время является наиболее перспективным направлением в теоретических исследованиях мозга. Поняв и объяснив принцип работы человеческого мозга, возможно последующее применение нейронных сетей в сложных математических вычисления, так как возможности нейронных сетей велики. Они во многом превосходят системы с последовательным вычислением при решении тех задач, в которых машинные вычисления превосходит человеческий мозг.

К задачам, которые на данном этапе своего развития успешно решаются нейронными сетями, относятся:

* распознавание зрительных и слуховых образов. Применение: от распознавания целей (лиц, изображений и прочего) и текстов до голосового управления;
* создание ассоциативных моделей и ассоциативный поиск информации. Применение: синтез человеческой речи, формирование естественного языка;
* формирование различных нелинейных математических систем, а также прогнозирование развития во времени таких систем. Применение: прогнозирование природных явлений (циклонов, извержений вулканов и прочее), изменение валютного рынка и финансовых процессов;
* принятие решений и диагностика в тех областях науки, в которых исключена возможность логического вывода: криминалистика, медицина и др.

Главное и уникальное свойство нейронных сетей — универсальность.

Для решения описанных выше задач, существуют и эффективные математические методы, но благодаря универсальности и перспективности нейронных сетей для решения широкого, можно сказать, глобального спектра задач, например, моделирования мыслительного процесса или проектирования искусственного интеллекта, нейронные сети являются важным направлением дальнейших исследований, которые требуют более тщательного и углублённого изучения.

* исследовать динамическую нейронную сеть, которая применяется при моделировании ассоциативной памяти и внимания
* применение методов оптимизации при исследовании динамических систем, применяемых при моделировании ассоциативной памяти и внимания
* разработка программной реализации методов численного поиска решения

Задачи:

* изучить общую структуру динамических нейронных сетей
* изучить методы оптимизации сложных систем
* рассмотреть способы сведения многокритериальных задач к задаче с одним критерием

,

*Кластеризация.* Под понятием кластеризации понимается возможность разбиения множества входных сигналов на классы, при том условии, что ни признаки классов ни их количество не известны до начала эксперимента. После обучения ИНС способна определить к какому классу относится входной сигнал или же подать знак о том, что ни в какой из выделенных ею классов входной сигнал не входит. Такой ответ означает, что в обучающей выборке отсутствовали классы, которые являются новыми для сети. Таким образом, нейронная сеть подобного образца может выделять новые, неизвестные ранее классы входных сигналов.

*Прогнозирование*. Способность нейронной сети к выделению и обобщению скрытых зависимостей между входными и выходными сигналами определяют ещё одно важное применение ИНС — прогнозирование. После своего обучения нейронная сеть способна спрогнозировать будущее значение некой входной последовательности значений на основе некоторых предыдущих значений или каких-либо значимых факторов. Следует заметить, что дальнейшее прогнозирование возможно только тогда, когда предыдущие изменения в какой-то степени определяют будущие.

*Ассоциативная память.* Способность нейронных сетей к выявлению зависимостей и связей между различными предметами и явлениями даёт возможность выражения данных большого объёма в более компактном виде, если эти данные связаны между собой достаточно тесно. Существует и обратный процесс — восстановление исходного набора данных из части информации. Это и называется ассоциативной памятью. Она позволяет восстанавливать исходный сигнал из частично повреждённых входных данных.

*.*

* Двоичные нейронные сети — работают с информацией, которая представлена в двоичной кодировке;
* Аналоговые нейронные сети — такие сети используют информацию в формате действительных чисел;
* Образные нейронные сети — имеют дело с информацией, которая представлена в виде образов: знаков, символов.

*Классификация по характеру синапсов.*

* Нейронные сети с фиксированными связями — все весовые коэффициенты сети выбираются сразу и исходя из условий задачи, при этом выполнено условие , где *w –* весовые коэффициенты
* Нейронные сети с динамическими связями — для таких сетей происходит настройка связей в процессе обучения, при этом , где *w –* весовые коэффициенты.

# Глава 2. Динамические модели нейронных сетей.

Динамические сети представляют собой сети, построенные из динамических нейронов. Поведение таких нейронов описывается дифференциальными уравнениями, как правило, - первого порядка. При этом организация сети такова, что каждый нейрон получает информацию из окружающей среды и от других нейронов. Этот тип сетей имеет довольно важное значение, так как с его помощью можно моделировать различные нелинейные динамические системы. Такие сети можно использовать главным образом при моделировании: ассоциативной памяти, задач управления, нелинейной обработки сигналов.

.

При этом сигналы в сети принимают значения {-1,1}. Из-за такого поведения, такие значения называют спинами. Обучение сети состоит в запоминании нескольких образов . Запоминание происходит за счёт настройки весовых коэффициентов по правилу:

(1.1)

При этом параметры сети можно определить следующим образом:

(1.2)

Где *w* матрица, составленная из весовых коэффициентов нейронов, а *m* – количество образов для запоминания. При этом по диагонали матрицы *w* обычно стоят нулевые коэффициенты, это сделано для того, чтобы исключить влияние нейронов на самих себя. Заданные таким образом весовые коэффициенты определяют устойчивые состояния сети, которые соответствуют запомненным образам.

Функция активации для нейронов в сети имеет вид:

(1.3)

Начальный, то есть входной образ, присваивается сети как начальное приближение. После задания начальных параметров, начинается итерационная процедура, которая приведёт к устойчивому состоянию. Под устойчивым состоянием в этом случае подразумевается один из запомненных образов. Причём не случайный образ, а тот образ, который наиболее похож на входной сигнал. Другими словами, тот образ, который ассоциативно связан с начальным.

При этом энергию сети, то есть, силу взаимодействия нейронов, можно описать следующим образом:

(1.4)

здесь N – количество нейронов в сети. В этом случае каждому запомненному образу будет соответствовать минимальное локальное значение энергии.

Несмотря на свою простоту, сети Хопфилда раскрывают три главных свойства мозга. Первое – это динамическая структура работы мозга, когда любые внутренние или внешние возмущения заставляют покинуть текущее состояние локального минимума энергии и перейти в режим динамического поиска нового состояния. Второе – способность прийти в стабильное состояние, которое определяется предыдущей памятью. Третье – ассоциативность переходов, то есть в смене состояний постоянно отслеживается обобщённая близость.

## §2. Модель Ходжкина – Хаксли.

Описание генерации нервного импульса впервые были представлены в работах британских нейрофизиологов, биофизиков Алана Ллойда Ходжкина и Эндрю Хаксли. Первую модель распространения нервного импульса, которая является базовой до сих пор для описания подобных явлений, они описали на примере гигантский нервных волокнах кальмара. Главная идея состоит в том, что механизм распространения импульса вдоль аксона связан с тем, что проницаемость аксонной мембраны зависит от имеющихся напряжений и током, и является различным для разного рода ионов. Главную роль в этом процессе отводят положительно заряженным ионам натрия и калия.

В первой модели, предложенной Ходжкином и Хаксли, рассматривается положительно направленный ток от внутренней к внешней стороне аксонной мембраны. Ток состоит из ионных потоков через мембрану аксонов и тока, который вызван изменением потенциала на аксонной мембране, обладающий в свою очередь ёмкостью С. При этом общее уравнение изменения тока имеет следующий вид:

(2.1)

где С – ёмкость мембраны, – вклад токов за счёт изменения ионов, V – изменение потенциала. На основании данных, которые были получены в результате эксперимента, Ходжкин и Хаксли получили следующее уравнение для для :

(2.2)

Здесь V – потенциал, – величины проводимости аксонной мембраны для соответствующих ионов, – соответственно натриевый, калиевый и ток «утечки», который обусловлен тем, что через мембрану могут протекать токи других ионов, g – величины проводимости аксонной мембраны для соответствующих ионов, , – равновесные потенциалы для соответствующих ионов. Величины m, h, n – переменные, значения которых находятся в интервале от 0 до 1, для них справедливы следующие дифференциальные уравнения:

(2.3)

При этом

,

, (2.4)

,

Если к аксонной мембране приложить импульс тока , то уравнение (1) примет вид:

(2.5)

Совокупность уравнений (1)-(4) представляет собой систему уравнений, которая известна как система Ходжкина – Хаксли. В ходе дальнейших исследований оказалось, что модель воспроизводила такие явления, как порог возбуждения, гиперполяризацию волокна после импульса, рефрактерность (кратковременное снижение возбудимости) и др. Тем самым было окончательно подтверждено, что основой всех явлений, которые так или иначе связаны с возбуждением, лежит свойство аксонной мембраны, а именно её переменная и избирательная проводимость для ионов натрия и калия.

## §3. Модель Курамото.

Одна из наиболее успешных попыток понять коллективный феномен синхронизации принадлежит Йошики Курамото. Он анализировал модель фазовых осцилляторов, которые колеблются с произвольной внутренней частотой, но соединены между собой через синус разности своих фазовых колебаний. Модель Курамото состоит из популяции N связанных осцилляторов, где фаза каждого i-ого нейрона, обозначенная , изменяется во времени в соответствии со следующей формулой:

(3.1)

где – описывает собственную частоту i – ого нейрона и S описывает силу связи между нейронами. При этом сила связи S, описывает тенденцию к синхронизации осцилляторов по отношению к дисперсии этих собственных частот,. Когда сила связи S=0, каждый осциллятор пытается вести себя независимо от своей частоты. Несмотря на это, если сила связи осцилляторов достаточно сильная, то все осцилляторы будут работать синхронно, это называется глобальной синхронизацией. Для того, чтобы охарактеризовать уровень синхронизации между осцилляторами, определяется, так называемый, параметр порядка, который описывается следующей формулой:

(3.2)

Где значение параметра характеризует связность набора осцилляторов, и функция – это средняя фаза.

Модель Курамото хорошо описывает глобальную синхронизацию поведения всего набора фазовых осцилляторов, которая означает что все осцилляторы будут находиться в одной фазе при завершении взаимодействия. Эта ситуация редко встречается в реальной системе. Блокировка фазы или частичная синхронизация случаются более часто. Это случай, когда локальные ансамбли осцилляторов синхронизированы между собой, где каждый осциллятор это расщепление нескольких кластеров взаимно синхронизированных осцилляторов.

## §4. Модификация модели Курамото.

Для того чтобы модель Курамото описывала кластеризацию, необходимо переформулировать формулу (3.1). Для этого необходимо формально описать концепцию локальной синхронизации фазы осцилляторов для отражения внутренней структуры данных. В таблице ниже приведены обозначения, которые мы будем использовать при этом.

Таблица 1. Основные обозначения и определения.

|  |  |
| --- | --- |
| Обозначение | Определение |
| КМ | Модель Курамото |
|  | Набор данных |
| *x* | Объект в наборе данных D |
|  | i-ая величина в объекте x |
| N | Число объектов в системе D |
| d | Размерность набора данных D |
| К | Число кластеров в наборе данных D |
| *x(t)* | Значение объекта х в момент времени t |
| S | Сила связи в КМ |
|  | e-соседство объекта х |
|  | Параметр порядка |
|  | i-ом кластер в наборе данных D |
|  | Количество объектов в i-ом кластере |
|  | Количество свободных параметров |

Для применения КМ для описания кластеризации, необходимо переформулировать формулу (3.1). Нам надо формально описать процесс локальной синхронизации фазовых осцилляторов, чтобы отразить внутреннюю структуру данных. В таблице 1 приведены обозначения, которые мы будем использовать ниже. Для описания локальной синхронизации, необходимо ввести несколько понятий.

*ε-соседство для объекта х*. Под ε-соседством для объекта *х,* которое обозначается , понимают:

(4.1)при этом соответствующаяметрика.

*Расширение модели Курамото для кластеризации*. Пусть объект данных в наборе и - i-ая величина в объекте x соответственно. Мы считаем каждый объект *х* в качестве фазового осциллятора, в соответствии с формулой (1), на основании ε-соседства. Динамика каждой величины объекта *х* определяется следующей формулой:

(4.2)

Положим , тогда перепишем формулу выше в следующем виде:

+ (4.3)

Необходимым условием является то, что все объекты должны иметь одинаковую частоту *ω*, т.к. разные частоты будут препятствовать образованию кластера. Несмотря на это, конкретный выбор частоты *ω*  не влияет на результат кластеризации. Поэтому слагаемое может быть спокойно проигнорировано. Слагаемое является константой и обычно принимается за единицу. В конечном итоге, динамика каждой величины объекта *х* может быть выражено следующей формулой:

(4.4)

Рассмотрим поведение объекта *х* в момент времени *t = 0*:

- определяют начальную фазу объекта (начальное расположение объекта *х).* Значение – описывает изменение фазы *i-* ого значения объекта *х* в моменты времени *t = (0, …, T)*.

Для того чтобы описать уровень синхронизации между осцилляторами во время процесса синхронизации, необходимо определить ещё несколько параметров. Параметры, используемые в формуле (3.2), подходят для описания процесса глобальной синхронизации, но они являются неэффективными для описания процесса локальных изменений. В частности, они не дают информации о пути синхронизации с точки зрения локальных кластеров, которые важны при определении конкретных кластеров синхронизированных объектов. Для этой цели, вместо рассмотрения глобальной синхронизации, мы определим параметр порядка , означающий силу связи в популяции локальных осцилляторов.

*Параметр порядка кластеризации*. Параметр порядка кластеризации определяет степень локальной синхронизации и выражается формулой:

(4.5)

Значение параметра возрастает в том случае, когда в популяции становиться больше синхронизированных вместе осцилляторов. Динамическая кластеризация завершается при , это показывает локальную связность, которая также называется совершенной локальной синхронизацией. В этот момент, все локальные объекты (в одном кластере) имеют одинаковую фазу.

Рис. 11. Фазовые портреты (слева) и временные реализации колебаний (справа) осциллятора Ван дер Поля: (а), (б), (в)

# Глава 3. Постановка задачи оптимального управления

Теореме (принцип максимума Понтрягина для задачи оптимального управления с запаздыванием). Пусть является локально-оптимальным для задачи (6.1) – (6.6), тогда оптимальное управление удовлетворяет принципу максимума Понтрягина:

(6.7)

где

а сопряженная функция *p(t)* удовлетворяет система дифференциальных уравнений:

(6.8)

с граничными условиями

(6.9)

Условия (6.9) называются условиями трансверсальности.

# Глава 4. Методы сведения многокритериальных задач к задаче с одним критерием.

Едва ли не любая практическая задача приятия решений является многокритериальной задачей. Исходя из этого, в настоящее время теория принятия решений на основе многокритериальности играет важное значение.

Одним из основополагающих понятий, которые используются в этой теории, является понятие эффективности по Парето, или оптимальности решения, представляющее собой своеобразное обобщение понятие точки максимума числовой функции, в том случае, когда функция зависит от нескольких функций. При этом решение является Парето-оптимальным, когда значение любого из параметров можно улучшить, только лишь в случае ухудшения значений остальных параметров. При этом множество состояний системы, эффективных по Парето, называются множеством Парето. Вопросы, связанные с изучением свойств множеств Парето и методов отыскания оптимальных решений, возникают в математической экономике, теории игр, теории оптимального управления и других дисциплинах, в которых возможно принятие решений на основе многокритериальной модели.

Если при принятии решений используются математические методы, то такой подход предполагает построение математической модели, которая формально описывает проблемную ситуацию, т.е ситуацию в которой необходимо сделать выбор. Для таких задач принятия решений или задач оптимизации при условии, когда отсутствуют неопределённые или случайные факты, элементы множества всех альтернативных решений являются компонентами такой модели. При этом необходимо выбрать один элемент из такого множества, он будет являться наилучшим. Выбранный таким образом элемент, называют оптимальным решением.

Сравнение решений по предпочтительности в многокритериальных задачах осуществляется с помощью числовых функций Такие функции называются показателями качества оптимальности или критериями.

При этом общее условие многокритериальности формулируется следующим образом: требуется минимизировать критериальные функции на заданном множестве , которое называется множеством допустимых значений, и

Из вышесказанного, понятие оптимальности при рассмотрении многокритериальных задач является существенным отличием от однокритериальных задач. Т.к. в однокритериальной задаче понятие оптимального решения определяется несколько иначе – это решение, которое обеспечивает максимально возможное значение критерия. При этом в многокритериальной задаче увеличение одного из критериев может привести к уменьшению других, поэтому при рассмотрении понятия оптимальности требуется знать ряд уточняющих условий.

Предположим, что множество допустимых значений строится в n-мерном пространстве. Пусть решение полностью характеризуется вектором Y , т.е. соответствующими значениями всех частных критериев. Числовое m-мерное пространство называется критериальным пространством, его координатами являются . При этом естественно, что каждому Х можно поставить в соответствие точку в пространстве . Если решение Х допустимо, то соответствующая точка в , определяемая вектором Y, является достижимой. Множество таких точек в критериальном пространстве называется множеством достижимости (достижимых векторов) и обозначается G. Таким образом, векторная функция . отображает допустимое множество на множестве достижимости G:

Задача выбора в этом случае состоит в выборе вектора из этого множества, который является наилучшим с точки зрения лица, которое принимает решение. При этом лицом, принимающим решение, может быть как отдельный человек, так и группа лиц, также без дополнительной информации о предпочтениях этого лица нет смысла говорить об оптимальном решении.

В общем случае построение множества достижимости G для реальных многокритериальных задач весьма непростая задача, но если задача обладает, к примеру, линейными свойствами, то такое множество может быть построено.

В основном, в зависимости от предпочтений лица, которое принимает решение можно выделить 3 основных группы многокритериальных методов принятия решений:

1. Диалоговые (интерактивные) методы. Предполагают, что лицо, принимающее решение будет, участвовать в процессе оптимизации и на каждой итерации компьютер предлагает решения, а лицо, которое принимает решение, оценивает предложенные ему варианты и с учётом этих оценок компьютер предлагает новые решения.
2. Методы построения множества эффективных решений с последующим представлением его лицу, которое принимает решение. При этом методы этой группы отличаются друг от друга различными способами построения и представления множества эффективных решений.
3. Основаны на том, что лицо, которое принимает решение, может выразить свои предпочтения до начала процесса многокритериальной оптимизации. В методах этой группы используются различные способы свёртки критериев, установление желаемых уровней критерий и др.

Приведём примеры наиболее часто встречающихся методов сведения многокритериальных задач к задаче с одним критерием.

Метод линейной свёртки. Этот метод основан на линейной свёртке всех частных критериев в один критерий. Это наиболее простой и часто применяемый метод формирования едино целевой функции.

В общем случае выражается формулой:

здесь весовые коэффициенты , i = 1, ..., 4, рассматриваются как показатели относительной значимости отдельных критериев.

Свёртка с неотрицательными весовыми коэффициентами. Суть данного метода состоит в решении последовательности однокритериальных задач, т.е. поиска максимального и минимального значений, которые может принимать целевая функция на допустимом множестве значений. В этом случае целевой функционал можно выразить формулой:

где – положительные числа.

При этом:

Принцип гарантированного результата. В этом случае целевая функция сводится к следующему виду:

При этом:

## . Применение методов оптимизации при исследовании динамической нейронной сети

1.1 Постановка задачи

Ррассматривается задача оптимального управления для системы с запаздыванием, моделирующая динамику искусственной нейронной сети. Динамика сети из нейронов описывается системой обыкновенных дифференциальных уравнений:

Также эту систему дифференциальных уравнений можно представить в следующем виде:

;

, где:

– амплитуда колебаний i-нейрона;

– скорость изменения амплитуды колебаний i-нейрона;

– функция управления, которая характеризует внешнее воздействие на i–ый нейрон;

N – количество нейронов;

T – момент времени;

–коэффициенты, характеризующие нелинейное воздействие на i-нейрон, в частности, влияет на силу затухания колебания i-ого нейрона;

– собственная частота колебания *i* нейрона;

В общем виде совокупность слагаемых  
 и представляет собой гармонический осциллятор.

Совокупность слагаемых называется осциллятором Ван дер Поля и является аналогом силы трения в нелинейной системе, т.е. отвечает за нелинейное затухание, а слагаемое отражает силу воздействия ансамбля нейронов на i-нейрон.

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

1. в конечный момент времени характеристики нейронов должны совпадать с входными данными т.е. за время T должна достигаться целевая точка;
2. Минимизация управления.

Задачи управления можно формализовать в виде следующего целевого функционала:

- весовые коэффициенты в двухкритериальной задаче оптимального управления;

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

1.2. Построение Решения.

#### Принцип максимума Понтрягина

При нахождении оптимального управления в поставленной задаче необходимо воспользоваться принципом максимума Понтрягина. Для применения принципа максимума Понтрягина необходимо записать функцию Понтрягина.

Для поставленной задачи функция Понтрягина имеет следующий вид:

Тогда, принцип максимума будет выглядеть следующим образом:

Исходя из вышесказанного, оптимальное управление может быть найдено, как максимум следующего выражения:

⇒

⇒

#### Краевая задача принципа максимума

Для того чтобы выписать краевую задачу принципа максимума, найдём вид сопряжённых функций.

Сопряженные функции и и их граничные условия и имеют следующий вид ():

Таким образом, краевая задача для принципа максимума включает в себя следующие уравнения:

;

;

;

;

;

;

где ,

Заметим что, принцип максимума позволяет свести задачу нахождения оптимального процесса к решению краевой задачи.

#### Дискретная аппроксимация

Численное решение задачи можно получить для дискретной задачи оптимального управления. Для этого нужно привести исходную задачу к дискретному виду. Разобьем отрезок [0;Т] на q отрезков. Шаг дискретизации = Т/q.

Аппроксимируем интеграл по правилу левых прямоугольников:

Тогда перепишем целевую функцию в следующем виде:

Применим правило Эйлера для аппроксимации производных:

#### Метод множителей Лагранжа

C помощью метода множителей Лагранжа получим необходимое условие оптимальности. Для того, чтобы воспользоваться методом множителей Лагранжа, необходимо составить функцию Лагранжа. Она имеет следующий вид:

Таким образом, условия стационарности имеют следующий вид:

Выразим из полученных условий , , , :

#### Предельный переход

Решение задачи методом множителей Лагранжа должно совпадать с решением краевой задачи. Для того, чтобы проверить правильность вычислений, выполним предельный переход для импульсов:

⇒ =

⇒ =

;

;

1.3. Алгоритм поиска численного решения

#### Метод градиентного спуска

Опишем алгоритм метода градиентного спуска.

1. Задание параметров метода:
   * Начальный набор управлений
   * Шаг градиентного спуска
   * Точность вычислений
2. Задаём параметры модели:
   * Количество нейронов n
   * Количество слоёв q
   * Набор констант , , , ,
   * Время Т
   * Начальные значения Xi, Yi
   * Ограничения Ai
   * Набор частот
3. По формулам

и

вычисляем начальные значения нейронов

1. С помощью формул

,

и

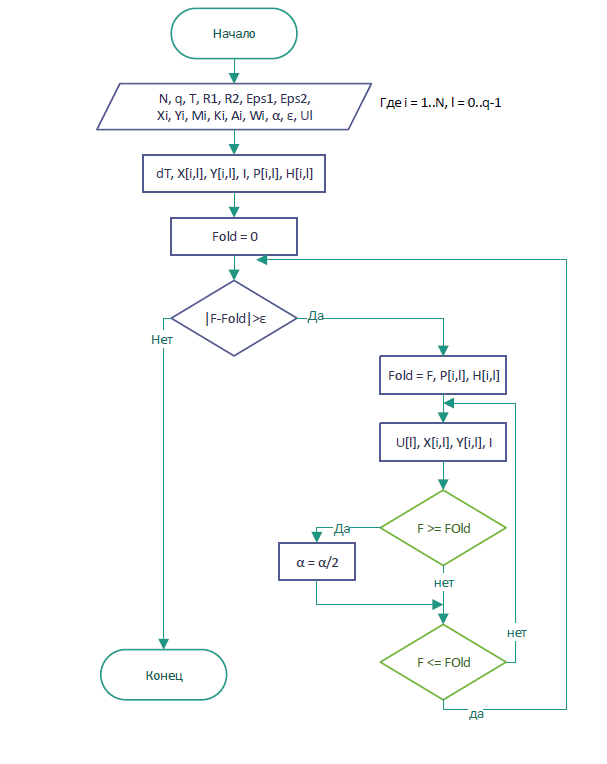
вычисляем значения импульсов

1. По формуле

вычисляем градиент

1. Улучшаем управление на k-ой итерации по формуле
2. Вычисляем [x](k), затем [I](k)
3. Сравниваем значения [I](k) c [I](k-1). Если , то переходим к шагу 3 и значение итерации увеличиваем на 1. Если , то уменьшаем шаг градиентного спуска и переходим к шагу 5 не меняя значения итерации.
4. Проверяем выполнено ли условие остановки . Если условие выполнено, то найдено оптимальное решение в соответствии с заданными параметрами

#### Блок-схема метода



## §2. Программная реализация алгоритма поиска численного решения

В процессе практической реализации оптимизации были поставлены и решены следующие задачи:

1. Проектирование пользовательского интерфейса.
2. Программная реализация.

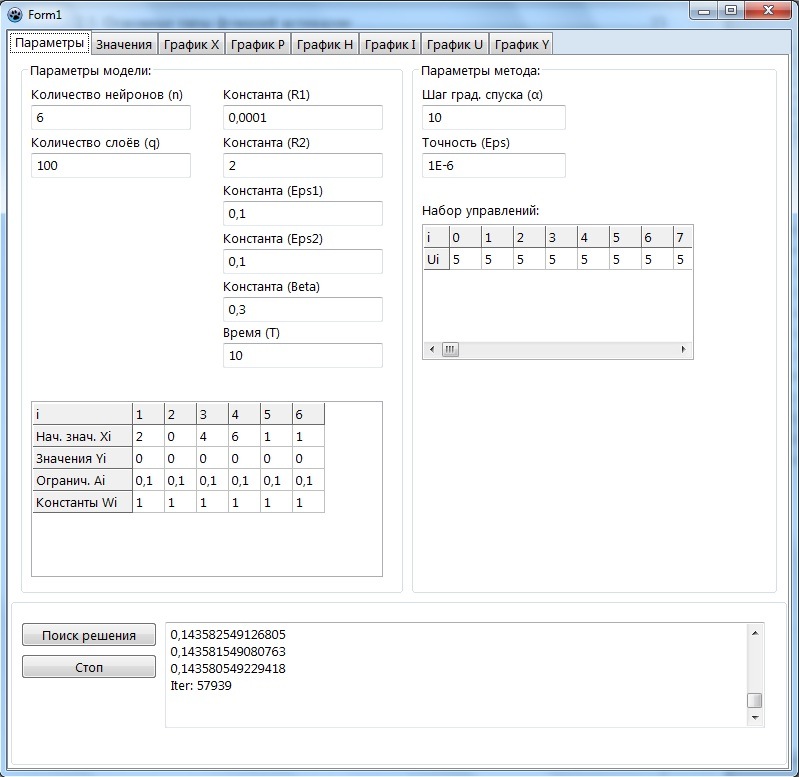
Основными требованиями к пользовательскому интерфейсу являются гибкость, простота и возможность введения всех исходных данных и отображения результатов работы алгоритма в форме, удобной для непосредственного восприятия пользователем программы: в виде числовых значений, таблиц и графиков. Результат проектирования пользовательского интерфейса приведен на рисунке ниже:

Рис.11. Пользовательский интерфейс

Для программной реализации был выбран язык программирования Delphi и среда разработки Lazarus.

Delphi - императивный, структурированный, объектно-ориентированный язык программирования со строгой статической типизацией переменных. Основная область использования — написание прикладного программного обеспечения.

Lazarus – свободная для распространения среда разработки отечественного производства. Позволяет писать программы на языке Delphi с использованием графического интерфейса.

## §3.Результаты численного эксперимента.

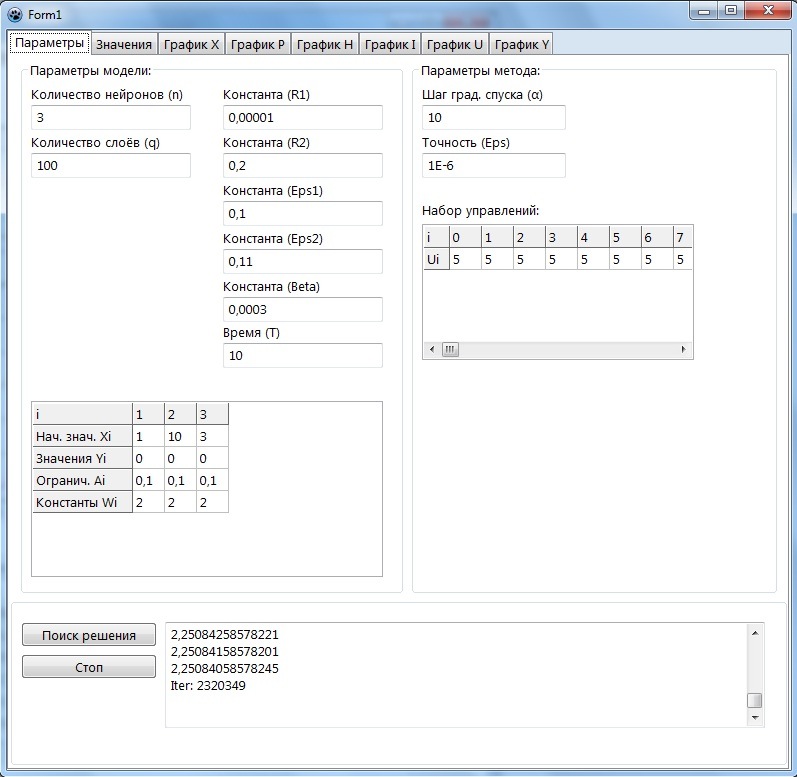
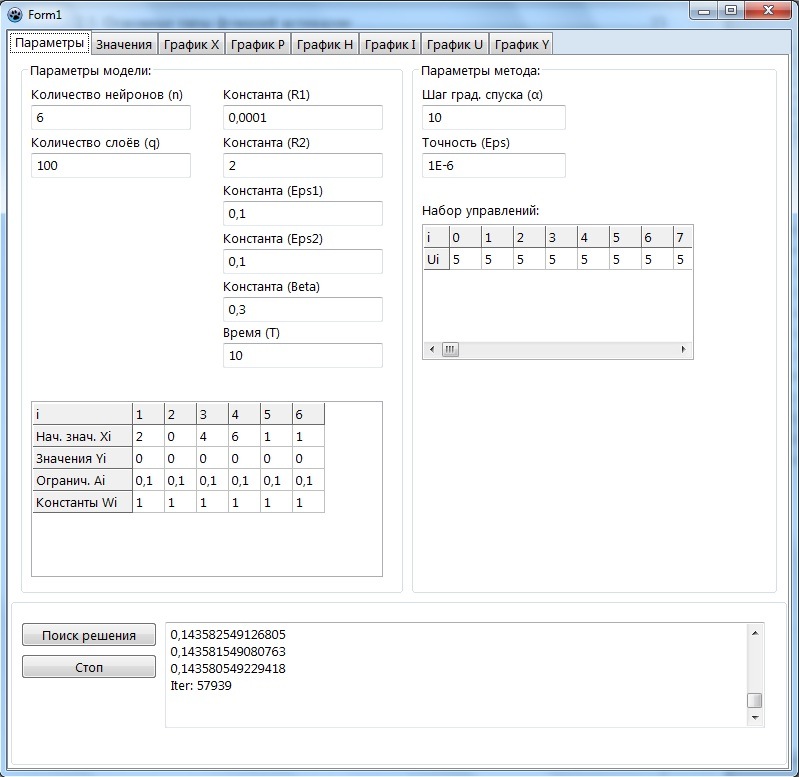
Запустим программу, предварительно введя все начальные данные.

Рис. 13. Начальные данные при исследовании сети из 3 нейронов

Рис. 12. Начальные данные при исследовании сети из 6 нейронов

# 

Рис. 15. Полученные значения для сети из 3 нейронов

Рис. 14. Полученные значения для сети из 6 нейронов

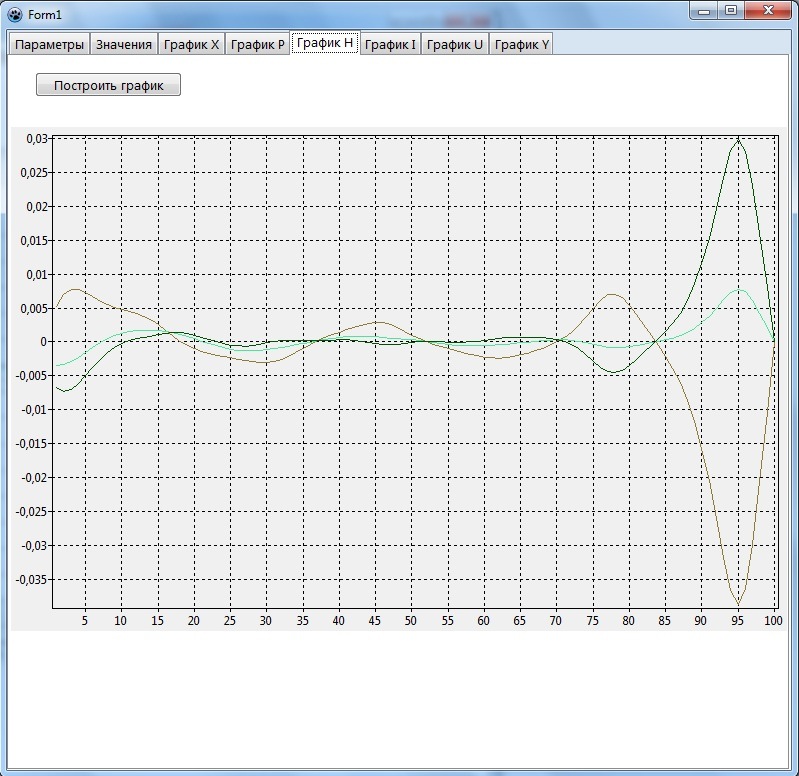
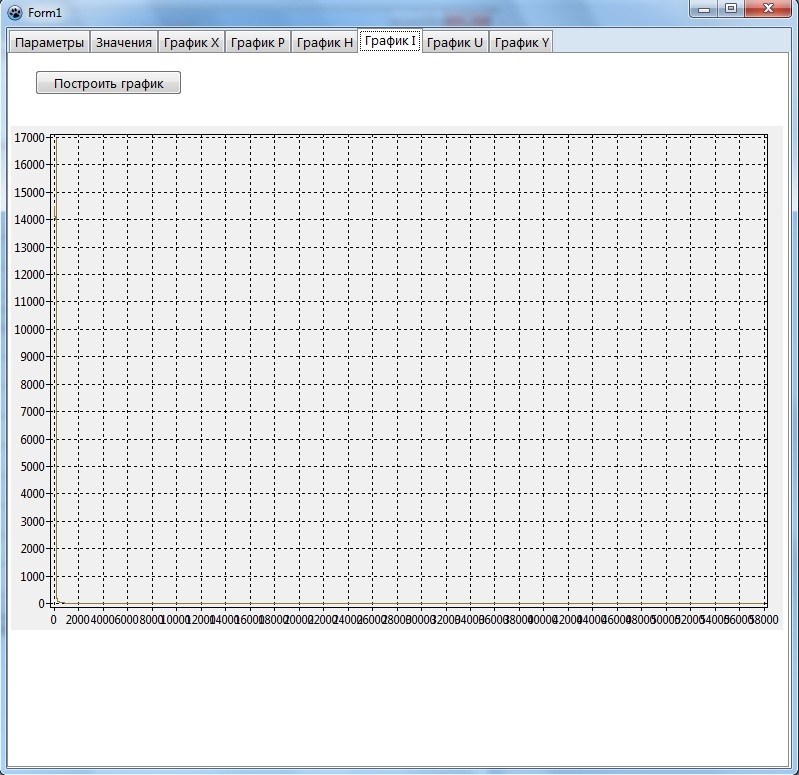
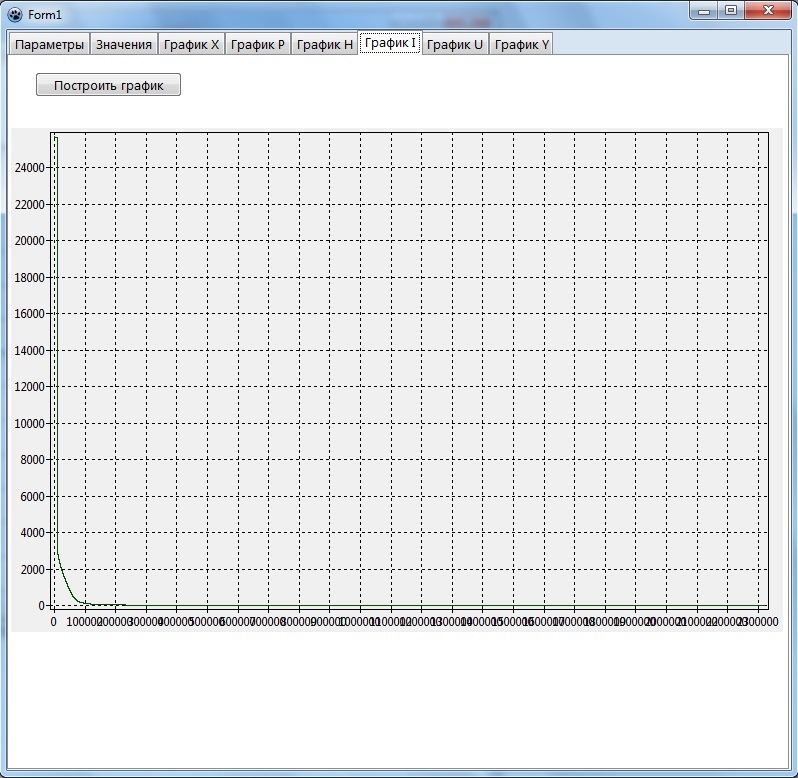


Рис. 16. График сопряжённой функции h(t) для сети из 6 нейронов

Рис. 17. График сопряжённой функции h(t) для сети из 3 нейронов



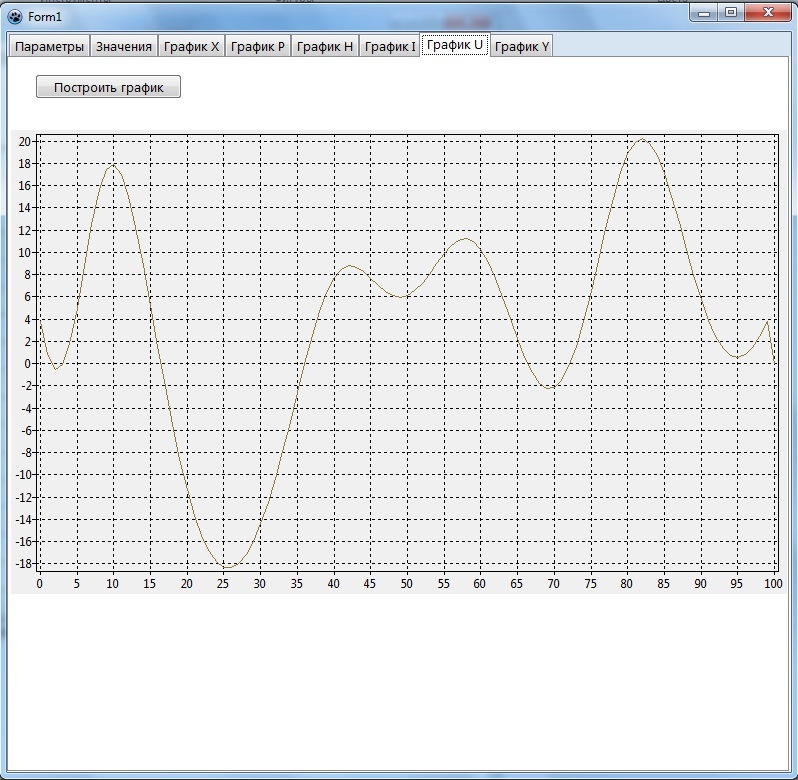
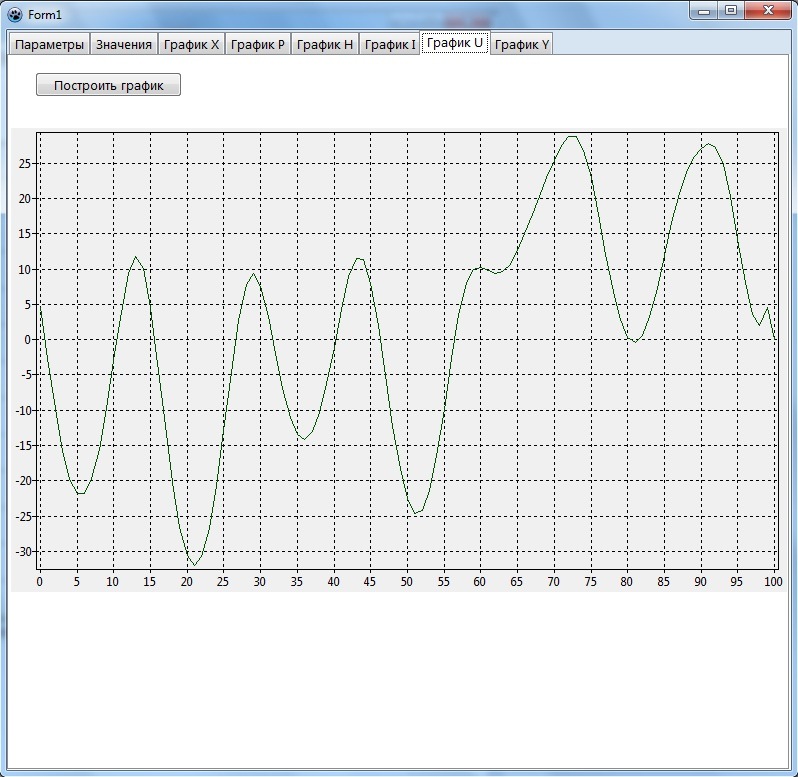
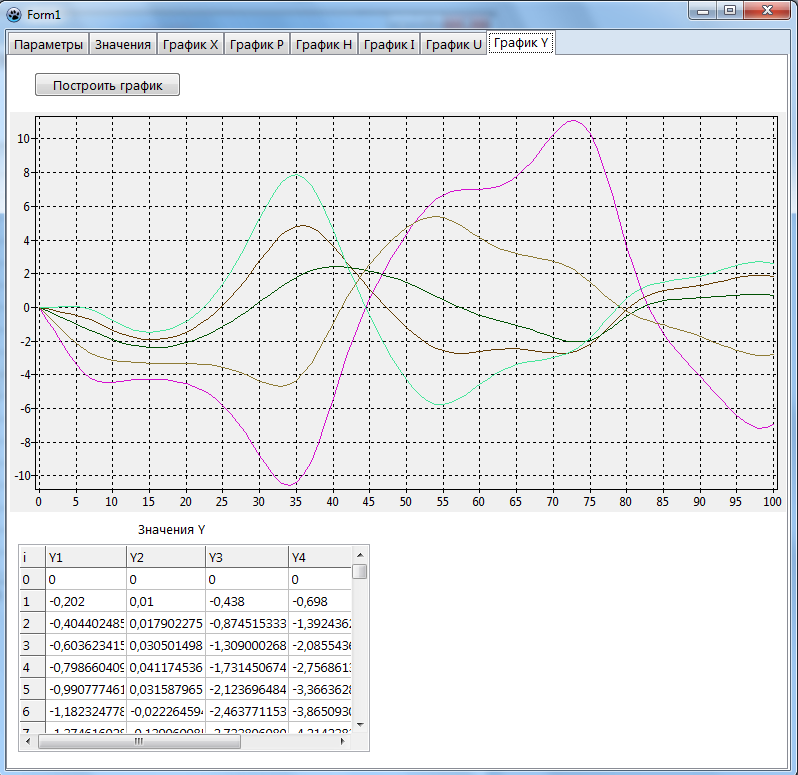
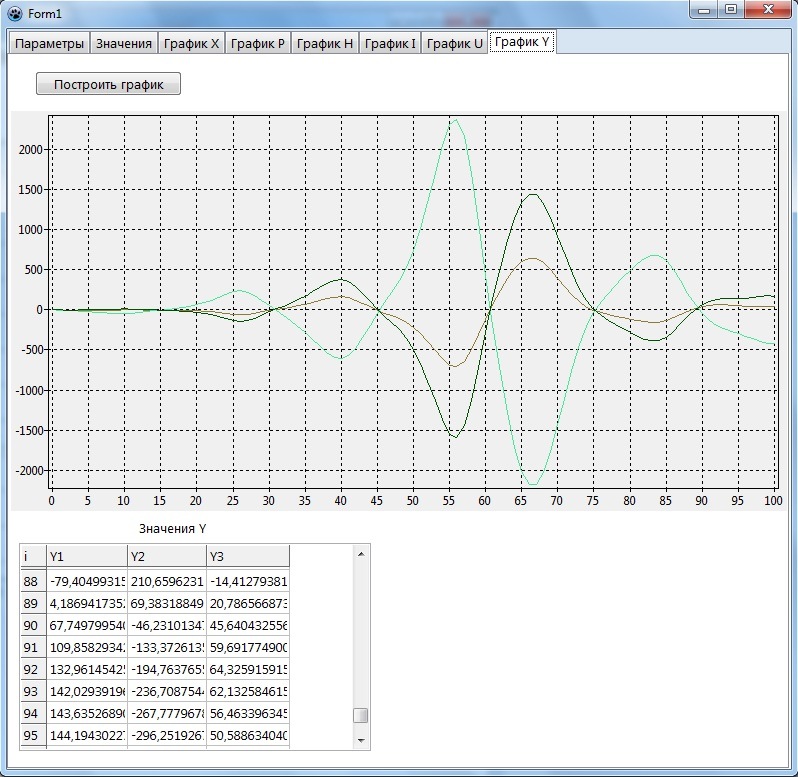


Рис. 19. График целевой функции I(u) для сети из 3 нейронов

Рис. 18. График целевой функции I(u) для сети из 6 нейронов



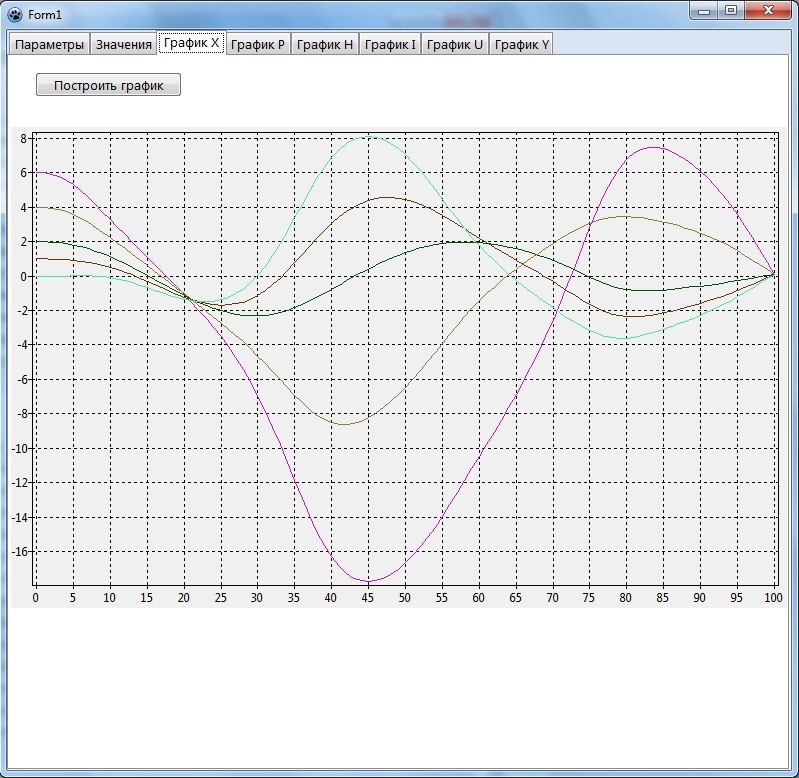


Рис. 23. График динамики нейронов Xi(t) для сети из 3 нейронов

Рис. 22. График динамики нейронов Xi(t) для сети из 6 нейронов

Рис. 21. График функции y(t) для сети из 3 нейронов

Рис. 20. График функции y(t) для сети из 6 нейронов

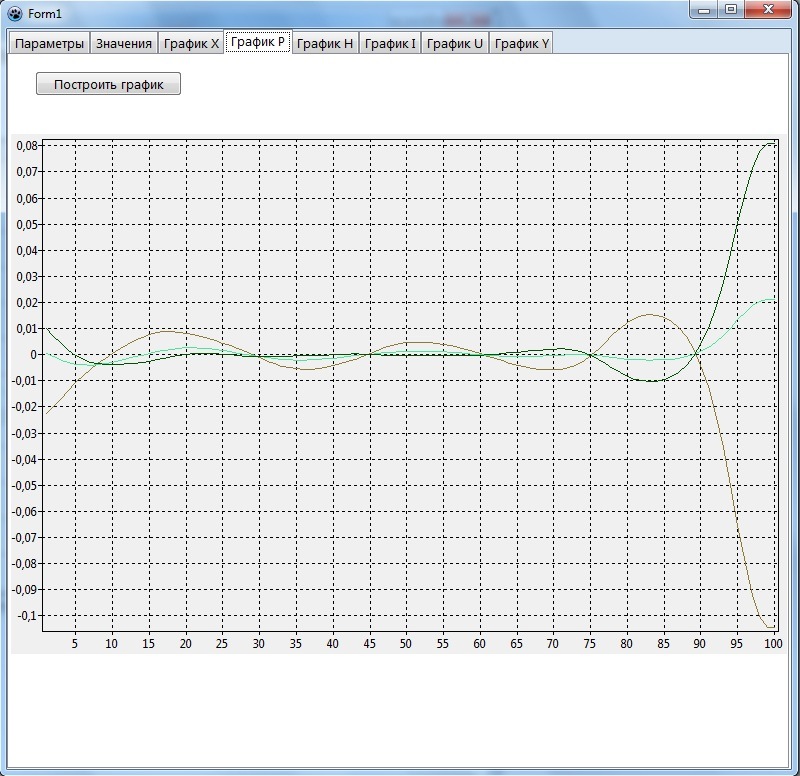
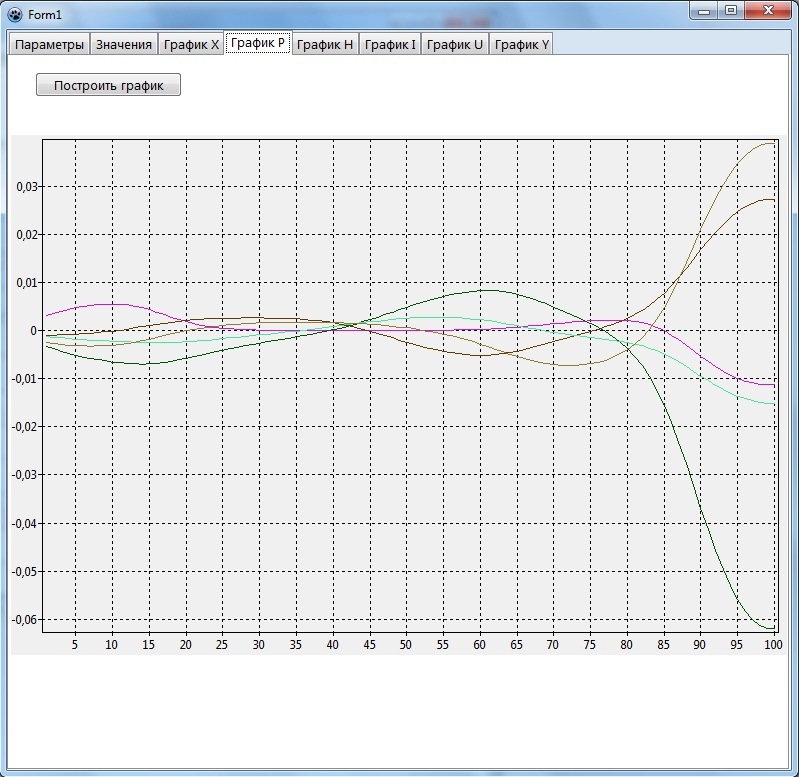


Рис. 25. График сопряжённой функции р(t) для сети из 3 нейронов

Рис. 24. График сопряжённой функции р(t) для сети из 6 нейронов

При рассмотрении сетевых конфигураций или алгоритмов, которые основаны на искусственных нейронных сетях, разработчики, вдохновлённые биологической структурой нейрона, мыслят в терминах организации мозговой деятельности. Но наши знания о работе мозга столь невелики, что разработчикам порой приходится выходить за рамки современных биологических знаний в поисках тех структур, которые были бы способны выполнять те или иные полезные и значимые функции. Во многих случаях это приводит к полному отказу от биологического правдоподобия, мозг в этом случае служит лишь неким идеалом, к которому приходится стремиться, ведь, в ходе исследований и разработки сложных систем, порой создаются такие сети, существование которых невозможно в живой материи или при их функционировании требуется учесть большое количество допущений при рассмотрении анатомии и функционировании мозга.

Но несмотря на это искусственные нейронные сети продолжают сравнивать с мозгом, так как их функционирование зачастую напоминает человеческое познания, поэтому трудно избежать аналогий при изучении.

Учитывая успехи, которые были достигнуты в ходе изучения такой грубой модели мозга, можно ожидать дальнейшего продвижения в исследованиях с помощью более точной модели мозга. Для разработки такой модели требуется более детальное и более точное понимание структуры и функционирования мозга. Это в свою очередь требует определения более точных характеристик нейронов, включая их вычислительные элементы и элементы связи. К сожалению, биохимия нейронов, фундаментальных строительных блоков мозга, остается не разгадано до конца. Ясно одно: нейрон является намного более сложным, чем представлялось несколько лет назад и нет наиболее полного понимания процесса его функционирования.

# Список литературы.

1. Андреева Е.А.Управление динамическими системами ТВГ Тверь 2016
2. Андреева Е.А. Цирулева В.М. Оптимальное управление Москва Высшая школа 2006
3. Уоссермен Ф. - Нейрокомпьютерная техника: теория и практика - М.: Мир, 1992г.
4. Ногин В.Д. - введение в оптимальное управление - Петербург 2008 год Ютас, Санкт-
5. Андреева Е.А., Кратович П.В. Оптимизация нейронных сетей: учебное пособие. – Тверь: Твер. Гос. Ун-т, 2015. - 116 с.
6. Г.Н.Борисюк, Р.М.Борисюк, Я.Б.Казанович, Г.Р.Иваницкий – Модели динамики нейронной активности при обработке информации мозгом –итоги «десятилетия».:Успехи физических наук, том 172, №10, 2002г.
7. В.В.Зайцев, С.В.Линдт, А.Н.Шилин. Осцилляторы Ван дер Поля и Рэлея в дискретном времени.:Вестник СамГУ – Естественнонаучная серия, 2014г. №7(118)
8. С.П.Кузнецова, Ю.В.Седова.Фазовый хаос в динамике ансамбля осцилляторов с модулированной во времени глобальной связью.: Журнал технической физики, 2013г, том 83, вып.1
9. Динамические нейронные сети. Ассоциативность [Электронный ресурс] Режим доступа:[http://www.aboutbrain.ru/2014/03/10/динамические-нейронные-сети-ассоциа/](http://www.aboutbrain.ru/2014/03/10/%25D0%25B4%25D0%25B8%25D0%25BD%25D0%25B0%25D0%25BC%25D0%25B8%25D1%2587%25D0%25B5%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B5-%25D0%25BD%25D0%25B5%25D0%25B9%25D1%2580%25D0%25BE%25D0%25BD%25D0%25BD%25D1%258B%25D0%25B5-%25D1%2581%25D0%25B5%25D1%2582%25D0%25B8-%25D0%25B0%25D1%2581%25D1%2581%25D0%25BE%25D1%2586%25D0%25B8%25D0%25B0/)
10. Нейронная сеть Хопфилда[Электронный ресурс] Режим доступа: <http://microtechnics.ru/nejronnaya-set-xopfilda/>
11. Обучение нейронной сети с запаздыванием[Электронный ресурс] Режим доступа: <http://cyberleninka.ru/article/n/obuchenie-neyronnoy-seti-s-zapazdyvaniem>
12. Классификация нейронных сетей [Электронный ресурс] Режим доступа: <http://cyberleninka.ru/article/n/klassifikatsiya-neyronnyh-setey>
13. Оптимизация нейронных сетей с учётом запаздывания [Электронный ресурс] Режим доступа: <http://tekhnosfera.com/view/40252/d?#?page=5>
14. Моделирование нейронной сери с учётом запаздывания [Электронный ресурс] Режим доступа: <http://www.mathnet.ru/links/64a0e56539997cf7d9bb2f6684237a6d/mm2481.pdf>
15. Некоторые модели клеточных нейронных сетей и их исследование [Электронный ресурс] Режим доступа: <http://www.dissercat.com/content/nekotorye-modeli-kletochnykh-neironnykh-setei-i-ikh-issledovanie>
16. Организация интеллектуальных вычислений [Электронный ресурс] Режим доступа: <http://victoria.lviv.ua/html/oio/index_rus.html> (Дата обращения 15.10.2016)
17. Реализация однослойной нейронной сети [Электронный ресурс] Режим доступа: <http://perpetum-mobile.ru/realizatsiya-odnosloinoi-neiroseti-perceptron/> (Дата обращения 15.10.2016)
18. Элементы нейронных сетей [Электронный ресурс] Режим доступа: <http://www.intuit.ru/studies/courses/6/6/lecture/178?page=2> (Дата обращения 10.11.2016)
19. Модели нейронных сетей [Электронный ресурс] Режим доступа: <http://www.intuit.ru/studies/courses/6/6/lecture/178?page=5> (Дата обращения 10.11.2016)
20. Функции активации в нейронных сетей [Электронный ресурс] Режим доступа: <http://www.aiportal.ru/articles/neural-networks/activation-function.html> (Дата обращения 12.11.2016)

# Приложение 1. Код программы

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, TAGraph, TASeries, Forms, Controls, Graphics,

Dialogs, StdCtrls, ComCtrls, ExtCtrls, Grids, Math;

type

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

Chart1: TChart;

Chart1LineSeries1: TLineSeries;

Chart1LineSeries2: TLineSeries;

Chart1LineSeries3: TLineSeries;

Chart1LineSeries4: TLineSeries;

Chart2: TChart;

Chart3: TChart;

Chart4: TChart;

Chart5: TChart;

Chart6: TChart;

Chart7: TChart;

GroupBox1: TGroupBox;

GroupBox2: TGroupBox;

GroupBox3: TGroupBox;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

LabeledEdit1: TLabeledEdit;

LabeledEdit2: TLabeledEdit;

LabeledEdit3: TLabeledEdit;

LabeledEdit4: TLabeledEdit;

LabeledEdit5: TLabeledEdit;

LabeledEdit6: TLabeledEdit;

LabeledEdit7: TLabeledEdit;

LabeledEdit8: TLabeledEdit;

LabeledEdit9: TLabeledEdit;

LabeledEdit10: TLabeledEdit;

Memo1: TMemo;

PageControl1: TPageControl;

StringGrid1: TStringGrid;

StringGrid2: TStringGrid;

StringGrid3: TStringGrid;

StringGrid4: TStringGrid;

StringGrid5: TStringGrid;

StringGrid6: TStringGrid;

StringGrid7: TStringGrid;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

TabSheet4: TTabSheet;

TabSheet5: TTabSheet;

TabSheet6: TTabSheet;

TabSheet7: TTabSheet;

TabSheet8: TTabSheet;

TabSheet9: TTabSheet;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure LabeledEdit1Change(Sender: TObject);

procedure LabeledEdit2Change(Sender: TObject);

private

{ private declarations }

public

{ public declarations }

end;

type mas1 = Array of Real;

mas2 = Array of Array of Real;

mas3 = Array of Array of Array of Real;

var

Form1: TForm1;

GlobalFlagStop: boolean;

VneshU:mas1;

VneshX:mas2;

VneshY:mas2;

VneshP:mas2;

VneshH:mas2;

VneshN:integer;

VneshQ:integer;

VneshHPontr: mas2;

implementation

{$R \*.lfm}

{ TForm1 }

//вычисление СУММ(Yj-Yi)

function Calc\_sum\_Y(i,l:integer; Y:mas2; n:integer):real;

var a:real;

j:integer;

begin

a:=0;

for j:=1 to n do

a:= a + Y[j,l] - Y[i,l];

Calc\_sum\_Y := a;

end;

//вычисление CУММ(Hi)

function Calc\_sum\_H(H:mas2; n,l:integer):real;

var i:integer;

a:real;

begin

a:=0;

for i:=1 to n do

a:= a + H[i,l];

Calc\_sum\_H := a;

end;

//вычисление CУММ(Hi\*(Yj-Yi))

function Calc\_sum\_HY(H,Y:mas2; n,l:integer):real;

var i, j:integer;

a: real;

begin

a:=0;

for i:=1 to n do

for j:=1 to n do

a:= a + H[i,l+1]\*(Y[j,l] - Y[i,l]);

Calc\_sum\_HY := a;

end;

//Вычисление импульсов P и H

procedure calc\_PH(var P,H:mas2; X:mas2; U,W,A:mas1; dT,Eps1,Eps2,R2,B:real; n,q:integer);

var i,l:integer;

begin

for i:= 1 to n do

begin

P[i,q]:=-2\*R2\*(X[i,q]-A[i]);

H[i,q]:=0;

end;

for l:=q-1 downto 1 do

for i:=1 to n do

begin

P[i,l]:=P[i,l+1] - H[i,l+1]\*dT\*sqr(W[i]) - 2\*dT\*Eps1\*B\*H[i,l+1]\*X[i,l];

H[i,l]:=H[i,l+1] + dT\*P[i,l+1] + dT\*Eps2\*U[l]/n\*(Calc\_sum\_H(H,n,l+1))-dT\*Eps2\*U[l]\*H[i,l+1];

end;

end;

//Вычисление оптмального управления

procedure calc\_U(var U:mas1; UOld:mas1; YOld,H:mas2; alf,R1,Eps2,dT:real; n,q:integer);

var l:integer;

begin

for l:=0 to q-1 do

U[l]:= UOld[l] - alf\*(dT\*R1\*UOld[l] - dT\*Eps2/n\*Calc\_sum\_HY(H,YOld,n,l));

end;

procedure perebrosMas1(Mas:mas1;var MasOld:mas1; n:integer);

var i:integer;

begin

for i:=0 to n-1 do

MasOld[i]:=Mas[i];

end;

//Вычисление функции I

function calc\_I(X:mas2; U,A:mas1; R1,R2,dT:real; n,q:integer): real;

var i, l: Integer;

I1, I2: Real;

begin

I1:=0;

I2:=0;

for l:=0 to q-1 do

I1 := I1 + sqr(U[l]);

I1:= I1\*dT\*R1/2;

for i:=1 to n do

for l:=0 to q-1 do

I2:=I2 + R2\*sqr(X[i,q] - A[i]);

calc\_I := I1 + I2;

end;

//Вычисление нейронов X и Y

procedure calc\_XY(var X:mas2; var Y:mas2; U,NX,NY,W:mas1; n,q:integer; dT,Eps1,Eps2,B:real);

var i,l :integer;

begin

for i:=1 to n do

begin

X[i,0]:=NX[i];

Y[i,0]:=NY[i];

end;

for l:=0 to q-1 do

for i:=1 to n do

begin

while(abs(X[i,l] - X[i,l+1]) > Eps1) do

begin

X[i,l+1]:= X[i,l] + dT\*Y[i,l];

Y[i,l+1]:= Y[i,l] - dT\*sqr(W[i])\*X[i,l] + dT\*Eps1\*(1-B\*sqr(X[i,l])) + dT\*Eps2\*U[l]/n\*Calc\_sum\_Y(i,l,Y,n);

end;

end;

end;

//Вычисление функции Понтрягина

procedure calc\_HPontr(var X:mas2; var Y, HPontr:mas2;var P,H:mas2; U,W:mas1; n,q:integer; Eps1,Eps2,B,R1:real);

var i, l: Integer;

begin

for i:=1 to n do

for l:=0 to q-1 do

HPontr[i,l]:= P[i,l]\*Y[i,l]+H[i,l]\*Eps1\*(1-B\*sqr(X[i,l]))+R1\*(1/2)\*sqr(U[i])+H[i,l]\*Eps2\*U[i]\*Calc\_sum\_Y(i,l,Y,n)+H[i,l]\*sqr(W[i])\*X[i,l];

end;

//Поиск решения

procedure TForm1.Button3Click(Sender: TObject);

var i,j,k,iter:integer;

n,q:integer; //кол-во нейронов n и слоев q

alf, eps:real; //шаг град.спуска и точность вычислений

FOld, F: real; //старое и новое значение функции

NX, NY:mas1; //начальные значения X и Y

UOld, U:mas1; //управления

X,Y:mas2; //вычисленные значение нейронов X и Y

P, H, HPontr:mas2; //импульсы P и H

R1, R2, T, Eps1, Eps2, dT, B:real; //набор констант, время T и dT

A:mas1;

W:mas1;

begin

GlobalFlagStop := false;

memo1.clear;

//Cчитывание переменных

if (LabeledEdit1.Text <> '') then

q := StrToInt(LabeledEdit1.Text);

if (LabeledEdit2.Text <> '') then

n := StrToInt(LabeledEdit2.Text);

if (LabeledEdit3.Text <> '') then

Eps2 := StrToFloat(LabeledEdit3.Text);

if (LabeledEdit4.Text <> '') then

alf := StrToFloat(LabeledEdit4.Text);

if (LabeledEdit5.Text <> '') then

Eps := StrToFloat(LabeledEdit5.Text);

if (LabeledEdit6.Text <> '') then

R1 := StrToFloat(LabeledEdit6.Text);

if (LabeledEdit7.Text <> '') then

T := StrToFloat(LabeledEdit7.Text);

if (LabeledEdit8.Text <> '') then

Eps1 := StrToFloat(LabeledEdit8.Text);

if (LabeledEdit9.Text <> '') then

R2 := StrToFloat(LabeledEdit9.Text);

if (LabeledEdit10.Text <> '') then

B := StrToFloat(LabeledEdit10.Text);

//Размерность массивов

Setlength(NX, n+1);

Setlength(NY, n+1);

Setlength(A, n+1);

Setlength(W, n+1);

Setlength(U, q+1);

Setlength(UOld, q+1);

SetLength(X, n+1, q+1);

SetLength(Y, n+1, q+1);

Setlength(P, n+1, q+1);

SetLength(H, n+1, q+1);

SetLength(HPontr, n+1, q+1);

//Считываем значения из таблицы 1

for i := 1 to n do

begin

if (StringGrid1.Cells[i,1] <> '') then

NX[i] := StrToFloat(StringGrid1.Cells[i,1]);

if (StringGrid1.Cells[i,2] <> '') then

NY[i] := StrToFloat(StringGrid1.Cells[i,2]);

if (StringGrid1.Cells[i,3] <> '') then

A[i] := StrToFloat(StringGrid1.Cells[i,3]);

if (StringGrid1.Cells[i,4] <> '') then

W[i] := StrToFloat(StringGrid1.Cells[i,4]);

end;

//Считываем управление из таблицы 2

for k:=0 to q-1 do

if (StringGrid2.Cells[k+1,1] <> '') then

U[k] := StrToFloat(StringGrid2.Cells[k+1,1])

else

begin

U[k] := 0;

StringGrid2.Cells[k,1]:=IntToStr(0);

end;

//Вычисление

dT:= T/q;

iter:=0;

calc\_XY(X,Y,U,NX,NY,W,n,q,dT,Eps1,Eps2,B);

F := calc\_I(X,U,A,R1,R2,dT,n,q);

calc\_HPontr(X,Y, HPontr,P,H,U,W,n,q,Eps1,Eps2,B,R1);

calc\_PH(P,H,X,U,W,A,dT,Eps1,Eps2,R2,B,n,q);

FOld := 0;

memo1.lines.Add(FloatToStr(F));

while(abs(F - FOld) > eps) do

begin

FOld:= F;

perebrosMas1(U, UOld, q);

iter:=iter+1;

calc\_PH(P,H,X,U,W,A,dT,Eps1,Eps2,R2,B,n,q);

repeat

calc\_U(U,UOld,Y,H,alf,R1,Eps2,dT,n,q);

calc\_XY(X,Y,U,NX,NY,W,n,q,dT,Eps1,Eps2,B);

F:=calc\_I(X,U,A,R1,R2,dT,n,q);

calc\_HPontr(X,Y, HPontr,P,H,U,W,n,q,Eps1,Eps2,B,R1);

memo1.lines.Add(FloatToStr(F));

Application.ProcessMessages;

if (GlobalFlagStop) then

begin

GlobalFlagStop := false;

exit;

end;

if F >= FOld then alf:=alf/2;

until (F <= FOld);

end;

memo1.lines.Add('Iter: ' + IntToStr(iter));

//Вывод результатов

//Значения Х

StringGrid3.ColCount := n + 1;

StringGrid3.RowCount := q + 2;

for i:= 1 to n do

StringGrid3.Cells[i,0] := 'X'+IntToStr(i);

for i:=0 to q do

StringGrid3.Cells[0,i+1] := IntToStr(i);

for i := 1 to n do

for j := 1 to q + 1 do

StringGrid3.Cells[i,j] := FloatToStr(X[i,j-1]);

//Значения Y

StringGrid7.ColCount := n + 1;

StringGrid7.RowCount := q + 2;

for i:= 1 to n do

StringGrid7.Cells[i,0] := 'Y'+IntToStr(i);

for i:=0 to q do

StringGrid7.Cells[0,i+1] := IntToStr(i);

for i := 1 to n do

for j := 1 to q + 1 do

StringGrid7.Cells[i,j] := FloatToStr(Y[i,j-1]);

//Значения U

StringGrid4.RowCount:= q +1;

for i:=1 to q do

begin

StringGrid4.cells[0,i]:=IntToStr(i-1);

StringGrid4.cells[1,i]:=FloatToStr(U[i-1]);

end;

//Значения P

StringGrid5.ColCount := n + 1;

StringGrid5.RowCount := q + 2;

for i:= 1 to n do

StringGrid5.Cells[i,0] := 'P'+IntToStr(i);

for i:=0 to q do

StringGrid5.Cells[0,i+1] := IntToStr(i);

for i := 1 to n do

for j := 1 to q + 1 do

StringGrid5.Cells[i,j] := FloatToStr(P[i,j-1]);

//Значения H

StringGrid6.ColCount := n + 1;

StringGrid6.RowCount := q + 2;

for i:= 1 to n do

StringGrid6.Cells[i,0] := 'H'+IntToStr(i);

for i:=0 to q do

StringGrid6.Cells[0,i+1] := IntToStr(i);

for i := 1 to n do

for j := 1 to q + 1 do

StringGrid6.Cells[i,j] := FloatToStr(H[i,j-1]);

Setlength(VneshU, q+1);

SetLength(VneshX, n+1, q+1);

Setlength(VneshP,n+1,q+1);

SetLength(VneshY, n+1, q+1);

SetLength(VneshHPontr, n+1, q+1);

VneshU:=U;

VneshX:=X;

VneshP:=P;

VneshH:=H;

VneshN:=n;

VneshQ:=q;

VneshY:=Y;

VneshHPontr:=HPontr;

end;

//График Х

procedure TForm1.Button2Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart1.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart1);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart1.AddSeries(MyLineSerie);

for j := 0 to VneshQ do

MyLineSerie.AddXY(j, VneshX[i, j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График Y

procedure TForm1.Button9Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart6.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart6);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart6.AddSeries(MyLineSerie);

for j := 0 to VneshQ do

MyLineSerie.AddXY(j, VneshY[i, j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График Р

procedure TForm1.Button5Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart2.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart2);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart2.AddSeries(MyLineSerie);

for j := 1 to VneshQ do

MyLineSerie.AddXY(j, VneshP[i, j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График H

procedure TForm1.Button6Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart3.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart3);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart3.AddSeries(MyLineSerie);

for j := 1 to VneshQ do

MyLineSerie.AddXY(j, VneshH[i, j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График I

procedure TForm1.Button4Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart4.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart4);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart4.AddSeries(MyLineSerie);

for j := 0 to memo1.lines.Count-2 do

MyLineSerie.AddXY(j, StrToFloat(memo1.Lines[j]), IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График U

procedure TForm1.Button7Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart5.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart5);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart5.AddSeries(MyLineSerie);

for j := 0 to VneshQ do

MyLineSerie.AddXY(j, VneshU[j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

//График функции Понтрягина

procedure TForm1.Button8Click(Sender: TObject);

var

MyLineSerie: TLineSeries;

j, i: integer;

Temp\_color: TColor;

begin

Chart5.ClearSeries;

Temp\_color := $000000;

for i := 1 to VneshN do

begin

MyLineSerie := TLineSeries.Create(Chart5);

MyLineSerie.LinePen.Color := $025100 + Temp\_color;

MyLineSerie.LinePen.Width:= 3;

Chart5.AddSeries(MyLineSerie);

for j := 0 to VneshQ do

MyLineSerie.AddXY(j, VneshHPontr[i, j], IntToStr(j));

Temp\_color := Temp\_color + $999547;

end;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

GlobalFlagStop := true;

end;

//Значения переменных по-умолчанию при создании формы

procedure TForm1.FormCreate(Sender: TObject);

var alf,eps:real;

n,q: integer;

R1, R2,T,Eps1,Eps2,B:real;

begin

n:=4;

q:=100;

R1:=0.0001;

R2:=2;

T:=10;

B:=0.3;

Eps1:=0.1;

Eps2:=0.1;

alf:=10;

eps:=0.000001;

LabeledEdit1.text:=intToStr(q);

LabeledEdit2.text:=intToStr(n);

LabeledEdit3.text:=FloatToStr(Eps2);

LabeledEdit4.text:=floatToStr(alf);

LabeledEdit5.text:=floatToStr(Eps);

LabeledEdit6.text:=FloatToStr(R1);

LabeledEdit7.text:=FloatToStr(T);

LabeledEdit8.text:=FloatToStr(Eps1);

LabeledEdit9.text:=FloatToStr(R2);

LabeledEdit10.text:=FloatToStr(B);

end;

procedure TForm1.LabeledEdit1Change(Sender: TObject);

var q,i:integer;

begin

if (LabeledEdit1.Text <> '') then

begin

q:= StrToInt(LabeledEdit1.Text);

StringGrid2.ColCount := q + 1;

for i:=1 to q do

begin

StringGrid2.Cells[i,0] := IntToStr(i - 1);

StringGrid2.Cells[i,1] := IntToStr(5);

end;

end;

end;

procedure TForm1.LabeledEdit2Change(Sender: TObject);

var i,n:integer;

begin

if (LabeledEdit2.Text <> '') then

begin

n := StrToInt(LabeledEdit2.Text);

StringGrid1.ColCount := n + 1;

StringGrid1.Cells[1,1] := IntToStr(2);

StringGrid1.Cells[2,1] := IntToStr(3);

StringGrid1.Cells[3,1] := IntToStr(4);

StringGrid1.Cells[4,1] := IntToStr(6);

{

StringGrid1.Cells[1,2] := IntToStr(5);

StringGrid1.Cells[2,2] := IntToStr(0);

StringGrid1.Cells[3,2] := IntToStr(17);

StringGrid1.Cells[4,2] := IntToStr(8);

}

for i:= 1 to n do

begin

StringGrid1.Cells[i,2] := IntToStr(0);

StringGrid1.Cells[i,0] := IntToStr(i);

StringGrid1.Cells[i,3] := FloatToStr(0.1);

StringGrid1.Cells[i,4] := FloatToStr(0.8);

end;

end;

end;

end.