МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра механики, мехатроники и робототехники

СПЕЦГЛАВЫ ТЕОРИИ УПРАВЛЕНИЯ МЕХАТРОННЫМИ СИСТЕМАМИ

методические указания по выполнению лабораторных работ для студентов направления 15.03.06 «Мехатроника и робототехника»

Курск 2020

УДК 681.323
Составители: П.А. Безмен, В.В. Бартенев

Рецензент:
Кандидат технических наук, доцент Юго-Западного государственного университета *Е.Н. Политов*

Спецглавы теории управления мехатронными системами: методические указания по выполнению лабораторных работ для студентов направления 15.03.06 «Мехатроника и робототехника» / Юго-Зап. гос. ун-т; сост. П.А. Безмен, В.В. Бартенев. Курск, 2020. 70 с.

Изложены теоретические предпосылки, описание лабораторных работ, задания и примеры выполнения работ по дисциплине «Спецглавы теории управления мехатронными системами».

Методические указания предназначены для студентов направления 15.03.06 «Мехатроника и робототехника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать 21.02.20. Формат 60x84 1/16
Усл.печ.л. 4,5. Уч.-изд.л. 4,1 Тираж 100 экз. Заказ 135 Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94

Содержание

Планируемые результаты обучения по дисциплине «Спецглавы теории управления мехатронными системами»

Целью дисциплины является формирование профессиональных Целью дисциплины является формирование профессиональных знаний и навыков в области теории нечетких множеств, нечеткой логики, искусственных нейро-нечетких сетей и нечеткого управления, необходимых для дальнейшей деятельности в качестве исследователя, инженера-конструктора, инженера-робототехника и в других видах научно-исследовательской и инженерной деятельности.

Задачи дисциплины:

– изучение теории нечетких множеств;
– изучение операций над нечеткими множествами;
– изучение и исследование искуственных нейро-нечетких сетей;
– изучение и исследование нечеткого логического вывода на основе алгоритмов Мамдани, Ларсена, Цукамото, Сугено;
– изучение особенностей практического применения алгоритмов нечеткого логического вывода (в системах нечеткого управления, в том числе интеллектуального);
– изучение основ проектирования и кодирования программного обеспечения систем нечеткого управления, в том числе интеллектуального.

Процесс изучения дисциплины «Спецглавы теории управления мехатронными системами» направлен на формирование следующих компетенций:

ПК-1 способностью составлять математические модели мехатронных и робототехнических систем, их подсистем и отдельных элементов и модулей, включая информационные, электромеханические, гидравлические, электрогидравлические, электронные устройства и средства вычислительной техники

ПК-11 способностью производить расчеты и проектирование отдельных устройств и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием

Лабораторная работа№1

Разработка системы нечеткого логического вывода, реализующей подход Мамдани

Цель работы

Исследование алгоритма нечеткого вывода Мамдани, качественная оценка результатов аппроксимации строго детерминированной зависимости между аргументами и значением функции нечетким логическим выводом по набору лингвистических правил.

Теоретическая часть

В алгоритме Мамдани используется минимаксная композиция нечетких множеств. Алгоритм включает в себя следующую последовательность этапов:

1. фаззификацию (приведение к нечеткости);
2. нечеткий вывод;
3. дефаззификацию (приведение к четкости).

Сущность процедуры фаззификации заключается в определении степени истинности антецедентов, т.е. значения функций принадлежности для антецедентов каждого правила. Для базы с m правилами и n посылками в антецеденте степень истинности обозначается как $\mu_{ik}(x_k), i = 1,..., \mathrm{m}, \mathrm{k} = 1,..., \mathrm{n}.$

Формирование логического решения осуществляется следующим образом.

Изначально определяются уровни «отсечения» для антецедента каждого из правил:

$$alfa_i = \min_i(\mu_{ik}(x_k)) \quad (1)$$

Далее находятся «усеченные» функции принадлежности консеквента:

$$\mu_i^{'}(y) = \min_i(alfa_i, \mu_i(y)) \quad (2)$$

Затем осуществляется композиция полученных усеченных функций, для чего используется максимальная композиция нечетких множеств:

$$\mu(y) = \max_i(\mu_i^{'}(y)), \tag{3}$$

где $\mu(y)$ – функция принадлежности итогового нечеткого множества.

Последним этапом является дефаззификация, которая заключается в приведении к четкости результата композиции множеств на предыдущем этапе. В алгоритме Мамдани применяется метод центра тяжести, или центроидный метод:

$$y^* = \frac{\int_y y \cdot \mu(y)dy}{\int_y \mu(y)dy} \tag{4}$$

На рис. 1.1 представлена графическая интерпретация процесса нечеткого вывода по Мамдани для двух входных переменных $X_1^{'}$ и $X_2^{'}$ и двух нечетких правил R1 и R2.

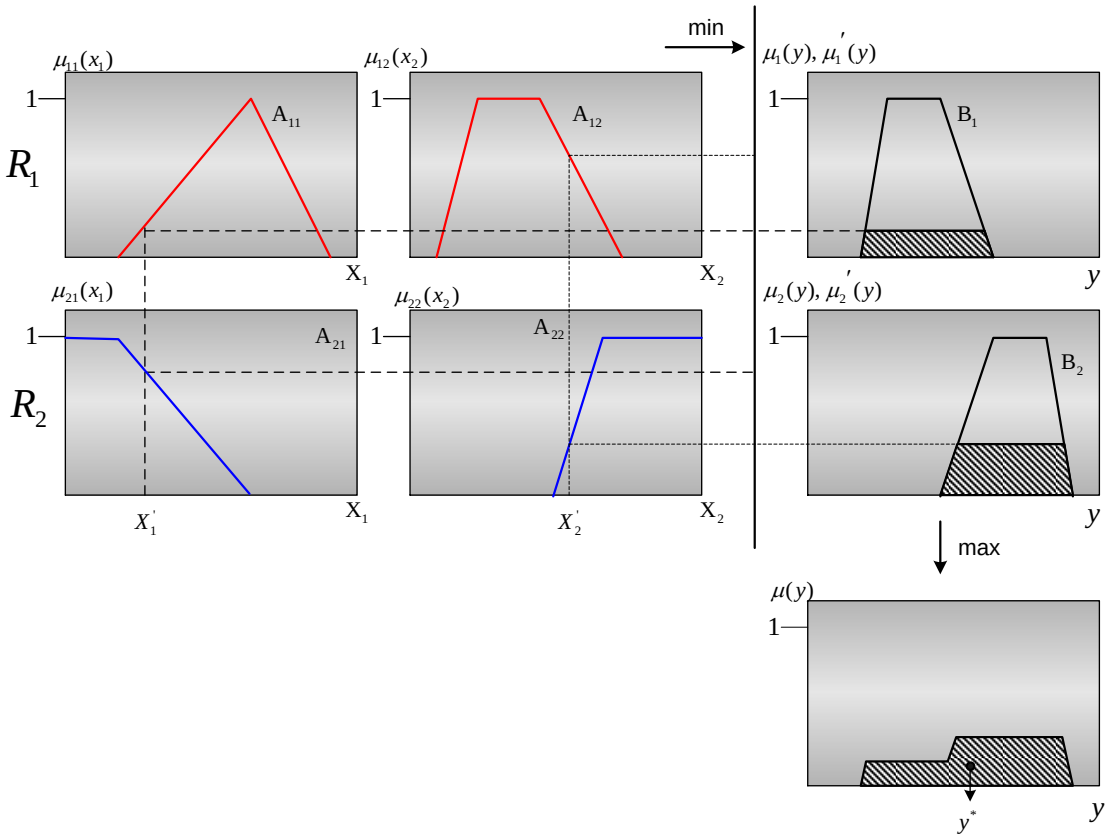

Рисунок 1.1– Графическая интерпретация процесса нечеткого вывода по Мамдани

Также существуют другие алгоритмы нечеткого вывода (Сугено, Ларсена, Цукамото). От алгоритма Мамдани они отличаются:

1. видом используемых правил;
2. типом логических операций;
3. методом дефаззификации.

Практическая часть

Проектируемая система должна моделировать зависимость $y = x_1^2 \sin(x_2 - 1)$ в области $x_1 \in [-7,3]$, $x_2 \in [-4.4,1.7]$. Проектирование системы необходимо осуществить на основе трехмерного изображения указанной зависимости (рис. 1.2), которое построено следующей программой:

```
n=15;
x1=linspace(-7,3,n); x2=linspace(-4.4,1.7,n);
y=zeros(n,n);
for j=1:n
y(j,:)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y);
xlabel('x_1'); ylabel('x_2'); zlabel('y');
```

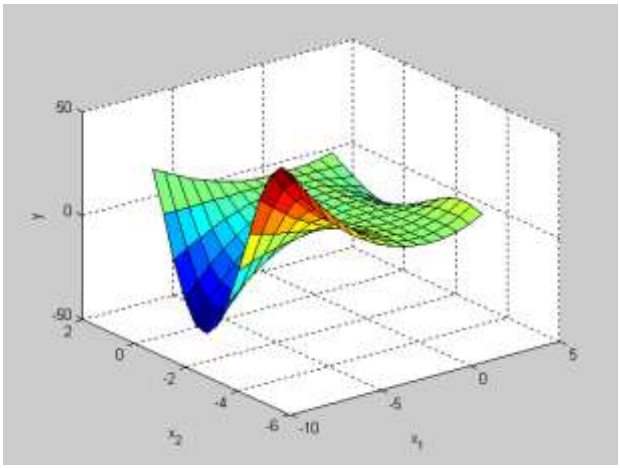

Рисунок 1.2 – График функции $y = x_1^2 \sin(x_2 - 1)$

Поверхности на рис. 2 ставится в соответствие следующие семь нечетких правил:

1. ЕСЛИ x_1 = «Низкий» И x_2 = «Низкий», ТО y = «Высокий»;
2. ЕСЛИ x_1 = «Низкий» И x_2 = «Средний», ТО y = «Низкий»;
3. ЕСЛИ x_1 = «Низкий» И x_2 = «Высокий», ТО y = «Высокий»;
4. ЕСЛИ x_1 = «Средний», ТО y = «Средний»;
5. ЕСЛИ x_1 = «Высокий» И x_2 = «Низкий», ТО y = «Выше среднего»;

6. ЕСЛИ x_1 = «Высокий» И x_2 = «Средний», ТО y = «Ниже среднего»;
7. ЕСЛИ x_1 = «Высокий» И x_2 = «Высокий», ТО y = «Выше среднего».

Проектирование нечеткой системы состоит в выполнении следующей последовательности шагов.

1. Открыть FIS-редактор, напечатав слово fuzzy в командной строке. После этого появится новое окно, показанное на рис. 1.3.

2. Добавить вторую входную переменную (Edit->Add Variable->Input).

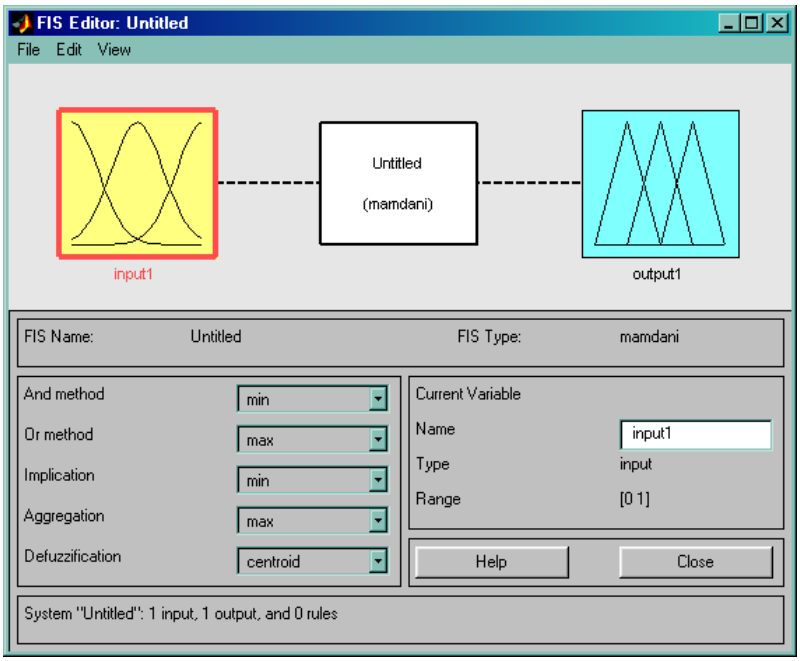

Рисунок 1.3 – Окно FIS-редактора

3. Переименовать входные и выходную лингвистические переменные в поле Name FIS-редактора.

4. Сохранить проект (File->Export->To Disk).

5. Перейти в редактор функций принадлежности двойным щелчком левой кнопки мыши на блоке x_1.

6. Задать диапазон изменения переменной x_1, напечатав -7 3 в поле Range и нажав Enter.

7. Задать функции принадлежности термов лингвистической переменной x_1. Для лингвистической оценки этой переменной предлагается использовать три терма с треугольными функциями принадлежности. Задать наименования термов переменной («Низкий»,

«Средний», «Высокий»), выделив их функции принадлежности и отредактировав поле Name (рис. 1.4.).

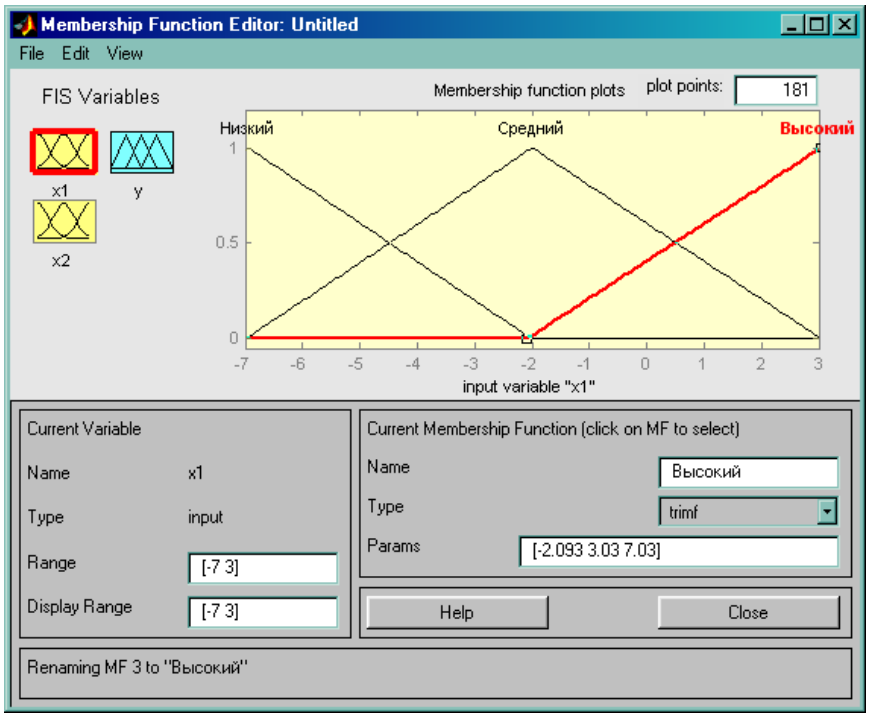

Рисунок 1.4 – Окно редактора функций принадлежности

8. Перейти в редакторе функций принадлежности к переменной x_2. Проделать шаги 6-7 для переменной x_2. Единственным отличием здесь будет диапазон.

9. Перейти в редакторе функций принадлежности к блоку y. Для лингвистической оценки этой переменной предлагается использовать пять термов с гауссовыми функциями принадлежности.

10. Задать диапазон изменения переменной y, напечатав -50 50 в поле Range и нажав Enter.

11. Удалить функции принадлежности, установленные по умолчанию, выполнив Edit->Remove All MFs.

12. Добавить гауссовы функции принадлежности, выполнив Edit->Add MFs и выбрав в появившемся диалоговом окне MF type = gaussmf, Number of MF's = 5. Задать следующие наименования термов переменной y: «Низкий», «Ниже среднего», «Средний», «Выше среднего», «Высокий».

13. Перейти в редактор базы знаний, выполнив Edit->Rules.

14. Для ввода правила необходимо выбрать в меню соответствующую комбинацию термов и нажать Add rule. Окно редактора

правил представлено на рис. 1.5. В конце каждого правила в скобках указан весовой коэффициент.

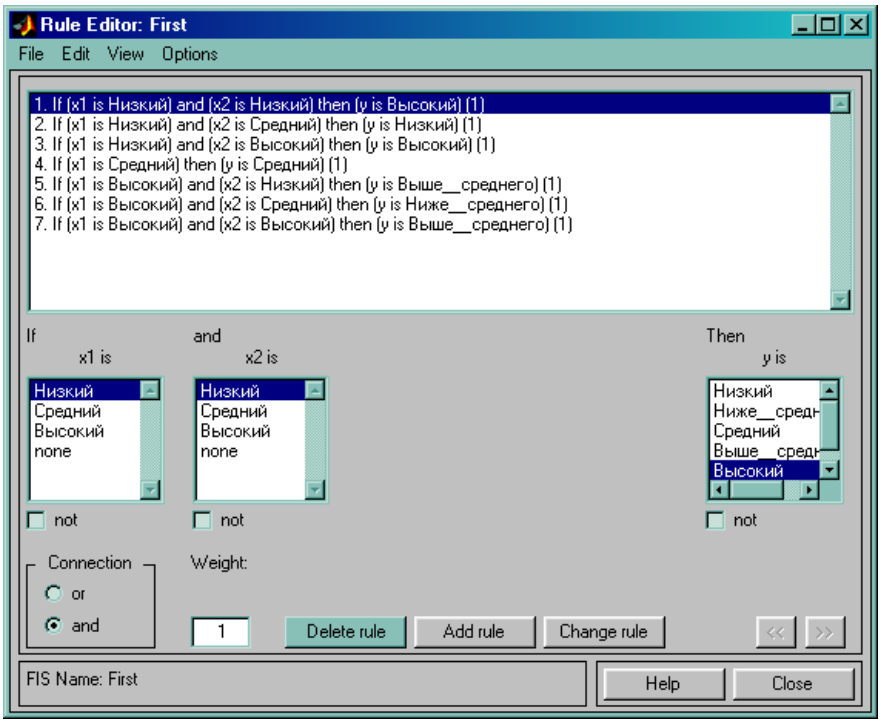

Рисунок 1.5 – Окно редактора правил

15. Сохранить проект (File->Export->To Disk).

16. Выполнить View->Rules. Появиться окно визуализации нечеткого вывода (рис. 6).

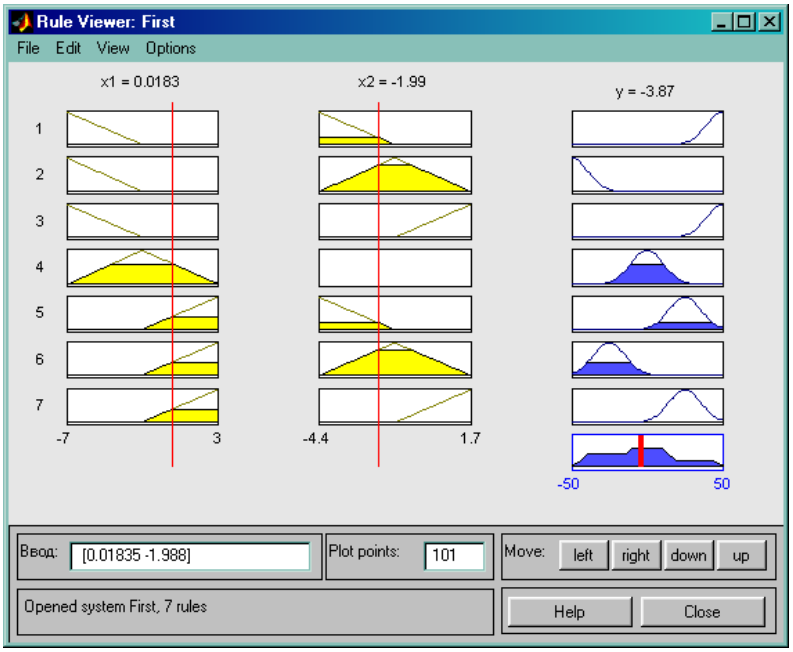

Рисунок 1.6 – Окно визуализации нечеткого вывода Мамдани

17. Выполнить View->Surface. Появится окно с изображением поверхности «входы – выход», соответствующей синтезированной нечеткой системе (рис. 1.7).

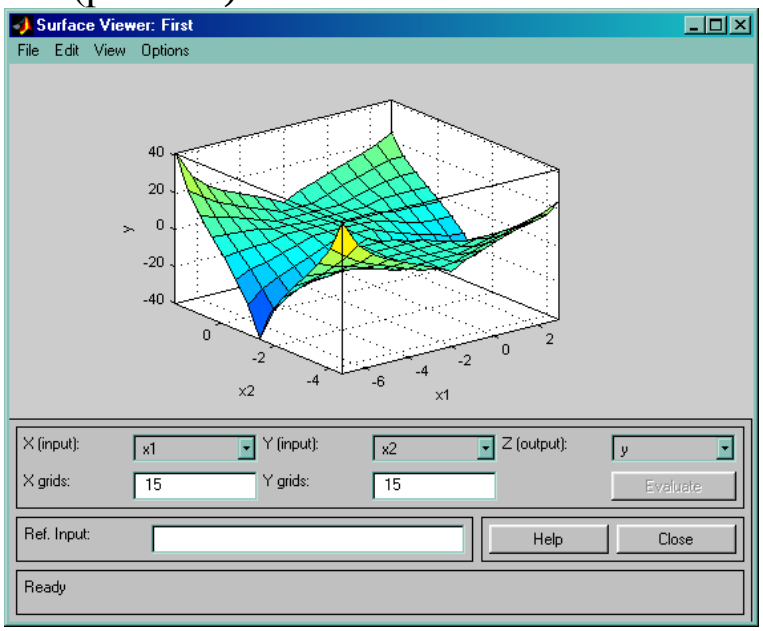

Рисунок 1.7 – Окно просмотра поверхности

Задание на лабораторную работу

1. Изучить материал теоретической части.
2. Синтезировать систему нечеткого логического вывода для функции, предложенной в практической части.
3. Построить трехмерное изображение функции, использую программный код, представленный в первом пункте практической части.
4. Сравнить полученную поверхность с поверхностью «входы – выход», которая соответствует синтезированной нечеткой системе. Сделать вывод.

Отчет должен содержать цель работы, задание и практическую часть, включающую:

1. скриншоты типа рис. 1.2, 1.4, 1.6, 1.7 с пояснениями;
2. качественную оценку сравнения поверхностей, полученных разными способами;
3. выводы, наличие которых определено заданием.

Лабораторная работа№2

Идентификация объекта управления на базе синтеза нечеткой нейронной сети типа ANFIS

Цель работы

Изучение структурно-функциональной организации многослойной сети прямого распространения (многослойного персептрона) и градиентного метода обучения, исследование влияния параметров обучения и структуры нейронной сети на качество идентификации.

Теоретическая часть

Искусственные нейронные сети (ИНС) – сети с конечным числом слоев из однотипных элементов – искусственных (формальных) нейронов с различными типами связей между слоями. При этом количество нейронов в слоях выбирается необходимым для обеспечения заданного качества решения задачи, а число слоев нейронов как можно меньшим с целью уменьшения времени решения задачи.

Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками живого организма, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Структурно-функциональная организация формального нейрона представлена на рис. 1.

Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости.

Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$S = \sum_{i=1}^{n} x_i \cdot w_i \,, \qquad (1)$$

где n – число входов нейрона; x_i – значение i – го входа нейрона; w_i – вес i – го синапса.

Выход нейрона есть функция его состояния

$$Y = f(S + w_0) = f(S), \qquad (2)$$

где f – некоторая функция, называемая активационной; w_0 – «нейронное смещение», вводимое для инициализации сети, – подключается к неизменяемому входу «+1» (на рис. 2.1 такой вход не показан).

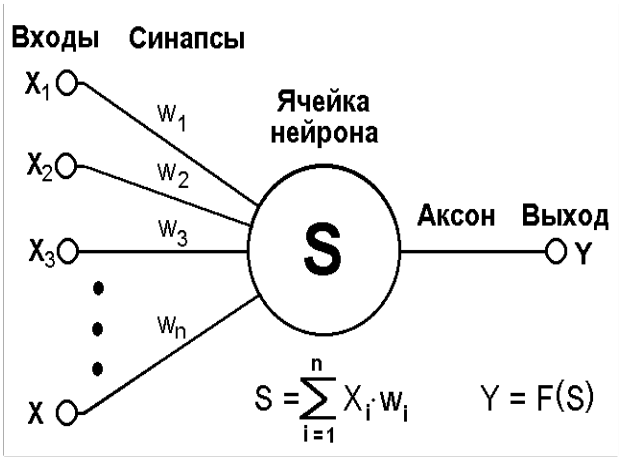

Рисунок 2.1 – Структурно-функциональная организация искусственного нейрона

Активационная функция может иметь различный вид, как показано на рис. 2.2.

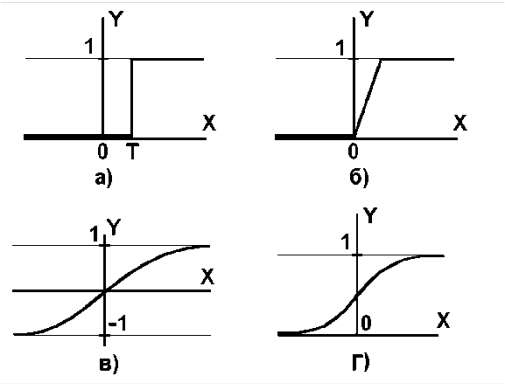

Рисунок 2.2 – а) ступенчатая функция; б) линейный порог; в) гиперболический тангенс; г) сигмоида

Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоида (т.е. функция *s*-образного вида):

$$f(x) = \frac{1}{1 + e^{-\alpha x}}. \qquad (3)$$

При уменьшении □ сигмоида становится более пологой, при □=0 вырождаясь в горизонтальную линию на уровне 0.5, при увели-

чении сигмоида приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоиды очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств логистической функции – простое выражение для ее производной:

$$f'(x) = \alpha f(x)(1 - f(x)). \quad (4)$$

Следует отметить, что логистическая функция является гладкой, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем сильные, и предотвращает насыщение от сильных сигналов, так как они соответствуют областям аргументов, где сигмоида имеет пологий наклон.

ИНС может рассматриваться как направленный граф с взвешенными дугами, в котором искусственные нейроны являются узлами. По архитектуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

В наиболее распространенном семействе сетей первого класса, называемых многослойными персептронами, нейроны расположены слоями, связанными однонаправлено. Синтез сети именно такого типа рассматривается в данной работе.

Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети.

Функциональные особенности нейронов и способ их объединения в сетевую структуру обуславливают ту или иную парадигму ИНС. Для решения задач идентификации и управления наиболее адекватными, без сомнения, являются многослойные ИНС (МНС) прямого действия или многослойные персептроны (МСП). При проектировании МНС нейроны объединяются в слои, каждый их которых обрабатывает вектор сигналов от предыдущего слоя (или входной вектор). Минимальной реализацией является двухслойная ИНС, состоящая из входного (распределительного), промежуточного (скрытого) и выходного слоя. При подсчете числа слоев входной слой обычно не учитывается, так как служит лишь для распределения входных сигналов по нейронам последующего слоя. На рис. 2.3 представлена структурная схема двухслойной ИНС прямого распро-

странения. Сигналы в сети распространяются от входа к выходу, связи между нейронами одного слоя и обратные связи отсутствуют.

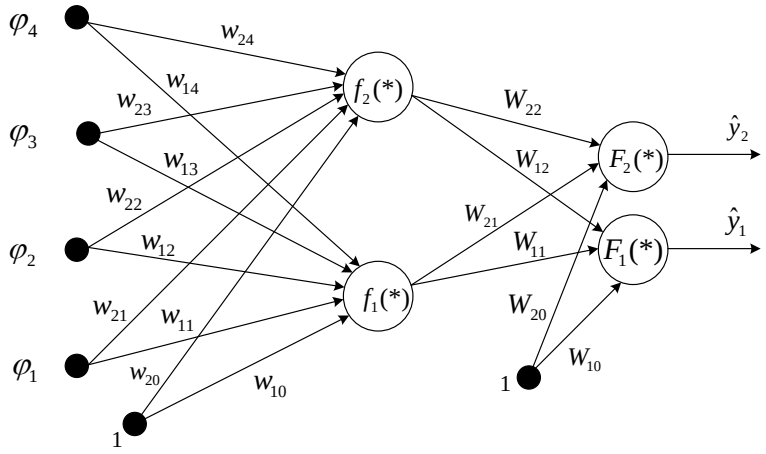

Рисунок 2.3 – Структурная схема двухслойной нейронной сети

Реализация модели двухслойной ИНС прямого распространения имеет следующее математическое представление:

$$\hat{y}_i(\boldsymbol{\theta}) = \hat{y}_i(\mathbf{w}, \mathbf{W}) = F_i(\sum_{j=1}^{n_h} W_{ij} f_j(\sum_{l=1}^{n_\varphi} w_{jl}\varphi_l + w_{j0}) + W_{i0}), \quad (5)$$

где n_φ – размерность вектора входов φ нейронной сети;

n_h – число нейронов в скрытом слое;

$\boldsymbol{\theta}$ – вектор настраиваемых параметров нейронной сети, включающий весовые коэффициенты и нейронные смещения (w_{jl}, W_{ij});

$f_j(x)$ – активационная функция нейронов скрытого слоя;

$F_i(x)$ – активационная функция нейронов выходного слоя.

Рассмотрим функционирование МНС как совокупности взаимосвязанных элементарных элементов (нейронов) с математической точки зрения. Каждый структурный элемент МНС получает на входе вектор сигналов **φ**, вычисляет его скалярное произведение на вектор весовых коэффициентов нейрона **w** и некоторую функцию F в выходной сигнал y. Результат поступает на входы других нейронов или на выход. Таким образом, нейронные сети вычисляют суперпозиции функций одного переменного и их линейные комбинации. Для обоснования возможности использования МНС в качестве моделей динамических систем нужно получить ответ на вопрос: можно ли произвольную непрерывную функцию n переменных получить с помощью

операций сложения, умножения и суперпозиции функций одного переменного?

В серии работ А.Н. Колмогорова и В.И. Арнольда решена следующая математическая проблема (составляющая существо тринадцатой проблемы Гильберта): любую непрерывную функцию n переменных можно получить с помощью операций сложения, умножения и суперпозиции из непрерывных функций одного переменного. На основе этих работ доказан ряд теорем об аппроксимации непрерывных функций многих переменных ИНС с использованием практически произвольной функции одного переменного. Помимо подтверждения общих аппроксимирующих свойств МНС необходимо получить ответы на ряд частных вопросов, касающихся структуры сети:

Сколько скрытых слоев должна содержать нейронная сеть?

Сколько нейронов должно быть включено в каждый слой?

Какой тип активационной функции должен быть выбран?

По мнению некоторых исследователей, любая непрерывная нелинейная функция может быть аппроксимирована с достаточной точностью ИНС с одним скрытым слоем, содержащим нейроны с сигмоидальными (или типа «гиперболический тангенс») функциями активации, и выходным слоем, содержащим нейроны с линейной активационной функцией. Сделаны попытки исследования влияния числа нейронов в скрытом слое на аппроксимирующие свойства сети, однако полученный результат практически невозможно применить на практике.

В настоящей работе рассматривается минимальная реализация МНС в соответствии с выражением (5) и активационными функциями типа «сигмоида» для нейронов в скрытом и выходном слоях. Возможно, репрезентативные способности МНС могут быть улучшены путем введения дополнительных скрытых слоев, особенно в случае моделирования сложных взаимосвязей. Однако усложнение структуры МНС приводит к значительным трудностям при практической реализации, обучении и последующем анализе. Это объясняет факт использования именно минимальной реализации МНС в большинстве технических приложений.

Предположим, что в результате проведения эксперимента и предварительной обработки данных получено некоторое множество

$$\mathbf{Z}^N = \left\{[u(t), y(t)], t = \overline{1, N}\right\}, \quad (6)$$

где $u(t), y(t)$ – соответственно входы и выходы системы, N – число дискретных отсчетов. Допустим также, сто выбрана некоторая модельная структура

$$y(t) = \hat{y}(t \mid \boldsymbol{\theta}) + e(t). \quad (7)$$

В соответствии с общей схемой реализации процедуры идентификации следующим этапом является оценка параметров выбранной модельной структуры. При использовании нейросетевых модельных структур этот этап представляет собой настройку весовых коэффициентов сети в результате реализации процедуры обучения на множестве примеров. Обучение представляет собой отображение множества экспериментальных данных на множество параметров нейросетевой модели

$$\mathbf{Z}^N \to \hat{\boldsymbol{\theta}} \quad (8)$$

с целью получения оптимального, в силу некоторого критерия, прогноза выходного сигнала $\hat{y}$. Традиционно используемым критерием является среднеквадратичная ошибка прогнозирования

$$V_N(\boldsymbol{\theta}, \mathbf{Z}^N) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t \mid \boldsymbol{\theta}))^2} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e^2(t, \boldsymbol{\theta})}. \quad (9)$$

Данный подход относится к классу методов ошибки прогнозирования (МОП), так как основной задачей является минимизация суммарной нормы ошибки прогнозирования. В некоторых случаях рассматриваются нормы, отличные от квадратичной, которые являются оптимальными при негауссовом распределении возмущений $e(t)$. При использовании критерия в виде (9) МОП соответствует оценке методом максимального правдоподобия при условии нормального распределения возмущений $e(t)$.

Наиболее привлекательной чертой метода является достаточно простой алгоритм оценки параметров (весовых коэффициентов) ИНС и независимость от возмущений (при условии их нормального распределения). В ряде случаев данный критерий не является абсолютно оптимальным, но в практических приложениях приводит к наилучшей модели.

В рамках данной работы для обучения ИНС применяется градиентный метод. В основе градиентного метода, или метода наиско-

рейшего спуска, лежит определение направления поиска как противоположного направлению градиента, т.е.

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \mu^{(i)} G(\boldsymbol{\theta}^{(i)}). \qquad (10)$$

Сходимость метода существенно зависит от выбора шага $\mu^{(i)}$: при достаточно малом шаге обеспечивается уменьшение критерия на каждой итерации: $V_N(\boldsymbol{\theta}^{(i+1)}, \mathbf{Z}^N) \le V_N(\boldsymbol{\theta}^{(i)}, \mathbf{Z}^N)$. Применение метода к обучению ИНС дает возможность организовать вычисления таким образом, чтобы рационально использовалась структура конкретной ИНС. В этом случае метод называется методом обратного распространения (ошибки), или обобщенным дельта-правилом.

Для выбора шага алгоритма, определяющего скорость сходимости, могут применяться различные методы, в том числе и адаптивные, хотя во многих приложениях используются методы с постоянным шагом $\mu^{(i)}$.

Независимо от выбора шага, градиентный метод может обеспечить только линейную сходимость, т.е. $\left|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{*}\right| \le c\left|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{*}\right|$, $c \in [0,1)$.

Недостаточно высокая скорость сходимости алгоритма делает невозможным применение метода для решения задач в режиме реального времени. Тем не менее, метод может быть эффективно использован в нейросетевых приложениях благодаря значительной простоте реализации, скромным требованиям к оперативной памяти и возможности использования естественной параллельности алгоритма при наличии специализированного аппаратного обеспечения.

Практическая часть

Данные, полученные в результате эксперимента с объектом управления, записаны в текстовый файл Lab_2_N.txt. Информация, содержащаяся в этом файле, должна использоваться для обучения ИНС.

Изначально необходимо открыть txt-файл (рис. 2.4).

Затем необходимо задать входные и выходные поля, а также их свойства. При этом нужно учитывать следующее.

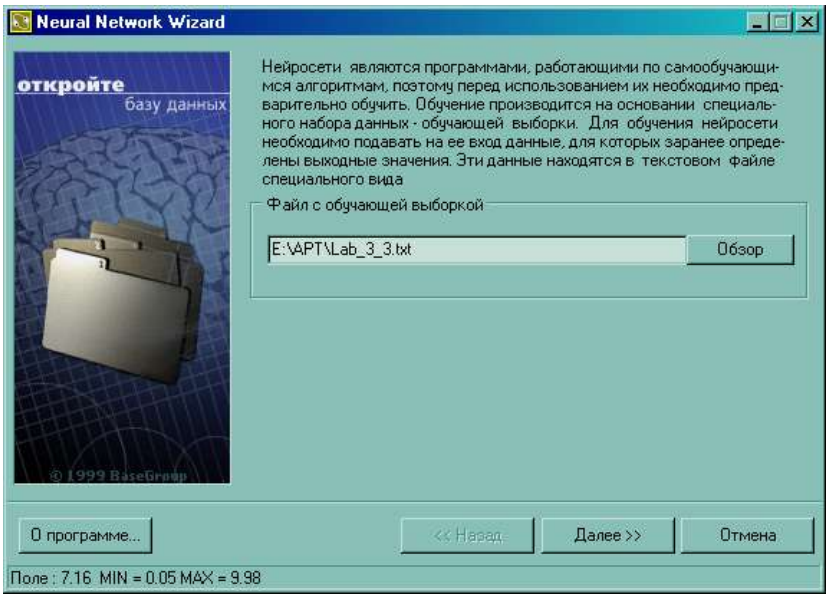

Рисунок 2.4 – Окно подключения к источнику данных

Синтезируемая ИНС состоит из входного, выходного и скрытых слоев. Количество нейронов в первом и последнем слое зависит от того, сколько полей определено как входные и выходные. Поля, отмеченные пометкой «не использовать», в обучении и тестировании ИНС применяться не будут.

На вход ИНС должна подаваться информация в нормализованном виде, т.е. числа в интервале [0,1]. Данный интервал называется интервалом допустимых значений и определяется функцией активации нейронов. Нормализация необходима для эффективного использования интервала максимальной чувствительности функции активации.

В Neural Network Wizard (NNW) реализована возможность выбора метода нормализации. Возможными вариантами являются: линейная нормализация, экспоненциальная нормализация и автонормализация, основанная на статистических характеристиках выборки. При необходимости можно изменять параметры конкретного метода нормализации. Окно настройки свойств полей представлено на рис. 2.5.

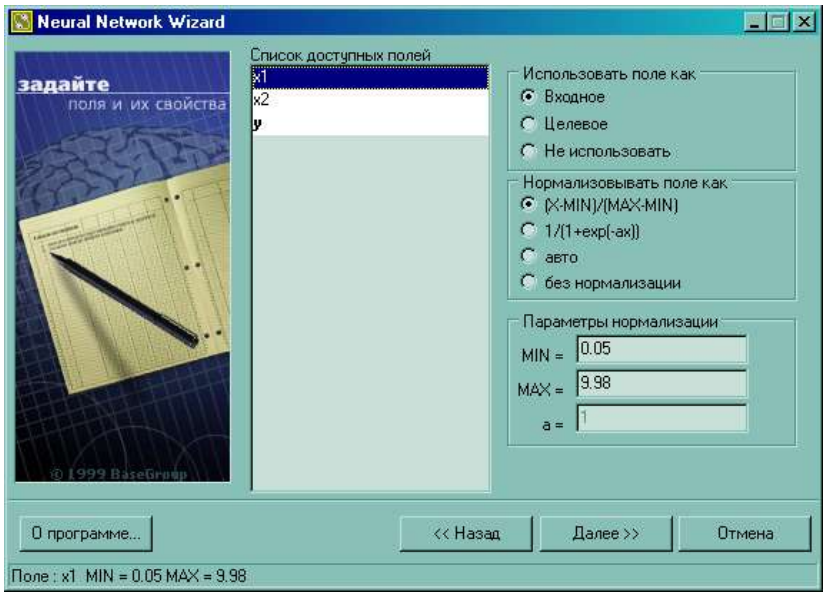

Рисунок 2.5 – Окно настройки свойств полей

Далее нужно указать количество скрытых слоев, количество нейронов в них и параметр активационной функции.

Общего правила, позволяющего определить количество скрытых слоев, нет (см. теор. часть). Обычно задается 1-3 скрытых слоев. Можно утверждать, что чем более нелинейная задача, тем больше скрытых слоев должно быть.

В NNW все элементы предыдущего слоя связаны со всеми элементами последующего. Количество нейронов в каждом скрытом слое также необходимо задать. Общих правил определения количества нейронов нет, но необходимо, чтобы число связей между нейронами было меньше количества примеров в обучающей выборке. Иначе ИНС потеряет способность к обобщению, а просто «запомнит» все примеры из обучающей выборки. Тогда при тестировании на примерах, присутствующих в обучающей выборке, ИНС будет демонстрировать прекрасные результаты, а на реальных данных – весьма плохие.

Сигмоида применяется для обеспечения нелинейного преобразования данных. В противном случае, ИНС сможет выделить только линейно разделимые множества. Чем выше параметр, тем больше активационная функция походит на пороговую. Параметр сигмоиды подбирается эмпирически. Окно настройки параметров ИНС представлено на рис. 2.6.

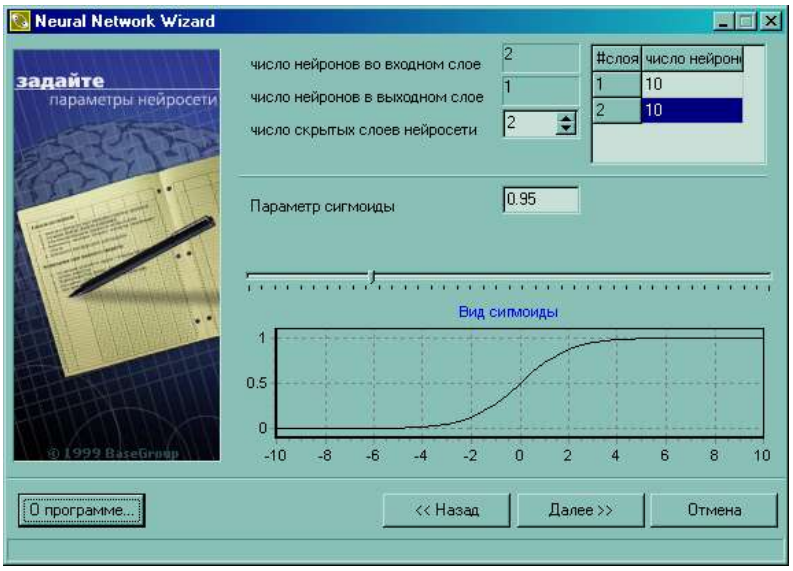

Рисунок 2.6 – Окно настройки параметров ИНС

После настройки параметров ИНС необходимо настроить параметры обучения, учитывая следующие сведения.

Все примеры, подаваемые на вход ИНС, делятся на 2 множества – обучающее и тестовое, поэтому требуется задать, сколько процентов примеров будут использоваться в обучающей выборке. Записи, используемые для тестирования, выбираются случайно, но пропорции сохраняются.

Параметр «Скорость обучения» определяет амплитуду коррекции весов на каждом шаге обучения.

Параметр «Момент» определяет степень воздействия $i-$ой коррекции весов на $i+1-$ую.

Параметр «Распознана, если ошибка по примеру <» определяет максимальную величину ошибки, при которой пример считается распознанным.

При установке флага напротив «Использовать тестовое множество как валидационное» обучение будет прекращено с выдачей сообщения, как только ошибка на тестовом множестве начнет увеличиваться. Это помогает избежать ситуации переобучения нейросети.

Критерии остановки обучения определяют момент, когда обучение будет закончено.

Окно настройки параметров обучения представлено на рис. 2.7.

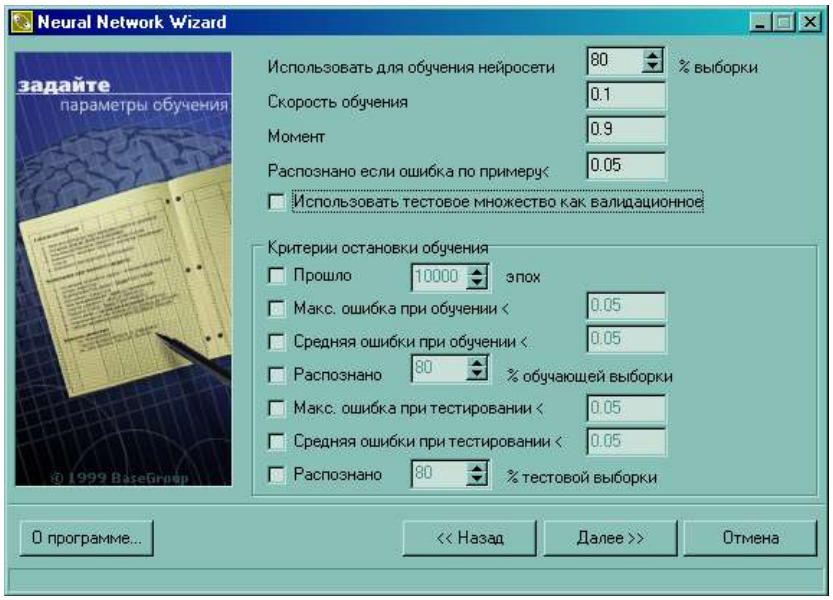

Рисунок 2.7 – Окно настройки параметров обучения

После завершения настройки параметров обучения сети необходимо запустить процесс обучение. На рис. 2.8 представлено окно запуска обучения ИНС и отображения его динамики.

В таблице над кнопкой можно наблюдать, как меняется ошибка обучения. В диаграмме отображается распределение ошибки. Зеленые столбцы – ошибка на обучающей выборке, красные – на тестовой выборке. Чем правее столбец, тем выше значение ошибки. Шкала – от 0 до 1. Чем выше столбец, тем больше примеров с указанной ошибкой.

На графиках «Распределение примеров в обучающей/тестовой выборке» можно отслеживать насколько результаты, предсказанные ИНС, совпадают со значениями в обучающей (слева) и тестовой (справа) выборках. Каждый пример обозначен на графике точкой. Если точка попадает на выделенную линию (диагональ), то ИНС предсказала результат с достаточно высокой точностью. Если точка находится выше диагонали, то ИНС недооценила его, ниже – переоценила. Необходимо добиваться, чтобы точки располагались как можно ближе к диагонали.

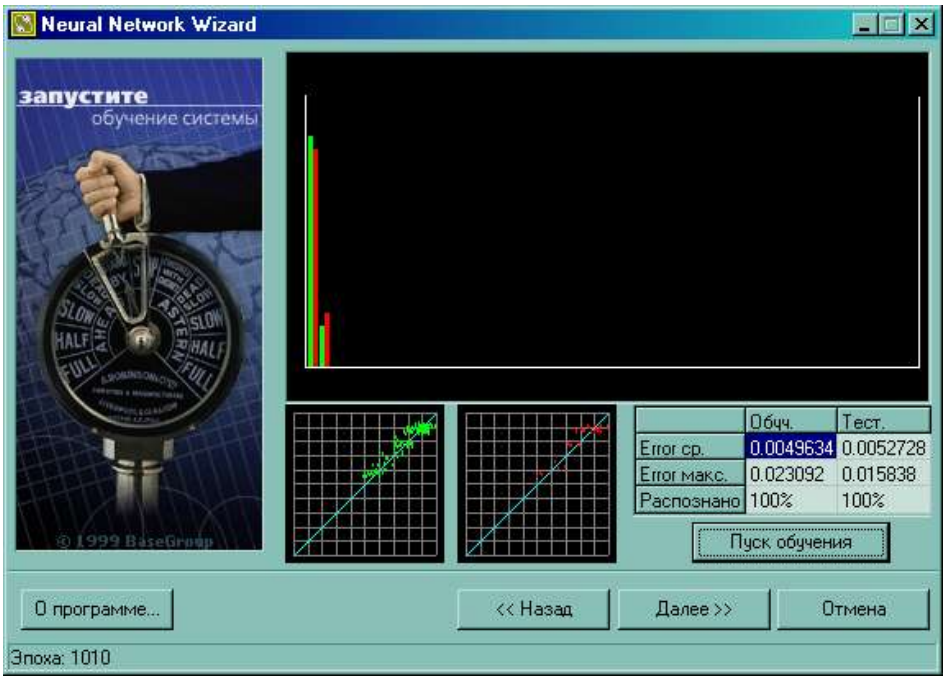

Рисунок 2.8 – Окно отображения динамики обучения ИНС

По завершению всех рассмотренных выше этапов можно использовать синтезированную ИНС как модель объекта управления.

На рис. 2.9 представлено окно, посредством которого осуществляется расчет на базе синтезированной ИНС. В наборе входных параметров нужно ввести требуемые значения и нажать на кнопку «Расчет». В таблице «Рассчитанные параметры» появится результат. Необходимо иметь в виду, что проверять ИНС на числах, выходящих за границы обучаемой и тестовой выборки, некорректно. Например, если ИНС обучена складывать числа от 0 до 10, то необходимо и тестировать ее на числах из этого диапазона.

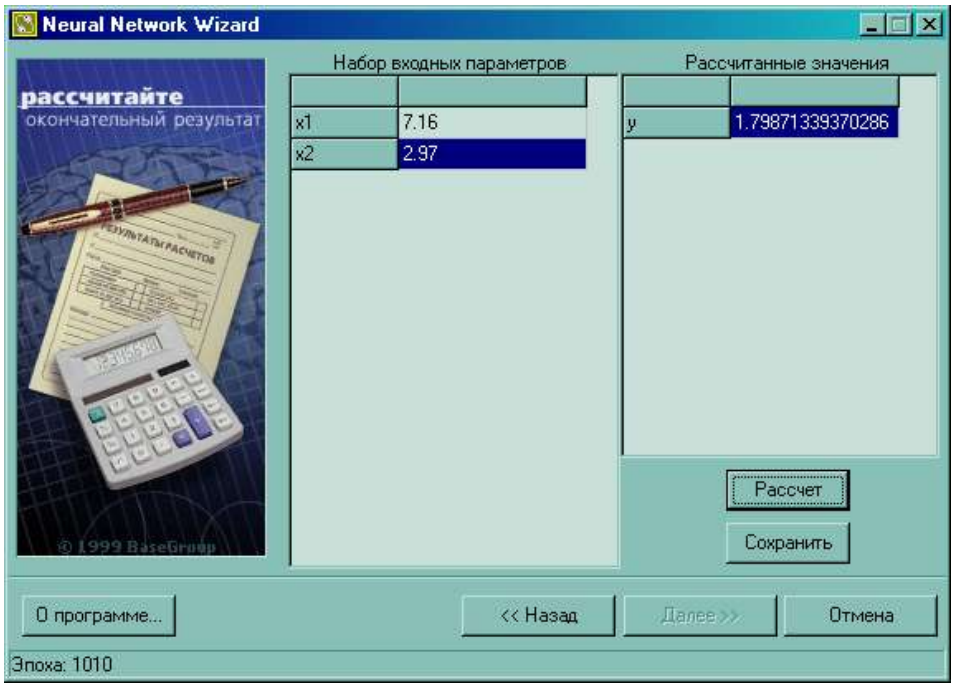

Рисунок 2.9 – Окно расчета на базе синтезированной ИНС

Задание на лабораторную работу

1. Изучить материал теоретической части.
2. Осуществить идентификацию объекта управления на базе синтеза ИНС прямого распространения, следуя указаниям из практической части (файл с экспериментальными данными прилагается к методическому материалу).
3. Изменяя параметры ИНС и обучения, определить характер влияния этих изменений на качество идентификации. Дать качественную оценку установленным зависимостям.

Отчет должен содержать цель работы, задание и практическую часть, включающую:

1. не менее 5 скриншотов типа рис. 2.8, отражающих результаты обучения при разных параметрах ИНС и обучения;
2. пояснения к скриншотам;
3. качественные оценки установленных зависимостей между параметрами ИНС (и ее обучения) и качеством идентификации;
4. выводы, наличие которых определено заданием.

Лабораторная работа№3

«Синтез САУ с применением методов вычислительного интеллекта»

Цель работы

Изучение структурно-функциональной организации многослойной сети прямого распространения (многослойного персептрона) и градиентного метода обучения, исследование влияния параметров обучения и структуры нейронной сети на качество идентификации.

Теоретическая часть

Искусственные нейронные сети (ИНС) – сети с конечным числом слоев из однотипных элементов – искусственных (формальных) нейронов с различными типами связей между слоями. При этом количество нейронов в слоях выбирается необходимым для обеспечения заданного качества решения задачи, а число слоев нейронов как можно меньшим с целью уменьшения времени решения задачи.

Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками живого организма, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Структурно-функциональная организация формального нейрона представлена на рис. 3.1.

Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости.

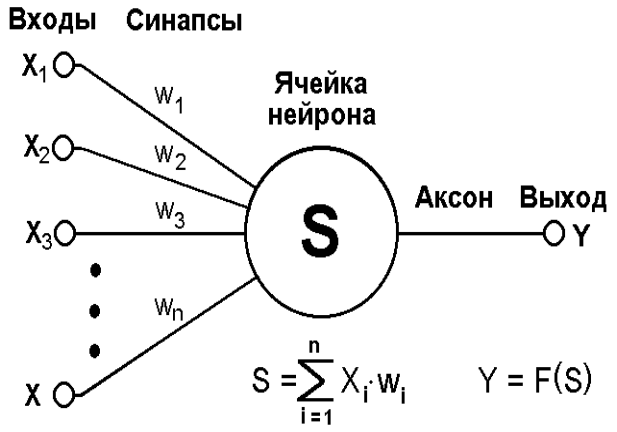

Рисунок 3.1 – Структурно-функциональная организация искусственного нейрона

Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$S = \sum_{i=1}^{n} x_i \cdot w_i , \qquad (1)$$

где n – число входов нейрона; x_i – значение i – го входа нейрона; w_i – вес i – го синапса.

Выход нейрона есть функция его состояния

$$Y = f(S + w_0) = f(S), \qquad (2)$$

где f – некоторая функция, называемая активационной; w_0 – «нейронное смещение», вводимое для инициализации сети, – подключается к неизменяемому входу «+1» (на рис. 1 такой вход не показан).

Активационная функция может иметь различный вид, как показано на рис. 3.2.

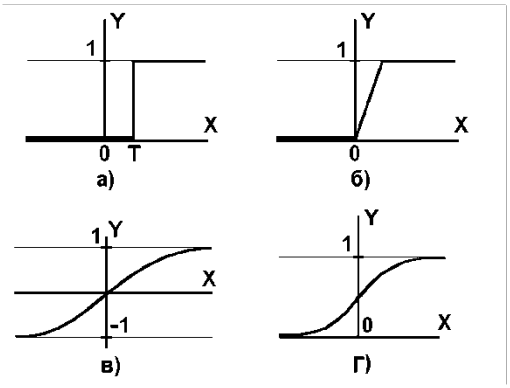

Рисунок 3.2 – а) ступенчатая функция; б) линейный порог; в) гиперболический тангенс; г) сигмоида

Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоида (т.е. функция *s*-образного вида):

$$f(x) = \frac{1}{1+e^{-\alpha x}}. \qquad (3)$$

При уменьшении сигмоида становится более пологой, при =0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении сигмоида приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоиды очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств логистической функции – простое выражение для ее производной:

$$f'(x) = \alpha f(x)(1 - f(x)). \qquad (4)$$

Следует отметить, что логистическая функция является гладкой, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем сильные, и предотвращает насыщение от сильных сигналов, так как они соответствуют областям аргументов, где сигмоида имеет пологий наклон.

ИНС может рассматриваться как направленный граф с взвешенными дугами, в котором искусственные нейроны являются узлами. По архитектуре связей ИНС могут быть сгруппированы в два класса: сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

В наиболее распространенном семействе сетей первого класса, называемых многослойными персептронами, нейроны расположены слоями, связанными однонаправлено. Синтез сети именно такого типа рассматривается в данной работе.

Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети.

Функциональные особенности нейронов и способ их объединения в сетевую структуру обуславливают ту или иную парадигму ИНС. Для решения задач идентификации и управления наиболее адекватными, без сомнения, являются многослойные ИНС (МНС) прямого действия или многослойные персептроны (МСП). При проектировании МНС нейроны объединяются в слои, каждый их кото-

рых обрабатывает вектор сигналов от предыдущего слоя (или входной вектор). Минимальной реализацией является двухслойная ИНС, состоящая из входного (распределительного), промежуточного (скрытого) и выходного слоя. При подсчете числа слоев входной слой обычно не учитывается, так как служит лишь для распределения входных сигналов по нейронам последующего слоя. На рис. 3.3 представлена структурная схема двухслойной ИНС прямого распространения. Сигналы в сети распространяются от входа к выходу, связи между нейронами одного слоя и обратные связи отсутствуют.

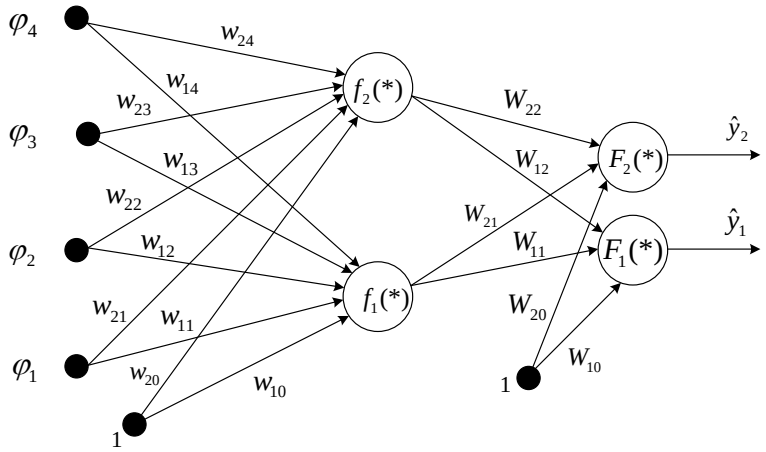

Рисунок 3.3 – Структурная схема двухслойной нейронной сети

Реализация модели двухслойной ИНС прямого распространения имеет следующее математическое представление:

$$\hat{y}_i(\boldsymbol{\theta}) = \hat{y}_i(\mathbf{w}, \mathbf{W}) = F_i\left(\sum_{j=1}^{n_h} W_{ij} f_j\left(\sum_{l=1}^{n_\varphi} w_{jl}\varphi_l + w_{j0}\right) + W_{i0}\right), \quad (5)$$

где n_φ – размерность вектора входов φ нейронной сети;

n_h – число нейронов в скрытом слое;

$\boldsymbol{\theta}$ – вектор настраиваемых параметров нейронной сети, включающий весовые коэффициенты и нейронные смещения (w_{jl}, W_{ij});

$f_j(x)$ – активационная функция нейронов скрытого слоя;

$F_i(x)$ – активационная функция нейронов выходного слоя.

Рассмотрим функционирование МНС как совокупности взаимосвязанных элементарных элементов (нейронов) с математической точки зрения. Каждый структурный элемент МНС получает на входе вектор сигналов **φ**, вычисляет его скалярное произведение на вектор весовых коэффициентов нейрона **w** и некоторую функцию F в вы-

ходной сигнал y. Результат поступает на входы других нейронов или на выход. Таким образом, нейронные сети вычисляют суперпозиции функций одного переменного и их линейные комбинации. Для обоснования возможности использования МНС в качестве моделей динамических систем нужно получить ответ на вопрос: можно ли произвольную непрерывную функцию n переменных получить с помощью операций сложения, умножения и суперпозиции функций одного переменного?

В серии работ А.Н. Колмогорова и В.И. Арнольда решена следующая математическая проблема (составляющая существо тринадцатой проблемы Гильберта): любую непрерывную функцию n переменных можно получить с помощью операций сложения, умножения и суперпозиции из непрерывных функций одного переменного. На основе этих работ доказан ряд теорем об аппроксимации непрерывных функций многих переменных ИНС с использованием практически произвольной функции одного переменного. Помимо подтверждения общих аппроксимирующих свойств МНС необходимо получить ответы на ряд частных вопросов, касающихся структуры сети:

Сколько скрытых слоев должна содержать нейронная сеть?

Сколько нейронов должно быть включено в каждый слой?

Какой тип активационной функции должен быть выбран?

По мнению некоторых исследователей, любая непрерывная нелинейная функция может быть аппроксимирована с достаточной точностью ИНС с одним скрытым слоем, содержащим нейроны с сигмоидальными (или типа «гиперболический тангенс») функциями активации, и выходным слоем, содержащим нейроны с линейной активационной функцией. Сделаны попытки исследования влияния числа нейронов в скрытом слое на аппроксимирующие свойства сети, однако полученный результат практически невозможно применить на практике.

В настоящей работе рассматривается минимальная реализация МНС в соответствии с выражением (5) и активационными функциями типа «сигмоида» для нейронов в скрытом и выходном слоях. Возможно, репрезентативные способности МНС могут быть улучшены путем введения дополнительных скрытых слоев, особенно в случае моделирования сложных взаимосвязей. Однако усложнение

структуры МНС приводит к значительным трудностям при практической реализации, обучении и последующем анализе. Это объясняет факт использования именно минимальной реализации МНС в большинстве технических приложений.

Предположим, что в результате проведения эксперимента и предварительной обработки данных получено некоторое множество

$$\mathbf{Z}^N = \left\{[u(t), y(t)], t = \overline{1, N}\right\}, \tag{6}$$

где $u(t), y(t)$ – соответственно входы и выходы системы, N – число дискретных отсчетов. Допустим также, сто выбрана некоторая модельная структура

$$y(t) = \hat{y}(t \mid \boldsymbol{\theta}) + e(t). \tag{7}$$

В соответствии с общей схемой реализации процедуры идентификации следующим этапом является оценка параметров выбранной модельной структуры. При использовании нейросетевых модельных структур этот этап представляет собой настройку весовых коэффициентов сети в результате реализации процедуры обучения на множестве примеров. Обучение представляет собой отображение множества экспериментальных данных на множество параметров нейросетевой модели

$$\mathbf{Z}^N \to \hat{\boldsymbol{\theta}} \tag{8}$$

с целью получения оптимального, в силу некоторого критерия, прогноза выходного сигнала $\hat{y}$. Традиционно используемым критерием является среднеквадратичная ошибка прогнозирования

$$V_N(\boldsymbol{\theta}, \mathbf{Z}^N) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t \mid \boldsymbol{\theta}))^2} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e^2(t, \boldsymbol{\theta})}. \tag{9}$$

Данный подход относится к классу методов ошибки прогнозирования (МОП), так как основной задачей является минимизация суммарной нормы ошибки прогнозирования. В некоторых случаях рассматриваются нормы, отличные от квадратичной, которые являются оптимальными при негауссовом распределении возмущений $e(t)$. При использовании критерия в виде (9) МОП соответствует оценке методом максимального правдоподобия при условии нормального распределения возмущений $e(t)$.

Наиболее привлекательной чертой метода является достаточно простой алгоритм оценки параметров (весовых коэффициентов) ИНС и независимость от возмущений (при условии их нормального рас-

пределения). В ряде случаев данный критерий не является абсолютно оптимальным, но в практических приложениях приводит к наилучшей модели.

В рамках данной работы для обучения ИНС применяется градиентный метод. В основе градиентного метода, или метода наискорейшего спуска, лежит определение направления поиска как противоположного направлению градиента, т.е.

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \mu^{(i)} G(\boldsymbol{\theta}^{(i)}). \qquad (10)$$

Сходимость метода существенно зависит от выбора шага $\mu^{(i)}$: при достаточно малом шаге обеспечивается уменьшение критерия на каждой итерации: $V_N(\boldsymbol{\theta}^{(i+1)}, \mathbf{Z}^N) \le V_N(\boldsymbol{\theta}^{(i)}, \mathbf{Z}^N)$. Применение метода к обучению ИНС дает возможность организовать вычисления таким образом, чтобы рационально использовалась структура конкретной ИНС. В этом случае метод называется методом обратного распространения (ошибки), или обобщенным дельта-правилом.

Для выбора шага алгоритма, определяющего скорость сходимости, могут применяться различные методы, в том числе и адаптивные, хотя во многих приложениях используются методы с постоянным шагом $\mu^{(i)}$.

Независимо от выбора шага, градиентный метод может обеспечить только линейную сходимость, т.е. $\left|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{*}\right| \le c\left|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^{*}\right|$, $c \in [0,1)$.

Недостаточно высокая скорость сходимости алгоритма делает невозможным применение метода для решения задач в режиме реального времени. Тем не менее, метод может быть эффективно использован в нейросетевых приложениях благодаря значительной простоте реализации, скромным требованиям к оперативной памяти и возможности использования естественной параллельности алгоритма при наличии специализированного аппаратного обеспечения.

Практическая часть

Данные, полученные в результате эксперимента с объектом управления, записаны в текстовый файл Lab_2_N.txt. Информация, содержащаяся в этом файле, должна использоваться для обучения ИНС.

Изначально необходимо открыть txt-файл (рис. 3.4).

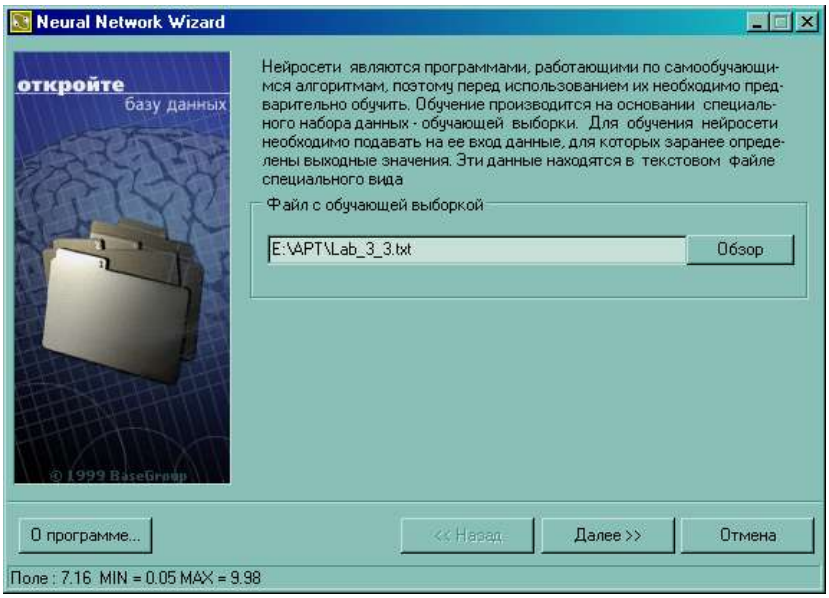

Рисунок 3.4 – Окно подключения к источнику данных

Затем необходимо задать входные и выходные поля, а также их свойства. При этом нужно учитывать следующее.

Синтезируемая ИНС состоит из входного, выходного и скрытых слоев. Количество нейронов в первом и последнем слое зависит от того, сколько полей определено как входные и выходные. Поля, отмеченные пометкой «не использовать», в обучении и тестировании ИНС применяться не будут.

На вход ИНС должна подаваться информация в нормализованном виде, т.е. числа в интервале [0,1]. Данный интервал называется интервалом допустимых значений и определяется функцией активации нейронов. Нормализация необходима для эффективного использования интервала максимальной чувствительности функции активации.

В Neural Network Wizard (NNW) реализована возможность выбора метода нормализации. Возможными вариантами являются: линейная нормализация, экспоненциальная нормализация и автонормализация, основанная на статистических характеристиках выборки. При необходимости можно изменять параметры конкретного метода нормализации. Окно настройки свойств полей представлено на рис. 3.5.

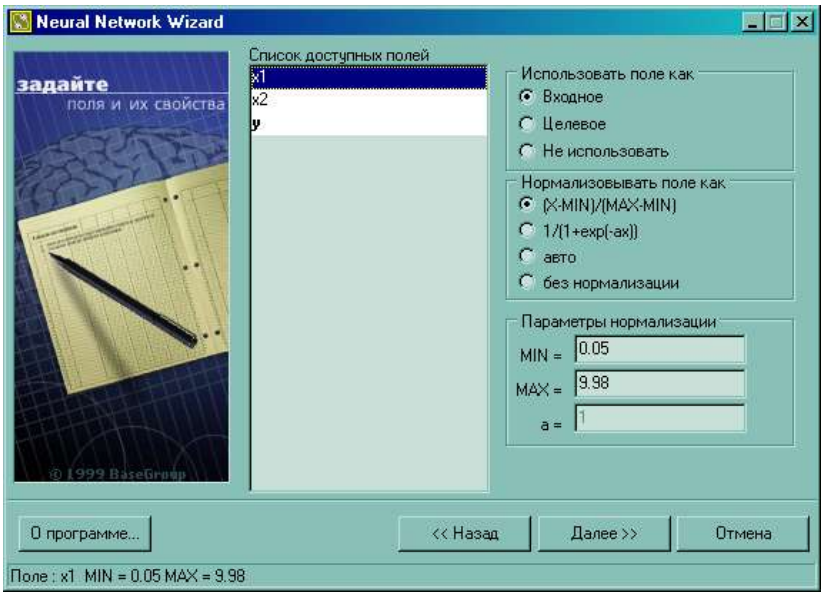

Рисунок 3.5 – Окно настройки свойств полей

Далее нужно указать количество скрытых слоев, количество нейронов в них и параметр активационной функции.

Общего правила, позволяющего определить количество скрытых слоев, нет (см. теор. часть). Обычно задается 1-3 скрытых слоев. Можно утверждать, что чем более нелинейная задача, тем больше скрытых слоев должно быть.

В NNW все элементы предыдущего слоя связаны со всеми элементами последующего. Количество нейронов в каждом скрытом слое также необходимо задать. Общих правил определения количества нейронов нет, но необходимо, чтобы число связей между нейронами было меньше количества примеров в обучающей выборке. Иначе ИНС потеряет способность к обобщению, а просто «запомнит» все примеры из обучающей выборки. Тогда при тестировании на примерах, присутствующих в обучающей выборке, ИНС будет демонстрировать прекрасные результаты, а на реальных данных – весьма плохие.

Сигмоида применяется для обеспечения нелинейного преобразования данных. В противном случае, ИНС сможет выделить только линейно разделимые множества. Чем выше параметр, тем больше активационная функция походит на пороговую. Параметр сигмоиды подбирается эмпирически. Окно настройки параметров ИНС представлено на рис. 3.6.

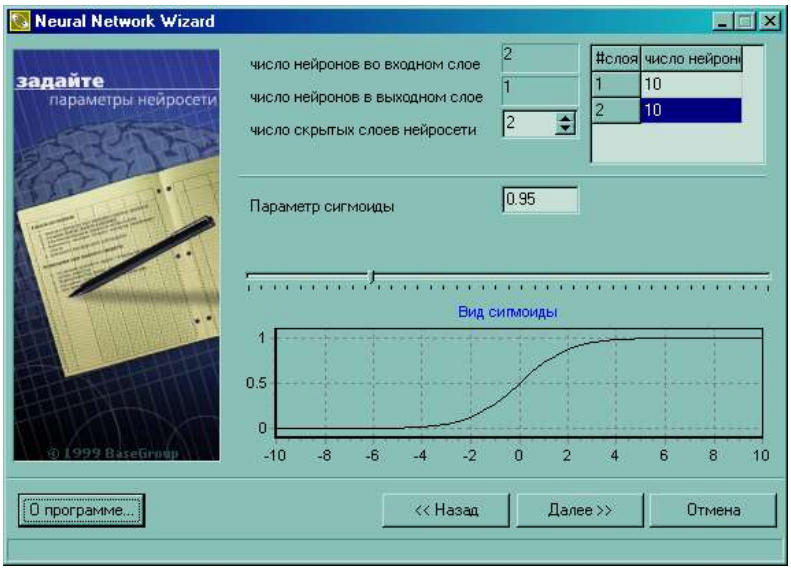

Рисунок 3.6 – Окно настройки параметров ИНС

После настройки параметров ИНС необходимо настроить параметры обучения, учитывая следующие сведения.

Все примеры, подаваемые на вход ИНС, делятся на 2 множества – обучающее и тестовое, поэтому требуется задать, сколько процентов примеров будут использоваться в обучающей выборке. Записи, используемые для тестирования, выбираются случайно, но пропорции сохраняются.

Параметр «Скорость обучения» определяет амплитуду коррекции весов на каждом шаге обучения.

Параметр «Момент» определяет степень воздействия $i-$ой коррекции весов на $i+1-$ую.

Параметр «Распознана, если ошибка по примеру <» определяет максимальную величину ошибки, при которой пример считается распознанным.

При установке флага напротив «Использовать тестовое множество как валидационное» обучение будет прекращено с выдачей сообщения, как только ошибка на тестовом множестве начнет увеличиваться. Это помогает избежать ситуации переобучения нейросети.

Критерии остановки обучения определяют момент, когда обучение будет закончено.

Окно настройки параметров обучения представлено на рис. 3.7.

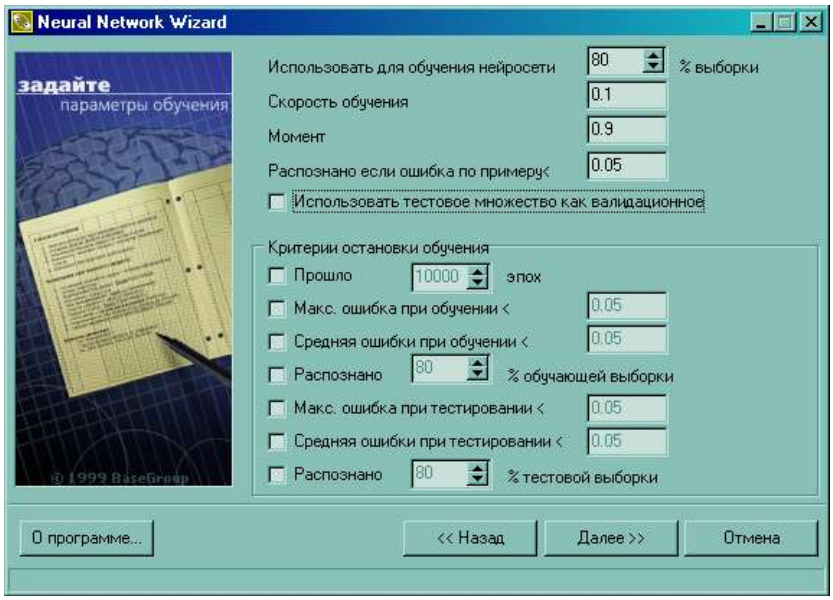

Рисунок 3.7 – Окно настройки параметров обучения

После завершения настройки параметров обучения сети необходимо запустить процесс обучение. На рис. 8 представлено окно запуска обучения ИНС и отображения его динамики.

В таблице над кнопкой можно наблюдать, как меняется ошибка обучения. В диаграмме отображается распределение ошибки. Зеленые столбцы – ошибка на обучающей выборке, красные – на тестовой выборке. Чем правее столбец, тем выше значение ошибки. Шкала – от 0 до 1. Чем выше столбец, тем больше примеров с указанной ошибкой.

На графиках «Распределение примеров в обучающей/тестовой выборке» можно отслеживать насколько результаты, предсказанные ИНС, совпадают со значениями в обучающей (слева) и тестовой (справа) выборках. Каждый пример обозначен на графике точкой. Если точка попадает на выделенную линию (диагональ), то ИНС предсказала результат с достаточно высокой точностью. Если точка находится выше диагонали, то ИНС недооценила его, ниже – переоценила. Необходимо добиваться, чтобы точки располагались как можно ближе к диагонали.

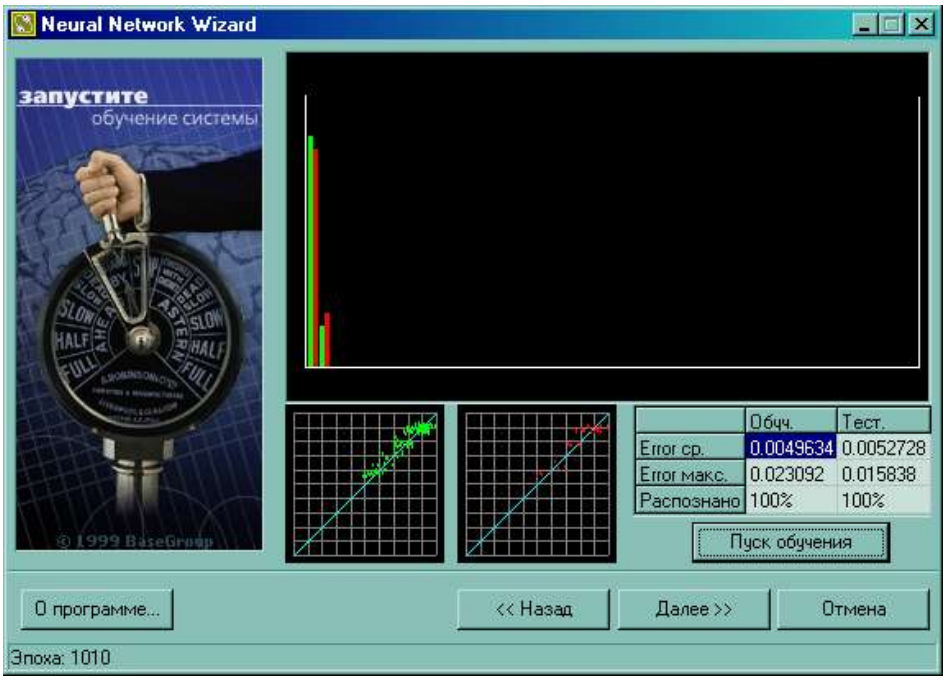

Рисунок 3.8 – Окно отображения динамики обучения ИНС

По завершению всех рассмотренных выше этапов можно использовать синтезированную ИНС как модель объекта управления.

На рис. 3.9 представлено окно, посредством которого осуществляется расчет на базе синтезированной ИНС. В наборе входных параметров нужно ввести требуемые значения и нажать на кнопку «Расчет». В таблице «Рассчитанные параметры» появится результат. Необходимо иметь в виду, что проверять ИНС на числах, выходящих за границы обучаемой и тестовой выборки, некорректно. Например, если ИНС обучена складывать числа от 0 до 10, то необходимо и тестировать ее на числах из этого диапазона.

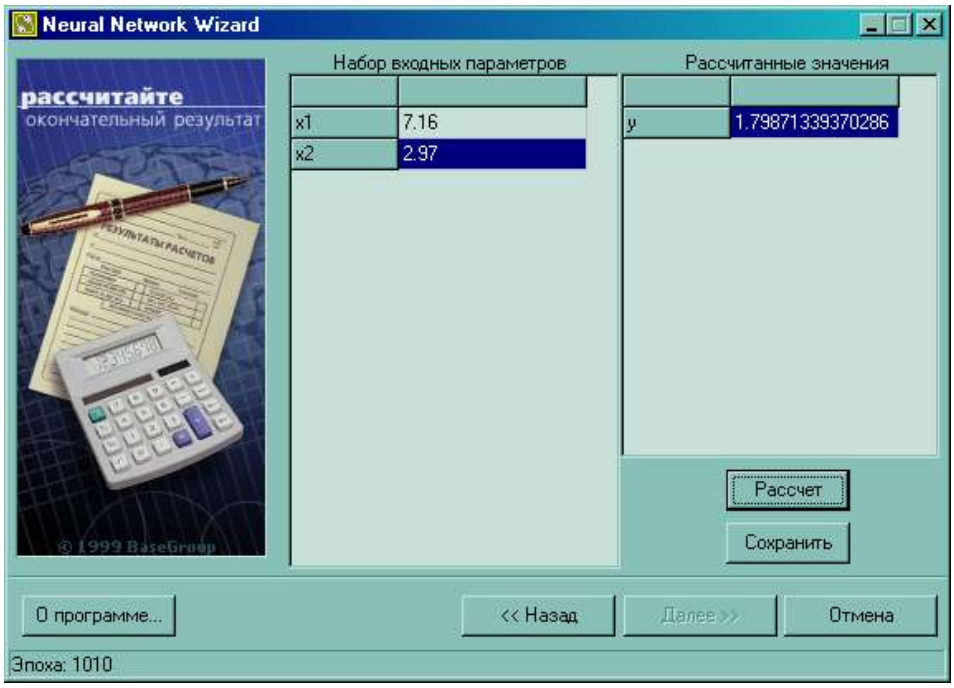

Рисунок 3.9 – Окно расчета на базе синтезированной ИНС

Задание на лабораторную работу

4. Изучить материал теоретической части.

5. Осуществить идентификацию объекта управления на базе синтеза ИНС прямого распространения, следуя указаниям из практической части (файл с экспериментальными данными прилагается к методическому материалу).

6. Изменяя параметры ИНС и обучения, определить характер влияния этих изменений на качество идентификации. Дать качественную оценку установленным зависимостям.

Отчет должен содержать цель работы, задание и практическую часть, включающую:

5. не менее 5 скриншотов типа рис. 8, отражающих результаты обучения при разных параметрах ИНС и обучения;

6. пояснения к скриншотам;

7. качественные оценки установленных зависимостей между параметрами ИНС (и ее обучения) и качеством идентификации;

8. выводы, наличие которых определено заданием.

Лабораторная работа№4

Разработка программного обеспечения нечеткого регулятора

Цель работы

Изучение генетического алгоритма, модифицированного реализацией в нем регулярного поиска локальных экстремумов на базе метода деформируемого многогранника; исследование влияния параметров инициализации алгоритма на скорость и вероятность получения требуемого решения.

Теоретическая часть

Генетические алгоритмы (ГА) применяются для решения многих конкретных научных и технических проблем, но самое популярное приложение генетических алгоритмов – оптимизация многопараметрических функций. Эффективность ГА определяется их способностью манипулировать одновременно многими параметрами, что является крайне необходимым в решении многих прикладных задач, включая проектирование интегральных схем, настройку параметров алгоритмов, поиск решений нелинейных дифференциальных уравнений и т. д.

Эффективность применения ГА зависит от характеристик задачи. Если пространство поиска в задаче небольшое, то решение можно найти методом полного перебора и применение ГА не требуется. Если пространство гладкое и унимодальное, то любой градиентный алгоритм будет эффективнее ГА. Если существует дополнительная информация о пространстве поиска, то методы поиска, использующие эвристики, часто превосходят ГА. Область применения ГА – задачи с большим пространством поиска решений и отсутствием эвристической информации.

Кроме того, эффективность ГА зависит от характеристик метода кодировки решений, выбора генетических операторов и значений инициализирующих параметров.

Оценим возможности применения ГА для решения задач оптимизации параметров элементов систем автоматического управления (САУ).

Опыт многих исследователей алгоритмов управления показал, что для простых одноконтурных САУ с линейными регуляторами задачи оптимизации, как правило, являются одноэкстремальными. Однако для сложных многоконтурных САУ и систем управления с нейроконтроллерами характерно наряду с глобальным наличие большого числа локальных экстремумов. Кроме того, локальные экстремумы появляются и при введении ограничений на пространство поиска.

Для решения одноэкстремальных задач оптимизации существует достаточное число градиентных и численных методов. Одним из них является метод деформируемого многогранника Нелдера-Мида. При оптимизации одноконтурной САУ с ПИ-регулятором применение метода позволяет устойчиво находить оптимальные значения настроечных параметров K_p и K_i для функции цели вида

$$F(K_p, K_i, \psi, t) = I_{m,y}(1 + a|\psi_0 - \psi|) \to \min, \qquad (1)$$

где

$$I_{m,y} = \int_0^{t_p} |y(t)| dt \qquad (2)$$

– интеграл по модулю регулируемой величины $y(t)$ на интервале времени переходного процесса t_∂; ψ_0, ψ – соответственно, заданная и текущая степень затухания переходного процесса регулирования; a – масштабный коэффициент, учитывающий вес штрафной функции.

Применение метода деформируемого многогранника для оптимизации двухконтурной САУ с дифференциатором и САУ с нейроконтроллерами различной структуры приводит к неоднозначности решения. В каждом случае результаты зависят от выбранных начальных координат. Из этого следует вывод о многоэкстремальности подобных задач, и для их решения требуются методы глобальной оптимизации.

В настоящее время наиболее предпочтительными методами многоэкстремальной оптимизации являются ГА, реализующие постулаты теории эволюции и опыта селекции растений и животных. Стратегия поиска оптимального решения в ГА опирается на гипотезу селекции: чем выше приспособленность особи, тем выше вероятность того, что у потомков, полученных с ее участием, признаки, определяющие приспособленность, будут выражены еще сильнее.

Если принять, что каждая особь популяции является точкой в координатном пространстве оптимизационной задачи $X_i[x_{1i}, x_{2i}, ..., x_{li}]$, а приспособленность особи – соответствующим значением функции цели $f(X_i)$, то популяцию особей можно рассматривать как множество координатных точек в пространстве, а процесс эволюции – как движение этих точек в сторону оптимальных значений целевой функции.

Следует отметить, что классический ГА находит глобальный экстремум в вероятностном смысле. И эта вероятность прямо зависит от числа особей в популяции. При оптимизации сложных многоконтурных и многосвязных систем регулирования и аналогичных систем с нейроконтроллерами ГА (в частности, диплоидная версия ГА) с достаточно высокой вероятностью находят глобальный экстремум. Однако вычисление функции цели, как правило, требует значительных вычислительных ресурсов, что существенно сказывается на общем времени расчета решения.

Особенностью ГА является то, что ни один из генетических операторов (кроссовер, мутация, инверсия) в процессе генерирования потомков не опирается на знание локального рельефа поверхности отклика функции цели. Формирование потомков генетическими операторами происходит случайным образом, и поэтому нет гарантии, что найденные решения будут лучше родительских. Следовательно, в процессе эволюции встречается немалое число «неудачных» потомков, которые увеличивают число обращений к функции цели и, тем самым, увеличивают время поиска глобального экстремума.

Кроме того, ГА находят оптимальное решение только внутри заданного диапазона поиска. Поэтому диапазоны поиска и число особей в популяции требуется задавать с большим запасом, что также увеличивает время решения.

Перечисленные особенности сдерживают широкое применение ГА в инженерной практике. Однако потребность в таких алгоритмах для решения прикладных задач сравнительно небольшой размерности постоянно растет, особенно в связи с намечающейся тенденцией внедрения в системы управления нейросетевых технологий.

В настоящей лабораторной работе для решения задачи оптимизации предлагается использовать модифицированный ГА (МГА), применение которого обеспечивает возможность нахождения гло-

бального экстремума с требуемой вероятностью при существенно меньшем, по сравнению с диплоидным ГА, числом обращений к функции цели.

МГА содержит в себе генетические качества статистической селекции популяции поисковых точек. Для исключения «неудачных» потомков в нем реализована процедура регулярного поиска локальных экстремумов с использованием метода деформируемого многогранника. При смене поколений в алгоритме заложена рекомендуемая во многих источниках 10-процентная замена «неперспективных» особей (элиминирование).

На рис. 4.1 приведена факсимильная копия программы МГА для Mathcad, в которой реализованы процедуры статистического задания особей в популяции, сортировки и элиминирования неперспективных особей, вероятностной селекции группы особей для целей осуществления регулярного поиска локальных экстремумов, операций регулярного поиска локальных экстремумов, а также процедур завершения работы алгоритма.

Для инициализации алгоритма необходимо установить: размерность задачи оптимизации N, численность особей в популяции μ, точность решения задачи ε и значения матрицы D, определяющие границы интервалов значений координат оптимизируемых векторов популяции.

В общем случае, МГА почти в два раза меньше обращается к функции цели по сравнению с диплоидной версией ГА, что вдвое повышает оперативность получения решения. Кроме того, в процессе регулярного поиска МГА способен найти глобальный экстремум за пределами заданного диапазона. Перечисленные особенности позволяют заключить, что применение модифицированного ГА в задачах многоэкстремальной оптимизации (в частности, параметров элементов САУ) является более эффективным, чем использование его классической версии.

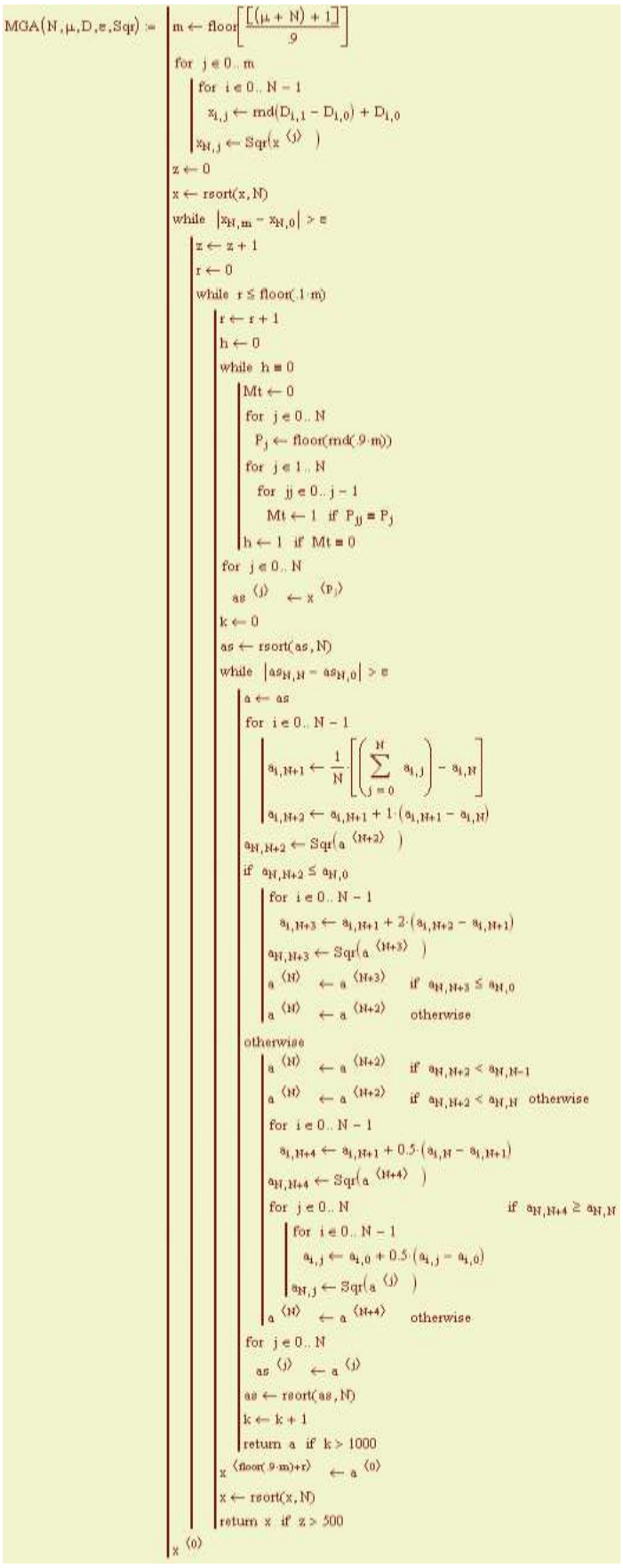

Рисунок 4.1 – Текст программы МГА для Mathcad

Практическая часть

МГА реализован в программном средстве Global Minimum Researcher (GMR). Скриншот основного окна GMR представлен на рис. 2.

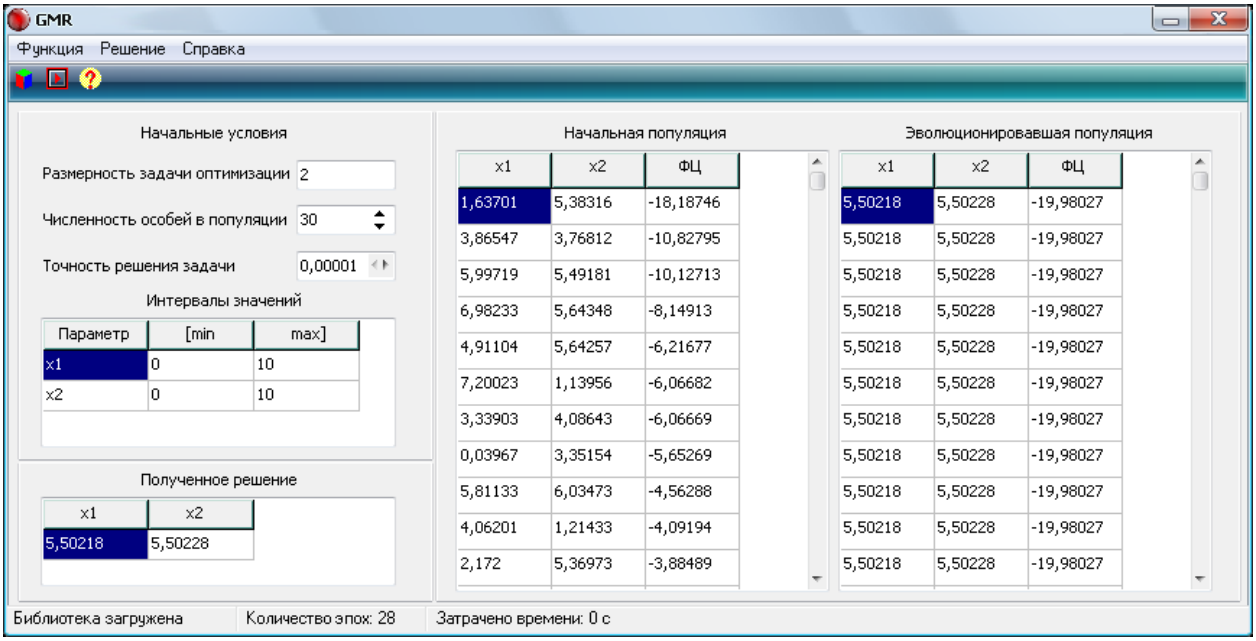

Рисунок 4.2 – Основное окно приложения «GMR»

На панели «Начальные условия» расположены:

1. окно «Размерность задачи оптимизации», которое служит для отображения числа оптимизируемых параметров;

2. окно «Численность особей в популяции», через которое вводится и отображается количество картежей типа <Решение | Значение функции цели>, обрабатываемых в процессе расчета решения;

3. окно «Точность решения задачи», необходимое для ввода и отображение величины, определяющей максимально допустимое рассогласование значений целевой функции, соответствующих полученным в результате работы программы решениям;

4. таблица «Интервалы значений», посредством которой вводятся и отображаются идентификаторы и интервалы допустимых значений оптимизируемых параметров.

Размерность задачи оптимизации определяется GMR автоматически, исходя из числа параметров функции, представленной про-

граммным блоком, скомпилированным в динамически присоединяемую библиотеку (DLL).

Численность особей в популяции (ЧОП) влияет на расчет решения неоднозначно. В случае если оптимизируемая функция является унимодальной, то ЧОП влияет только на скорость сходимости. Если оптимизируется многоэкстремальная функция, то, помимо скорости сходимости, в зависимости от ЧОП находится еще и вероятность нахождения глобального экстремума.

Точность решения задачи определяет размер окрестности глобального минимума, в которой по окончании работы программы должны находиться все точки $n-$мерного пространства задачи оптимизации, соответствующие особям эволюционировавшей популяции.

Параметр, определяющий область допустимых значений оптимизируемых переменных, влияет на расчет решения в комплексе с ЧОП и определяет границы области, в которой строится некоторая поверхность при формировании начальной популяции. При этом ЧОП определяет уровень разреженности точек поверхности. Чем больше размер области допустимых значений и количество точек, образующих в ней поверхность, тем более полная информация о характере целевой функции передается программе до непосредственного начала поиска.

Важно отметить, что программа GMR способна находить глобальный экстремум, даже в том случае, если он находится за пределами области, определенной пользователем.

На панели «Полученное решение» расположена таблица отображения оптимальных значений параметров.

На панелях «Начальная популяция» и «Эволюционировавшая популяция» расположены таблицы отображения решений с соответствующими значениями функции цели.

Рассмотрим порядок работы с приложением GMR.

Изначально необходимо создать динамически присоединяемую библиотеку, в которую должен быть скомпилирован код, представляющий функцию.

Для создания библиотеки в интегрированной среде разработки C++ Builder необходимо выполнить следующие действия.

1. Выполнить команду File | New | Other и выбрать пиктограмму Dynamic-link Library (либо DLL Wizard). Сделать в диалого-

вом окне установки, показанные на рис. 4.3. Откроется окно Редактора Кода.

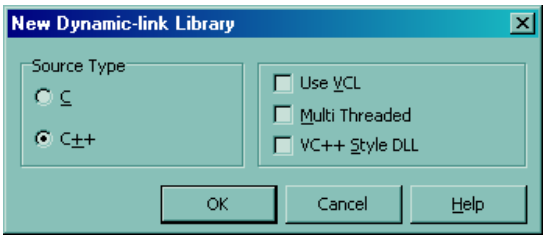

Рисунок 4.3 – Окно задания опций создания DLL

2. Удалить из текста комментарий.
3. Вставить перед описанием DllEntryPoint директиву включения головного файла:
   #include "MGA.h" (имя файла может быть другим). Файла MGA.h пока нет. Он будет создан позже. Также, при необходимости, нужно добавить в файл реализации директивы включения заголовочных файлов стандартных библиотек.
4. Вставить после описания DllEntryPoint описание функции Opt (в рассматриваемом примере операторы тела функции возвращают значение по формуле $y = x_0^2 + x_1^2$).
5. Сохранить модуль и проект.
   В результате рассмотренных действий файл MGA.cpp (имя файла может быть другим) примет следующий вид:

```
#include <windows.h>
#include <Math.h>
#include <Math.hpp>
#pragma argsused
#include "MGA.h"
int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void* lpReserved)
{
   return 1;
}
//---------------------------------------------------------------------------

double Opt (double * x)
{
```

```
double y=pow(x[0],2)+pow(x[1],2);
return y;
}
```

6. Выполнить команду File | New | Other и выбрать пиктограмму Header File. Откроется окно Редактора Кода.
7. Записать в файл следующий код:
   extern "C" double __declspec(dllexport) Opt (double * x);
8. Сохранить файл с именем MGA.h.
9. Выполнить команду Project | Build MGA. В результате будет создан файл MGA.dll.

Перед компиляцией рекомендуется выключить опции «Build with runtime packages» и «Dynamic RTL» на вкладках Project | Options | Packages и Project | Options | Linker | Linking соответственно. Это обеспечит возможность использования библиотеки в системе, где не установлены стандартные библиотеки (пакеты) C++ Builder.

Скомпилированную библиотеку нужно подключить к GMR, выполнив команду Функция | Загрузить DLL.

Затем нужно определить идентификаторы и диапазоны допустимых значений оптимизируемых параметров, вызвав двойным щелчком мыши по таблице «Интервалы значений» окно, представленное на рис. 4.4.

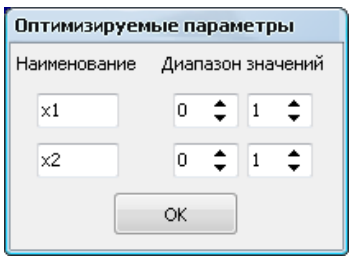

Рисунок 4.4 – Окно настройки свойств оптимизируемых параметров

После этого становится активной команда Решение | Рассчитать. Далее, настроив оставшиеся параметры, необходимо рассчитать решение.

Следует отметить, что скорость получения решения оценивается количеством итераций (эпох), обеспечивающим сходимость точек, соответствующих особям популяции, в окрестности минимума.

Задание на лабораторную работу

1. Изучить генетический алгоритм, модифицированный реализацией в нем регулярного поиска локальных экстремумов на базе метода деформируемого многогранника.

2. Создать динамически присоединяемую библиотеку, скомпилировав в нее программный код, представляющий многоэкстремальную функцию 2-4 переменных.

3. Найти глобальный минимум выбранной функции с использованием программного средства GMR.

4. Качественно оценить влияние численности особей в популяции и размеров области поиска на вероятность нахождения глобального минимума и скорость получения решения. Сделать выводы.

Отчет должен содержать цель работы, задание и практическую часть, включающую:

1. скриншот главного окна программного средства GMR, отображающего начальные условия и результат решения поставленной в рамках работы задачи;

2. листинг файла реализации *.cpp, содержащего описание целевой функции;

3. пояснения к скриншоту и листингу;

4. качественные оценки влияния параметров инициализации алгоритма на процесс и результат решения;

5. выводы, наличие которых определено заданием.

Лабораторная работа № 5

Моделирование работы нечеткого регулятора с одним входом

Краткие теоретические сведения

Нечеткий регулятор (в англоязычной литературе fuzzy controller) является элементом систем автоматического управления, использующим принципы нечеткой логики (в англоязычной литературе fuzzy logic). Нечеткий регулятор может иметь более одного входа. В рамках данной работы рассмотрим нечеткий регулятор с одним входом.

В своих работах, посвященных нечеткой логике, Лотфи Заде предложил использование логических правил для задания поведения нечеткого регулятора. Форма записи правил для нечеткого регулятора с одним входом выглядит следующим образом:

```
ЕСЛИ (условие №1) ТО (действие №1)
ЕСЛИ (условие №2) ТО (действие №2)
ЕСЛИ (условие №3) ТО (действие №3)
…
ЕСЛИ (условие №N) ТО (действие №N)
```

Составление набора правил является важной часть процесса проектирования регулятора. Рассмотрим процесс составления набора правил на примере задачи разработки регулятора для системы управления подводного робота, осуществляющего горизонтальное перемещение. Управляемым параметром является положение робота. Управление производится по ошибке – разнице между задающим воздействием и сигналом в цепи обратной связи. Общий вид схемы системы управления показан на рисунке 1.

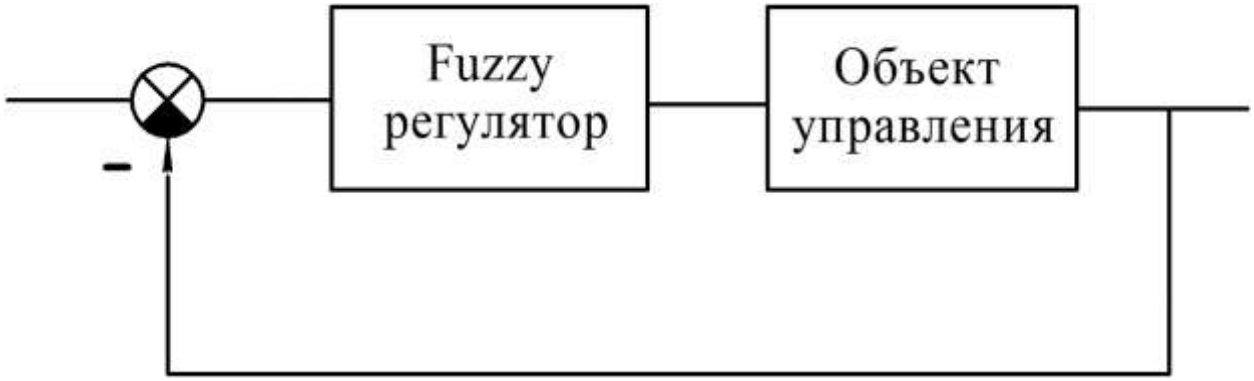

Рисунок 1 Схема системы автоматического управления

Для достижения поставленных задач будем использовать следующий набор правил:

ЕСЛИ *оставшееся расстояние – слишком малое* ТО *двигаться обратно (осуществлять реверс)*

ЕСЛИ *оставшееся расстояние – достаточное* ТО *остановиться*

ЕСЛИ *оставшееся расстояние – малое* ТО *двигаться вперед*

ЕСЛИ *оставшееся расстояние – большое* ТО *двигаться вперед*

ЕСЛИ *оставшееся расстояние – очень большое* ТО *двигаться вперед ускоренно*

Следует обратить внимание на то, что правила используют лингвистические категории, такие как «низкий», «высокий» и т.п. вместо численных значений.

При подобной настройке нечеткий регулятор будет работать следующим образом. Так называемый «четкий» сигнал поступает на вход регулятора. Этот сигнал подается на входные функции принадлежности (в англоязычной литературе «antecedent membership functions», «antecedents»), определяющие степень принадлежности сигнала к одной из лингвистических категорий. В нашем случае мы имеет три такие категории – «ошибка низкая», «ошибка средняя» и «ошибка высокая».

Вид функций принадлежности определяется при разработке регулятора. Рассмотрим некоторые виды функций принадлежности.

Треугольная функция принадлежности. Выходной сигнал функции, как очевидно из её названия, имеет вид треугольника. На рисунке 2 показаны выходные сигналы трёх треугольных функций принадлежности. На рисунке 3 показано подключение блоков Triangular MF пакета Simulink, моделирующих работу треугольных функций принадлежности.

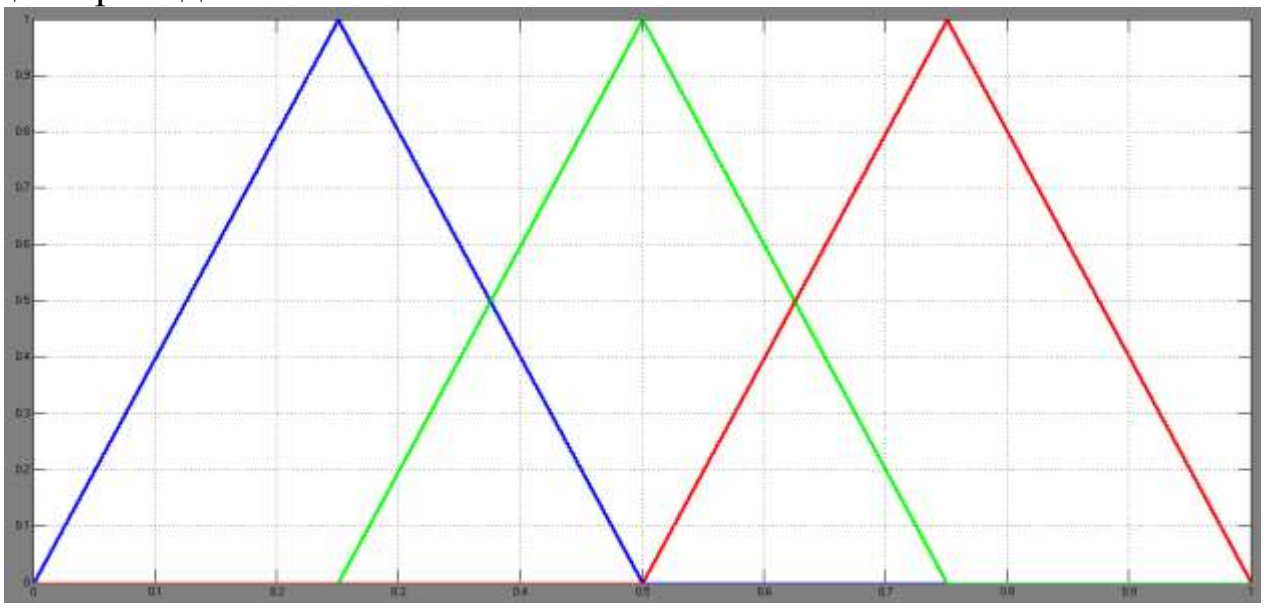

Рисунок 2 Выходной сигнал трёх треугольных функций принадлежности

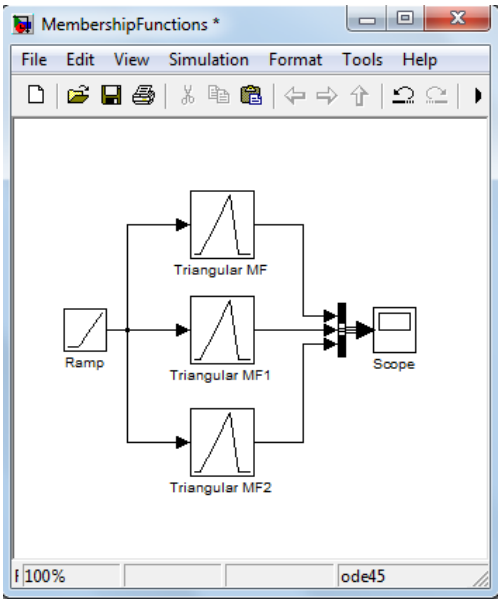

Рисунок 3 Подключение блоков Triangular MF пакета Simulink, моделирующих работу треугольных функций принадлежности

Трапецеидальная функция принадлежности. Выходной сигнал функции имеет вид трапеции. На рисунке 4 показаны выходные сигналы трёх трапецеидальных функций принадлежности. На рисунке 5 показано подключение блоков Trapezoidal MF пакета Simulink, моделирующих работу трапецеидальных функций принадлежности.

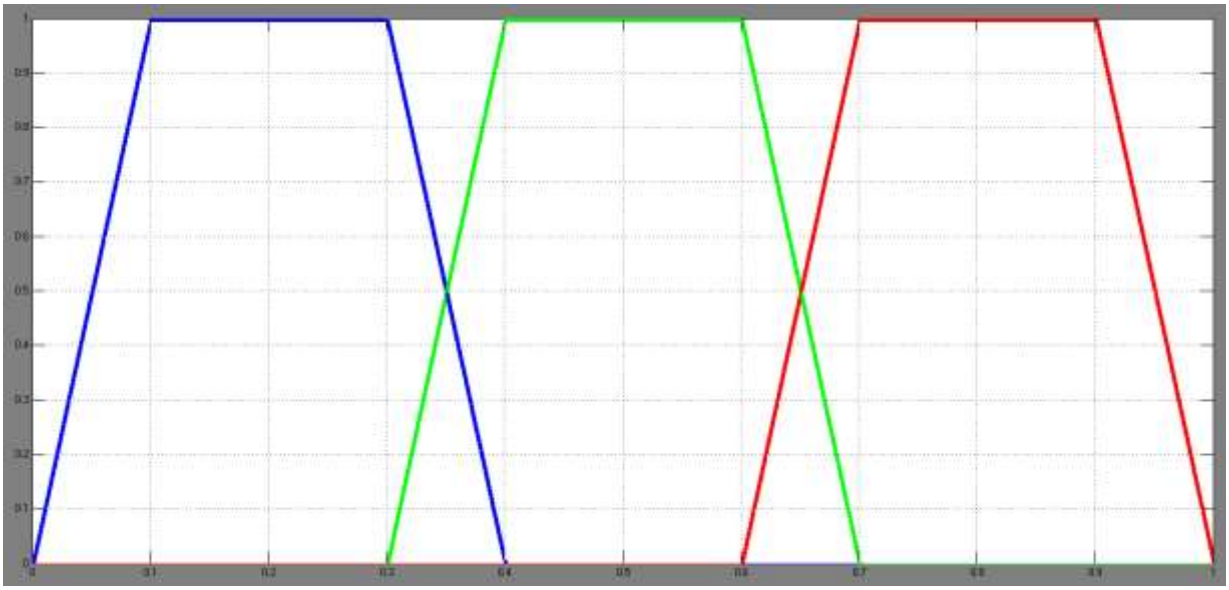

Рисунок 4 Выходной сигнал трёх трапецеидальных функций принадлежности

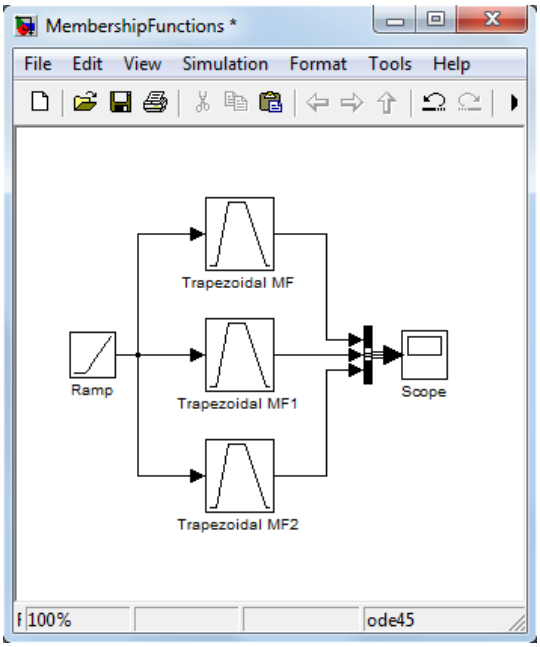

Рисунок 5 Подключение блоков Trapezoidal MF пакета Simulink, моделирующих работу трапецеидальных функций принадлежности

Гауссова функция принадлежности. Выходной сигнал функции имеет вид гауссовой кривой. На рисунке 6 показаны выходные сигналы трёх гауссовых функций принадлежности. На рисунке 7 показано подключение блоков Gaussian MF пакета Simulink, моделирующих работу гауссовых функций принадлежности.

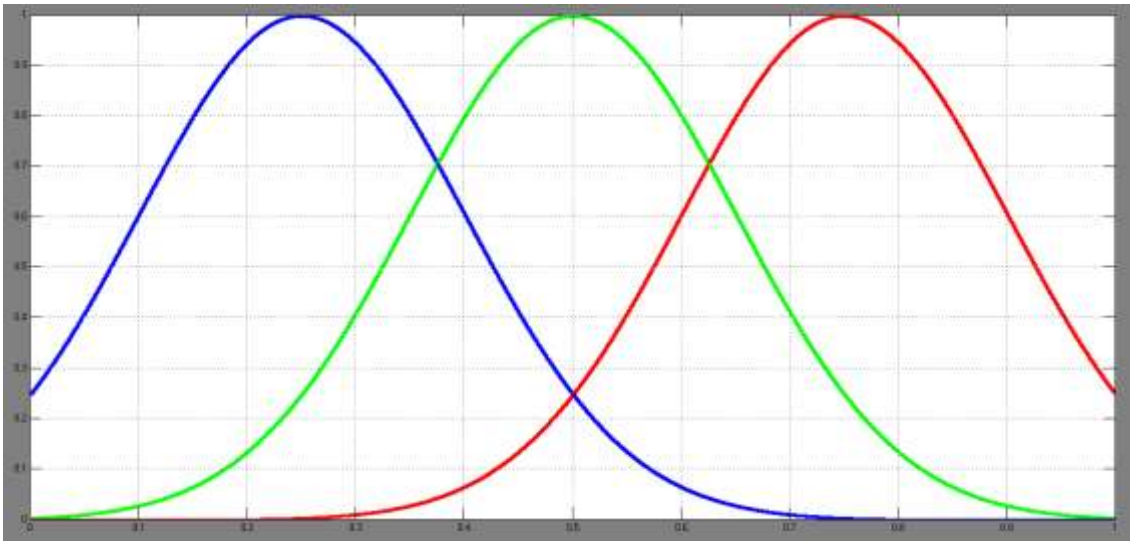

Рисунок 6 Выходной сигнал трёх гауссовых функций принадлежности

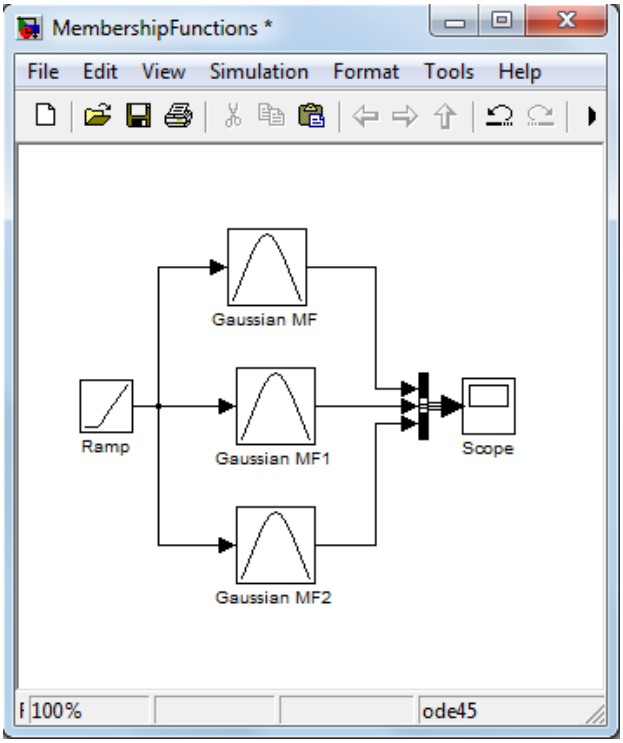

Рисунок 7 Подключение блоков Gaussian MF пакета Simulink, моделирующих работу гауссовых функций принадлежности

Сигмоидная функция принадлежности. Выходной сигнал функции имеет вид сигмоиды. На рисунке 8 показаны выходные сигналы трёх сигмоидных принадлежности. На рисунке 7 показано подключение блоков Sigmoidal MF пакета Simulink, моделирующих работу сигмоидных функций принадлежности.

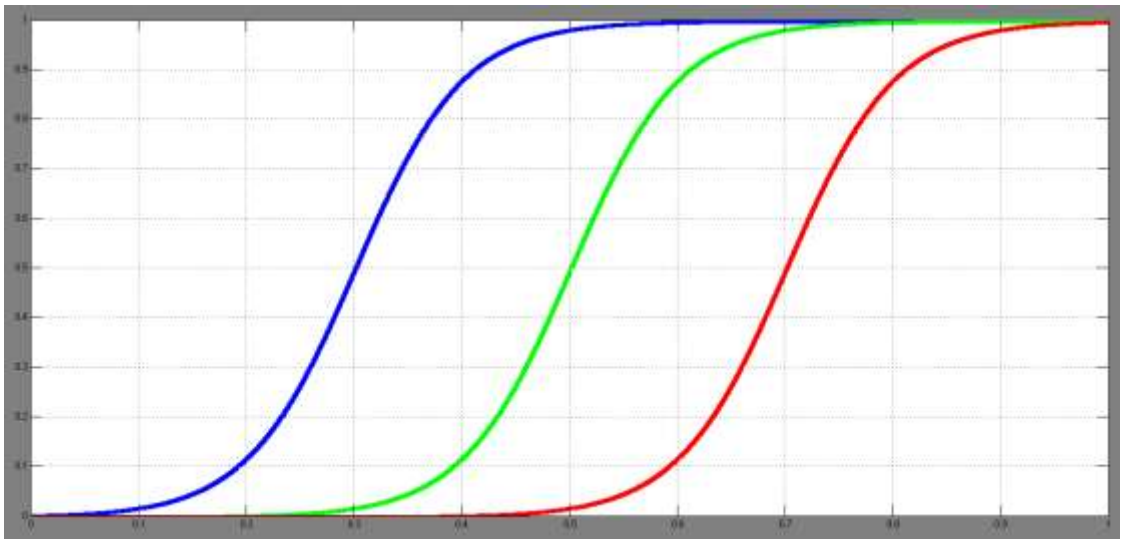

Рисунок 8 Выходной сигнал трёх сигмоидных функций принадлежности

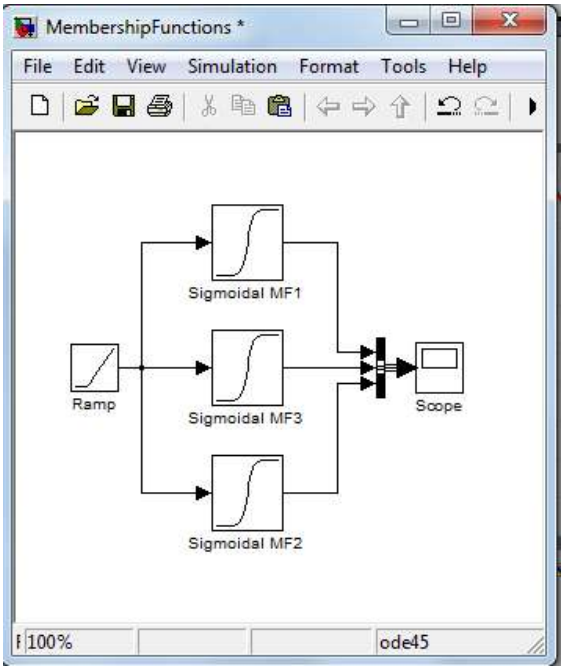

Рисунок 9 Подключение блоков Sigmoidal MF пакета Simulink, моделирующих работу сигмоидных функций принадлежности

На выходе функций принадлежности мы имеем несколько сигналов (по одному сигналу на каждую функцию; число сигналов должно быть равно числу лингвистических категорий, использованных в разработанном наборе правил). Далее данные передаются на заключительные функции принадлежности (в англоязычной литературе «consequent membership functions»). Эти функции могут иметь тот же вид что и входные функции принадлежности. Основное отличие работы с ними состоит в том, что в случае с заключительными функциями принадлежности нас интересует площадь образованной ими фигуры и её центр масс. Поступающий на вход такой функции сигнал обрезает её или масштабирует её (см. рисунок 10).

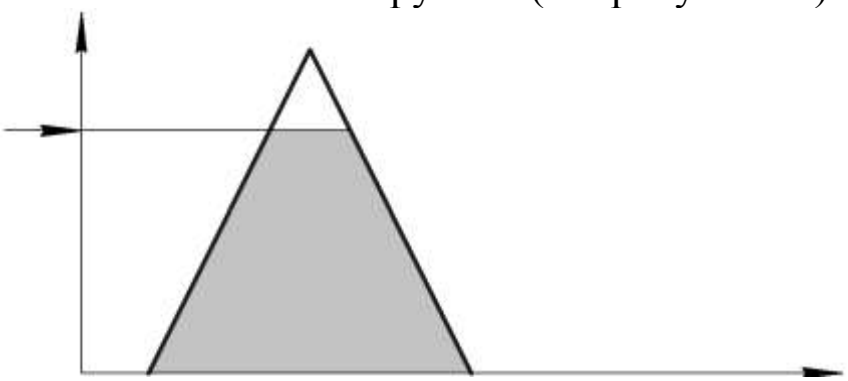

Рисунок 10 Сигнал, поступивший на вход заключительной функции принадлежности, обрезает её верхнюю часть; площадь и центр масс полученной фигуры являются выходными параметрами функции

Следующим и заключительным этапом обработки сигналов в нечетком регуляторе является дефаззификация. На этом этапе все полученные на прошлом этапе фигуры складываются и определяется их общий центр масс (см. рисунок 11)

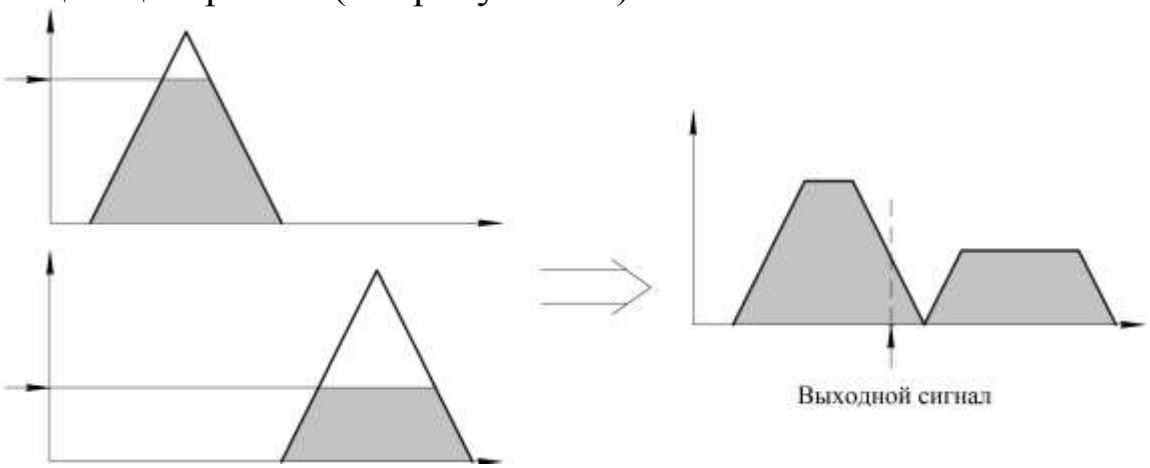

Рисунок 11 Сложение сигналов, нахождение центра масс

2. Методика выполнения работы

Для выполнения работы требуется разработать структуру нечеткого регулятора и интегрировать регулятор в систему управления подводным роботом, осуществляющим горизонтальное перемещение.

Динамику робота, при перемещении вдоль горизонтально оси, можно описать следующими уравнениями:

$$\begin{cases} m\ddot{x} + \mu\dot{x} = F \\ F = \omega \cdot C \end{cases}. \qquad (1)$$

где x - положение робота, m - масса робота, μ - коэффициент лобового сопротивления, F - сила тяги, создаваемая гребным винтом, ω - угловая скорость гребного винта, C - коэффициент, связывающий скорость вращения винта и создаваемую им тягу.

Модель системы автоматического управления, построенная в среде Simulink показана на рисунке 12.

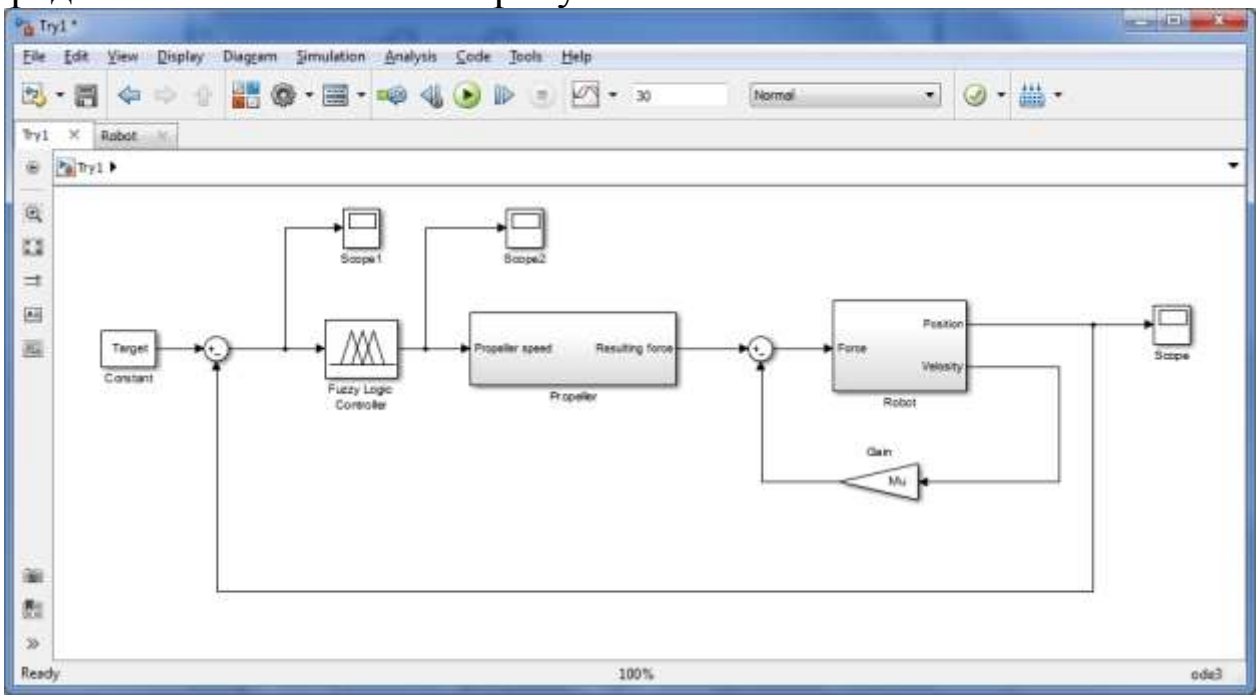

Рисунок 12 Модель системы автоматического управления

2.1 Модель робота

Модель робота реализована средствами SimMechanics – набора инструментов, доступного при использовании Simulink. Модель робота содержится в подсистеме Robot. На рисунке 13 показано содержание этой подсистемы.

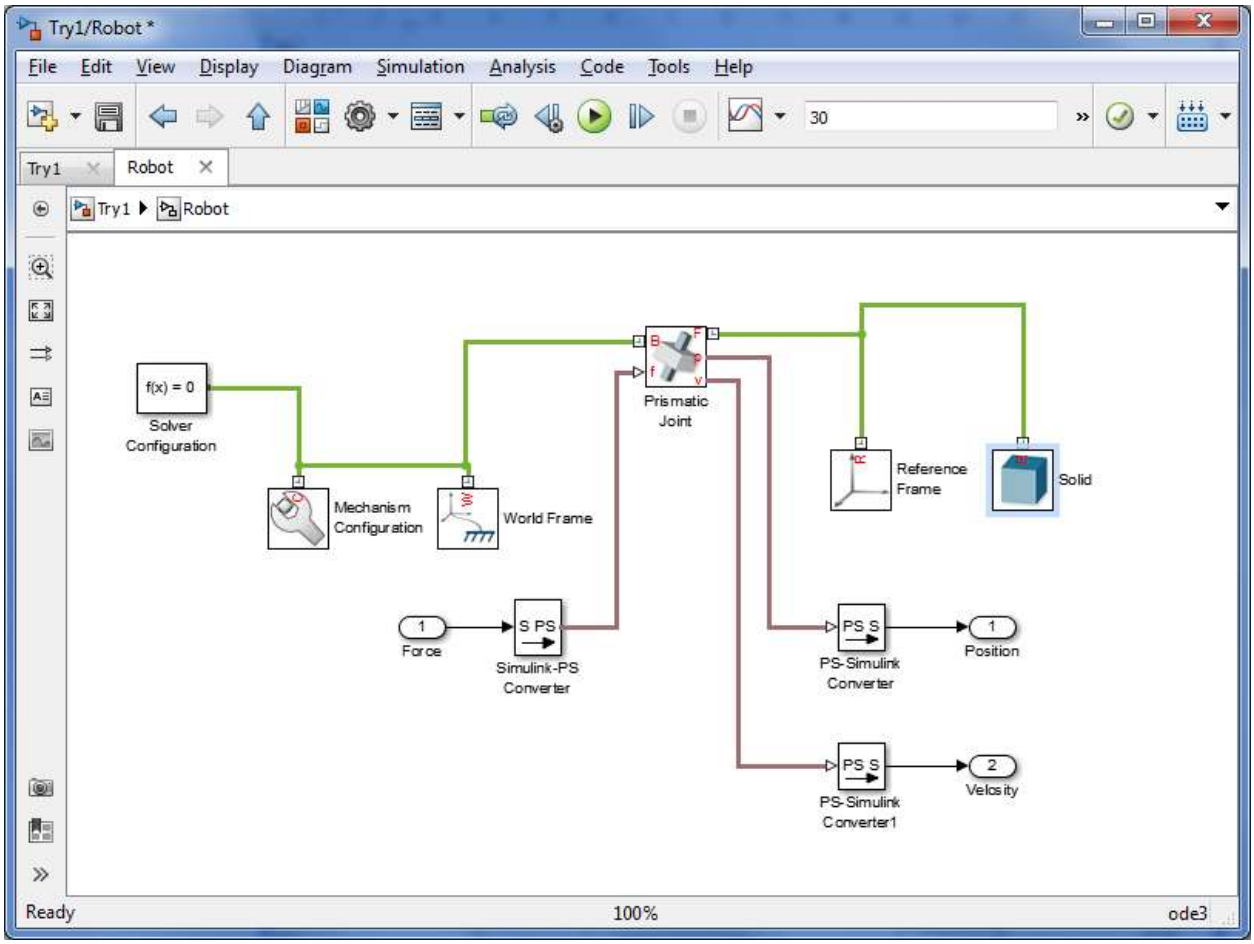

Рисунок 13 Подсистема Robot, содержащая модель робота, построенную с привлечением средств SimMechanics

Входным параметром для подсистем Robot является суммарная сила, приложенная к роботу. выходными – положение и скорость робота. Указанная модель использует блоки SimMechanics Second Generation, которые можно найти в библиотеке Simulink, как это показано на рисунке 14.

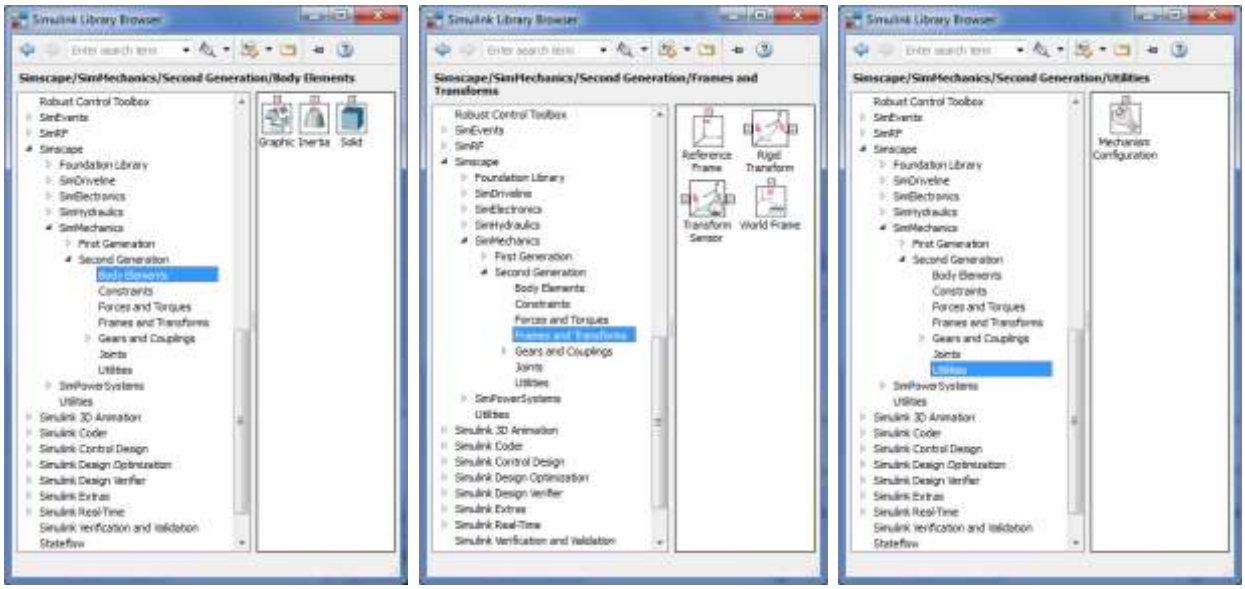

Рисунок 14 Расположение использованных блоков SimMechanics Second Generation в библиотеке Simulink

Часть использованных блоков относится к набору Simscape, частью которого является SimMechanics. Расположение этих блоков в библиотеке приведено на рисунке 15.

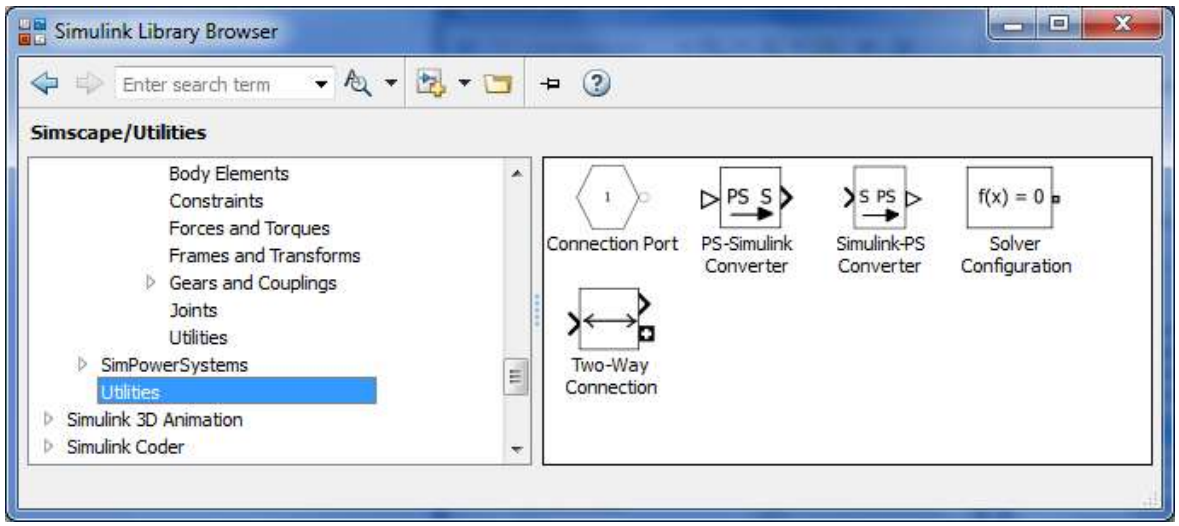

Рисунок 15 Расположение использованных блоков Simscape в библиотеке Simulink

Блоки Solid, Prismatic Joint, и Mechanism Configuration требуют настройки. На рисунках 16 и 17 показаны использованные настройки блоков. Настройка блока Solid заключается в указании массы робота. Настройка блока Prismatic Joint заключается в определении его входов и выходов (входом является сумма внешних сил, выходами по-

ложение и скорость робота). Настройка блока Mechanism Configuration заключается в указании величины и направления ускорения свободного падения. В данном случае нас интересует движение в плоскости, перпендикулярной линиям гравитационного поля, поэтому можем указать нулевую величину ускорения свободного падения.

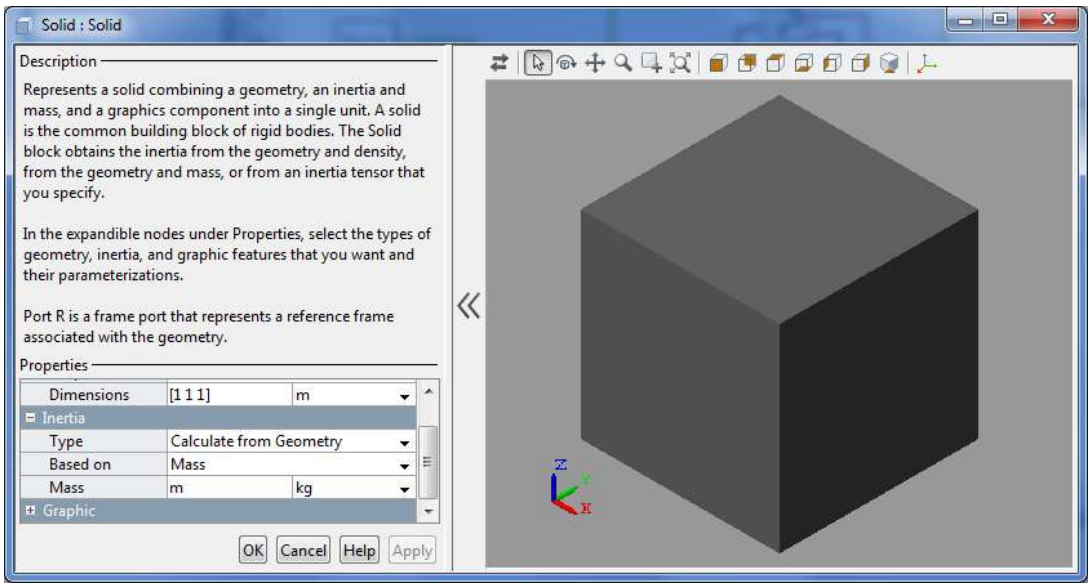

Рисунок 16 Настройки блока Solid

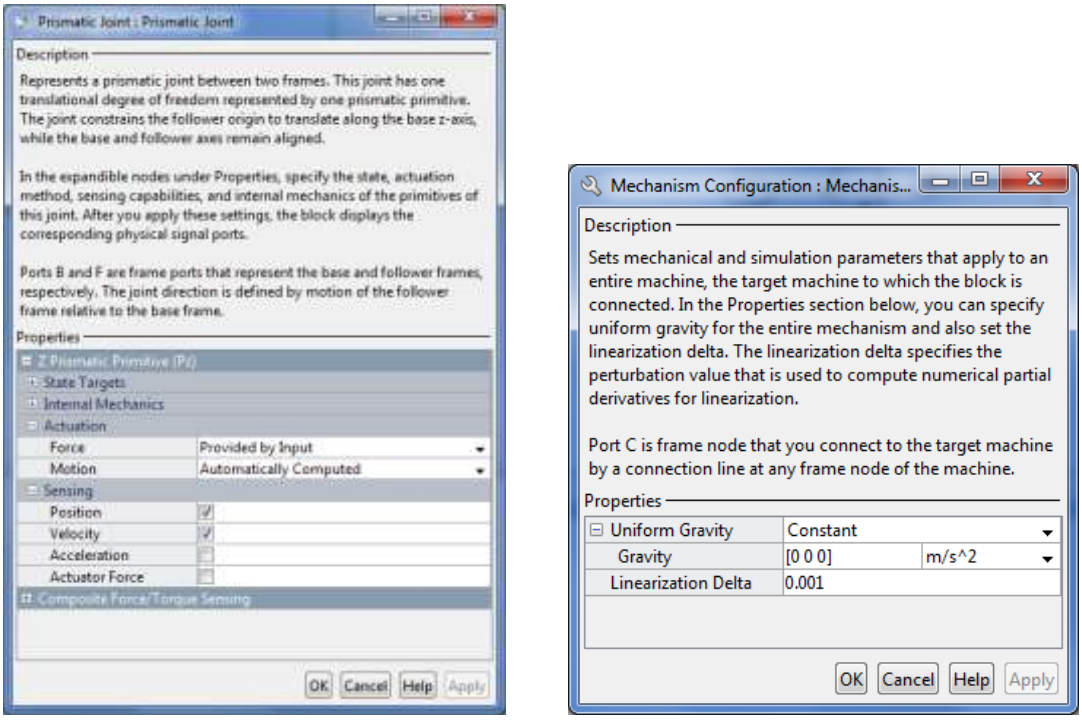

Рисунок 17 Настройки блоков Prismatic Joint, и Mechanism Configuration

Отметим, что сила сопротивления жидкости моделируется как пропоруиональная скорости робота, с помощью блока Gain.

2.2 Разработка и настройка нечеткого регулятора

Для создания нечеткого регулятора вопспользуемся расширением MATLAB - Fuzzy Logic Designer.

Для вызова приложения Fuzzy Logic Designer используется команда fuzzyLogicDesigner:

Листинг 1 Вызов приложения Fuzzy Logic Designer

```
>> fuzzyLogicDesigner
>>
```

После выполнения команды fuzzyLogicDesigner появляется окно, показанное на рисунке 18.

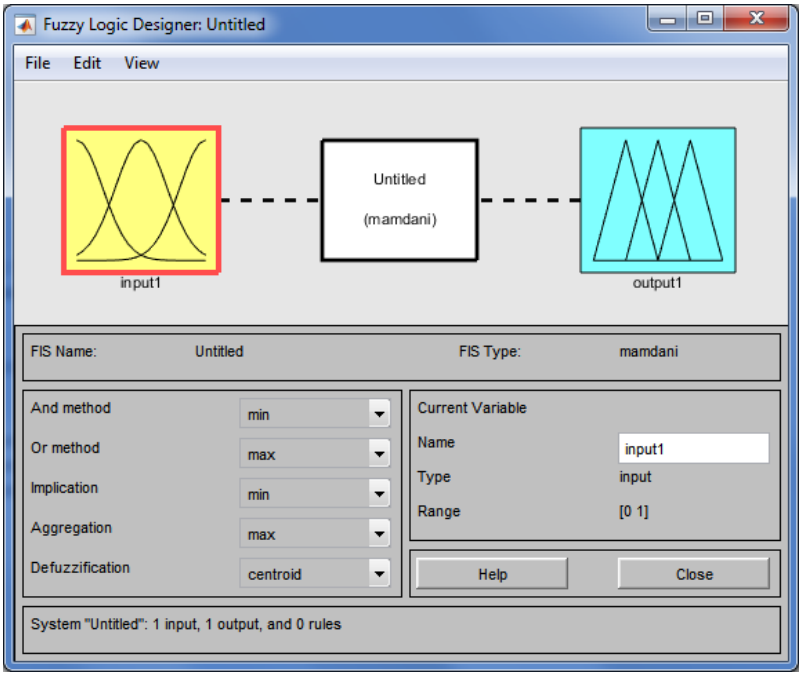

Рисунок 18 Окно приложения Fuzzy Logic Designer

По умолчанию приложение генерирует регулятор с одним входом input1 и одним выходом output1. Переименуем вход и выход в DistanceLeft, а выход в Speed. Результат показан на рисунке 19.

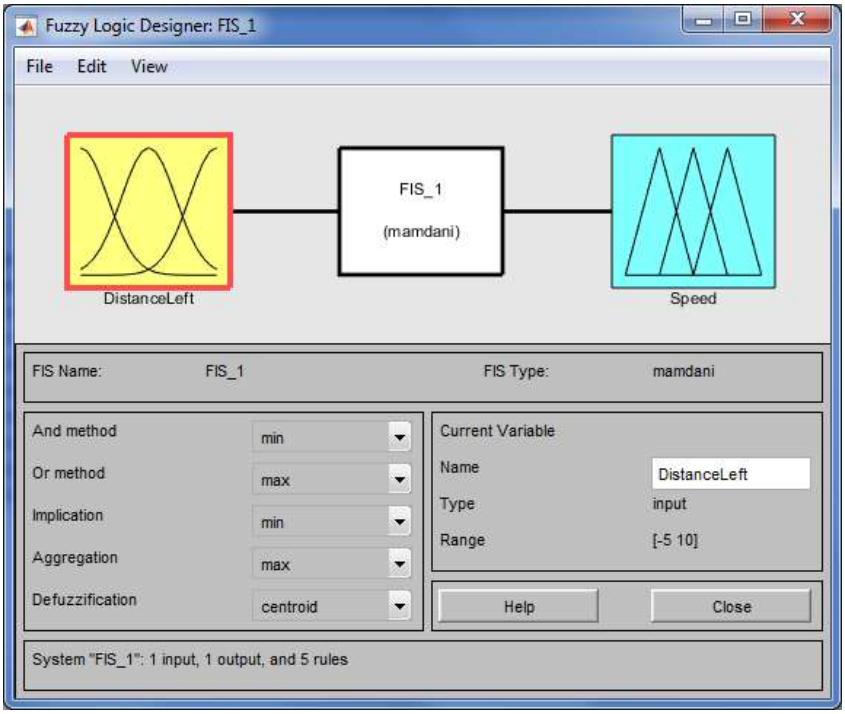

Рисунок 19 Окно приложения Fuzzy Logic Designer; вход и выход переименованы

Дважды кликнув на вход откроем окно редактирования функций принадлежности. Отредактируем эти функции так, как показано на рисунке 20.

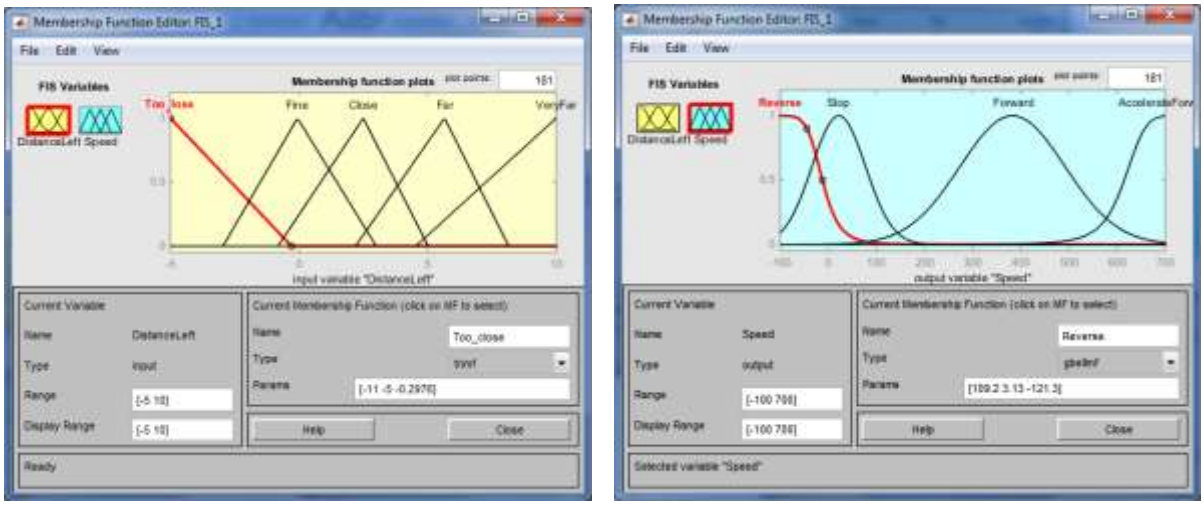

Рисунок 20 Настройка функций принадлежности

Обратим внимание, что для выхода были выбраны Гауссовы функции принадлежности, а для входа – треугольные. Менять тип функции можно используя выпадающий список Type. Гауссовым

функциям принадлежности соответствует тип gbellmf, а треугольным – trimf. Каждую функцию принадлежности следует переименовать в соответствии с её физическим смыслом.

Следующим шагом является запись набора нечетких правил. Используемый набор правил был приведен ранее. Результат записи набора правил показан на рисунке 21.

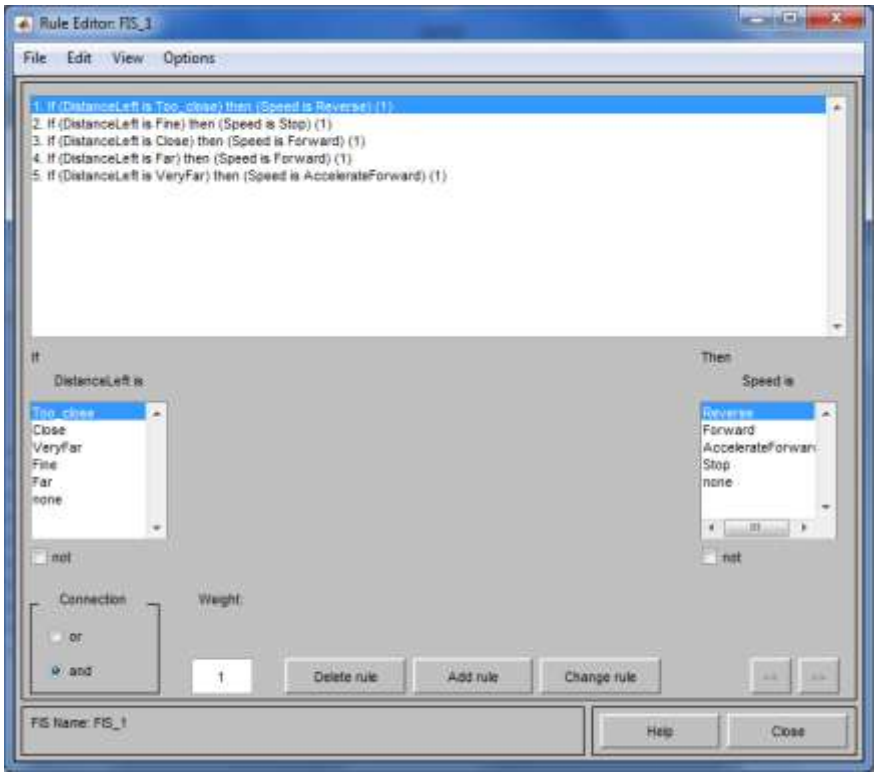

Рисунок 21 Набор нечетких правил

Работу нечеткого регулятора хорошо иллюстрирует поверхность отклика. Приложение Fuzzy Logic Designer имеет встроенный инструмент построения таких поверхностей. Для вызова данного инструмента следует кликнуть на пункт главного меню View->Surface, или нажать Ctrl+6. В рассматриваемом случае поверхность отклика представляет собой график, показывающий связь между входными и выходными величинами разработанного нечеткого регулятора. В случае, если бы регулятор имел более одного входа, поверхность отклика задавалась бы действительной функцией двух переменных, т.е. поверхностью в трехмерном пространстве, с чем и связано название «поверхность отклика». Внешний вид полученного графика показан на рисунке 22.

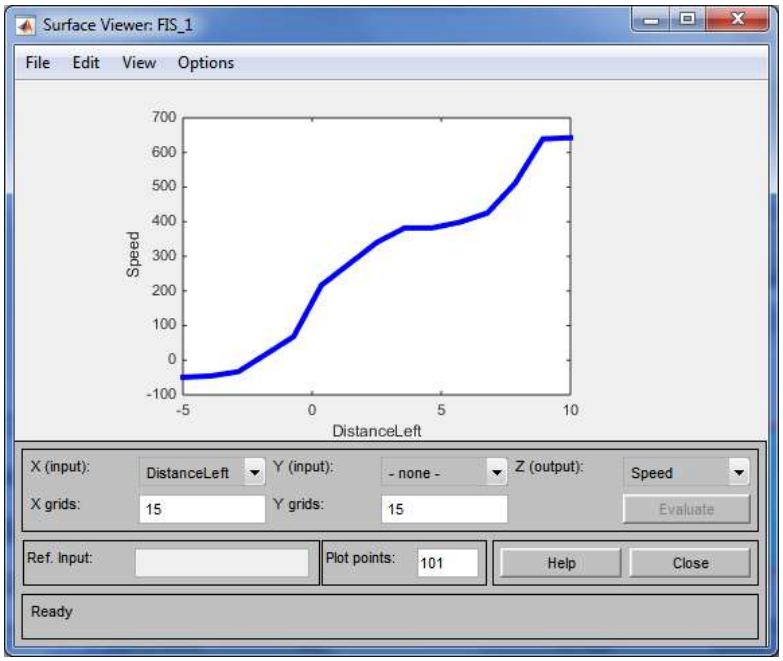

Рисунок 22 График, показывающий связь между входными и выходными величинами разработанного нечеткого регулятора

Приложение Fuzzy Logic Designer позволяет провести анализ влияния отдельных функций принадлежности на работу системы. Для этого используется Rule Viewer. Для того, чтобы вызвать окно Rule Viewer нужно кликнуть на пункт главного меню View->Rules, или нажать Ctrl+5. В открывшемся окне можно перемещать линию, моделирующую входное значение регулятора, задавая тем самым различные входные величины. Rule Viewer графически изображает разработанные для данного регулятора входные функции принадлежности. Те из них, в которые попадает данный входной сигнал, будут подсвечены. Для каждого входного сигнала показано как он влияет на работу выходных функция принадлежности, а также показан найденный центр масс выходных функций – выходное значение регулятора. Внешний вид окна Rule Viewer показан на рисунке 23.

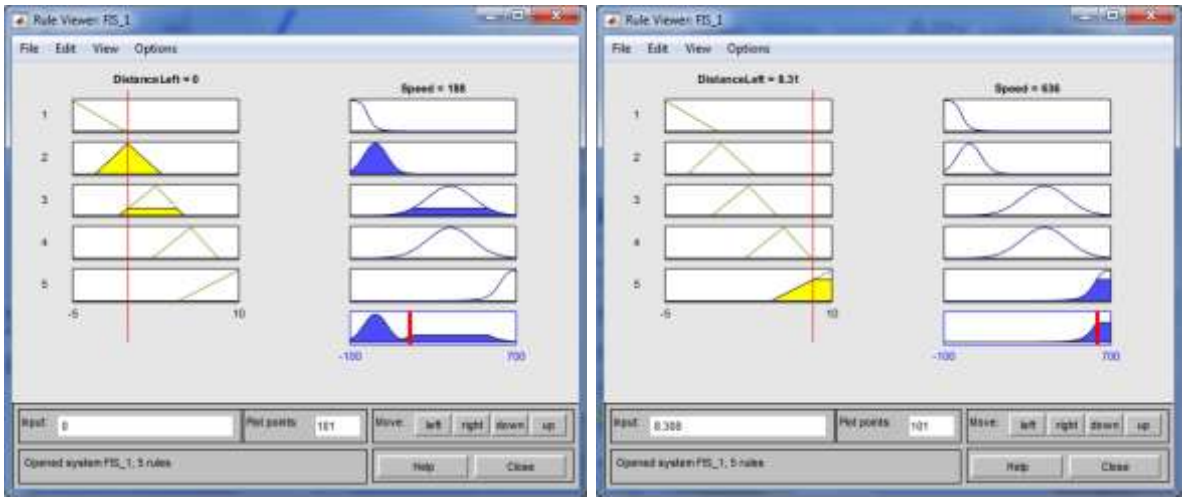

Рисунок 23 Внешний вид окна Rule Viewer при имитации различных входных сигналов

Для того, чтобы сохранить созданный нечеткий регулятор нужно кликнуть на пункт главного меню File->Export->To File, или нажать Ctrl+S. Можно сохранить регулятор в переменную, которая будет доступна в приложениях MATLAB, для этого нужно кликнуть на пункт главного меню File->Workspace->To File, или нажать Ctrl+T. Данная переменная будет удалена после окончания работы MATLAB, или после принудительной очистки памяти (например командой clear).

Для того, чтобы загрузить ранее созданный нечеткий регулятор нужно кликнуть на пункт главного меню File->Import->From File, или нажать Ctrl+O.

Для того, чтобы использовать созданный регулятор в среде Simulink, требуется использовать блок Fuzzy Logic Controller. на рисунке 24 показано расположение этого блока в библиотеке Simulink.

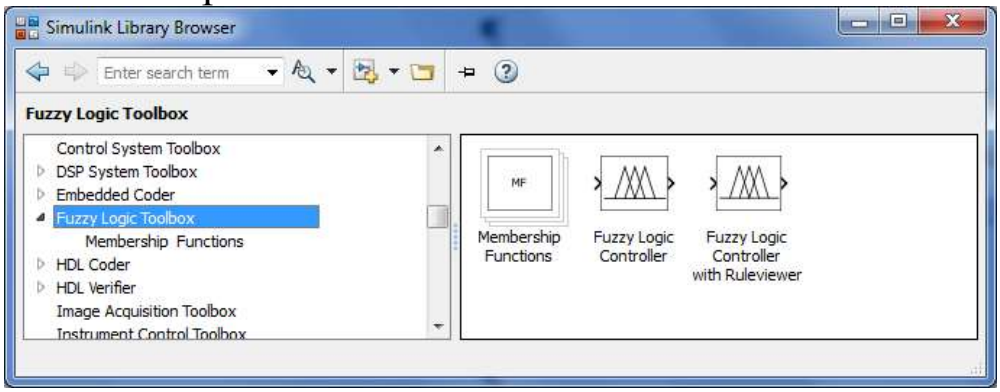

Рисунок 24 Расположение блока Fuzzy Logic Controller в библиотеке Simulink

Блок Fuzzy Logic Controller требует указать имя файла из которого будет считан нечеткий регулятор, или имя переменной содержащей этот регулятор. Здесь будем использовать второй способ. На рисунке 25 показаны настройки блока Fuzzy Logic Controller.

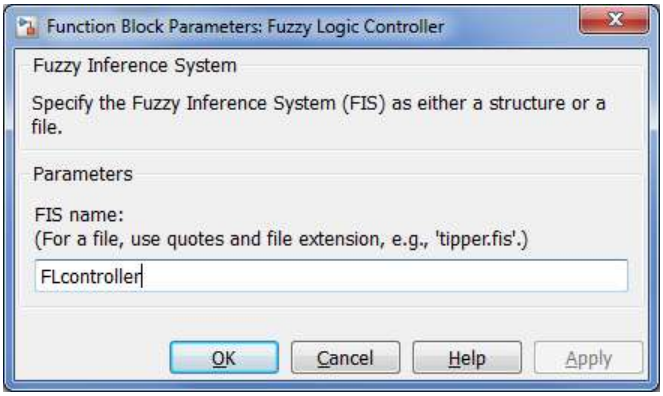

Рисунок 25 Настройки блока Fuzzy Logic Controller

Для того, чтобы создать переменную FLcontroller, которая будет использована блоком Fuzzy Logic Controller, выполним следующий скрипт (см. рисунок 26). Данный скрипт устанавливает значения всех использованных в данной работе переменных.

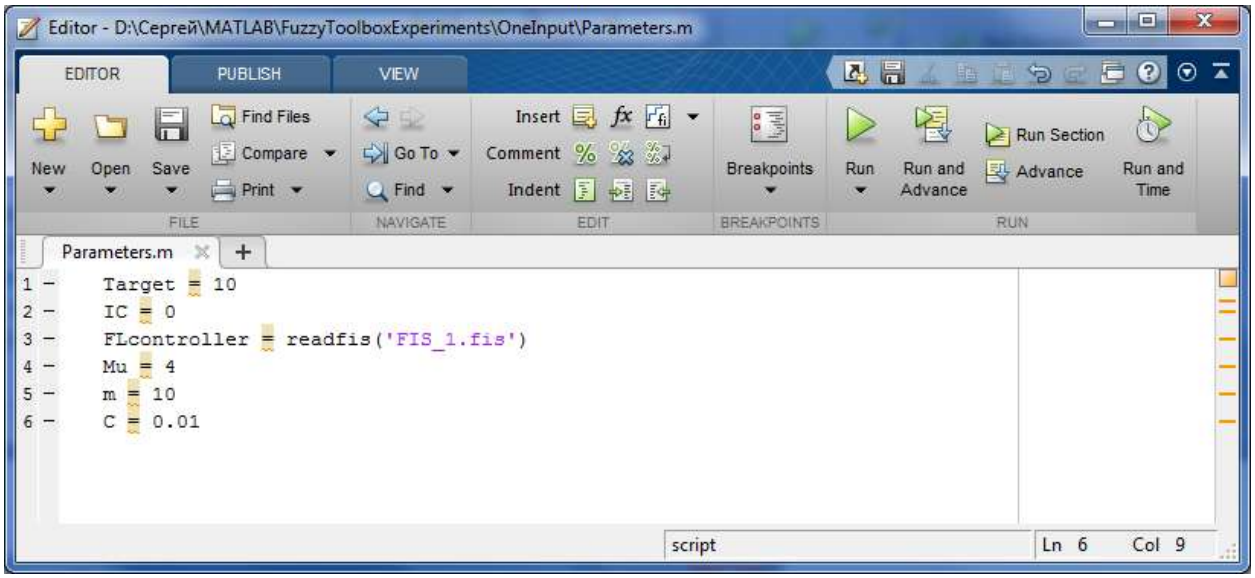

Рисунок 26 Скрипт, задающий переменную FLcontroller

Такой способ работы с блоком Fuzzy Logic Controller оправдан тем, что это позволяет динамически модифицировать нечеткий регулятор используя Fuzzy Logic Designer, сохраняя результаты в виде

переменной, что быстрее чем сохранение в файл. Сохранить результаты работы в файл будет необходимо один раз в конце сессии работы с Fuzzy Logic Designer.

Для запуска скрипта требуется нажать на кнопку Run.

Запустим моделирование созданной Simulink модели, отобразим результаты в виде графиков (см. рисунки 27 и 28).

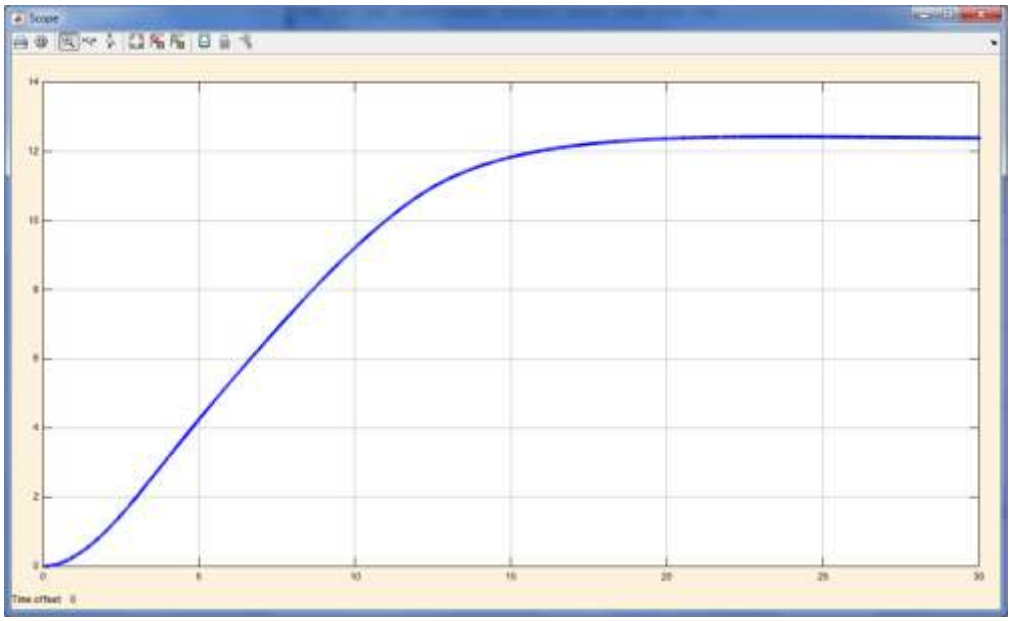

Рисунок 27 Временная зависимость положения робота

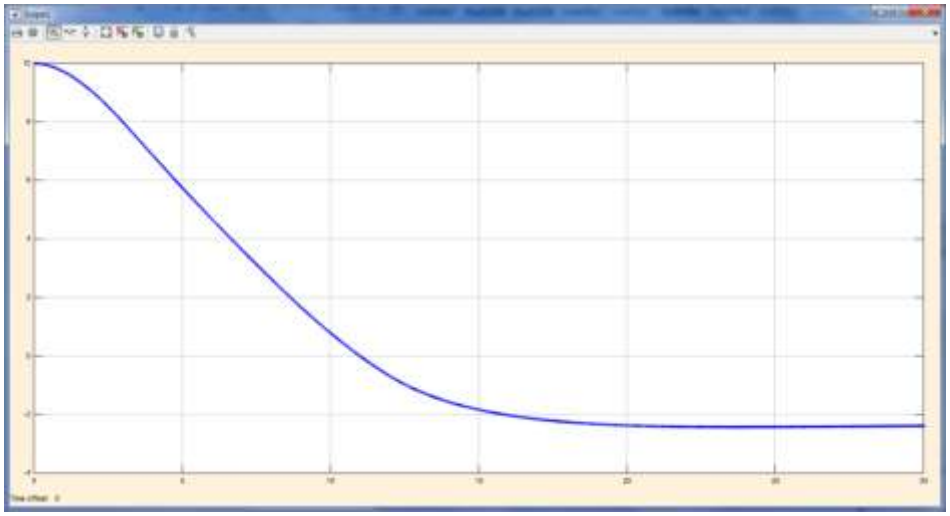

Рисунок 28 Временная зависимость входного сигнала блока Fuzzy Logic Controller

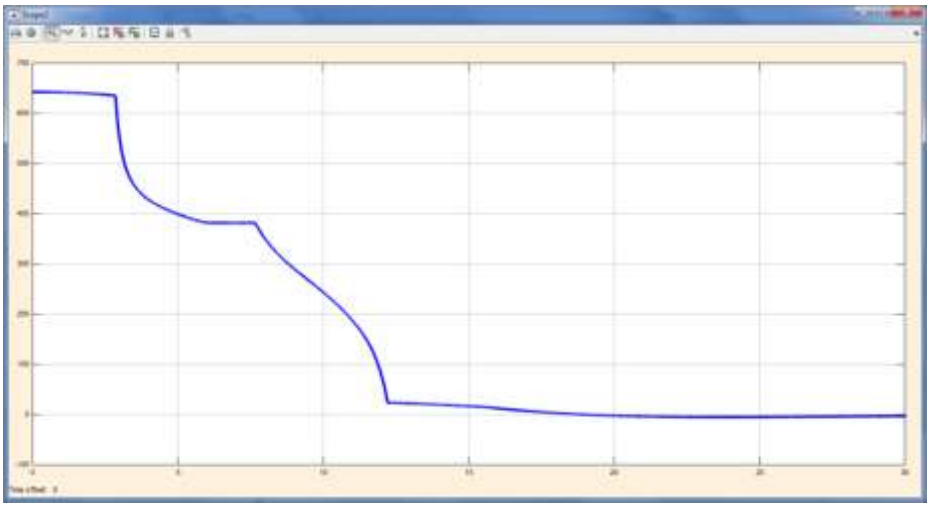

Рисунок 29 Временная зависимость выходного сигнала блока Fuzzy Logic Controller

Для того, чтобы блок Scope отображал графики темными тонами на светлом фоне, можно использовать следующие настройки блока (см. рисунок 29):

Рисунок 29 Настройки отображения графиков в блоке Scope

В случае, если количество точек на графике. отображаемом блоком Scope превышает 5000 нужно модифицировать его параметры хранения данных, иначе будут построены лишь последние 5000 точек. На рисунке 30 показаны соответствующие настройки блока Scope.

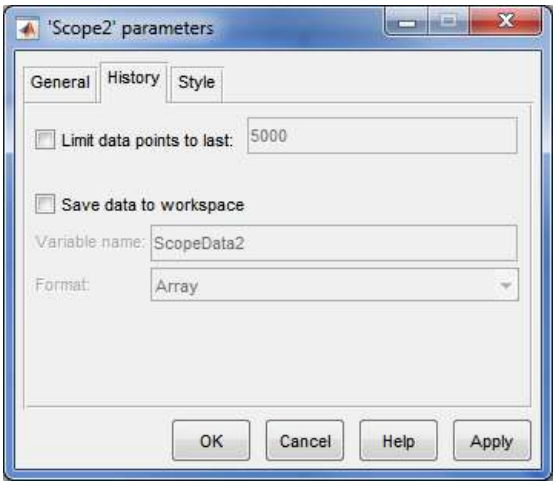

Рисунок 30 Настройки хранения информации в блоке Scope

Численные значения параметров, использованных в данной лабораторной работе, приведены в таблице 1.

3. Задание на выполнение самостоятельной работы

Задание на выполнение лабораторной работы следует выбирать из таблицы 3 в соответствии с номером студента в списке группы.

Таблица 1 Задания на выполнение лабораторной работы

№	Задание
1	Настроить нечеткий регулятор таким образом, чтобы уменьшить колебательность системы. Использовать единичное задающее воздействие.
2	Настроить нечеткий регулятор таким образом, чтобы понизить время переходного процесса. Использовать единичное задающее воздействие.
3	Настроить нечеткий регулятор таким образом, чтобы понизить перерегулирование. Использовать единичное задающее воздействие.
4	Настроить нечеткий регулятор на отработку нарастающего задающего воздействия.
5	Настроить нечеткий регулятор таким образом, чтобы уменьшить колебательность системы. Использовать задающее воздействие y = 20.

6	Настроить нечеткий регулятор таким образом, чтобы понизить время переходного процесса. Использовать задающее воздействие y = 20.
7	Настроить нечеткий регулятор таким образом, чтобы понизить перерегулирование. Использовать задающее воздействие y = 20.
8	Настроить нечеткий регулятор таким образом, чтобы устранить колебательность системы. Использовать задающее воздействие y = 30.
9	Настроить нечеткий регулятор таким образом, чтобы понизить время переходного процесса. Использовать задающее воздействие y = 30.
10	Настроить нечеткий регулятор таким образом, чтобы понизить перерегулирование. Использовать задающее воздействие y = 30.

4. Оформление отчета о выполнении работы

Требования к отчету:

- отчет содержит титульный лист, описание выполняемого задания, описание проделанной работы, анализ полученных результатов, выводы, список использованной литературы;
- отчет выполняется на листах формата А4, 14 кегль, одинарный межстрочный интервал;
- список литературы оформляется согласно ГОСТ 7.1-2003.

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине

Visual Studio Professional 2015, договор IT000012385
GNU Octave https://www.gnu.org/software/octave/ Бесплатная, GNU General Public License

Рекомендуемая литература

1. Круглов, В.В. Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода [Текст] / В.В. Круглов В.В., М.И. Дли // М.: Физматлит, 2002. – 517 с.
2. Емельянов, С.Г. Интеллектуальные системы на основе нечеткой логики и мягких арифметических операций [Текст] : учебник / С. Г. Емельянов, В. С. Титов, М. В. Бобырь. - Москва : Аргамак-Медиа, 2014. - 338, [7] с. : табл., граф. - Библиогр.: с. 325-336.
3. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы [Текст] / Д. Рутковская, М. Пилиньский, Л. Рутковский // М.: Наука, 2004. – 457 с.
4. Гаврилова, Т.А. Базы знаний интеллектуальных систем [Текст] / Т.А. Гаврилова, В.Ф. Червинский // СПб.: БХВ-Петербург, 2000. – 384 с.
5. Леоленков, А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH [Текст] // СПб.: БХВ-Петербург, 2003. – 417 с.
6. Яцун, С.Ф. Вибрационные мобильные роботы [Текст] / С.Ф. Яцун, П.А. Безмен, Л.Ю. Волкова, В.В. Бартенев // Курск, Юго-Зап. гос. ун-т, 2013. – 181 с.
7. Макаров, И.М. Искусственный интеллект и интеллектуальные системы управления [Текст] / И.М. Макаров, В.М. Лохин, С.В. Маньков, М.П. Романов // М.: Наука, 2006. — 347 с.