МИНОБРНАУКИ РОССИИ

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

Кафедра механики, мехатроники и робототехники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«09» 09 2021 г.

ТЕОРИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

методические указания по выполнению практических и самостоятельных работ

Курск 2021

УДК 621.865
Составители: С.Ф. Яцун, А.В. Мальчиков

Рецензент:

Кандидат технических наук, доцент Юго-Западного государственного университета *А.С. Яцун*

Теория искусственного интеллекта: методические указания по выполнению практических и самостоятельных работ / Юго-Зап. гос. ун-т; сост. С.Ф. Яцун, А.В. Мальчиков, Курск, 2021. – 26с.

Изложены сведения об основах проектирования и применения нейронных сетей в мехатронике и робототехнике. Приведены задания для выполнения практических и самостоятельных заданий.

Методические указания предназначены для студентов направлений 15.03.06 «Мехатроника и робототехника», 15.04.06 «Мехатроника и робототехника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60х84 1/16. Усл.печ.л.
Уч.-изд.л. Тираж экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Практическое занятие №1

РАЗРАБОТКА СИСТЕМЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА, РЕАЛИЗУЮЩЕЙ ПОДХОД МАМДАНИ

Цель работы

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

Теоретическая часть

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

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

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

Формирование логического решения осуществляется следующим образом.

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

$$alfa_i = \min_i\left(\mu_{ik}(x_k)\right)$$

Далее находятся «усеченные» функции принадлежности консеквента:

$$\mu_i(y) = \min_i\left(alfa_i, \mu_i(y)\right)$$

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

$$\mu(y) = \max_i(\mu_i(y)),$$

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

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

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

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

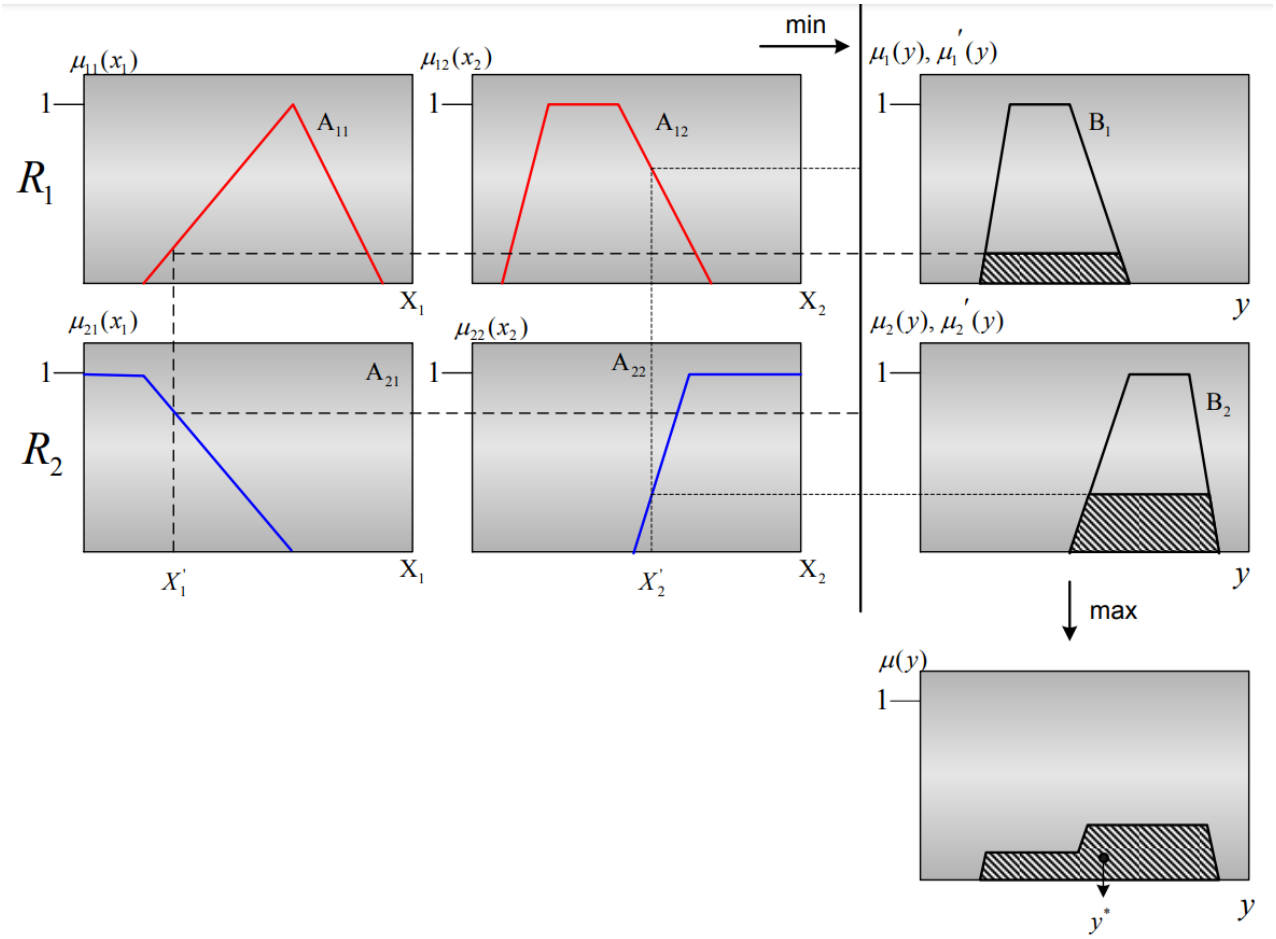

Рисунок 1.1– Графическая интерпретация процесса нечеткого вывода по Мамдани

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

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

Практическая часть

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

```
n = 15;                          % количество точек
x1 = linspace(-7,3,n);           % вектор переменной x1
x2 = linspace(-4.4,1.7,n);       % вектор переменной x1
y = zeros(n,n);                  % массив функции y

for j=1:n
y(j,:)=x1.^2*sin(x2(j)-1);       % расчет функций
end

surf(x1,x2,y);                   % построение графика функции
xlabel('x_1');
ylabel('x_2');
zlabel('y');
```

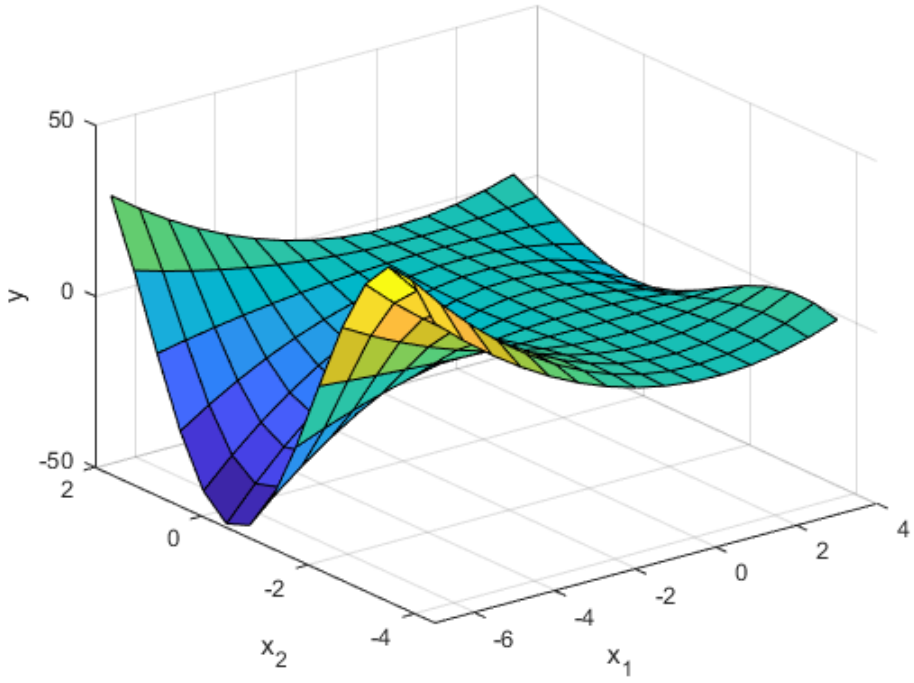

Рисунок 1.2 – График исследуемой функции

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

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

Проектирование нечеткой системы состоит в выполнении следующей последовательности шагов.

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

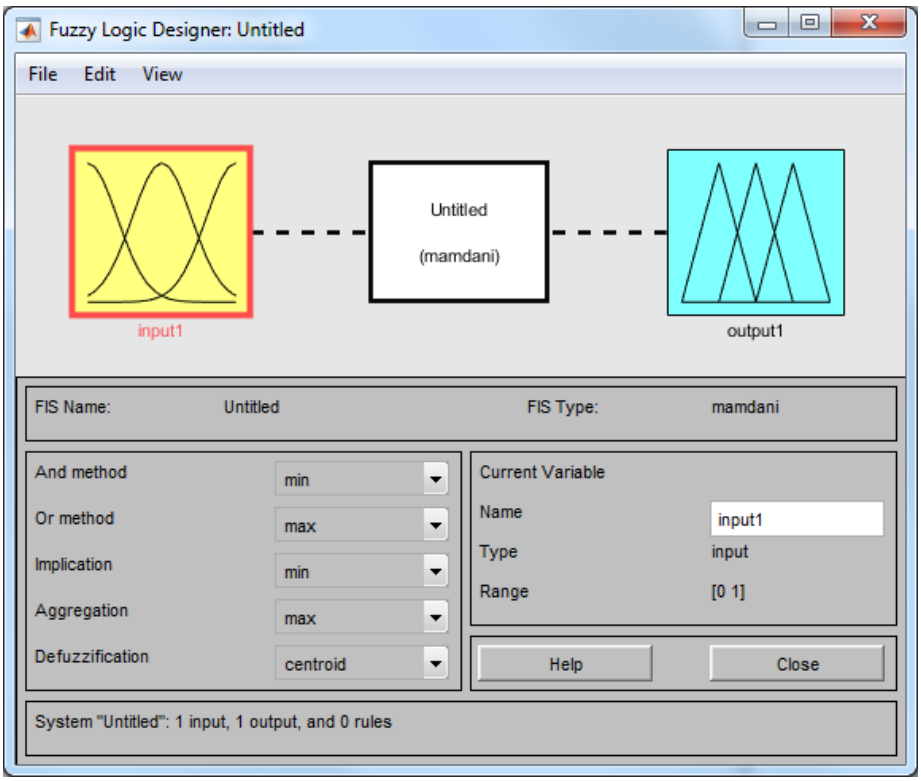

Рисунок 1.3 – Окно FIS-редактора

2. Добавить вторую входную переменную (Edit/AddVariable/Input).
3. Переименовать входные и выходную лингвистические переменные в поле Name FIS-редактора.
4. Сохранить проект (File/Export/To File).
5. Перейти в редактор функций принадлежности двойным щелчком левой кнопки мыши на блоке x1.

6. Задать диапазон изменения переменной x1 , напечатав -7 3 в поле Range и нажав Enter.

7. Задать функции принадлежности термов лингвистической переменной x1. Для лингвистической оценки этой переменной предлагается использовать три терма с треугольными функциями принадлежности. Задать наименования термов переменной («Низкий», «Средний», «Высокий»), выделив их функции принадлежности и отредактировав поле Name (рис. 1.4.).

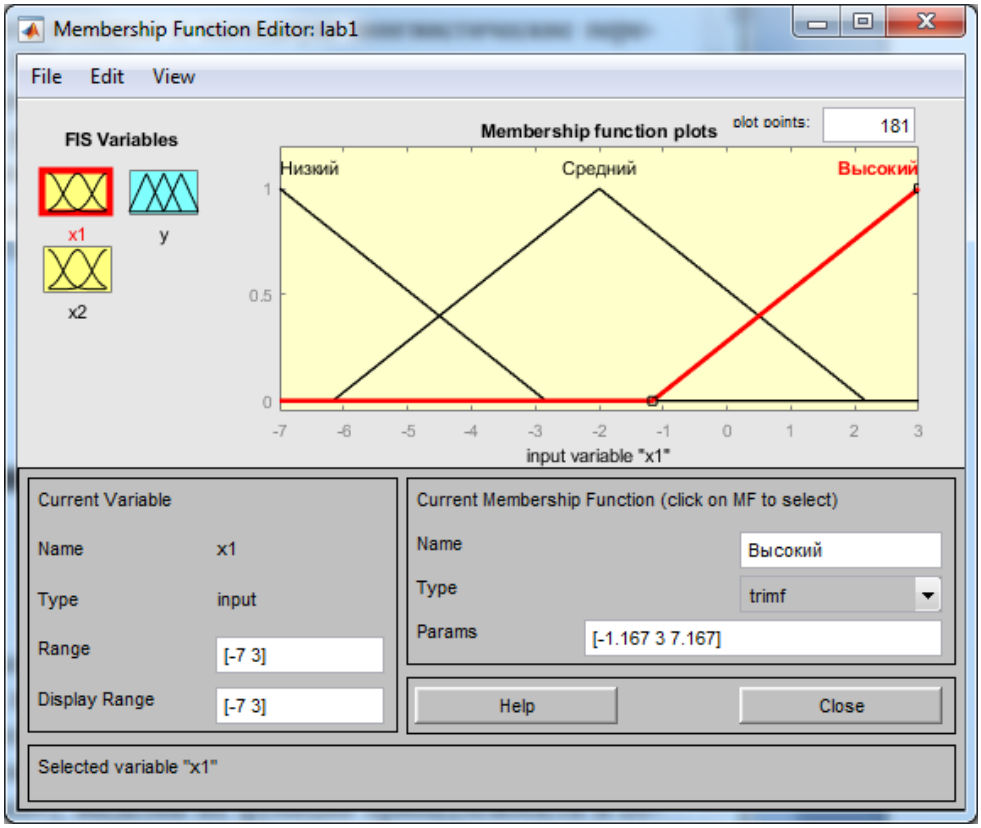

Рисунок 1.3 – Окно редактора функций принадлежности

8. Перейти в редакторе функций принадлежности к переменной x2. Проделать шаги 6-7 для переменной x2. Единственным отличием здесь будет диапазон.

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

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

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

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

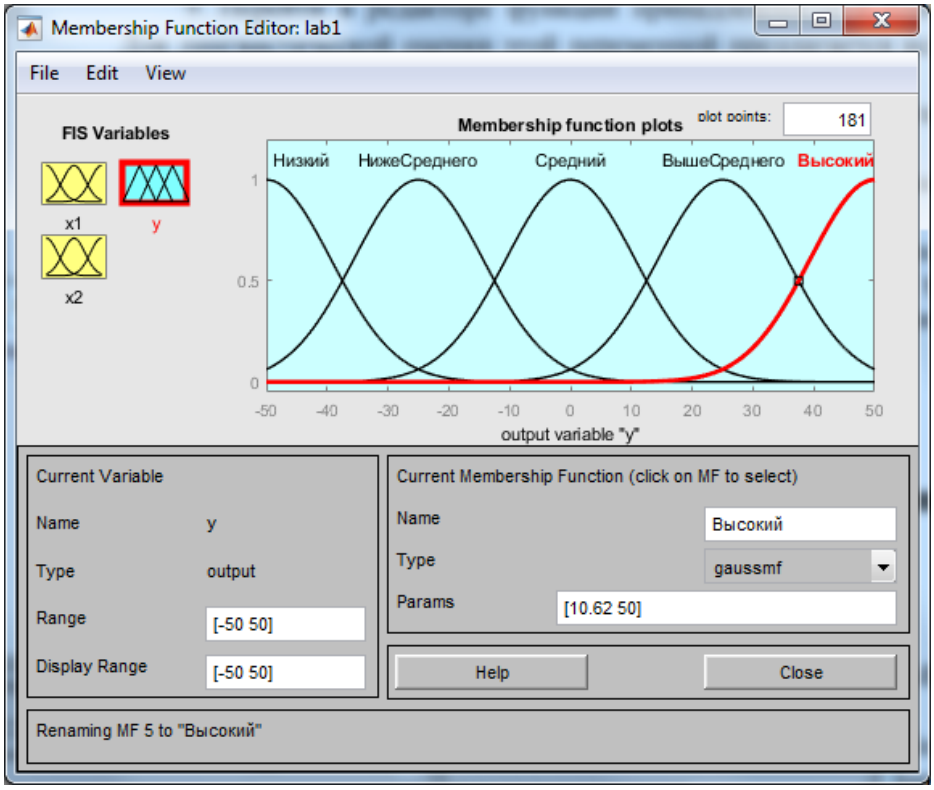

Рисунок 1.4 – Окно редактора функций принадлежности

13. Перейти в редактор базы знаний, выполнив Edit/Rules.

14. Для ввода правила необходимо выбрать в меню соответствующую комбинацию термов и нажать Add rule. Окно редактора правил представлено на рис. 1.5. В конце каждого правила в скобках указан весовой коэффициент.

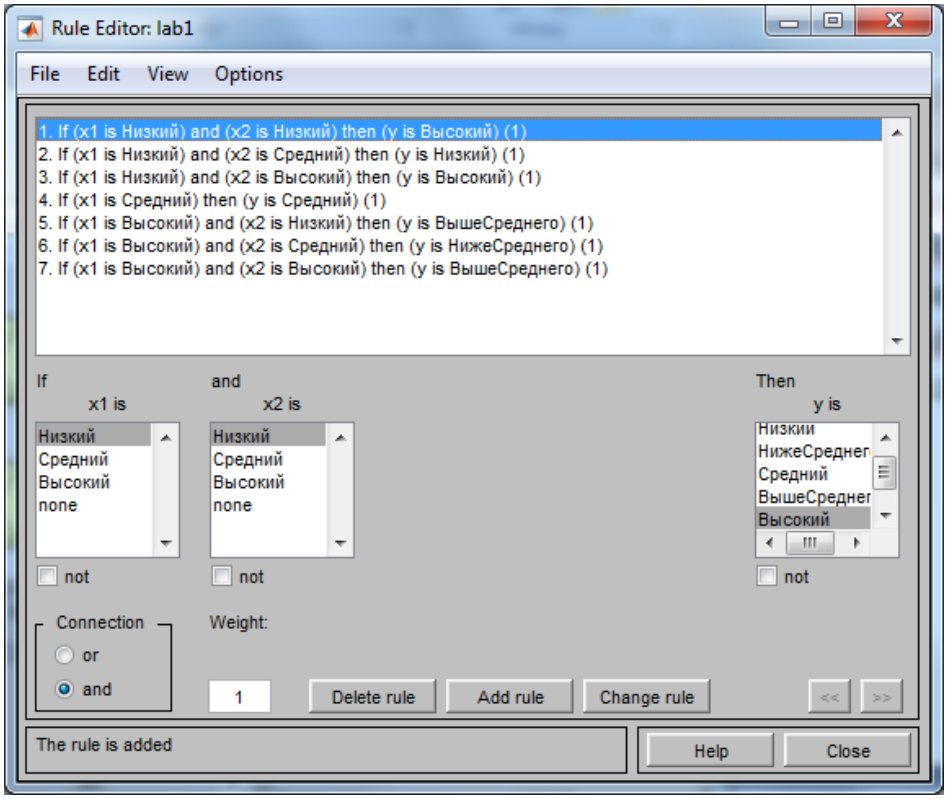

Рисунок 1.5 – Окно редактора правил

15. Сохранить проект (File/Export/To Disk).

16. Выполнить View/Rules. Появиться окно визуализации нечеткого вывода (рис. 1.6).

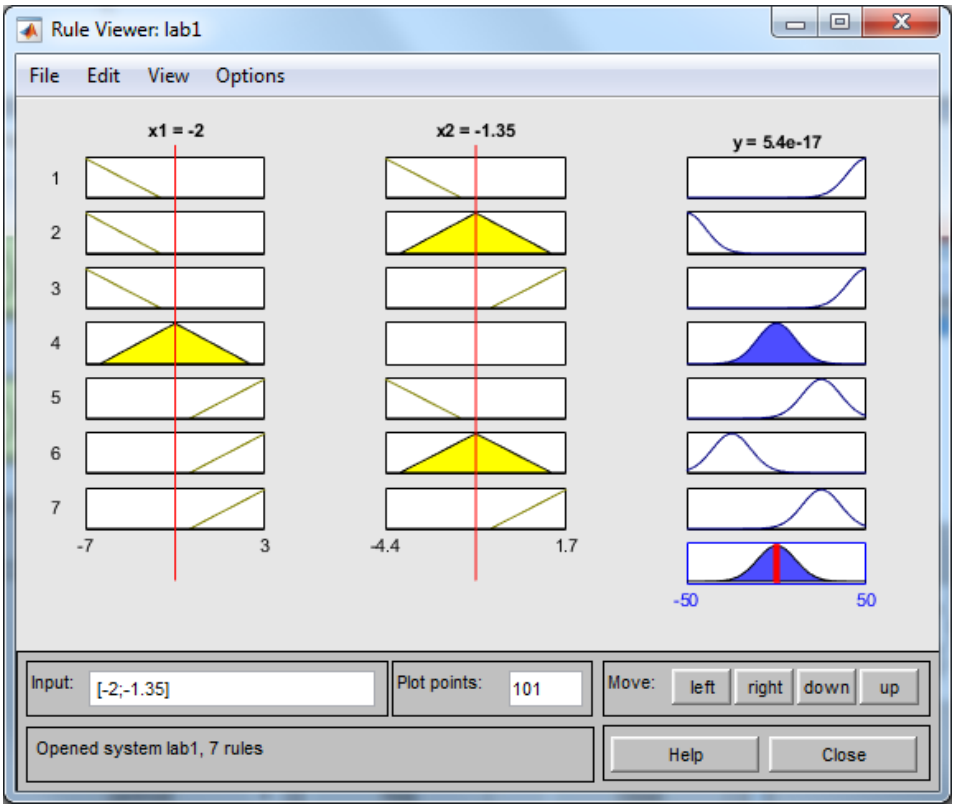

Рисунок 1.6 – Окно редактора правил

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

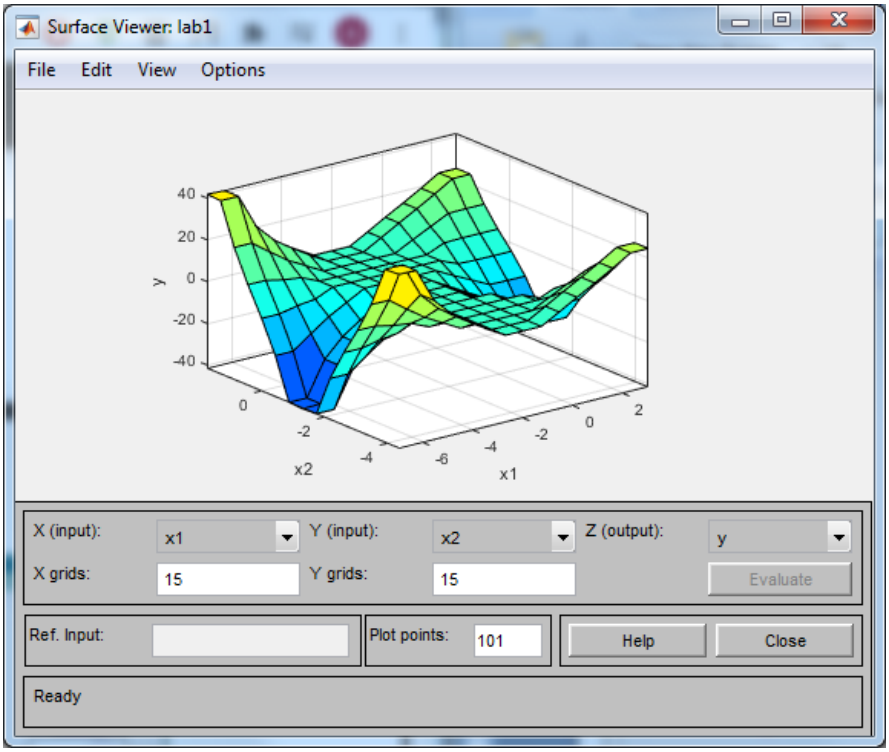

Рисунок 1.7 – Окно просмотра поверхности

Задания на самостоятельную работу

1. Изучить материал теоретической части.

2. Синтезировать систему нечеткого логического вывода для функции, предложенной в практической части.

3. Построить трехмерное изображение функции, использую программный код, представленный в первом пункте практической части.

4. Сравнить полученную поверхность с поверхностью «входы – выход», которая соответствует синтезированной нечеткой системе.

Содержание отчета

- цель работы,
- задание,
- практическая часть, включающая:

1. скриншоты типа рис. 1.2, 1.4, 1.6, 1.7 с пояснениями;

2. качественную оценку сравнения поверхностей, полученных разными способами;

3. выводы, наличие которых определено заданием.

Практическое занятие №2

АППРОКСИМАЦИЯ ГЛАДКОЙ ФУНКЦИИ С ПОМОЩЬЮ ОДНОСЛОЙНОЙ НЕЙРОННОЙ СЕТИ

Цель работы:

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

Краткие теоретические сведения

Нейронная сеть – это система, состоящая из многих простых вычислительных элементов, работающих параллельно, функция которых определяется структурой сети, силой взаимосвязанных связей, а вычисления производятся в самих элементах или узлах. Нейронная сеть – это набор нейронов, определенным образом связанных между собой. Работу искусственного нейрона можно описать следующим образом [1] (рис. 1).

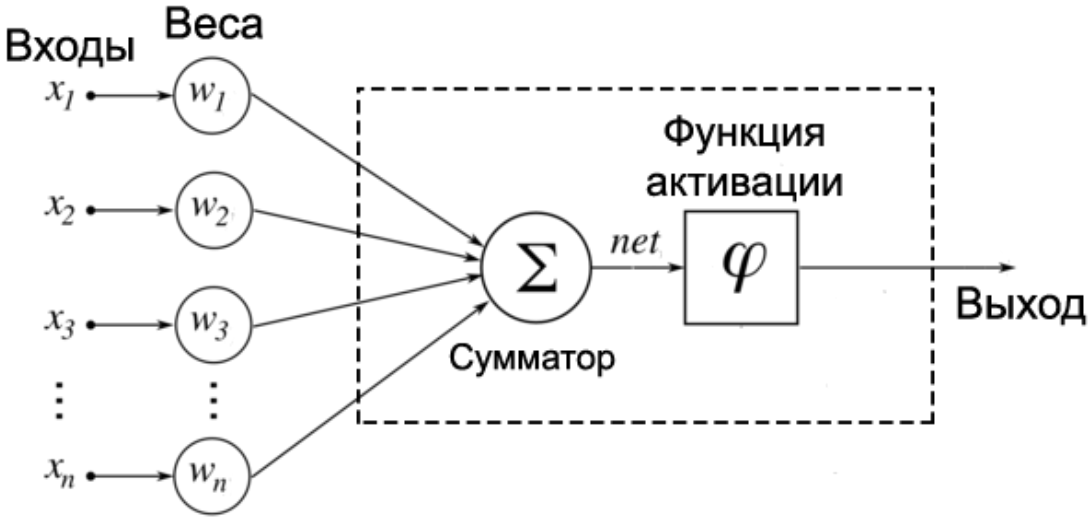

Рисунок 1 – Схема искусственного нейрона

Как видно на рисунке: у нейрона есть n входов x_i, у каждого из которого есть вес w_i, на который умножается сигнал, проходящий по связи. После этого взвешенные сигналы $x_i w_i$ направляются в сумматор, который агрегирует все сигналы во взвешенную сумму.

Эту сумму также называют *net*. Таким образом,

$$net = \sum_{i=1}^{i=n} w_i x_i = w^T x$$

Просто так передавать взвешенную сумму *net* на выход достаточно бессмысленно – нейрон должен ее как-то обработать и сформировать адекватный выходной сигнал. Для этих целей используют функцию активации, которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона. Функция активации обозначается $\phi(net)$. Таким образом, выходов искусственного нейрона является $\phi(net)$.

Для разных типов нейронов используют самые разные функции активации, но одними из самых популярных являются:

- Функция единичного скачка.
- Сигмоидальная функция.
- Гиперболический тангенс.
- Функция Rectified linear units (ReLU).

В рамках практического занятия будем рассматривать однослойную нейронную сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдает ответ.

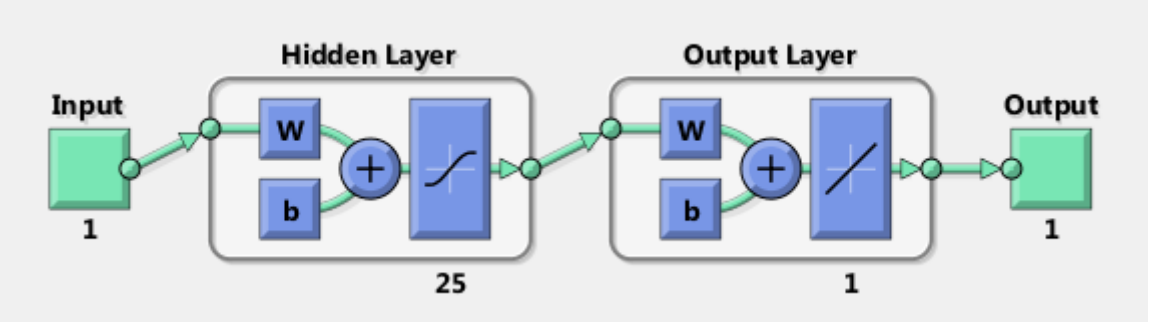

Рисунок 2 – Схема нейронной сети с одним скрытым слоем

Обучение нейронной сети – поиск такого набора весовых коэффициентов, при котором входной сигнал после прохода по сети преобразуется в нужный нам выходной.

Это определение «обучения нейронной сети» соответствует и биологическим нейросетям. Наш мозг состоит из огромного количества связанных друг с другом нейросетей, каждая из которых в отдельности

состоит из нейронов одного типа (с одинаковой функцией активации). Наш мозг обучается благодаря изменению синапсов — элементов, которые усиливают или ослабляют входной сигнал.

Если обучать сеть, используя только один входной сигнал, то сеть просто «запомнит правильный ответ», а как только мы подадим немного измененный сигнал, вместо правильного ответа получим бессмыслицу. Мы ждем от сети способности *обобщать* какие-то признаки и решать задачу на различных входных данных. Именно с этой целью и создаются *обучающие выборки*.

Обучающая выборка – конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит обучение сети.

После обучения сети, то есть когда сеть выдает корректные результаты для всех входных сигналов из обучающей выборки, ее можно использовать на практике. Однако прежде чем сразу использовать нейронную сеть, обычно производят оценку качества ее работы на так называемой *тестовой* выборке.

Тестовая выборка – конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит оценка качества работы сети.

Выполнение работы

Рассмотрим пример использования нейронной сети для аппроксимации данных. Данные могут быть получены, например, в ходе математического моделирования. Обучившаяся по результатам математического моделирования, нейронная сеть сможет мгновенно выдавать результаты, которые могут использоваться, например, для управления приводами робота или при фильтрации показаний с датчиков измерительной системы. Также нейронная сеть может аппроксимировать данные полученные в ходе натурных испытаний, чтобы затем использовать при математическом моделировании. При этом даже при малом исходном количестве данных можно получить достаточную точность аппроксимации. Часто такой способ гораздо удобнее, чем использование кусочно-полиномиальных функций аппроксимации.

Важным этапом при работе с нейронными сетями является получение обучающей и тестовой выборки, подготовка и анализ входных данных.

В данной практической работе в качестве исходного объекта возьмем сложную степенную функцию (выбирается согласно варианту задания):

$$y(x) = 2.5 \cdot x^{\sin(0.5x)}$$

График данной функции в диапазоне значений $x \in (-5,\ 5]$,будет иметь вид, показанный на рис. 3.

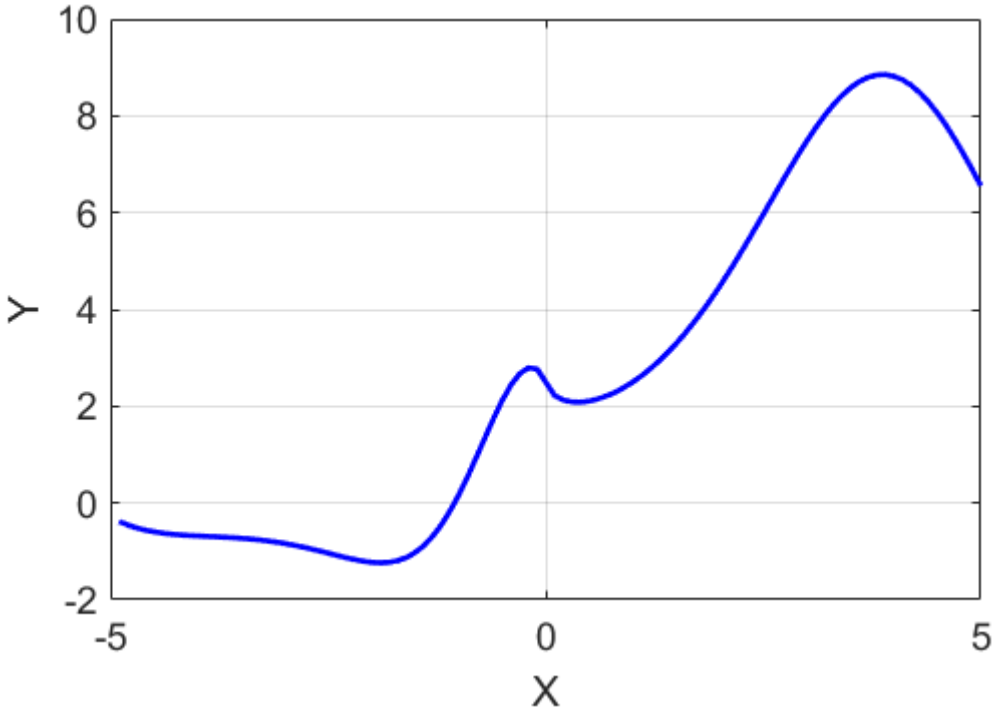

Рисунок 3 – График исходной функции

Как видно график имеет достаточно сложную структуру и аппроксимация его полиномиальными функциями представляет собой сложную задачу.

Используя скрипт, получим массив из 100 точек x и y для указанного диапазона. Для создания нового скрипта в Matlab, необходимо выбрать соответствующую кнопку «new script» в основном окне Matlab, после чего в открывшемся новом окне m-файла необходимо набрать следующий код и сохранить его в новой папке (куда далее и будет сохранена выборка и функция нейронной сети):

```
Xmin = -5;
Xmax = 5;
dx = 0.1; % шаг, определяющий количество точек
count = (Xmax - Xmin)/dx; % количество точек

result.X = zeros(count, 1); % массив значений x
result.Y = zeros(count, 1); % массив значений y

fileX = fopen('X.txt', 'w'); % открытие текстовых файлов
fileY = fopen('Y.txt', 'w'); % для записи массивов значений

for i = 1:count
    x = Xmin + i*dx;
    y = 2.5*x^sin(0.5*x);   %искомая функция
    result.X(i) = x;        % сохранение данных в массив
    result.Y(i) = y;
    fprintf(fileX,'%4.2f \n',x); % сохранение данных в файл
    fprintf(fileY,'%4.2f \n',y);

end
fclose(fileX); % закрытие текстовых файлов
fclose(fileY);

% выведение результатов на график
figure('Color','w')
plot(result.X, result.Y, 'Color', 'b', 'LineWidth', 2);
set(gca,'YDir','normal','FontSize',14,'XGrid','on','YGrid','on');
grid on
xlabel('X');
ylabel('Y');
```

Обратим внимание, что при сохранении данных используется ранее созданные файлы `'X.txt'` и `'Y.txt'`. При использовании в качестве генератора выборки Matlab – сохранять файлы не обязательно, можно обращаться непосредственно к массиву, однако, данный функционал пригодится при работе, например, с данными полученными в ходе экспериментов.

Для удобства работы с нейронными сетями в Matlab есть специализированный инструмент, для начала работы с кororым необходимо ввести в командной строке: `nnstart`, после чего запустится помощник, внешний вид окна которого показан на рис. 4.

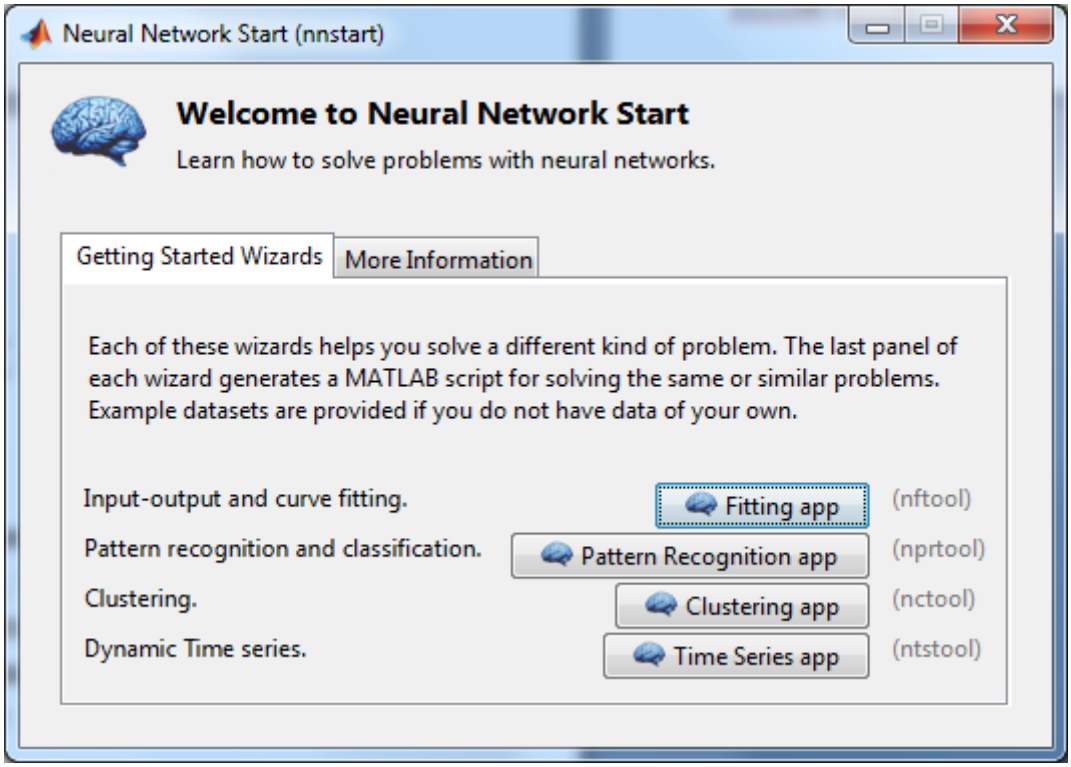

Рисунок 4 – Стартовое окно помощника

Здесь доступны сразу несколько инструментов, в рамках настоящей работы будем работать с «Fitting app», наиболее простым и подходящим для задачи аппроксимации инструментом.

После выбора Fitting app появляется окно с описанием инструмента, после ознакомления нажимаем «Next». Далее появляется окно выбора исходных данных

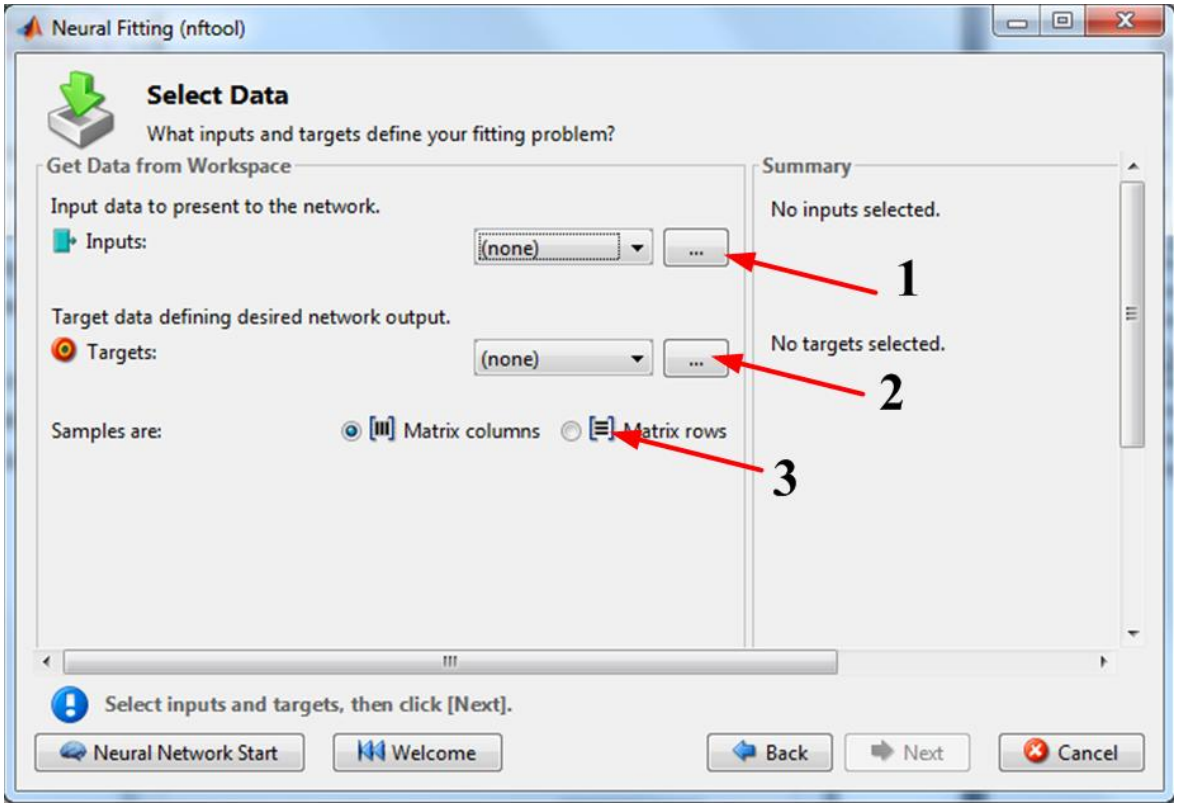

Рисунок 5 – Окно выбора данных

В данном окне нажав на кнопку 1 (рис.5), необходимо указать файл с входными данными `'X.txt'`. Нажав на кнопку 2, указываем данные выходные `'Y.txt'`. После чего программе необходимо указать, что данные у нас представлены в виде столбца (именно так данные сохранялись при использовании скрипта, приведенного ранее). Обратите внимание, что при указании файла с данными Matlab предлагает выбрать тип разделителя (запятая, пробел и т.д.) в нашем случае, ничего менять не требуется.

После нажатия кнопки «Next» появляется окно настройки обучающей выборки.

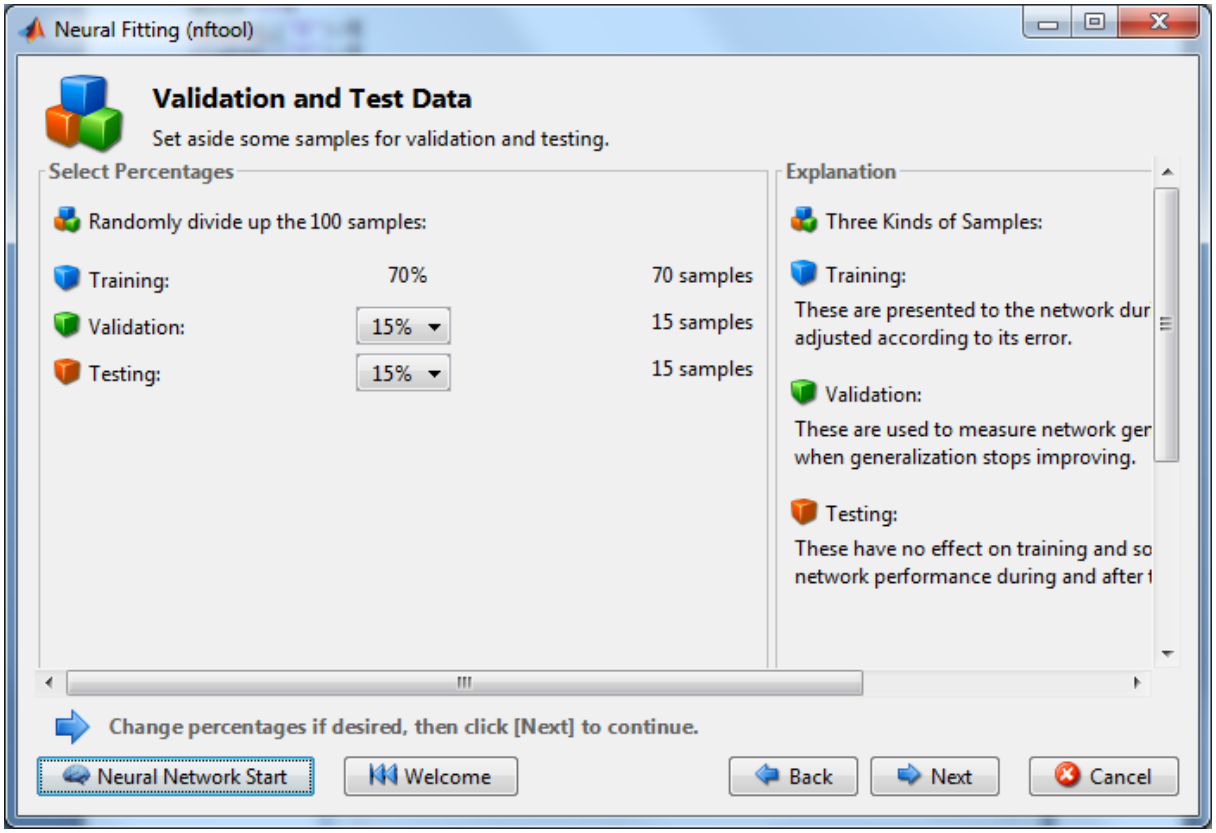

Рисунок 6 – Окно настройки обучающей выборки

Здесь весь объем данных предлагается случайным образом разбить на 3 отдельные части:

- **Training** – эти данные будут использоваться сетью во время обучения
- **Validation** – эти данные будут использоваться для оценки результатов работы сети и остановки обучения, когда ошибка перестает уменьшаться.
- **Testing** - эти данные не влияют на обучение и поэтому обеспечивают независимую оценку производительности сети во время и после обучения.

В рамках работы, оставим значения по умолчанию и нажимаем «Next».

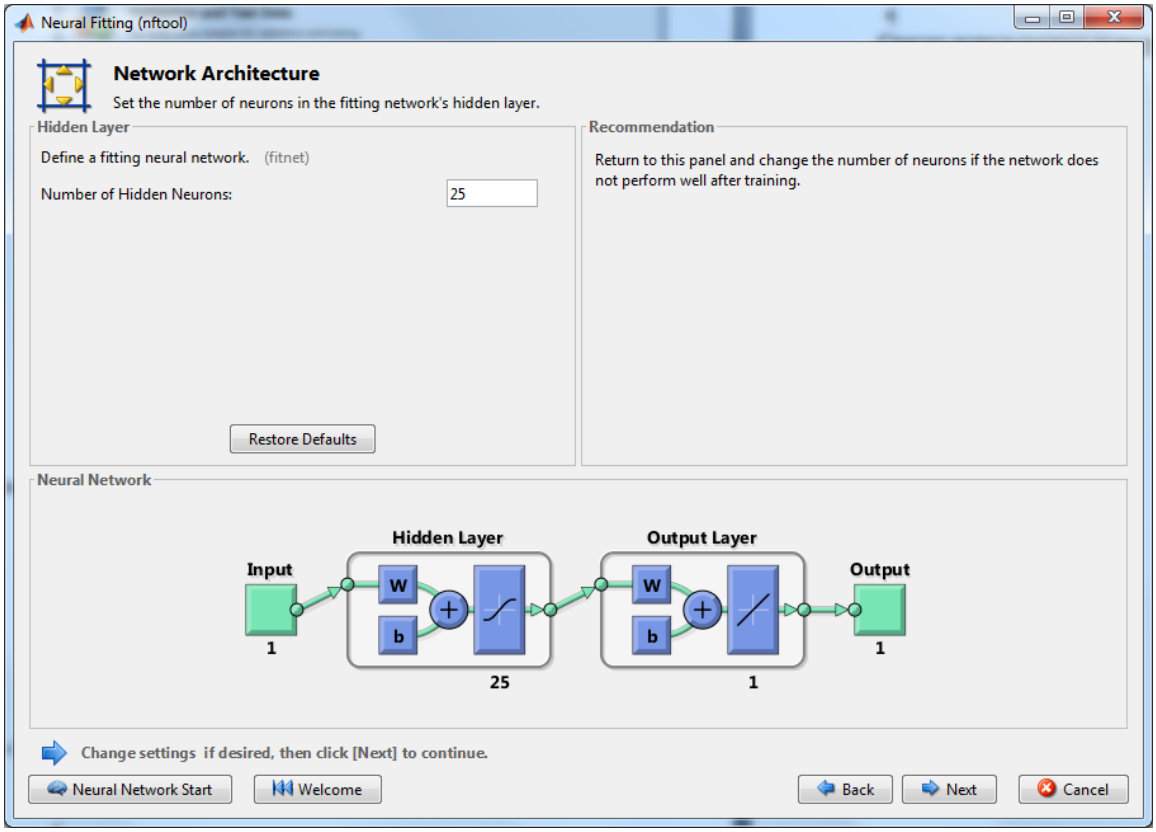

Рисунок 7 – Окно настройки нейронной сети

В появившемся окне необходимо указать количество нейронов скрытого слоя, выставляем значение 25 (увеличение количества нейронов всегда приводит к увеличению времени обучения, но не всегда к повышению точности). После нажатия кнопки далее, Matlab сгенерирует сеть и предложит ее обучить одним из трех доступных способов, описание методов также представлено в этом окне.

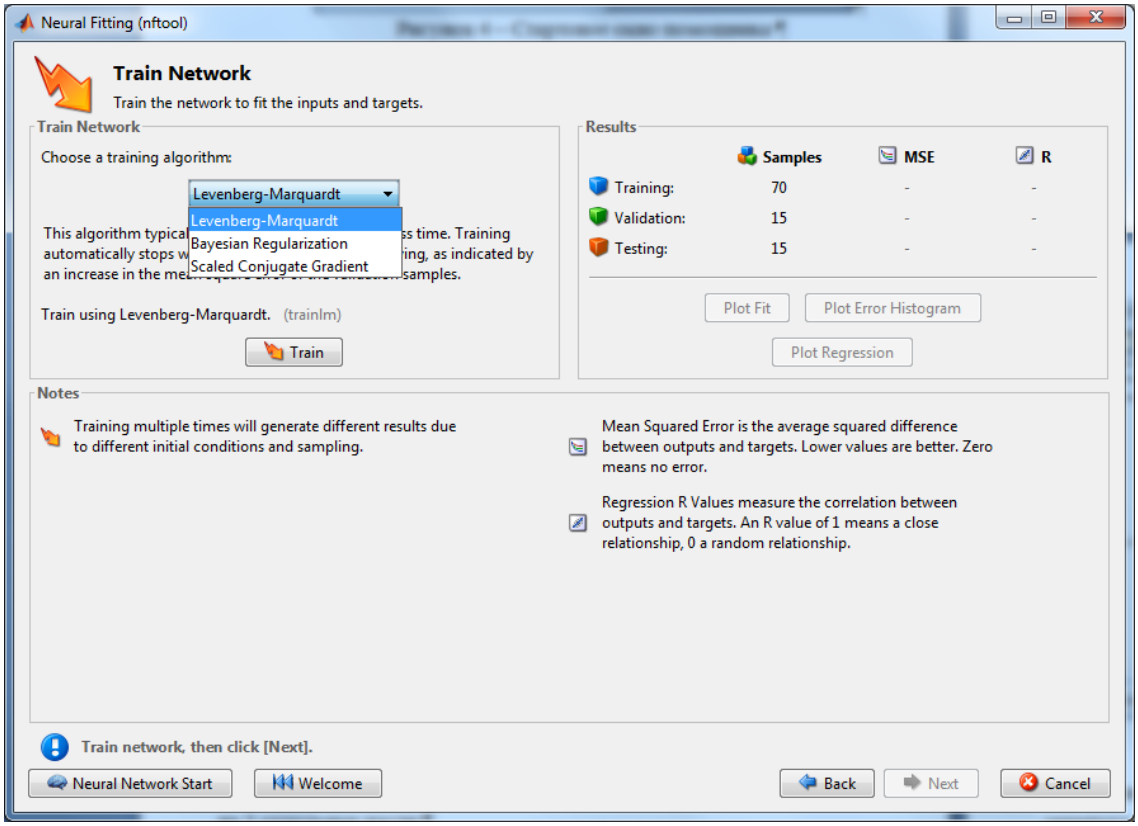

Рисунок 8 – Окно настройки обучения

При выполнении работы студенту предлагается найти оптимальный метод и количество нейронов в скрытом слое для обеспечения наилучшей точности нейронной сети.

В данном примере выберем первый метод «Levenberg-Marquardt» и нажмем кнопку «Train».

В процессе обучения появляется окно, показанное на рис. 9.

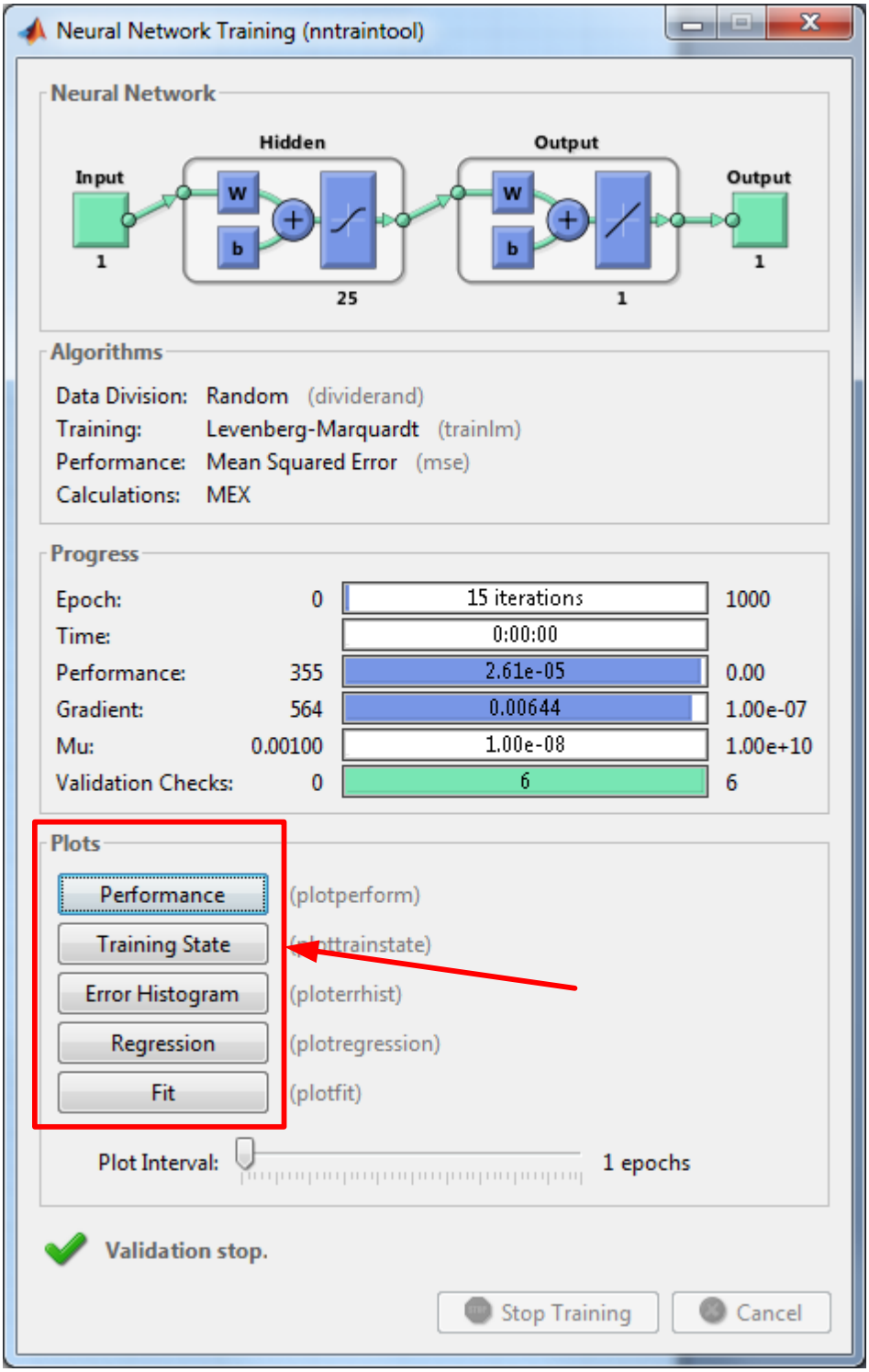

Рисунок 9 – Ход и результаты обучения

В данном окне показывается ход обучения. Также воспользовавшись кнопками в нижней части окна можно показать графики характеризующие результирующую точность, точность, ошибки обучения и т.д.

Далее вернемся к окну Matlab, где в зависимости от результатов анализа обучения мы можем вернуться к настройке параметров сети,

если результаты обучения нас не удовлетворили. В нашем примере мы дважды нажмем «Next» и попадем в окно сохранения нейронной сети.

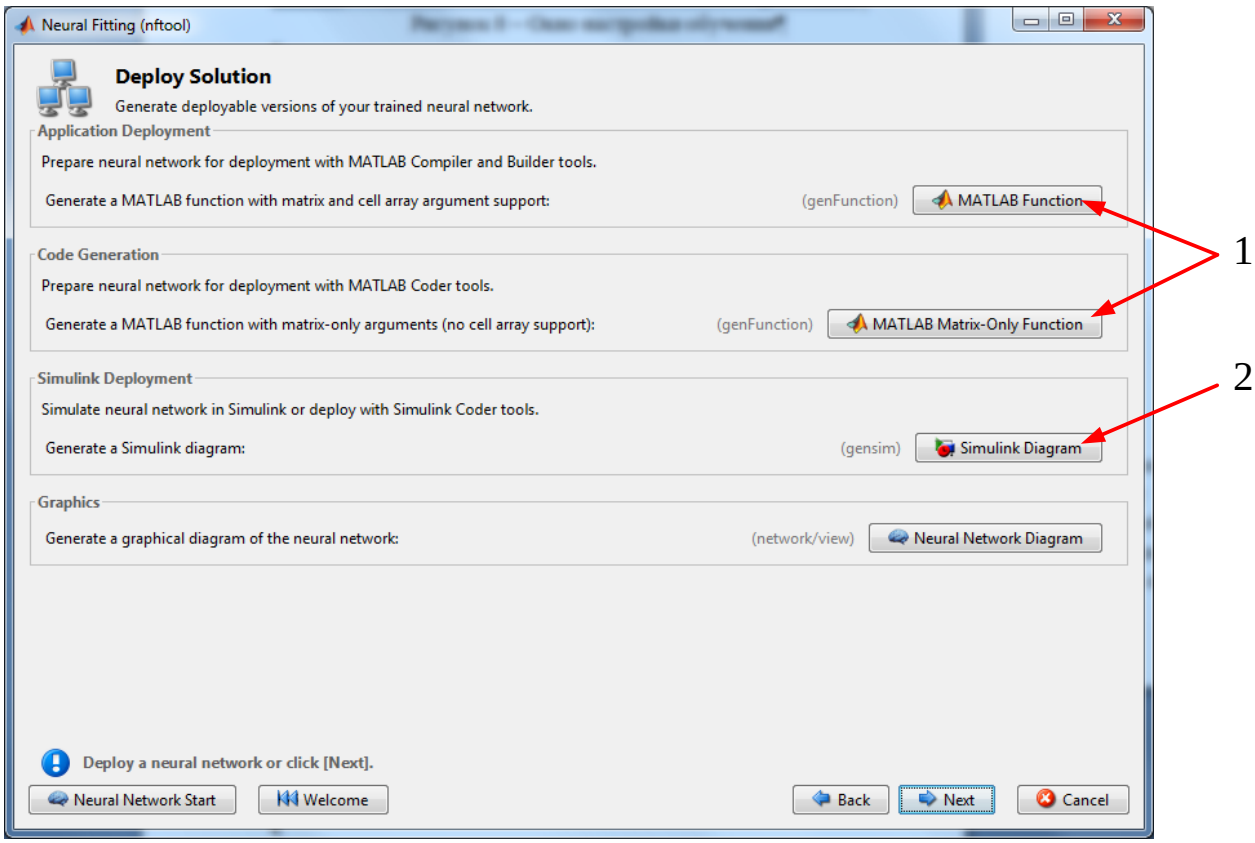

Рисунок 10 – Экспорт нейронной сети

Здесь нам предлагают сохранить нейронную сеть в виде функций (1, рис.10), которые мы можем использовать далее, либо в виде блока Simulimk, который можно использовать при моделировании систем управления или движения объекта.

В данном примере нажмем на верхнюю кнопку и сохраним сгенерированную функцию, в папку с проектом под названием «NetworkFunction». Обратите внимание что в самом файле функции, также необходимо сменить название, первая строчка скрипта будет иметь вид:

```
function [y1] = NetworkFunction(x1)
```

Затем отредактировав текст первоначального скрипта, выведем одновременно график, полученный напрямую по формуле и с применением нейронной сети.

```
Xmin = -5;
Xmax = 5;
dx = 0.1;
count = (Xmax - Xmin)/dx;

result.X = zeros(count, 1);
result.Y = zeros(count, 1);
result.NY = zeros(count, 1);

for i = 1:count
    x = Xmin + i*dx;

    y = 2.5*x^sin(0.5*x);
    yNet = NetworkFunction(x);

    result.X(i) = x;
    result.Y(i) = y;
    result.NY(i) = yNet;
end

figure('Color','w')
plot(result.X, result.Y, 'Color', 'b', 'LineWidth', 2); hold on;
plot(result.X, result.NY, 'Color', 'r', 'LineWidth', 2);
set(gca,'YDir','normal','FontSize',14,'XGrid','on','YGrid','on');
grid on
xlabel('X');
ylabel('Y');
```

При выполнении данный скрипт покажет график.

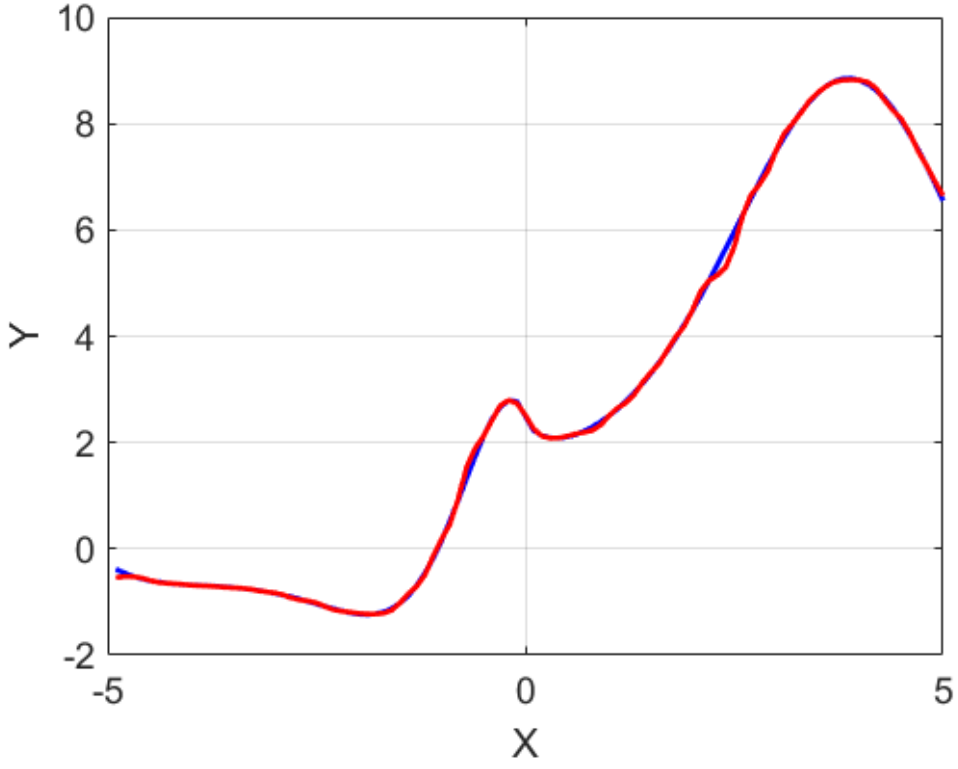

Рисунок 11 – Графики исходной (синий) и построенной с помощью нейронной сети (красный) функций

Как видно из результатов построения нейронная сеть достаточно точно повторяет контуры исходной функции, однако, присутствуют и ошибки аппроксимации.

Для повышения точности аппроксимации, вернемся к настройки сети и поменяем количество слоев на 30, а метод обучения на «Bayesian Regularization»

После выполнения всех предыдущих пунктов, результат построения будет иметь вид, показанный на рис. 12.

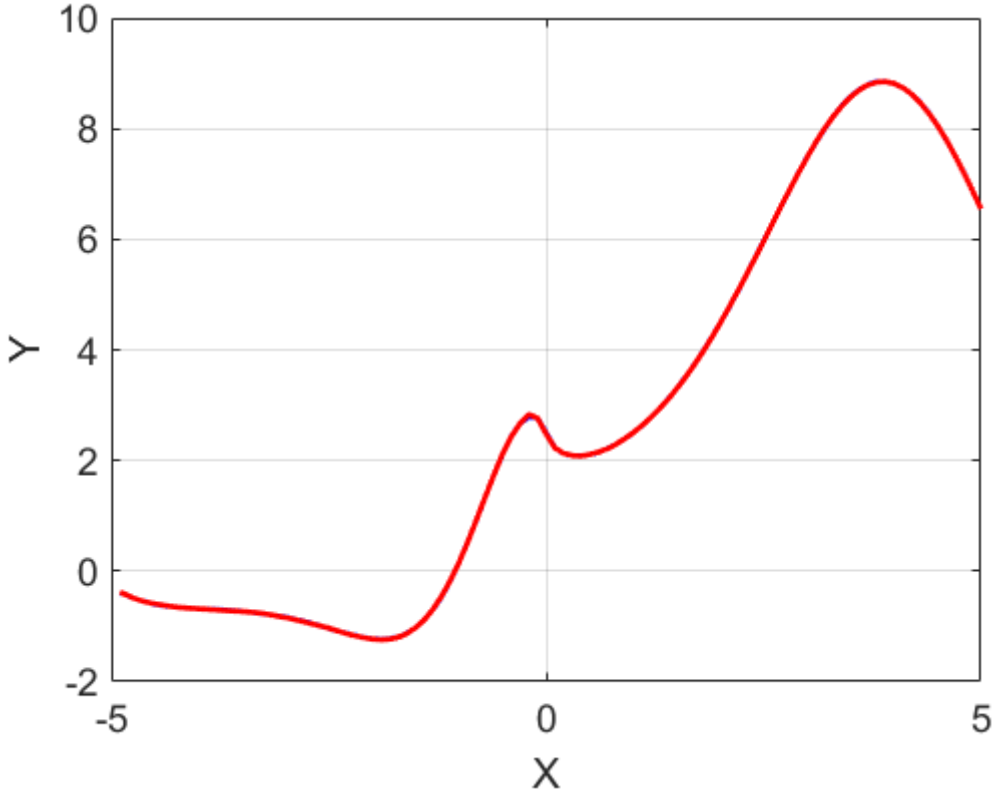

Рисунок 12 – Графики исходной (синий) и построенной с помощью нейронной сети (красный) функций

На данном рисунке, красный график почти полностью закрывает синий, что говорит о высокой точности полученного решения.

В ходе практической работы студенту предлагается поэкспериментировать с параметрами сети и обучения самостоятельно.

Задания для практических работ

Вариант по списку	Исходная функция
1	`y = cos(10)*x^sin(0.5*x)`
2	`y = 3*cos(0.1*x)^sin(0.5*x)`
3	`y = 0.25*x + 3*cos(x)^3`
4	`y = 0.25*x^2.4 - 1.2*sin(2*x^0.9)`
5	`y = 0.3*x^0.3*x + 0.5*cos(1.2*x)`
6	`y = 0.5*x^1.2*cos(x)`
7	`y = cos(5)*x^sin(0.7*x)`
8	`y = cos(3.2)*2*x^cos(0.65*x)`
9	`y = cos(1.2*x)*x^cos(0.1*x)`
10	`y = sin(0.8*x)*x^sin(0.5*x)`
11	`y = 0.5*cos(0.2*x)*x^sin(0.3*x)`
12	`y = 2.5*sin(0.2*x)*x^cos(0.6*x)`

Задание на самостоятельную работу

1. Изучить материал теоретической части
2. Сгенерировать файлы обучающей выборки
3. Настроить нейронную сеть и обучить с использование обучающей выборки
3. Построить графики и выполнить сравнительный анализ полученных результатов. Сделать вывод

Содержание отчета

- цель работы,
- задание,
- практическую часть,
- выводы.

Список используемых источников

1.Системы искусственного интеллекта. Практический курс [Текст] : учебное пособие / ред. И. Ф. Астахова. - М. : БИНОМ. Лаборатория знаний, 2008. - 292 с.

2. Емельянов, С. Г. Интеллектуальные системы на основе нечеткой логики и мягких арифметических операций [Текст] : учебник / С. Г. Емельянов, В. С. Титов, М. В. Бобырь. - Москва : Аргамак-Медиа, 2014. - 338с.

3. Интеллектуальные системы [Электронный ресурс] : учебное пособие / А. Семенов [и др.]. - Оренбург : ОГУ, 2013. - 236 с. – Режим доступа : http://biblioclub.ru/index.php?page=book_red&id=259148&sr=1

4. Сидоркина, И. Г. Системы искусственного интеллекта [Текст] : учебное пособие / И. Г. Сидоркина. – Москва : КНОРУС, 2016. - 246 с.

5. Финн, В. К. Искусственный интеллект [Текст]: методология, применения, философия / науч. ред. М. А. Михеенкова; Российская академия наук, Всероссийский институт научной и технической информации. - М. : Красанд, 2011. - 448 с.

6. Рассел, С. Искусственный интеллект. Современный подход [Текст] / С. Рассел, Т. Норвиг – 2-е изд. – М. : Вильямс, 2006. – 1408 с.

7. Ясницкий, Л. Н. Введение в искусственный интеллект [Текст]: учебное пособие / Л. Н. Ясницкий; М.: Академия, 2005. – 176 с.

8. Смолин, Д. В. Введение в искусственный интеллект [Электронный ресурс] : конспект лекций / Д. В. Смолин. - 2-е изд., перераб. - М. : Физматлит, 2007. - 292 с. – Режим доступа : http://biblioclub.ru/index.php?page=book_red&id=76617&sr=1