МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт высокоточных систем им. В.П. Грязева

Кафедра систем автоматического управления

Утверждено на заседании кафедры
«Системы автоматического управления»
«13» января 2021 г., протокол № 8

Заведующий кафедрой

_____О.В.Горячев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине (модулю)

«Интеллектуальные системы управления»

основной профессиональной образовательной программы
высшего образования – программы специалитета

по специальности
24.05.06 Системы управления летательными аппаратами

со специализацией
Системы управления беспилотными летательными аппаратами

Форма обучения: очная

Идентификационный номер образовательной программы: 240506-01-21

Тула 2021 год

Разработчик(и) методических указаний

Горячев О.В., зав. кафедрой САУ, д.т.н, проф. _____

(ФИО, должность, ученая степень, ученое звание) *(подпись)*

СОДЕРЖАНИЕ

№	Наименование работы	Стр.
1.	Разработка экспертной системы настройки параметров ПИД-регулятора ЭСП постоянного тока	4
2.	Работа с нечеткими множествами в среде MATLAB	20
3.	Разработка нечеткой системы типа Мамдани	27
4.	Разработка нечеткой системы типа Сугено на основе экспертных знаний	37
5.	Проектирование нечеткого регулятора для ЭСП с двигателем постоянного тока	44
	Краткие теоретические сведения по пакету прикладных программ NNT (The Neural Network Toolbox)	57
6.	Модели искусственного нейрона. Решение задач в нейросетевом базисе	63
7.	Искусственные нейронные сети	73
8.	Формирование и обучение нейросети с помощью инструментальных средств MATLAB	83
9.	Создание и обучение нейросети заданной структуры с помощью инструмента Network/Data Manager	97
10.	Синтез нейронной сети для решения систем обыкновенных дифференциальных уравнений	112
11.	Применение нейронных сетей для проектирования систем управления динамическими процессами	118
12.	Формирование и обучение нейронечеткой сети для целей управления динамическими объектами с помощью инструментальных средств MATLAB	132
13.	Создание и обучение нейронечеткого регулятора, реализующего стандартные алгоритмы (П-, ПИ-, ПД-, ПИД- Регуляторы)	146

Лабораторная работа №1

РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ НАСТРОЙКИ ПАРАМЕТРОВ ПИД-РЕГУЛЯТОРА ЭСП ПОСТОЯННОГО ТОКА

1. ЦЕЛЬ РАБОТЫ

Получение навыков разработки экспертных систем, которые включают в себя базу знаний по формированию алгоритмов настройки параметров ПИД-регуляторов, обеспечивающих выполнение требований ТЗ по качеству переходного процесса, статической ошибке и динамической точности электрических следящих приводов (ЭСП) постоянного тока.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Основные замечания

ПИД-регуляторы широко распространены в управлении автоматизированными приводами, используемыми в технологических процессах, что обусловлено их простой реализацией и достаточно высокой эффективностью. Однако, как правило, вследствие изменения параметров объекта управления (ОУ) в процессе эксплуатации, ПИД-регуляторы требуют периодического обслуживания и подстройки.

Одним из возможных путей решения данной проблемы является разработка экспертной системы, которая осуществляет активную диагностику привода, т.е. определяет возможные отклонения в выполнении требований ТЗ на основе анализа изменений параметров выходных координат и компенсирует обнаруженные отклонения, с помощью коррекции параметров регулятора, основанной на теоретических, эмпирических знаниях и опыте подстройки коэффициентов (рис. 1.1).

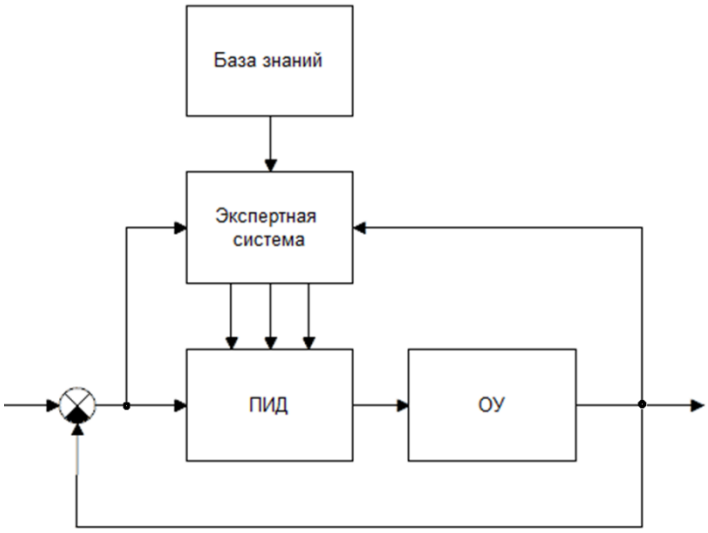


Рис. 1.1. Структурная схема ЭСП постоянного тока с экспертной системой настройки ПИД-регулятора.

2.2. ПИД-регулятор

В практике проектирования следящих приводов мехатронных систем наиболее часто используется пропорционально-интегро-дифференциальный (ПИД) закон управления, описываемый формулой

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int e(t)dt$$

или в операторной форме

$$u(p) = K_p e(p) + K_d p e(p) + K_i \frac{1}{p} e(p)$$

где K_p, K_d, K_i–параметры ПИД -регулятора (закона управления) (рис.1.2).

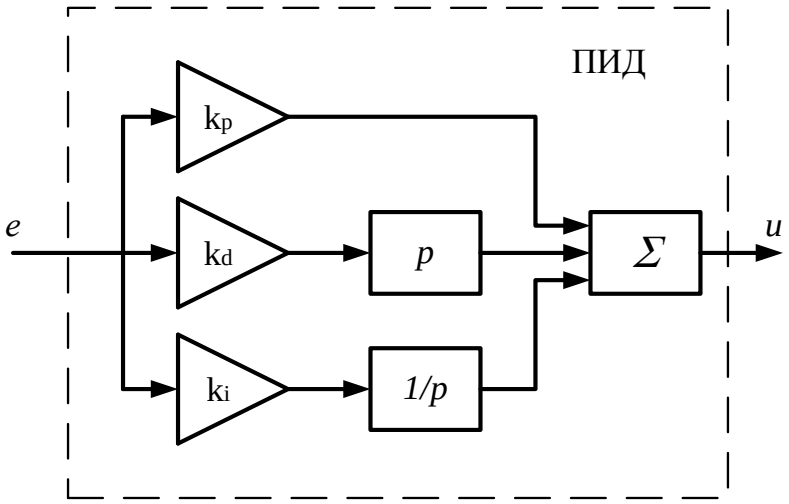


Рис. 1.2. Структура ПИД-регулятора

Параметры ПИД-регулятора оказывают различное влияние на переходный процесс системы. В табл. 1.1 представлено влияние увеличения коэффициентов ПИД регулятора на характеристики управляемого процесса.

Таблица 1.1

Коэффициент	**Время нарастания**	**Установив-шаяся ошибка**	**Перерегули-рование**
По пропорциональной составляющей	уменьшает	уменьшает	увеличивает
По дифференциальной составляющей	не влияет	не влияет	уменьшает
По интегральной составляющей	скорее увеличивает	устраняет	увеличивает

Таким образом, на основе анализа таблицы 1 можно сделать следующие выводы:

- пропорциональная составляющая отвечает за быстродействие системы (время нарастания), а также за величину установившейся ошибки;
- дифференциальная составляющая позволяет увеличить демпфирование системы, т. е. подавить нежелательные колебания и уменьшить перерегулирование;
- интегральная составляющая позволяет уменьшить до нуля установившуюся ошибку в системе.

В общем случае ПИД-регуляторы можно отнести к категории экспертных регуляторов, поскольку они могут настраиваться путем непосредственных экспериментов с объектом в соответствии с известными методиками (Зиглера-Николса, Коэна-Куна и др.).

2.3. Определение параметров ПИД-регулятора

Существует множество методов настройки ПИД-регулятора. Одним из них является метод Коэна-Куна. / http://pages.mtu.edu/~tbco/cm416/cctune.html/

Метод Коэна-Куна (Cohen-CoonTuningMethod) позволяет определить параметры П, ПИ и ПИД регуляторов по одной кривой отклика, то есть по переходному процессу нескорректированного объекта ОУ при воздействии ступенчатого сигнала.

Достоинство этого метода заключается в том, что ОУ запускается один раз и на основе полученной характеристики по известным формулам рассчитываются параметры КУ, при этом передаточная функция ОУ может оставаться неизвестной.

Недостатком является то, что нескорректированный ОУ должен быть устойчивым. В случае если объект неустойчив, необходима дополнительная коррекция другими методами. Также метод не учитывает требования ТЗ.

Для расчета коэффициентов ПИД-регулятора по методу Коэна-Куна необходимо на замкнутую систему без регулятора и при наличии возмущений подать ступенчатое воздействие с некоторым запаздыванием, при котором переходные процессы, вызванные возмущением, успеют прийти к устойчивому состоянию.

На рис. 1.3 представлены процессы на входе и выходе системы.

На вход системы в момент времени t_0 подается ступенчатое воздействие амплитудой – А.

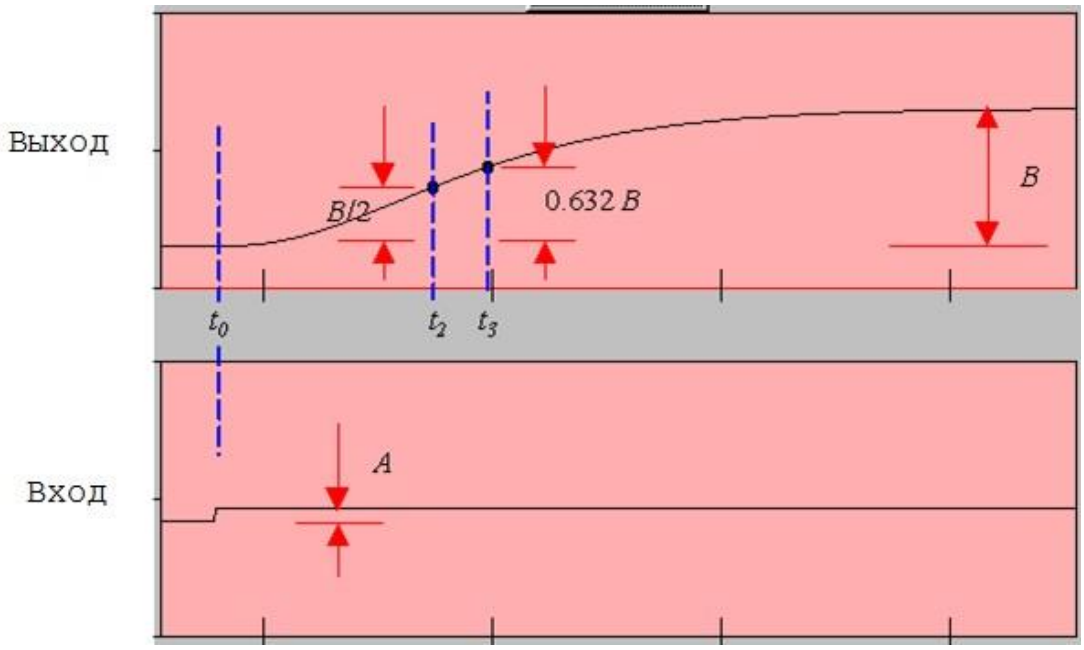


Рис. 1.3. Процессы на входе и выходе системы

По полученной кривой отклика необходимо снять следующие параметры:

B– разница между установившимся значением выходной величины и значением в момент времени t_0;

t_2–время, при котором выходная величина достигает половины *B*;

t_3–время, при котором выходная величина достигает 63.2% *B*.

Далее по следующим зависимостям рассчитываются дополнительные параметры процесса:

$$t_1 = \frac{t_2 - ln2 \cdot t_3}{1 - ln2};$$

$$\tau = t_3 - t_1;$$

$$\tau_{DEL} = t_1 - t_0;$$

$$K = \frac{B}{A};$$

$$r = \frac{\tau_{DEL}}{\tau}.$$

По полученным значениям рассчитываются коэффициенты корректирующего устройства (КУ) из табл. 1.2.

Таблица1. 2

	K_p	$T_i = 1/K_i$	$T_d = K_d$
П	$\frac{1}{K \cdot r}\left(1 + \frac{r}{3}\right)$	-	-
ПИ	$\frac{1}{K \cdot r}\left(0.9 + \frac{r}{12}\right)$	$\tau_{DEL}\frac{30 + 3r}{9 + 20r}$	-
ПИД	$\frac{1}{K \cdot r}\left(\frac{4}{3} + \frac{r}{4}\right)$	$\tau_{DEL}\frac{32 + 6r}{13 + 8r}$	$\tau_{DEL}\frac{4}{11 + 2r}$

Рассчитанные значения параметров ПИД - регулятора по приведенным зависимостям обеспечивают удовлетворительные переходные процессы. Однако, если в соответствии с ТЗ требуется обеспечить более высокие характеристики, то необходимо выполнить процедуру коррекции или дополнительной настройки (подстройки) параметров регулятора.

2.4. Алгоритм подстройки параметров ПИД-регулятора

Для создания экспертной системы по настройке параметров ПИД-регулятора возьмем за основу эвристический метод настройки, как правило реализуемый на практике квалифицированным сотрудником, который заключается в поочередной установке параметров регулятора.

Настройка проводится на основе анализа реакции системы на единичный ступенчатый сигнал при наличии возмущений, действующих на ОУ.

На первом этапе изменяют пропорциональную составляющую ПИД-регулятора так, чтобы кривая переходного процесса на выходе системы состояла из 2-3 колебаний. На практике это соответствует перерегулированию $\sigma = 20..30\%$.

На втором этапе переходный процесс демпфируют увеличением дифференциальной составляющая регулятора до тех пор, пока перерегулирование $\sigma \approx 0\%$, при этом следует учитывать время регулирования системы.

На третьем этапе увеличивают интегральную составляющую регулятора с целью обеспечить выполнение требований ТЗ по точности.

Алгоритм данного метода представлен на рис.1.4.

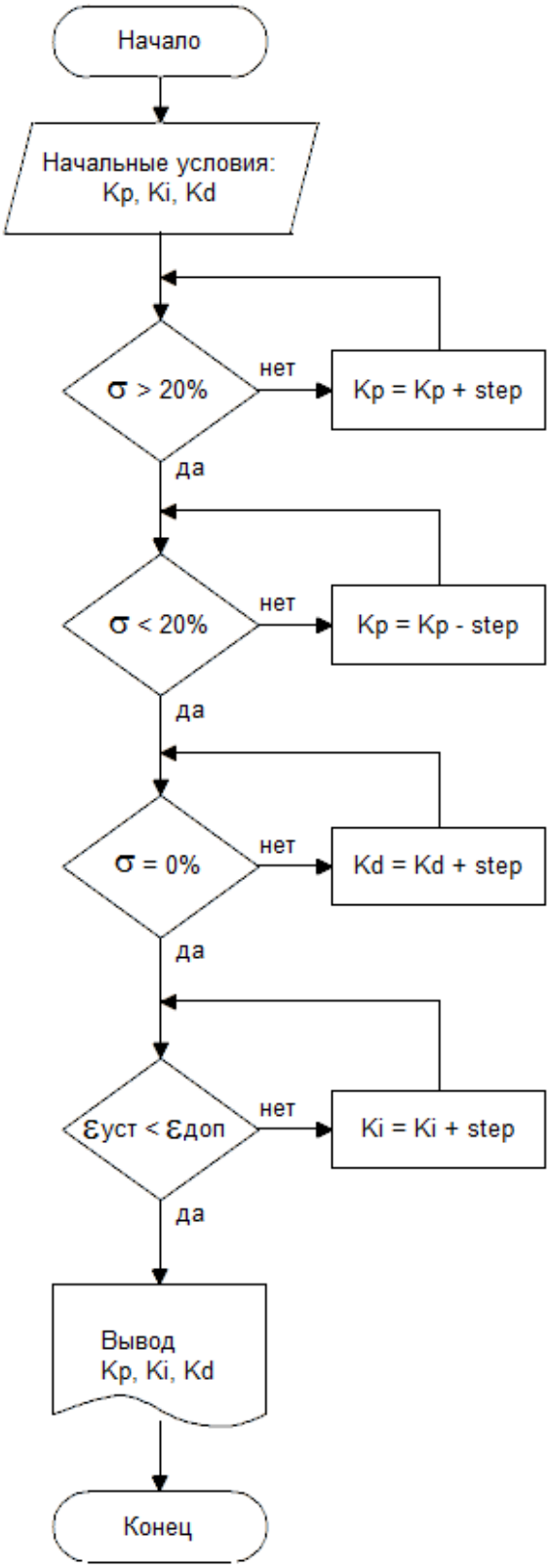


Рис. 1.4. Алгоритм эвристической подстройки ПИД-регулятора

3. ПОРЯДОК ВПОЛНЕНИЯ РАБОТЫ

Объектом управления является ЭСП, исполнительным двигателем которого является двигатель постоянного тока (ДПТ) ЭДМ-20, технические характеристики которого приведены в табл. 1.3.

Таблица 1.3

Технические характеристики ЭДМ-20	
Номинальное напряжение питания	27 В
Момент инерции якоря	0.0003 кг·м^2
Сопротивление обмотки якоря	0.15 Ом
Индуктивность обмотки якоря	0.15 мГн
Коэффициент противо-ЭДС	0.078 В·с/рад
Коэффициент момента двигателя	0.078 Н·м/А

Simulink-модель ДПТ представлена на рис. 1.5.

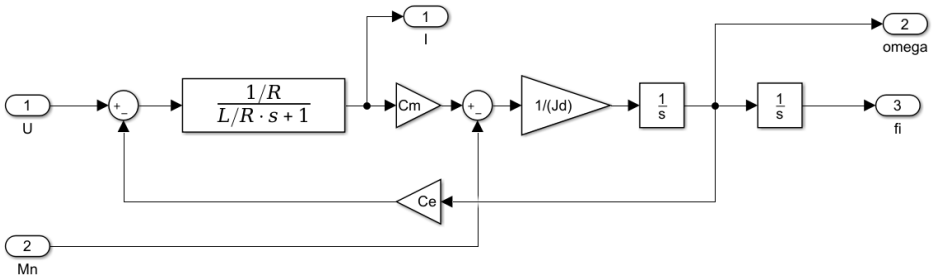


Рис. 1.5. Simulink-модель ДПТ

Simulink-модель ЭСП представлена на рис. 1.6.

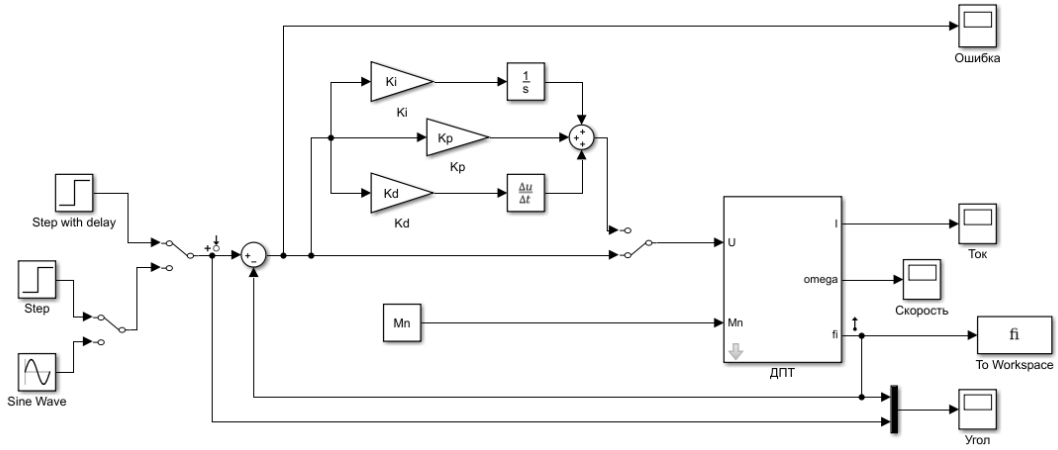


Рис. 1.6. Simulink-модель ЭСП

Требуется рассчитать параметры ПИД-регулятора методом Коэна-Куна и подстроить их, реализовав алгоритм ручной настройки.

Требования ТЗ:

- перерегулирование 0..5%;
- статическая ошибка 0.1 мрад;
- динамическая ошибка 0.5 мрад;
- активная нагрузка - момент нагрузки 1 Нм.

Схема моделирования показана на рис.1. 6.

Подадим на вход замкнутой нескорректированной системы в условиях возмущения по моменту единичный ступенчатый сигнал с временной задержкой 1 с, получим на выходе отклик системы (рис. 1.7).

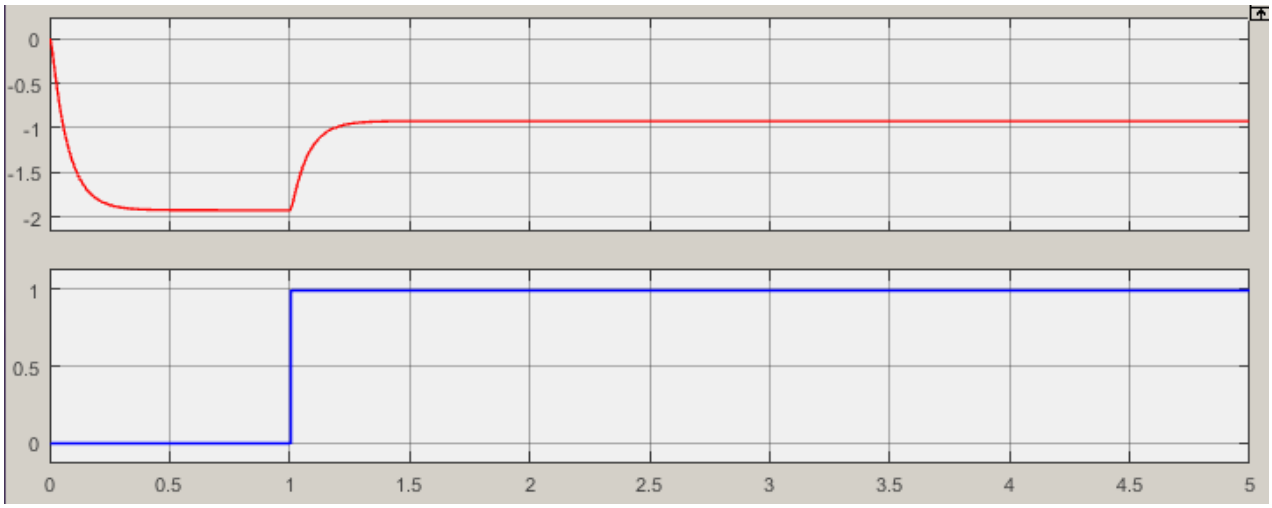


Рис. 1.7. Процессы на входе и выходе системы

Далее необходимо по формулам, приведенным в табл. 1.2 выполнить предварительный расчет параметров ПИД-регулятора:

$K_p = ..;\ K_d = ..;\ K_i = ..$

Подставив указанные параметры в расчетную модель, получим переходный процесс в скорректированной системе. Примерный вид переходного процесса представлен на рис. 1.8-1.9.

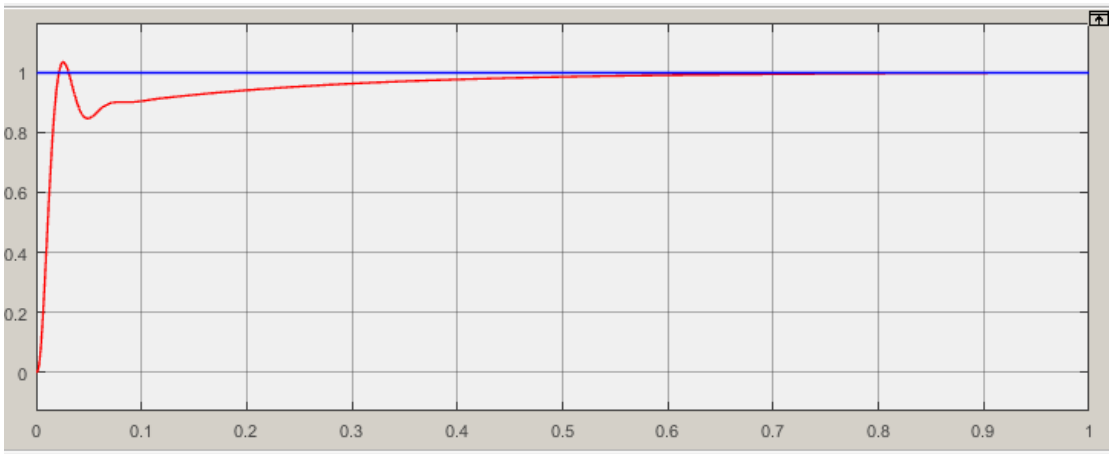


Рис. 1.8. Переходный процесс на выходе скорректированной системы

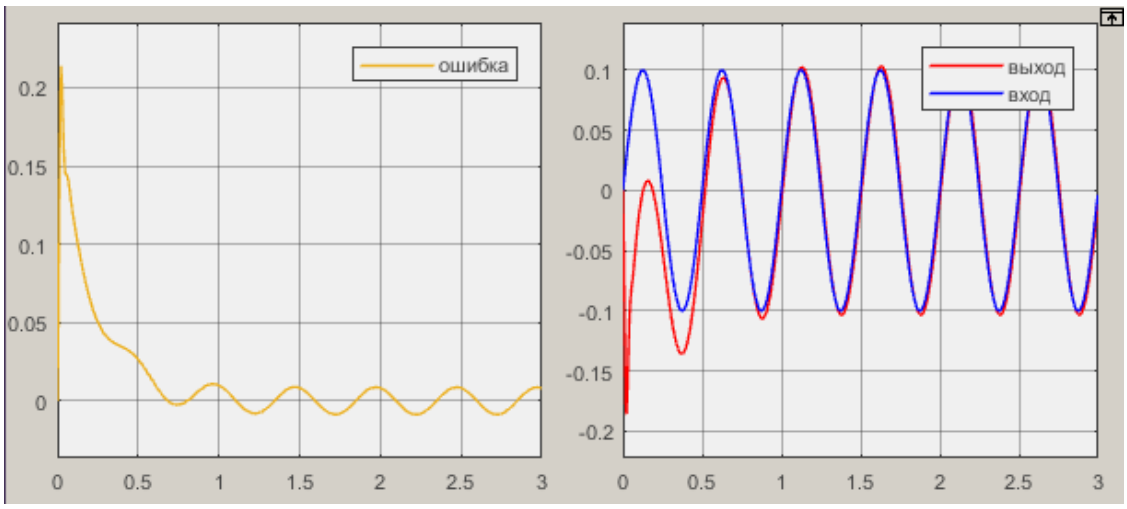


Рис.1. 9. Реакция системы на гармоническое входное воздействие

Для данного процесса характерны статическая ошибка 1.5 мрад, динамическая ошибка 9 мрад. Время регулирования составляет 0.25с, перерегулирование 3.5%.

Листинг m-файла, реализующего алгоритм эвристической настройки параметров ПИД-регулятора представлен на рис.1.10.

Важно! Перед запуском m-файла в Simulink-схеме необходимо учесть:

- на вход системы подается единичное ступенчатое воздействие;
- конечное время моделирования системы 0.3 с;
- SampleTime в блоке ToWorkspace составляет 0.0001;
- имя блок-схемы *tuning_const_PID.slx*.

Работа m-файла может быть остановлена в CommandWindow нажатием комбинации клавиш Ctrl+C.

```
% Требования ТЗ:-----------------------------------------------
% перерегулирование 0..5%
% статическая ошибка 0.1 мрад
%-----------------------------------------------------------------
Stat_error=0.0001;
Step_Kp=0.5;
Step_Kd=0.005;
Step_Ki=1;
%-----------------------------------------------------------------
sim('tuning_const_PID.slx')
if max(fi)<1.05 & abs(1-fi.Data(fi.Length))<Stat_error
        disp('Параметры регулятора удовлетворяю поставленным требованиям')

else
disp('Настройка пропорциональной составляющей ПИД-регулятора')
```

```
%при перерегулировании 20% в системе возникают колебания, затухающие за 2-3 периода
ifmax(fi)<1.2%если перерегулирование меньше 20% коэффициент Kp увеличивается
        while max(fi)<1.2
                Kp=Kp+Step_Kp;
        sim('tuning_const_PID.slx')
        end
else%если перерегулирование больше 20% коэффициент Kp уменьшается
        whilemax(fi)>1.2 &Kp>0    %при этом Kp не может быть отрицательной
        величиной
                Kp=Kp-Step_Kp;
        sim('tuning_const_PID.slx')
        end
        ifKp<=0      %выход из программы, если Kp не положительно
                disp('Ошибка!!!')
                keyboard
        end
end
Kp=Kp                   %параметр Kp найден!!!,
%-----------------------------------------------------------------
%  блок анализа переходного процесса
        [max_Value,i_max]=max(fi.Data(:));  %нахождение максимального значения и
    его позиции в ряде выходных данных
        [min_Value,i_min]=min(fi.Data(i_max:fi.Length)); %нахождение минимального значения и его позиции после максимальной величины
        disp('Настройка дифференциальной составляющей ПИД-регулятора')
while max_Value>min_Value        %приближение максимальной величины к последующей минимальной сводит перерегулирование к 0%
        Kd=Kd+Step_Kd;
        sim('tuning_const_PID.slx')
        [max_Value,i_max]=max(fi.Data(1:500));  %расчет максимальной величины
        проводится в диапазоне от 0 до 0.05с (от 1 до 500),
                т.е. времени нарастания
        [min_Value,i_min]=min(fi.Data(i_max:fi.Length));
end
Kd=Kd                   %параметр Kd найден!!!,
%-----------------------------------------------------------------
disp('Настройк интегральной составляющей ПИД-регулятора')
while abs(1-fi.Data(fi.Length))>Stat_error %модуль ошибки не должен превышать
        заданного значения в конечный момент времени
                Ki=Ki+Step_Ki;
```

```
                sim('tuning_const_PID.slx')
end
Ki=Ki           %параметр Ki найден!!!
%-----------------------------------------------------------------
end
```

Рис.10. Листинг программы, реализующей эвристические настройки

После подстройки параметры ПИД-регулятора принимают значения:

$K_p =..;\ K_d =..;\ K_i =..$

Примерный вид переходного процесса с новыми параметрами ПИД-регулятора представлен на рис.1.11.

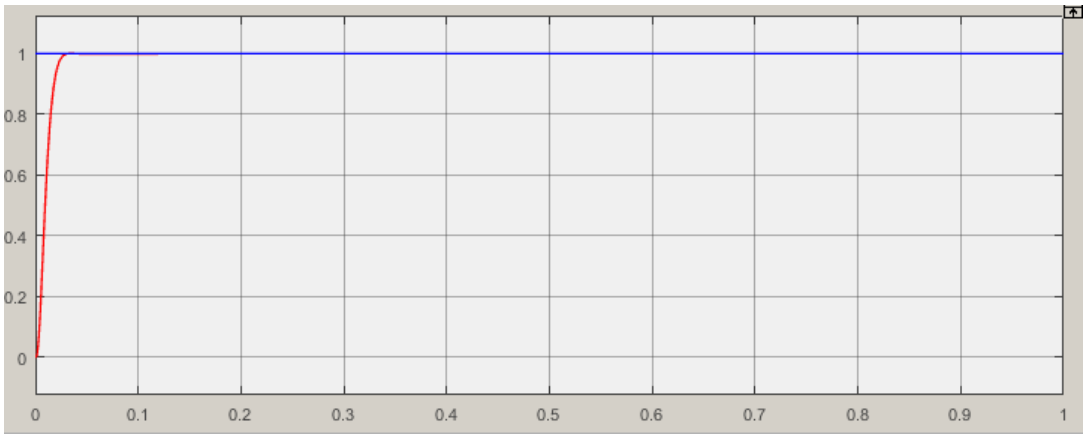


Рис. 1.11. Переходный процесс на выходе скорректированной системы

Анализ рассчитанного переходного процесса позволяет сделать вывод о том, что система с подстроенными параметрами имеет перерегулирование ≈ 0%, время регулирования 0.021 с, статическая ошибка не превышает 0.1 мрад, что удовлетворяет требованиям ТЗ по постоянному сигналу.

Проверим отработку системой гармонического сигнала (рис.1.12),т.е. на синусоидальный входной сигнал амплитудой 0.1 рад и частотой 2 Гц.

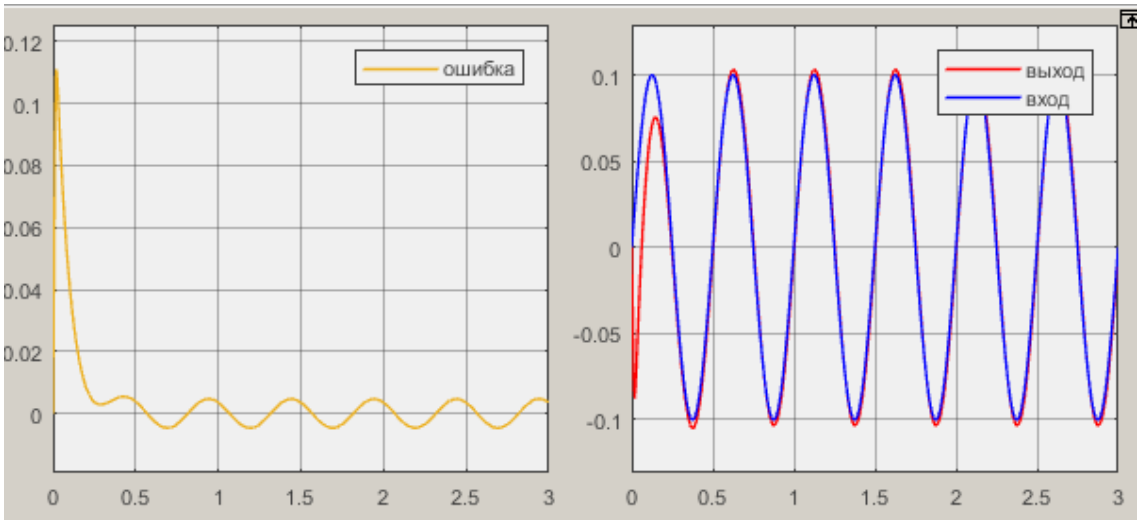


Рис. 1.12. Реакция системы на гармоническое входное воздействие

Из приведенных графиков видно, что динамическая ошибка составляет 4.7 мрад, значение которой превышает требований ТЗ.

Для улучшения работы системы при слежении за гармоническим воздействием и выполнения требований ТЗ дополним схему компенсацией динамической ошибки (рис.1.13).

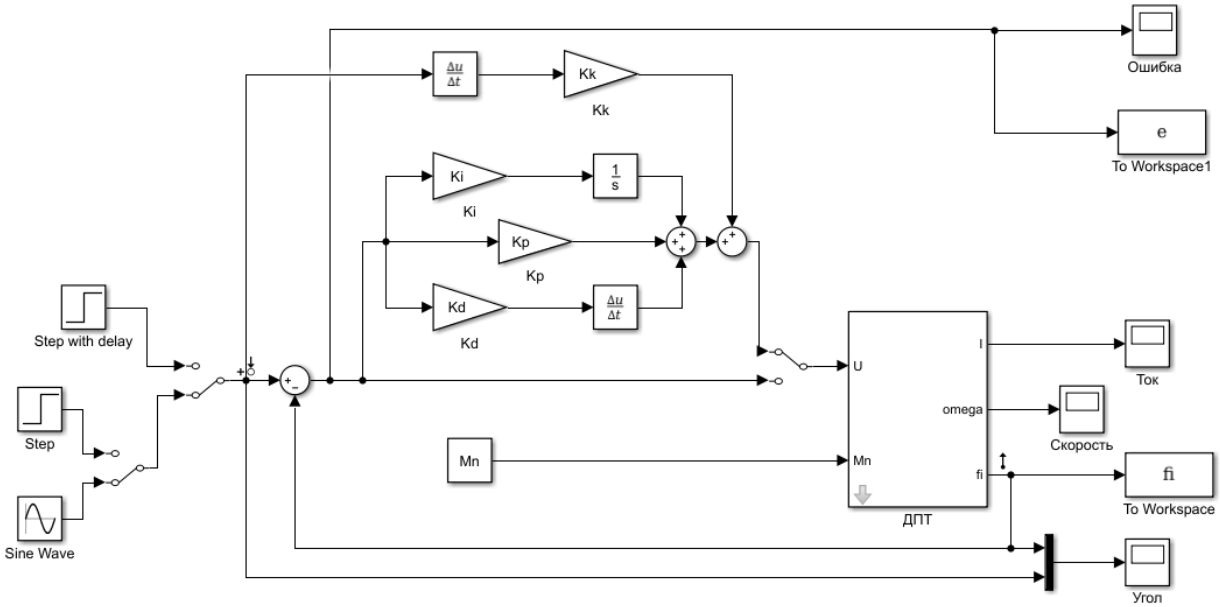


Рис. 1.13. Simulink-схема с компенсацией динамической ошибки

Такое дополнение учитывает скорость изменения входной координаты в формировании управляющего воздействия. Следует принять во внимание то, что скорость изменения ступенчатого сигнала равна 0 и данная составляющая не оказывает дополнительного влияния на систему.

Листинг m-файла, реализующий настройку параметра компенсации динамической ошибки приведен на рис.1.14.

Важно! Перед запуском m-файла в Simulink-схеме необходимо учесть:

- на вход системы подается гармоническое воздействие;
- конечное время моделирования системы 3 с;
- SampleTime в блоке ToWorkspace1 составляет 0.0001;
- имя блок-схемы tuning_const_PID.slx.
-

Ctrl+C в CommandWindow останавливает работу m-файла.

```
%-----------------------------------------------------------------------
% Требования ТЗ:
```

```
% динамическая ошибка 0.5 мрад.
%-----------------------------------------------------------------------
Kk=0;
Dyn_error=0.0005;
Step_Kk=0.001;
%-----------------------------------------------------------------------
sim('tuning_const_PID.slx')
[error,i]=max(e.Data(15000:e.Length));      %нахождение значения максимальной ошибки в диапазоне от 1.5 до 3с (режим слежения)
iferror<Dyn_error%если максимальная ошибка отработки гармонического воздействия меньше требуемой, то система не нуждается в дополнительной коррекции
    disp('Параметры регулятора удовлетворительны')
    else  %если ошибка больше требуемой, увеличиваем коэффициент Kk до удовлетворения заданным требованиям
        disp('Настройка параметра компенсации')
        while error>Dyn_error
            Kk=Kk+Step_Kk;
            sim('tuning_const_PID.slx')
        [error,i]=max(e.Data(15000:e.Length));
        end
    Kk=Kk          %параметр Ki найден!!!
%-----------------------------------------------------------------------
end
```

Рис.1.14. Листинг программы коррекции коэффициента компенсирующей связи

После настройки получаем:

$K_k =..$

Расчетные характеристики системы с компенсирующей связью представлены на рис.1.15.

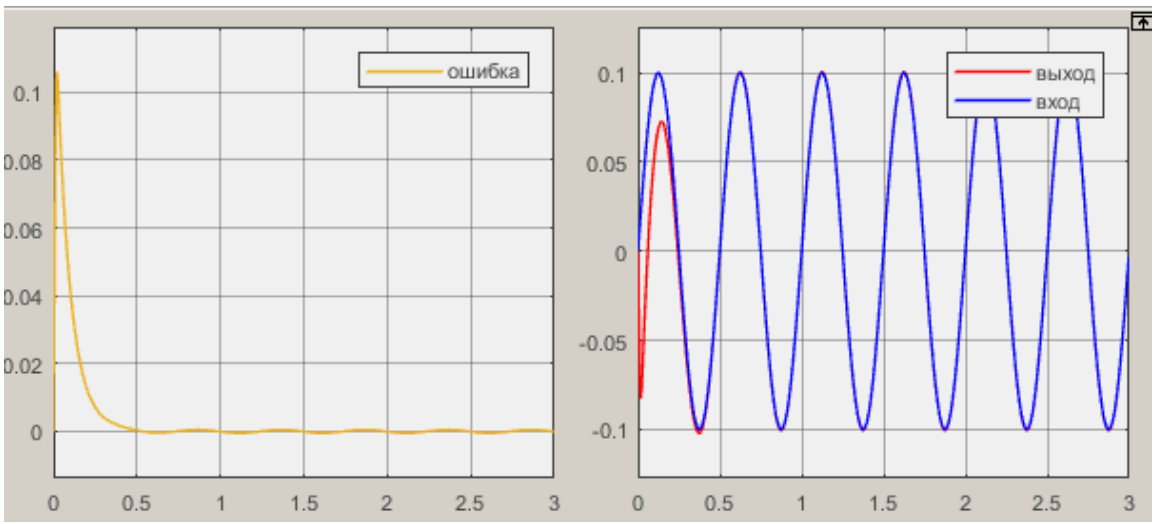


Рис. 1.15. Реакция системы на гармоническое входное воздействие

Величина динамической ошибки отработки гармонического воздействия не превышает 0.5 мрад, что удовлетворяет требованиям ТЗ.

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Для тестового варианта рассчитать методом Коуэна-Куна в первом приближении значения параметров ПИД-регулятора, затем с помощью экспертной системы провести коррекцию параметров регулятора и рассчитать коэффициент компенсирующей связи, которые обеспечивают выполнение требований технического задания.

Выполнить коррекцию ЭСП в соответствии с вариантом (табл. 4), заданным преподавателем.

Гармоническое входное воздействие — синусоидальный сигнал амплитудой 0.05 рад и частотой 2 Гц.

Схемы моделирования представлены на рис. 1.5-1.6 и 1.13.

Обеспечить выполнение требования ТЗ. При невозможности обеспечения заданной динамической ошибки определить минимальное ее значение, наблюдая за графиком ошибки при запуске программы.

Таблица 1.4

	Сопротивление обмотки якоря	Индуктивность обмотки якоря	Коэффициенты противо-ЭДС и по моменту	Момент инерции якоря	Приведенный момент нагрузки
	R	L		J	Mn
	Ом	мГн	Ce = Cm	кг·м^2	кг·м^2
1	2	3	4	5	6
1	0.9	0.9	0.055	$2.8\cdot10^{-5}$	0.01
2	0.475	0.475	0.053	$4.7\cdot10^{-5}$	0.1
3	0.0865	0.0865	0.095	$100\cdot10^{-5}$	0.5
4	0.82	1.23	0.081	$9\cdot10^{-5}$	0.1
5	0.6	0.6	0.062	$11.5\cdot10^{-5}$	0.05
6	1.08	1.62	0.051	$3\cdot10^{-5}$	0.01
7	0.325	0.4875	0.047	$10\cdot10^{-5}$	0.05
8	0.2	0.2	0.075	$33\cdot10^{-5}$	0.5
9	0.88	1.32	0.058	$2.1\cdot10^{-5}$	0.01
10	0.255	0.3825	0.071	$42\cdot10^{-5}$	0.1
11	0.53	0.53	0.0515	$4\cdot10^{-5}$	0.05
12	0.11	0.11	0.092	$130\cdot10^{-5}$	0.5
13	0.72	1.08	0.079	$11\cdot10^{-5}$	0.1
14	0.77	0.77	0.078	$9.5\cdot10^{-5}$	0.05
15	0.185	0.185	0.077	$35\cdot10^{-5}$	0.1
16	0.096	0.144	0.092	$170\cdot10^{-5}$	0.5
17	0.125	0.1875	0.096	$140\cdot10^{-5}$	0.5
18	0.42	0.42	0.06	$15\cdot10^{-5}$	0.05
19	0.85	0.85	0.082	$9.7\cdot10^{-5}$	0.05
20	1.2	1.2	0.0575	$3.9\cdot10^{-5}$	0.01

5. ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач, схемы моделирования, пояснение работы компенсирующей связи, текст программ автоматизированной настройки (экспертной системы), алгоритм настройки ПИД-регулятора и компенсирующей связи (в структуре комбинированного управления), логику выполнения настройки параметров ПИД-регулятора.

5. Контрольные вопросы

1. Что такое экспертная система?
2. Возможные варианты использования экспертной системы в контуре управления динамическим объектом?
3. Основные свойства и характеристики ПИД - регулятора?
4. Алгоритмы настройки ПИД-регулятора?
5. Зачем требуется выполнять дополнительную настройку ПИД-регулятора?
6. Каким образом коэффициенты ПИД - регулятора влияют на качество процессов управления?

Лабораторная работа№2

РАБОТА С НЕЧЕТКИМИ МНОЖЕСТВАМИ В СРЕДЕ MATLAB.

ПАКЕТ FUZZY LOGIC TOOLBOX

1. Цель и задачи работы

Знакомство с пакетом Fuzzy Logic Toolbox интегрированным в среде Matlab.
Задание функций принадлежности в пакете Fuzzy Logic Toolbox .

2. Теоретические сведения

ПАКЕТ FUZZY LOGIC TOOLBOX

В основу теоретического части данной лабораторной работы положен материал, изложенный в /Штовба/.

В вычислительную среду MATLAB интегрированы десятки пакетов прикладных инженерных и математических программ, одним из них является Fuzzy Logic Toolbox. В данном разделе представлен пакет Fuzzy Logic Toolboxv.2.1 вычислительной системы MATLAB 7, предназначенный для проектирования и исследования систем с нечеткой логикой. Материал организован следующим образом: вначале рассматриваются основные возможности пакета, затем показывается пошаговый процесс проектирования систем нечеткого вывода с применением GUI-модулей пакета и из командной строки, далее приводятся руководства пользователя по GUI-модулям, и функции пакета. В конце рассматриваются форматы данных пакета и описывается взаимодействие Fuzzy Logic Toolbox с Simulink. При написании главы использовалась документация фирмы MathWorksInc.

СТРУКТУРА И ВОЗМОЖНОСТИ ПАКЕТА

Пакет FuzzyLogicToolbox поддерживает все фазы разработки нечетких систем, включая синтез, исследование, проектирование, моделирование и внедрение в режиме реального времени. Встроенные GUI-модули пакета создают интуитивно понятную среду, обеспечивающую легкое продвижения по всем ступенькам проектирования нечетких систем. Функции пакета реализуют большинство современных нечетких технологий, включая нечеткий логический вывод, нечеткую кластеризацию и адаптивную нейро-нечеткую настройку (ANFIS). FuzzyLogicToolbox, как и все пакеты расширения системы MATLAB, открыт для пользователя: можно просмотреть алгоритмы, модифицировать исходный код, добавить собственные функции принадлежности или процедуры дефаззификации.

Ключевыми особенностями пакета Fuzzy Logic Toolbox являются:

1. специализированные GUI-модули для создания систем нечеткого вывода;
2. реализация популярных алгоритмов нечеткого вывода Мамдани и Сугено;
3. библиотека функций принадлежности;
4. настройка функций принадлежности ANFIS-алгоритмом;
5. экстракция нечетких правил с помощью кластеризации данных;
6. возможность внедрения систем нечеткого вывода в Simulink через модуль «нечеткий контроллер»;
7. Си-код алгоритмов нечёткого вывода, позволяющий использовать спроектированные нечёткие системы вне среды MATLAB.

FuzzyLogicToolbox включает следующие GUI-модули (рис.2.1):

1. **FuzzyInferenceSystemEditor** - редактор общих свойств системы нечеткого вывода. Позволяет установить количество входов и выходов системы, выбрать тип системы (Мамдани или Сугено), метод дефаззификации, реализации логических операций, а также вызвать другие GUI-модули, работающие с системами нечеткого вывода;

2. **MembershipFunctionEditor** - редактор функций принадлежности. Редактор выводит на экран графики функций принадлежности входных и выходных переменных. Позволяет выбрать количество термов для лингвистической оценки входных и выходных переменных, а также задать тип и параметры функций принадлежности каждого терма;

3. **RuleEditor** - редактор нечеткой базы знаний. Позволяет задавать и редактировать нечеткие правила в лингвистическом, логическом и индексном форматах. Редактирование правил осуществляется выбором необходимого сочетания термов из меню;

4. **RuleViewer** - браузер нечеткого вывода. Визуализирует выполнение нечеткого вывода по каждому правилу, получение результирующего нечеткого множества и его дефаззификацию;

5. **SurfaceViewer** - браузер поверхности «входы — выход» нечеткой системы. Выводит графики зависимости выходной переменной от любых двух входных переменных;

6. **ANFISEditor** - редактор нейро-нечеткой сети. Позволяет синтезировать и настроить нейро-нечеткие сети по выборке данных «входы - выход». Для настройки используется алгоритм обратного распространения ошибки или его комбинация с методом наименьших квадратов;

7. **Findcluster** - инструмент субтрактивной кластеризации по горному методу и нечеткой кластеризации по алгоритму нечетких с-средних. Позволяет найти центры кластеров данных, которые используются для экстракции нечетких правил.

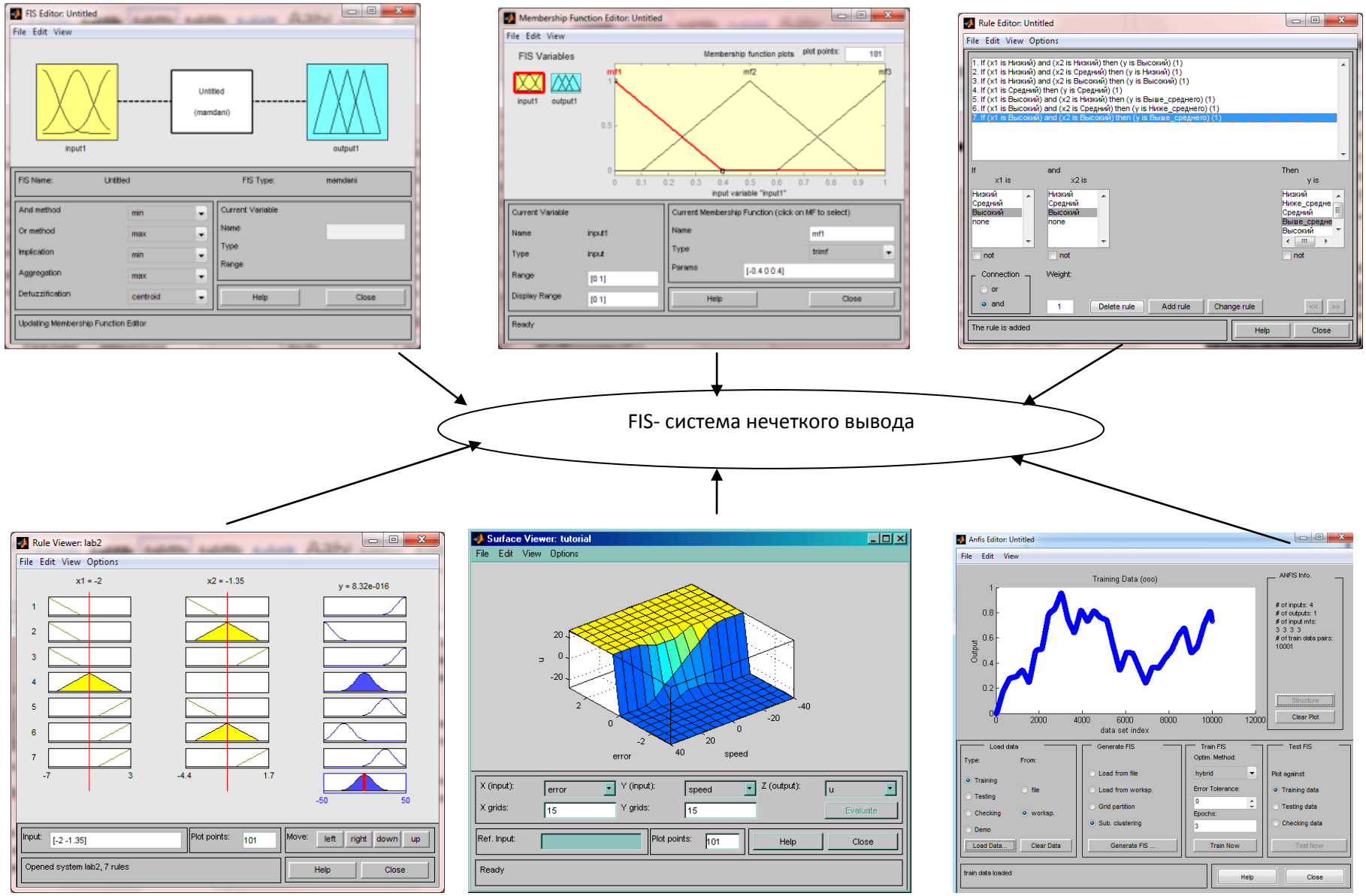


Рис. 2.1. Взаимодействие GUI-модулей нечеткого вывода

Все GUI-модули, за исключением Findcluster, динамически обмениваются данными и могут быть вызваны друг из друга (рис.2.1).

Кроме GUI-модулей, FuzzyLogicToolbox содержит библиотеку функций для разработки нечетких систем из командной строки, а также для написания программ, автоматизирующих проектирование и исследование нечетких систем. Ниже приведены названия и краткое описание основных функций пакета (аббревиатура FIS обозначает систему нечеткого вывода) (таблица 2.1):

Таблица 2.1

Название функции	***Краткое описание***
addmf	добавление функции принадлежности в FIS
addrule	добавление правил в FIS
addvar	добавление переменной в FIS
anfis	настройка FIS типа Сугено с помощью технологии ANFIS
convertfis	преобразование FIS-матрицы из FuzzyLogicToolboxv.l в FIS-структуру для FuzzyLogicToolboxv.2
defuzz	дефаззификация нечеткого множества
discfis	дискретизация функций принадлежности нечетких термов из FIS
distfcm	расчет расстояния по Евклиду
dsigmf	функция принадлежности в виде разности двух сигмоидных функций
evalfis	выполнение нечеткого логического вывода
evalmf	вычисление значений произвольной функции принадлежности
evalmmf	расчет степеней принадлежностей для нескольких функций принадлежностей
fcm	кластеризация алгоритмом нечетких с-средних
fuzarith	нечеткий калькулятор
gauss2mf	двухсторонняя гауссова функция принадлежности
gaussmf	гауссова функция принадлежности

gbellmf	обобщенная колоколообразная функция принадлежности
genfi 1	генерирование из данных исходной FIS типа Сугено без применения кластеризации
genfis2	енерирование из данных исходной FIS типа Сугено через субтрактивную кластеризацию
genparam	генерирования начальных параметров функций принадлежности для ANFIS-обучения
gensurf	вывод поверхности «входы - выход», соответствующей FIS
getfis	получение свойств FIS
initfis	генерирование исходной матрицы нечеткого разбиения
isfis	проверка структуры данных системы нечеткого вывода
mam2sug	преобразование FIS типа Мамдани в FIS типа Сугено
mf2mf	пересчет параметров встроенных функций принадлежности различных типов
newfis	создание новой FIS
parsrule	вставка в FIS правил, заданных предложениями на естественном языке
pimf	Пи-подобная функция принадлежности
plotfis	вывод основных параметров FIS в виде графической схемы
plotmf	вывод графиков функций принадлежности термов одной переменной
probor	вероятностная реализация логической операции ИЛИ
psigmf	функция принадлежности в виде произведения двух сигмоидных функций
readfis	загрузка FIS из файла
rmmf	удаление функции принадлежности нечеткого терма из FIS
rmvar	удаление переменной из FJS
setfis	назначение свойств FIS
howfis	вывод на экран FIS-структуры в текстовом формате
showrule	вывод базы знаний FIS

sigmf	сигмоидная функция принадлежности
smf	s-подобная функция принадлежности
subclust	субтрактивная кластеризация
sugmax	расчет диапазона изменения выходной переменной в FIS типа Сугено
trapmf	трапециевидная функция принадлежности
trimf	треугольная функция принадлежности
writefis	сохранение FIS на диске
zmf	z-подобная функция принадлежности

Пакет FuzzyLogicToolbox позволяет внедрить разработанные системы нечеткого вывода в динамические модели пакета Simulink. Для этой цели служит pimulink-модуль FuzzyLogicController - нечеткий контроллер. Для быстрого нечеткого вывода в пакете Simulink оптимизирован код функции sffis, возможности рэторой аналогичны функции evalfis. С использованием Real-TimeWorkshop Можно сгенерировать эффективный код нечеткого вывода.

3. Задание на выполнение

В редакторе функций принадлежности **MembershipFunctionEditor** сформировать функции принадлежности, соответствующие значениям лингвистической переменной (или лингвистическим переменным), определенным преподавателем из таблицы 2.2.

Варианты переменной «А»

№ п/п	NB	NL	NS	Z	PS	PL	PB
1.	trapmf	trimf	trapmf	trimf	trapmf	trimf	trapmf
2.	trapmf	trimf	trimf	trimf	trimf	trimf	trapmf
3.	trapmf	trimf	trapmf	trimf	trapmf	trimf	trapmf
4.	trimf	trimf	trimf	trimf	trimf	trimf	trimf
5.	trimf	trimf	trapmf	trimf	trapmf	trimf	trimf

Варианты переменной «B»

№ п/п	NS	Z	PS
1.	trapmf	trimf	trapmf
2.	trimf	trimf	trimf
3.	trapmf	trimf	trapmf
4.	trimf	trimf	trimf
5.	trapmf	trimf	trapmf

К определению численных значений функций принадлежности

№ п/п	NB	NL	NS	Z	PS	PL	PB
1.	-100	-50	-25	0	25	50	100
2.	0	10	20	30	40	50	60
3.	-20	-12	-7,5	0	7,5	12	20
4.	-0,1	-0.05	-0.02	0	0,02	0,05	0,1
5.	-60	-50	-40	-30	-20	-10	0

№ п/п	NS	Z	PS
1.	-0.01	0	0.01
2.	-1	0	1
3.	0	1	2
4.	-0,001	0	0,001
5.	1	2	3

Требуется:

- сформировать границы областей значений аргумента функций принадлежности, в которых указанные функции принадлежности отличны от нуля.
- изобразить границы в виде соответствующих диаграмм.
- с помощью FIS-редактора построить заданные преподавателем функции принадлежности.

1. Отчет

Отчет должен содержать краткое описание пакета FuzzyLogicToolbox, описание заданных лингвистических переменных, значения лингвистических переменных, разбиение диапазона значений аргумента лингвистических переменных (носителя) на области, в которых соответствующие функции принадлежности отличны от нуля.

Вид функций принаддежности, сформированных с помощью пакета FuzzyLogicToolbox.

2. Контрольные вопросы

1. Что такое лингвистическая переменная?
2. Что такое терм-множество?
3. Какие типовые функции принадлежности Вы знаете?
4. Что нечеткое разбиение?
5. Опишите структуру библиотек пакета FuzzyLogicToolbox?

Лабораторная работа№3

РАЗРАБОТКА НЕЧЕТКОЙ СИСТЕМЫ ТИПА МАМДАНИ

1. Цель и задачи работы

Создание нечеткой системы Мамдани. Изучение методики применения нечеткого вывода Мамдани для аппроксимации нелинейной функции двух переменных.

2. Теоретические сведения

Предположим, что мы имеем некоторую логическую модель вывода с несколькими входами (размерность входного вектора X равна n) и одним выходом Y.

В соответствие с теорией нечетких множеств, каждой i-ой входной переменной (x_i) соответствует своя лингвистическая переменная (a_i), значениями которой являются определенные термы, например a_{ij}, где j – индекс соответствующего терма i-ой переменной.

Выходной переменной Y соответствует терм- множество d_j.

База знаний нечеткого вывода по алгоритму Мамдани может быть представлена совокупностью продукционных правил:

$$(x_1 = a_{1j}\ \Theta_j\ \ x_2 = a_{2j}\ \ \Theta_j ....\Theta_j\ \ x_n = a_{n1j}\ \text{ с весом }\ w_j)\ \ \Rightarrow Y = d_j,$$

в которой значения входных и выходной переменной заданы нечеткими множествами. Всего таких правил m, т.е $j = \overline{1,m}$. Символ Θ_j обозначает одну из двух возможных логических операций (операция И обозначается $\wedge$ и операция ИЛИ обозначается $\vee$. Вес в правиле вывода указывает степень доверия эксперту, сформировавшего данное правило.

Обозначим нечеткие термы:

для входной переменной $a_{ij} = \int_{\underline{x_i}}^{\overline{x_i}} \mu_{ij}(x_i)/x_i$;

для выходной переменной $d_j = \int_{\underline{y}}^{\overline{y}} \mu_{dj}(y)/y$.

Степень выполнения посылки j-ого правила для текущего входного вектора $X^* = \left(x_1^*, x_2^*, ..., x_n^*\right)$ определяется одним из двух уравнений, в зависимости от используемой операции (И или ИЛИ) в $j_{\text{-ом}}$ правиле вывода.

Для операции И используется минимума (MIN-операция):

$$\mu_j(X^*) = MIN(\mu_{1j}(x_1^*), \mu_{2j}(x_2^*), ..., \mu_{nj}(x_n^*)) \quad j = \overline{1,m}.$$

Для операции ИЛИ используется минимума (MAX-операция):

$$\mu_j(X^*) = MAX(\mu_{1j}(x_1^*), \mu_{2j}(x_2^*), ..., \mu_{nj}(x_n^*)) \quad j = \overline{1,m}.$$

В результате получаем нечеткое множество

$$y^* = \left(\frac{\mu_1(X^*)}{d_1}, \frac{\mu_2(X^*)}{d_2}, \ldots\ldots, \frac{\mu_m(X^*)}{d_m} \right),$$

соответствующее входному вектору X^*.

Особенностью этого нечеткого множества является то, что универсальным множеством для него является терм-множество выходной переменной Y.

Сформированное таким образом нечеткое множество определяет степень исполнения каждого правила и фактически его вклад в формирование выходной переменной.

Для перехода от нечеткого множества, заданного на универсальном множестве нечетких термов $\{d_1, d_2, \ldots\ldots, d_m\}$ к нечеткому множеству на носителе Y ($\underline{y}, \overline{y}$) необходимо выполнить операции:

1)импликации - "срезать" функции принадлежности $\mu_{dj}(y)$ на уровне $\mu_j(X^*)$; 2) объединить (агрегировать) полученные нечеткие множества. Операция агрегирования нечетких множеств математически записывается следующим образом:

$$y^* = agg(d_1, d_2, \ldots.. d_m),$$

где agg - агрегирование нечетких множеств, которое наиболее часто реализуется операцией нахождения максимума (MAX).

Четкое значение выхода y, соответствующее входному вектору X^* определяется в результате деффаззификации нечеткого множества y. Наиболее часто применяется дефаззификация по методу центра тяжести.

3. Задание на выполнение

Разработать модель нечеткого вывода в среде MatLab для приближенного расчета значений функции $y = x_1^2 \sin(x_2 - 1)$ в области x_1 [-7, 3], x_2 [4.4, 1.7].

В соответствие с общим алгоритмом построения модели нечеткого вывода первым этапом такого построения должно являться анализ значений рассматриваемой функции в заданном диапазоне. Для формирования визуального образа функции воспользуемся программой, представленной на рис.3.1.

Проектирование осуществим на основе трехмерного изображения указанной зависимости (рис.3.1), которое построено следующей программой:

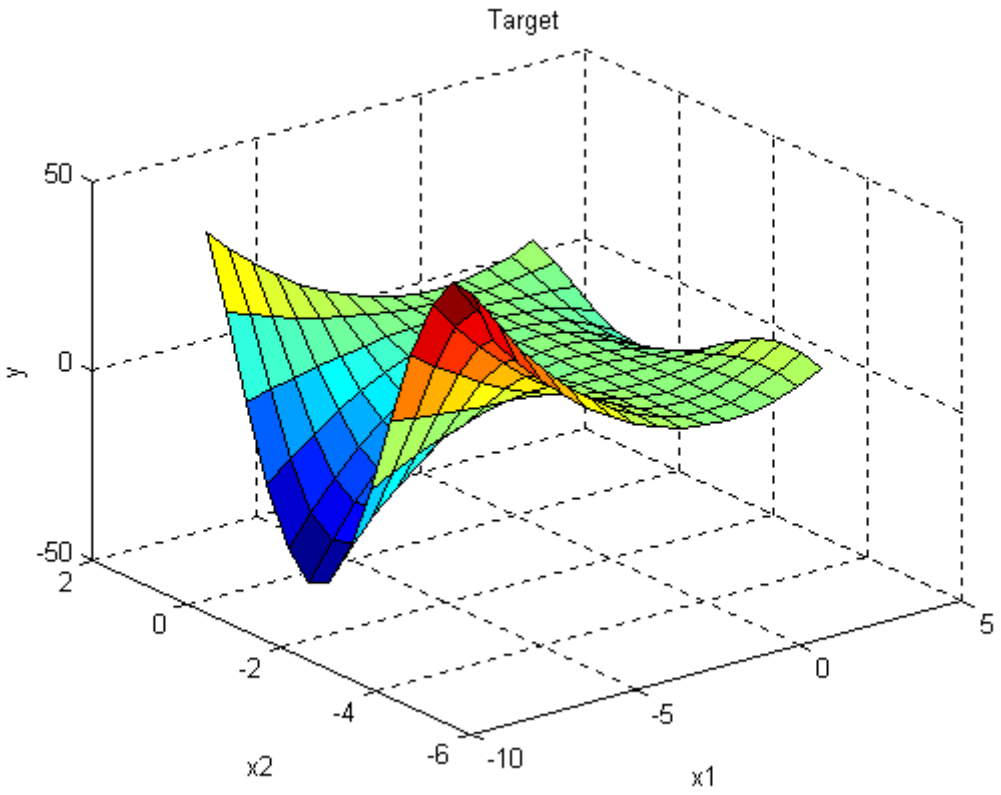


Рис. 3.1 График функции $y = x_1^2 \sin(x_2-1)$.

```
%Построение графика функции y=x1^2*sin(x2-1)
n=15;
x1=-7:10/(n-1):3;
x2=-4.4:6.1/(n-1):1.7;
y=zeros(n,n);
for j=1:n
y(j,:)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

Для формирования правил нечеткого вывода разобьем полученное графическое изображение функции на семь уровней, что соответствует семи значениям лингвистической переменной используемой для качественной оценки исследуемой функции.

Поверхности на рис.3.1 поставим 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 >> = «ниже среднего»;

7. ЕСЛИ x_1 = «высокий» И x_2 = «высокий», ТО y = «выше среднего».

Проектирование нечеткой системы состоит в выполнении следующей последовательности шагов:

Шаг 1. Открыть FIS-редактор, напечатав слово fuzzy в командной строке. После этого появится новое графическое окно.

Шаг 2. Добавим вторую входную переменную. Для этого в меню **Edit** выбираем команду **Add input.**

Шаг 3. Переименуем первую входную переменную. Для этого сделаем щелчок левой кнопкой мыши на блоке inputl, введем новое обозначение xl в поле редактирования имени текущей переменной и нажмем <Enter>.

Шаг 4. Переименуем вторую входную переменную. Для этого щелкнем мышкой на блоке input2, введем новое обозначение x2 в поле редактирования имени текущей переменной и нажмем <Enter>, результат на рис. 3.2.

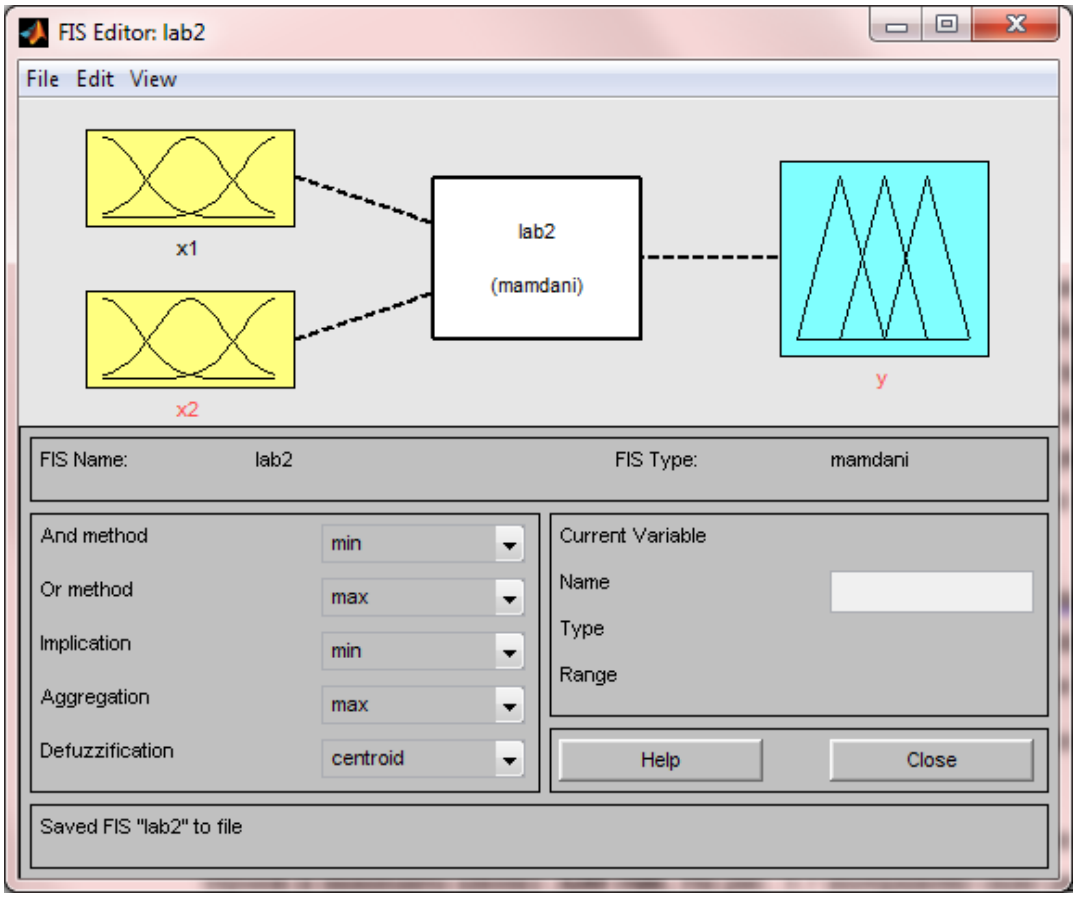


Рис. 3.2 FIS-редактор

Шаг 5. Переименуем выходную переменную. Для этого щелкнем на блоке outputl, введем новое обозначение y в поле редактирования имени текущей переменной и нажмем <Enter>.

Шаг 6. Зададим имя системы. Для этого в меню **File** выберем в подменю **Export** команду To **Disk...**

Шаг 7. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке xl.

Шаг 8. Зададим диапазон изменения переменной $x_{1г}$ напечатав -7 3 в поле Range (рис. 3.3).

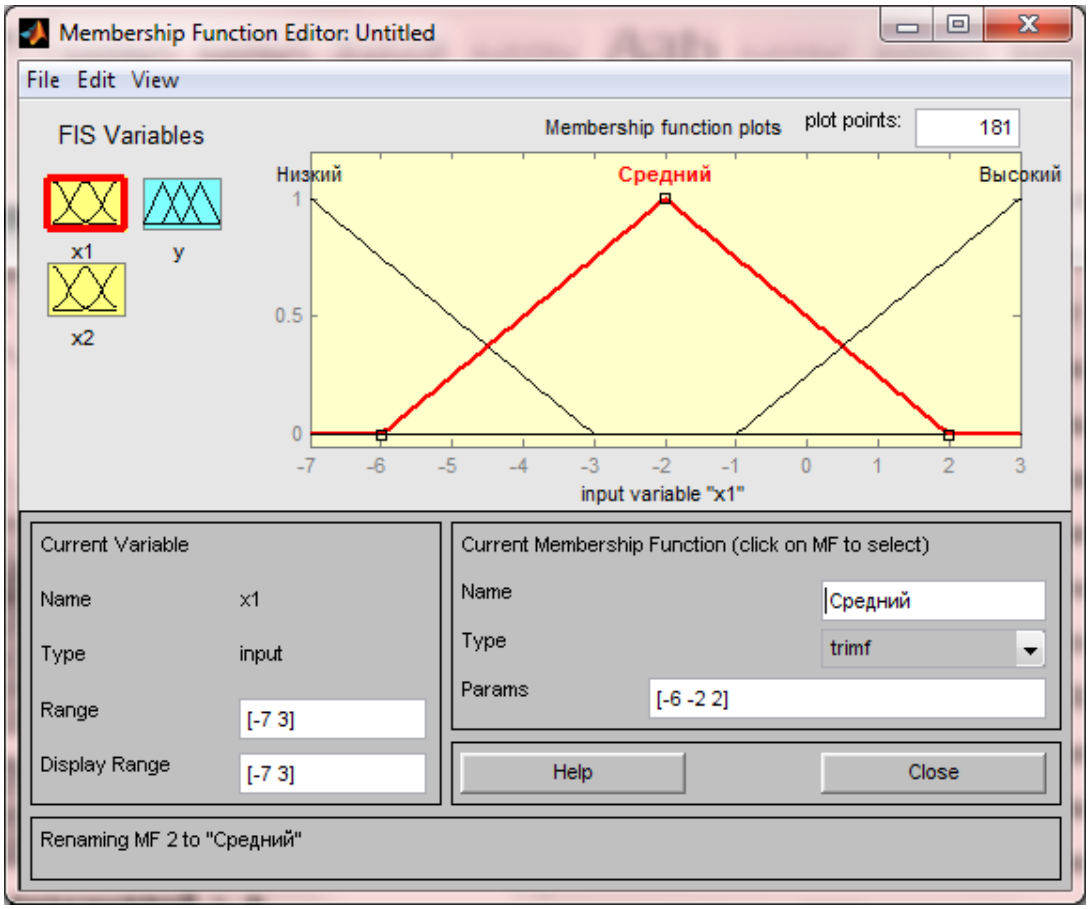


Рис. 3.3 Функция принадлежности переменной x1 в Membership Function Edition

Шаг 9. Зададим функции принадлежности переменной x,. Для лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Эти функции установлены по умолчанию, поэтому переходим к следующему шагу.

Шаг 10. Зададим наименования термов переменной *x,*. Для этого щелкнем мышкой по графику первой функции принадлежности (см. рис.3.4). График активной функции принадлежности красной жирной линией. Затем введем наименование **Низкий** в поле Name и нажмем **<Enter>.** Щелкнем мышкой по графику второй функции принадлежности, введем наименование терма **Средний** в поле Name и нажмем **<Enter>.** Щелкнем мышкой по графику третьей функции принадлежности, введем наименование терма **Высокий** в поле Name и нажмем **<Enter>.** В результате получим графическое окно, изображенное на рис. 3.3.

Шаг 11. Зададим функции принадлежности переменной x_2. Для этого активизируем переменную x_2 щелчком мышкой по блоку x2. Зададим диапазон изменения переменной $x_г$. Для этого напечатаем -4.4 1.7 в поле Range (рис.3.4) и нажмем **<Enter>.** Для

лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Они установлены по умолчанию, поэтому переходим к следующему шагу.

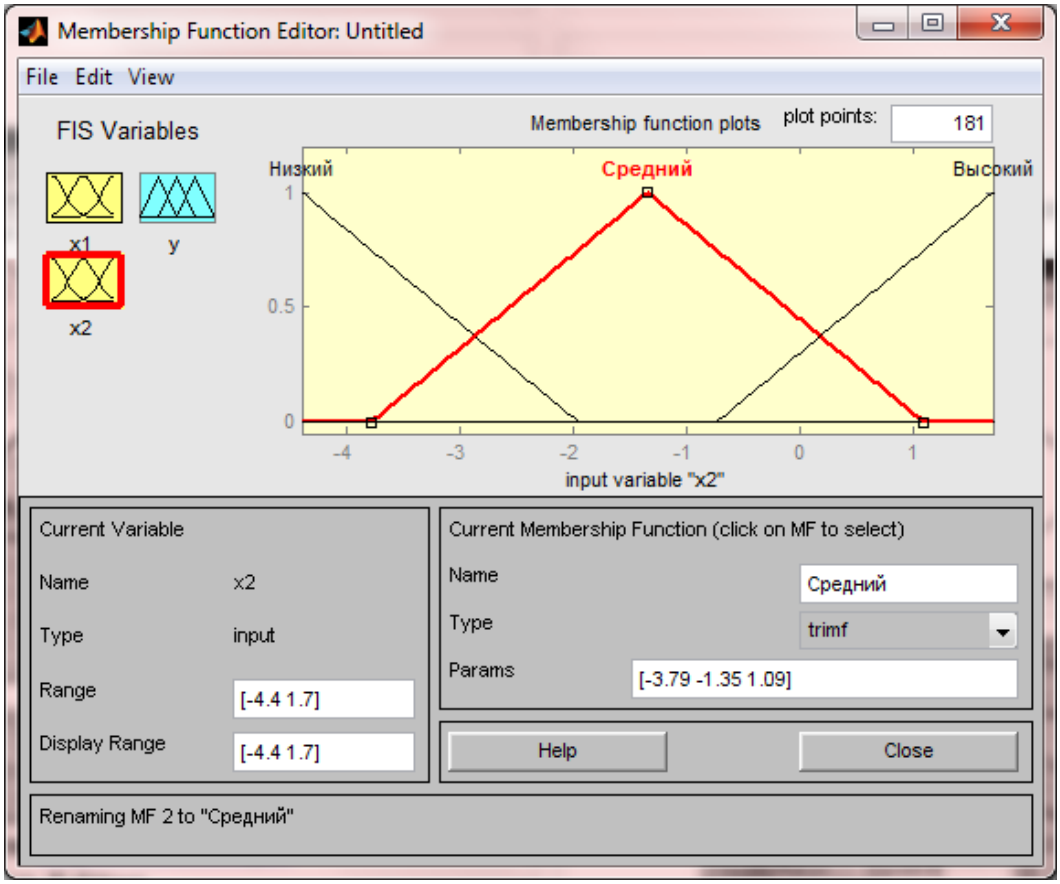


Рис. 3.4 Функция принадлежности переменной x2 в Membership Function Edition

Шаг 12. По аналогии *с шагом 10* зададим следующие наименования термов переменной x_2: *Низкий, Средний, Высокий* (рис 3.5).

Шаг 13. Зададим функции принадлежности переменной. Для лингвистической оценки этой переменной будем использовать пять термов с гауссовыми функциями принадлежности. Для этого щелчком мыши по блоку y активизируем переменную ***y.*** Зададим диапазон изменения переменной ***y.*** Для этого напечатаем -50 50 в поле **Range** (рис. 3.6) и нажмем <Enter>. Затем в меню **Edit** выберем команду **Remove All MFs** для уцаления установленных по умолчанию функций принадлежности. После этого в меню **Edit** выберем команду **Add MFs**...В появившемся диалоговом окне выберем тип функции принадлежности gaussmf в поле MF type и пять термов в поле Number of MFs. После ввода функций принадлежности редактор активизирует первую входную переменную, поэтому для продолжения работы щелкнем мышкой по пиктограмме y.

Шаг 14. По аналогии *с шагом 10* зададим следующие наименования термов переменной *y:* Низкий, Ниже среднего, Средний, Выше среднего, Высокий. В результате получим графическое окно, изображенное на рис. 3.5.

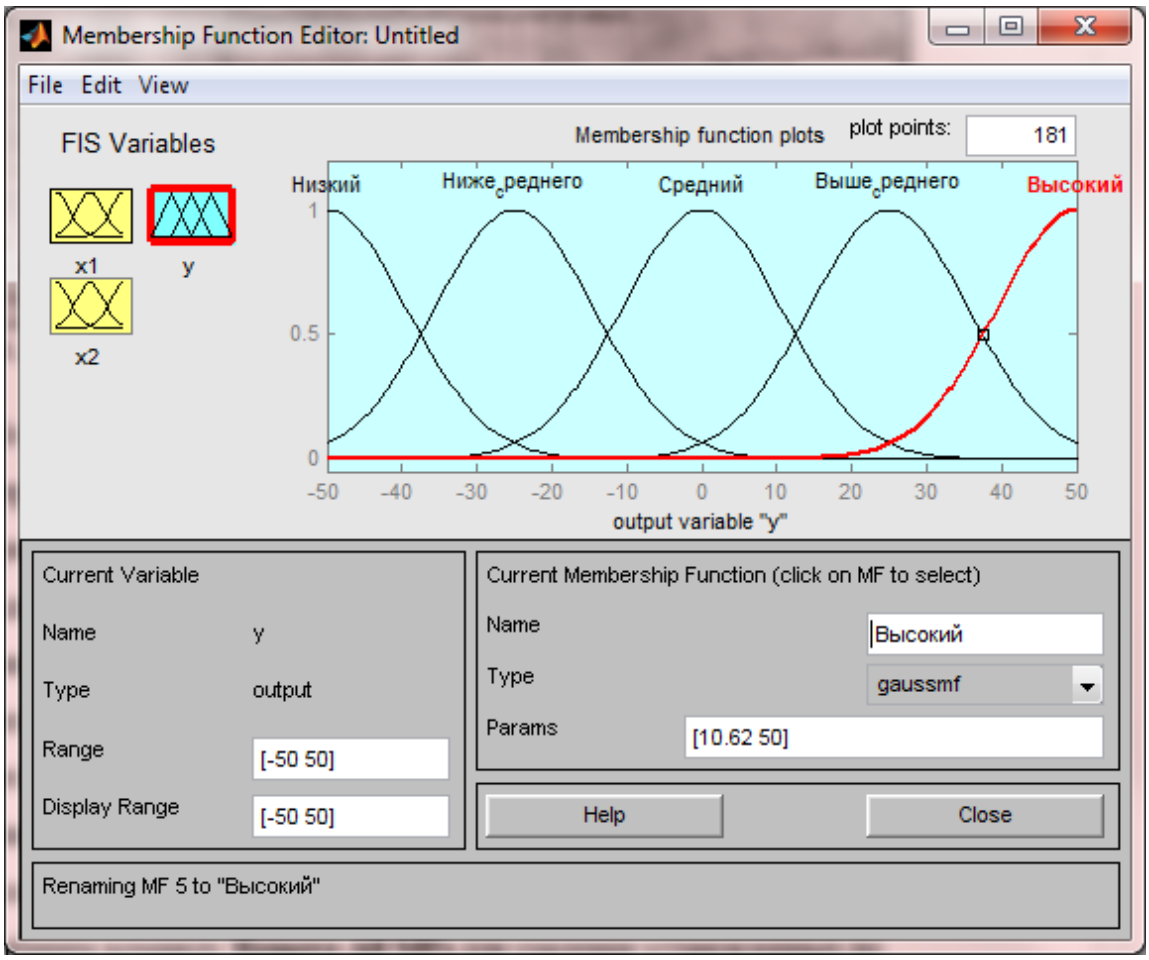


Рис. 3.5 Функция принадлежности переменной y в Membership Function Edition

Шаг 15. Перейдем в редактор базы знаний RuleEditor. Для этого в меню Edit выберем команду **Rules....**

Шаг 16. Для ввода правила выбираем в меню соответствующую комбинацию термов и нажимаем кнопку **Add rule.** На рис. 3.6 изображено окно редактора базы знаний после ввода всех семи правил. В конце правил в скобках указаны весовые коэффициенты.

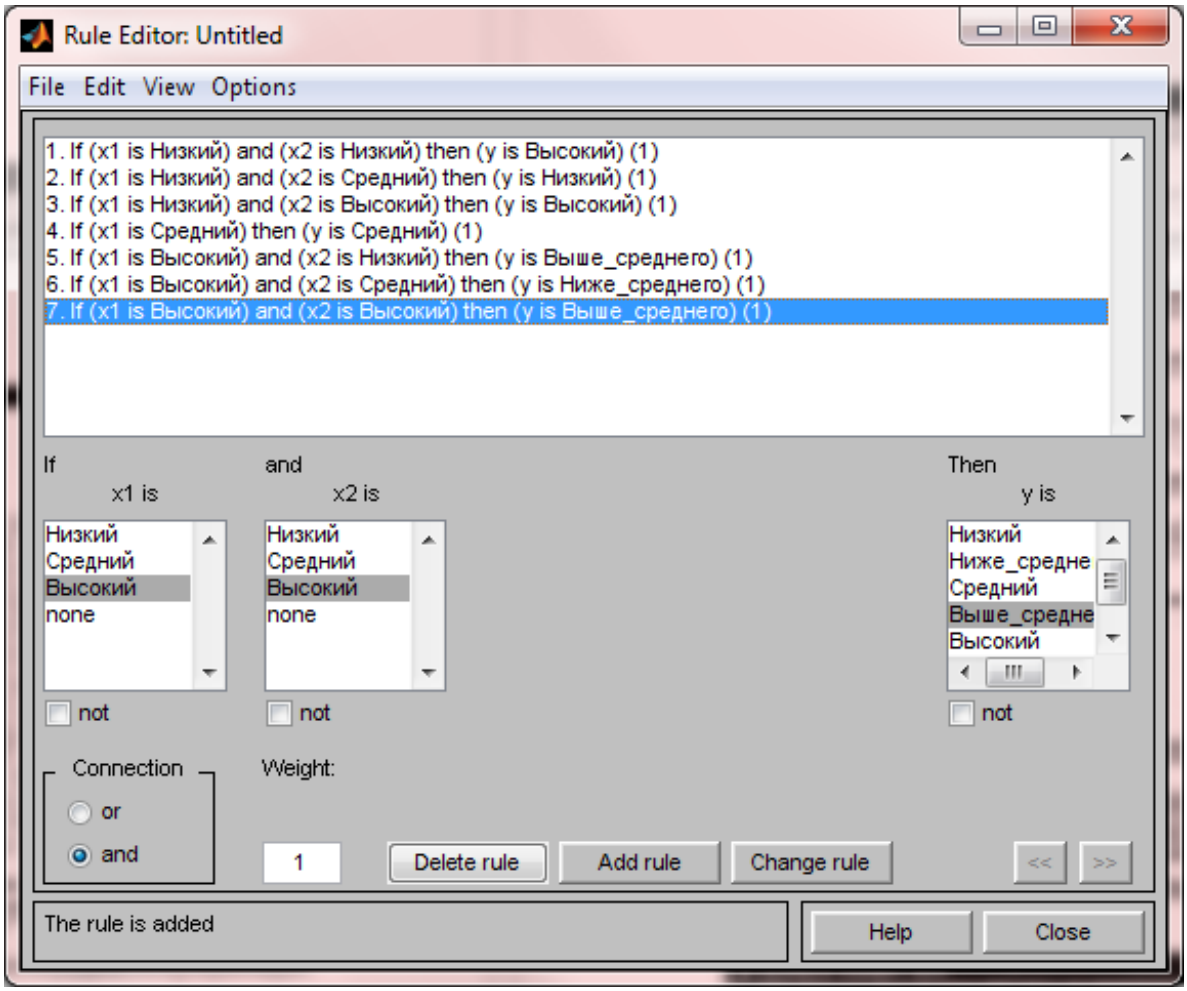


Рис. 3.6 База нечетких правил в Rule editor

Шаг 17. Сохраним созданную систему. Для этого в меню **File** выберем в подменю **Export** команду **To disk.** На рис. 3.7 приведено окно визуализации нечеткого вывода. Окно активизируется командой **Rules** меню **View**. В поле **Input** указываются значения входных переменных, для которых выполняется нечеткий логический вывод.

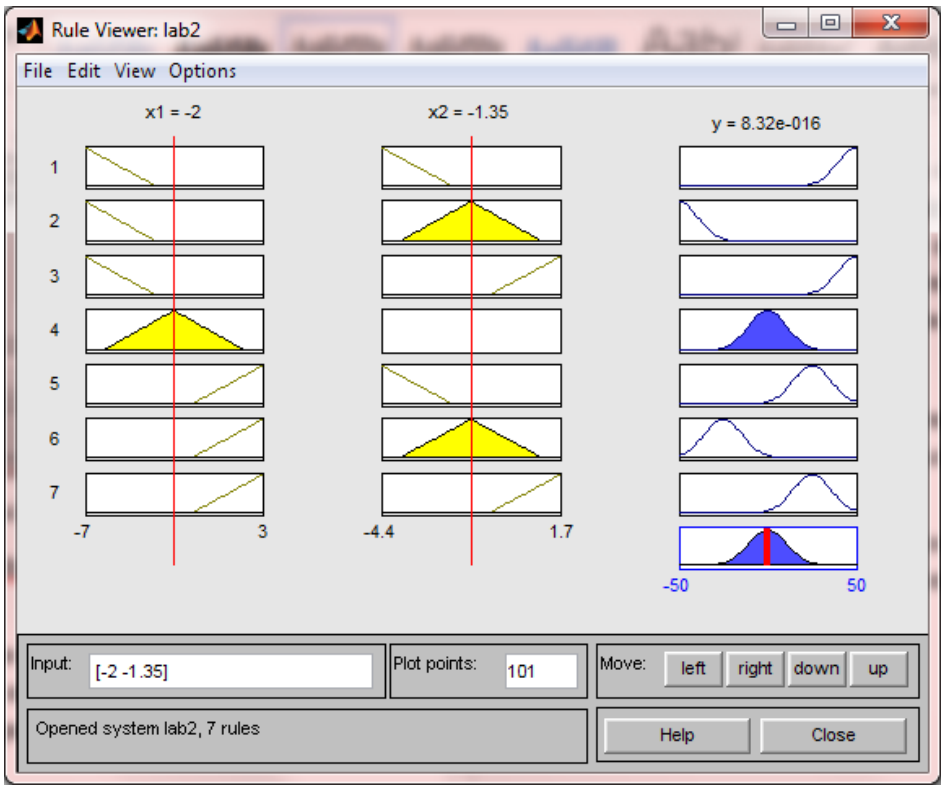


Рис. 3.7 Реализация нечеткого вывода Мамдани в Rule viewer

На рис. 2.8 приведена поверхность «входы — выход», соответствующая синтезированной нечеткой системе. Окно выводится по команде **Surface** меню **View**.

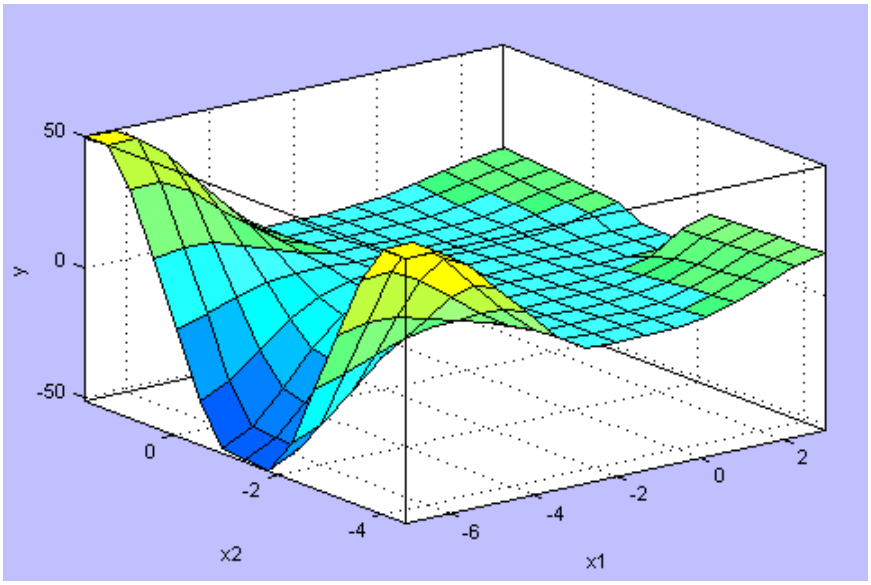


Рис. 3.8 Поверхность «вход-выход» для базы правил Мамдани

Сравнивая поверхности на рис. 2.1 и на рис. 2.8, можно сделать вывод, что нечеткие правила описывают особенности моделируемой нелинейной зависимости.

4.ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач. Сравнительный анализ результатов лабораторных работ № 2 и 3.

5.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое система нечеткого вывода?
2. Чем обусловлена разница поверхностей управления?
3. Для чего нужно окно Rule editor?

Лабораторная работа №4

РАЗРАБОТКА НЕЧЕТКОЙ СИСТЕМЫ ТИПА СУГЕНО НА ОСНОВЕ ЭКСПЕРТНЫХ ЗНАНИЙ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Создать систему нечеткого вывода типа Сугено.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Нечеткий логический вывод по алгоритму Сугено (иногда говорят алгоритм Такаги-Сугено) выполняется по нечеткой базе знаний:

База знаний нечеткого вывода по алгоритму Мамдани может быть представлена совокупностью продукционных правил:

$$(x_1 = a_{1j}\ \Theta_j\ \ x_2 = a_{2j}\ \ \Theta_j ....\Theta_j\ \ x_n = a_{n1j}\ \text{ с весом }\ w_j)\ \ \Rightarrow,$$

$$\Rightarrow y = b_{j0} + b_{j1}\cdot x_1 + b_{j2}\cdot x_2 + ... + b_{jn}\cdot x_n, \qquad j = \overline{1,m}$$

где b_{ji} - некоторые числа.

База знаний Сугено аналогична базе знаний Мамдани за исключением заключений правил d_j, которые задаются не нечеткими термами, а линейной функцией от входов :

$$d_j = b_{j0} + b_{j1}\cdot x_1 + b_{j2}\cdot x_2 + ... + b_{jn}\cdot x_n, \qquad j = \overline{1,m}.$$

Правила в базе знаний Сугено являются своего рода переключателями с одного линейного закона "входы - выход" на другой, тоже линейный. Границы подобластей размытые, следовательно, одновременно могут выполняться несколько линейных законов, но с различными степенями.

Степень выполнения посылки j-ого правила для текущего входного вектора $X^* = \left(x_1^*, x_2^*, ..., x_n^*\right)$ определяется одним из двух уравнений, в зависимости от используемой операции (И или ИЛИ) в $j_{\text{-ом}}$ правиле вывода.

Для операции И используется минимума (MIN-операция):

$$\mu_j(X^*) = MIN(\mu_{1j}(x_1^*), \mu_{2j}(x_2^*), ..., \mu_{nj}(x_n^*)) \quad j = \overline{1,m}.$$

Для операции ИЛИ используется минимума (MAX-операция):

$$\mu_j(X^*) = MAX(\mu_{1j}(x_1^*), \mu_{2j}(x_2^*), ..., \mu_{nj}(x_n^*)) \quad j = \overline{1,m}.$$

В результате получаем нечеткое множество

$$y^* = \left(\frac{\mu_1(X^*)}{d_1}, \frac{\mu_2(X^*)}{d_2}, ..., \frac{\mu_m(X^*)}{d_m}\right),$$

соответствующее входному вектору X^*.

Обратим внимание, что в отличие от результата вывода Мамдани, приведенное выше нечеткое множество является обычным нечетким множеством первого порядка. Оно задано на множестве четких чисел. Результирующее значение выхода y определяется как суперпозиция линейных зависимостей, выполняемых в данной точке X^* n-мерного факторного пространства.

Для этого дефаззифицируют нечеткое множество $\tilde{y}$, находя

взвешенное среднее $$y = \frac{\sum_{j=1,m} \mu_j(X^*) \cdot d_j}{\sum_{j=1,m} \mu_j(X^*)}$$

или взвешенную сумму $$y = \sum_{j=1,m} \mu_j(X^*) \cdot d_j$$.

3. Задание на выполнение

Смоделировать зависимость $y = x_1^2 \sin(x_2 - 1)$ в области x_1 [-7, 3], x_2 [4.4, 1.7]. Проектирование осуществим на основе трехмерного изображения указанной зависимости (рис.4.1), которое построено следующей программой:

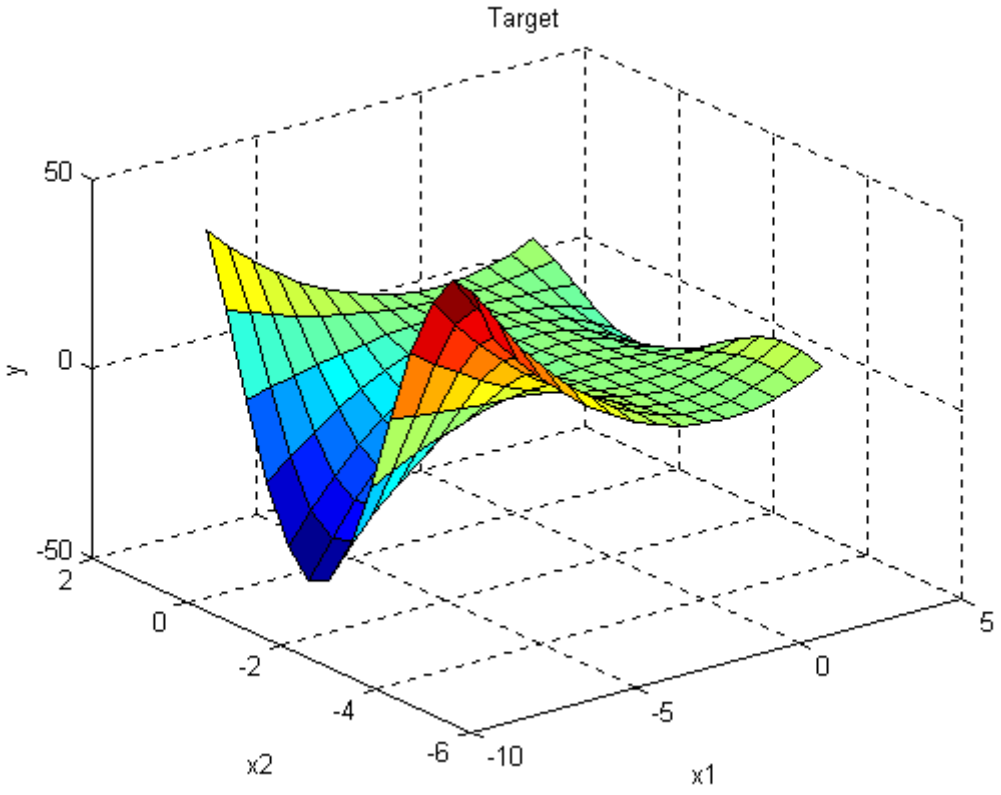


Рис. 4.1 График функции $y = x_1^2 \sin(x_2-1)$

8. ЕСЛИ x_1= «низкий» И x_2 = «средний», ТО $y = 50$;
9. ЕСЛИ x_1= «низкий» И x_2 = «средний», ТО $y = 4x_1 - x_2$;
10. ЕСЛИ x_1 = «низкий» И x_2 = «высокий», ТО $y = 50$;
11. ЕСЛИ x_1 = «средний», ТО $y = 0$;

12. ЕСЛИ x_1= «высокий» И x_2 = «низкий», ТО $y = 2x_1 - 2x_2 - 3$;

13. ЕСЛИ x_1= «высокий» И x_2 = «высокий», ТО $y = 2x_1 + 2x_2 + 1$;

Создание системы нечеткого вывода типа Сугено состоит в выполнении следующей последовательности шагов.

Шаг 1. Загрузим FIS-редактор, напечатав слова fuzzy в командной строке.

Шаг 2. Выберем тип системы. Для этого в меню **File** выбираем в подменю **New FIS**... команду **Sugeno.**

Шаг 3. Добавим вторую входную переменную. Для этого в меню Edit выберем команду **Add input**.

Шаг 4. Переименуем входы и выход системы. Для этого выполним *шаги 3-5* предыдущего алгоритма.

Шаг 5. Зададим имя системы. Для этого в меню **File** выберем в подменю **Export** команду **To disk** и введем имя файла, например, second.

Шаг 6. Зададим терм-множества и функции принадлежности входных переменных. Для этого выполним *шаги 7-12* предыдущего алгоритма.

***Шаг* 7.** Зададим заключения правил. Для этого щелчком мыши по блоку y активизируем переменную *y*. В правом верхнем угле появилось обозначение трех функций принадлежности, каждая из которых соответствует одной линейной зависимости между входами и выходом. В базе знаний Сугено указаны пять различных зависимостей. Поэтому добавим еще два заключения правил, выбрав из меню **Edit** команду **Add Mfs**... Затем в появившемся диалоговом окне в поле **Number of MFs** выберем **2** и нажмем кнопку **ОК**.

Шаг 8. Зададим наименования и параметры линейных зависимостей. Для этого щелкнем мышкой по наименованию первого заключения **mfl**. Затем в поле **Name** печатаем наименование зависимости, например, 50, и устанавливаем тип зависимости - константа через опцию **Constant** в меню **Type**. После этого введем значение параметра 50 в поле **Params**.

Аналогично, для второго заключения **mf2** введем наименование, например, 4 x 1 - x 2, укажем линейный тип зависимости «входы — выход» через опцию **Linear** в меню **Type** и введем параметры зависимости 4 -1 0 в поле **Params**. Для линейной зависимости порядок параметров следующий: первый параметр — коэффициент при первой переменной, второй — при второй и т.д., и последний параметр — свободный член зависимости.

Для третьего заключения **mf3** введем наименование, например, 0, укажем тип зависимости - константа и введем параметр 0.

Для четвертого заключения mf4 введем наименование, например, 2x1-2x2-3, укажем линейный тип зависимости и введем параметры 2 -2 -3..

Для пятого заключения mf5 введем наименование, например,

2x1+2x2+1, укажем линейный тип зависимости и введем параметры 2 2 1. В результате получим графическое окно, изображенное на рис. 4.2.

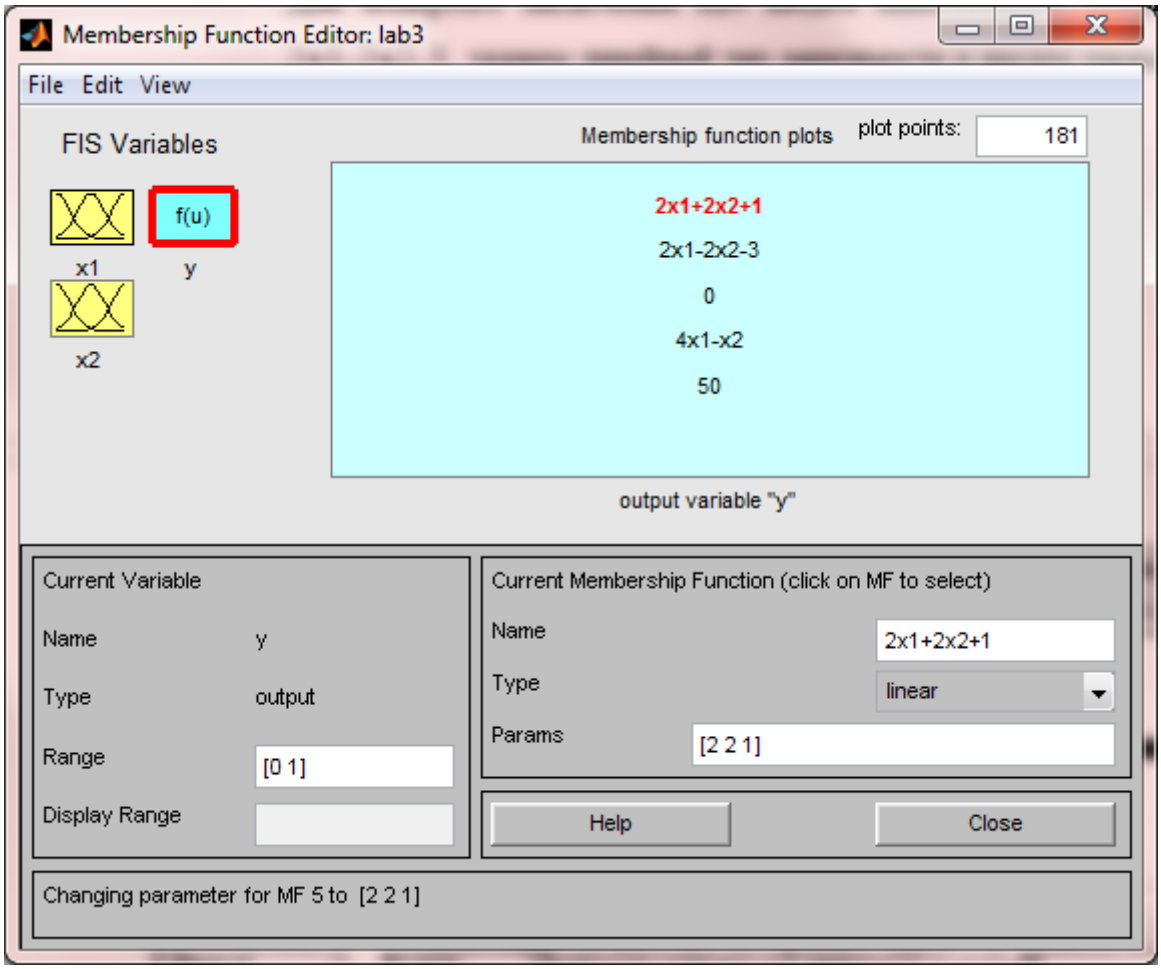


Рис. 4.2 Заключение правил Сугено в Membership Function

Шаг 9. Перейдем в редактор базы знаний Rule Editor через команду **Rules...** меню **Edit.** Для ввода правил необходимо выбрать соответствующую комбинацию посылок и заключений правил и нажать кнопку **Add rule**. На рис. 4.3 изображено окно редактора базы знаний после ввода всех шести правил.

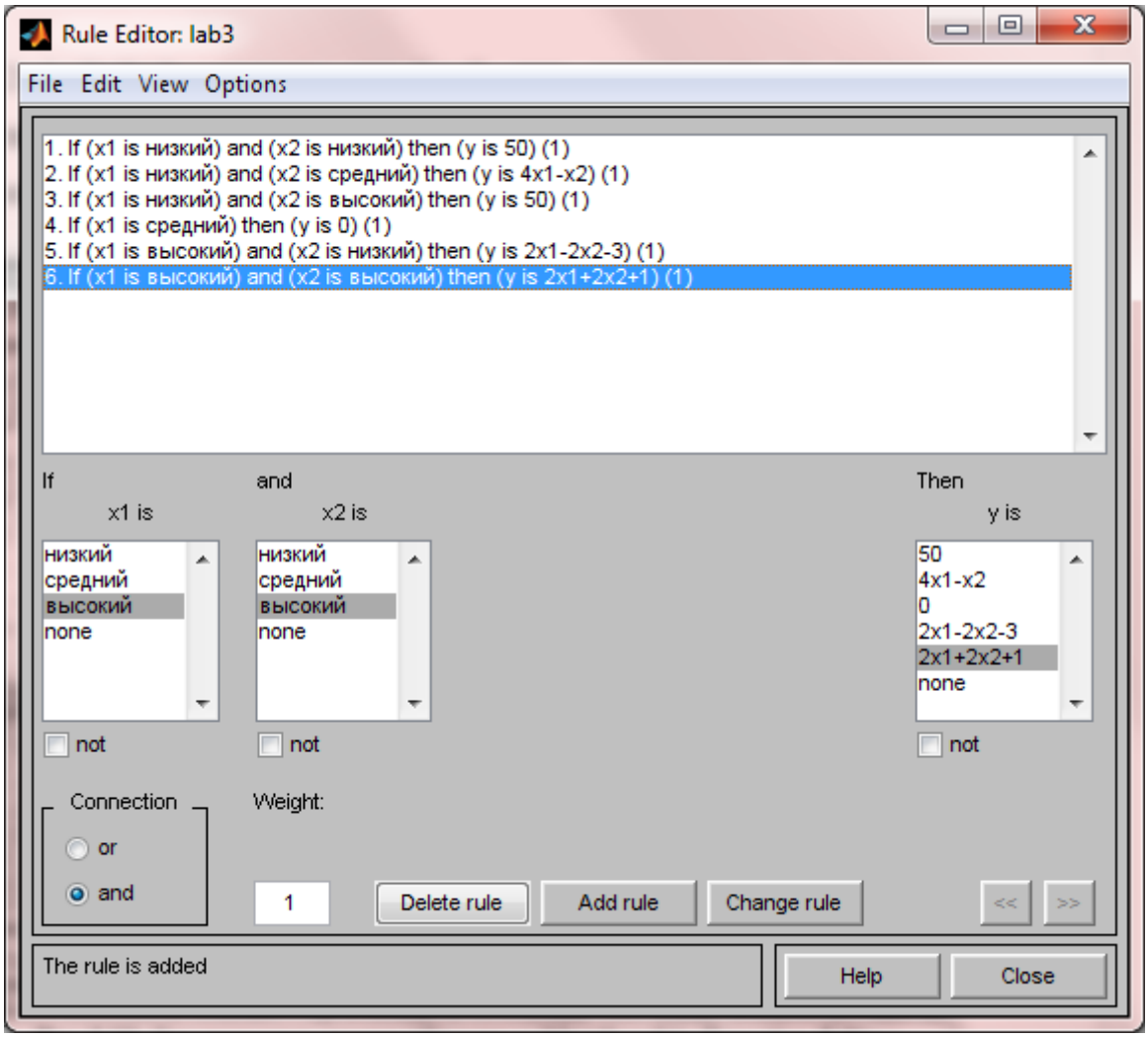


Рис. 4.3 Нечеткая база знаний Сугено

На рис. 4.4 приведено окно визуализации нечеткого вывода. Окно активизируется командой **Rules** меню **View**. На рис. 4.5 изображена поверхность «входы - выход», соответствующая синтезированной нечеткой системе. Окно выводится по команде **Surface** меню **View.**

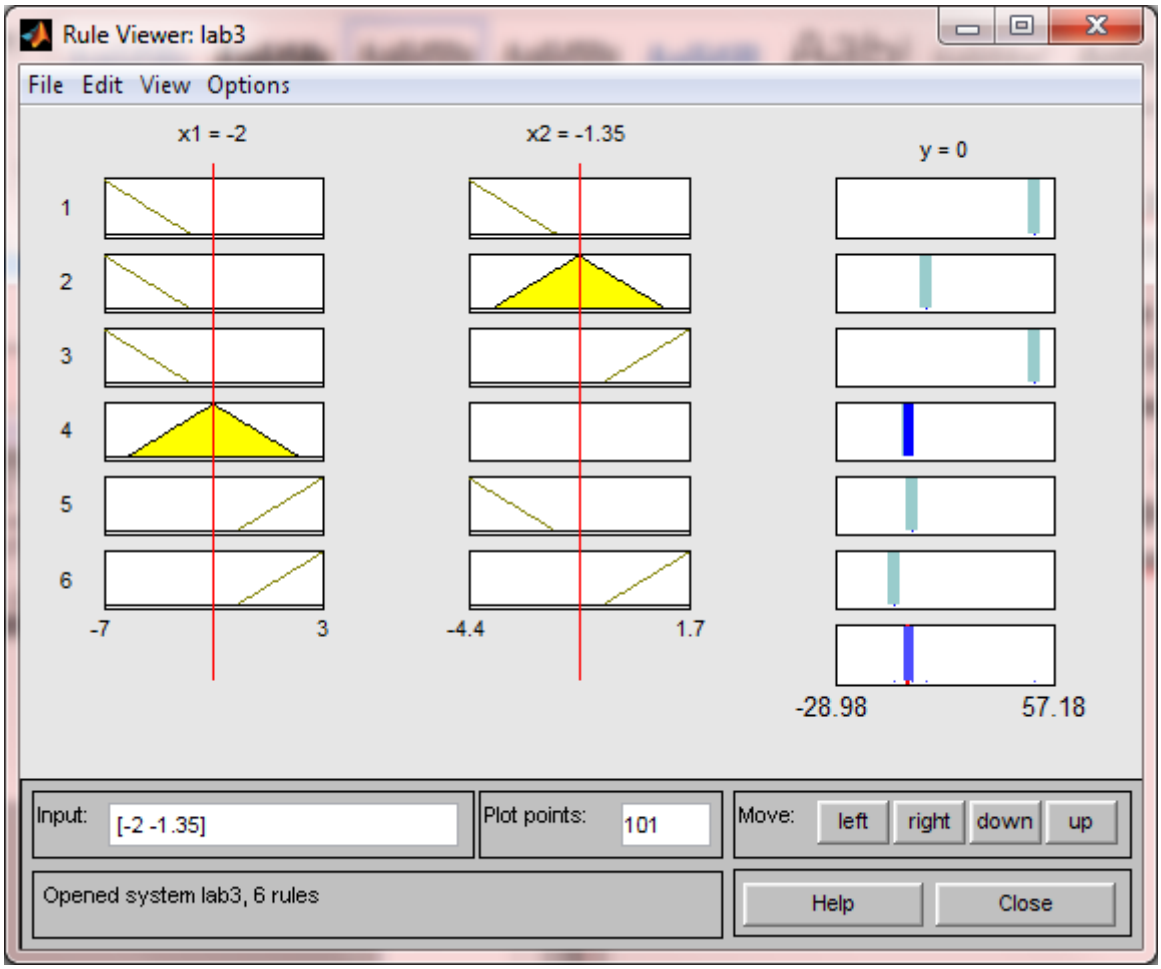


Рис. 4.4 Визуализация нечеткого вывода Сугено в Rule Viewer

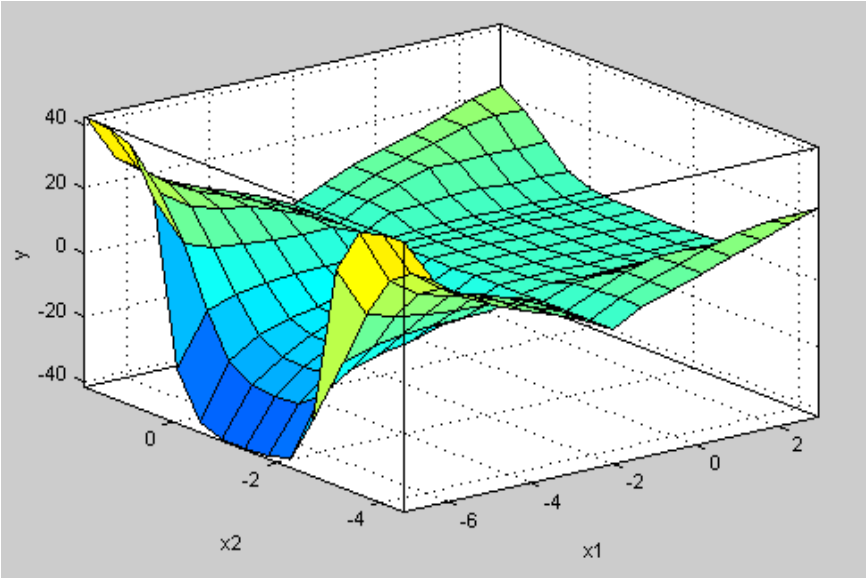


Рис. 4.5 Визуализация нечеткого вывода Сугено в Rule Viewer

Сравнивая поверхности на рис. 3.1, рис. 3.8 и рис. 4.5, можно сделать вывод, что нечеткие правила достаточно хорошо описывают сложную нелинейную зависимость. Модель типа Сугено более точная, однако подобрать подходящие заключения правил не всегда просто.

4.ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач.

5.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Будет ли работать система, если при задание правил использовать не все переменные?
2. Чем обусловлена разница поверхностей управления?
3. Для чего нужно окно Rule viwer?

Лабораторная работа№5

ПРОЕКТИРОВАНИЕ НЕЧЕТКОГО РЕГУЛЯТОРА ДЛЯ ЭСП С ИСПОЛНИТЕЛЬНЫМ ДВИГАТЕЛЕМ ПОСТОЯННОГО ТОКА

1. Цель и задачи работы

Синтез нечеткого регулятора для управления электрическим следящим приводом постоянного тока.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Функциональная схема привода представлена на рис. 5.1 (а) и (б). Параметры исполнительного двигателя: Се=См=0.732, J =0.05, Те =0.00234с, R = 0,96 Ом, коэффициент передачи редуктора: Кр=10.

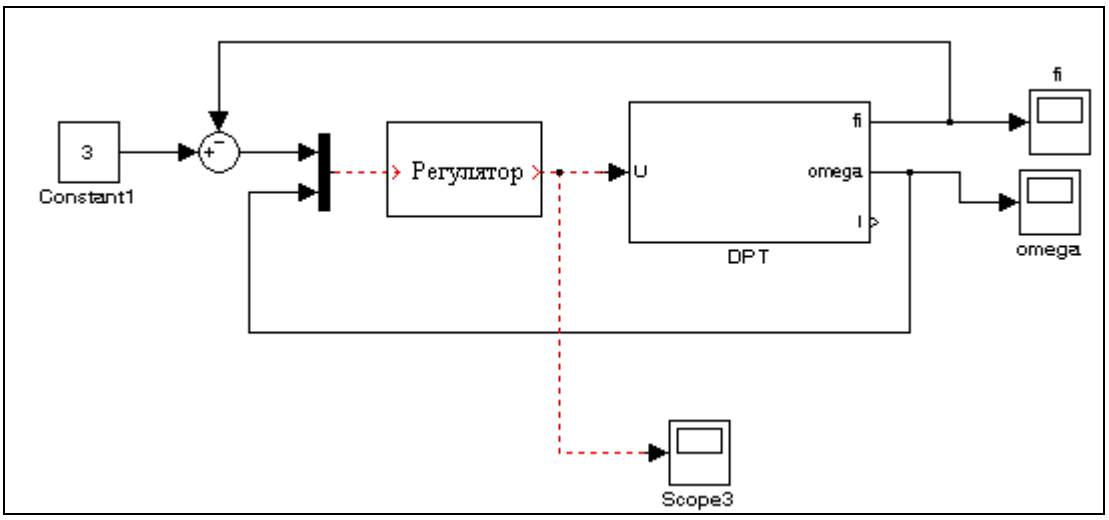


Рис. 5.1 (а) Структурная схема ЭСП с нечетким регулятором

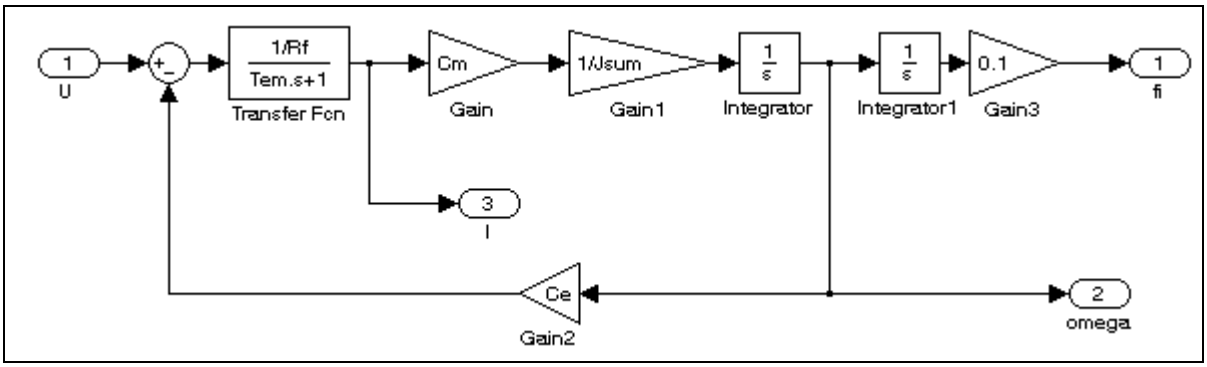


Рис. 5.1 (б) Структурная схема двигателя постоянного тока, подсистемы DPT

Допустим требуется построить нечеткий регулятор для управления приводом в диапазоне изменения угла поворота выходного вала +/- 1 рад. с использованием в качестве входных сигналов – ошибки отработки входного воздействия и скорости вращения вала исполнительного двигателя.

С целью формирования блока правил (накопления знаний об объекте управления), рассмотрим переходный процесс по скорости вращения вала и углу поворота вала двигателя, при подаче максимального управляющего напряжения +/- 27 В (рис. 5.2).

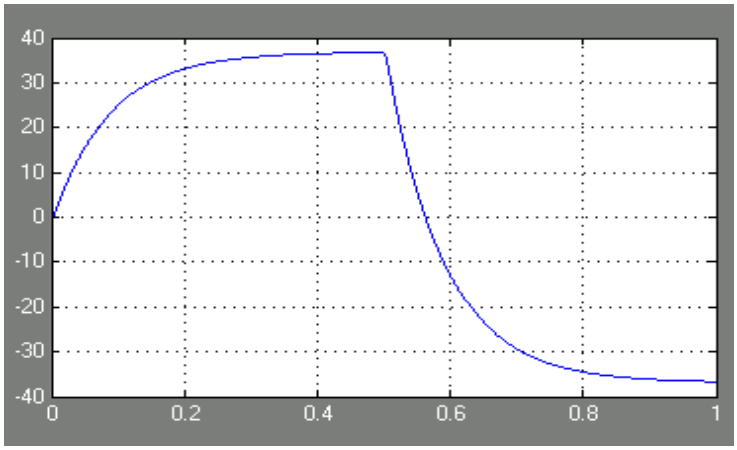


Рис. 4.2 (а) Диаграммы изменения скорости вращения вала нагрузки (рад/с)

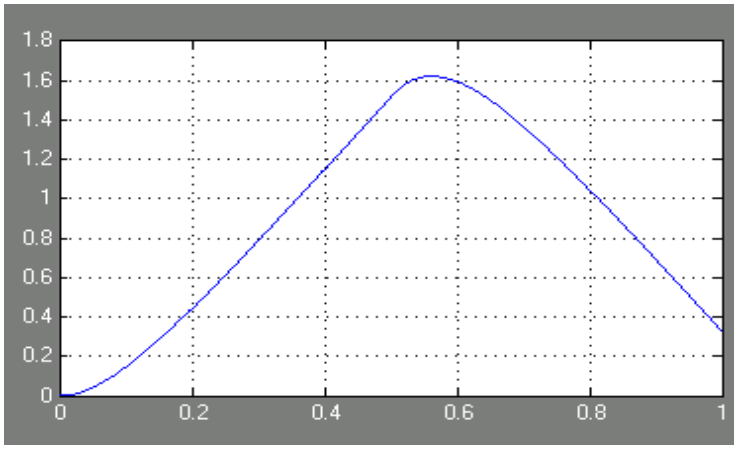


Рис. 4.2(б) Диаграммы изменения угла поворота вала нагрузки (рад)

Из приведенных диаграмм видно, чтопри максимальном напряжении до 1рад привод разгоняется за ~0.35 сек. Скорость двигателя при этом почти равна максимальной (для данного напряжения ~35 рад/с). Возьмем эти данные за основу для синтеза регулятора.

Определим две входные лингвистические переменные: «ошибка» и «скорость вала двигателя» (или просто «скорость»), определив для каждой терм множество в виде:

{«отрицательная большая» NL, «отрицательная малая» NS, «нулевая» Z, «положительная малая» PS, «положительная большая» PL}.

Для каждого терма из терм множеств лингвистических переменных «ошибка» и «скорость» необходимо задать функцию принадлежности $\mu(x)$. Выберем функции принадлежности треугольного вида для термов NS, PS и Z, и трапециидального вида для термов NL и PL.

Границы определения для функций принадлежности переменных:

«ошибка» - [-3.14 … 3.14] рад;

«скорость» - [-40 … 40] рад/с.

В качестве системы нечеткого вывода выберем нечеткий вывод Сугено 0-го порядка, где для каждого сработавшего правила задается выход нечеткого регулятора (уровень управляющего воздействия) в виде константы.

Определим несколько таких уровней:

U0 = 0В, US = 13В, UL = 27В.

При этом в зависимости от сочетания входных переменных (ошибки и скорости) возможно формирование как положительных уровней выходного сигнала, так и отрицательных.

Теперь необходимо построить блок правил функционирования нечеткого регулятора в виде матрицы 5x5, исходя из следующих предпосылок:

- если «ошибка» нулевая и «скорость» нулевая, то делать ничего не надо, управление также должно быть нулевым;
- если «ошибка» малая положительная и «скорость» нулевая, то управление должно быть малым положительным;
- если «ошибка» большая положительная и «скорость» нулевая, то управление должно быть большим положительным;
- если «ошибка» нулевая и «скорость» малая положительная, то управление должно быть малым отрицательным;
- если «ошибка» нулевая и «скорость» большая положительная, то управление должно быть большим отрицательным;

и т.д.

Таким образом, определим 13 правил для наиболее вероятных значений лингвистических переменных «ошибка» и «скорость» (таблица 5.1):

Таблица 5.1

Скорость вала двигателя	**PL**			-UL		
	PS		-UL	-US	U0	
	Z	-UL	-US	U0	US	UL
	NS		U0	US	UL	
	NL			UL		
		NL	**NS**	**Z**	**PS**	**PL**
		Ошибка				

Для создания нечеткого регулятора в системе Matlab необходимо открыть соответствующий редактор (FISeditor), введя в командной строке команду:

>> fuzzy

В результате выполнения данной команды на экране открывается окно FIS-редактора (рис. 5.3):

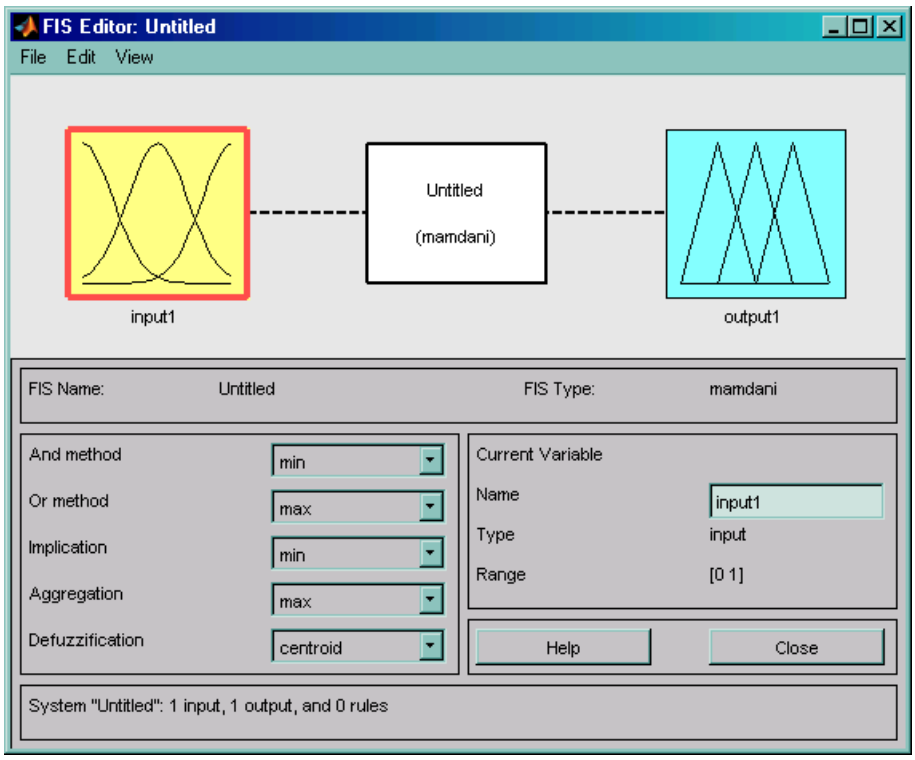


Рис.5.3 FIS-редактор

При формировании нечеткого регулятора необходимо в первую очередь определить его тип, в нашем случае - Сугено нулевого порядка. Создаем новый регулятор типа Сугено командой меню:

File ->NewFIS ->Sugeno

Учитывая, что исходя в исходных данных определены две входные переменные, добавляем еще одну входную переменную в формируемый нечеткий регулятор:

Edit ->AddVariable ->Input

Далее следует установить имена для входных переменных, отображаемых на панели нечеткого регулятора (рис. 5.4).

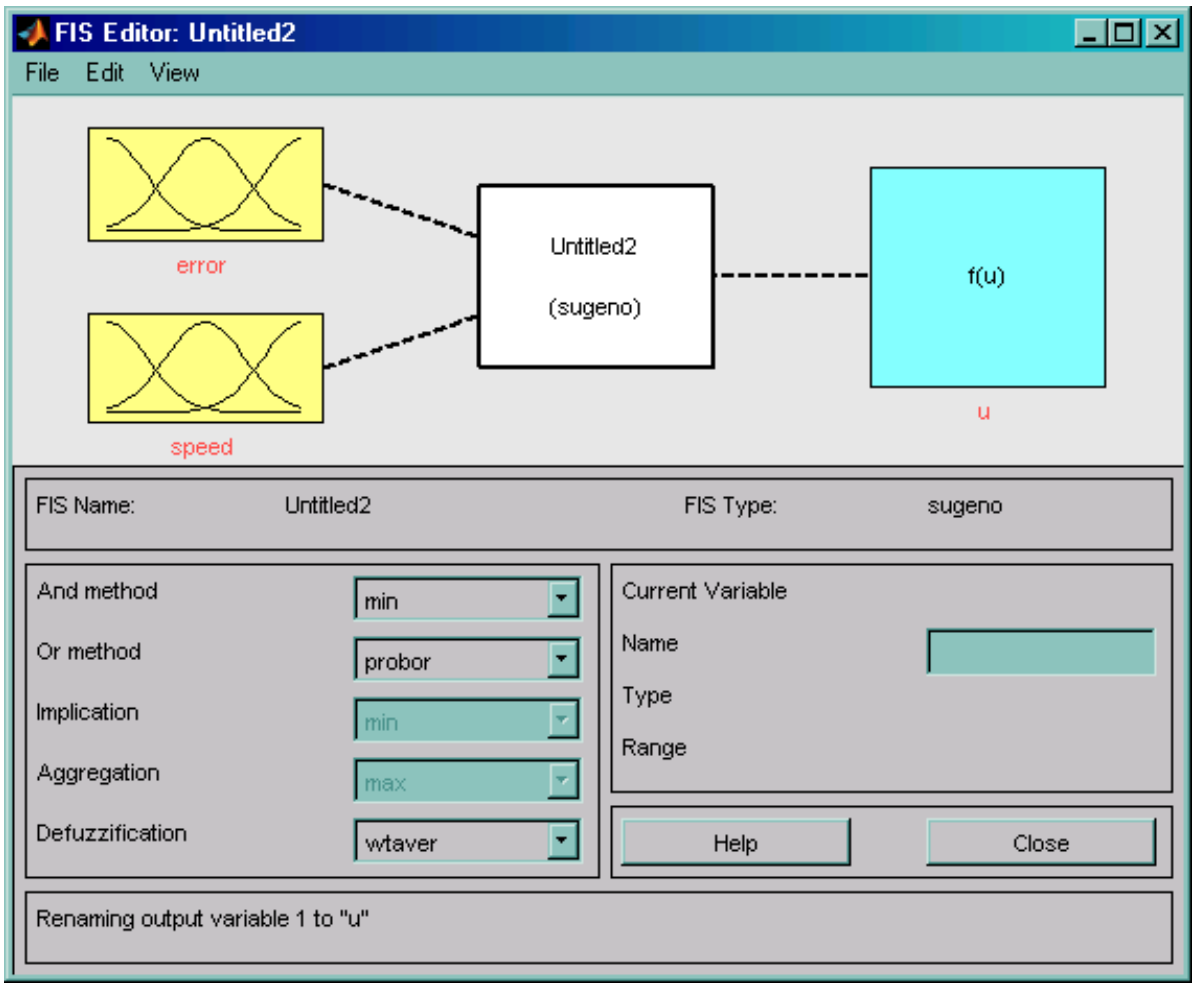


Рис.5.4 FIS-редактор с двумя входными переменными:
ошибка (error) и скорость (speed)

Открываем редактор функций принадлежности:

Edit ->MembershipFunctions

С помощью редактора функций принадлежности определяем имена и параметры функций принадлежности, а также редактируем границы изменения входных переменных.

Для **переменной «ошибка»** сначала определяем границы термов «нулевая» и «положительная малая» и «положительная большая»:

Z =[-0.3 0 0.3]

PS =[0 0.3 0.6]

PL =[0.3 0.6 3.297 4.553]

Аналогично определяем симметричные им отрицательные значения лингвистической переменной «ошибка» (рис.5.5).

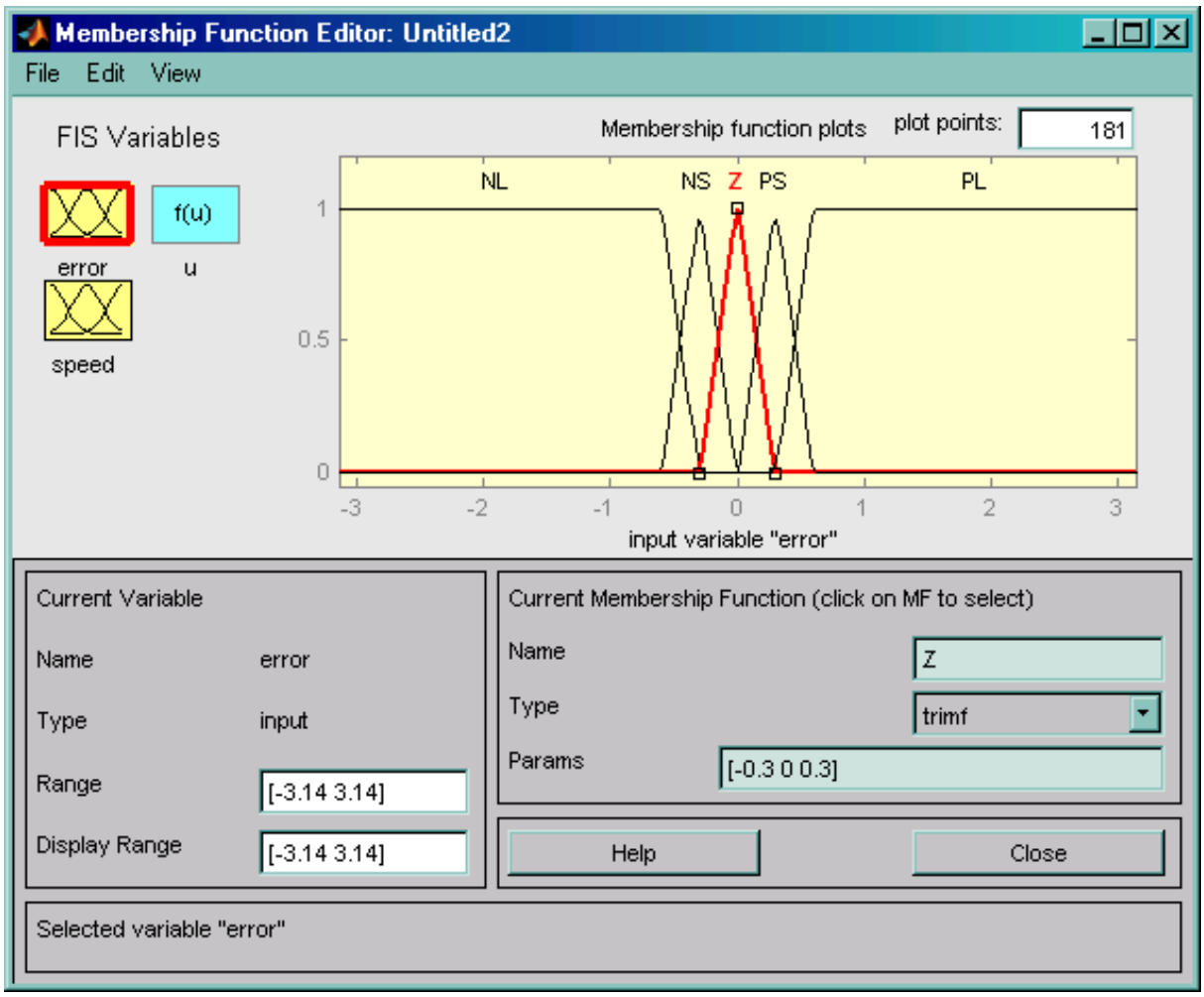


Рис.5.5 Настройка функции принадлежности для ошибки

Для **переменной «скорость»** аналогично определяем терм-множество значений:Z = [-10 0 10]; PS = [0 10 20]; PL =[10 20 42 58] и симметричные им отрицательные значения (рис.5.6).

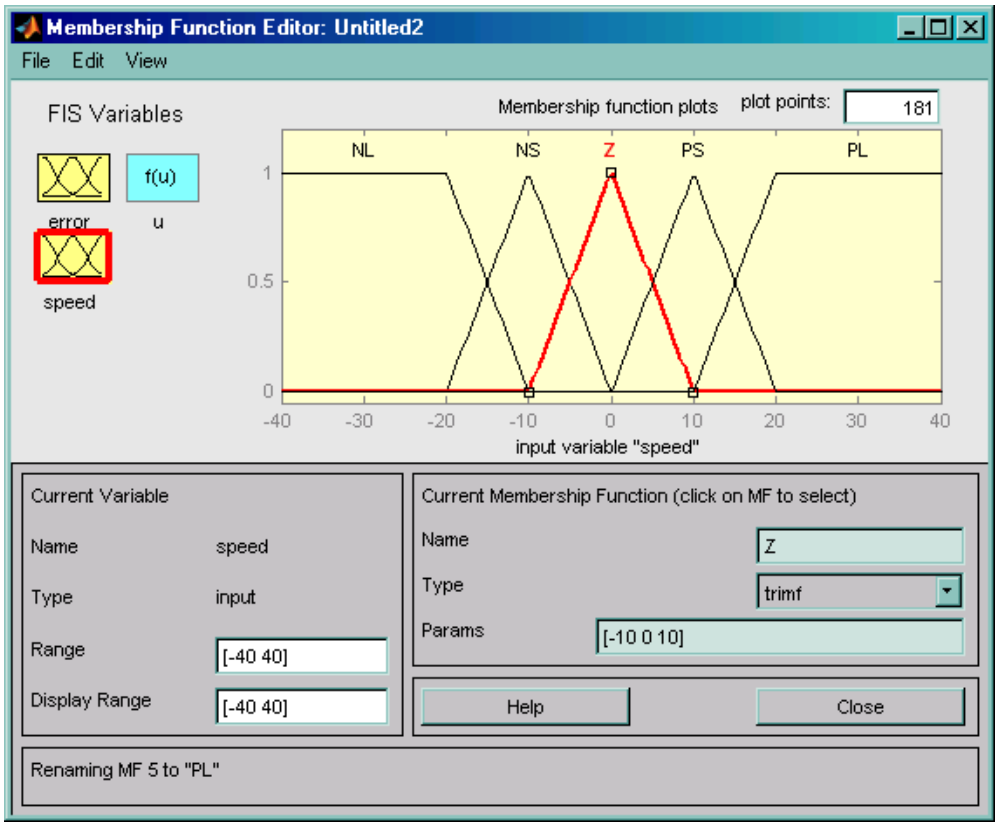


Рис.5.6 Настройка функции принадлежности для скорости

Значения выходной переменной для регулятора Сугено определяем в виде пяти констант nUL, nUS, U0, US, UL (nUL = -27, nUS = -13) (рис.5.7).

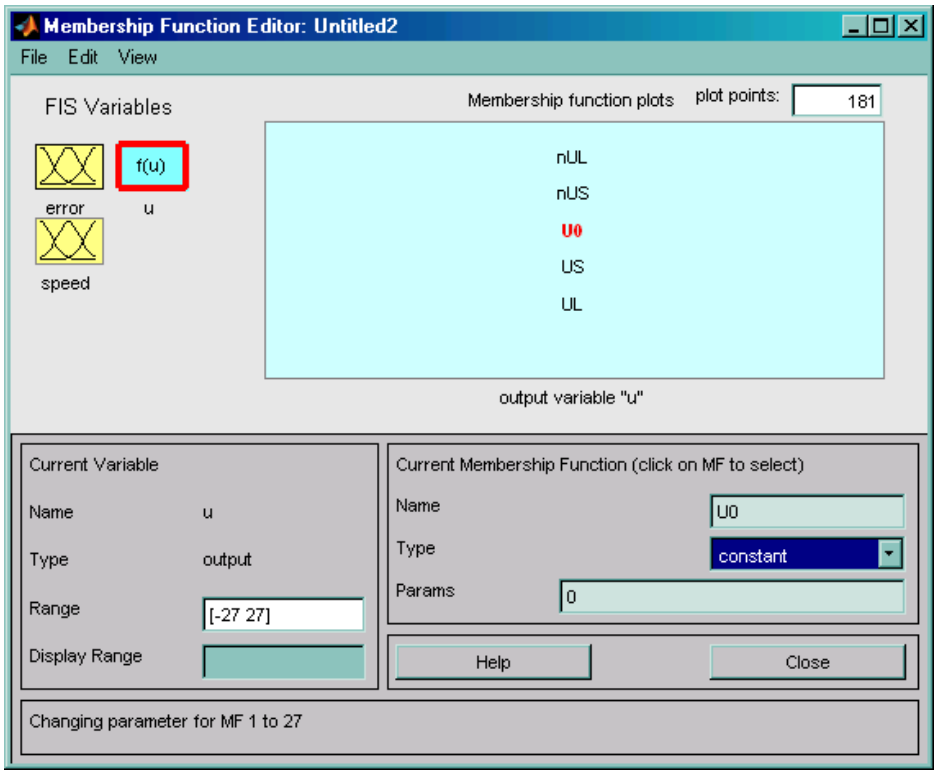


Рис.5.7 Возможные значения для выходной переменной напряжение (U)

Далее необходимо сформировать базу знаний, представляющую собой правила вывода. С этой целью открываем редактор правил:

Edit ->Rules и задаем 13 правил определенных в таблице ранее (рис.5.8).

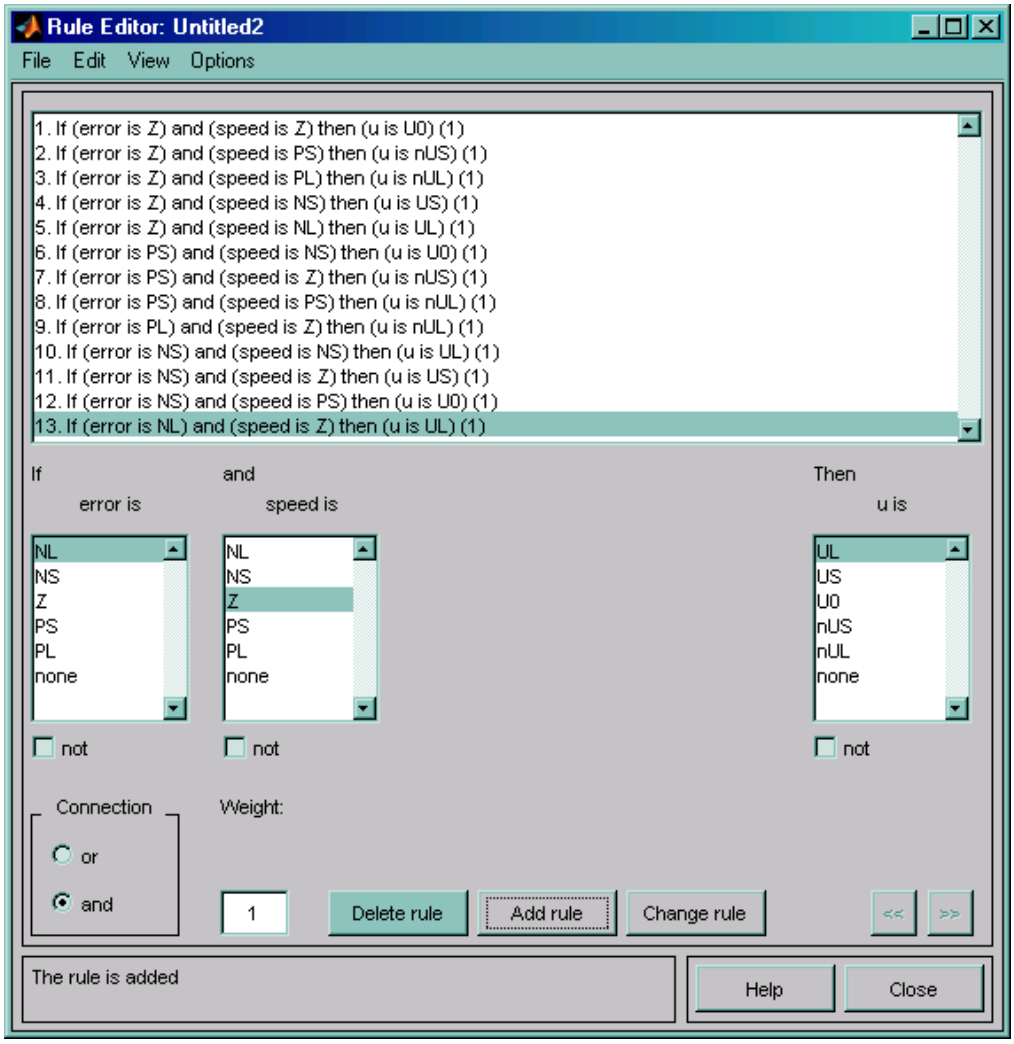


Рис.5.8 Блок правил системы управления ЭСП

Сформированный таким образом нечеткий регулятор необходимо сохранить в файле текущего раздела.С этой целью выполняем в командном окне следующую последовательность команд:

File ->Export ->ToFile

После сохранения нечеткого регулятора в файле его необходимо экспортировать (перенести) в рабочее пространство -Workspace:

File ->Export ->ToWorkspace

Таким образом нечеткий регулятор сформирован и готов к использованию в рабочей модели следящего привода. Для включения нечеткого регулятора в модель ЭСП, необходимо перенести из библиотеки FuzzyLogicToolbox соответствующую пиктограмму «FuzzyLogicController» в область модели, выполнив при этом необходимые коммутации (рис.5.9 и рис.5.10).

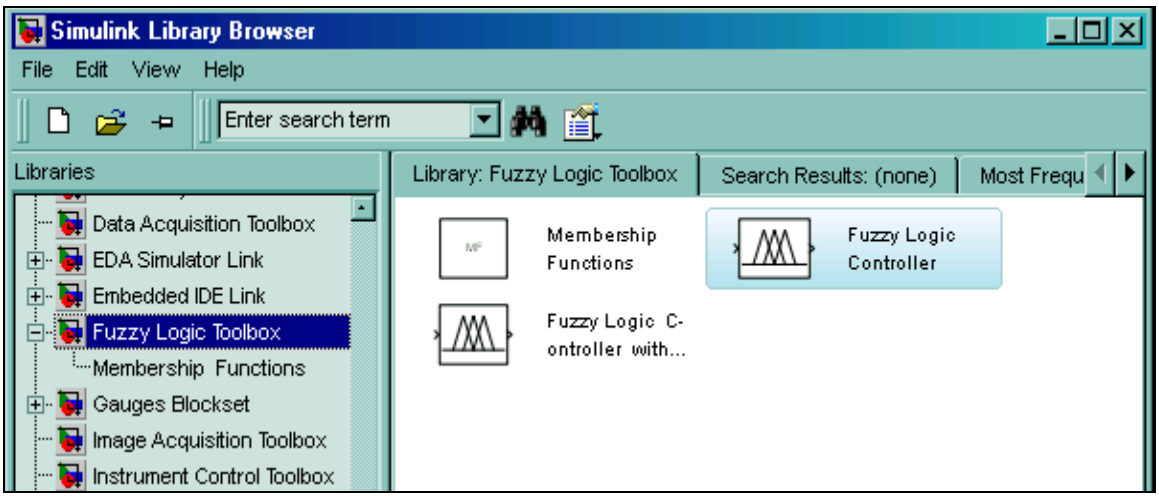


Рис. 5.9 Выбор нечеткого регулятора в Simulink-схеме из Library browser

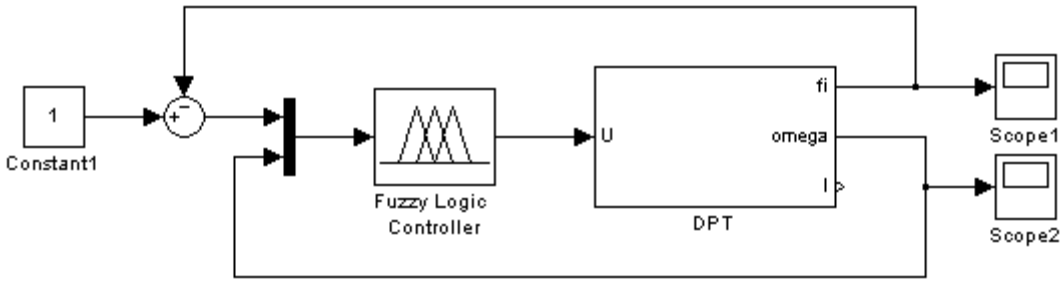


Рис.5.10 Модель ЭСП с нечетким регулятором в пакете Simulink

Для включения разработанного нечеткого регулятора необходимо в параметрах модуля «FuzzyLogicController» указать имя, которое было присвоено регулятору при экспортировании его в Workspace (рис.5.11).

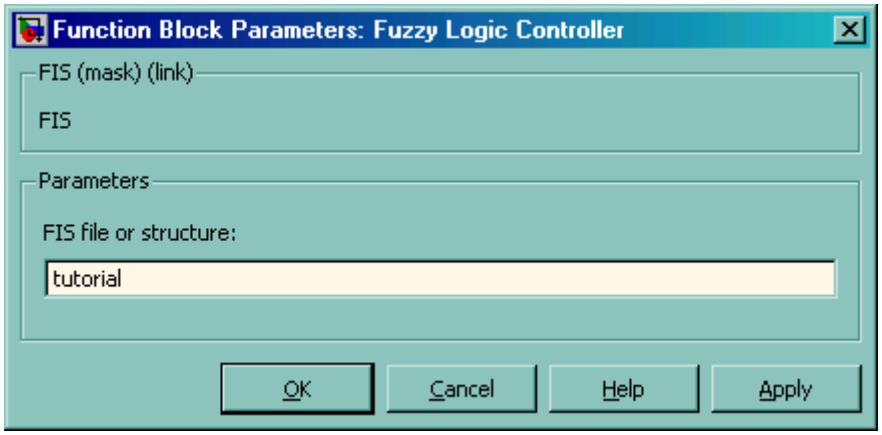


Рис.5.11 Выбор нечеткого регулятора из Workspace в блоке Fuzzy logic controller

Для оценки 8.арактеристик привода с нечетким регулятором, проведем расчет переходных процессов при отработке входного сигнала (ступеньки) в 1 рад. Переходные характеристики ЭСП с синтезированным регулятором показаны на рис.5.12 (а) и (б).

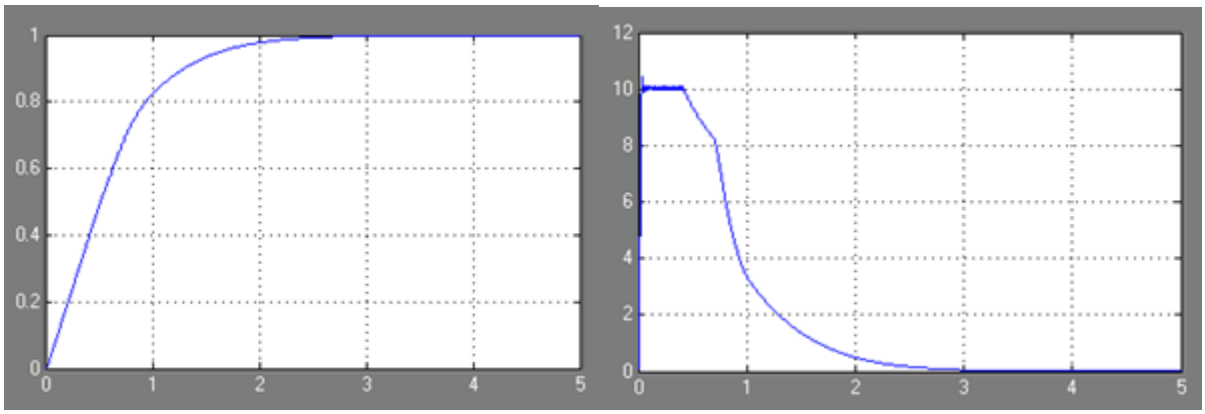


Рис.4.12 (а) Графики переходных характеристик в ЭСП (угол; (б) Графики переходных характеристик в ЭСП (скорость)

Анализ переходных процессов в ЭСП с синтезированным нечетким регулятором позволяет сделать вывод о том, что переходный процесс значительно затянут, причем очевидно, что причиной этого является пульсация управляющего напряжения от +27В до 0 на начальном этапе управления (рис.5.13).

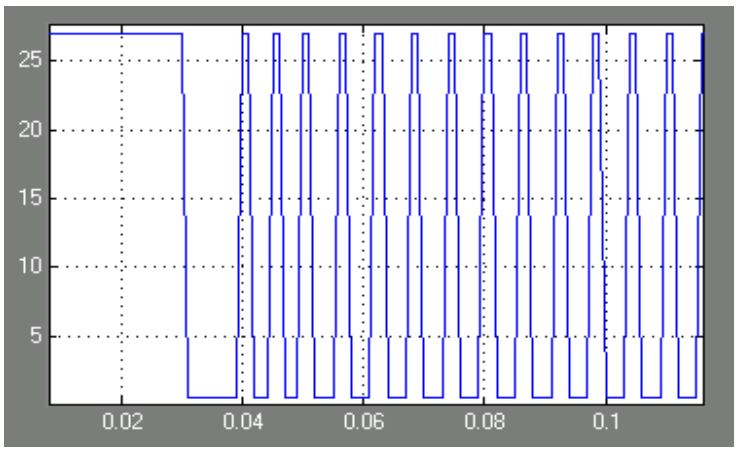


Рис.5.13 График пульсации управляющего напряжения

Причиной возникновения пульсаций управляющего напряжения на начальном участке отработки входного воздействия является недостаточное количество правил, определяющих выходное напряжение в зависимости от значений входных сигналов.

Пульсации начинаются при достижении двигателем скорости 10рад/с при еще большой ошибке, поскольку для указанных значений переменных «ошибка» и «скорость» не заданы правила, выход регулятора в этом случае принимает нулевое значение.

Для устранения этого недостатка добавим в блок правил следующие правила:

- если «ошибка» положительная большая, то управление положительное большое при любом значении скорости.

- если «ошибка» отрицательная большая, то управление отрицательное большое при любом значении скорости.

Дополненный блок правил будет иметь вид, представленный в таблице (таблица 5.2).

Таблица 5.2

	ошибка					
скорость		NL	NS	Z	PS	PL
	NL	-UL		UL		UL
	NS	-UL	U0	US	UL	UL
	Z	-UL	-US	U0	US	UL
	PS	-UL	-UL	-US	U0	UL
	PL	-UL		-UL		UL

Расчетные характеристики ЭСП с модифицированным нечетким регулятором представлены на рис.5.14.

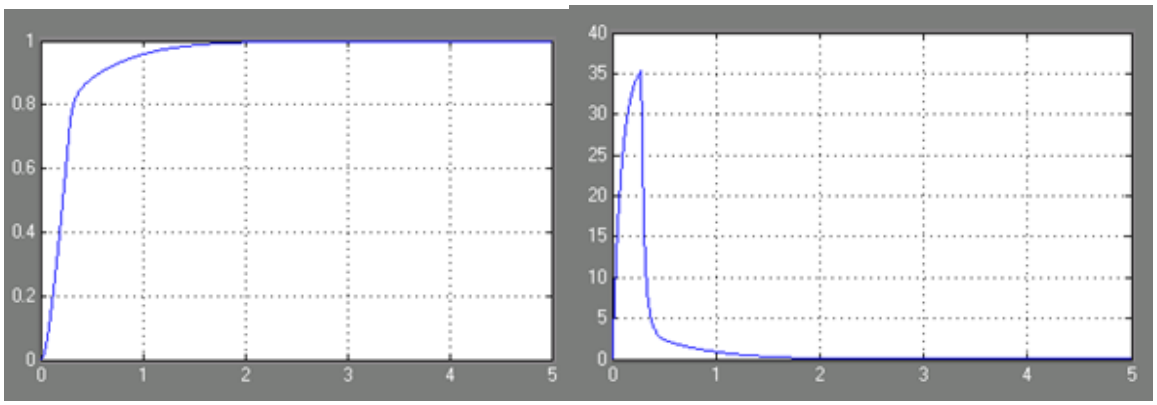


Рис.5.14 (а) Переходные характеристики в ЭСП с модифицированным регулятором (угол; (б) Переходные характеристики с модифицированным регулятором (скорость)

Диаграмма изменения управляющего воздействия приведена на рис.5.15.

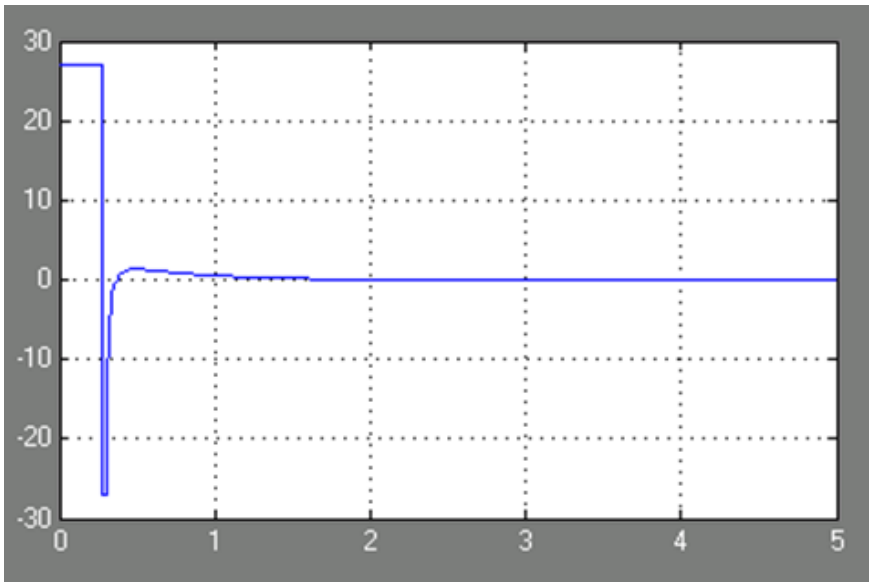


Рис.5.15 Изменение управляющего напряжения

Сравнительный анализ переходных процессов свидетельствует о том, что получены значительно лучшие характеристики, однако видно, что при уменьшении ошибки, скорость сбрасывается слишком быстро, из-за этого переходный процесс по-прежнему остается затянутым.

Для коррекции этого исправим в блоке правил два правила, соответствующие указанной ситуации в сторону увеличения уровня управляющего напряжения (таблица 5.3):

Таблица 5.3

	ошибка					
скорость		NL	NS	Z	PS	PL
	NL	-UL		UL		UL
	NS	-UL	-US	US	UL	UL
	Z	-UL	-US	U0	US	UL
	PS	-UL	-UL	-US	US	UL
	PL	-UL		-UL		UL

Итоговые переходные процессы имеют вид, представленный на рис.5.16.

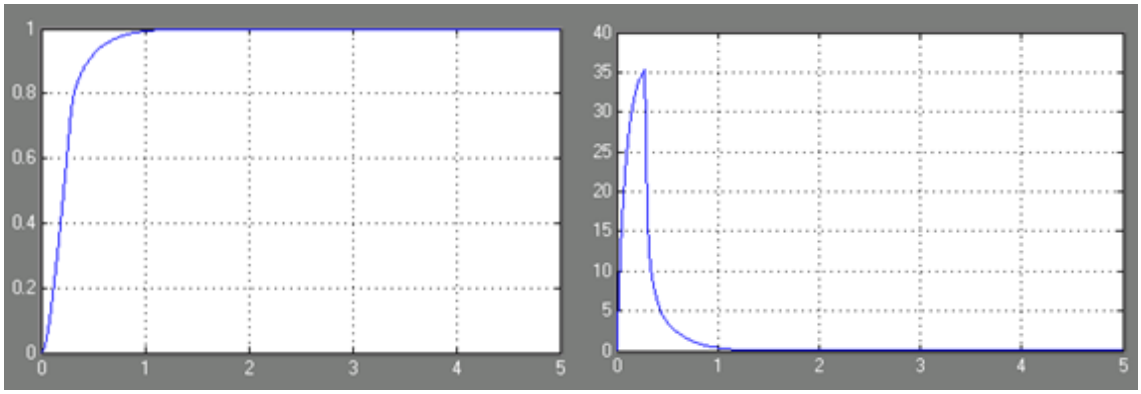


Рис.4.16 (а) Переходные характеристики после коррекции регулятора (угол)
(б) Переходные характеристики после коррекции регулятора (скорость)

Диаграмма изменения управляющего сигнала представлена на рис.5.17.

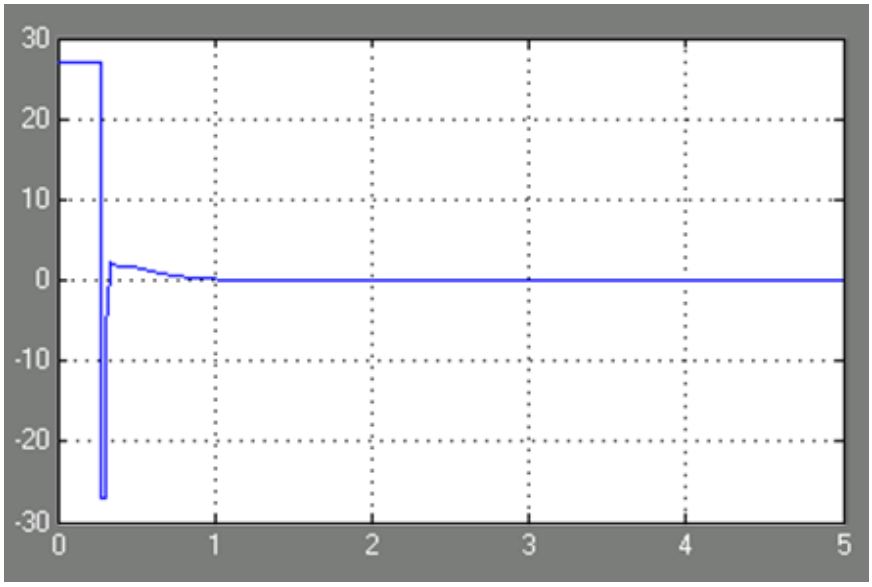


Рис.4.17 График управляющего напряжения после коррекции скорости

Анализ переходных процессов позволяет сделать вывод о том, что сформированная система управления с нечетким регулятором обеспечивает переходные процессы при отработке больших углов рассогласования, близкие к оптимальным по быстродействию. Вид итоговой поверхности, формирующей управляющее воздействие в зависимости от состояния объекта управления представлен на рис.5.18.

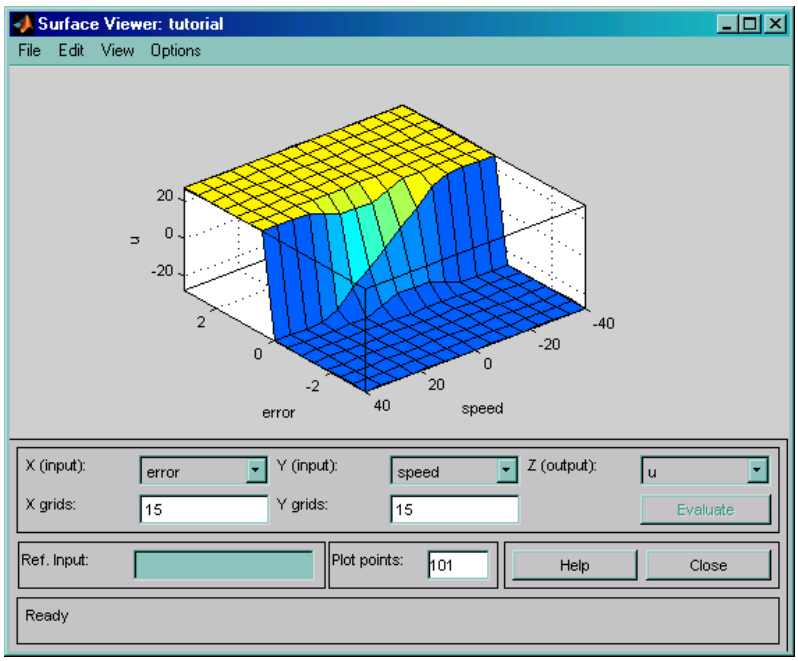


Рис.5.18 Поверхность переключения модифицированного нечеткого регулятора для системы управления ЭСП

3. ЗАДАНИЕ

Отработать практически в среде MatLab все рассмотренные этапы синтеза ЭСП постоянного тока.

4. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab.

5. ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как задаются значения нечеткого множество в реальных системах?
2. Как уменьшить пульсацию управляющего напряжения в данной САУ?
3. Нужно ли проводить дополнительные тесты САУ при ее реализации на реальном объекте? Почему?

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ПО ПАКЕТУ ПРИКЛАДНЫХ ПРОГРАММ NNT (The Neural Network Toolbox)

Предварительные сведения

Пакет прикладных программ NNT (The Neural Network Toolbox) расширяет возможности системы MATLAB.

Авторами нейроимитатора NNT являются профессор Г. Демус, специалист по нейросетевым технологиям, и М. Билл, проектирующий аппаратные реализации искусственных нейронных сетей.

Любой нейрон в пакете NNT характеризуется векторами весов, смещений и функцией активации. Нейроны, принимающие одинаковые входы и использующие одинаковые функции преобразования, группируются в слои.

Возможные нейросетевые парадигмы имитатора, определяемые выбранной структурой сети и используемым алгоритмом обучения, можно разделить на наблюдаемые и ненаблюдаемые.

Наблюдаемые сети обучаются производить требуемые выходы на входы примеров. Они могут моделировать и исследовать динамические системы, классифицировать зашумленные данные и решать многие другие задачи. Возможные обучающие правила и методы проектирования сетей прямого распространения: персептроны, линейные сети (Adaline), сети с обратным распространением ошибки по Левенбергу – Марквардту, радиальные базисные сети. Алгоритм обучения Левенберга – Марквардта относится к нелинейным методам оптимизации, является самым быстрым из алгоритмов обратного распространения ошибки и связан с использованием доверительных областей. При продвижении к минимуму среднеквадратичной функции ошибок в этом алгоритме анализируются промежуточные точки, которые получились бы по обычному методу градиентного спуска, и в случае улучшения модели осуществляется переход в эти точки, а в противном случае уменьшается шаг продвижения и переход не осуществляется. Этот алгоритм требует памяти порядка l^2, если l – количество весовых коэффициентов сети [8].

Рекуррентные сети используют обратное распространение ошибки для пространственных и временных образов и включают сети Элмана и Хопфилда. Сети Элмана характеризуются наличием линии задержки во входном слое и обратной связью выходного слоя со скрытым. Обучаемый вектор квантования (LVQ) – наиболее мощный метод для классификации линейно неотделимых векторов, позволяющий точно определить границы классов и зернистость классификации.

Ненаблюдаемые нейронные сети могут определять схемы классификации, адаптивные к изменениям входных данных. Поддерживаются обучающие ассоциативные правила Хебба, Кохонена, правила входящей звезды (instar) и исходящей звезды (outstar). Самоорганизующиеся сети могут обучаться определению регулярности и корреляции входных данных и адаптировать их будущие образы этим входам. Самоорганизующиеся нейронные сети включают сети встречного распространения, самоорганизующиеся сети и разделяющие карты.

Встроенные функции активации, такие как гиперболический тангенс (tansig), симметричная линейная (satlins), радиальная базисная (radbas), логистическая (logsig), конкурирующая (compet) функции и другие, могут быть дополнены самим пользователем.

Другие встроенные функции пакета NNT системы MATLAB позволяют осуществлять нормализацию данных, визуализацию весовых коэффициентов и смещений и представлять результаты анализа ошибок на диаграммах.

ВЫЧИСЛИТЕЛЬНАЯ МОДЕЛЬ НЕЙРОННОЙ СЕТИ

Пакет NNT использует специальный класс объектов **network.object**. Эти объекты представлены в пакете в виде массива записей, поля которых определяют их свойства, характеристики и параметры. Массивы записей позволяют задать вычислительную модель нейронной сети, для которой используется стандартное имя **net**, являющееся также и именем массива записей.

Архитектура нейронной сети характеризуется количеством входов, слоев, выходов, целей, смещений, а также топологией их соединения. Описание полей массива сети net приведено в табл. 1.

Таблица 1

net	Имя поля	Размер	Тип
.numInputs	Количество входов	1×1	integer
.numLayers	Количество слоев	1×1	integer
.biasConnect	Матрица связности для смещений	numLayers×1	Boolean array
.inputConnect	Матрица связности для входов	numLayers× numInputs	Boolean array
.layerConnect	Матрица связности для слоев	numLayers× numLayers	Boolean array
.outputConnect	Матрица связности для выходов	1×numLayers	Boolean array
.targetConnect	Матрица связности для целей	1×numLayers	Boolean array
.numOutputs	Количество выходов	1×1	integer
.numTargets	Количество целей	1×1	integer
.numInputDelays	Максимальное значение задержки для входов	1×1	integer
.numLayerDelays	Максимальное значение задержки для слоев	1×1	integer

Количество векторов входа сети **numInputs** следует отличать от числа элементов вектора входа. Количество входов задается целым положительным числом и указывает, как много векторов входа подано на сеть. В свою очередь количество элементов каждого входного вектора задается свойством inputs{i}.size. Любое изменение поля numInputs будет влиять на размеры матрицы связности inputConnect и массивов ячеек inputs{i}.

Количество слоев **numLayers** задается целым положительным числом. Любое его изменение будет влиять на размеры матриц связности biasConnect, inputConnect, layerConnect, outputConnect, targetConnect, а также размеры массивов весов и смещений IW, LW, b.

Одномерная матрица связности для смещений **biasConnect** является булевой матрицей размера $N_l \times 1$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие смещений в слое *i* отмечается в элементе вектора biasConnect(i) как 1 или 0 соответственно. Наличие смещения означает, что в массивах ячеек biases{i} и b{i} будут созданы структуры, задающие все характеристики смещения.

Матрица связности для входов **inputConnect** является булевой матрицей размера $N_l \times N_i$, где N_l – количество слоев, определяемых свойством numLayers, и N_i – количество входов, определяемых свойством numInputs. Наличие или отсутствие веса при связывании слоя *i* со слоем *j* отмечается в элементе матрицы inputConnect(i,j) как 1 или 0, соответсвенно. Наличие веса означает, что в массивах ячеек inputWeights{i} и IW{i} будут созданы структуры, задающие характеристики весов входа.

Матрица связности для слоев **layerConnect** является булевой матрицей размера $N_l \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие веса в слое *i* по входу *j* отмечается в элементе матрицы layerConnect(i,j) как 1 или 0 соответственно. Наличие веса означает, что в массивах ячеек layerWeights{i} и LW{i} будут созданы структуры, задающие характеристики весов слоя.

Матрица связности для выходов **outputConnect** – одномерная булева матрица размера $1 \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие выхода в слое *i* отмечается в элементе вектора outputConnect(i) как 1 или 0 соответственно. Наличие выхода изменяет значение свойства numOutputs и означает, что в массиве ячеек outputs{i} будет сформирована структура, задающая характеристики выхода.

Матрица связностей для целей **targetConnect** – одномерная булева матрица размера $1 \times N_l$, где N_l – количество слоев, определяемых свойством numLayers. Наличие или отсутствие целевого выхода в слое *i* отмечается в элементе вектора targetConnect(i) как 1 или 0 соответственно. Наличие цели изменяет значение свойства numTargets и означает, что в массиве ячеек targets{i} будет сформирована структура, задающая характеристики целевого выхода.

Число выходов **numOutputs,** определенное только для чтения, задается количеством единиц в матрице связности для выходов.

Число целей **numTargets,** определенное только для чтения, задается количеством единиц в матрице связности для целей.

Максимальное значение задержки для входов **numInputDelays** (только для чтения) определяет максимальное значение задержки для входных последовательностей.

Максимальное значение задержки для слоев **numLayerDelays** определяет максимальное значение задержки для всех слоев сети.

Перечень функций, используемых для инициализации, адаптации и обучения нейронной сети net, приводится в табл. 2.

Таблица 2

net	Имя поля	Состав	Тип
.initFcn	Функции инициализации	initcon |initiay initnw | initnwb | initzero	char

.initParam	Параметры функции инициализации		
.adaptFcn	Функции адаптации	adaptwb | trains	char
.adaptParam	Параметры функции адаптации		
.trainFcn	Функции обучения	trainbr | trainc | traincgb | traincgf| traingda | traingdm| traingdx | trainlm| trainoss | trainr | trainrp | trainscg	char
.trainParam	Параметры функции обучения		
.performFcn	Функции оценки качества обучения	mae | mse | sse | msereg	char
.performParam	Параметры функции оценки качества обучения		

Параметры всех функций этой таблицы определяют набор параметров для используемой функции. Узнать набор таких параметров можно, применив оператор help, например:

Help(net.trainFcn).

Функции инициализации **initFcn** определяют, какая из функций инициализации будет использована для задания начальных матриц весов и векторов смещений при вызове метода init для всей сети. При изменении этого свойства параметры функции инициализации initParam будут использовать значения, соответствующие новой функции.

Функции адаптации **adaptFcn** определяют, какая из функций адаптации будет использована при вызове метода adapt.

Функции обучения **trainFcn** определяют, какая из функций обучения будет использована при вызове метода train.

Функции оценки качества обучения **performFcn** определяют, какая из функций оценки качества обучения будет использована при вызове метода train.

Пользователь может дополнить список используемых функций инициализации, адаптации, обучения и оценки качества обучения.

Элементы сети задаются с помощью массивов ячеек, которые включают структуры для описания входов, слоев, выходов, целей, смещений и весов входа. Все эти структуры определяются однотипно, поэтому в пособии рассмотрим только описание слоев.

Описание полей структуры, используемой для определения каждого слоя нейронной сети net.layers{i}, приводится в табл. 3.

Таблица 3

net	Имя поля	Тип	Размер, состав
.layers{i}	Описатель i-го слоя сети	Cell array	numLayers×1
.dimensions	Распределение нейронов по размерностям слоя	Double array	1×numdim
.distanceFcn	Функции вычисления расстояния между нейронами	Char	boxdist | dist | linkdist | mandist
.distances	Расстояния между нейронами	Double array	
.initFcn	Функции инициализации	Char	initw | initwb
.netInputFcn	Функции накопления	Char	netprod| netsum
.positions	Положения нейронов	Array	
.size	Количество нейронов	Integer	1×1
.topologyFcn	Функции топологии	Char	gridtop | hextop | randtop
.transferFcn	Функции активации	Char	compet| logsig | poslin |purelin | radbas | satlin | satlins | tansig | tribas | hardlims
.userdata	Информация пользователя	Struct	1×1
.note	текст	Char	1×var

Описатель слоев нейронной сети **layers{i}** является массивом ячеек размера N_l×1, где N_l – число слоев сети, равное numLayers, состоящим из ячеек layers{i}, каждая из которых является массивом записей для описания *i*-го слоя сети.

Вектор распределения по размерностям слоя **dimensions** позволяет описывать многомерные слои нейронов реальных геометрических размерностей: одномерные, двумерные, трехмерные. Многомерный слой размерности numdim может быть задан вектор-строкой, элементы которой указывают число нейронов по каждой размерности, тогда их произведение будет определять общее количество нейронов в многомерном слое layers{i}.**size**. Это свойство позволяет определить положение нейронов layers{i}.positions, если известна функция топологии слоя layers{i}.**topologyFcn**. При изменении этого вектора будут автоматически изменяться layers{i}.size, положение нейронов layers{i}.**positions** и расстояния между ними layers{i}.**distances**.

Функция оценки расстояния между нейронами **distanceFcn** задает функцию, используемую для вычисления расстояния между нейронами в слое *i*. При замене функции будут автоматически пересчитаны значения расстояний между нейронами слоя layers{i}. distances. Список применяемых функций оценки расстояния может быть расширен самим пользователем.

Расстояние между нейронами в *i*-ом слое определяет свойство layers{i}.**distances**.

Функция инициализации слоя **initFcn** определяет, какая функция инициализации используется для слоя *i*.

Функция накопления **netInputFcn** определяет, какая функция накопления применяется для слоя *i*.

Для построения графика расположения нейронов в многомерном слое для слоя *i* можно использовать функцию plotsom с аргументом net.layers{i}.positions.

Функция активации слоя **transferFcn** определяет функцию активации, используемую для задания нейрона в слое *i*.

Поле для записи информации пользователя для слоя *i* предусмотрено в массиве записей **userdata.** По умолчанию поле inputs{i}.userdata.**note**, предусмотренное для записи текста, содержит строку «Put your custom input information here», что означает – «Информацию разместите здесь».

Матрицы весов и векторы смещений описываются функциями, представленными в табл. 4.

Таблица 4

net	Имя поля	Тип	Размер
.IW	Массив ячеек для матриц весов входа	Cell array	numLayers× numInputs
.LW	Массив ячеек для матриц весов слоя	Cell array	numLayers× numLayers
.b	Массив ячеек для векторов смещений	Cell array	numLayers×1

Массив ячеек **IW** имеет размеры $N_l \times N_i$, где N_l – число слоев numLayers и N_i – число входов numInputs сети net, каждый элемент которого является матрицей весов, связывающе слой *i* со входом *j* сети. Структура этого массива согласована с матрицей связности inputConnect(i,j). Каждая матрица весов должна иметь число строк, равное параметру layers{i}.size, а число столбцов соответствовать параметру inputWeights{i,j}.size.

Массив ячеек **LW** имеет размеры $N_l \times N_l$, где N_l – число слоев numLayers сети net, каждый элемент которого является матрицей весов, связывающий слой *i* со слоем *j* сети. Структура этого массива согласована с матрицей связности layerConnect(i,j). Каждая матрица весов должна иметь число строк, равное параметру layers{i}.size , а число столбцов соответствовать параметру layerWeights{i,j}.size.

Вектор смещений **b** является массивом ячеек размера $N_l \times 1$, где N_l – число слоев numLayers объекта net, каждый элемент которого является вектором смещений для слоя *i* сети. Структура этого вектора согласована с вектором связности biasConnect(i,j). Длина вектора смещений для слоя *i* должна соответствовать параметру biases{i}.size. Следует отметить, что для получения информации о структуре полей инициированного объекта network можно использовать функцию fieldnames(<имя_сети>), которая отражает текущее состояние нейронной сети.

Лабораторная работа №6.

МОДЕЛИ ИСКУССТВЕННОГО НЕЙРОНА. РЕШЕНИЕ ПРОСТЫХ ЗАДАЧ В НЕЙРОСЕТЕВОМ БАЗИСЕ

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение основных моделей искусственного нейрона, их математического описания, а также функционального и структурного графических представлений, исследование функций активации и рассмотренных моделей нейронов с помощью инструментального пакета имитационного моделирования Simulink системы MATLAB.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Простой нейрон

Элементарной ячейкой нейронной сети является нейрон. Структура нейрона с единственным скалярным входом показана на рис.6.1.

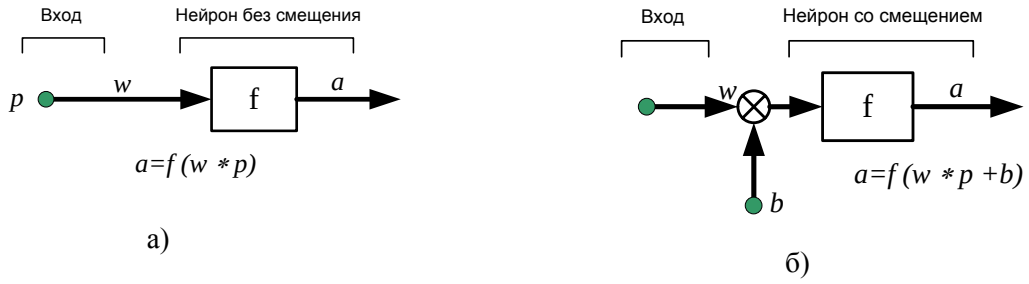


Рис.6.1.

Скалярный входной сигнал p умножается на скалярный весовой коэффициент w, и результирующий взвешенный вход w*p является аргументом функции активации нейрона, которая порождает скалярный выход a. Нейрон, показанный на рис. 1.1,б, дополнен скалярным смещением b. Смещение суммируется со взвешенным входом w*p и приводит к сдвигу аргумента функции на величину b.

Действие смещения можно свести к схеме взвешивания, если представить что нейрон имеет второй входной сигнал со значением, равным 1. Вход n функции активации нейрона по-прежнему остается скалярным и равным сумме взвешенного входа и смещения b. Эта сумма является аргументом функции активации f; выходом функции активации является сигнал a. Константы w и b являются скалярными параметрами нейрона.

Основной принцип работы нейронной сети состоит в настройке параметров нейрона с тем, чтобы функционирование сети соответствовало некоторому желаемому поведению. Регулируя веса или параметры смещения, можно “научить” сеть выполнять конкретную работу; возможно также, что сеть сама будет корректировать свои параметры, чтобы достичь требуемого результата.

Уравнение нейрона со смещением имеет вид a = f(w*p+b*l). Как уже отмечалось, смещение b – настраиваемый скалярный параметр нейрона, который не является входом, а константа 1, которая управляет смещением, рассматривается как вход и может быть учте- на в виде линейной комбинации векторов входа

$$a = \begin{bmatrix} w & \mathrm{b} \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}.$$

Функция активации

Функции активации (передаточные функции) нейрона могут иметь самый разнообразный вид. Функция активации f, как правило, принадлежит классу сигмоидальных функций с аргументом n и выходом a.

Ниже рассмотрены три наиболее распространенные функции активации.

Единичная функция активации с жестким ограничениям hardlim

Эта функция описывается соотношением a = hardlim(n) = 1(n) и показана на рис. 6.2. Она равна 0, если n < 0, и 1,если n >= 0.

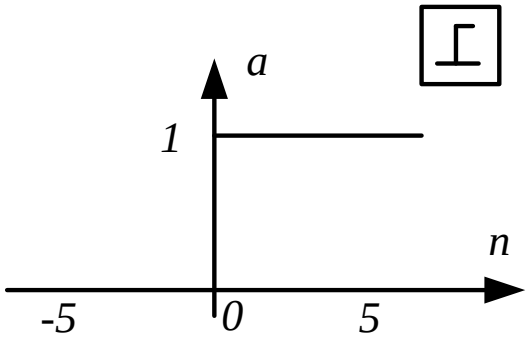


Рис.6.2.

В состав пакета ППП Neural Network Toolbox входит М-функция hardlim, реализующая функцию активации с жесткими ограничениями.

Линейная функция активации purelin.

Эта функция описывается соотношением a = purelin(n) = n и показана на рис. 6.3.

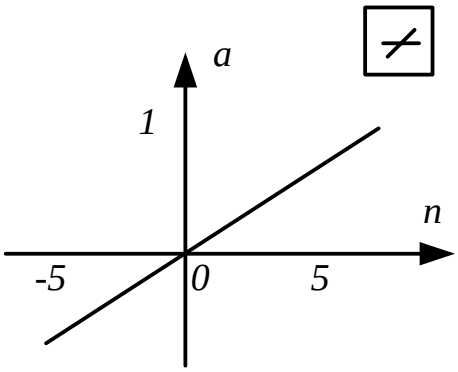


Рис.6.3.

Логистическая функция активации logsig.

Эта функция описывается соотношением a = logsig(n) = 1/(1 + exp(-n)) и показана на рис. 1.4. Она принадлежит к классу сигмоидальных функций, и ее аргумент может принимать любое значение в диапазоне от $\infty -$ до $\infty +$, а выход изменяется в диапазоне от 0 до 1. В пакете ППП Neural Network Toolbox она представлена М-функцией logsig. Благодаря свойству дифференцируемости эта функция часто используется в сетях с обучением на основе метода обратного распространения ошибки.

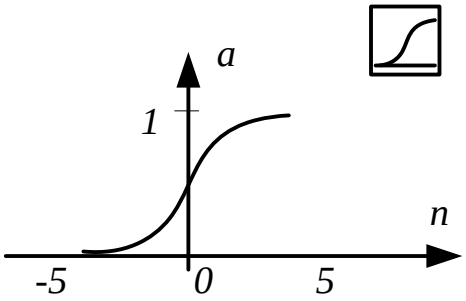


Рис.6.4.

Символ в квадрате в правом верхнем углу графика характеризует функцию активации. Это изображение используется на структурных схемах нейронных сетей. В пакет ППП Neural Network Toolbox включены и другие функции активации. Используя язык MATLAB, пользователь может создавать и свои собственные уникальные функции.

Нейрон с векторным входом

Нейрон с одним вектором входа p с R элементами $p_1, p_2, ..., p_R$ показан на рис.6.5. Здесь каждый элемент входа умножается на веса $w_{11}, w_{12}, ..., w_{1R}$ соответственно и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора- строки W на вектор входа p.

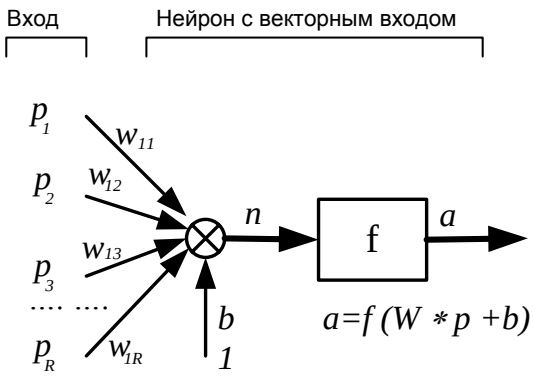


Рис.6.5.

Нейрон имеет смещение b, которое суммируется со взвешенной суммой входов. Результирующая сумма n определяется как

$$n = w_{11}p_1 + w_{12}p_2 + ... + w_{1R}p_R + b$$

и служит аргументом функции активации f. В нотации языка MATLAB это выражение записывается так:

$$n = W * \mathrm{p} + b.$$

Структура нейрона, показанная на рис. 6.5, содержит много лишних деталей. При рассмотрении сетей, состоящих из большого числа нейронов, будет использоваться укрупненная структурная схема нейрона (рис. 6.6). Вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа R. Размер вектора входа p указывается ниже символа p и равен Rxl. Вектор входа умножается на вектор-строку W длины R. Как и прежде, константа 1 рассматривается как вход, который умножается на скалярное смещение b. Входом n функции активации нейрона служит сумма смещения b и произведения W*p. Эта сумма преобразуется функцией активации f, на выходе которой получается выходная величина нейрона a, которая в данном случае является скалярной величиной. Структурная схема, приведенная на рис. 3.6, называется слоем сети. Слой характеризуется матрицей весов W, смещением b, операциями умножения W*p, суммирования и функцией активации f. Вектор входов p обычно не включается в характеристики слоя.

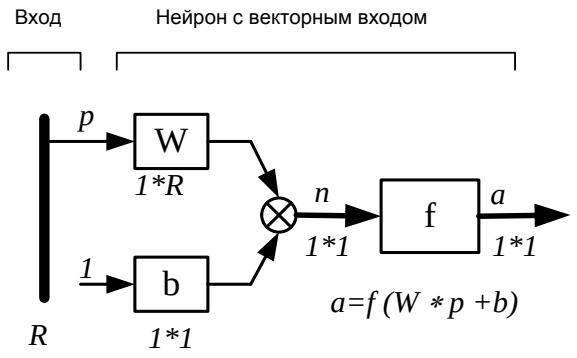


Рис.6.6.

Каждый раз, когда используется сокращенное обозначение сети, размерность матриц указывается под именами векторно-матричных переменны1. Эта система обозначений поясняет строение сети и связанную с ней матричную математику. На укрупненной структурной схеме для обозначения типа функции активации применяются специальные графические символы; некоторые из них приведены на рис. 6.7, где а – ступенчатая; б – линейная; в – логистическая функция.

hardlim
а)

purelin
б)

logsig
в)

Рис.6.6.

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Задание 1.

Для функции активации (***hardlim, hardlims, purelin, poslin, satlin, satlins, radbas, tribas, logsig, tansig***) и её производной определяемыми следующими соотношениями:

$$\mathrm{hardlim}(n) = \begin{cases} 0, \mathrm{n} < 0; \\ 1, \mathrm{n} \geq 0; \end{cases}$$

$$\mathrm{dhardlim}(n) = \begin{cases} 0, \mathrm{n} < 0; \\ 0, \mathrm{n} \geq 0, \end{cases}$$

выполнить следующие действия:

1. Вывести на экран информацию о функции с помощью следующих команд (на примере функции ***radbas*** и ***dradbas***):

name = radbas('name') % – полное название функции;
dname = radbas('deriv') % – название производной;
inrange = radbas('active') % – диапазон входа;
outrange = radbas('output') % – диапазон выхода;

2. Построить графики функции и производной:

n = -5:0.1:5;
a = radbas(n);
da = dradbas(n, a);
plot(n,a,'r') % – график функции активации – красный;
hold on
plot(n,da,'b') % – график производной – синий;

3. Рассчитать векторы выхода ***A*** и производной ***dA_dN*** для слоя из трёх нейронов с вектором входа ***N***, состоящим из трёх компонентов:

N = [-0.7; 0.1; 0.8];
A = radbas(N) % – вектор выхода функции актива;
dA_dN = dradbas(N,A) % – вектор выхода производной.

4. Оформить рассмотренную последовательность команд в виде скрипта в m-файле.

Задание 2.

Для симметричной функции активации с жёсткими ограничениями hardlims и её производной dhardlims, определяемыми соотношениями

$$\text{hardlims}(n) = \begin{cases} -1, \text{n} < 0; \\ 1, \text{n} \geq 0; \end{cases}$$

$$\text{dhardlims}(n) = \begin{cases} 0, \text{n} < 0; \\ 0, \text{n} \geq 0, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем hardlimsfile.

Задание 3.

Для линейной функции активации purelin и ее производной, dpurelin, определяемыми соотношениями

purelin = n; dpurelin = 1,

выдать на экран информацию об этих функциях, построить их гра- фики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем purelinfile.

Задание 4 .

Для положительной линейной функции активации poslin и ее производной dposlin, определяемыми соотношениями

$$poslin(n) = \begin{cases} 0, \text{n} < 0; \\ n, \text{n} \geq 0; \end{cases}$$

$$\text{d}poslin(n) = \begin{cases} 0, \text{n} < 0; \\ 1, \text{n} \geq 0, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем poslinfile.

Задание 5 .

Для линейной функции активации с ограничениями satlins и ее производной dsatlins, определяемыми соотношениями

$$\text{satlins}(n) = \begin{cases} 0, \text{n} < 0; \\ \text{n}, \ 0 \leq \text{n} \leq 1; \\ \text{n}, \ \text{n} > 1, \end{cases}$$

$$\mathrm{dsatlins}(n) = \begin{cases} 0, \mathrm{n} < 0; \\ 1,\ 0 \le \mathrm{n} \le 1; \\ 0,\ \mathrm{n} > 1, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем satlinfile.

<u>Задание 6.</u>

Для треугольной функции активации tribas и ее производной dtribas, определяемыми соотношениями

$$\mathrm{tribas}(n) = \begin{cases} 0, \mathrm{n} < -1; \\ 1 - \mathrm{abs(n)},\ -1 \le \mathrm{n} \le 1; \\ 0,\ \mathrm{n} > 1, \end{cases}$$

$$\mathrm{dtribas}(n) = \begin{cases} 0, \mathrm{n} < -1; \\ 1,\ -1 \le \mathrm{n} \le 0; \\ -1,\ 0 < \mathrm{n} \le 1; \\ 0,\ \mathrm{n} > 1, \end{cases}$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода, воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем tribasfile.

<u>Задание 7.</u>

Для логистической функции активации logsig и ее про- изводной dlogsig определяемыми соотношениями

$$\mathrm{logsig(n)} = 1/\left(1 + \mathrm{e}^{-\mathrm{n}}\right);$$
$$\mathrm{dlogsig(n)} = \mathrm{e}^{-\mathrm{n}}/\left(1 + \mathrm{e}^{-\mathrm{n}}\right)^2,$$

выдать на экран информацию об этих функциях, построить их графики и рассчитать векторы выхода воспользовавшись скриптом из М-файла hardlimfile. Новый скрипт записать в файл под именем logsigfile.

<u>Задание 8.</u>

Для простого нейрона с одним входом **p** и функции активации **hardlim** подобрать весовой коэффициент **w** и смещение **b** таким образом, чтобы обеспечить инвертирование входного сигнала, т. е. замену нуля единицей, а единицы нулем.

Уравнение нейрона со смещением имеет вид
a = f(w*p+b)
в данном случае:

a = hardlim(w*p+b)

Для инвертирования входного сигнала значение **w** и **b** должны быть **w=-1, b=-0,5:**

```
w = -1;
b = 0;
p = 0;
a = hardlim(w*p+b);
p = 1;
a = hardlim(w*p+b);
```

<u>Задание 9.</u>

Для нейрона с двумя двоичными входами $\mathbf{p_1}$ и $\mathbf{p_2}$ и функцией активации **hardlim** подобрать весовые коэффициенты и смещение таким образом, чтобы нейрон выполнял функции логического сложения и логического умножения.

Логическое сложение:

P1	P2	Выход
0	0	0
0	1	1
1	0	1
1	1	1

Логическое умножение:

P1	P2	Выход
0	0	0
0	1	0
1	0	0
1	1	1

Решение:

```
% логическое сложение
w = [1, 1];
b = -0.5;

p = [0; 0]
a = hardlim(w*p + b)
p = [0; 1]
a = hardlim(w*p + b)
p = [1; 0]
a = hardlim(w*p + b)
p = [1; 1]
a = hardlim(w*p + b)

% логическое умножение
w = [1, 1];
b = -1.5;

p = [0; 0]
```

a = hardlim(w*p + b)
p = [0; 1]
a = hardlim(w*p + b)
p = [1; 0]
a = hardlim(w*p + b)
p = [1; 1]
a = hardlim(w*p + b)

Задание 10.

Для нейрона с двумя двоичными входами $\mathbf{p_1}$ и $\mathbf{p_2}$ и функцией активации **hardlim** подобрать весовые коэффициенты w11 w12 и смещение b таким образом, чтобы нейрон мог классифицировать входные двоичные наборы на два класса – нулевой и первый:

а) {00,01}- нулевой класс, {10,11}- первый класс;

б) {11}- нулевой класс, {00,01,10}- первый класс;

в) {00,10}- нулевой класс, {01,11}- первый класс;

г) {10}- первый класс, {00,01,11}- нулевой класс;

д) {00,11}- нулевой класс, {10,01}- первый класс.

Задание 11.

Для нейрона с двумя непрерывными входами p_1 и p_2 и функции активации hardlim построить график разделяющей линии, определяемой уравнением

$$\mathbf{W_{11}\, p_1 + W_{12}\, p_2 + b = 0,}$$

считая, что значения весовых коэффициентов W_{11} W_{12} и смещения b заданы. Убедиться, что наборы входов p_1 и p_2 по разную сторону от разделяющей линии принадлежат разным классам и что не всякое множество наборов значений входов можно разделить на два класса, используя нейрон рассмотренного типа.

Задание 12.

Используя блоки имитационного моделирования инструментального пакета Simulink системы MATLAB (блоки источников и регистраторов сигналов, математические и нелинейные блоки) построить рассмотренные в заданиях 1 – 12 модели нейронов, провести исследования моделей нейронов и оформить отчет.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных в

Лабораторная работа №7

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Цель работы: изучение архитектуры искусственных нейронных сетей, способов их графического изображения в виде функциональных и структурных схем и программного представления в виде объектов специального класса network, включающих массив структур с атрибутами сети и набор необходимых методов для создания, инициализации, обучения, моделирования и визуализации сети, а также приобретение навыков построения сетей различной архитектуры с помощью инструментального программного пакета Neural Network Toolbox системы MATLAB.

Теоретические сведения

Хотя отдельные нейроны и способны после некоторой процедуры обучения решать ряд задач искусственного интеллекта, все же для эффективного решения сложных задач по распознаванию образов, идентификации и классификации объектов, распознаванию и синтезу речи, оптимальному управлению применяют достаточно большие группы нейронов, образуя из них искусственные нейронные сети в виде связанных между собой слоёв, напоминающие биологические нейронные (нервные) сети человека и животных.

Существует множество способов организации искусственных нейронных сетей, которые могут содержать различное число слоёв нейронов. Нейроны могут быть связаны между собой как внутри отдельных слоёв, так и между слоями. В зависимости от направления передачи сигнала эти связи могут быть прямыми или обратными.

Слой нейронов, непосредственно принимающий информацию из внешней среды, называется входным слоем, а слой, передающий информацию во внешнею среду, – выходным слоем. Остальные слои, если они имеются в сети, называются промежуточными, или скрытыми. В ряде случаев такого функционального распределения слоёв сети не производится, так что входы и выходы могут присоединяться к любым слоям и иметь произвольное число компонент.

Структура, или архитектура сети искусственных нейронов зависит от той конкретной задачи, которую должна решать сеть. Она может быть однослойной без обратных связей или с обратными связями, двухслойной с прямыми связями, трехслойной с обратными связями и т. д. Сети с обратными связями называют часто рекуррентными.

Описание архитектуры искусственной нейронной сети помимо указания числа слоёв и связей между ними должно включать сведения о количестве нейронов в каждом слое, виде функций активации в каждом слое, наличии смещений для каждого слоя, наличии

компонент входных, выходных и целевых векторов, а в ряде случаев и характеристики топологии слоёв. Например, для аппроксимации любой функции с конечным числом точек разрыва широко используется сеть с прямой передачей сигналов. В этой сети имеется несколько слоёв с сигмоидальными функциями активации. Выходной слой содержит нейроны с линейными функциями активации. Данная сеть не имеет обратных связей, поэтому её называют сетью с прямой передачей сигналов (FF-net).

Графически искусственная нейронная сеть изображается в виде функциональной или структурой схемы. На функциональной схеме сети с помощью геометрических фигур изображаются её функциональные блоки, а стрелками показываются входы, выходы и связи. В блоках и на стрелках указываются необходимые обозначения и параметры.

Структурная схема сети изображается с помощью типового набора блоков, соединительных элементов и обозначений, принятых в инструментальном программном пакете Neural Network Toolbox системы MATLAB и пакете имитационного моделирования Simulink той же системы. Структурная схема сети может быть укрупнённой или

детальной, причём степень детализации определяется пользователем. Системы обозначений блоков, элементов и параметров сети является векторно-матричной, принятой в системе MATLAB. Если в обозначении используется два индекса, то, как правило, первый индекс (индекс строки) указывает адресата, или пункт назначения, а второй индекс (индекс столбца) – источник структурной схемы сети. Структурные схемы создаются системой автоматически с помощью команды gensim. Если элементом вектора или матрицы на структурной схеме является сложный объект, то используются соответственно ячейка и массив ячеек.

Программным представлением, или вычислительной моделью искусственной нейронный сети, является объект специального класса network, который включает массив структур с атрибутами сети и набор методов, необходимых для создания сети, а также для её инициализации, обучения, моделирования и визуализации. Класс Network имеет два общих конструктора, один из которых не имеет параметров и обеспечивает создание массива структур с нулевыми значениями полей, а второй – имеет минимальный набор параметров для создания модели нейронной сети, достраиваемой затем до нужной конфигурации с помощью операторов присваивания. Для создания нейронных сетей определённого вида используются специальные конструкторы.

Практические задания

Задание 1. Создать вычислительную модель нейронной сети с двумя входами, тремя слоями и одним выходом, используя общий конструктор сети с параметрами

net = network (numInputs, numLayers, biasConnect, imputConnect, layerConnect, outputConnect, tartegtConnect).

Связи между слоями должны быть только прямыми, входы необходимо соединить с первым слоем, а выход – с последним. Первый слой должен иметь смещения.
Смысл и значения параметров конструктора для создания модели сети заданной архитектуры:

numImputs =2 – количество входов сети;
numLayers =3 – количество слоёв в сети;
biasConnect =[1; 0; 0] – матрица связности для смещений размера numLayers * 1;
inputConnect =[1 1; 0 0; 0 0] – матрица связности для входов размера numLayers * numImputs;
layerConnect=[0 0 0; 1 0 0; 0 1 0] – матрица связности для слоёв размера numLayers * numLayers;
outputConnect=[0 0 1] – матрица связности для выходов размера 1* numLayers;
targetConnect=[0 0 1] – матрица связности для целей размера 1 * numLayers.

Порядок выполнения заданий следующий:

1. Создать шаблон сети, выполнив команду
net = network (2, 3, [1; 0; 0], [1 1; 0 0 ; 0 0], …. , [0 0 0 ; 1 0 0 ; 0 1 0], [0 0 1])

2. Проверить значения полей вычислительной модели нейронной сети net и их соответствие заданным значениям в списке параметров.

3. Проверить значения вычисляемых полей модели, которые дополняют описание архитектуры сети

Пример:
```
>> Net.numInputs
ans =
    2

>> Net.InputConnect
ans =
    1    1
    0    0
    0    0
```

Пример: (количество выходов сети, максимальное значение задержки для входов сети, максимальное значение задержки для слоев сети):
```
>> Net.numOutputs
ans =
    1

>> Net.numInputDelays
ans =
    0

>> Net.numLayerDelays
ans =
    0
```

numOutputs = 1 – количество выходов сети;
numInputDelays = 0 – максимальное значение задержки для входов сети.
numLayersDelays = 0 – максимальное значение задержки для слоёв сети.

Заметим, что каждый выход присоединяется к одному или нескольким слоям при этом количество компонент выхода равно количеству нейронов в соответствующем слое. Для

увеличения возможности модели в сеть включают линии задержки либо на её входах, либо между слоями. Каждая линия задерживает сигнал на один такт.
Параметры numInputDelays и NumLayerDelays определяют максимальное число линий для какого-либо входа или слоя соответственно.

4. Проанализировать структурную схему построенной сети, выполнив команду **gensim(net)** и детализируя блоки с помощью двойного щелчка левой клавиши мыши по рассматриваемому блоку.

На структурных схемах искусственных нейронных сетей в пакете NNT используются следующие обозначения:

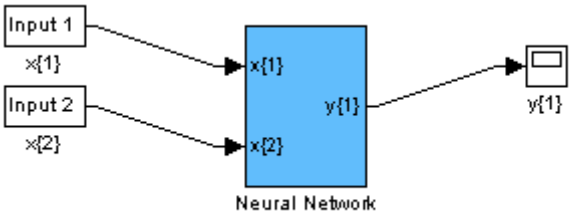


Рис.7.1. Свернутая нейросеть.

а) Neural Network – искусственная нейронная сеть с обозначениями входов p{1}, p{2}, … и выхода y{1};
б) входы Input1 , или p{1} и Input2 , или p{2};
в) дисплей y{1};

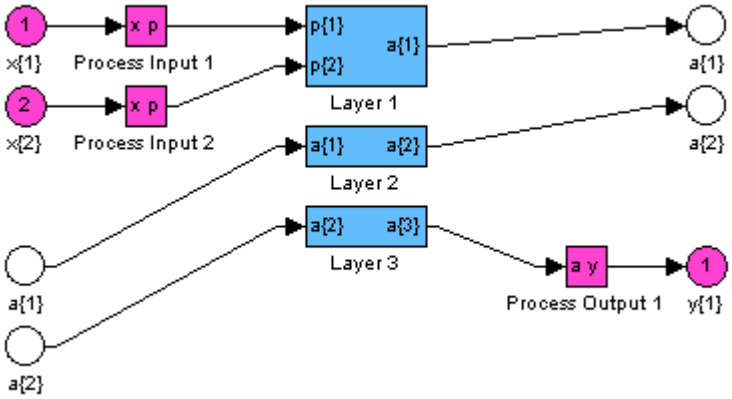


Рис.7.2. Архитектура нейросети.

г) Layer 1, Layer 2, Layer 3, … слои нейронов с обозначениями входов p{1}, p{2], a{1}, a{2}, … и выходов a{1}, a{2}, a{3}, … , y{1};

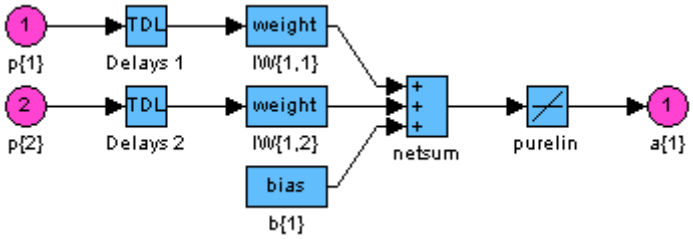


Рис.7.3. Архитектура первого слоя нейросети.

д) TDL – линии задержки (Time Delay) с именами Delays1, Delays2, ..., которые обеспечивают задержку входных сигналов или сигналов между слоями нейронов на 1, 2, 3, … такта;
е) Weights – весовая матрица для входных сигналов или сигналов между слоями нейронов; размер матрицы весов для каждого вектора входа S×R, где S – число нейронов входного слоя, а R – число компонент вектора входа, умноженное на число задержек; размер матрицы для сигналов от слоя j к слою i равен S×R, где S – число нейронов в слое i, а R – число нейронов в слое j, умноженное на число задержек;

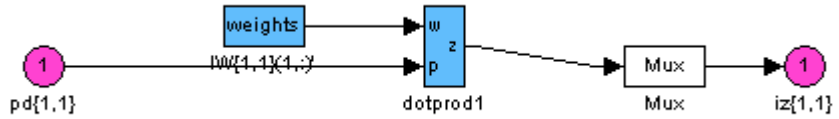


Рис.7.4.Задание весовых коэффициентов синаптических связей.

ж) dotprod – блок взвешивания входных сигналов и сигналов между слоями, на выходе которого получается сумма взвешенных, т. е. умноженных на соответствующие веса компонент сигнала;
з) mux – концентратор входных сигналов и сигналов между слоями, преобразует набор скалярных сигналов в вектор, а набор векторов в один вектор суммарной длины;
и) netsum – блок суммирования компонент для каждого нейрона слоя: компонент от нескольких векторов входа с учётом задержек, смещения и т. п.;
к) hardlim, purelin и т. д. – блоки функций активации;
л) pd{1, 1}, pd{1, 2}, ad{2, 1}, ... – сигналы после линий задержки (d - delay);
м) iz{1, 1}, iz{1, 2}, lz{2, 1}, lz{3, 2} – вектор-сигналы с выхода концентратора;
н) bias – блок весов смещений для слоя нейронов;
о) IW – массив ячеек с матрицами весов входов: IW{i, j} – матрицы для слоя i от входного вектора j;

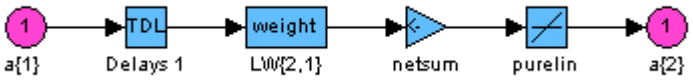


Рис.7.5. Установка параметров связей между слоями.

п) LW – массив ячеек с матрицами весов для слоёв: LW{i, j} – матрицы для слоя i от слоя j.
5. Проанализировать все параметры каждого блока структурной схемы рассматриваемой нейронной сети и в случае необходимости обратиться к справочной системе пакета NNT.
6. Задать нулевые последовательности сигналов для входов
P = [0 0 ; 0 0]
и произвести моделирование сети
A = sim(net, P).

7. Задать диапазоны входных сигналов и весовые матрицы с помощью следующих присваиваний:
net.inputs{1}.range = [0 1];
net.inputs{2}.range = [0 1];
net.b{1} = - ¼;
net.IW{1, 1} = [0.5];
net.IW{1, 2} = [0.5];
net.LW{2, 1} = [0.5];

net.LW{3, 2} = [0.5].
Исполнить команду **gensim(net)** и проверить параметры блока.

8. Вывести на экран поля вычислительной модели и их содержимое, используя функцию **celldisp.** Убедиться в правильности значений полей модели.
Пример:
celldisp(Net.IW)
celldisp(Net.LW)
celldisp(Net.b)

9. Промоделировать созданную статическую сеть, т. е. сеть без линий задержки, используя групповое и последовательное представление входных сигналов

PG = [0.5 1 ; 1 0.5];
PS = {[0.5 1] [1 0.5]};
AG = sim(net, PG);
AS = sim(net, PS).

Убедиться, что для статической сети групповое и последователь-
ное представления входных сигналов дают один и тот же результат.

10. Дополнить архитектуру созданной нейронной сети линиями задержки для входных сигналов и для сигналов между 2-м и 3-м слоями, превратив таким образом статическую сеть в динамическую:

net.inputWeights{1, 1}.delays = [0 1];
net.inputWeights{1, 2}.delays = [0 1];
net.layerWeights{3, 2}.delays = [0 1 2].

Построить структурную схему динамической сети и выяснить смысл используемых операторов присваивания.

11. Скорректировать весовые матрицы:
net.IW{1, 1} = [0.5 0.5];
net.IW{1, 2} = [0.5 0.25];
net.LW{3, 2} = [0.5 0.25 1].

12. Промоделировать динамическую сеть, используя групповое и последовательное представление входных сигналов:

AG = sim(net, PG);
AS = sim(net, PS).

Убедиться, что групповое представление входных сигналов искажает результат, так как в этом случае работа одной сети заменяется параллельной работой двух (по числу последовательностей) одинаковых сетей с нулевыми начальными значениями сигналов на выходах линий задержки.

13. Вывести на печать поля вычислительной модели и их содержимое, используя функцию celldisp.

14. Сохранить содержимое командного окна в М-файле для последующего использования.

Задание 2.

Создать точно такую же динамическую сеть **asgne**t, используя конструктор класса **network** без параметров и задавая значения соответствующих полей вычислительной модели с помощью операторов присваивания.

asgnet.numImputs =2 – количество входов сети;
asgnet.numLayers =3 – количество слоёв в сети;
asgnet.biasConnect =[1; 0; 0] – матрица связности для смещений размера numLayers * 1;
asgnet.inputConnect =[1 1; 0 0; 0 0] – матрица связности для входов размера numLayers * numImputs;
asgnet.layerConnect =[0 0 0; 1 0 0; 0 1 0] – матрица связности для слоёв размера numLayers * numLayers;
asgnet.outputConnect=[0 0 1] – матрица связности для выходов размера 1* numLayers;
asgnet.targetConnect=[0 0 1] – матрица связности для целей размера 1 * numLayers.

Убедиться в идентичности сетей net и asgnet.
Сравнить результаты работы полученных сетей.

Задание 3.

Используя конструктор класса **network** с параметрами и операторы присваивания для полей и ячеек объектов этого класса, построить, выдать на экран и промоделировать искусственные нейронные сети следующей архитектуры:

а) однослойная сеть с тремя нейронами, тремя входами и одним выходом;

б) двухслойная сеть с двумя нейронами в каждом слое, двумя входами и одним выходом.

Для выполнения задания 3(б) предлагается следующая последовательность действий:

1. Строится базовая двухслойная сеть средствами конструктора класса **network**, например с помощью следующего m-файла:

```
numInputs = 2              % количество входов сети;
numLayers = 2              % количество слоёв в сети;
biasConnect = [1; 1]       % матрица связности для смещений размера
                                 numLayers * 1;
inputConnect=[1 1; 0 0]    % матрица связности для входов размера
                                 numLayers * numImputs;
layerConnect=[0 0; 1 0]    % матрица связности для слоёв размера
                                 numLayers * numLayers;
outputConnect=[0 1]        % матрица связности для выходов размера
                                 1* numLayers;

net = network(numInputs, numLayers, biasConnect,inputConnect,layerConnect,
          outputConnect)
```

```
net.layers{1}.transferFcn = 'hardlim';
net.layers{2}.transferFcn = 'hardlim';

gensim(net);
```

2. На основе базовой структурной нейросети, выполняя последовательно операции копирования нейронов в слоях и, при необходимости, корректируя связи, формируется нейросеть требуемой структуры:

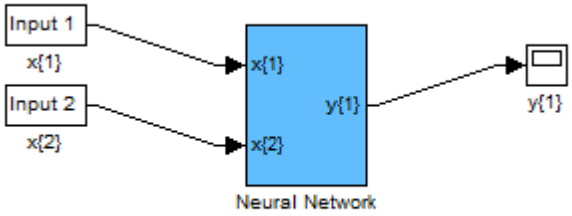


Рис. 7.6. Свернутая базовая нейросеть

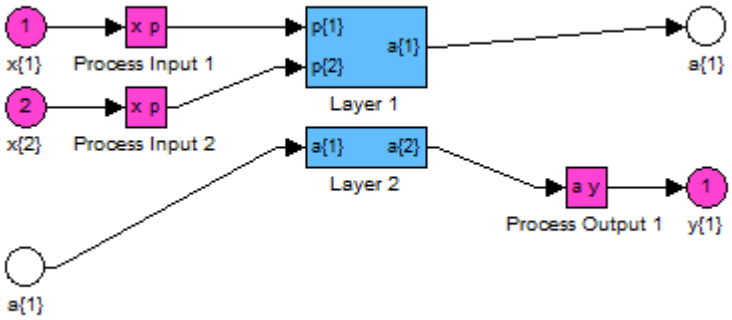


Рис. 7.7. Связи между слоями базовой нейросети

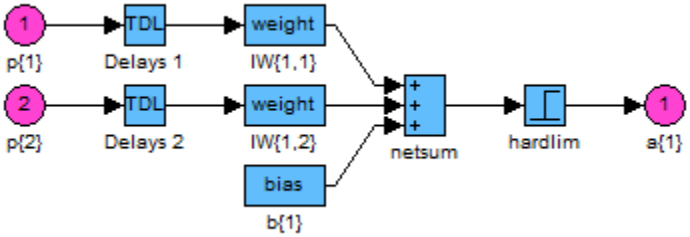


Рис. 7.8.Архитектура первого слоя базовой нейросети.

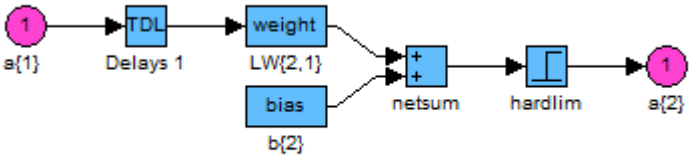


Рис. 7.9. Архитектура второго слоя базовой нейросети.

1. На основе базовой структурной нейросети, выполняя последовательно операции копирования нейронов в слоях и, при необходимости, корректируя связи, формируется нейросеть требуемой структуры:

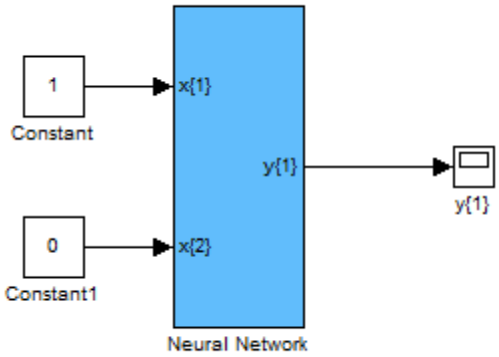


Рис. 7.10. Свернутая нейросеть, выполняющая логическую операцию XOR.

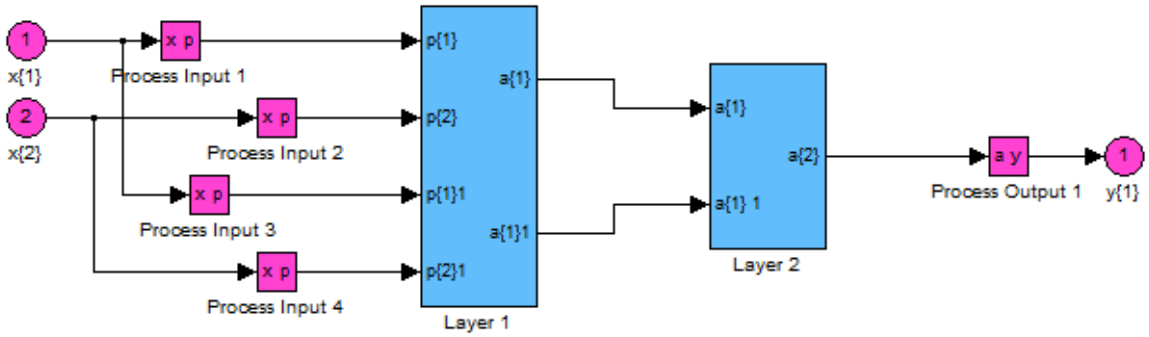


Рис. 7.11. Связи между слоями нейросети.

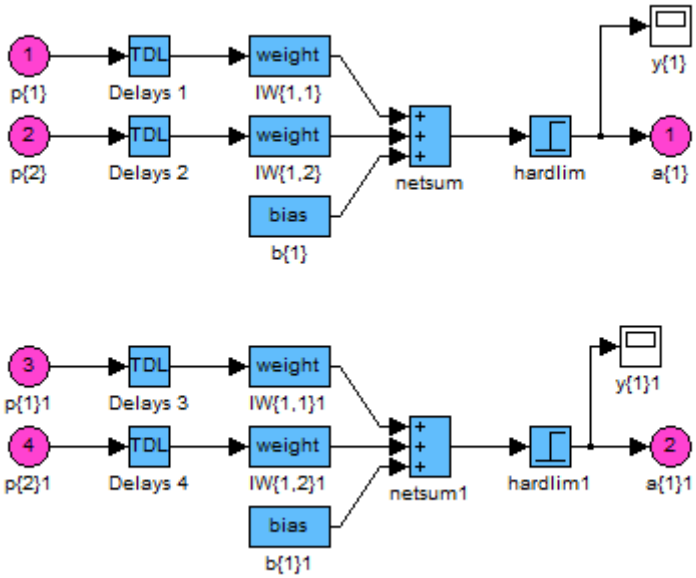


Рис. 7.12. Архитектура первого слоя нейросети

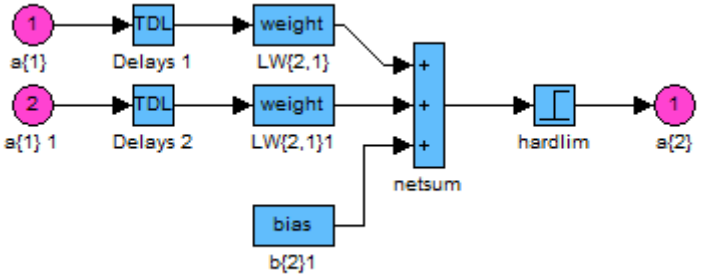


Рис. 7.13. Архитектура второго слоя нейросети.

в) настроить нейросеть на выполнение логической операции XOR для двоичных переменных x1 и x2.

г) трёхслойная сеть с прямой передачей сигналов и с тремя нейронами в каждом слое; количество входов – три с двумя, пятью и тремя компонентами; для всех слоёв имеется смещение; выход – один.

4. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

5.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных в разделе 4 настоящего описания.

Лабораторная работа №8

ФОРМИРОВАНИЕ И ОБУЧЕНИЕ НЕЙРОСЕТИ С ПОМОЩЬЮ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ MATLAB

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

ознакомиться на практике с нейросетями, особенностями их применения и использованием в среде Matlab.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

АЛГОРИТМЫ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

Адаптация и самоорганизация искусственных нейронных сетей достигается в процессе их обучения. Целью *обучения* нейронных сетей является такая подстройка их весов, которая обеспечивала бы для некоторого множества входов требуемое множество выходов.

При решении прикладных задач с помощью нейронных сетей необходимо собрать представительный объем данных, для того чтобы обучить нейронную сеть решению таких задач. Обучающий набор данных – это набор наблюдений, содержащих признаки изучаемого объекта. Первоначальный выбор признаков осуществляется на основе имеющегося опыта, с учетом мнения экспертов. Вопрос о том, сколько необходимо иметь наблюдений для обучения сети, часто оказывается непростым. Известен ряд эвристических правил, которые устанавливают связь между количеством используемых для обучения наблюдений и размерами сети. Простейшее из них гласит, что количество наблюдений должно быть в 10 раз больше числа связей в сети. Процесс обучения – это процесс определения параметров модели процесса или явления, реализуемого нейронной сетью. Ошибка обучения для конкретной конфигурации сети определяется после прогона через сеть всех имеющихся наблюдений и сравнения выходных значений с целевыми значениями в случае обучения с учителем. Соответствующие разности позволяют сформировать так называемую функцию ошибок. Если ошибка сети, выходной слой которой имеет n нейронов, $e_i = y_i - t_i$ есть разность между реальным и желаемым сигналами на выходе i-го нейрона, то в качестве функций ошибок могут быть использованы следующие функции:

- сумма квадратов ошибок sse = $\sum_{i-1}^{n} e_i^2$,
- средняя квадратичная ошибка mse = $\frac{1}{n} \sum_{i-1}^{n} e_i^2$,

- регулируемая или комбинированная ошибка

  msereg = $\frac{\gamma}{n}\sum_{i=1}^{n} e_i^2 + \frac{1-\gamma}{n}\sum_{i=1}^{n} e_i^2$, где γ - параметр регуляции,

- средняя абсолютная ошибка mae = $\frac{1}{n}\sum_{i=1}^{n}|e_i|$.

При моделировании нейронных сетей с линейными функциями активации нейронов можно построить алгоритм, гарантирующий достижение абсолютного минимума ошибки обучения. Для нейронных сетей с нелинейными функциями активации в общем случае нельзя гарантировать достижение глобального минимума функции ошибки. Поверхность функции ошибок определяется как совокупность точек-значений ошибок в N+1-мерном пространстве всевозможных сочетаний весов и смещений с общим числом N. Цель обучения при геометрическом анализе или изучении поверхности ошибок состоит в том, чтобы найти на ней глобальный минимум. В случае линейной модели сети и минимизации суммы квадратов ошибок поверхность ошибок представляет собой параболоид, имеющий единственную точку минимума. В случае нелинейной модели поверхность ошибок имеет более сложное строение и может иметь локальные минимумы, плоские участки, седловые точки и длинные узкие овраги. Как правило, при обучении такой сети вычисляется градиент функции ошибок в случайно выбранной точке поверхности, а затем эта информация используется для продвижения вниз по склону. Алгоритм продвижения завершается в точке минимума, локального или глобального. По существу алгоритмы обучения нейронных сетей аналогичны алгоритмам поиска глобального экстремума функции многих переменных.

Сети с большим количеством весов позволяют воспроизводить сложные функции, но в этом случае возможно так называемое «переобучение» сети, когда ошибки обучения малы, но полученная модель имеет слабое отношение к истинной зависимости. Нейронная сеть же с небольшим количеством весов может оказаться недостаточно гибкой, чтобы смоделировать имеющуюся зависимость. Для преодоления эффекта переобучения используется механизм контрольной проверки. Часть обучающих наблюдений резервируется как контрольные наблюдения и не используется при обучении сети. Если контрольная ошибка перестает убывать или начинает расти, то это означает, что сеть слишком близко следует исходным данным и обучение следует остановить. В этом случае следует уменьшить количество нейронов или слоев, так как сеть является слишком мощной для решаемой задачи. Если же сеть имеет недостаточную мощность для воспроизведения имеющейся зависимости, явление переобучения, скорее всего, наблюдаться не будет и обе ошибки, обучения и проверки, не достигнут достаточно малого значения.

Способность сети, обученной на некотором множестве данных выдавать правильные результаты для достаточно широкого класса новых данных, в том числе и не представленных при обучении, называется *свойством обобщения* нейронной сети.

Для настройки параметров нейронных сетей широко используется также процедура *адаптации,* когда подбираются веса и смещения с использованием произвольных функций их настройки, обеспечивающие соответствие между входами и желаемыми значениями на выходе.

Методы определения экстремума функции нескольких переменных делятся на три категории – методы нулевого, первого и второго порядка:

- методы *нулевого порядка*, в которых для нахождения экстремума используется только информация о значениях функции в заданных точках;
- методы *первого порядка*, где для нахождения экстремума используется градиент функционала ошибки по настраиваемым параметрам;
- методы *второго порядка*, вычисляющие матрицу вторых производных функционала ошибки (матрицу Гессе).

Кроме перечисленных методов можно выделить также *стохастические алгоритмы* оптимизации и алгоритмы *глобальной оптимизации*, перебирающие значения аргументов функции ошибки.

В пакете NNT системы MATLAB реализованы два способа адаптации и обучения, последовательный и групповой, в зависимости от того, применяется ли последовательное или групповое представление входов и целевого вектора (массив ячеек cell и массив формата double соответственно).

Названия процедур обучения и адаптации нейроимитатора NNT системы MATLAB приводятся по мере рассмотрения соответствующих правил обучения нейронных сетей.

1. ПРАВИЛА ОБУЧЕНИЯ ХЕББА

Правила обучения Хебба относятся к ассоциативным обучающим правилам. Согласно правилу Хебба обучение сети происходит в результате усиления силы связи между одновременно активными элементами, его можно определить так:

$$w_{ij}(t+1) = w_{ij}(t) + x_i \cdot y_j,$$

где t – время, x_i, y_j – соответственно выходные значения i-го и j-го нейронов. В начальный момент предполагается, что $w_{ij}(0)=0$ для всех i и j. Правило Хебба можно использовать как при обучении с учителем, так и без него. Если в качестве выходных значений сети используются целевые значения, тогда реализуется обучение с учителем. При использовании в качестве Y реальных значений, которые получаются при подаче входных образов на вход сети, правило Хебба соответствует обучению без учителя. В этом случае коэффициенты сети в начальный момент времени инициализируются случайным образом. Обучение заканчивается после подачи всех имеющихся входных образов на нейронную сеть. В общем случае это правило не гарантирует сходимости процедуры обучения сети.

2. ПРАВИЛА ОБУЧЕНИЯ ПЕРСЕПТРОНА

Персептрон – нейронная сеть прямой передачи сигнала с бинарными входами и бинарной пороговой функцией активации.

Правило обучения Розенблатта в общем случае является вариантом правил обучения Хебба, формирующих симметричную матрицу связей, и в тех же обозначениях имеет вид:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot (t_i - y_j) \cdot x_i ,$$

где η – коэффициент обучения, $0<\eta<1$, t_j – эталонные или целевые значения. Правило обучения персептрона с нормализацией используется тогда, когда имеется большой разброс в значениях компонент входного вектора. Тогда каждая компонента делится на длину вектора:

$$x_i' = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}}, \qquad i=1,2,\ldots,n.$$

В нейроимитаторе NNT обучающие процедуры настройки персептрона, обычная и с нормализацией, соответственно, learnp и learnpn.

Алгоритм обучения Розенблатта включает такие шаги:

1. Весовые коэффициенты сети инициализируются случайным образом или устанавливаются нулевыми.
2. На вход сети поочередно подаются входные образы из обучающей выборки, которые затем преобразуются в выходные сигналы Y.
3. Если реакция нейронной сети y_j совпадает с эталонным значением t_j, то весовой коэффициент w_{ij} не изменяется.
4. Если выходное значение не совпадает с эталонным значением, то есть y_j не равно t_j, то производится модификация весовых коэффициентов w_{ij} по соответствующей формуле.
5. Алгоритм повторяется до тех пор, пока для всех входных образов не будет достигнуто совпадение с целевыми значениями или не перестанут изменяться весовые коэффициенты.

Линейная разделяющая поверхность, формируемая персептроном, позволяет решать ограниченный круг задач.

3. ПРАВИЛО ОБУЧЕНИЯ ВИДРОУ – ХОФФА

Правило Видроу – Хоффа используется для обучения сети, состоящей из слоя распределительных нейронов и одного выходного нейрона с линейной функцией активации. Такая сеть называется адаптивным нейронным элементом (Adaptive Linear Element) или «ADALINE». Выходное значение такой сети определяется по формуле

$$y = \sum_{i=1}^{n} w_{i1} \cdot x_i - S_0 ,$$

где n – число нейронов распределительного слоя, S_0 – смещение.

Правило обучения Видроу – Хоффа известно под названием дельта-правила (delta-rule). Оно предполагает минимизацию среднеквадратичной ошибки нейронной сети, которая для L входных образов определяется следующим образом:

$$E = \frac{1}{2} \cdot \sum_{k=1}^{L} \left(y^{(k)} - t^{(k)} \right)^2,$$

где $y^{(k)}$ и $t^{(k)}$ – выходное и целевое значения сети соответственно для k-го образа. Правило Видроу – Хоффа базируется на методе градиентного спуска в пространстве весовых коэффициентов и смещений нейронной сети. По этому методу веса и смещения изменяются с течением времени следующим образом:

$$w_{i1}(t+1) = w_{i1}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{i1}(t)}, \quad S_0(t+1) = S_0(t) - \alpha \cdot \frac{\partial E(k)}{\partial S_0(t)}, \quad i=1,2,\ldots,n,$$

где α – скорость обучения, $0<\alpha<1$.

Вычислим частные производные:

$$\frac{\partial E(k)}{\partial w_{i1}(t)} = \frac{\partial E(k)}{\partial y^{(k)}} \cdot \frac{\partial y^{(k)}}{\partial w_{i1}(t)} = (y^{(k)} - t^{(k)}) \cdot x_i^{(k)},$$

$$\frac{\partial E(k)}{\partial \mathrm{T}(t)} = \frac{\partial E(k)}{\partial y^{(k)}} \cdot \frac{\partial y^{(k)}}{\partial S_0(t)} = -(y^{(k)} - t^{(k)}),$$

где $x_i^{(k)}$ – i-я компонента k-го образа. По дельта-правилу

$$w_{i1}(t+1) = w_{i1}(t) - \alpha \cdot (y^{(k)} - t^{(k)}) \cdot x_i^{(k)},$$

$$S_0(t+1) = S_0(t) + \alpha \cdot (y^{(k)} - t^{(k)}), \qquad i = 1,2,\ldots,n.$$

Выражение для пересчета весов сети эквивалентно правилу обучения Розенблатта, если на выходе линейного адаптивного элемента использовать пороговый элемент. Видроу и Хофф доказали, что минимизация среднеквадратичной ошибки сети производится независимо от начальных значений весовых коэффициентов. Суммарная среднеквадратичная ошибка сети E должна достигнуть заданного минимального значения E_{min}. Для алгоритма Видроу – Хоффа рекомендуется выбирать значение скорости обучения α по следующему правилу: $\alpha(l) = \frac{1}{l}$, где l – номер итерации в алгоритме обучения. Адаптивный шаг обучения для линейной нейронной сети позволяет повысить скорость обучения и определяется по формуле [2]:

$$\alpha(t) = \frac{1}{1 + \sum_{i=1}^{n} x_i^2(t)}.$$

В пакете NNT процедура настройки для обучения по правилу Видроу – Хоффа – learnwh.

4. МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Метод обратного распространения ошибки предложен несколькими авторами независимо в 1986 г. для многослойных сетей с прямой передачей сигнала. Многослойная нейронная сеть способна осуществлять любое отображение входных векторов в выходные (рис. 1).

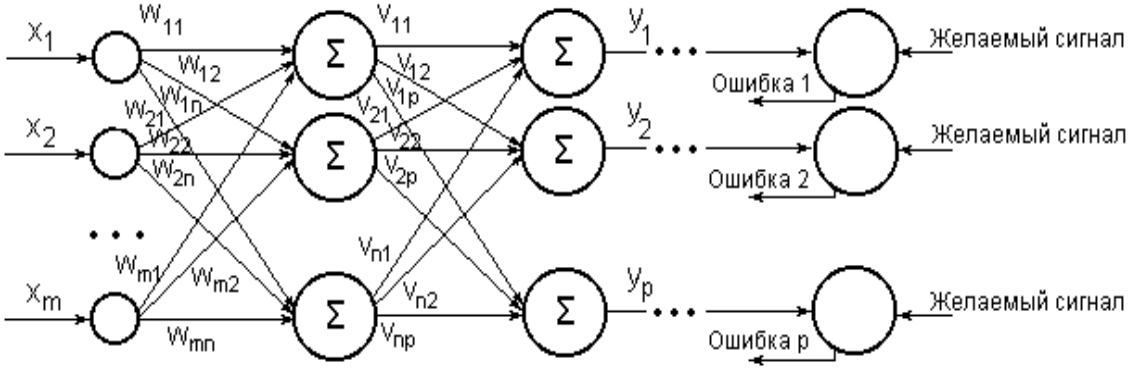


Рис. 1. Многослойная сеть со скрытым слоем из *n* нейронов.

Входной слой нейронной сети выполняет распределительные функции. Выходной слой нейронов служит для обработки информации от предыдущих слоев и выдачи результатов. Слои нейронных элементов, расположенные между входным и выходным слоями, называются скрытыми (hidden). Как и выходной слой, скрытые слои являются обрабатывающими. Выход каждого нейрона предыдущего слоя сети соединен синаптическими связями со всеми входами нейронных элементов следующего слоя. В качестве функций активации в многослойных сетях чаще других используются логистическая функция и гиперболический тангенс. Метод обратного распространения ошибки минимизирует среднеквадратичную ошибку нейронной сети, при этом используется метод градиентного спуска в пространстве весовых коэффициентов и смещений нейронных сетей. Согласно методу градиентного спуска по поверхности ошибок изменение весовых коэффициентов и смещений сети осуществляется по формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{ij}(t)},$$

$$S_{0j}(t+1) = S_{0j}(t) - \alpha \cdot \frac{\partial E(k)}{\partial S_{0j}(t)}, \qquad i=1,2,\ldots,n; \quad j=1,2,\ldots,p,$$

где E – среднеквадратичная ошибка сети для одного из k образов, определяемая по формуле

$$E = \frac{1}{2}\sum_{j=1}^{p}(y_j - t_j)^2 ,$$

здесь t_j – желаемое или целевое выходное значение j-го нейрона.

Ошибка j-го нейрона выходного слоя равна: $\gamma_j = y_j - t_j$.

Для любого скрытого слоя ошибка i-го нейронного элемента определяется рекурсивно через ошибки нейронов следующего слоя j:

$$\gamma_i = \sum_{j=1}^{m}\gamma_j \cdot F'(S_j) \cdot w_{ij} ,$$

где m – количество нейронов следующего слоя по отношению к слою i, w_{ij} – вес или синаптическая связь между i-ым и j-ым нейронами разных слоев, S_j – взвешенная сумма j-го нейрона.

Весовые коэффициенты и смещения нейронных элементов с течением времени изменяются следующим образом:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i ,$$

$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot F'(S_j), \qquad i=1,2,\ldots,n, \quad j=1,2,\ldots,p,$$

где α – скорость обучения. Это правило называется <u>обобщенным дельта-правилом</u>. Для логистической функции активации:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot y_j(1-y_j) \cdot y_i$$

$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot y_j(1-y_j), \quad i=1,2,\ldots,n, \quad j=1,2,\ldots,p,$$

ошибка j-го нейрона выходного слоя определяется как

$$\gamma_j = y_j - t_j,$$

а j-го нейрона скрытого слоя

$$\gamma_j = \sum_{i=1}^{m}\gamma_j \cdot y_i \cdot (1-y_i) \cdot w_{ij} , \text{ где}$$

m – количество нейронов следующего слоя по отношению к слою j.

К недостаткам метода обратного распространения ошибки относят следующие:

- медленную сходимость градиентного метода при постоянном шаге обучения;
- возможное смешение точек локального и глобального минимумов;
- влияние случайной инициализации весовых коэффициентов на скорость поиска минимума.

Для их преодоления предложено несколько модификаций алгоритма обратного распространения ошибки:

1) с импульсом, позволяющим учесть текущий и предыдущий градиенты (метод тяжелого шарика), изменение веса тогда:

$$\Delta w_{ij}(t+1) = \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i + \eta \cdot \Delta w_{ij}(t),$$

где α – коэффициент скорости обучения, η – импульс или момент, обычно $0<\alpha<1$, $\eta \approx 0{,}9$;

2) с адаптивным шагом обучения, изменяющимся по формуле

$$\alpha(t) = \frac{1}{1 + \sum x_i^2(t)};$$

3) с модификацией по Розенбергу, предложенной им для решения задачи преобразования английского печатного текста в качественную речь:

$$\Delta w_{ij}(t+1) = (1-\alpha)\gamma_j \cdot F'(S_j) \cdot y_i + \alpha \cdot \Delta w_{ij}(t),$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \Delta w_{ij}(t+1).$$

Использование производных второго порядка для коррекции весов алгоритма обратного распространения ошибки, как показала практика, не дало значительного улучшения решений прикладных задач.

Процедуры, обеспечивающие настройку и обучение методом обратного распространения ошибки с импульсом и адаптивным шагом обучения, в пакете NNT названы соответственно learngdx и traingdx.

Для многослойных сетей прямой передачи сигнала с логистической функцией активации рекомендуется случайные начальные значения весов инициализировать по правилу (Р. Палмер)

$$w_{ij} \approx \frac{1}{\sqrt{n(i)}},$$

где $n(i)$ – количество нейронных элементов в слое i.

Начальные весовые коэффициенты рекомендуется выбирать в диапазоне [-0.05;0.05] или [-0.1;0.1]. При этом смещение S_0 принимает единичные значения в начальный момент времени. Кроме этого, количество нейронных элементов скрытых слоев должно быть меньше тренировочных образов. Для обеспечения требуемой обобщающей способности сети можно использовать сеть с несколькими скрытыми слоями, размерность которых меньше, чем для сети с одним скрытым слоем. Однако нейронные сети, имеющие несколько скрытых слоев, обучаются значительно медленнее.

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

сформировать и обучить нейросеть для вычисления заданной функции.

Порядок выполнения работы:

1. Запустить Matlab.

2. Ознакомиться с функциями ***newff***, ***train***, ***sim***. Использовать для этого команду help (например, “help newff”) и doc (“doc newff”).

3. Создать нейросеть командой **newff** .

Например,

net = newff ([0 25], [5 7 1], {'tansig' 'tansig' 'purelin'});

Здесь 0 и 25 - минимальное и максимальное значения входа нейросети (он здесь один), нейросеть имеет 5 нейронов в первом слое, 7 нейронов во втором слое и 1 нейрон на выходе, в качестве передаточной функции для обоих промежуточных слоев используется сигмоид, а на выходе стоит нейрон с линейной передаточной функцией.

4. Для заданной функции создать обучающие массивы P и T, где P - массив входных значений (аргумент функции), а T - массив выходных значений (результат функции).

5. Задать количество итераций обучения, например, так: net.trainParam.epochs = 150;

Здесь 150 - количество итераций (эпох) обучения нейросети.

6. Обучить нейросеть командой: net = train(net,P,T);

Посмотреть, что получилось и объяснить полученные результаты.

7. Если на шаге 6 выполнено все заданное число итераций, попробовать повторить обучение. Объяснить полученные результаты.

8. Несколько раз повторять две команды - создание сети и ее обучение. Объяснить полученные результаты.

9. Создать массив проверочных входных значений функции P1, отличающийся от P и вычислить для него значения функции (массив T1). Достаточно взять десять разных значений P1.

10. С помощью полученной нейросети предсказать, какие должны были получиться значения функции: Y = sim(net,P1);

Найти ситуации, в которых нейросеть ошибается. Пояснить, почему.

11. Сравнить реальные и предсказанные значения функции: plot(P1,T1,P1,Y,'o')

12. Сделать тоже самое для обучающих массивов: Y = sim(net,P); plot(P,T,P,Y,’o’)

13. Занести полученные графики в отчет. Объяснить увиденное.

Листинг программы, реализующей перечисленные выше шаги, приведен на рис.8.1.

```
net = newff([0 25],[5 7 1],{'tansig' 'tansig' 'purelin'});
% формирование обучающих данных
P=zeros(1, 25);
T=zeros(1, 25);
for i=0:24
    P(1, i+1) = i;
    T(1, i+1) = 2*i^1.5;
end;
plot(T)
% число итераций обучения
net.trainParam.epochs = 250;
% обучение нейросети.
net = train(net,P,T);
% формирование проверочных данных
P1=zeros(1, 25);
T1=zeros(1, 25);
for i=0:24
    P1(1, i+1) = i + 0.5;
    T1(1, i+1) = 2*(i+ 0.5)^1.5;
end;
Y = sim(net,P1);
plot(P1,T1,P1,Y,'o');
Y = sim(net,P);
figure;
plot(P,T,P,Y,'o');
```

Рис.8.1. Листинг script- функции.

В результате вызова функции **`train`** на экран выводится окно обучения нейросети (педставлено на рис.8.2) и происходит процесс настройки весов в соответствии с выбранным алгоритмом.

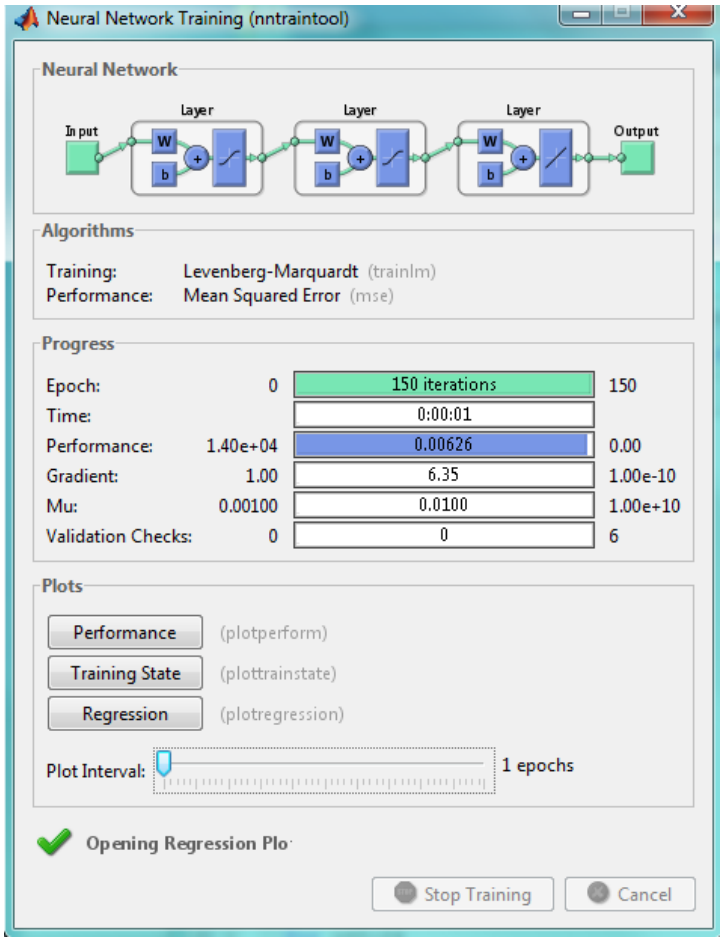


Рис.8.2.

Изменение функции качества в процессе обучения представлено на рис.8.3.

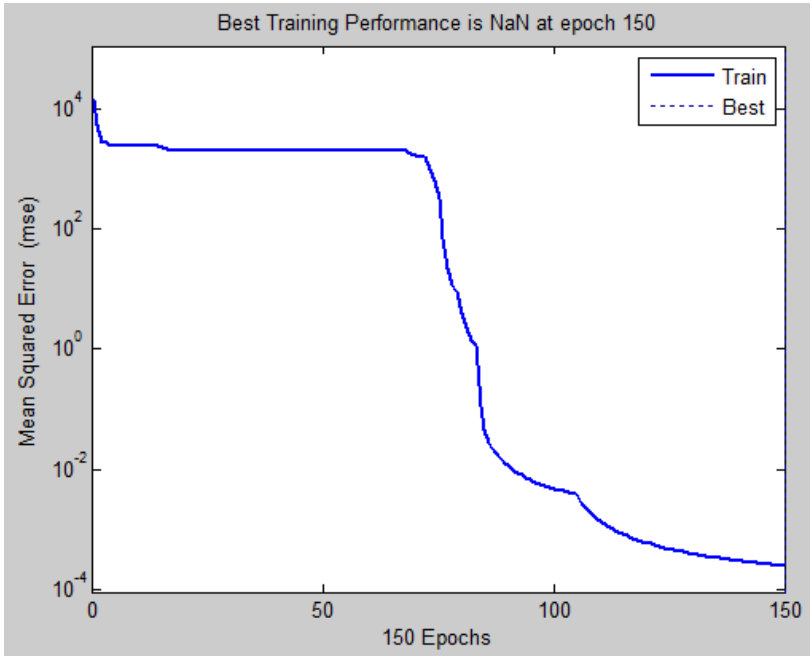


Рис.8.3.

На рис.8.4. представлены результаты расчета функции по аналитической зависимости и с помощью аппроксимации «обученной» нейросети.

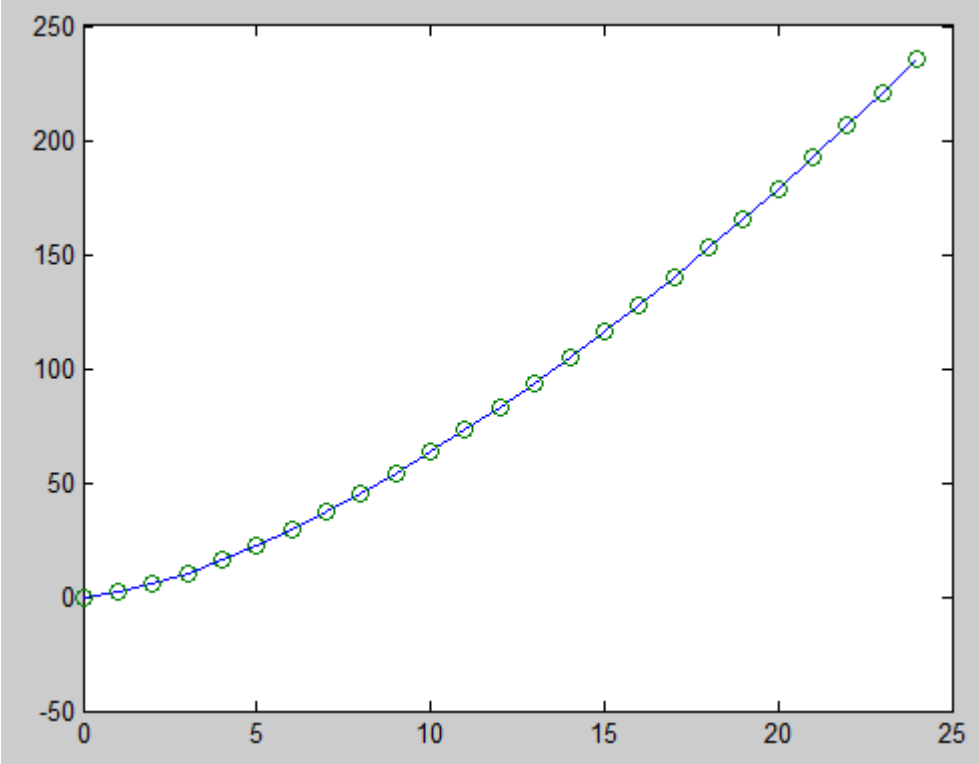


Рис.8.4.

1. Варианты аппроксимируемых функций:

1. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.4x 1.3 ; f(x) = 5, если x<4 и 8, если x>=4

2. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 2x 1.5 ; f(x) = 5, если x<3 и 10, если x>=3

3. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 3.2x 1.25 ; f(x) = 4.5, если x<3 и 8, если x>=3

4. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.72x 1.6 ; f(x) = 50, если x<7 и 78, если x>=7

5. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.5x 1.4 ; f(x) = 5.2, если x<3.45 и 9, если x>=3.45

6. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.6x 1.32 ; f(x) = 3, если x<3 и 8, если x>=3

7. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.22x 1.43 ; f(x) = 6, если x<4.2 и 2, если x>=4.2

8. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.8x 1.11 ; f(x) = 5, если x<4.1 и 3, если x>=4.1

9. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.23x 1.37 ; f(x) = 7.5, если x<9 и 1, если x>=9

10. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.45x 1.35 ; f(x) = 0.5, если x<42 и 80, если x>=42

2. Варианты функций активации:

tansig – гиперболический тангенс;

tribas – треугольная функция активации;

radbas – радиальная базисная функция активации;

logsig – сигмоидальная функция активации.

3. Количество итераций обучения:

100, 200, 300

Порядок выполнения работы.

1. Написать script – файл, реализующий процесс формирования и обучения нейросети по представленному выше алгоритму.
2. Провести процедуру обучения нейросети для трех различных функций, заданных преподавателем.
3. Вывести на экран структуру сформированной нейросети с помощью команды gensim(net). Проанализировать структуру нейросети.
4. Провести анализ влияния количества итераций обучения на качество аппроксимации функции.
5. Рассмотреть п.3 для различных функций активации.

Составить отчет. В отчете необходимо сделать выводы относительно полученных результатов.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание пакет NNT MatLab, описание математических моделей искусственных нейронов, описание активационных функций и решение заданий, представленных в разделе 4 настоящего описания.

Лабораторная работа №9

СОЗДАНИЯ И ОБУЧЕНИЯ НЕЙРОСЕТИ ЗАДАННОЙ СТРУКТУРЫ С ПОМОЩЬЮ ИНСТРУМЕНТА NETWORK/DATA MANAGER

1. ЦЕЛЬ РАБОТЫ

Знакомство с решением практической задачи создания и обучения нейросети заданной структуры с помощью инструмента Network/Data Manager.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

См.лекции.

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ.

Требуется сформировать нейросеть, имеющую два слоя (при использовании этого инструмента имеется ограничение на большее число слоев), 10 нейронов в первом слое с тангенциальными функциями активации (передаточными функциями). Нейросеть должна реализовывать нелинейную функцию:

$F(x) = Sin((x-10)/3)*x^{1.3}$ в диапазоне x=0..25.

Алгоритм решения задачи

1 шаг. Формируем обучающие и проверочные данные для заданной функции. С этой целью можно создать и выполнить следующий script – файл (рис.4.1):

```
P=zeros(1, 25);
T=zeros(1, 25);
for i=0:24
   P(1, i+1) = i;
   T(1, i+1) = sin((i-10)/3)*i^1.3;
end;

P1=zeros(1, 25);
T1=zeros(1, 25);
for i=0:24
   P1(1, i+1) = i + 0.5;
   T1(1, i+1) = sin(((i+0.5)-10)/3)*(i+0.5)^1.3;
end;
```

Рис.9.1.

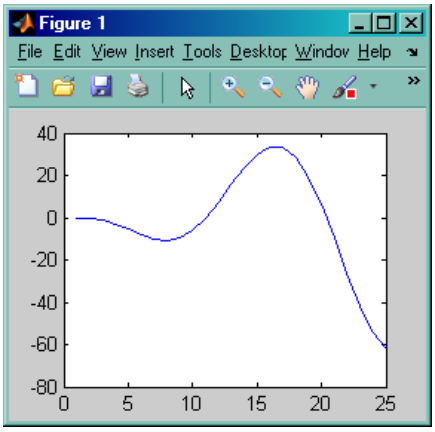


Рис.9.2.

График функции, построенный для диапазона изменения аргумента 0..24 представлен на рис.9.2.

По окончании данного шага (по результатам работы script – функции) в рабочей области MatLab будут сформированы массивы обучающих и проверочных входов и целей (выходов):

P, P1, T, T1.

2 шаг. Вызвать инструмент Network/Data Manager, набрав в командной строке матлаба nntool.

Появится окно (рис.9.3):

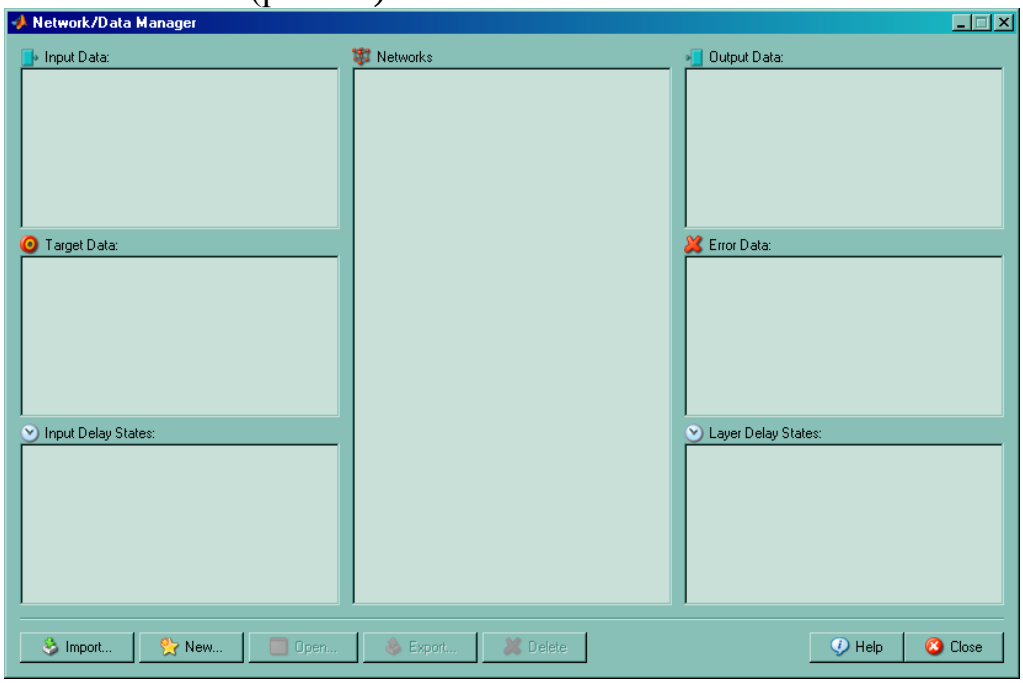


Рис.9.3.

2. добавить обучающие и проверочные данные в окна Input Data и Target Data с помощью кнопки **Import**, импортировав их из переменных MatLab (рис.9.4.).

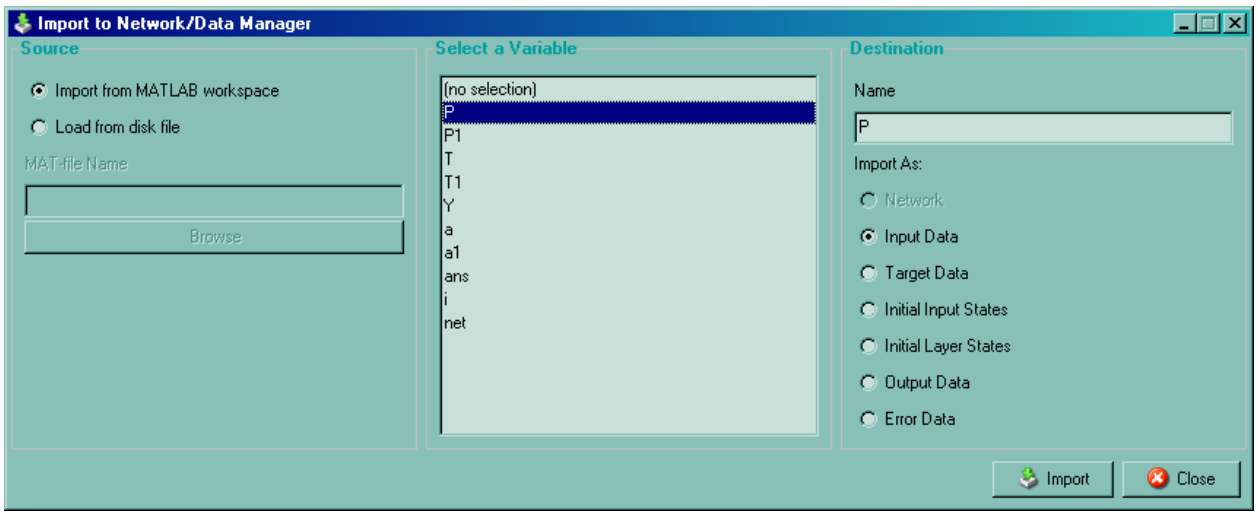


Рис.9.4.

После этого массивы обучающих и проверочных входов и целей (выходов) (P, P1, T, T1) должны оказаться в окнах Input Data и Target Data (рис.9.5):

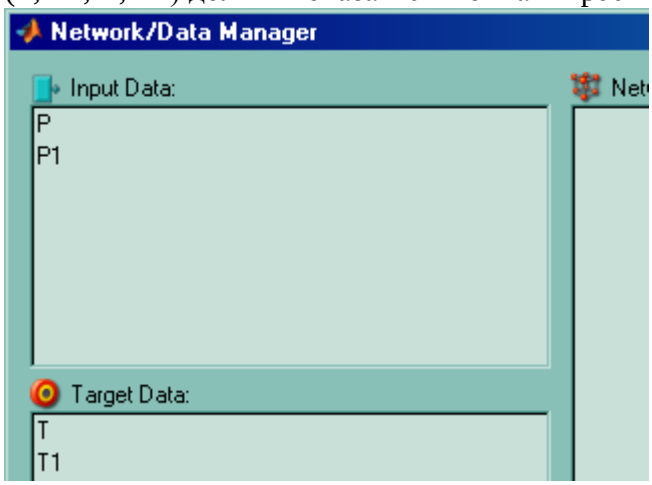


Рис.9.5.

3 шаг. Создание нейросети заданной структуры. С этой целью необходимо «нажать» кнопку **New** (рис.9.6).

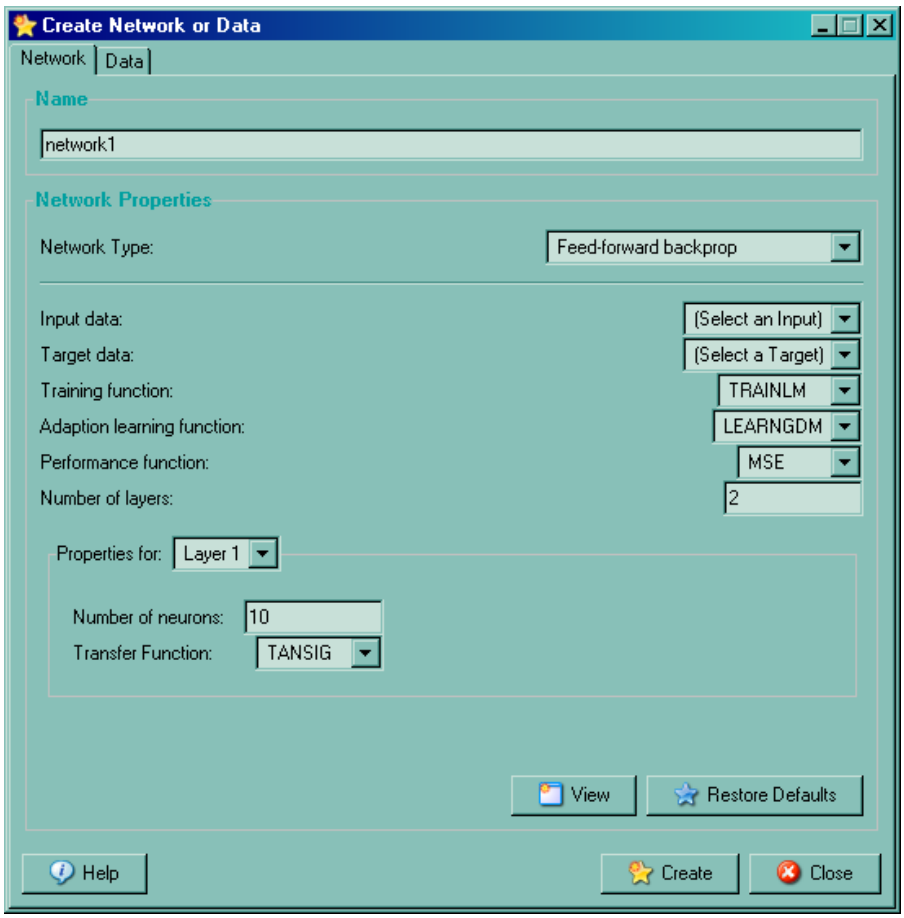


Рис.9.6.

Здесь необходимо указать параметры сети.

Network Type – тип используемой сети.

Input Data, Target Data – обучающие данные.

Training Function – обучающий алгоритм

Perfomance Function – Функция определяющая качество обучения (отклонение выхода нейросети от выходных данных обучающей выборки)

Number of layers – число слоев

Proprties for, Number Neurons, Transfer Function – параметры для каждого слоя, число нейронов и передаточная функция.

В соответствии с заданием, выберем 2 слоя, 10 нейронов в первом слое и тангенциальные передаточные функции.

Необходимо задать имена соответствующих переменных в окнах “Input data” и “Target data”.

4 шаг. Контроль структуры сформированной нейросети.

Нажав кнопку **View** можно посмотреть структурную схему создаваемой сети (рис.9.7):

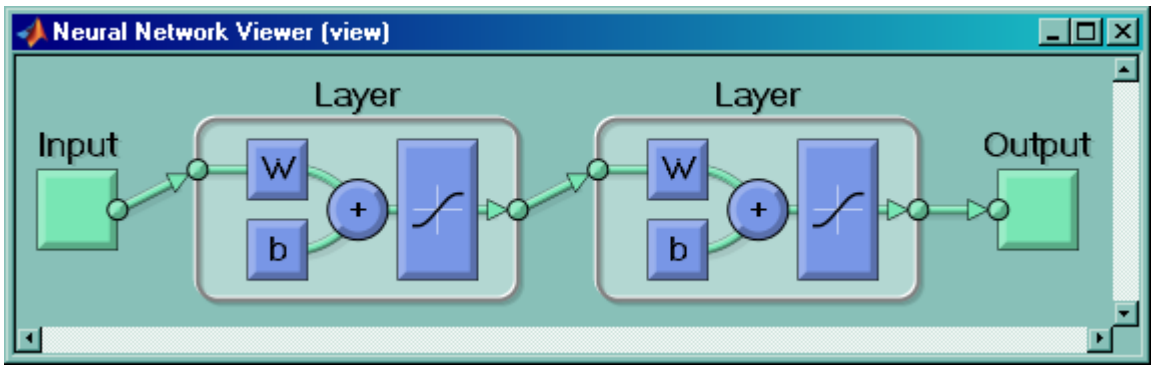


Рис.9.7.

При нажатии кнопки **Create** сеть создается и добавляется в окно Networks (рис.9.8):

Рис.9.8.

Двойной щелчек левой клавишей мыши по созданной сети вызывает окно работы с сетью, позволяющей просматривать ее структуру, обучать, симулировать, адаптировать и редактировать веса нейронов (рис.9.9):

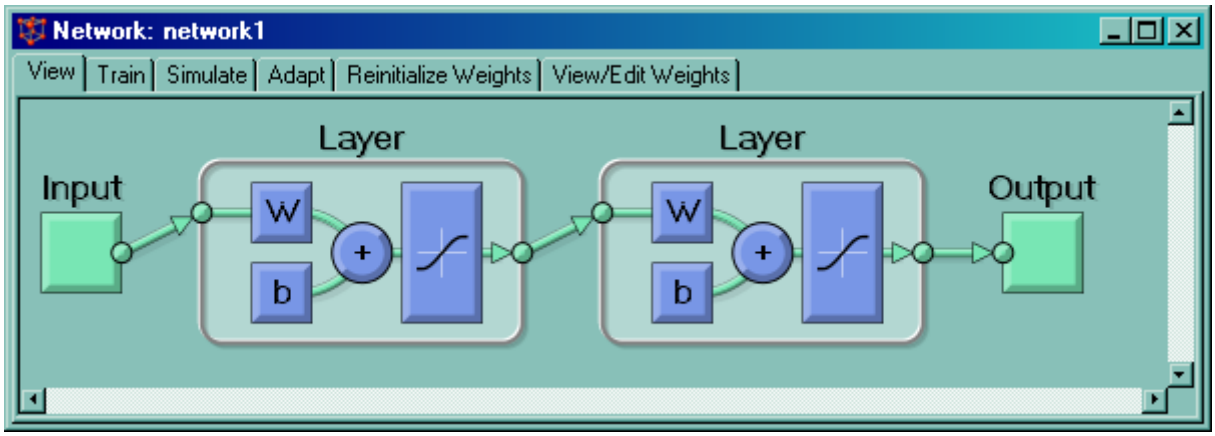


Рис.9.9.

5 шаг. Обучение нейросети.

Перейдем на закладку **Train** для обучения нейросети. В качестве обучающих данных укажем массивы P и T (рис.9.10):

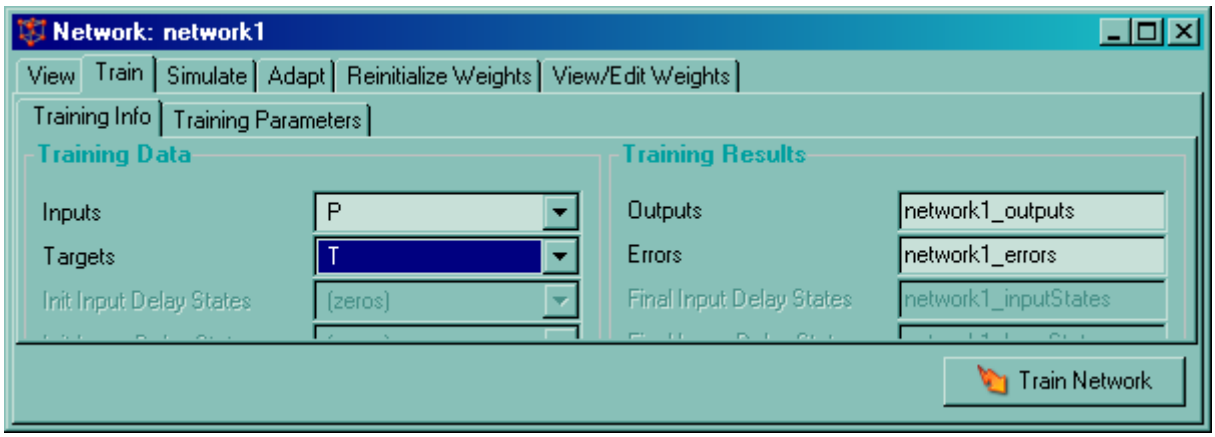


Рис.9.10.

На закладке Train Parameters можно редактировать параметры обучения. Увеличим параметр max_fail до 100 (рис.9.11).

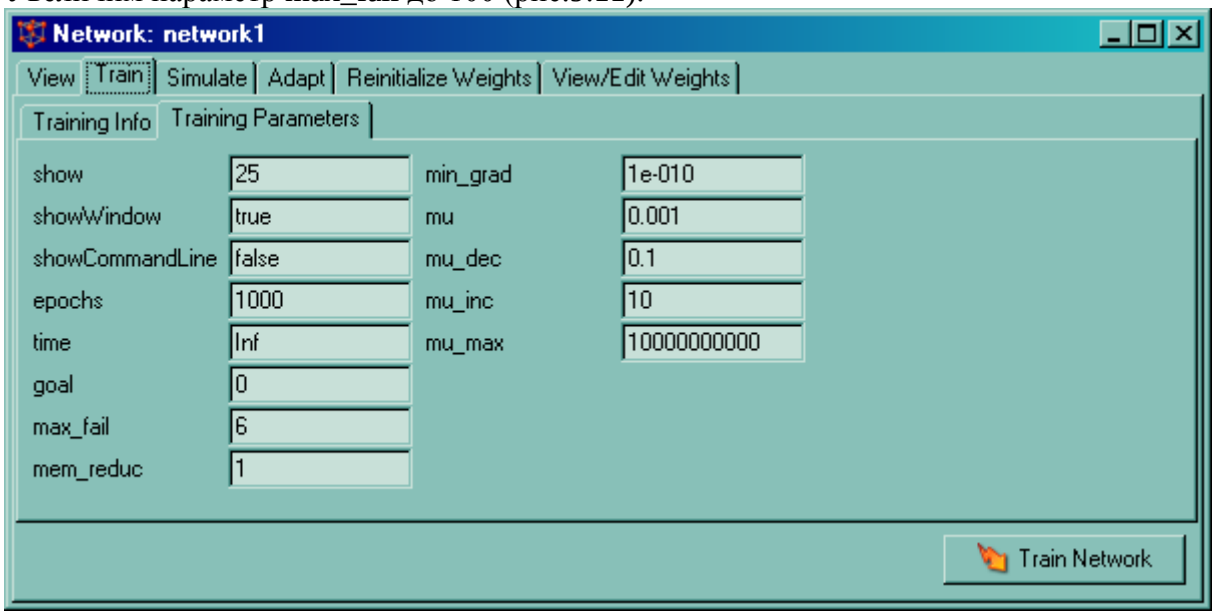


Рис.9.11.

6. При нажатии кнопки **Train Network** открывается окно обучения нейросети и происходит процесс настройки весов в соответствии с выбранным алгоритмом (рис.9.12):

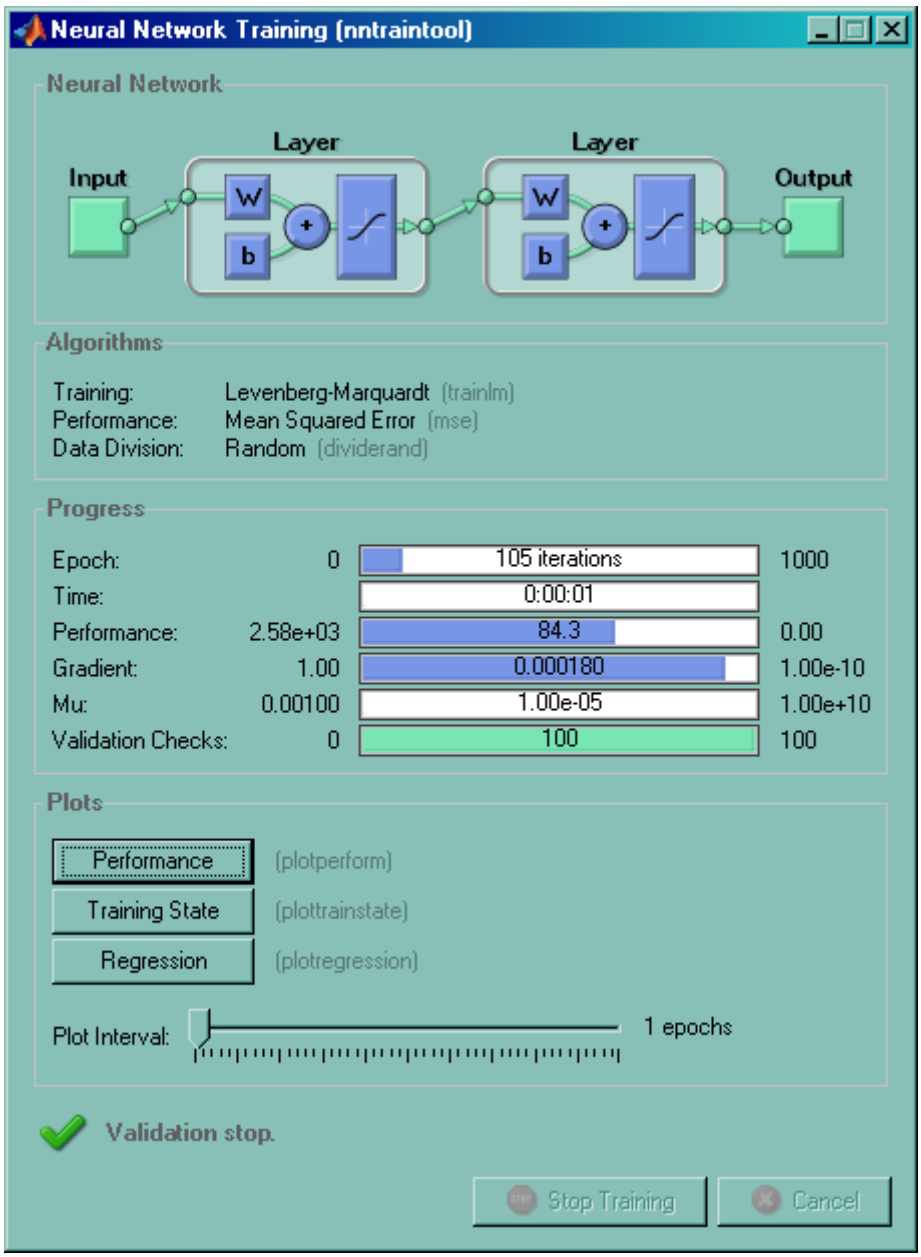


Рис.9.12.

При нажатии кнопки **Perfomance** можно посмотреть изменение функции качества обучения (рис.9.13):

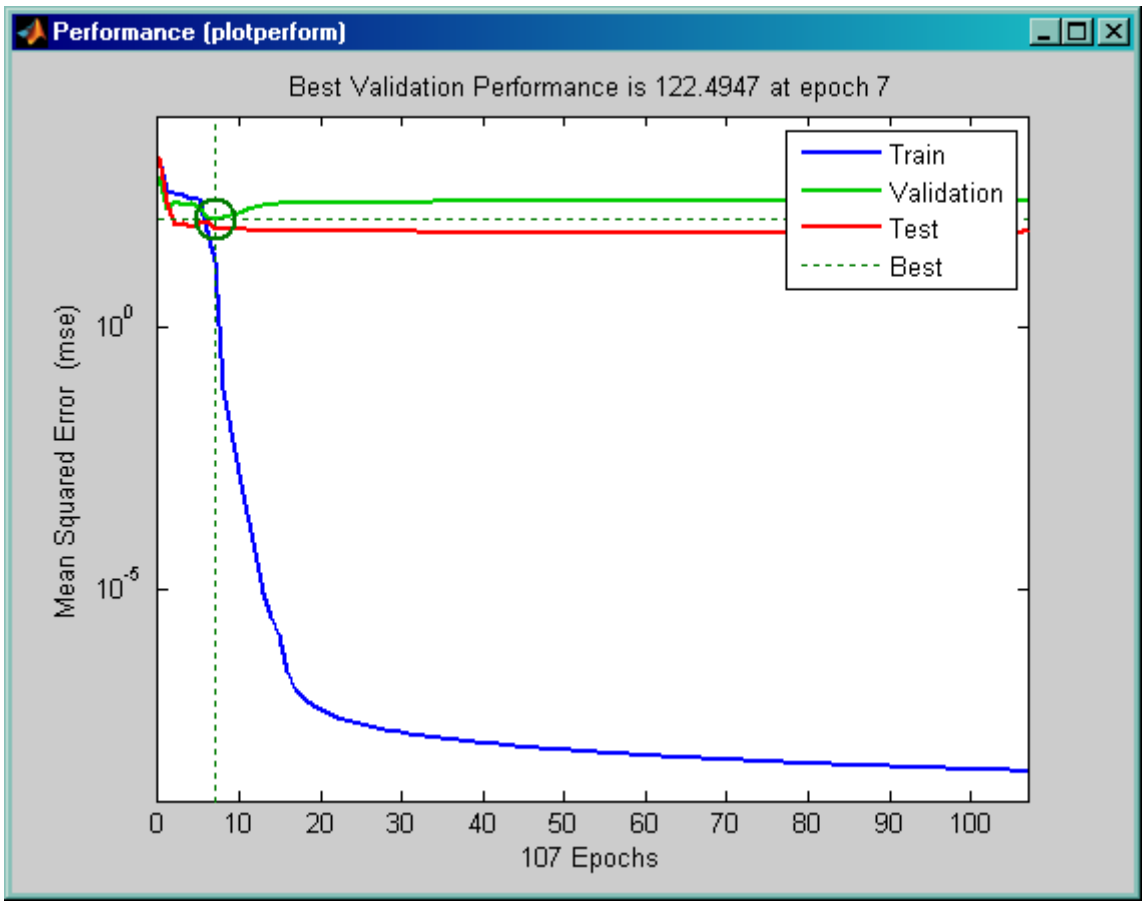


Рис.9.13.

7. После обучения проведем моделирование работы сети и получим выходы Y и Y1 сети для обучающей и проверочной выборок (рис.9.14):

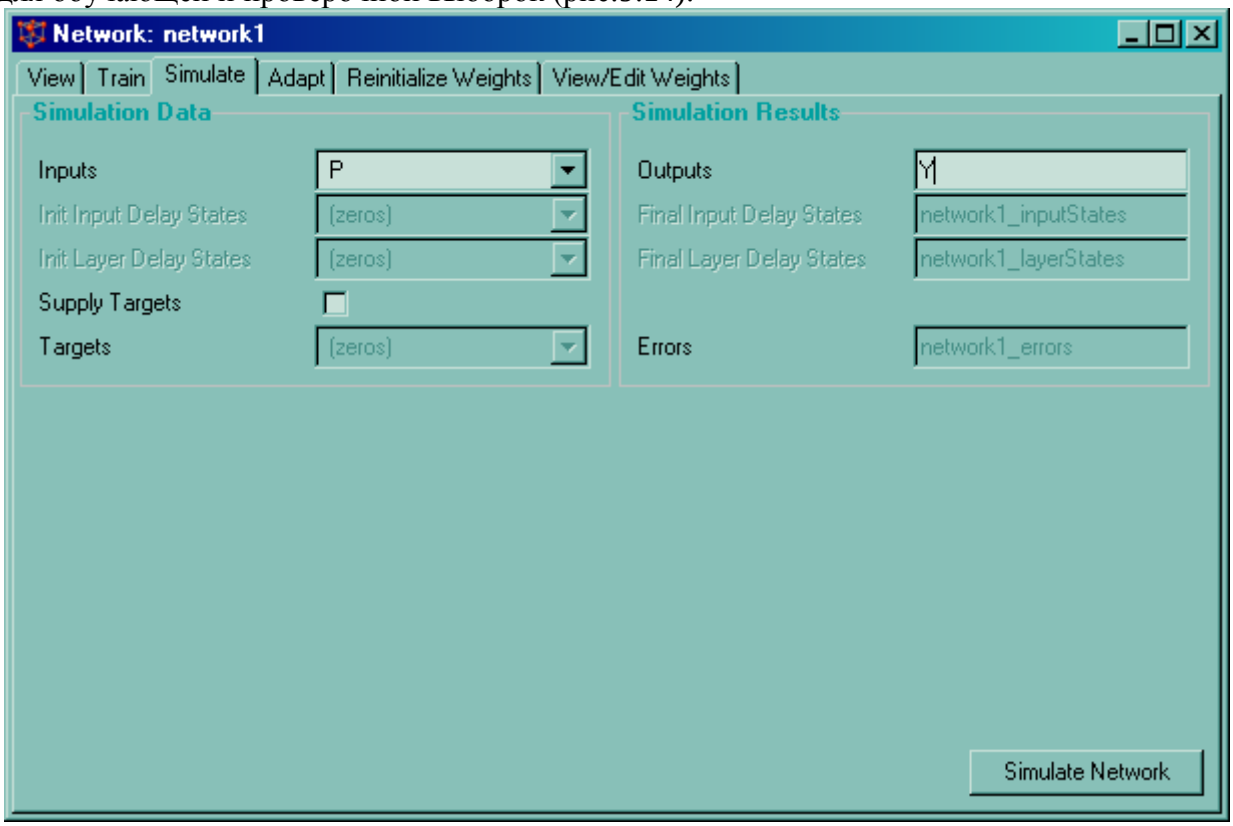


Рис.9.14.

Кнопка Simulate запускает моделирование сети.

После этого сформированные в результате моделирования выходы сети Y и Y1 окажутся в окне Output Data (рис.9.15).

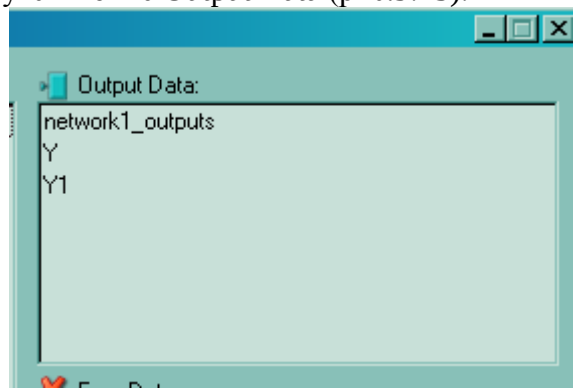


Рис.9.15.

8. Экспортируем данные Y и Y1 в Workspace матлаба нажатием кнопки **Export** (рис.9.16**):**

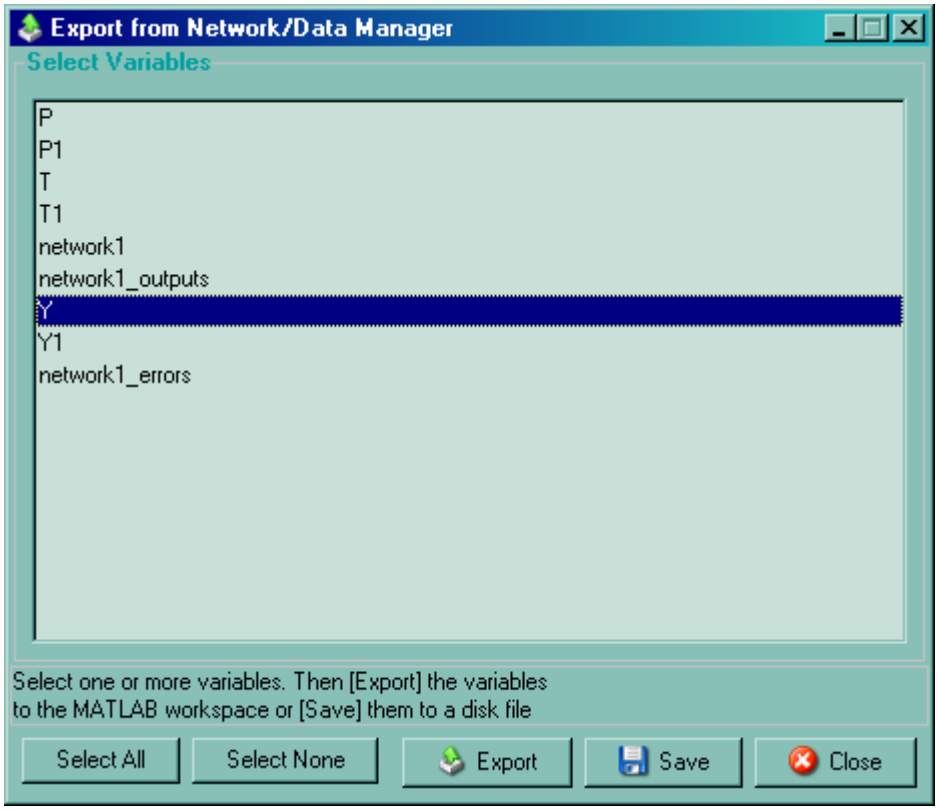


Рис.9.16.

9. С помощью команд

>> plot(P,T,P,Y,'o');
>> plot(P1,T1,P1,Y1,'o');

Можно построить графики работы нейросети с обучающими и проверочными данными (рис.9.17 а,б соответственно):

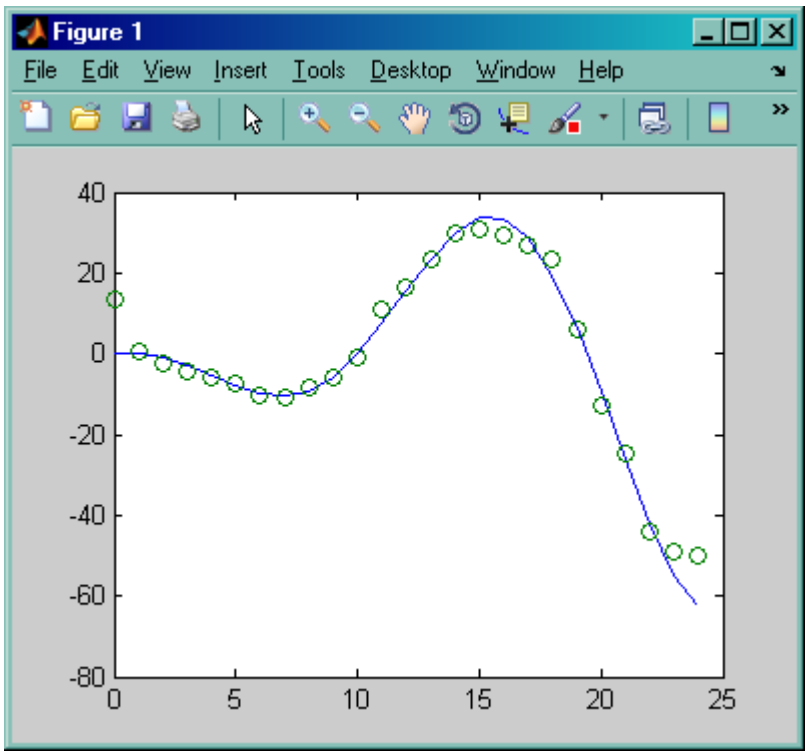


Рис.9.17.а

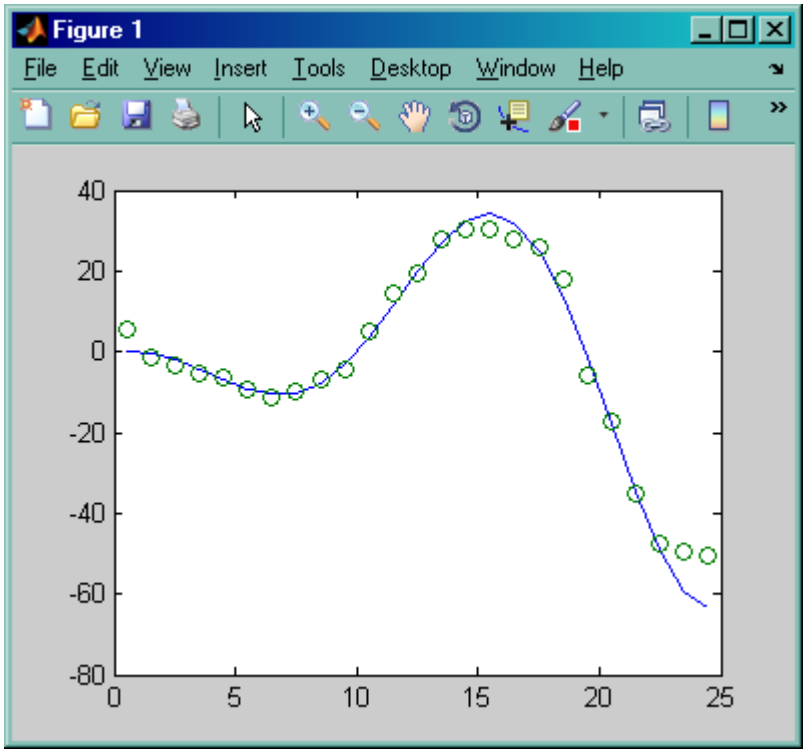


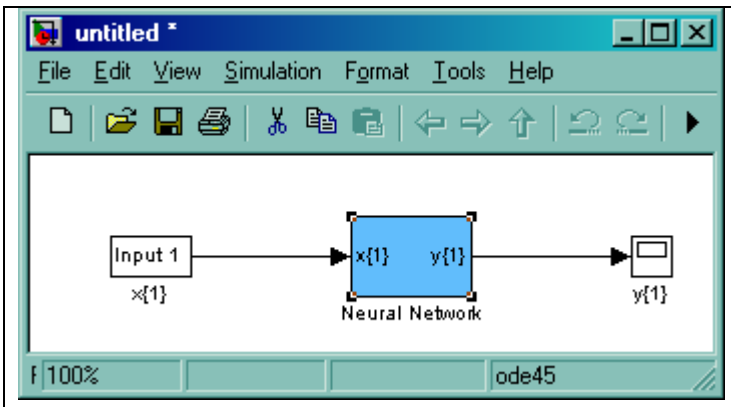
Рис.9.17.б

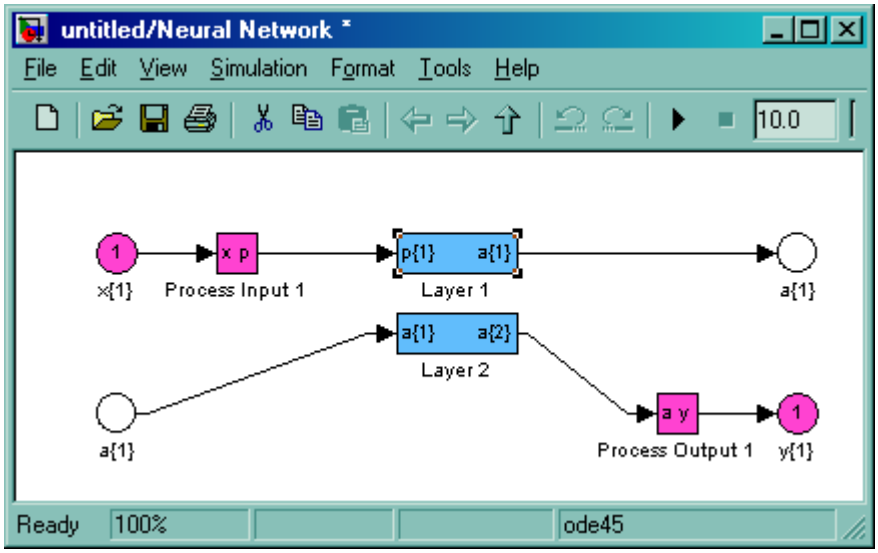
10. С помощью команды Export экспортируем в Workspase и саму обученную нейросет network1. Далее построим модель этой сети в SimuLink:

>> gensim(network1, -1)

(второй параметр здесь – sample time или такт квантования генерируемой модели сети. -1 – для непрерывной модели.)

Структура модели сети в **Simulink** (рис.9.18, 9.19):


untitled *
File Edit View Simulation Format Tools Help
Input 1
x{1}
x{1} y{1}
Neural Network
y{1}
100%
ode45


untitled/Neural Network *
File Edit View Simulation Format Tools Help
10.0
1
x{1}
x p
Process Input 1
p{1} a{1}
Layer 1
a{1}
a{1} a{2}
Layer 2
a{1}
a y
Process Output 1
1
y{1}
Ready
100%
ode45

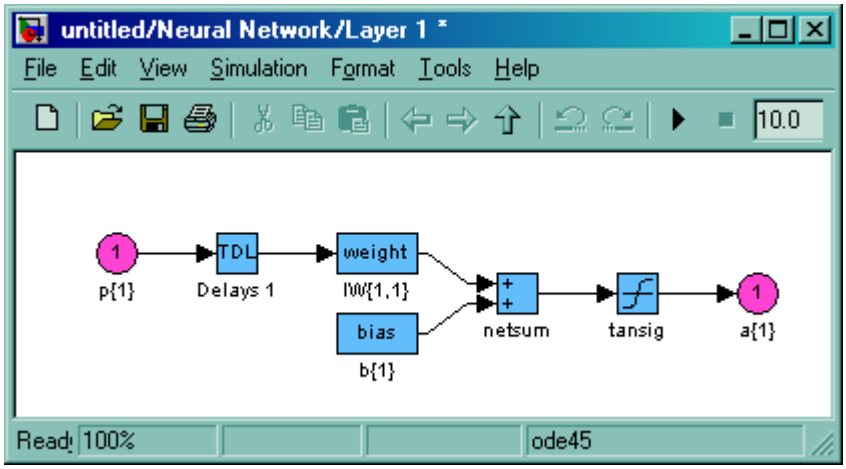

untitled/Neural Network/Layer 1 *
File Edit View Simulation Format Tools Help
10.0
1
p{1}
TDL
Delays 1
weight
IW{1,1}
bias
b{1}
+
+
netsum
tansig
1
a{1}
Read
100%
ode45

Рис.9.18.

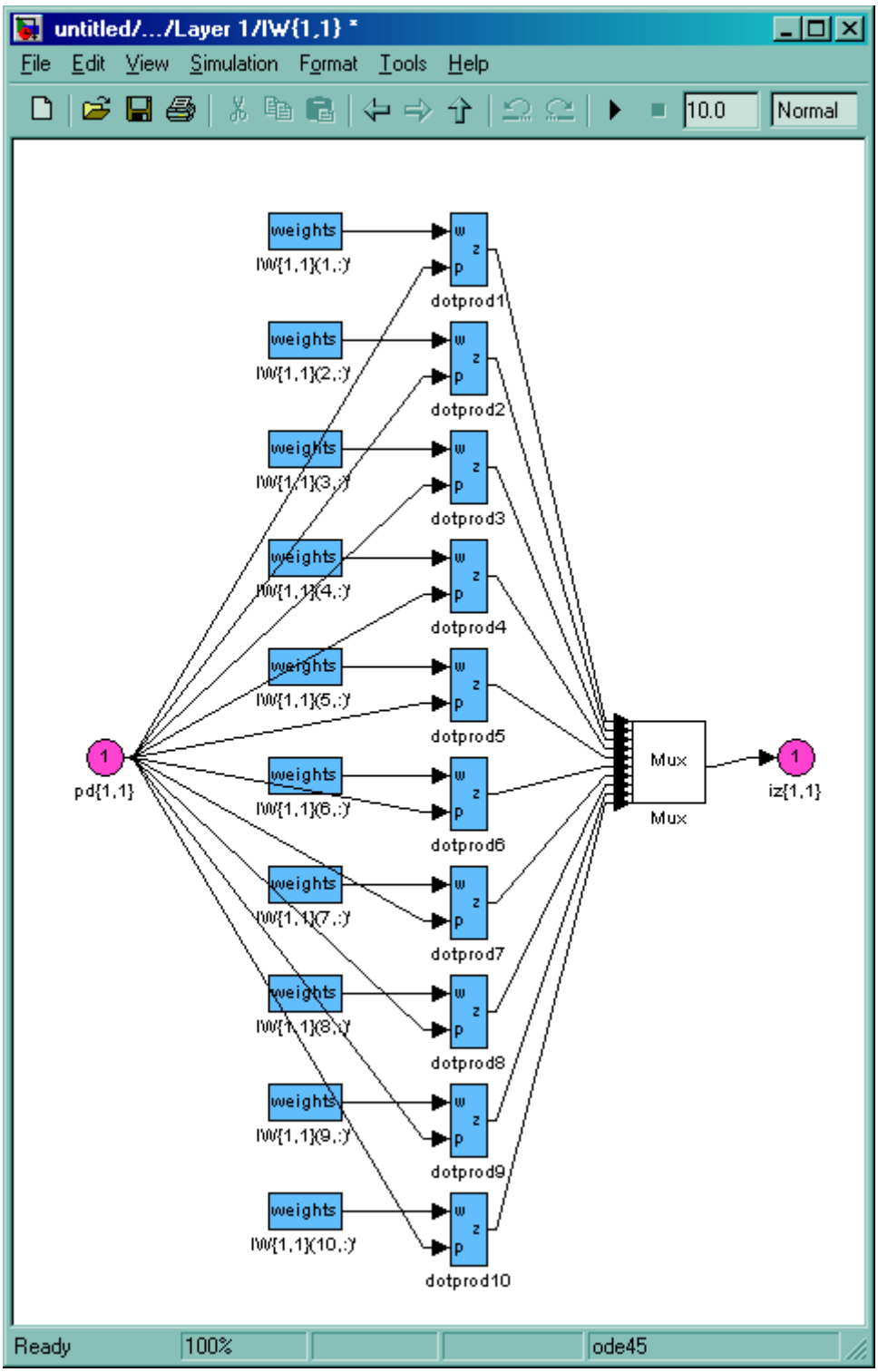

untitled/.../Layer 1/IW{1,1} *
File Edit View Simulation Format Tools Help
10.0
Normal
weights
IW{1,1}(1,:)
dotprod1
IW{1,1}(2,:)
dotprod2
IW{1,1}(3,:)
dotprod3
IW{1,1}(4,:)
dotprod4
IW{1,1}(5,:)
dotprod5
IW{1,1}(6,:)
dotprod6
IW{1,1}(7,:)
dotprod7
IW{1,1}(8,:)
dotprod8
IW{1,1}(9,:)
dotprod9
IW{1,1}(10,:)
dotprod10
pd{1,1}
Mux
iz{1,1}
Ready
100%
ode45

Рис.9.20.

11. Проверим работу сети по аппроксимации исходной функции в Simulink (рис.9.20):

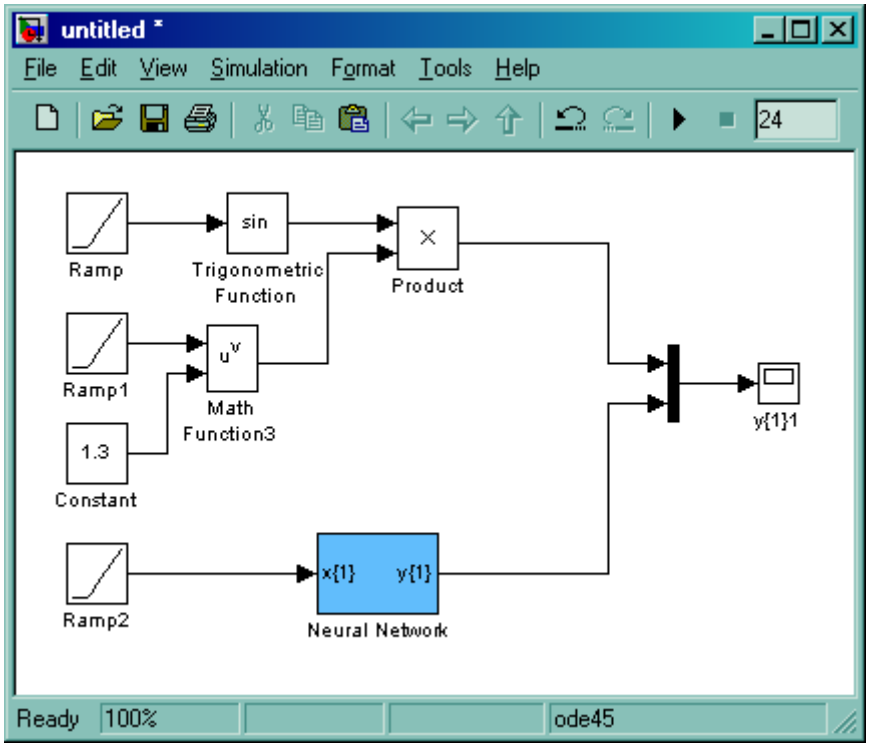


Рис.9.20.

Графики исходной функции и аппроксимирующей ее нейросети (рис.9.21):

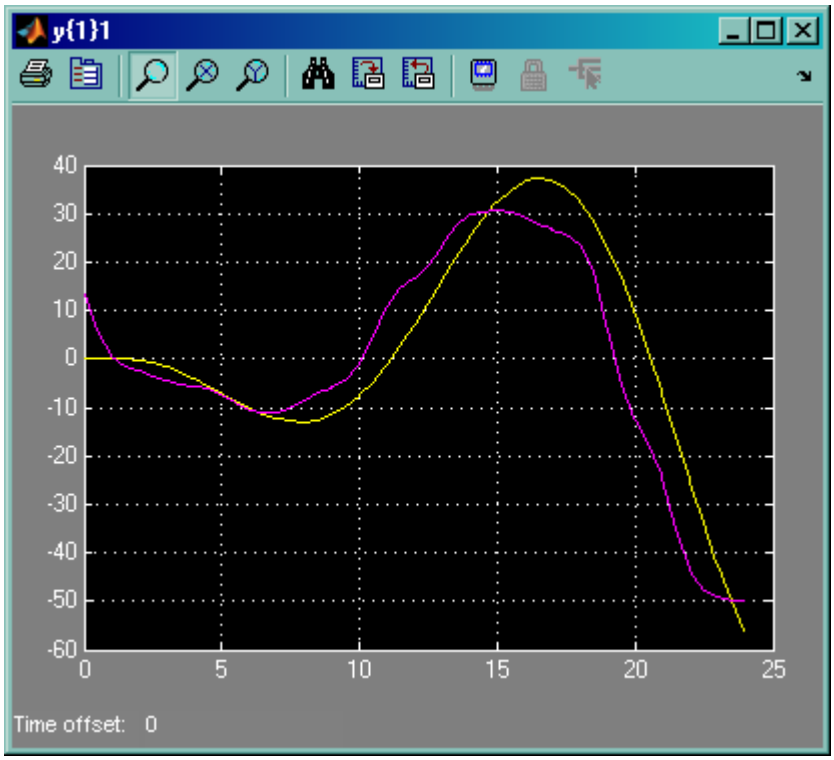


Рис.9.21.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Реализовать в среде MatLab процесс формирования и обучения нейросети по представленному выше алгоритму.
2. Провести процедуру обучения нейросети одной из функций, заданных преподавателем.
3. Рассмотреть п. для различных методов обучения.
4. Составить отчет. В отчете необходимо сделать выводы относительно полученных результатов.

1. Варианты аппроксимируемых функций:

1. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.4x 1.3 ; f(x) = 5, если x<4 и 8, если x>=4

2. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 2x 1.5 ; f(x) = 5, если x<3 и 10, если x>=3

3. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 3.2x 1.25 ; f(x) = 4.5, если x<3 и 8, если x>=3

4. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.72x 1.6 ; f(x) = 50, если x<7 и 78, если x>=7

5. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.5x 1.4 ; f(x) = 5.2, если x<3.45 и 9, если x>=3.45

6. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.6x 1.32 ; f(x) = 3, если x<3 и 8, если x>=3

7. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.22x 1.43 ; f(x) = 6, если x<4.2 и 2, если x>=4.2

8. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.8x 1.11 ; f(x) = 5, если x<4.1 и 3, если x>=4.1

9. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.23x 1.37 ; f(x) = 7.5, если x<9 и 1, если x>=9

10. Выполнить задание для двух функций (сначала для одной, а потом для другой): f(x) = 1.45x 1.35 ; f(x) = 0.5, если x<42 и 80, если x>=42

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание порядка выполнения работы и результаты обучения нейросети.

Лабораторная работа №10

СИНТЕЗ НЕЙРОННОЙ СЕТИ ДЛЯ РЕШЕНИЯ СИСТЕМ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

1. ЦЕЛЬ РАБОТЫ

С использованием нейросетевых технологий синтезировать алгоритм интегрирования и сформировать модель в среде MatLab для решения систем дифференциальных уравнений

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Рассмотрим задачу решения систем обыкновенных дифференциальных уравнений (ОДУ) в нейросетевом базисе. Нейросетевой базис представляет собой операцию нелинейного преобразования взвешенной суммы, т.е. операцию взвешенного суммирования нескольких входных сигналов с последующим преобразованием этой суммы посредством функции активации формального нейрона, которая может быть как линейной, так и нелинейной.

Пусть задана система n дифференциальных уравнений 1-го порядка

$$\dot{Y}(x) = A\,Y(x), \qquad (10.1)$$

где A - квадратная матрица постоянных коэффициентов размера n×n; Y - n - мерный вектор искомой функции аргумента x.

Для метода Рунге-Кутты 1-го порядка (метод прямоугольников) решение системы (10.1) можно представить в виде:

$$Y_{t+1} = Y_t + hAY_t = (E + hA)Y_t = B_t\,Y_t, \qquad (10.2)$$

где E - единичная матрица размером n×n.

Элементы матрицы B есть первые два члена разложения матричной экспоненты в степенной ряд.

Для системы двух уравнений выражение (10.2) принимает вид:

$$y_{1t+1} = b_{11}y_{1t} + b_{12}y_{2t} = (1 + ha_{11})y_{1t} + ha_{12}y_{2t},$$

$$y_{2t+1} = b_{21}y_{1t} + b_{22}y_{2t} = ha_{21}y_{1t} + (1 + ha_{22})y_{2t}.$$

Схема соединения нейронов, реализующая решение системы показана на рис.10.1.

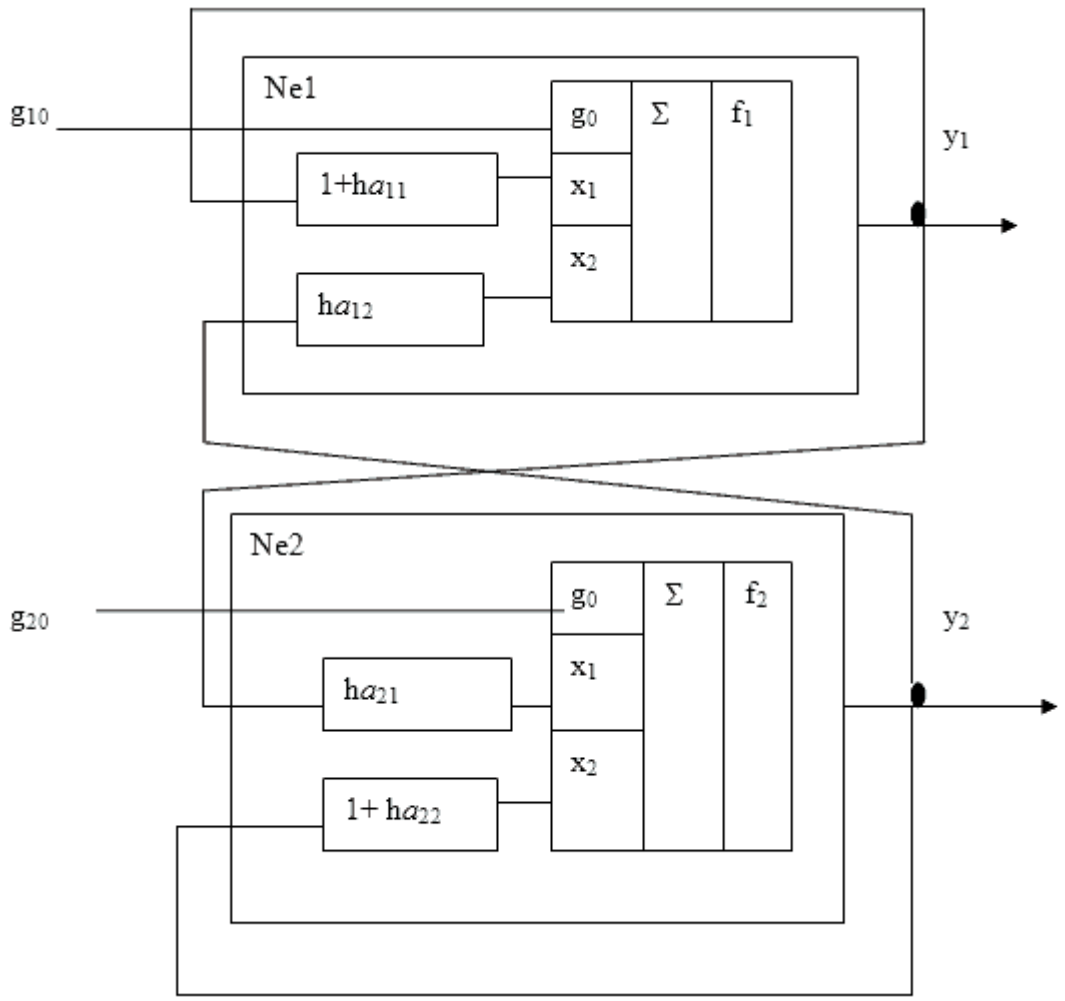


Рис. 10.1. Схема соединения нейронов, реализующая решение системы ОДУ в нейросетевом базисе методом Рунге-Кутты 1-го порядка.

Здесь Ne1 и Ne2 - нейроны, участвующие в операции интегрирования. Выходом сети являются сигналы y_1 и y_2. Входные сигналы g_{10} и g_{20} вводят в нейроны начальное возбуждение, эквивалентное начальным условиям решения системы уравнений. Функция активации у обоих нейронов - симметричная линейная функция.

3. Пример решения задачи разработки ИНС для решения ОДУ в системе Simulink

Разработаем ИНС для решения системы уравнений (10.3)

$$y'1 = y2, \qquad (10.3)$$
$$y'2 = -y1$$

при начальных условиях: y1(0) = -1, y2(0) = 0, t = 0…2π.

Блок-схема решения системы (10.3) в нейросетевом базисе, созданная в среде Simulink, показана на рис.10.2.

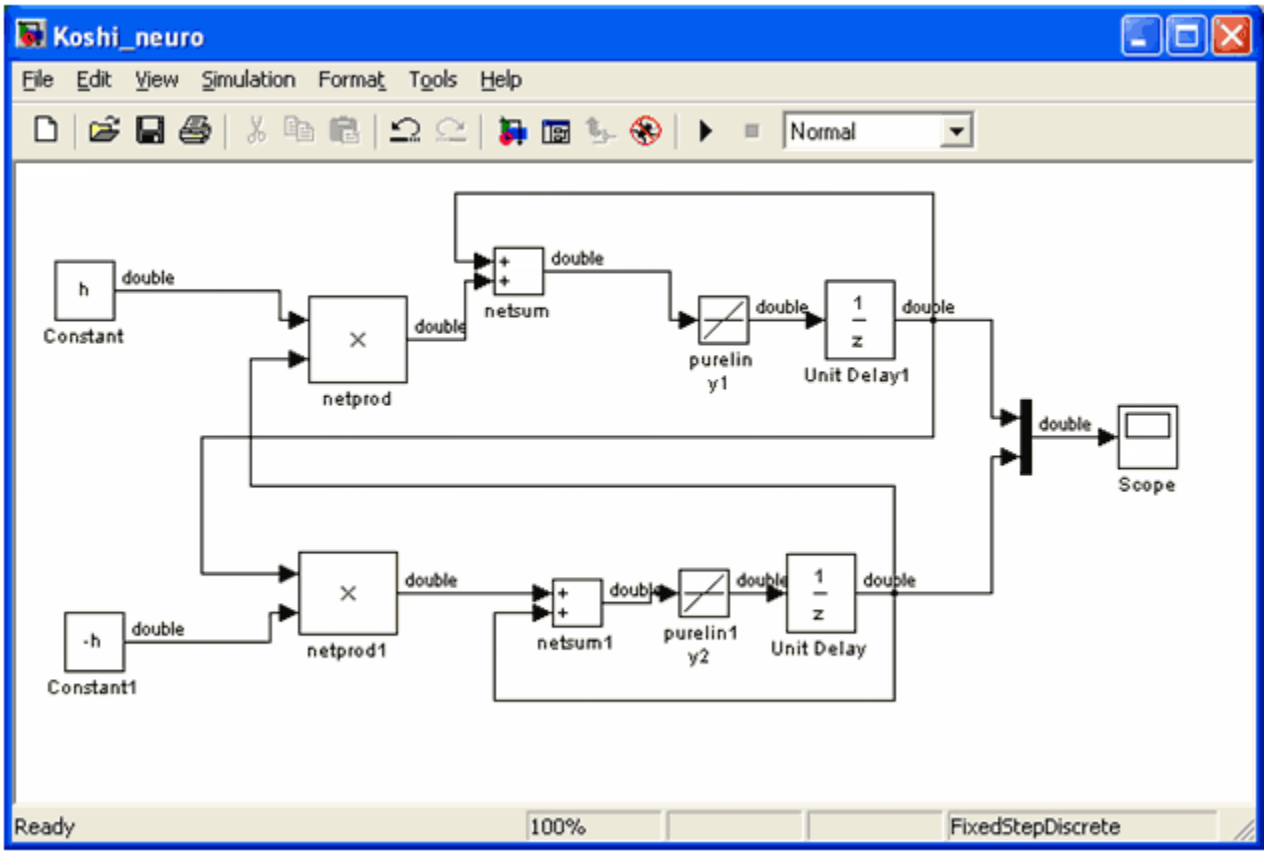


Рис. 10.2. Блок-схема решения системы (10.3) в нейросетевом базисе.

При построении схемы используются блоки из меню Neural Network Blockset: netprod, netsum (Net Input Functions), purelin (Transfer Functions). Для визуализации результатов используется блок Scope (Sinks), для задания шага интегрирования - блок Constant (Sources), для установки начальных значений - блок Unit Delay (Discrete). Окно задания параметров блока Unit Delay показано на рис10.3. Для показа двух графиков в блоке Scope использован блок Mux (Signals&Systems).

Окно браузера библиотеки Simulink показано на рис.10.4.

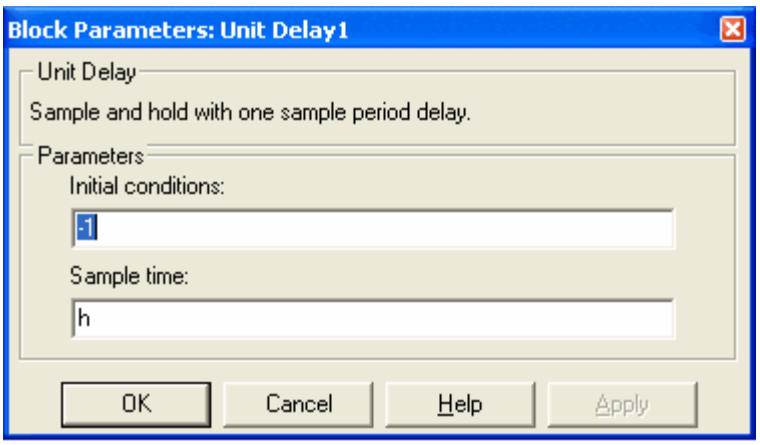


Рис. 10.3. Окно задания параметров блока Unit Delay.

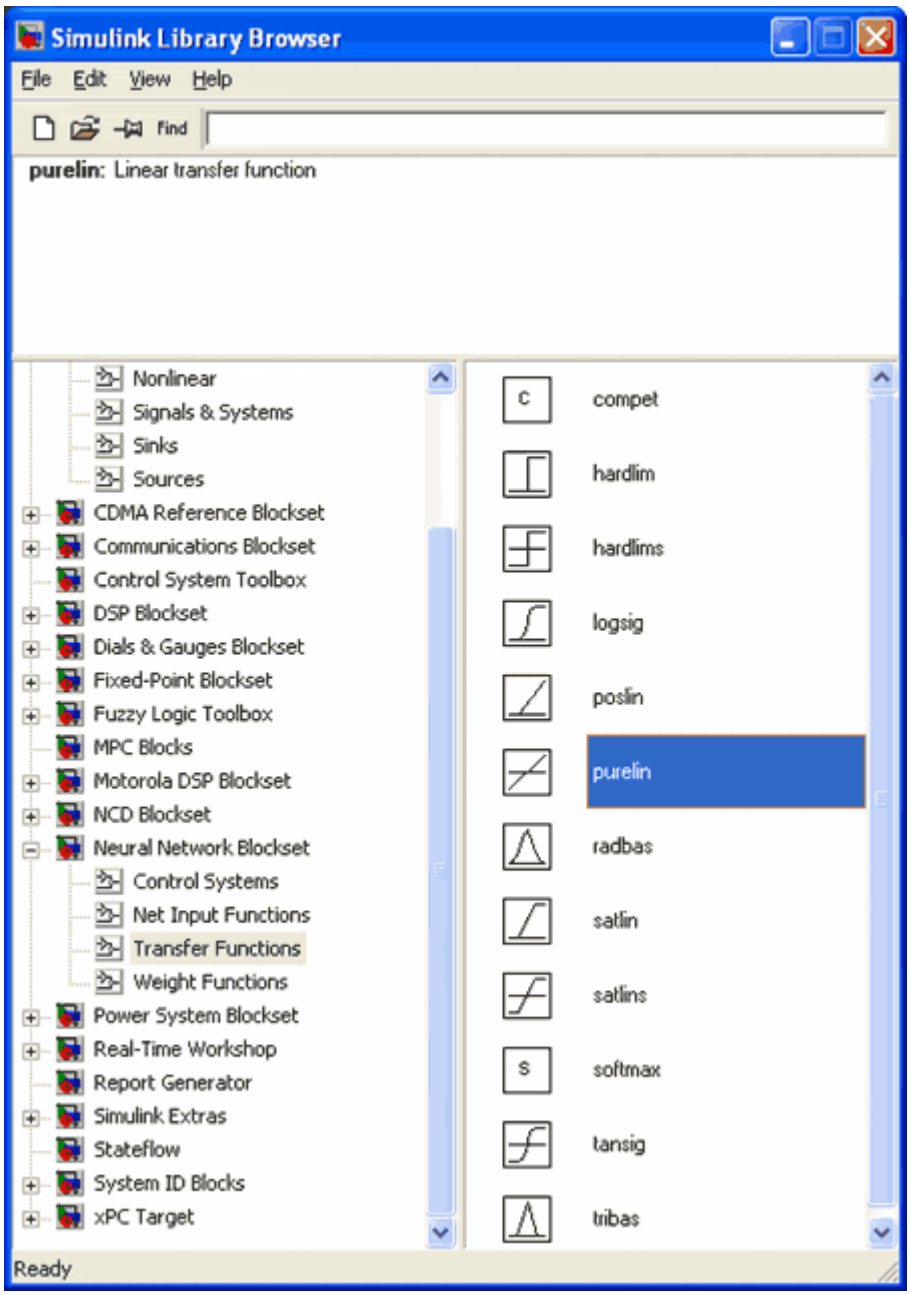


Рис. 10.4. Окно библиотеки блоков системы Simulink.

Для запуска схемы из командной строки MatLab следует набрать:

```
>> N=100;
>> h=2*pi/N;
>> sim('Koshi_neuro')
```

После открытия блока Scope можно увидеть графики решения задачи (рис.10.5).

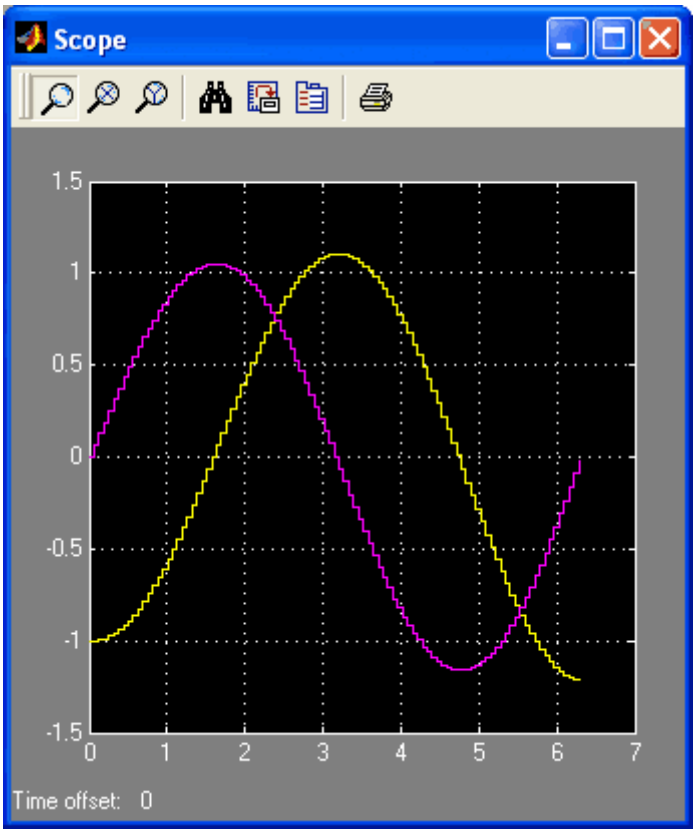


Рис. 10.5. Графики решения задачи в Simulink.

Сравним полученные графики с аналогичными графиками решения задачи в MatLab. Для этого создадим М-файл (рис.10.6), задающий систему уравнений (3):

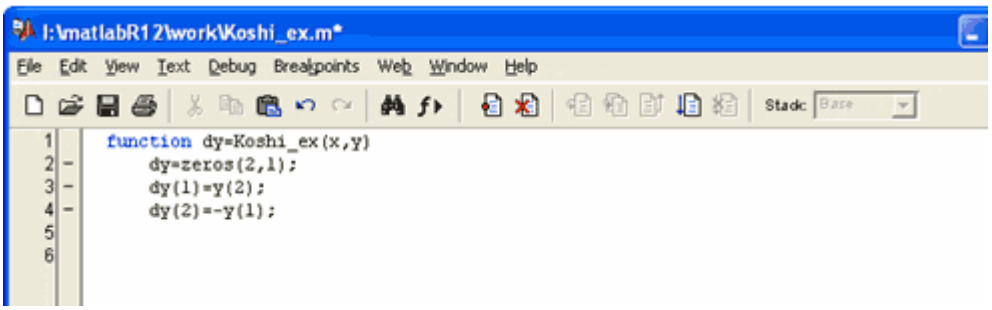


Рис.10.6.

После сохранения М-файла с именем Koshi_ex используем для решения рассматриваемой системы ОДУ функцию ode45, набрав в командном окне MatLab:

```
>> [T,Y] = ode45('Koshi_ex',[0 2*pi],[-1 0]);
>> plot(T,Y(:,1),'-',T,Y(:,2),'-.')
```

Получаем графики решения, идентичные графикам на рис.10.7 (рис.10.7).

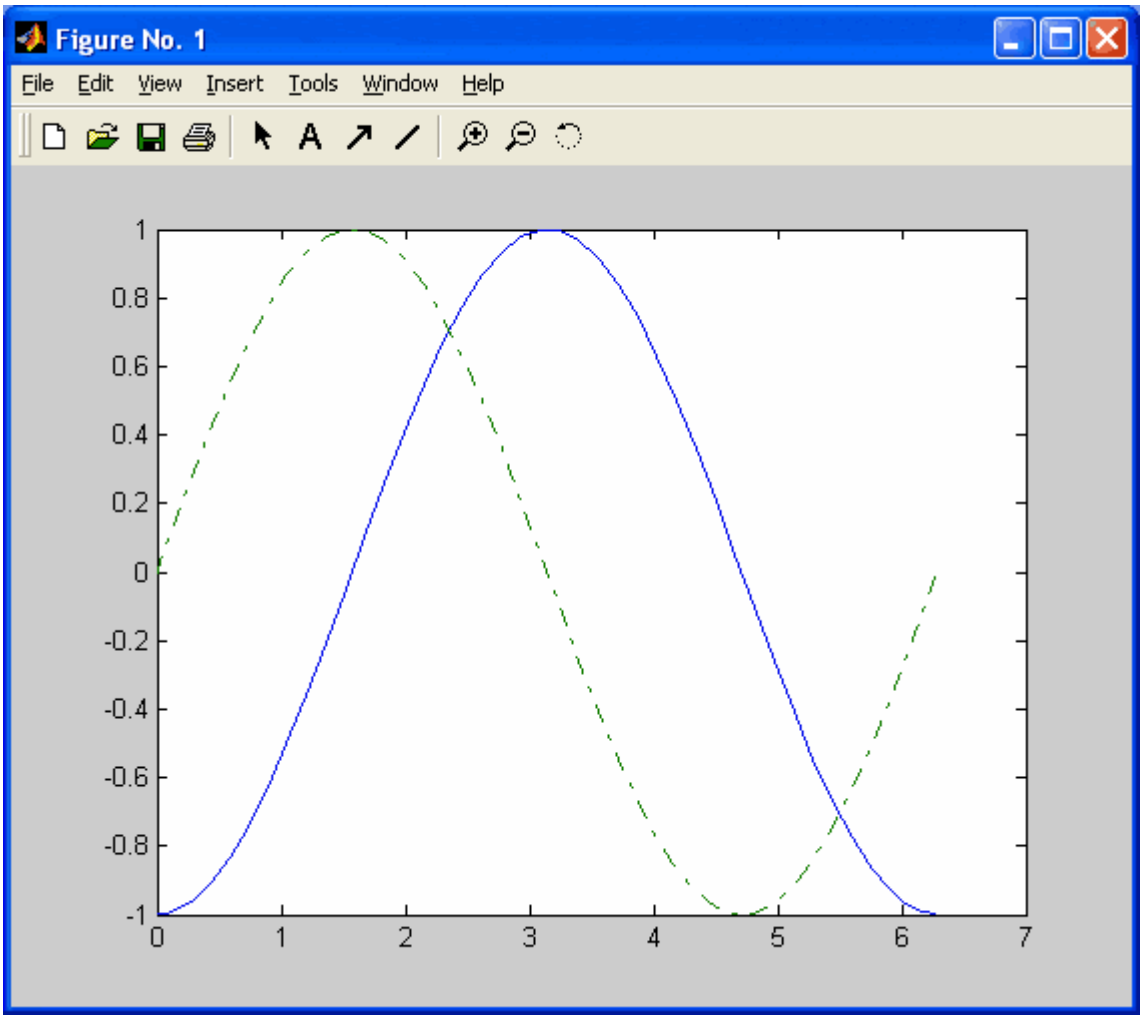


Рис. 10.7. Графики решения задачи с помощью функции ode45.

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Разработать в среде Simulink ИНС для решения системы ОДУ:

$$\frac{dy_1}{dt} = -y_2 + y_3,$$
$$\frac{dy_2}{dt} = y_1 - y_3,$$
$$\frac{dy_{31}}{dt} = -y_1 + y_2$$

при начальных условиях: $y_1(0) = 0$, $y_2(0) = 0$, $y_3(0) = 1$, $t = 0\ldots15$.

5. ОБОРУДОВАНИЕ

Лабораторная работа выполняется на персональных компьютерах с применением пакетов моделирования SimuLink, NNT в среде MatLab

6.ОТЧЕТ

Отчет должен содержать краткое описание решения поставленных задач.

7. СПИСОК ЛИТЕРАТУРЫ

1.

Лабораторная работа № 11

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ПРОЕКТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ПРОЦЕССАМИ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Познакомиться с основными архитектурами нейронных сетей,которые реализованы в пакете прикладных программ (ППП)NeuralNetworkToolbox в виде специализированных контроллеров. Освоить основные приемы работы с архитектурой NARMA-L2 Controller, реализованной в среде компьютерной математики MATLAB.

Учебно-методическое обеспечение – методическая разработка, класс ПЭВМ.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В практике проектирования систем управления динамическими процессами все большее применение находят нейронные сети. Это связано с одной стороны со все возрастающими требованиями к характеристикам систем управления и усложняющимися объектами и процессами, которыми необходимо управлять, а с другой с интенсивным развитием специализированной вычислительной техники, например, нейропроцессоров. Универсальные возможности аппроксимации с помощью многослойного персептрона делают их полезным инструментом для решения задач идентификации, проектирования и моделирования нелинейных регуляторов.

В настоящем методическом пособии кратко представлены три архитектуры нейронных сетей, которые реализованы в пакете прикладных программ (ППП)NeuralNetworkToolbox в виде следующих контроллеров:

- контроллера с предсказанием (NN PredictiveController);
- контроллера на основе модели авторегрессии со скользящим средним (NARMA-L2 Controller);
- контроллера на основе эталонной модели (ModelReferenceController).

Применение нейронных сетей для решения задач управления позволяетпредполагает два этапа проектирования:

- этап идентификации управляемого процесса;
- этап синтеза закона управления.

На первом этапе (этапе идентификации) формируется модель управляемого процесса в виде нейронной сети, которая на втором этапе (этапе синтеза) используется для

синтеза регулятора. Для каждой из трех архитектур используется одна и та же процедура идентификации, однако этапы синтеза существенно различаются.

При управлении с предсказанием модель управляемого процесса используется для того чтобы предсказать его будущее поведение, а алгоритм оптимизации применяется для расчета такого управления, которое минимизирует разность между желаемыми и действительными изменениями выхода модели.

При управлении на основе модели авторегрессии со скользящим средним регулятор представляет собой достаточно простую реконструкцию модели управляемого процесса.

При управлении на основе эталонной модели регулятор – это нейронная сеть, которая обучена управлять процессом так, чтобы он отслеживал поведение эталонного процесса. При этом модель управляемого процесса активно используется при настройке параметров самого регулятора.

В последующих разделах обсуждаются все три структуры систем управления и архитектуры соответствующих нейросетевых контроллеров. Каждый раздел включает краткое изложение принципа управления динамическим процессом и сопровождается описанием сценария функционирования проектируемой системы, который реализован в виде комбинации GUI-интерфейса и динамической модели регулятора в системе Simulik.

Динамические модели систем управления с нейросетевыми регуляторами размещены в разделе **ControlSystems**библиотеки блоковдля моделирования нейронных сетей **NeuralNetworkBlockset**, доступ к которым обеспечивается браузером **LibraryBrowser**пакета**Simulink**или командой **neural**(рисунок 11.1) .

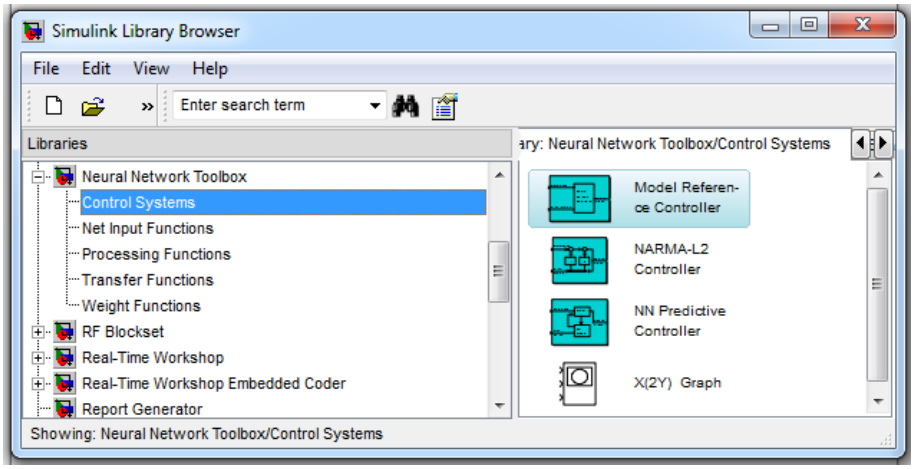


Рис.11.1.

Поскольку ни один конкретный регулятор не является универсальным, то описаны функциональные возможности всех трех типов регуляторов, каждый из которых имеет свои преимущества и недостатки.

Регулятор с предсказанием - NeuralNetPredictiveController– регулятор с предсказанием будущих реакций процесса на случайные сигналы управления. Этот регулятор использует модель управляемого процесса в виде нейронной сети. Алгоритм оптимизации вычисляет управляющие сигналы, которые минимизируют разность между желаемыми и действительными изменениями сигнала на выходе модели и таким образом оптимизируютуправляемый процесс. Построение модели управляемого процессавыполняется автономно с использованием нейронной сети, котораяобучается в групповом режиме с использованием одного из алгоритмов обучения. Реализация такого регулятора требует значительногообъема вычислений, поскольку расчеты по оптимизации выполняются на каждом такте управления.

Регулятор NARMA-L2 (NonlinearAutoregressive – MovingAverage) – регулятор на основе модели авторегрессии со скользящим средним. Данный регулятор представляет собой модифицированную нейросетевую модель управляемого процесса, полученную на этапе автономной идентификации.Изо всех архитектур этот регулятор требует наименьшего объем вычислений. Данный регулятор – это просто некоторая реконструкция нейросетевой модели управляемого процесса, полученной на этапе автономной идентификации. Вычисления в реальном времени связаны только с реализацией нейронной сети. Недостаток метода состоит в том, что модель процесса должна быть задана в канонической форме пространства состояния, которой соответствует сопровождающая матрица, что может приводить к вычислительным погрешностям.

Регулятор на основе эталонной модели - ModelReferenceController– регулятор на основе эталонной модели. Требуемый объем вычислений для этого регулятора сравним с предыдущим. Однако архитектура регулятора с эталонной моделью требует обучения нейронной сети управляемого процесса и нейронной сети регулятора. При этом обучение регулятора оказывается достаточно сложным, поскольку обучение основано на динамическом варианте метода обратного распространения ошибки, так как нейронная сеть использует линии задержки. Достоинством регуляторов на основе эталонной модели является то, что они применимы к различным классам управляемых процессов.

Для каждой из трёх архитектур регуляторов используется одна ита же процедура идентификации управляемого процесса. Нейроннаямодель во всех случаях представляет собой двухслойную сеть с прямой передачей сигнала и с линиями задержки на каждом слое. Входной, или скрытый (**hiden**) слой может иметь произвольное числонейронов. Выходной слой имеет только один нейрон. Для входногослоя функции взвешивания, накопления и активизации являются соответственно **dotprod**, **netsum**и **logsig**. Выходной слой имеет такиеже функции взвешивания и накопления, а функцией активизации для

него является линейная функция **purelin**. Известно, что сети с такойархитектурой могут воспроизводить весьма сложные нелинейныезависимости между входом и выходом сети.

Настройка параметров нейронной сети, являющейся модельюобъекта, выполняется автономно методом последовательного обучения с использованием данных, полученных при испытаниях реального объекта. Для обучения сети может быть использован любой из обучающих алгоритмов для нейронных сетей. Использование контрольного и тестового множеств обучающих данных позволяет избежать явления переобучения сети. Изменяя число нейронов в первом слое, количество линий задержки на входе и выходе сети, а также интервал квантования, или дискретности, можно обеспечить требуемую точность моделирования управляемого процесса.

Диалоговая панель для идентификации управляемого процесса**PlantIdentification**входит в состав всех трёх регуляторов раздела**ControlSystems**библиотеки нейронных блоков системы **Simulink**,является универсальным средством и может быть использована дляпостроения нейросетевых моделей любых динамических объектов,которые могут быть представлены блоками этой системы.

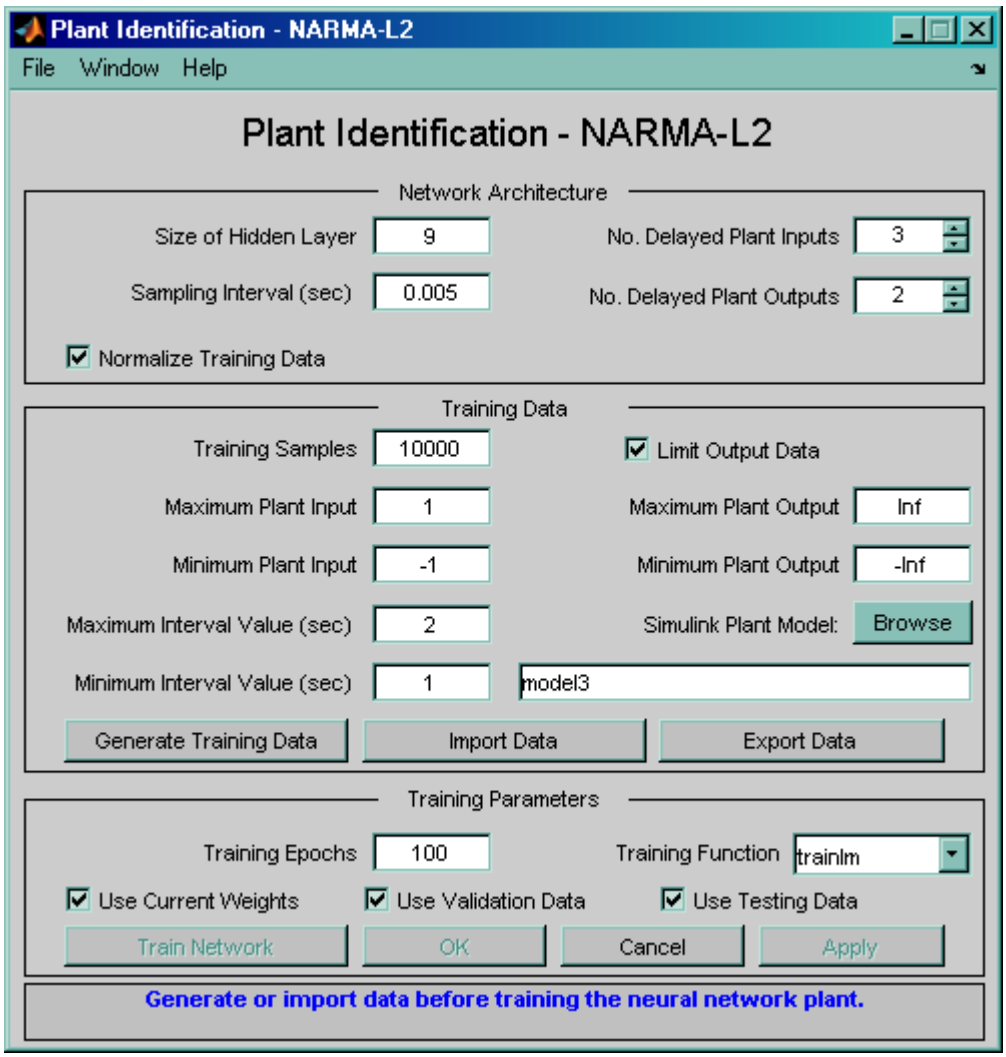


Рис. 11.2

С помощью управляющих элементов панели **PlantIdentification**можно задать архитектуру нейронной сети, параметры обучающейпоследовательности и параметры обучения, а также управлять процессом идентификации и оценивать качество этого процесса.

Набор управляющих элементов для задания архитектурных параметров нейронной сети следующий:

1. **SizeoftheHidenLayer**– количество нейронов на входном илискрытом слое;

2. **No. DelayedPlantInputs**– число линий задержки для входногослоя;

3. **No. DelayedPlantOutputs**– число линий задержки для выходного слоя;

4. **SamlingInterval**– интервал квантования или шаг дискретности, в секундах, между двумя последовательными моментами отсчёта данных;

5. **NotmalizeTrainingData**– переключатель нормирования дляпреобразования обучающих данных к диапазону **[0 1]**.

Набор управляющих элементов для задания характеристик обучающей последовательности таков:

1. **TrainingSamples**– число точек отсчёта для получения обучающей последовательности в виде пар значений вход-выход дляуправляемого процесса, определяемого моделью **Simulink**. В качестве обучающей последовательности генерируется случайный прямоугольный сигнал;

2. **MaximumPlantInput**– максимальное значение случайного входного сигнала;

3. **MinimumPlantInput**– минимальное значение случайного входного сигнала;

4. **MaximumIntervalValue (sec)** – максимальный интервал периода в секундах случайного сигнала;

5. **MinimumIntervalValue (sec)** – минимальный интервал периода в секундах случайного сигнала;

6. **LimitOutputData**– переключатель для ограничения значений выходного сигнала;

7. **MaximumPlantOutput**– максимальное значение выходногосигнала, задаваемое при включённом переключателе **LimitOutputData**;

8. **MinimumPlantOutput**– минимальное значение выходногосигнала, задаваемое при включённом переключателе **LimitOutputData**;

9. **SimulinkPlantModel**– для задания модели управляемого процесса, реализованной с помощью блоков **Simulink**, имеющий портывхода и выхода и сохранённой в файле ***.mdl**; выбор модели производится с помощью кнопки **Browse**;

Параметры обучения задаются следующим образом:

1. **TrainingEpochs**– количество циклов обучения;

2. **TrainingFunction**– задание обучающей функции;

3. **UseCurrentWeights**– переключатель для использования текущих весов нейронной сети;

4. **UseValidationData**– переключатель для использования контрольного множества в объёме **25 %** от обучающего множества;

5. **UseTestingData**– переключатель для использования тестового множества в объёме **25%** от обучающего множества.

РЕГУЛЯТОР NARMA-L2

Нейросетевой регулятор, описанный в этом разделе, использует в качестве модели управляемого процесса модель нелинейной авторегрессии со скользящим средним (NonlinearAutoregressive-MovingAverage - NARMA-L2)

Рассмотрим пример обучения нейросетевого регулятора на основе модели авторегрессии со скользящимсреднийм: **Narmal2(NonlinearAutoregressive – MovingAverage).** В качестве объекта управления выберем модель двигателя постоянного тока:

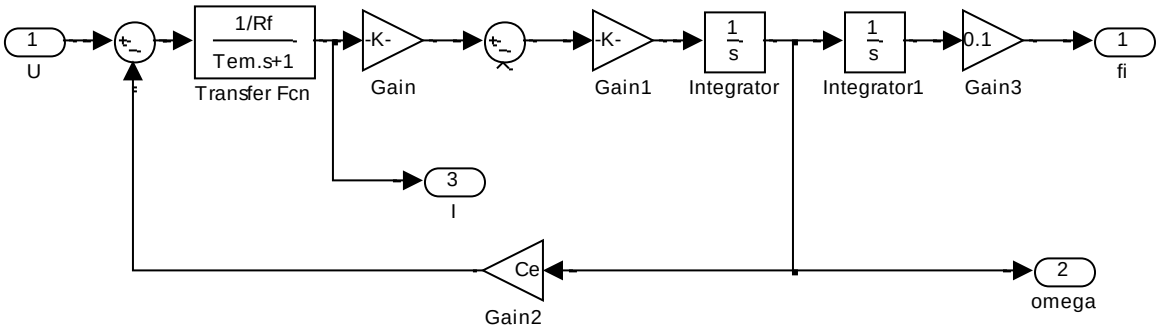


Рис. 11.3.

Модель системы управления с нейроконтроллером выглядит следующим образом:

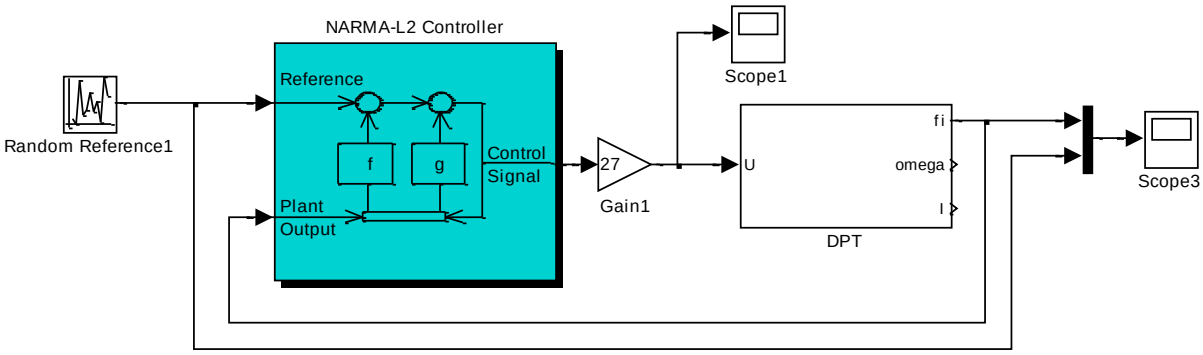


Рис. 11.4.

Для идентификации управляемого процесса необходимо в диалоговой панели **PlantIdentification** выполнить следующие действия:

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 11.5).

4. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

1. Задать архитектуру нейронной сети, которая будет моделью управляемого процесса (задание количества нейронов и элементов задержек).

Рекомендации: для линейных объектов до 3/4 порядков количество нейронов должно быть не менее 10 из соображений обеспечения требуемой точности моделирования процессов; количество задержек определяется порядком моделируемого объекта.

2. Задать параметры обучения.

3. Выбрать модель **Simulink**для управляемого процесса.

4. Сгенерировать обучающую последовательность заданногообъёма, запустив модель **Simulink**с помощью кнопки **GenerateTrainingData**. Генерация обучающей последовательности производится с помощью воздействия ряда ступенчатых сигналов на модельуправляемого процесса и снятия значений на выходе и входе моделичерез каждый шаг квантования. Графики входного и выходного сигнала отображаются в окне **PlantInput-OutputData**.

Иллюстрация рисунками 11.5-11.15.

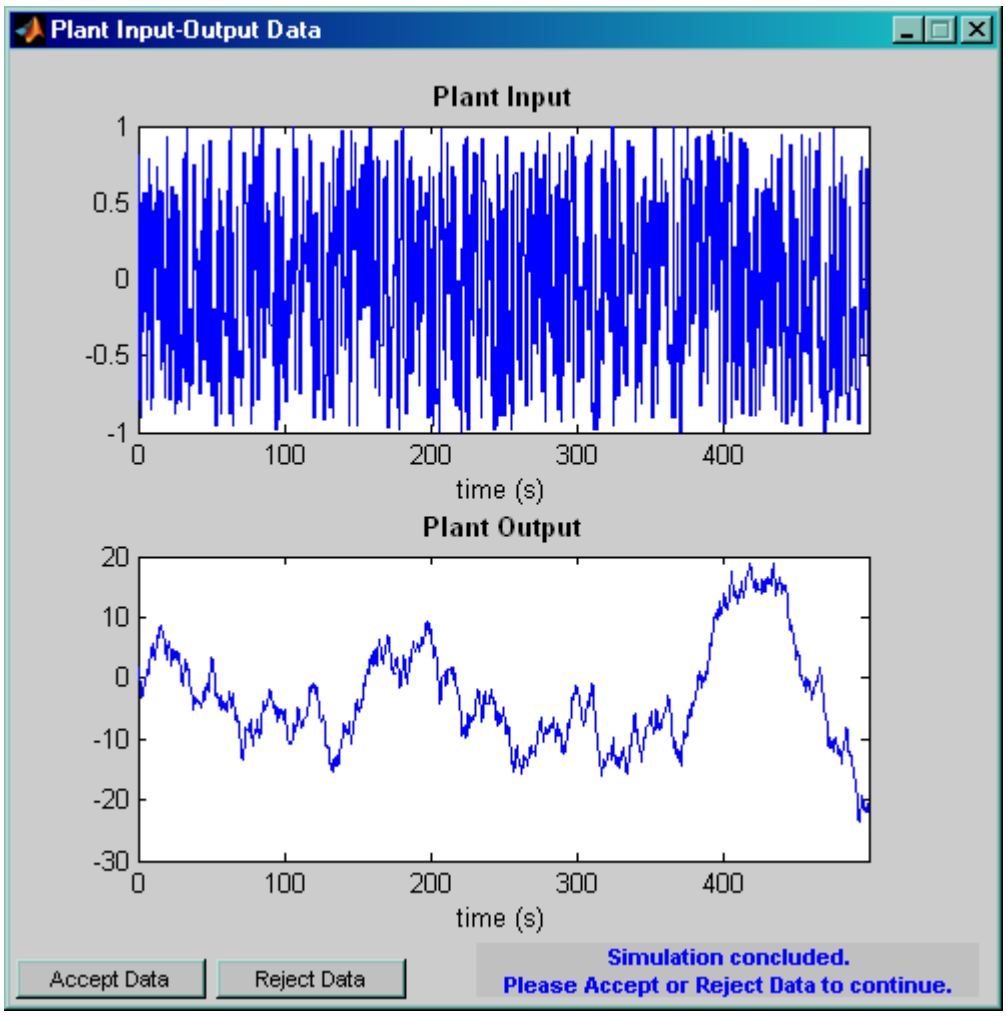


Рис.11.5 Пример формирования входной и выходной последовательности для обучения нейросети

5. По завершении генерации обучающей последовательности необходимо либо принять эти данные, нажав на кнопку **AcceptData**, итогда они будут использованы для обучения нейронной сети, либоотвергнуть их, нажав кнопку **RejectData**, и повторить процесс иден

тификации управляемого процесса, представленного моделью **Simulink**.

6. После получения обучающей последовательности необходимоустановить требуемые параметры обучения и с помощью кнопки**TrainNetwork**запустить процесс обучения нейронной сети.

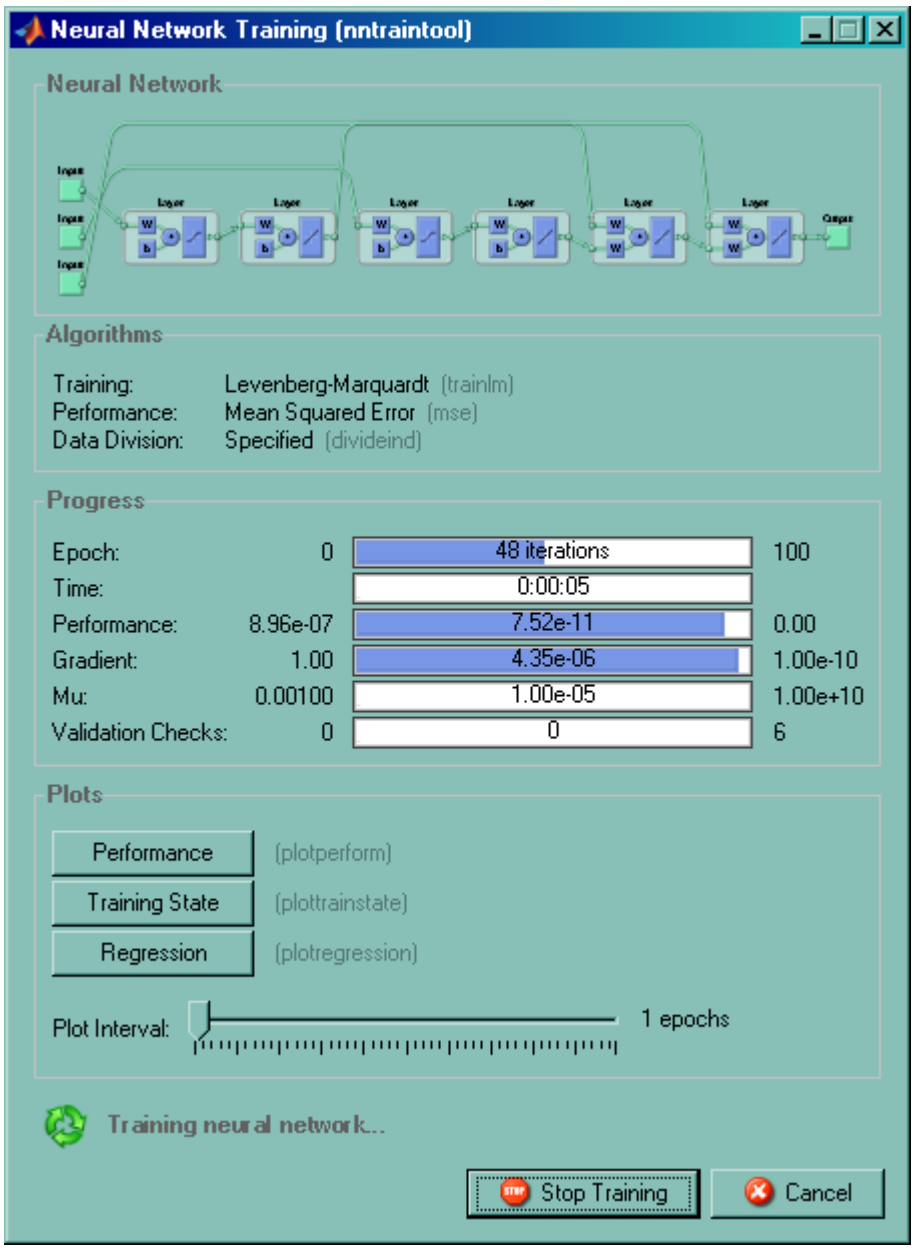


Рис.11.6

7. После завершения обучения его результаты отображаются награфиках изменения ошибки сети для обучающей, контрольной итестирующей последовательностей, а также выходных значений модели и сети при подаче на вход указанных последовательностей.

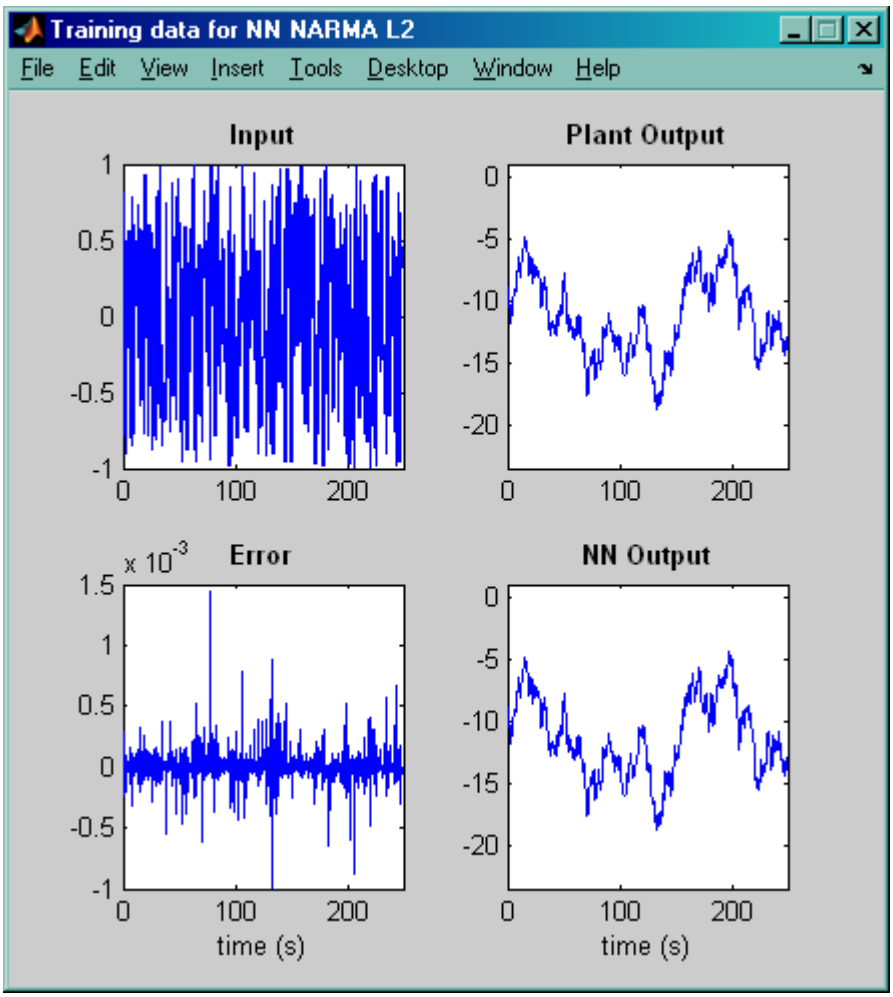


Рис.11.7 Анализ качества обучения

8. Если результаты обучения приемлемы, то надо сохранить параметры нейросетевой модели управляемого процесса и приступитьк синтезу регулятора того или иного класса, нажав кнопки **Apply**и **Ok**.

9. Если результаты обучения неприемлемы, то следует нажатькнопку **Cancel**и повторить процесс идентификации сначала, изменяяархитектуру сети и параметры обучающей последовательности.

10. Обучающую последовательность можно импортировать израбочей области или из файла, нажав на кнопку **ImportData**. Еслинеобходимо обучающую последовательность сохранить в рабочейобласти или в файле для подбора параметров архитектуры нейронной сети, то следует после получения данных нажать на кнопку**ExportData**.

11. Удалить только что сгенерированные данные при необходимости можно с помощью кнопки **EraseGeneratedData**.

Таким образом, диалоговая панель **PlantIdentification**позволяетидентифицировать управляемый процесс, представленный в виде имитационной модели **Simulink**, построить

двухслойную нейроннуюсеть прямой передачи сигнала с необходимым числом нейронов илиний задержки, обучить эту сеть для получения нейронной моделиуправляемого процесса, оценить качество обучения и работу нейронной сети.

Для регулятора на основе авторегрессии со скользящимсредним, этап его синтеза отсутствует, так как такой регулятор представляет собой полученную нейросетевую модель управляемогопроцесса с предсказанием.Для регуляторов на основе эталонной модели с предсказанием, этап синтеза необходим.

Работа системы управления с обученной нейросетью при отработке ступенчатых сигналов (Рис. 11.8).

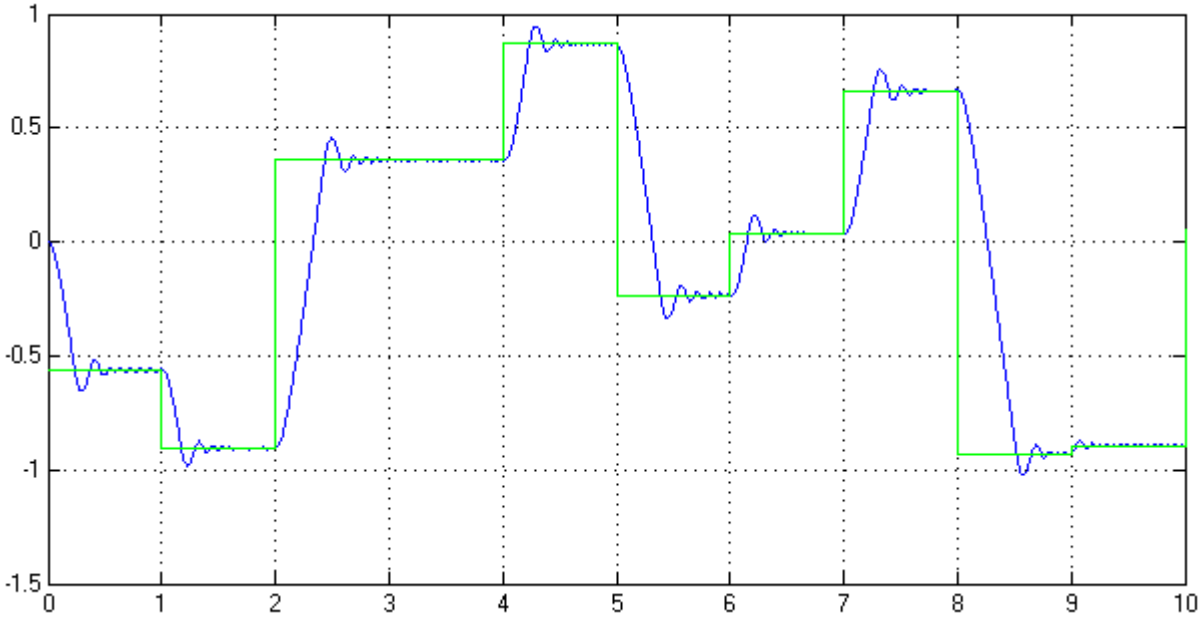


Рис. 11.8.

Управление (Рис. 11.9.):

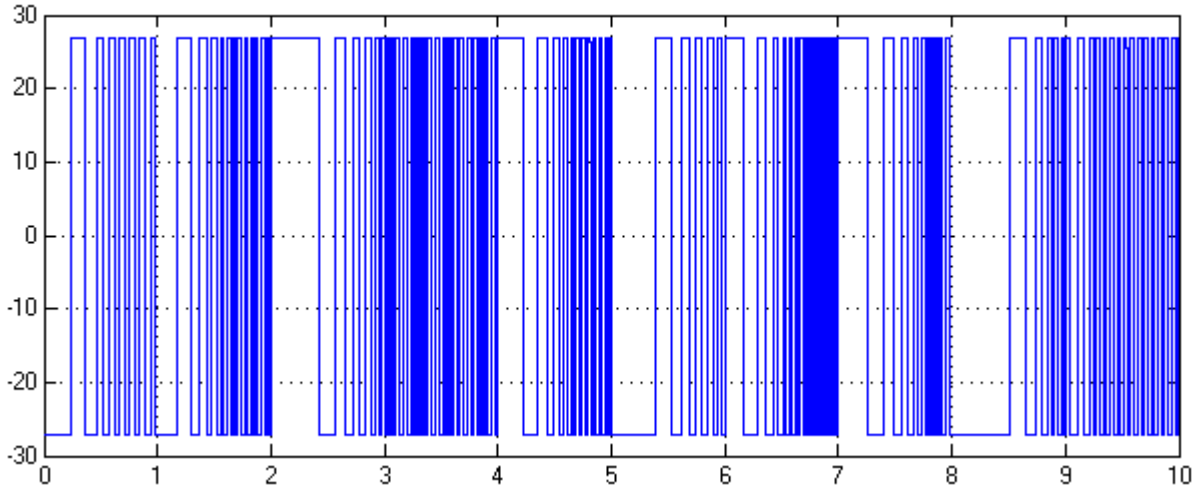


Рис. 11.9.

Из анализа полученных данных следует, что реакция системы на ступенчатые воздействия со случайной амплитудой вполне удовлетворительна, имеет колебательный характер с достаточно быстрым затуханием, на интервале 5 с все установки эффективно отрабатываются. Таким образом, регулятор NARMA-L2, реализованный в виде нейронной сети, можно использовать для управления исполнительным двигателем постоянного тока.

5.ОТЧЕТ

Отчет должен содержать краткое описание нейросетевых регуляторов и решение заданий, представленных в разделе 4 настоящего описания.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие архитектуры нейронных сетей были рассмотрены?
2. Что собой представляет регулятор с предсказанием?
3. Что собой представляет регулятор на основе модели авторегрессии со скользящим средним?
4. Что собой представляет регулятор на основе эталонной модели?
5. Что входит в набор управляющих элементов для задания характеристик обучающей последовательности?

Лабораторная работа №12

ФОРМИРОВАНИЕ И ОБУЧЕНИЕ НЕЙРОНЕЧЕТКОЙ СЕТИ ДЛЯ ЦЕЛЕЙ УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ С ПОМОЩЬЮ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ MATLAB

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Освоение на практике основных приемов работы с редактором anfisedit, позволяющих формировать нейронечеткие модели динамических объектов и нейронечеткие регуляторы с использованием среды Matlab.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В системе MatLabреализован ANFIS-редактор, который позволяет с помощью нейронечеткого описания автоматически синтезировать из экспериментальных данных нечеткие правила. Сеть ANFIS описывает систему нечеткого логического вывода типа Сугено. Параметры сети после обучения настраиваются так, чтобы минимизировать отклонения между результатами моделирования и экспериментальными данными.

Загрузка ANFIS-редактора осуществляется по команде

>>anfisedit

Графическое окно AnfisEditorпоказано на рис.12.1.

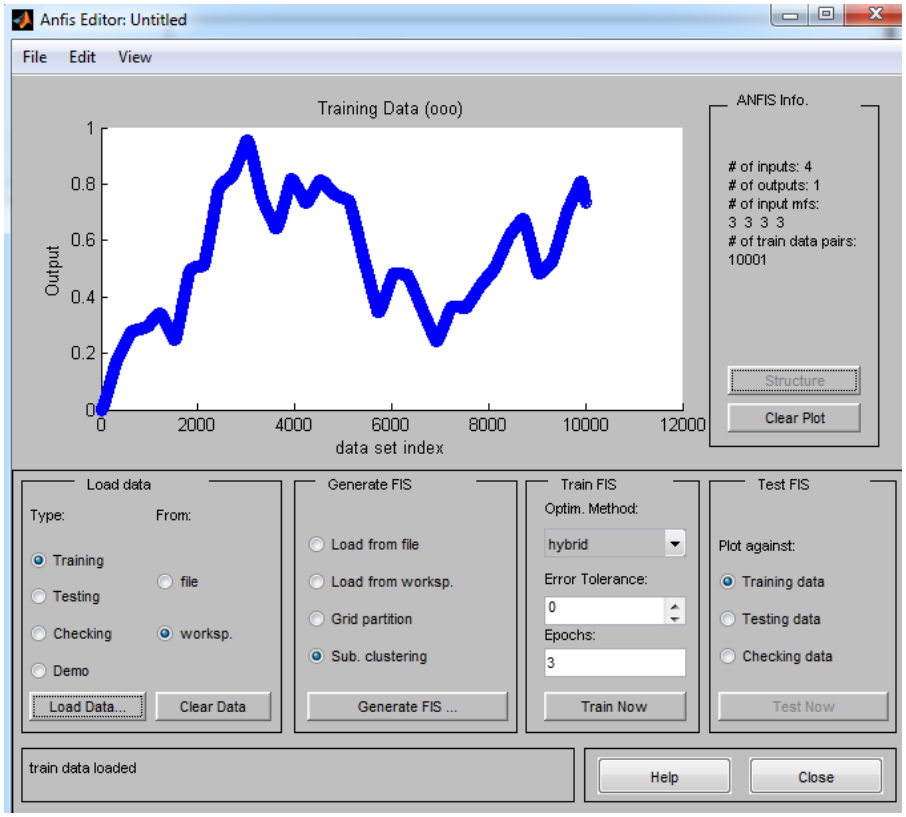


Рис.12.1.

В центре окна изображена область визуализации. В этой области выводится два типа информации:

- при обучении системы – график зависимости ошибки обучения от порядкового номера итерации;

- при загрузке данных и тестировании системы – экспериментальные данные и результаты моделирования. При этом по оси абсцисс откладывается порядковый номер строчки данных в выборке (обучающей, тестирующей или контрольной), а по оси ординат – значение выходной переменной для данной строчки выборки. Используются следующие символы: голубая точка (.) – тесирующая выборка; голубая окружность (o) – обучающая выборка; голубой плюс (+) – контрольная выборка; красная звездочка (*) – результаты моделирования.

Правее области визуализации находится область свойств ANFISinfo.

В области свойств выводится информация о количестве входных и выходных переменных, о количестве функций принадлежностей для для каждой входной переменной, а также о количестве строчек в выборках.

В этой области расположены две кнопки: Structureи ClearPlot.

Нажатие кнопки Structureоткрывает новое графическое окно, в котором система нечеткого логического вывода представляется в виде нейро-нечеткой сети. Количество входов этой сети зависит от числа столбцов обучающей выборки. Количество нейронов – 2-4 слоя соответствует количеству правил.

На рис.12.2. показан пример сети ANFIS, в которой две входные переменные (для описания каждой использовано по три терма) и восемь правил. Количество правил зависит от выбора параметров в области GenerateFIS.

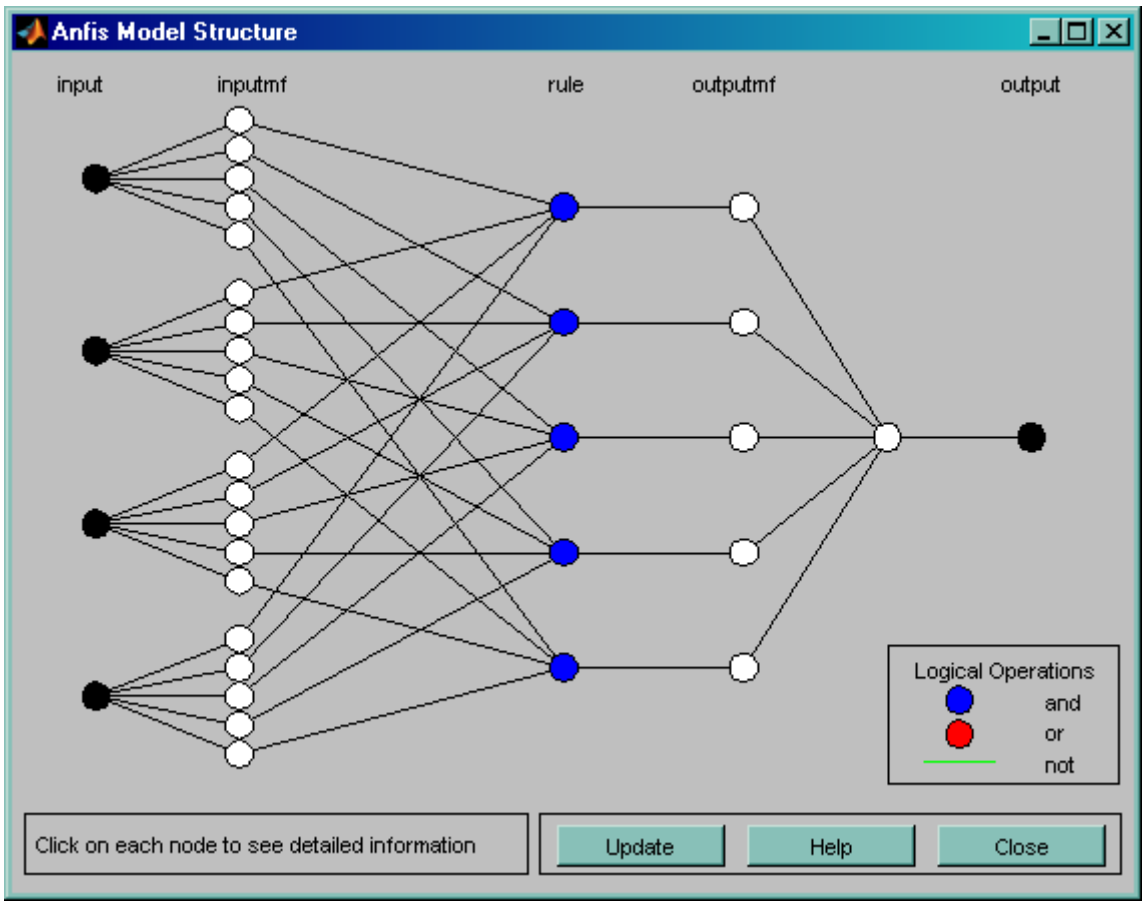


Рис.12.2.

Нажатие кнопки ClearPlot позволяет очистить область визуализации.

В области загрузки данных LoadDataрасположены:

- Type - меню выбора типа данных: Traning- обучающая выборка; Testing - тестирующая выборка; Checking- контрольная выборка; Demo- демонстрационный пример;

- From - меню выбора источника данных: Disk–диск; Worksp. –рабочая область MatLab;

- LoadData – кнопка загрузки данных;

-ClearData - кнопка очистки данных.

В течение одного сеанса работы ANFIS - редактора следует загружать данные одного формата, т.е. количество входных переменных в выборках должно быть одинаковым.

GenerateFIS – меню создания исходной системы нечеткого логического вывода, содержащее альтернативы:

- Loadfromdisk – загрузка существующей системы с диска;

- Loadfromworksp. – загрузка системы из рабочей области MatLab;

-– генерирование описания входных переменных по методу решетки (без кластеризации);

- Sub. Clustering – генерирование по методу субкластеризации.

В области также расположена кнопка Generate, по нажатию котрой генерируется исходная система нечеткого логического вывода.

При выборе Gridpartition появляется окно ввода параметров метода решетки (рис.12.3), в котором нужно указать количество термов для каждой входной переменной и тип функции принадлежности для входных и выходной переменных.

Таким образом, при выборе метода решетки выполняется нечеткое разбиение базовых шкал входных переменных. Настройка параметров термов не требуется, а количество правил равно произведению мощносте терм-множеств лингвистических переменных, описывающих посылки (см. рис.12.3).

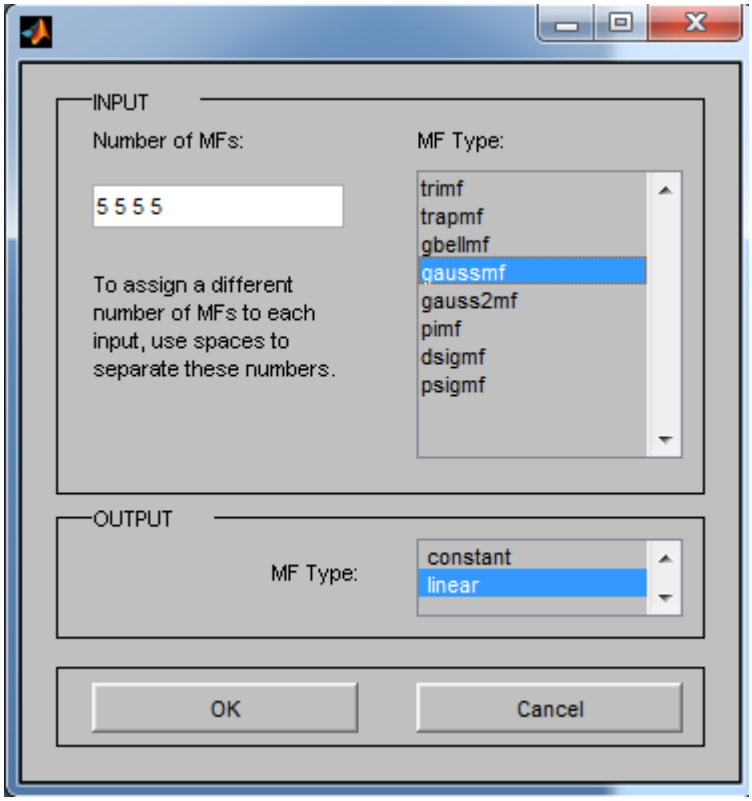


Рис.12.3.

При выборе Sub.clustering появляется окно ввода следующих параметров метода субкластеризации (рис.12.4).

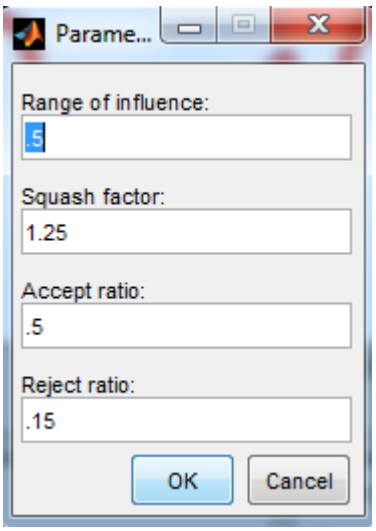


Рис.12.4.

Нарис.12.4обозначены:

Rangeofinfluence– уровни влияния входных переменных;

Squashfactor– коэффициентподавления;

Acceptratio – коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть выше потенциала центра первого кластера для того, чтобы центром одного из кластеров была назначена рассматриваемая точка;

Rejectratio – коэффициент, устанавливающий во сколько раз потенциал данной точки должен быть ниже потенциала центра первого кластера, чтобы рассматриваемая точка была исключена из возможных центров кластеров.

Область обучения (TrainFIS) содержит меню выбора метода оптимизации (Optim.method), поле задания требуемой точности обучения (Errortolerance), поле задания количества итераций обучения (Epochs) и кнопка TrainNow, нажатие которой запускает режим обучения. Промежуточные результаты обучения выводятся в область визуализации и в рабочую область MatLab. В ANFIS - редакторе реализованы два метода обучения:

-Backpropa - метод обратного распространения ошибки, основанный на идеях метода наискорейшего спуска;

- Hybrid - гибридный метод, объединяющий метод обратного распространения ошибки с методом наименьших квадратов.

В области тестирования (TestFIS) расположены меню выбора выборки и кнопка TestNow, при нажатии которой происходит тестирование нечеткой системы с выводом результатов в область визуализации.

3. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

4.ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Рассмотреть решение задачи построения нейронечеткой модели динамического объекта, заданного своей передаточной функцией. Тип и параметры передаточной функции задает преподаватель. Нейронечеткая модель в последующем может быть использована для реализации нейронечеткого управления, например, для построения инверсного нейронечеткого регулятора.

5.ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Запустить Matlab.

2. Для передаточной функции, заданной преподавателем, сформировать SimuLink-модель, которая в последующем будет использоваться для обучения нейронечеткой сети.

В качестве примера рассмотрим динамический объект, передаточная характеристика которого имеет вид (рис.12.6):

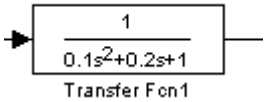


Рис.12.6.

3. Сформировать SimuLink- модель для расчета динамических характеристик объекта (рис.12.7)

Объект:

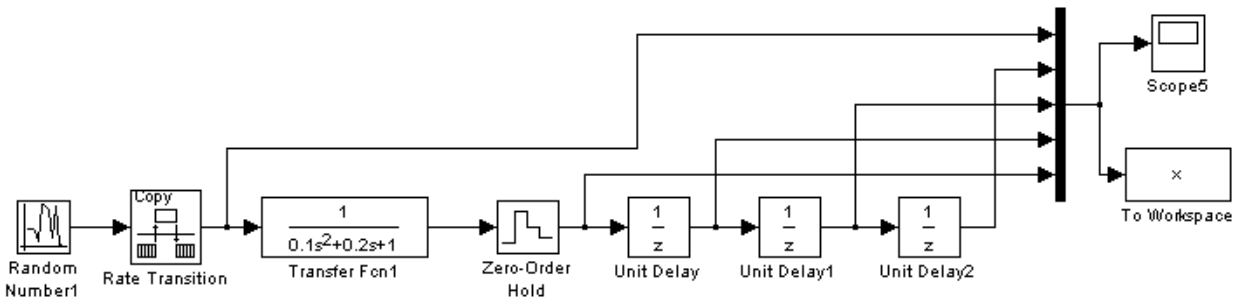


Рис.12.7.

Подача на объект случайного сигнала и снятие тестовых характеристик для последующего обучения нейроэмулятора (рис.12.8).

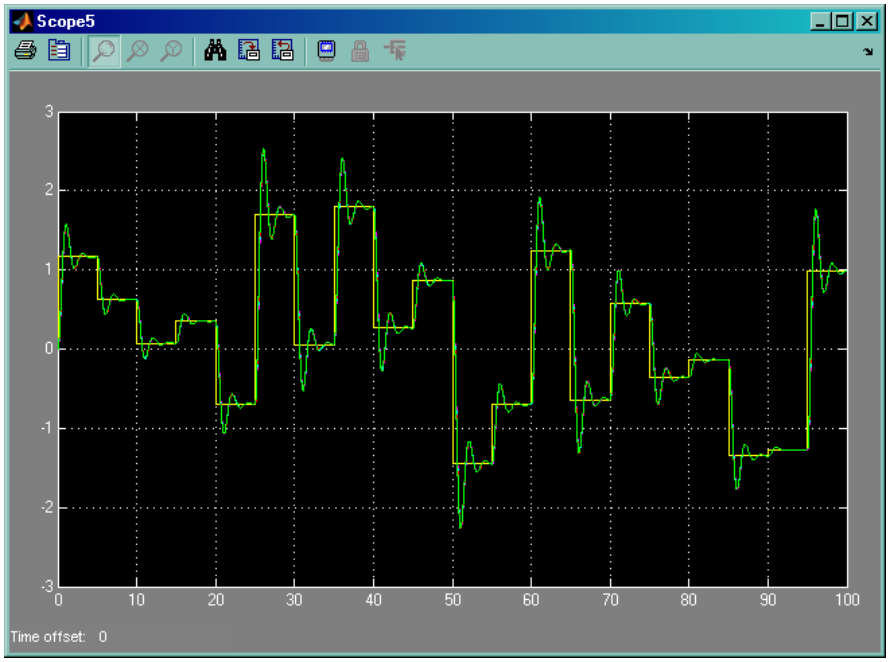


Рис.12.8.

Загрузка полученных данных в редактор Anfis (вызывается командой anfiseditв командной строке Matlab) (рис.12.9):

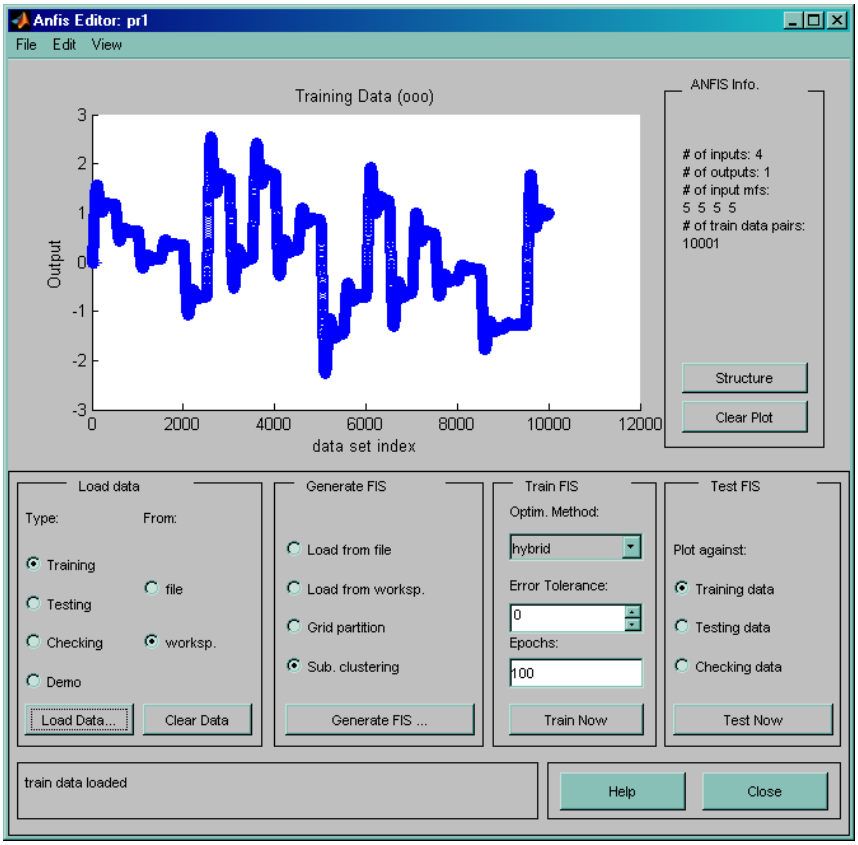


Рис.12.9.

Генерация нечеткого регулятора (GenerateFIS) (рис.12.10):

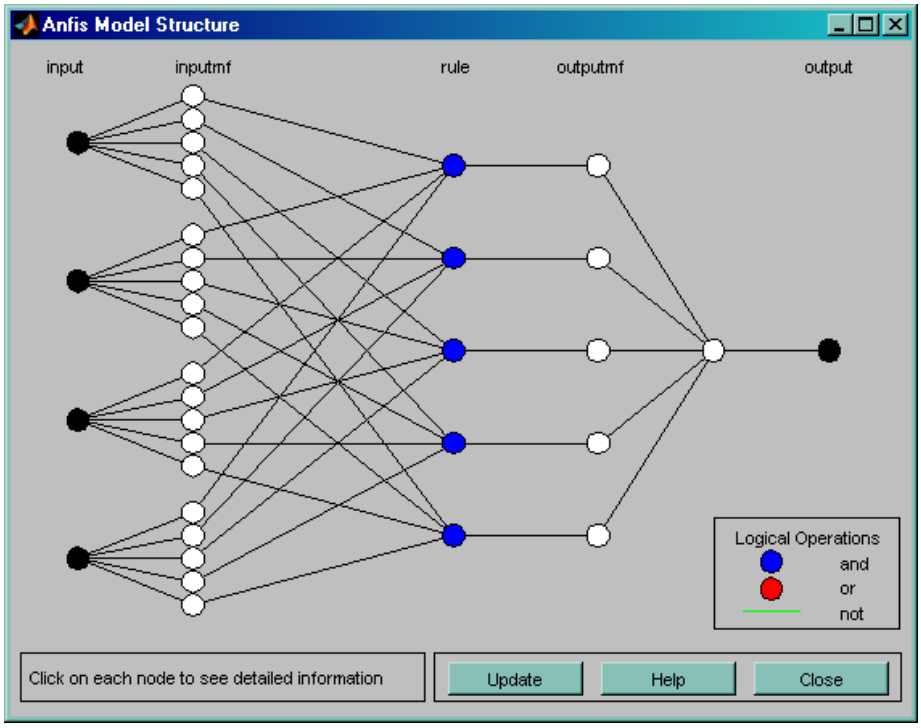


Рис.12.10.

Обучение контроллера (Trainnow), тестирование (TestFIS) и экспорт обученного контроллера в WorkSpace (рис.12.11).

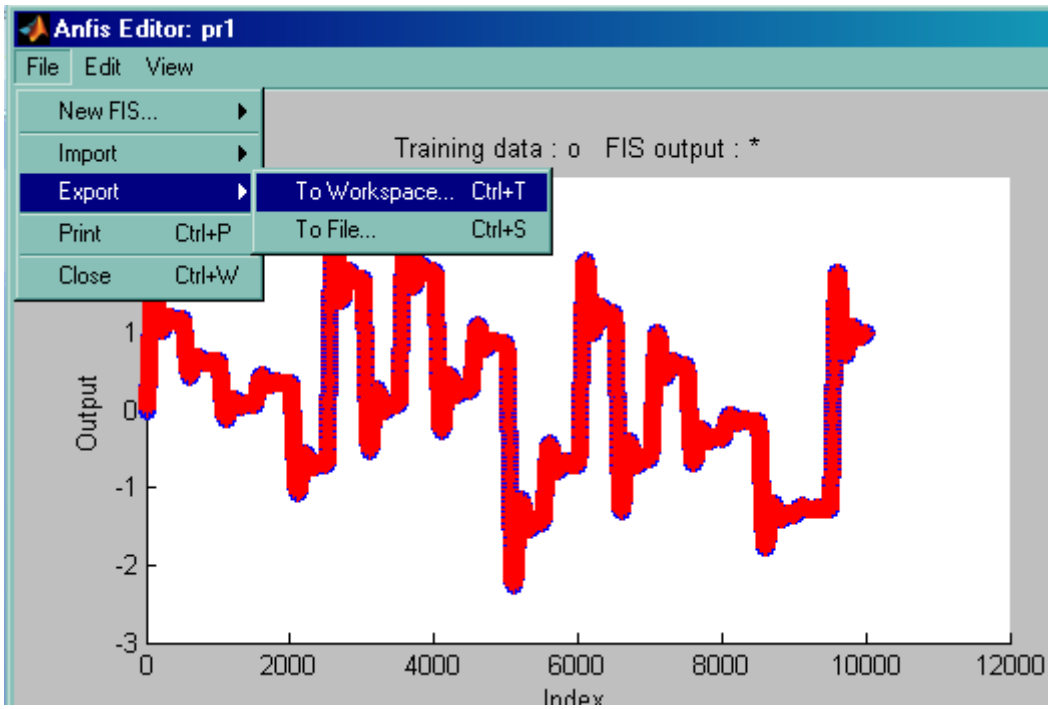


Рис.12.11.

Проверка работы обученногонейроконтроллера (рис.12.12- 12.13).

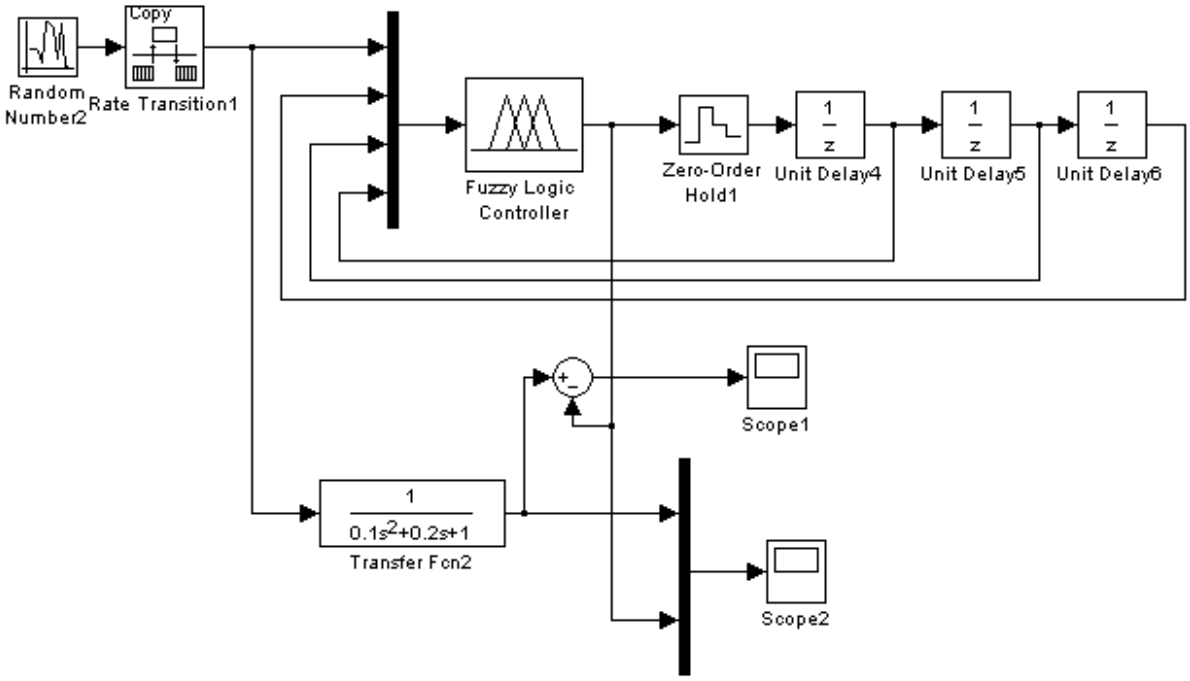


Рис.12.12.

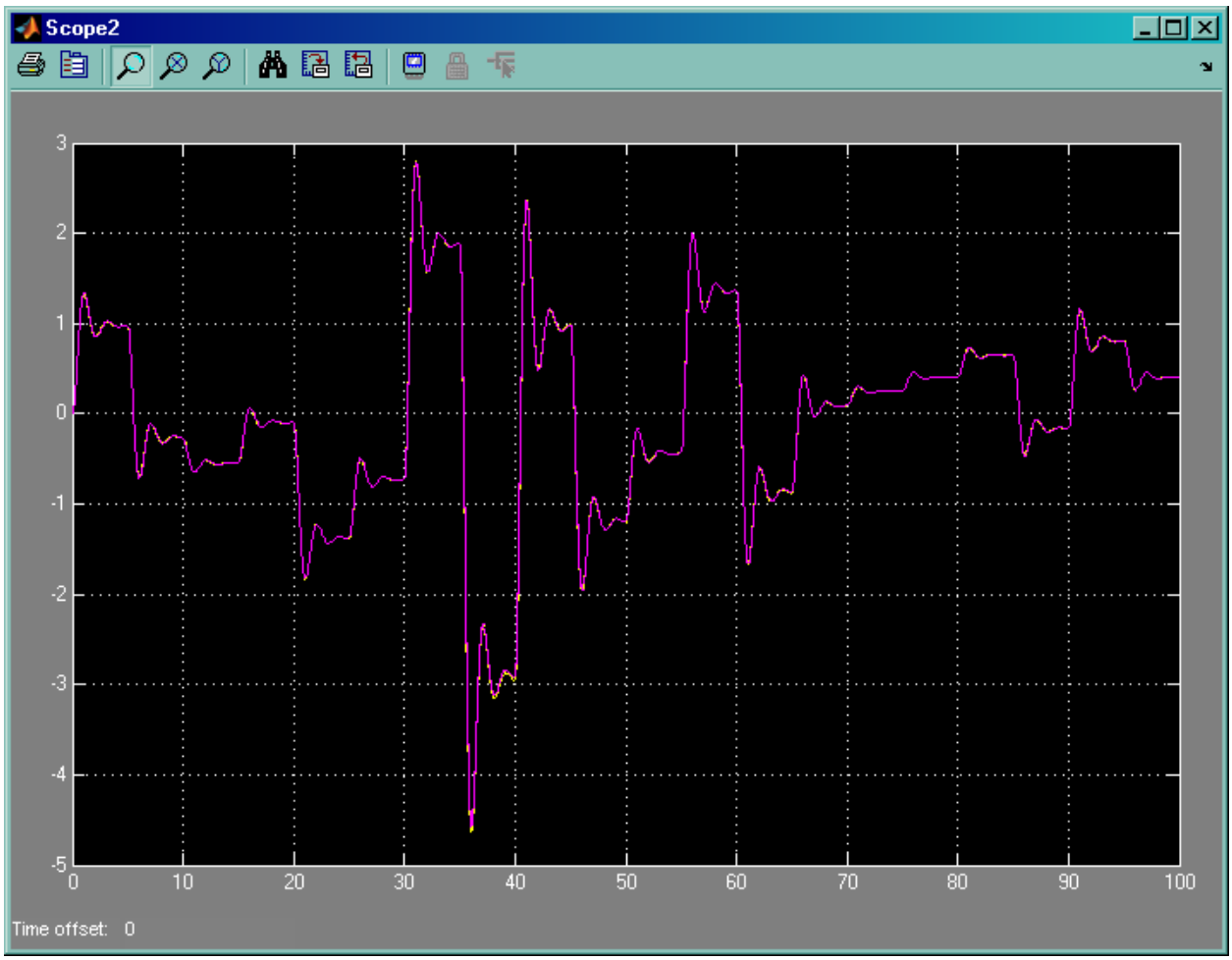


Рис.12.13.

Ошибка (рис.12.14).

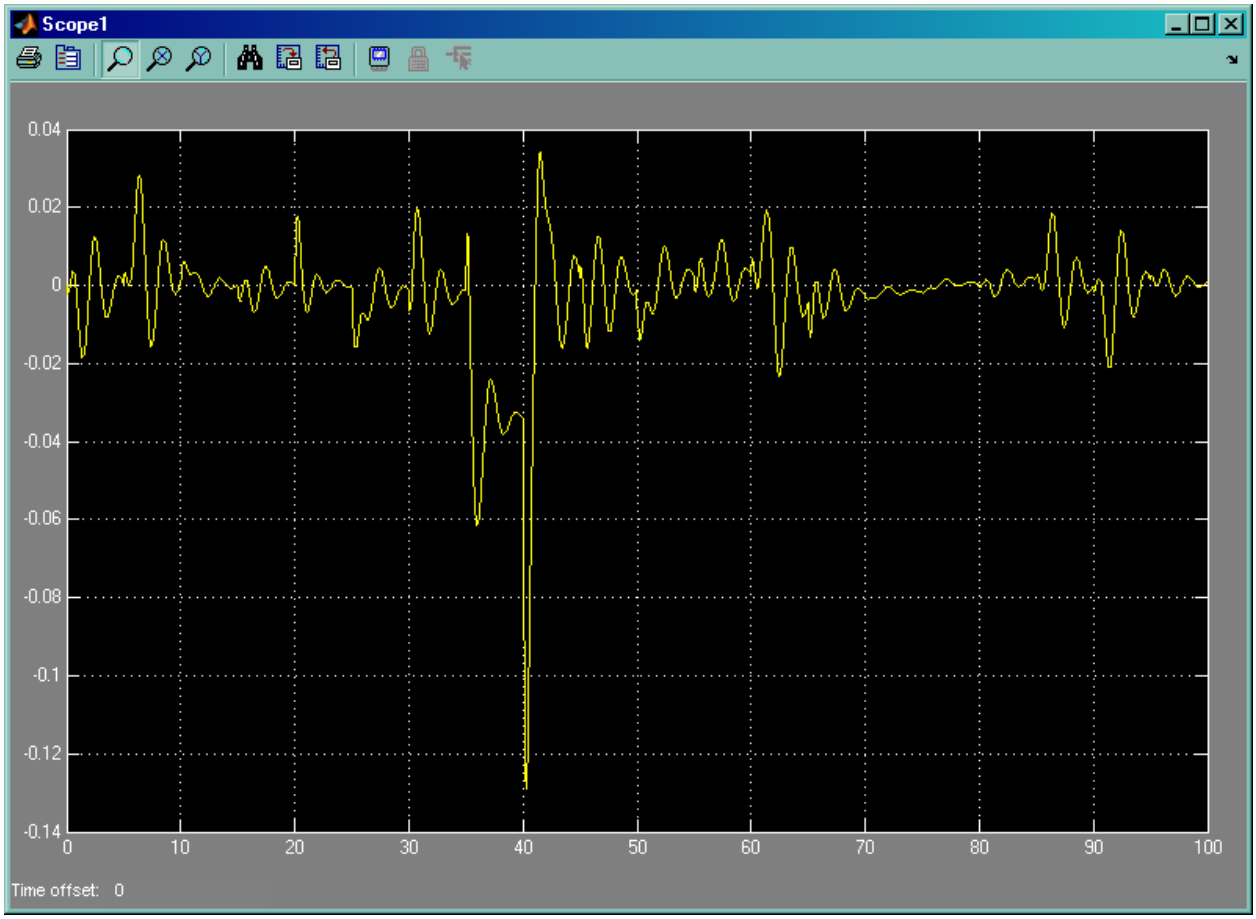


Рис.12.14.

Инверсное управление (рис.12.15).

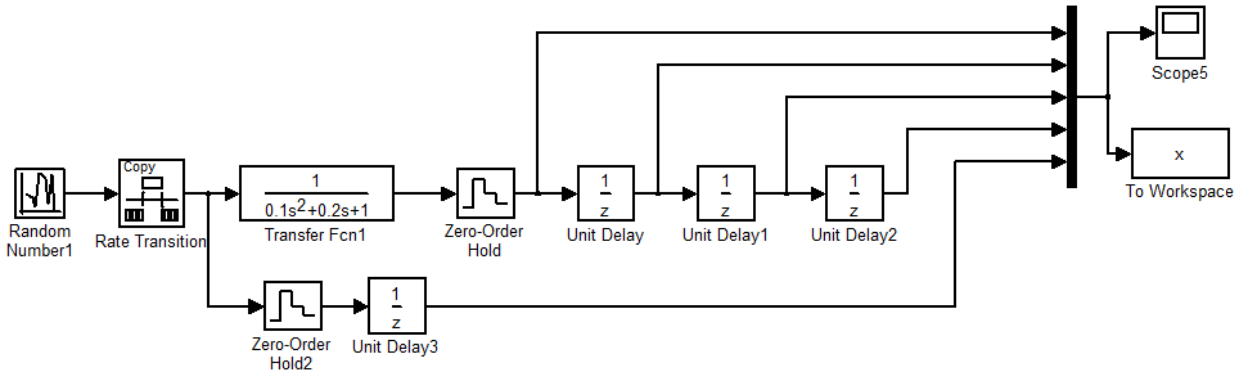


Рис.12.15.

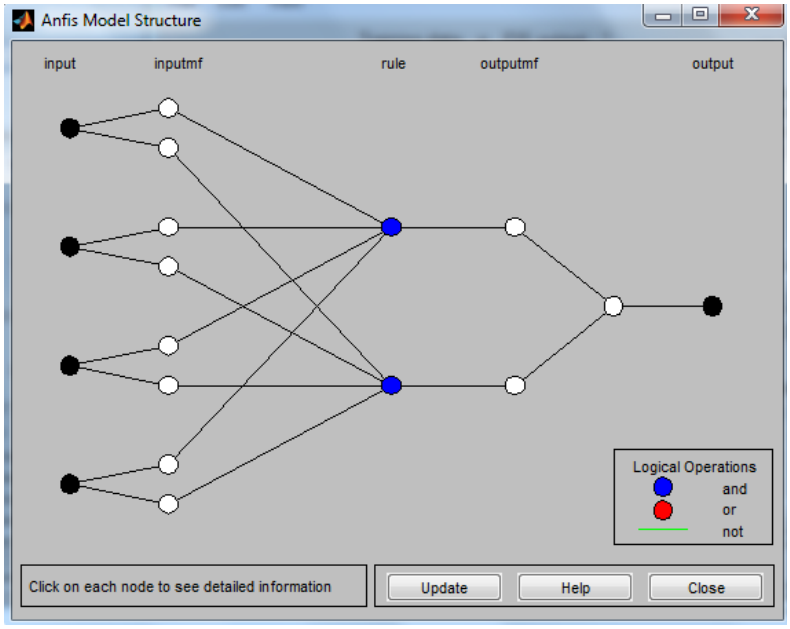


Рис.12.16.

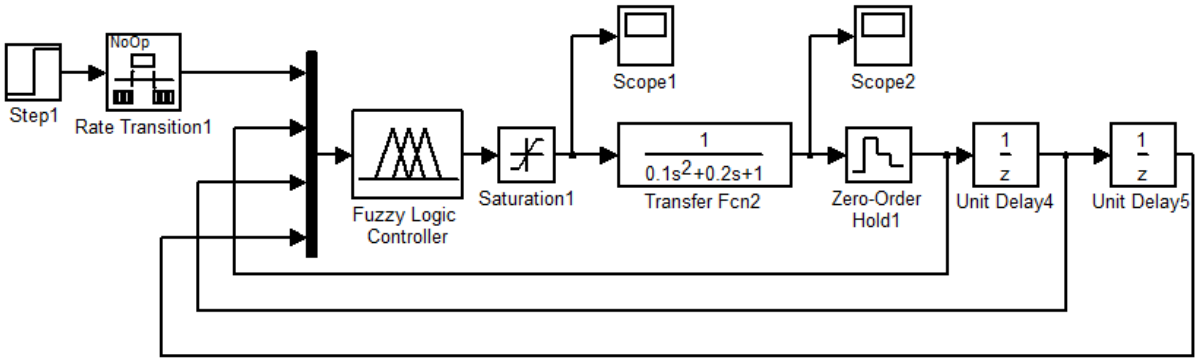


Рис.12.17.

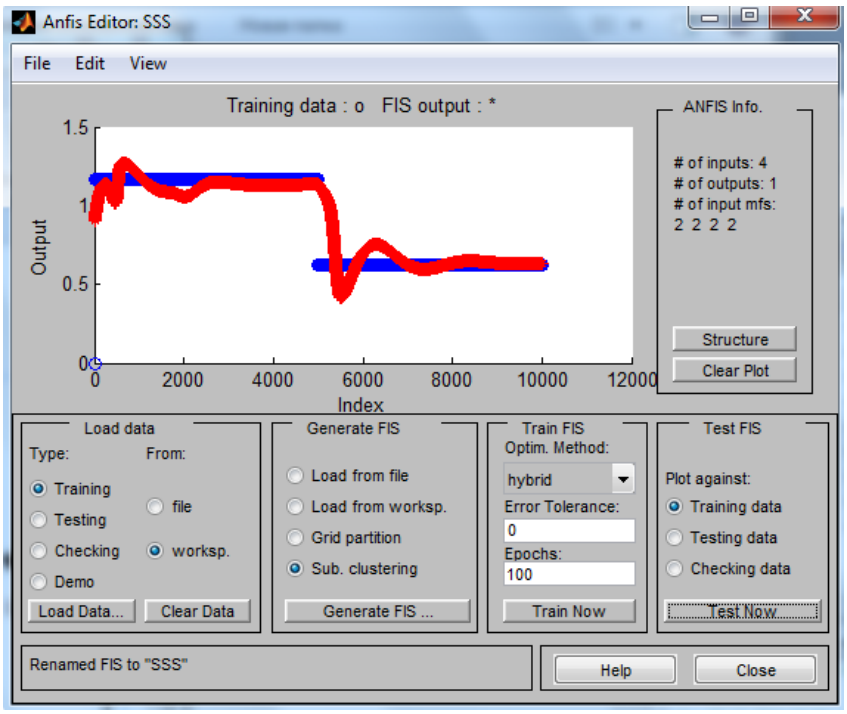


Рис.12.18.

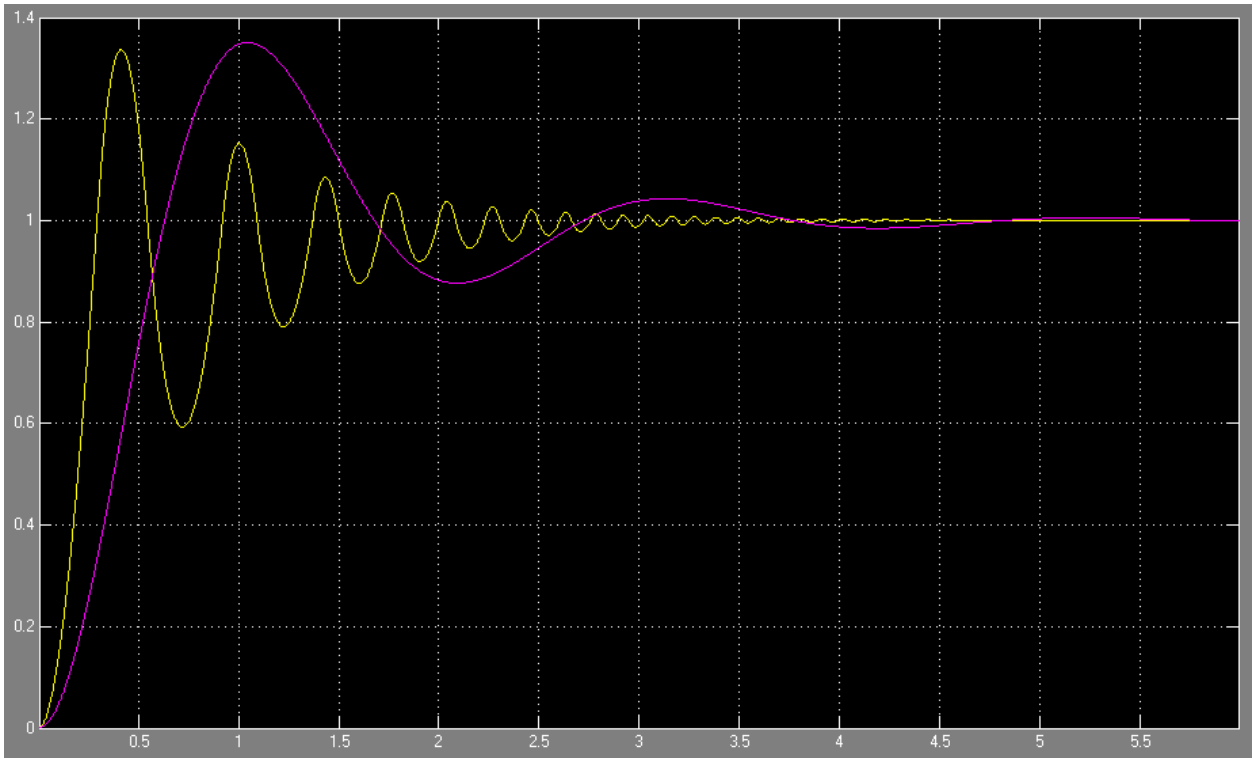


Рис.12.19.

Инверсное управление динамическим объектом (рис.12.20):

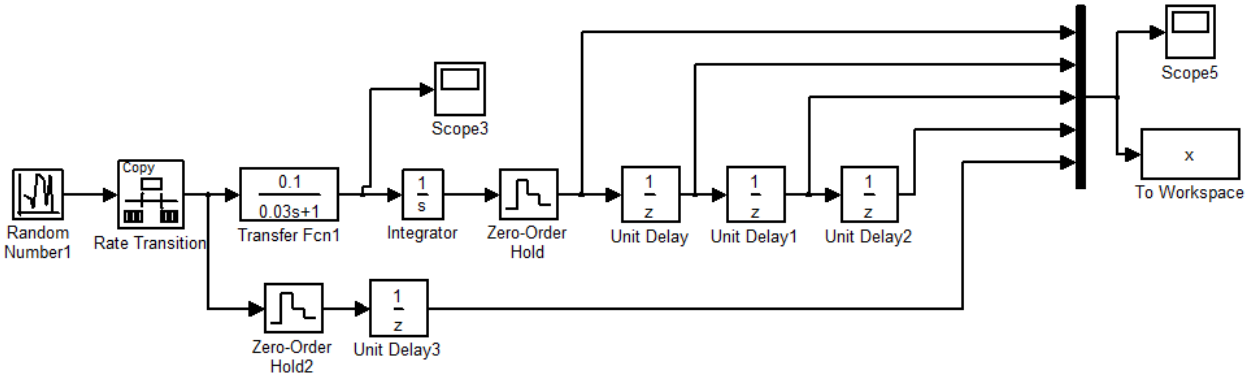


Рис.12.20.

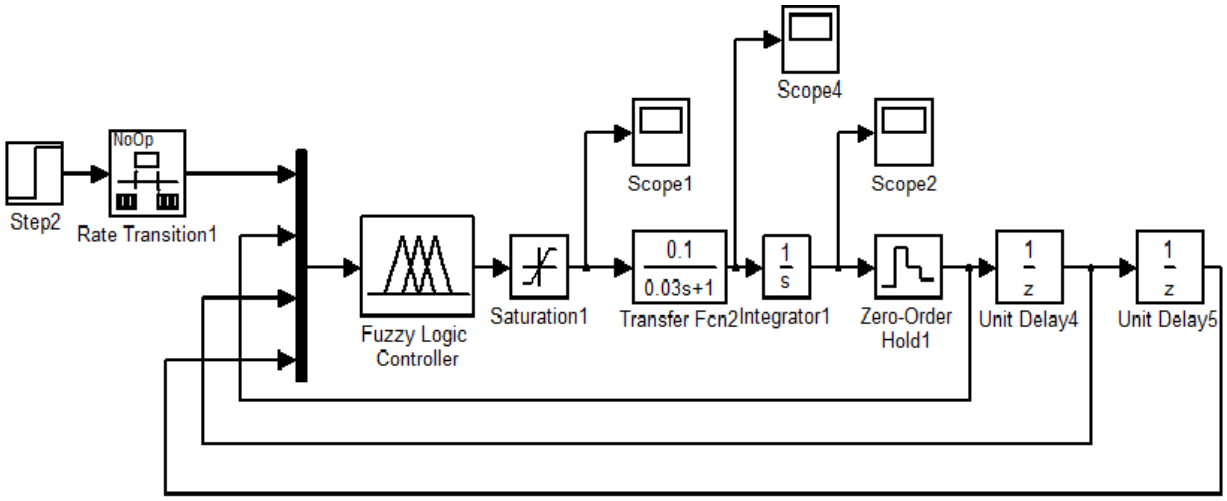


Рис.12.21.

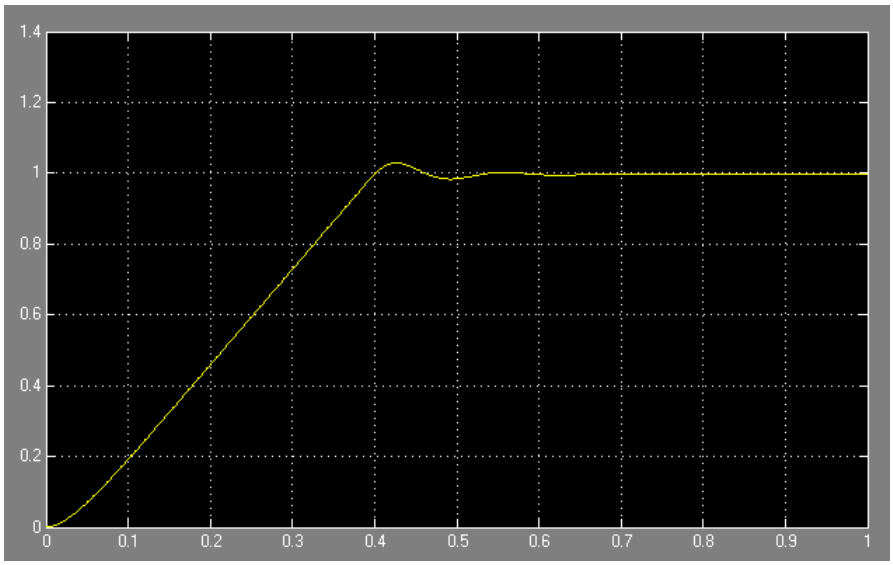


Рис.12.22.

6.ОТЧЕТ

Отчет должен содержать краткое описание ANFIS-редактора и решение заданий, представленных в разделе 4 настоящего описания.

7.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что собой представляет сеть ANFIS?
2. Как функционирует первый слой ANFIS?
3. Какие методы обучения использует ANFIS?
4. К какому классу нейронных сетей относится ANFIS?
5. Как функционирует второй слой ANFIS?
6. Опишите механизм функционирования третьего слоя ANFIS.
7. Опишите механизм функционирования четвертого и пятого слоев ANFIS.
8. Опишите назначение слоев искусственной нейронной сети ANFIS.

Лабораторная работа №13

СОЗДАНИЕ И ОБУЧЕНИЕ НЕЙРОНЕЧЕТКОГО РЕГУЛЯТОРА, РЕАЛИЗУЮЩЕГО СТАНДАРТНЫЕ АЛГОРИТМЫ (П-, ПИ-, ПД-, ПИД- РЕГУЛЯТОРЫ)

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Знакомство с решением практической задачи создания и обучения нейросети заданной структуры с помощью инструмента Network/DataManager.

2. ОБОРУДОВАНИЕ

В лабораторной работе используются персональные компьютеры (не ниже Pentium 4) с установленным пакетом моделирования MatLab (версии не ниже 6.5).

3. ЗАДАНИЕ НА ВЫПОЛНЕНИЕ

Требуется сформировать нейросеть,имеющую два слоя (при использовании этого инструмента имеется ограничение на большее число слоев), 10 нейронов в первом слое с тангенциальными функциями активации (передаточными функциями). Нейросеть должна реализовывать нелинейную функцию:

4.ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. По заданным параметрам необходимо сформировать SimuLink - модель силовой системы ЭСП и оформить ее субмоделью (рис.13.1).

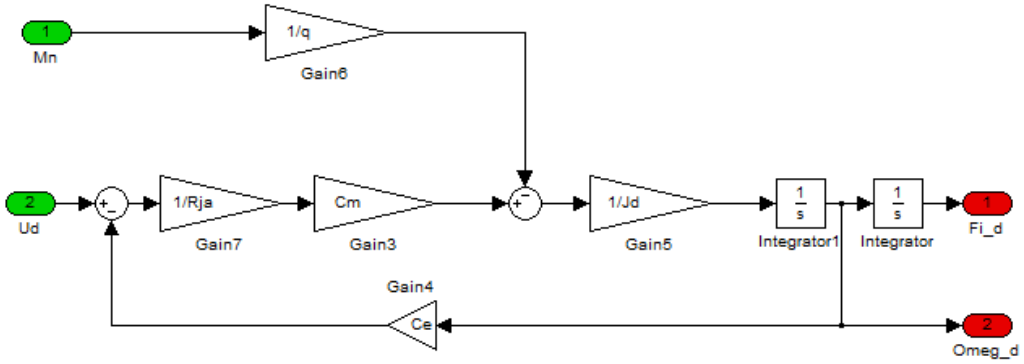


Рис13.1.

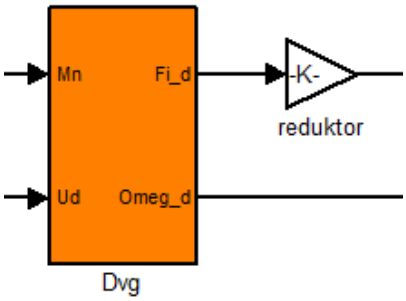


Рис.13.2.

Параметры силовой системы должны определяться m- функцией (рис.13.3)

```
% программа расчета динамических характеристик привода с ЭМУ

Ce = 0.056;                    % коэффициент противоЭДС двигателя
Cm = 0.056;
Rja = 0.14;
Mn = 1;
Jd = 0.0001;

Kmost = 1;
q = 1;
Tmost = 0.0002;
kred = 459;

Kd = 1/Ce;
Km = Rja/(Ce*Cm);
Td = Jd*Rja/(Ce*Cm);
gbx = 2;
Ke = 1000*20;
Ti = 1;
Td= 180*0;
T0 = 0.001;

q0 = Ke+Td/T0;
q1 = -(Ke +Td/T0 - T0/Ti);
q2 = Td/T0;


privod_1;
```

Рис.13.3.

Сформировать SimuLink - модель ЭСП с ПИД - регулятором. (При выполнении задания реализуется регулятор, заданный преподавателем). Схема SimuLink - модель ЭСП с ПИД - регулятором представлена на рис.13.4.

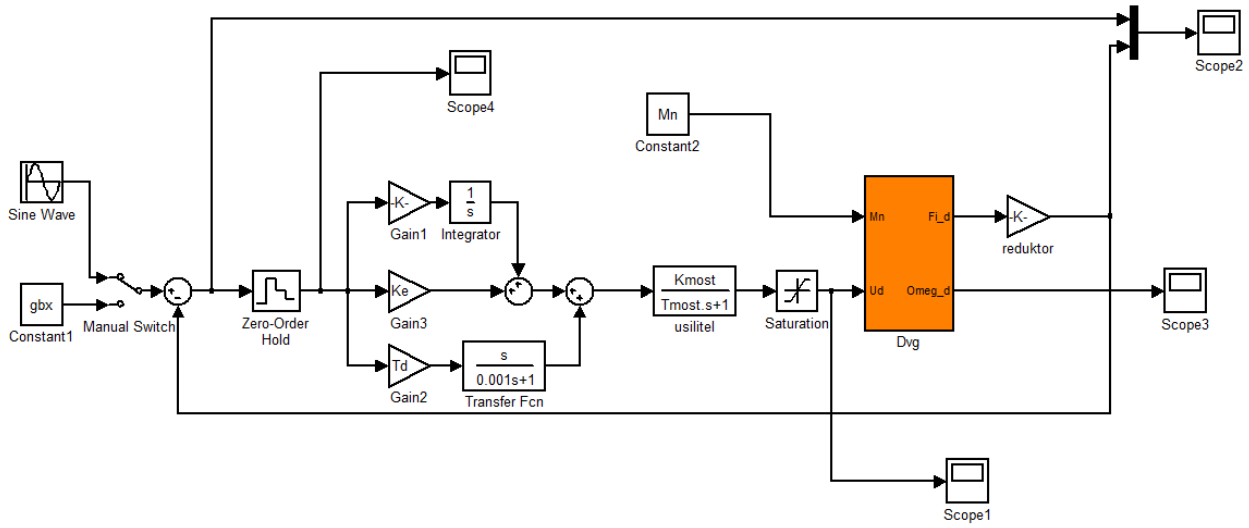


Рис.13.4.

С целью обучения нейронечеткого регулятора, приведенная выше схема дополняется датчиками информации о величине ошибки, определяемой в реальном времени и со смещением. При управлении динамическим объектом рекомендуется количество элементов задержки устанавливать равным порядку объекта управления. Информация о реализации процесса управления записывается в рабочее пространство MatLab (рис.13.5).

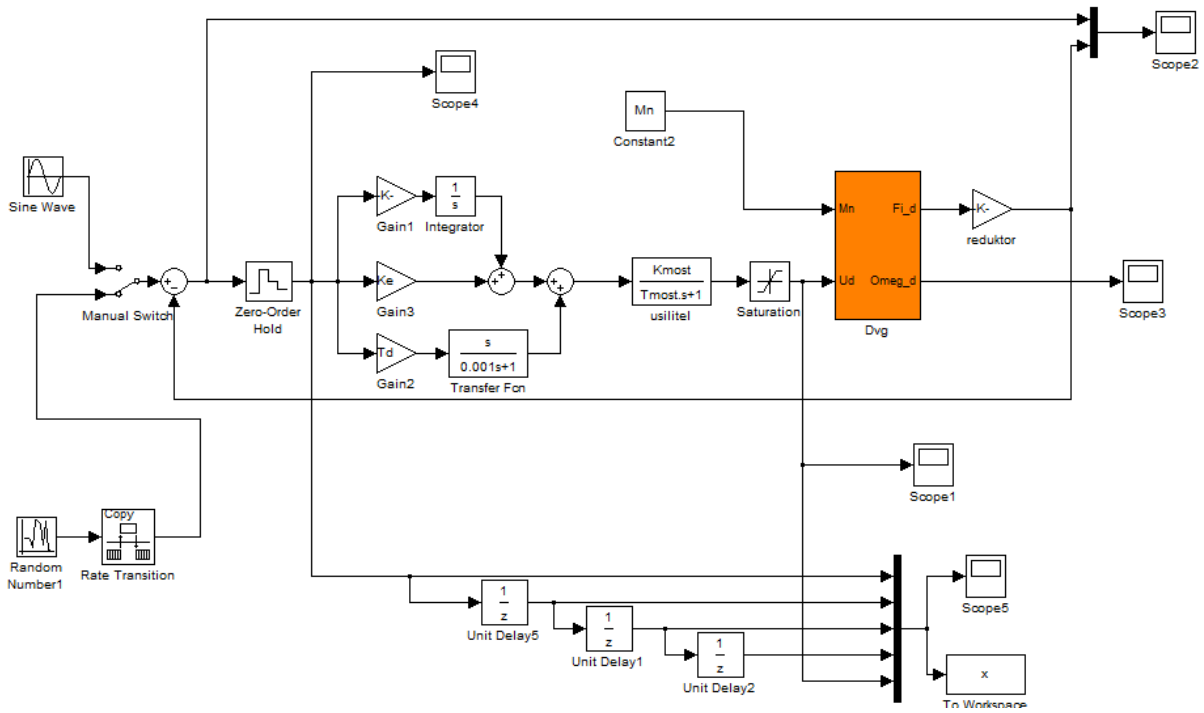


Рис.13.5.

Процедура обучения выполняется аналогично процедурам, описанным в лабораторной

работе №7.

Схема моделирования ЭСП с нейронечетким регулятором имитирующим ПИД-алгоритм управления, представлена на рис.13.6.

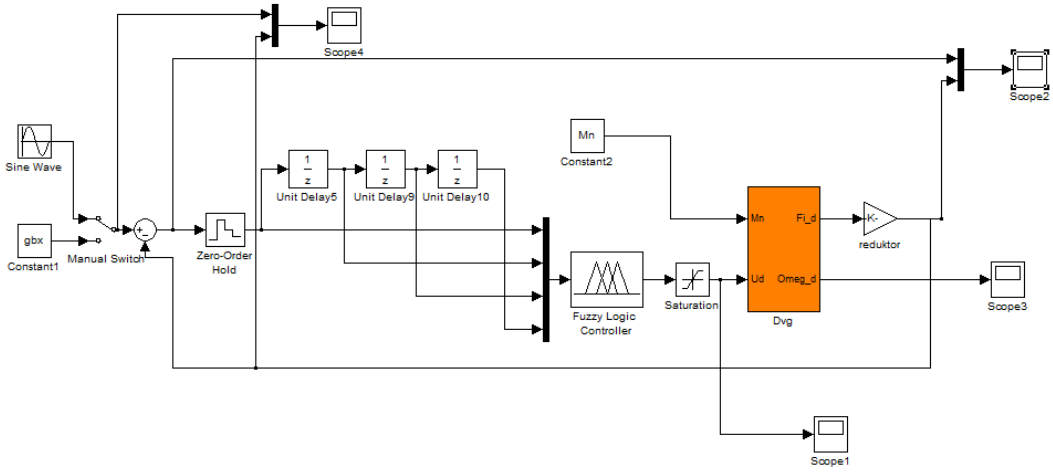


Рис.13.6.

Необходимо отметить, что для отработки типовых входных воздействий нейронечеткий регулятор должен настраиваться на свой тип сигнала. В противном случае необходимо формировать специальные тестовые сигналы.

Для отработки синусоидального входного воздействия настройка нейронечеткого регулятора проводилась на основе гармонического сигнала.

Поверхность управления, сформированная в результате обучения нейронечеткой сети для этого случая представлена на рис.13.7.

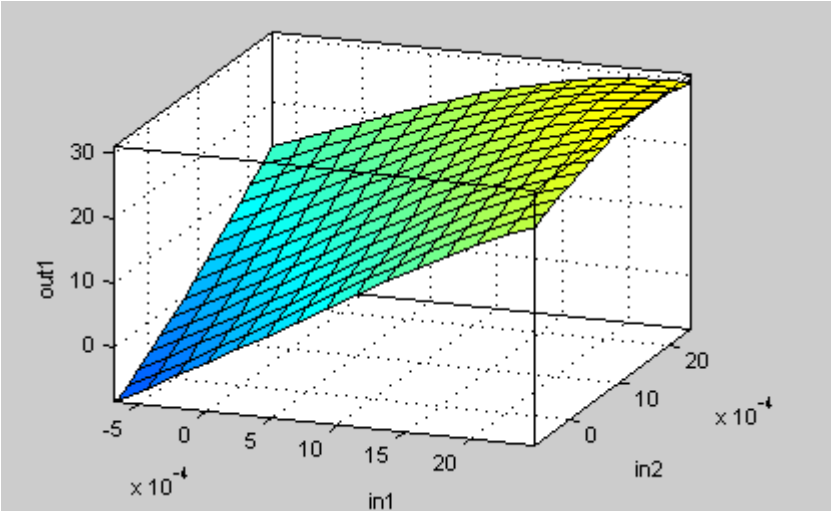


Рис.13.7.

Результаты отработки гармонического входного сигнала и ошибка отработки с применением аналогового и нейронечеткого регулятора представлены на рис.13.8а,б,в (соответственно).

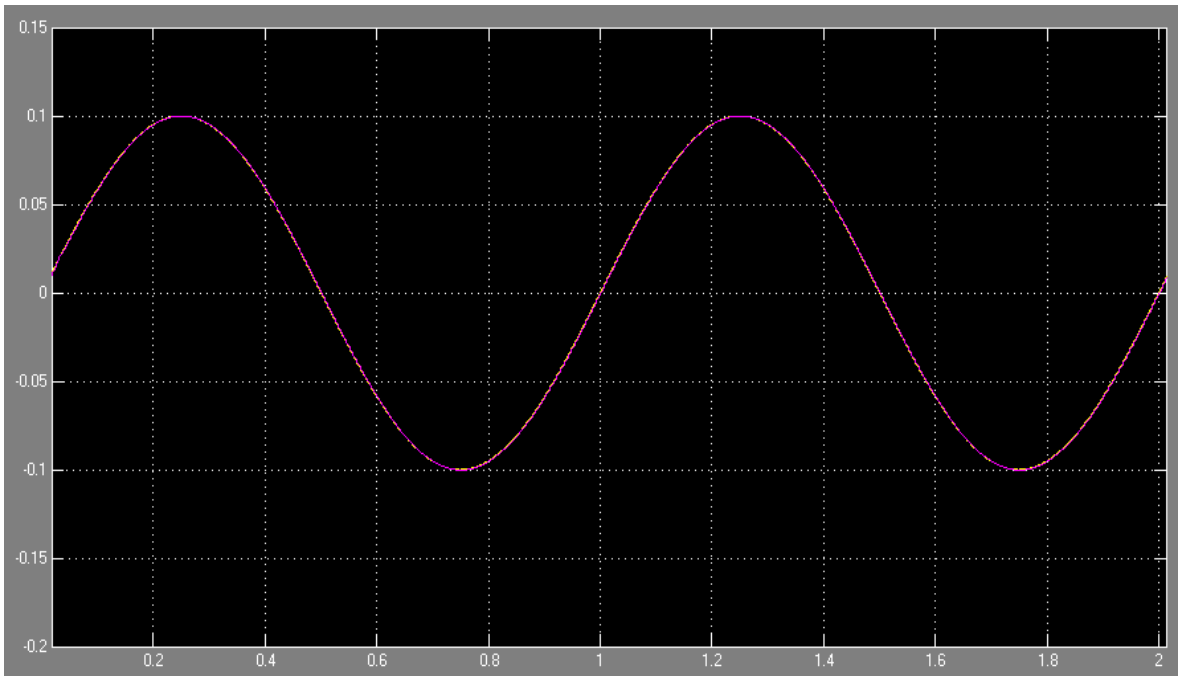


Рис.13.8.а

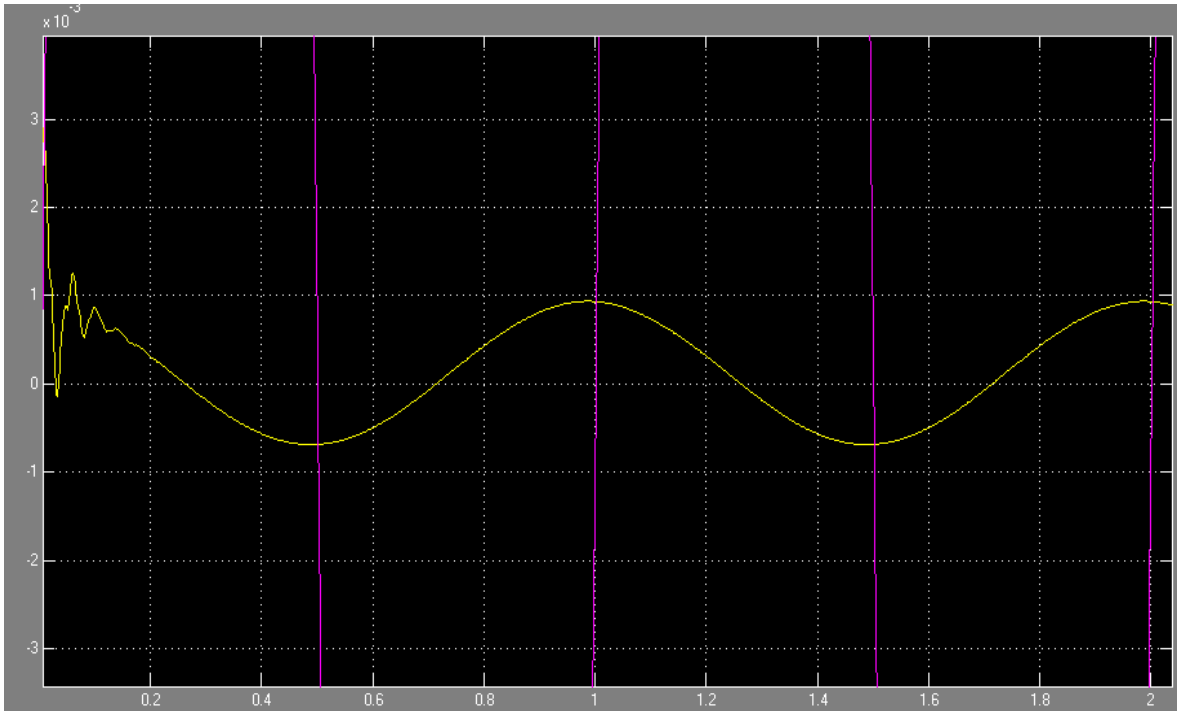


Рис.13.8.б.

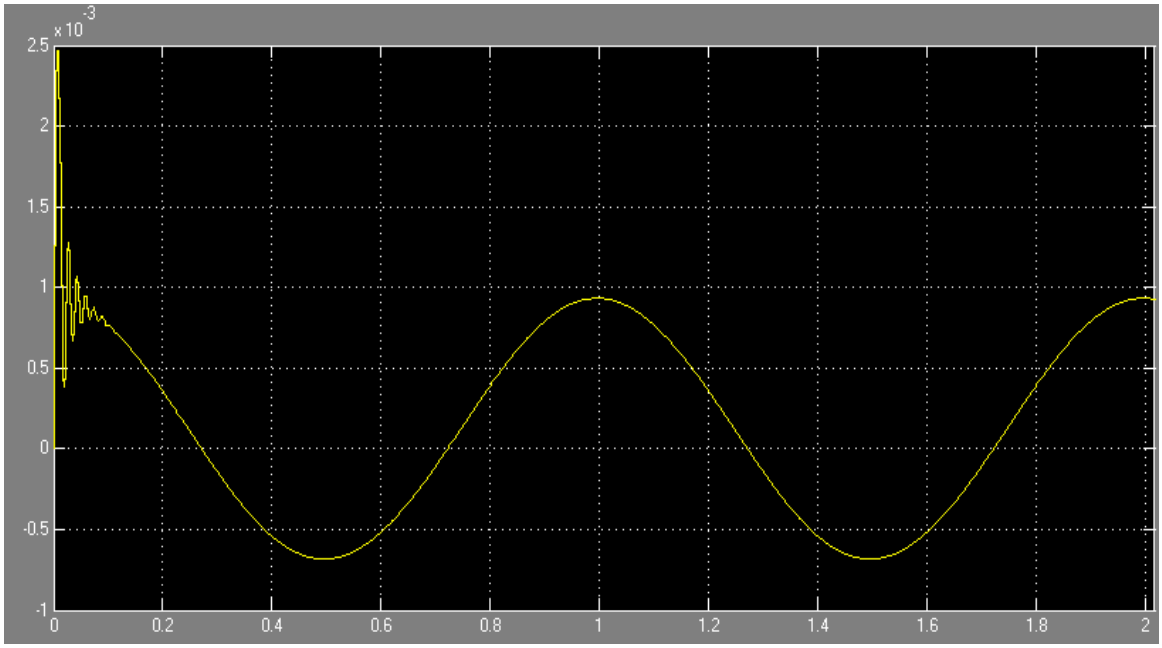


Рис.13.8.в

Для отработки ступенчатых входных воздействий нейронечеткий регулятор обучался на примере отклика на случайный входной сигнал. Сформированная поверхность управления для этого случая имеет вид, представленный на рис.13.9.

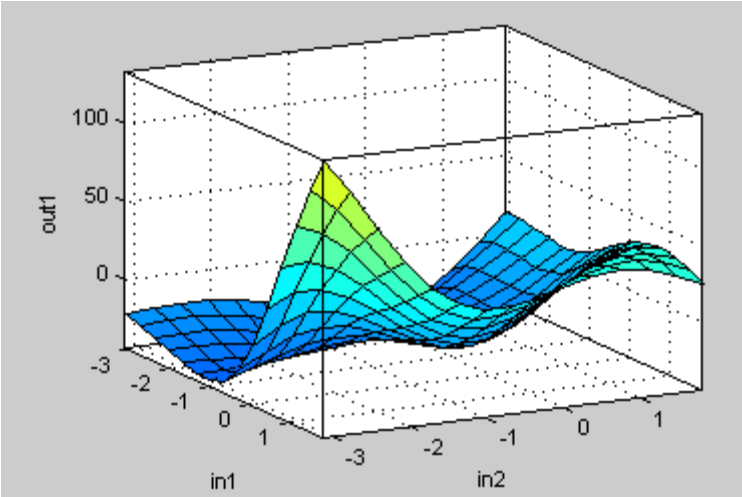


Рис.13.9.

Результаты расчета переходных процессов в ЭСП с нейронечетким регулятором представлены на рис.13.10.а и б.

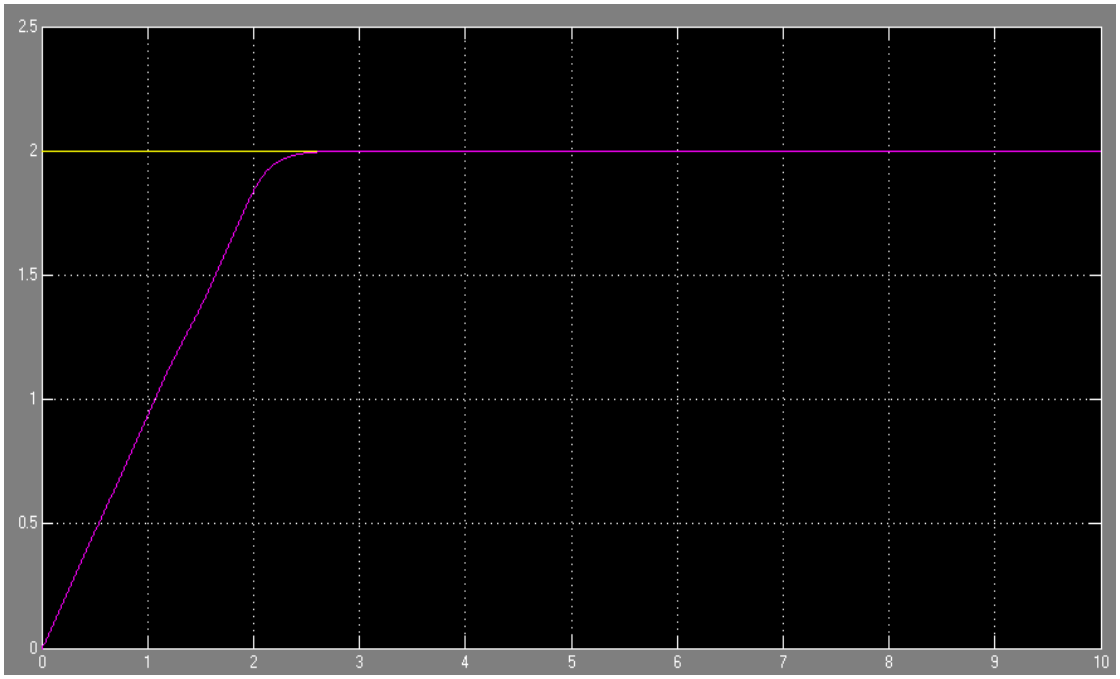


Рис.13.10.а.

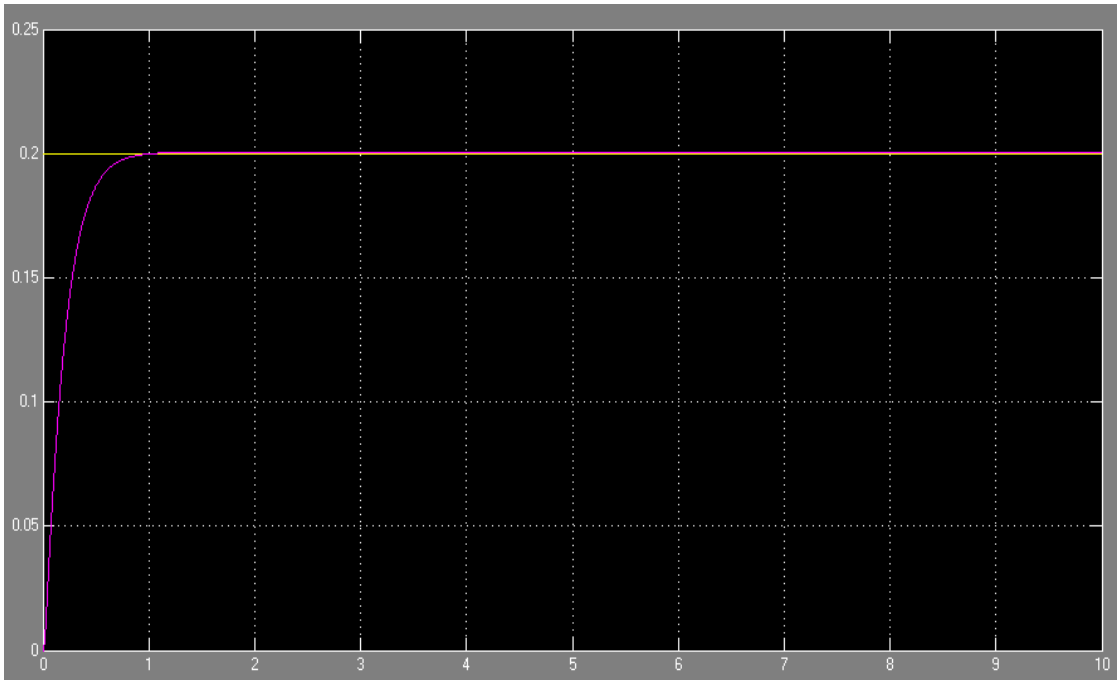


Рис.13.10.б.

5.ОТЧЕТ

Отчет должен содержать решение заданий, представленных в разделе 3 настоящего описания.

6.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На каком примере нейронечеткий регулятор обучается для отработки

гармонического сигнала?

2. На каком примере нейронечеткий регулятор обучается для отработки ступенчатого входного сигнала?

3. Какое влияние на переходный процесс оказывают различные коэффициенты ПИД регулятора?

4. Что такое нечеткий супервизор?

5. Что такое скользящий режим нечеткого регулятора?

6. В чем заключается проблема использования обычных регуляторов П-типа?

СПИСОК ЛИТЕРАТУРЫ:

1. Методы классической и современной теории автоматического управления : учебник для вузов : в 5 т.. Т. 5. Методы современной теории автоматического управления / К. А. Пупков [и др.] ; под ред. К. А. Пупкова, Н. Д. Егупова. 2-е изд., перераб. и доп. М. : МГТУ им.Баумана, 2004. 784 с. : ил. (Методы теории автоматического управления/под общ.ред.К.А.Пупкова) . ISBN 5-7038-2190-8 (Т. 5) ((в пер.)) . ISBN 5-7038-2194-0. [*12 экземпляров ТулГУ*]
2. Гаскаров, Д.В. Интеллектуальные информационные системы : Учебник для вузов / Д.В.Гаскаров. М. : Высш.шк., 2003. 431с. ISBN 5-06-004611-7 /в пер./ : 111.00. [*10 экземпляров ТулГУ*]
3. Ясницкий, Л.Н. Введение в искусственный интеллект : учеб. пособие для вузов / Л.Н. Ясницкий. М. : Академия, 2005. 176с. : ил. (Высшее профессиональное образование: Информатика и вычислительная техника) . ISBN 5-7695-1958-4 : 139.00. [*5 экземпляров ТулГУ*]
4. Усков, А.А. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика / А.А.Усков, А.В.Кузьмин. М. : Горячая линия-Телеком, 2004. 143с. : ил. ISBN 5-93517-181-3 : 123.00. [*3 экземпляра ТулГУ*]
5. Смолин, Д.В. Введение в искусственный интеллект: конспект лекций / Д.В.Смолин. М. : ФИЗМАТЛИТ, 2004. 208с. : ил. ISBN 5-9221-0513-2 /в пер./ : 236.00. [*3 экземпляра ТулГУ*]
6. Павлов, С. Н. Системы искусственного интеллекта. Часть 1 : учебное пособие / С. Н. Павлов. Системы искусственного интеллекта. Часть 1, Весь срок охраны авторского права. Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2011. 176 с. ISBN 978-5-4332-0013-5. [*ЭБС “IPRbooks”*].
7. Пальмов, С. В. Интеллектуальные системы и технологии : учебное пособие / С. В. Пальмов. Интеллектуальные системы и технологии, Весь срок охраны авторского права. Самара : Поволжский государственный университет телекоммуникаций и информатики, 2017. 195 с. ISBN 2227-8397. [*ЭБС “IPRbooks”*].
8. Назаров, Дмитрий Михайлович. Интеллектуальные системы: основы теории нечетких множеств : Учебное пособие для вузов / Назаров Д. М., Конышева Л. К. 3-е изд., испр. и доп. Москва: Юрайт, 2020. 186 с. (Высшее образование) . ISBN 978-5-534-07496-3 : 519.00. [*ЭБС “Юрайт”*]

9. Васильев, В.И.Уфимский авиацион. техн. ин-т. Интеллектуальные системы управления с использованием нечеткой логики : учеб.пособие / В.И.Васильев, Б.Г.Ильясов; Уфимский авиац.техн.ун-т. Уфа, 1995. 100с. : ил. : 6.00. [*1 экземпляр ТулГУ*]
10. Системы искусственного интеллекта. Практический курс : учеб. пособие для вузов / В. А. Чулюков [и др.]. М. : БИНОМ. Лаборатория знаний: ФИЗМАТЛИТ, 2008. 293 с : ил. (Адаптивные и интеллектуальные системы) . ISBN 978-5-94774-731-7 ((в пер.)) . [*1 экземпляр ТулГУ*]
11. Аверкин, А.Н. Нечеткие множества в моделях управления и искусственного интелекта / А.Н.Аверкин [и др.]; под ред. Д.А.Поспелова. М.: Наука, 1986. 312с.: ил. (Проблемы искусственного интеллекта; 8) . ISBN /В пер./ : 2.40. [*1 экземпляр ТулГУ*]
12. Батыршин, И.З. Нечеткие гибридные системы.Теория и практика / Батыршин И.З.[и др.];под ред.Н.Г.Ярушкиной. М. : Физматлит, 2007. 208с. (Информационные и компьютерные технологии) . ISBN 978-5-9221-0786-0 /в пер./ : 130.00. [*1 экземпляр ТулГУ*]
13. Пенькова, , Т. Г. Модели и методы искусственного интеллекта : учебное пособие / Т. Г. Пенькова, Ю. В. Вайнштейн. Модели и методы искусственного интеллекта, 2025-10-09. Красноярск : Сибирский федеральный университет, 2019. 116 с. ISBN 978-5-7638-4043-8. [*ЭБС “IPRbooks”*].
14. Трофимов, В. Б. Экспертные системы в АСУ ТП: учебник / В. Б. Трофимов, И. О. Темкин. Экспертные системы в АСУ ТП, 2025-08-03. Москва, Вологда : Инфра-Инженерия, 2020. 284 с. ISBN 978-5-9729-0480-8 [*ЭБС “IPRbooks”*].
15. Лубенцова, Е. В. Системы управления с динамическим выбором структуры, нечеткой логикой и нейросетевыми моделями: монография / Е. В. Лубенцова. Системы управления с динамическим выбором структуры, нечеткой логикой и нейросетевыми моделями, Весь срок охраны авторского права. Ставрополь : Северо-Кавказский федеральный университет, 2014. 248 с. ISBN 978-5-88648-902-6. [*ЭБС “IPRbooks”*].
16. Сырецкий, Г. А. Моделирование систем. Часть 2. Интеллектуальные системы : учебное пособие / Г. А. Сырецкий. Моделирование систем. Часть 2. Интеллектуальные системы, 2025-02-05. Новосибирск : Новосибирский государственный технический университет, 2010. 80 с. ISBN 978-5-7782-1341-8. [*ЭБС “IPRbooks”*].