МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет Автоматизация, мехатроника и управление

Кафедра Автоматизация производственных процессов

Методические указания к выполнение контрольной работы по дисциплине «Нейросетевые алгоритмы обработки данных в системах управления» по направлению 15.04.04 Автоматизация технологических процессов и производств

Ростов-на-Дону
2022 г.

УДК 62-192

Составитель: доцент Лапшин В.П.

Методические указания к выполнению контрольной работы по дисциплине «Нейросетевые алгоритмы обработки данных в системах управления» предназначены для студентов заочной формы обучения по направлению подготовки 15.04.04 «Автоматизация технологических процессов и производств» профиль «Интеллектуальные системы сбора и анализа больших данных».

УДК 62-192

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Функциональный аппарат Matlab позволяет создание с помощью него нейронных сетей. В данной работе будет создана двуслойная нейросеть и рассчитана суммарная ошибка её вычислений, произведена минимизация этой суммарной ошибки. Прежде всего введём терминологию, которой будем пользоваться во время работы.

Термином «искусственная нейронная сеть» или просто «нейронная сеть» часто обозначают математическую модель, очень упрощённо описывающую общий алгоритм работы человеческой нервной системы. Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передаёт её дальше. Нейроны группируются в последовательные слои. Выходные сигналы одного слоя поступают на рецепторы следующего слоя и так далее.

Суммарная ошибка вычисляется как разность между ожидаемым значением «Yt» (из обучающего набора) и полученным значением «Y» (посчитанное на этапе прямого распространения ошибки).

Градиент — это вектор, который определяет крутизну склона графика функции и указывает его направление относительно какой либо из точек на графике. Чтобы найти градиент, нужно взять производную от графика по данной точке. Градиент позволит найти вес, при которой значение суммарной ошибки будет минимальным.

Количество входных параметров и нейронов выбирается по двум последним цифрам зачетки, предпоследняя цифра обозначает количество входов, а последняя количество нейронов, если одна из этих цифр равна нулю, то вместо этой цифры используется 1.

Для примера работы выбрана система с 8 входами и 4 нейронами. Входные параметры представляют собой блоки типа Constant, где в разделе Constant value прописывается номер блока: x1, x2 и так далее.

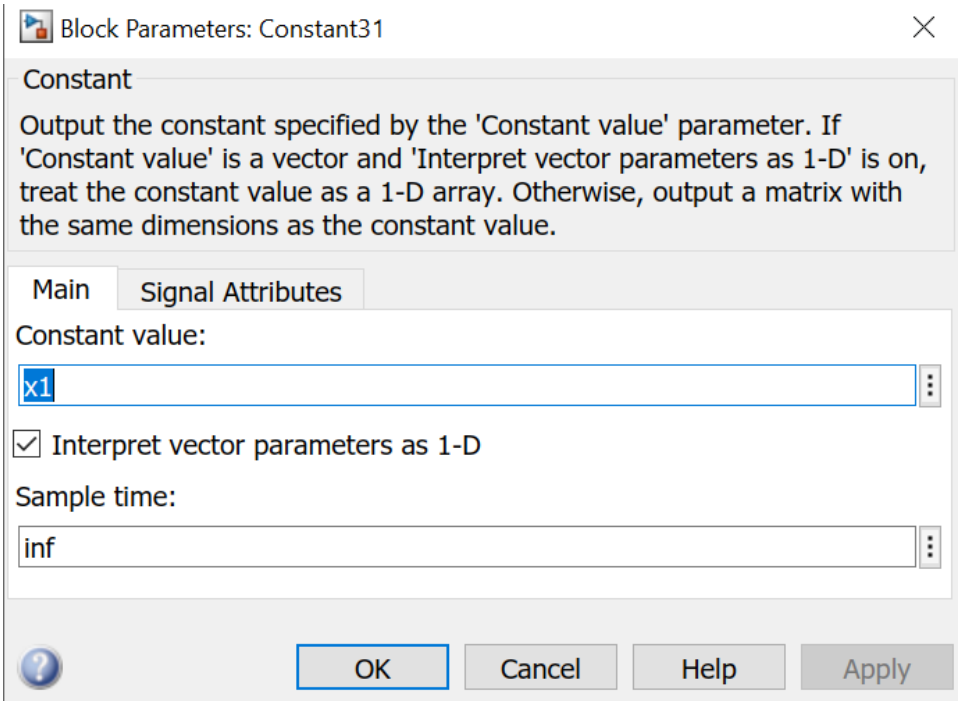

Рисунок 1 – Параметры блоков Constant

В качестве нейронов использованы блоки типа Mux. В его параметрах в раздел Number of inputs («Число входов») вписываем число входов. У нас их 8. Дублируем блоки Mux, чтобы их стало 4 (по числу нейронов).

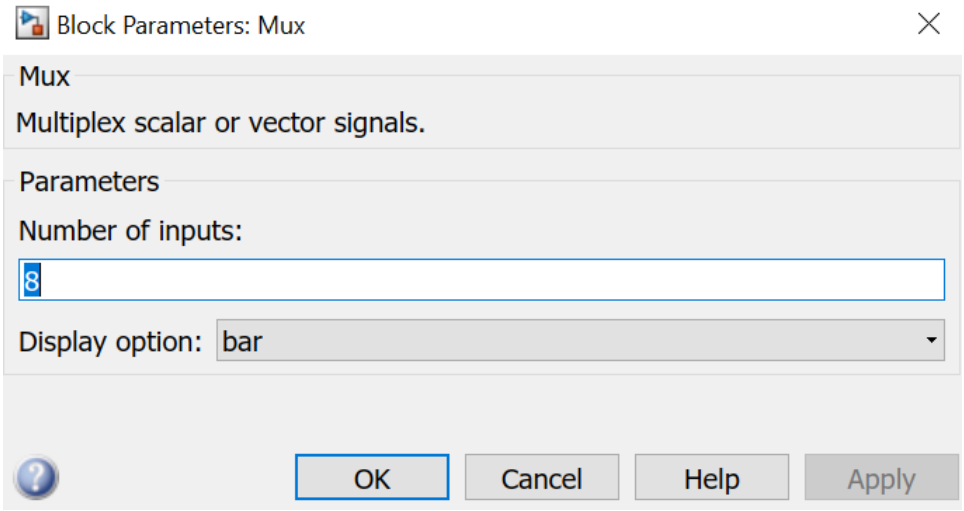

Рисунок 2 – Параметры блоков Mux

Напротив каждого блока Mux ставим блок Gain. В параметры каждого блока пишем его вес по формуле: **название матрицы весов (номер входа; номер нейрона).** На рисунке 3 приведены параметры блока Gain: он соединён с первым входом и вторым нейроном.

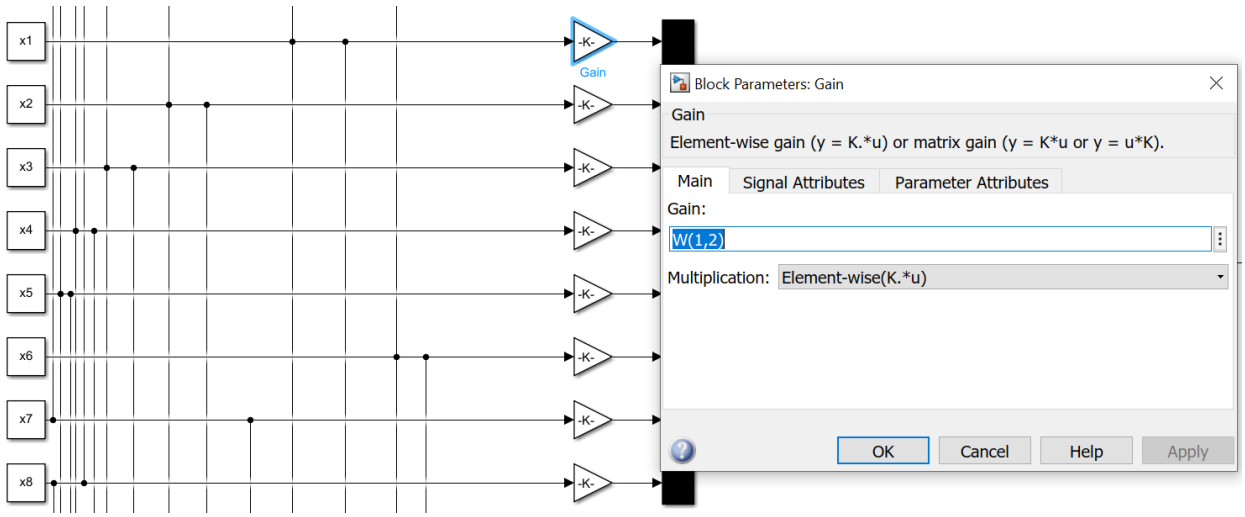

Рисунок 3 - Параметры блока Gain

Установив блоки Gain напротив каждого входа в нейрон, соединяем их со входами. Общий вид соединений показан на рисунке 4.

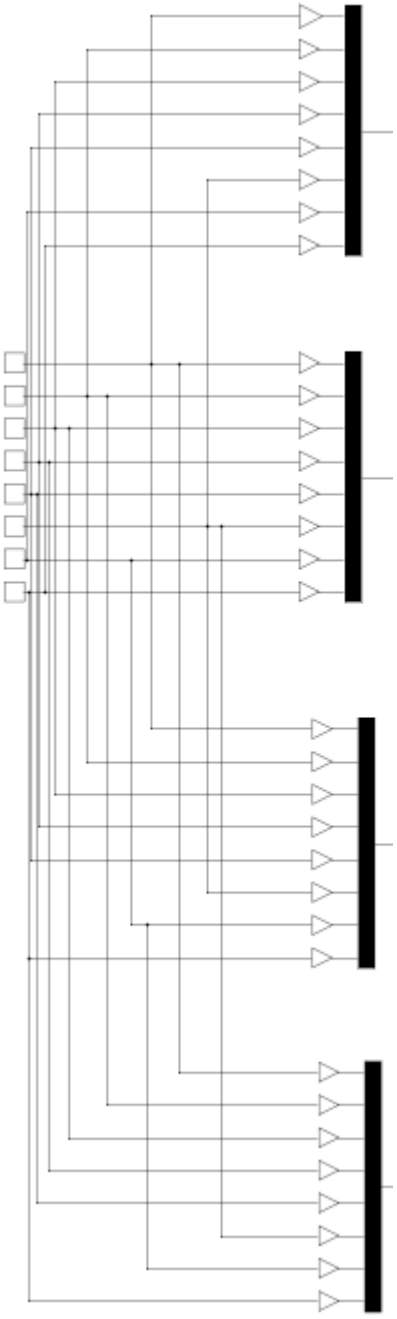

Рисунок 4 – Общий вид соединений

На выходе из каждого нейрона устанавливаем синапсы (сигмоиды). Каждый синапс представляет собой подсистему, состоящую из блоков Constant со значением 1, Gain со значением 1, Math Function с параметром «exp», блоков Add (с параметрами ++) и Divide (с параметрами */). После настройки и соединения всех блоков подсистема должна иметь вид, показанный на рисунке 5.

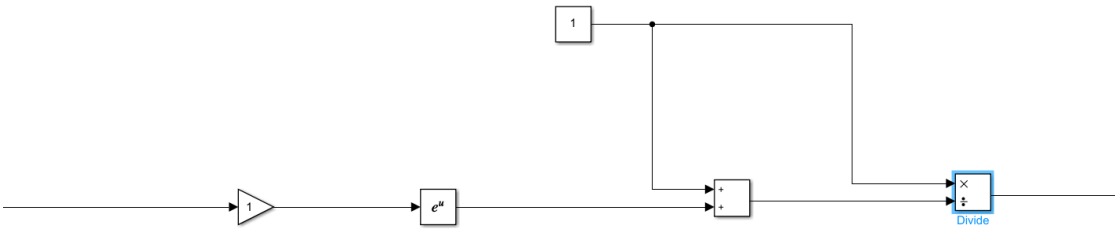

Рисунок 5 – Подсистема

Из получившейся системы можно создать подсистему двумя способами: либо выделить необходимое, нажать правую кнопку мыши и выбрать пункт «Create Subsystem from Selection», либо выделить и в разделе «Edit» выбрать «Create Subsystem». Система «сворачивается» в прямоугольник, нажав на который, можно рассмотреть комплектующие подсистемы. Копируем подсистемы и расставляем напротив каждого нейрона.

После выполнения всех операций схема должна иметь вид, подобный приведённому на рисунке 6.

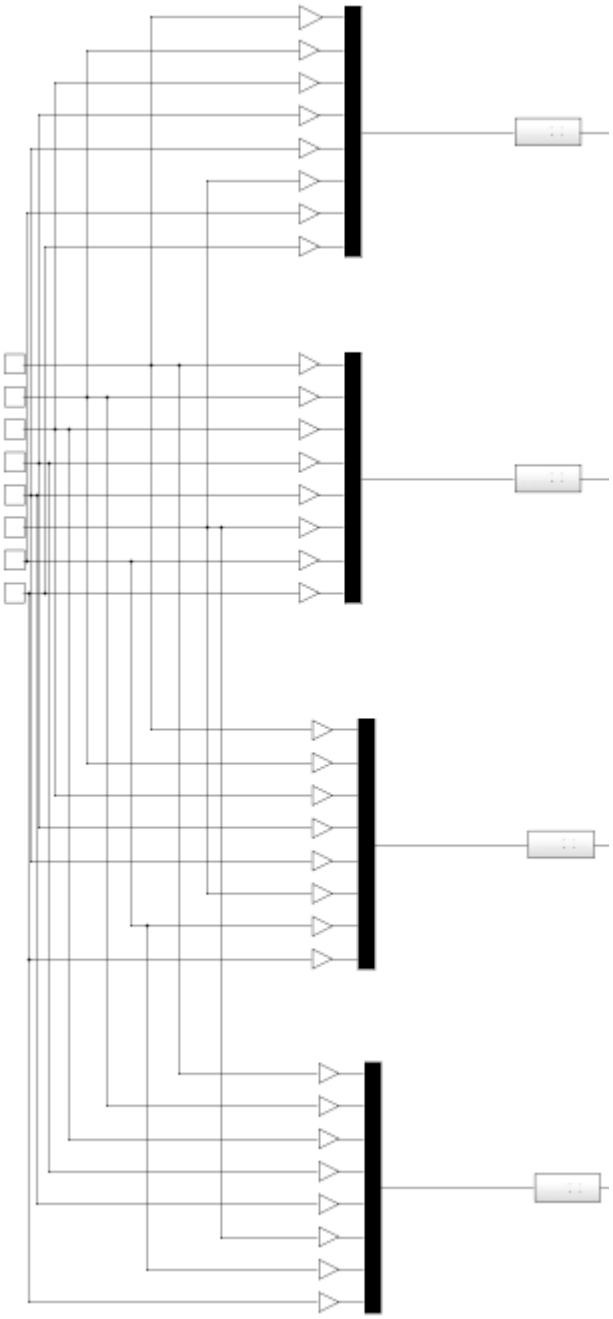

Рисунок 6 – Схема после создания синапсов

После каждой подсистемы устанавливаем блок Gain с новой нумерацией параметров. Это веса новой матрицы, для которой наши 4 первые нейрона будут являться входами, а 5й нейрон будет являться единственным выходом. Этой матрице мы дадим новое название, в примере это h. На рисунке 7 показан пример настройки веса на выходе из нейрона №4.

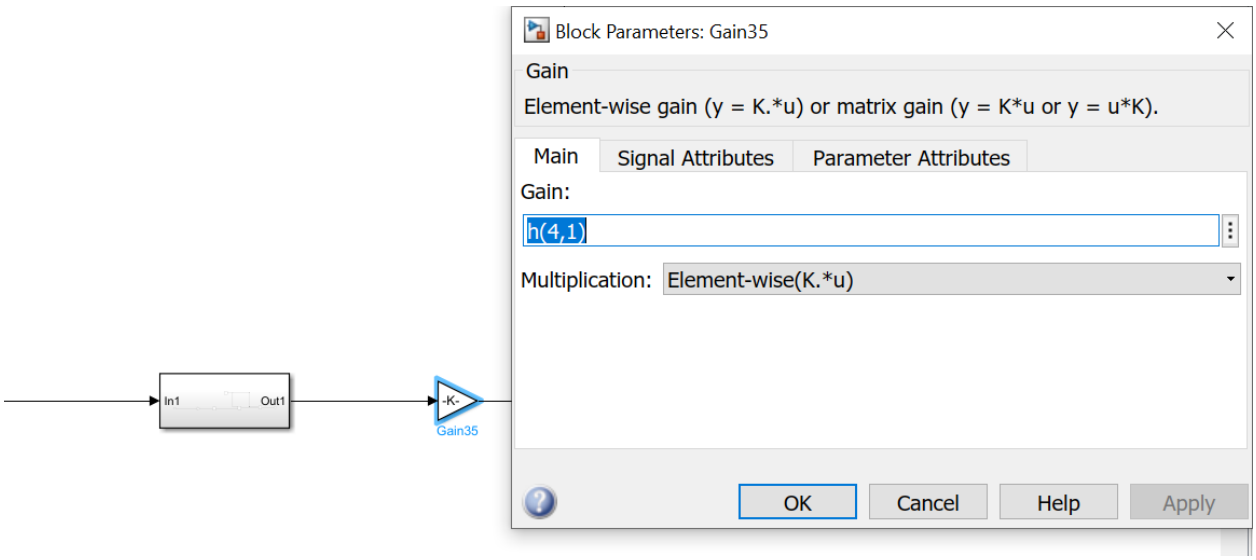

Рисунок 7 – Настройка весов для новой матрицы

После каждого блока Gain ставим блок Display, куда будут выводиться параметры. Настройки каждого дисплея должны иметь вид, указанный на рисунке 8.

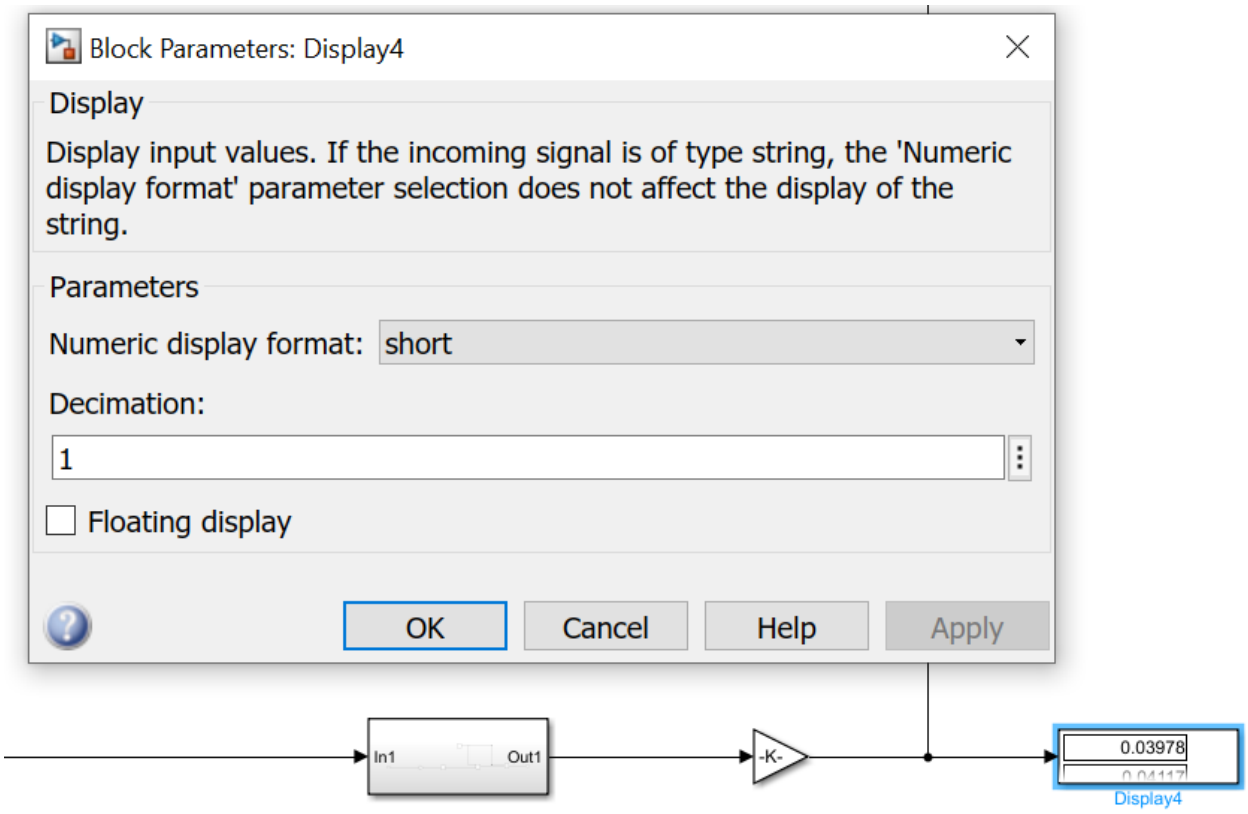

Рисунок 8 – Настройки блоков Display

Создаём блок Add с четырьмя входами (в его параметрах прописываем - «++++»). Блок должен иметь вид, указанный на рисунке 9.

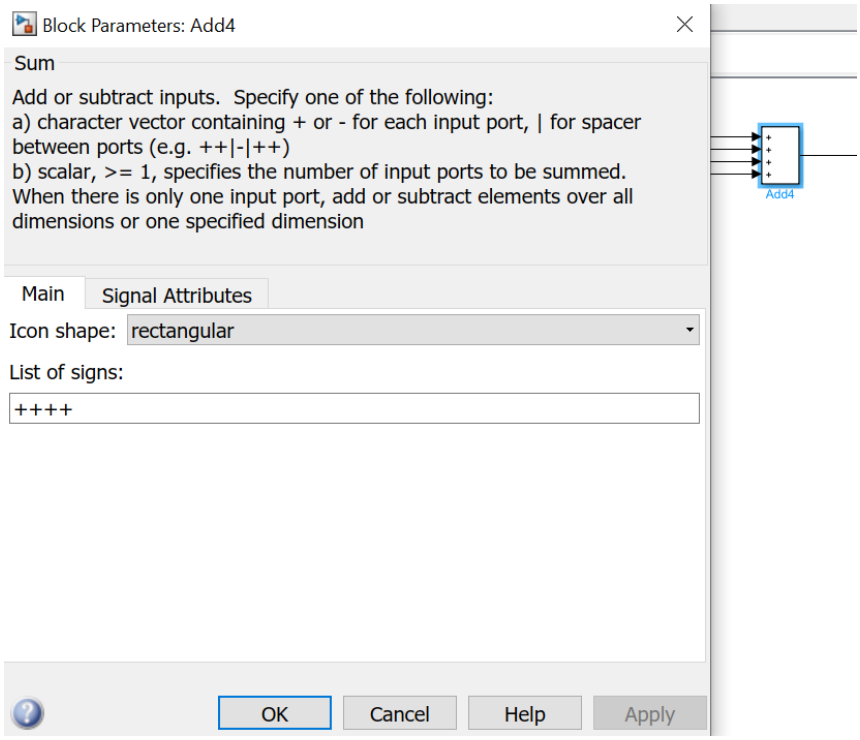

Рисунок 9 – Параметры нового блока Add

Необходимо настроить соединение всех блоков. Соединяем выходные блоки Gain с блоками Display, от этих соединений протягиваем связь к соответственным входам в новый блок Add. Общий вид соединений должен быть подобен приведённому на рисунке 10.

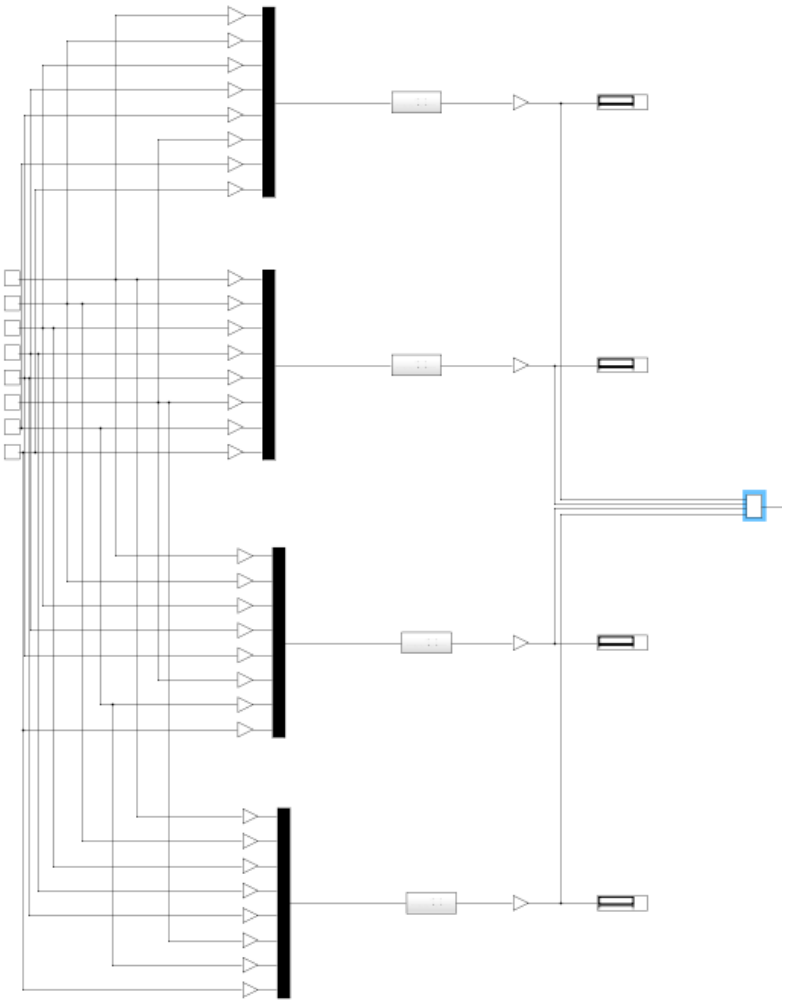

Рисунок 10 – Общий вид новых соединений

На выходе из конечного блока Add ставим ещё одну подсистему, идентичную тем, что были созданы ранее. Её можно копировать перетаскиванием и одновременным нажатием кнопки Ctrl. Настраиваем соединение, как показано на рисунке 11.

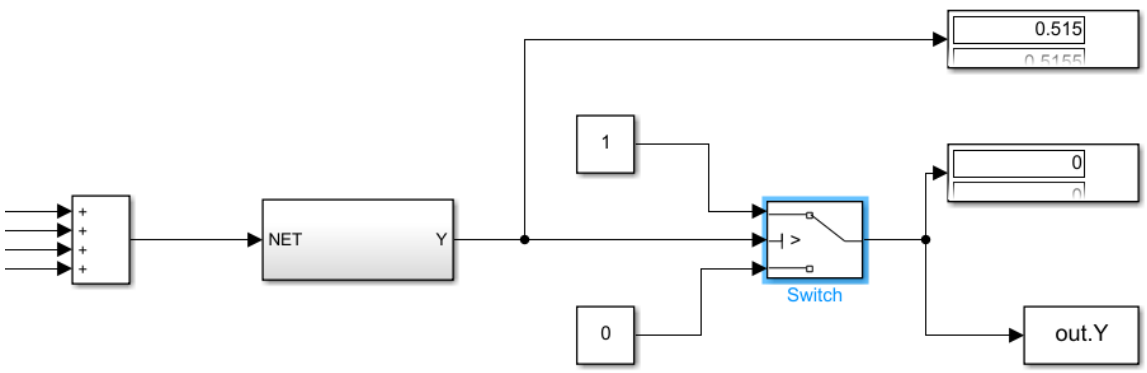

Рисунок 11 – Выходной нейрон

На рисунке 11 можно увидеть построение выходного нейрона. Конечное значение, которое выдаёт выходной нейрон, будет зависеть от весов нейронов первого слоя и от значений на входах. Блок Switch со входными блоками Constant, равными 0 и 1, необходимы для обучения сети. Блоки Display отражают значения нейрона второго слоя (верхний блок) и непосредственно второго нейрона (нижний блок). Блок под названием «out.Y» (его тип - To Workspace) передаёт значение переменной Y в рабочее пространство Matlab, код системы будет рассмотрен ниже. Настройки блока «out.Y» показаны на рисунке 12.

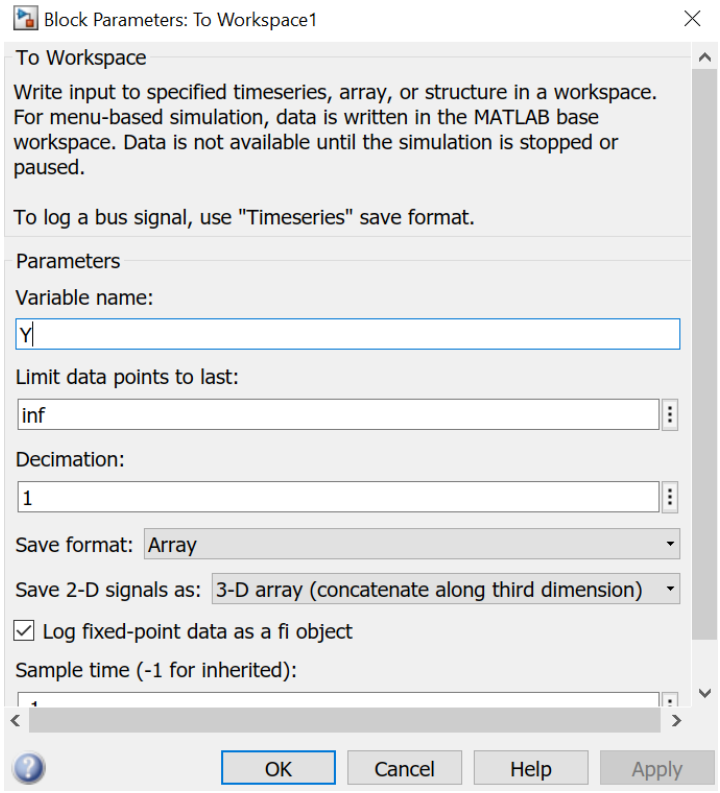

Рисунок 12 – Настройки блока «out.Y»

После завершения всех настроек система имеет вид, показанный на рисунке 13.

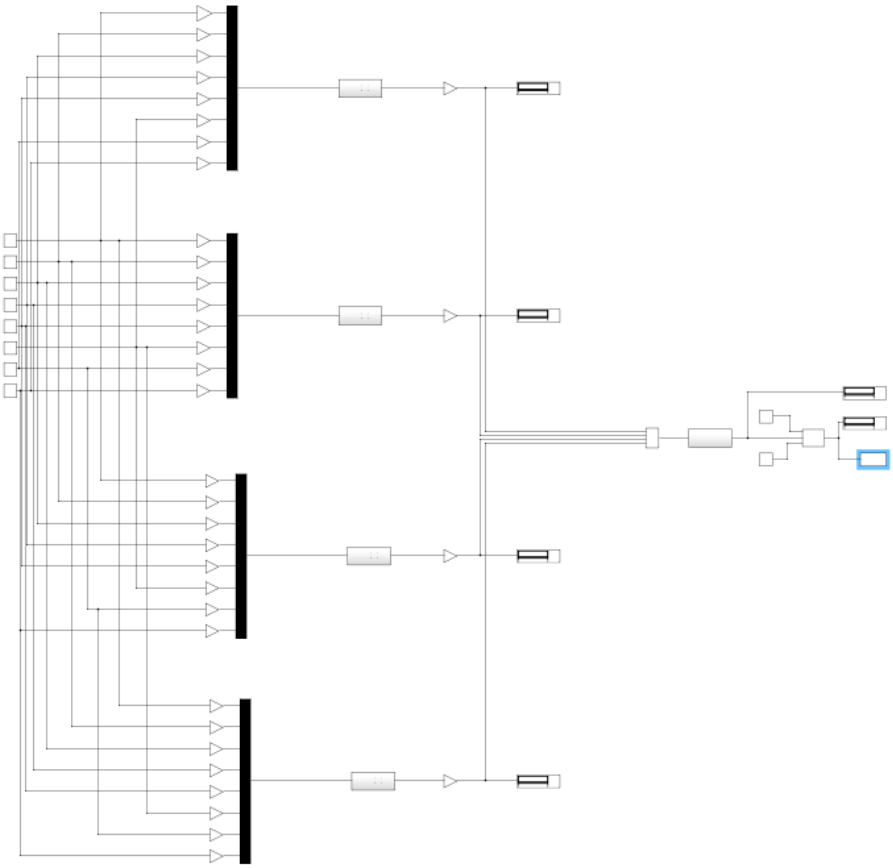

Рисунок 13 – Общий вид системы

Перейдём к рассмотрению программной части работы. Код для m-файла приведён ниже, далее мы рассмотрим отдельные его составляющие.

```
W = rand(8,4)*0.1;
h = rand(4,1)*0.1;
sum=0;
e=1;
eteration=0;
delt=0.00001;
load('MatV.mat');
load('MatYt.mat');
for I = 1:40
 x1=V(1,i);
 x2=V(2,i);
 x3=V(3,i);
 x4=V(4,i);
 x5=V(5,i);
 x6=V(6,i);
 x7=V(7,i);
 x8=V(8,i);
sim('R1.slx');
E=sqrt((Yt(I,1)-ans.Y(1)).^2);
if e > 0
    delt = - delt;
else
    delt = delt;
    M1 = M1 + delt*H1;
    M1 = M1 + delt*H1;
```

```
    M1 = M1 + delt*H1;
    M1 = M1 + delt*H1;

delta_out = e * dY_dt_out(1);  % локальный градиент для выходного нейрона

        % Корректировка весов последнего нейрона:
        W(8,1) = W(8,1) - lambda * delta_out * Y_1(1);
        W(8,2) = W(8,2) - lambda * delta_out * Y_2(1);
        W(8,3) = W(8,3) - lambda * delta_out * Y_3(1);
        W(8,4) = W(8,4) - lambda * delta_out * Y_4(1);

        % Определение локальных градиентов для первого слоя нейронов:
        delta_1 = delta_out * W(8,1) * dY_dt_1(1);
        delta_2 = delta_out * W(8,2) * dY_dt_2(1);
        delta_3 = delta_out * W(8,3) * dY_dt_3(1);
        delta_4 = delta_out * W(8,4) * dY_dt_4(1);

        % Корректировка весов первого слоя нейронов:
        W(1,1) = W(1,1) - lambda * delta_1 * V(i,1);
        W(1,2) = W(1,2) - lambda * delta_2 * V(i,1);
        W(1,3) = W(1,3) - lambda * delta_3 * V(i,1);
        W(1,4) = W(1,4) - lambda * delta_4 * V(i,1);

        W(2,1) = W(2,1) - lambda * delta_1 * V(i,2);
        W(2,2) = W(2,2) - lambda * delta_2 * V(i,2);
        W(2,3) = W(2,3) - lambda * delta_3 * V(i,2);
        W(2,4) = W(2,4) - lambda * delta_4 * V(i,2);

        W(3,1) = W(3,1) - lambda * delta_1 * V(i,3);
        W(3,2) = W(3,2) - lambda * delta_2 * V(i,3);
        W(3,3) = W(3,3) - lambda * delta_3 * V(i,3);
        W(3,4) = W(3,4) - lambda * delta_4 * V(i,3);

        W(4,1) = W(4,1) - lambda * delta_1 * V(i,4);
        W(4,2) = W(4,2) - lambda * delta_2 * V(i,4);
        W(4,3) = W(4,3) - lambda * delta_3 * V(i,4);
        W(4,4) = W(4,4) - lambda * delta_4 * V(i,4);

        W(5,1) = W(5,1) - lambda * delta_1 * V(i,5);
        W(5,2) = W(5,2) - lambda * delta_2 * V(i,5);
        W(5,3) = W(5,3) - lambda * delta_3 * V(i,5);
        W(5,4) = W(5,4) - lambda * delta_4 * V(i,5);

        W(6,1) = W(6,1) - lambda * delta_1 * V(i,6);
        W(6,2) = W(6,2) - lambda * delta_2 * V(i,6);
        W(6,3) = W(6,3) - lambda * delta_3 * V(i,6);
        W(6,4) = W(6,4) - lambda * delta_4 * V(i,6);

        W(7,1) = W(7,1) - lambda * delta_1 * V(i,7);
        W(7,2) = W(7,2) - lambda * delta_2 * V(i,7);
        W(7,3) = W(7,3) - lambda * delta_3 * V(i,7);
        W(7,4) = W(7,4) - lambda * delta_4 * V(i,7);

        W(8,1) = W(8,1) - lambda * delta_1 * V(i,8);
        W(8,2) = W(8,2) - lambda * delta_2 * V(i,8);
        W(8,3) = W(8,3) - lambda * delta_3 * V(i,8);
        W(8,4) = W(8,4) - lambda * delta_4 * V(i,8);

        sum = sum + E;  % суммарная ошибка
        eteration = eteration + 1;  % количество итераций для обучения
end;
end;
```

а)

```
W = rand(8,4)*0.1;
h = rand(4,1)*0.1;
sum=0;
e=1;
eteration=0;
delt=0.00001;
```

Первые две строки задают значения входных параметров для первого и второго слоя соответственно. Значения весов задаются произвольно, с помощью функции **rand**. Первое значение в скобках – число входных параметров, второе – число нейронов. Поскольку получающиеся значения весов велики, мы умножаем их на 0,1.

Переменные **sum, e, eteration** и **delt** понадобятся нам для обучения функции.

б)

```
load('MatV.mat');
load('MatYt.mat');
for I = 1:40
 x1=V(1,i);
 x2=V(2,i);
 x3=V(3,i);
 x4=V(4,i);
 x5=V(5,i);
 x6=V(6,i);
 x7=V(7,i);
 x8=V(8,i);
sim('R1.slx');
```

Первые две строки отражают создание матрицы весов. Для первой матрицы выбрана выборка в 40 различных наборов нулей и единиц для каждого входного параметра. Соответственно, в таблице **MatV** должно быть 8 строк и 40 столбцов, и наборы в столбцах не должны повторяться.

MatYtr - это набор выходных значений, которые нейросеть должна выдавать при соответствующем наборе входных параметров (**MatV**). Матрица **MatYtr** имеет 40 строк и 1 столбец. Первые 20 строк – это нули, остальные 20 строк заполнены единицами.

Есть два способа задать эти матрицы. Можно прописать их в самом коде, с перечислением их элементов, или импортировать, как это сделано в нашем коде.

Чтобы импортировать матрицу, можно сначала задать все значения в Microsoft Excel и перенести сразу все значения в соответствующую

переменную, либо внести поштучно в самой переменной. Работа с матрицей **MatV** показана на рисунке 14.

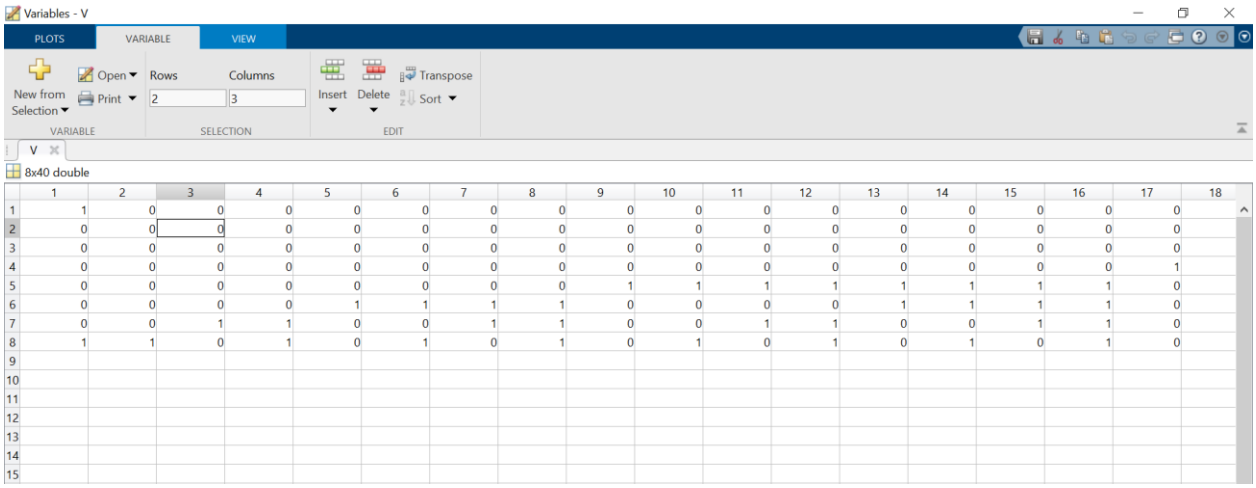

Рисунок 14 – Матрица **MatV**

Работа с переменными проводится в левом нижнем углу рабочей области Matlab (см. Рисунок 15).

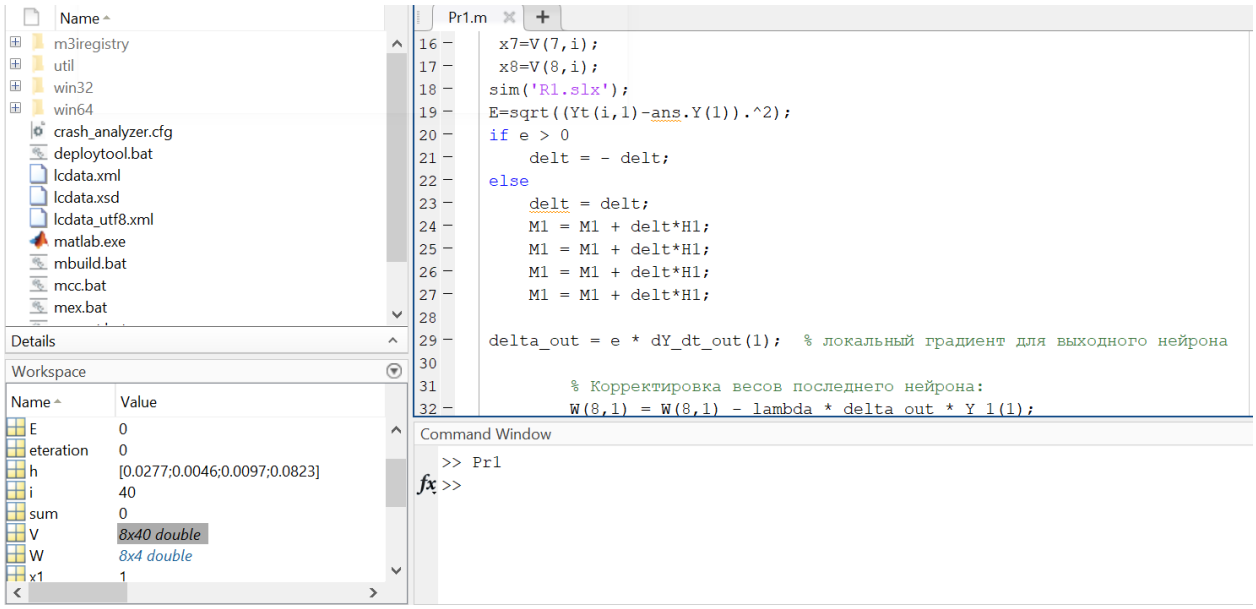

Рисунок 15 – Матрица **MatV**

Цикл из восьми строк отражает последовательный перебор наборов значений для входных параметров. Функция **sim** запускает симуляцию ранее созданной схемы.

в)

```
E=sqrt((Yt(I,1)-ans.Y(1)).^2);
if e > 0
    delt = - delt;
```

```
else
    delt = delt;
    M1 = M1 + delt*H1;
    M1 = M1 + delt*H1;
    M1 = M1 + delt*H1;
    M1 = M1 + delt*H1;
```

Начинаем «обучение» сети. Первая строка рассчитывает действительное значение ошибки. Если значение больше 0, знак значения переменной **delt** меняется на противоположный. В противном случае значение веса этого нейрона увеличивается.

г)

```
delta_out = e * dY_dt_out(1);  % локальный градиент для выходного нейрона

        % Корректировка весов последнего нейрона:
        W(8,1) = W(8,1) - lambda * delta_out * Y_1(1);
        W(8,2) = W(8,2) - lambda * delta_out * Y_2(1);
        W(8,3) = W(8,3) - lambda * delta_out * Y_3(1);
        W(8,4) = W(8,4) - lambda * delta_out * Y_4(1);

        % Определение локальных градиентов для первого слоя нейронов:
        delta_1 = delta_out * W(8,1) * dY_dt_1(1);
        delta_2 = delta_out * W(8,2) * dY_dt_2(1);
        delta_3 = delta_out * W(8,3) * dY_dt_3(1);
        delta_4 = delta_out * W(8,4) * dY_dt_4(1);

        % Корректировка весов первого слоя нейронов:
        W(1,1) = W(1,1) - lambda * delta_1 * V(i,1);
        W(1,2) = W(1,2) - lambda * delta_2 * V(i,1);
        W(1,3) = W(1,3) - lambda * delta_3 * V(i,1);
        W(1,4) = W(1,4) - lambda * delta_4 * V(i,1);

        W(2,1) = W(2,1) - lambda * delta_1 * V(i,2);
        W(2,2) = W(2,2) - lambda * delta_2 * V(i,2);
        W(2,3) = W(2,3) - lambda * delta_3 * V(i,2);
        W(2,4) = W(2,4) - lambda * delta_4 * V(i,2);

        W(3,1) = W(3,1) - lambda * delta_1 * V(i,3);
        W(3,2) = W(3,2) - lambda * delta_2 * V(i,3);
        W(3,3) = W(3,3) - lambda * delta_3 * V(i,3);
        W(3,4) = W(3,4) - lambda * delta_4 * V(i,3);

        W(4,1) = W(4,1) - lambda * delta_1 * V(i,4);
        W(4,2) = W(4,2) - lambda * delta_2 * V(i,4);
        W(4,3) = W(4,3) - lambda * delta_3 * V(i,4);
        W(4,4) = W(4,4) - lambda * delta_4 * V(i,4);

        W(5,1) = W(5,1) - lambda * delta_1 * V(i,5);
        W(5,2) = W(5,2) - lambda * delta_2 * V(i,5);
        W(5,3) = W(5,3) - lambda * delta_3 * V(i,5);
        W(5,4) = W(5,4) - lambda * delta_4 * V(i,5);

        W(6,1) = W(6,1) - lambda * delta_1 * V(i,6);
        W(6,2) = W(6,2) - lambda * delta_2 * V(i,6);
        W(6,3) = W(6,3) - lambda * delta_3 * V(i,6);
        W(6,4) = W(6,4) - lambda * delta_4 * V(i,6);

        W(7,1) = W(7,1) - lambda * delta_1 * V(i,7);
```

```
        W(7,2) = W(7,2) - lambda * delta_2 * V(i,7);
        W(7,3) = W(7,3) - lambda * delta_3 * V(i,7);
        W(7,4) = W(7,4) - lambda * delta_4 * V(i,7);

        W(8,1) = W(8,1) - lambda * delta_1 * V(i,8);
        W(8,2) = W(8,2) - lambda * delta_2 * V(i,8);
        W(8,3) = W(8,3) - lambda * delta_3 * V(i,8);
        W(8,4) = W(8,4) - lambda * delta_4 * V(i,8);

        sum = sum + E;  % суммарная ошибка
        eteration = eteration + 1;  % количество итераций для обучения
end;
end;
```

В этом разделе проводится настройка градиентов для каждого нейрона с целью минимизации суммарной ошибки, с циклическим пересчётом этой ошибки и последовательным увеличением итераций для обучения.

Если работа выполнена верно, в дисплеях Simulink-схемы можно будет увидеть рассчитанные значения весов. Чтобы посмотреть значение конкретной переменной в файле Matlab, необходимо убрать точку с запятой в конце той строки, где рассчитывается значение этой переменной. После выполнения кода она будет выведена в командное окно.