**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САУ**

отчет

**по лабораторной работе №4**

**по дисциплине «Интеллектуальные системы управления»**

Тема: **РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ «И», «ИЛИ», «НЕ»**

**С ПОМОЩЬЮ ИСКУССТВЕННОГО НЕЙРОНА**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9492 |  | Ливаренко С.С |
| Преподаватель |  | Порохненко К.А. |

Санкт-Петербург

2024

**Цель работы.**

Исследование искусственных нейронов (ИН) типа перцептрона, обучение ИН выполнению логических функций «не», «и», «или»; решение задачи классификации с помощью ИН в пакете *Neural Networks Toolbox*, моделирование функций в *Toolbox Simulink.*

**Основные сведения**

*Искусственный нейрон* – элементарный преобразовательный элемент, содержащий *n* вектор входов ***r***, суммирующий блок, блок преобразования сигналаспомощью  функции активации, скалярный выход *q* (рис. 4.1, а). В суммирующем блоке вычисляется взвешенная сумма *n* входных сигналов *ri*

*s* =, где *Wi* – весовой коэффициент *ri* входа. Вход  и коэффициент *W*0 вводят специально для смещения нейронов сети, обычно = 1. В модели ИН типа *перцептрон* (модель МакКаллока–Питса) в качестве функции активации *f*(*s*) используется пороговая функция, в нейроне *сигмоидального* типа – униполярная (логистическая) или биполярная (гиперболический тангенс) сигмоидальные функции, в нейроне типа *адалина* – линейная функция [1].

*Радиальный базисный нейрон* (рис. 4.1, б) включает *n* вектор входов ***r***, блок, в котором вычисляется расстояние между вектором входа ***r*** и вектором весовых коэффициентов ***W***, блок преобразования с помощью функции активации, в качестве которой используется *радиальная базисная функция*. Полученное в первом блоке расстояние умножается на фиксированный порог α, который позволяет управлять чувствительностью ИН. Радиальная базисная функция (*RBF*) имеет максимум равный единице, когда вход равен нулю, т.е. единица на выходе, когда входной вектор равен вектору весовых коэффициентов.

**Результаты работы**

1. Реализация логических функций «и», «или», «не»*.*

На рисунках 1-4 сформировано изображение разделяющих линий для различных логических функций.

1. *Реализация функции «и»*

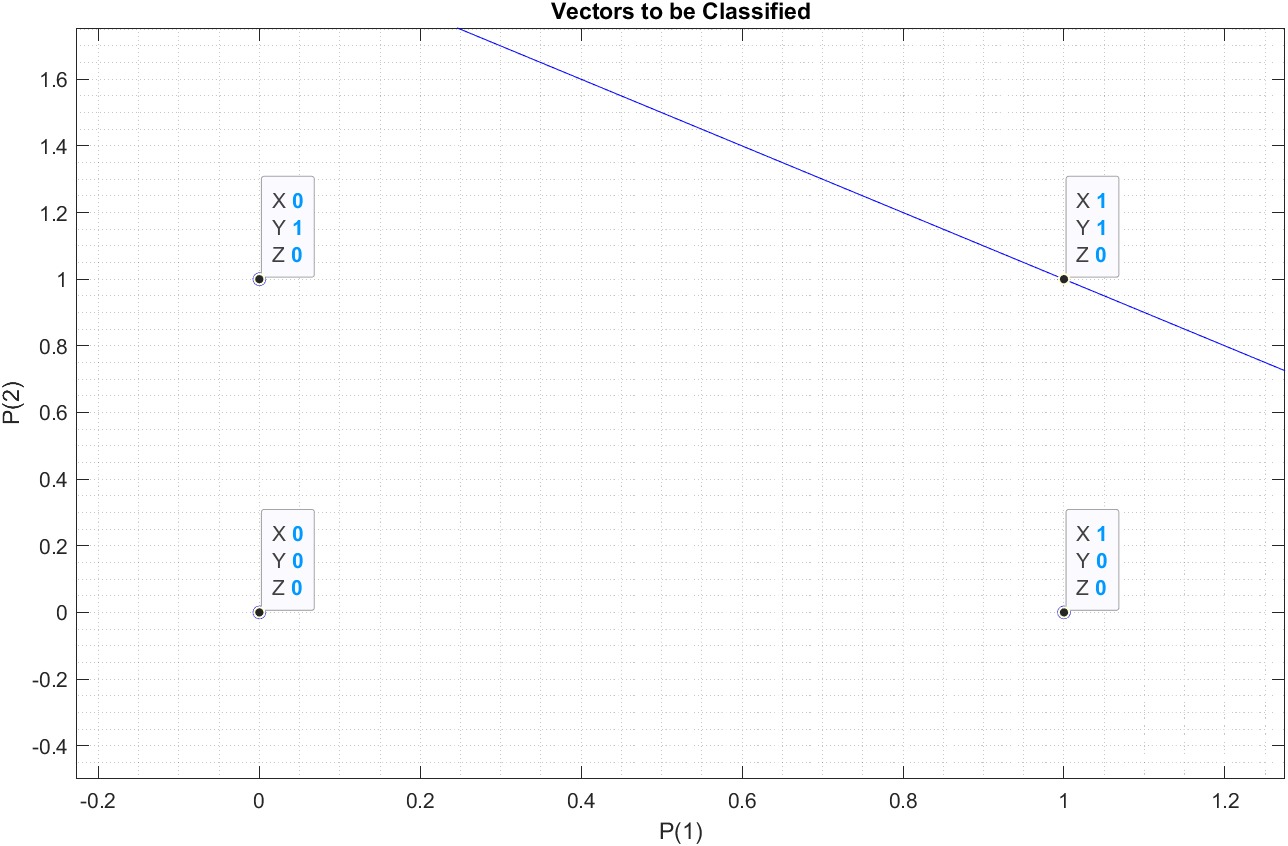
**

Рисунок 1 – Изображение разделяющей линии

1. *Реализации функции «или»*

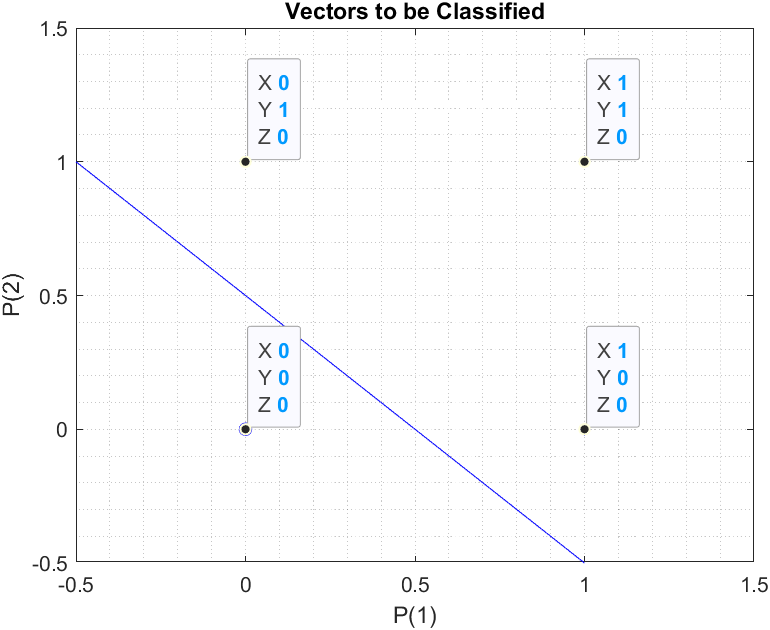
**

Рисунок 2 – Изображение разделяющей линии

1. *Реализации функции «не»*

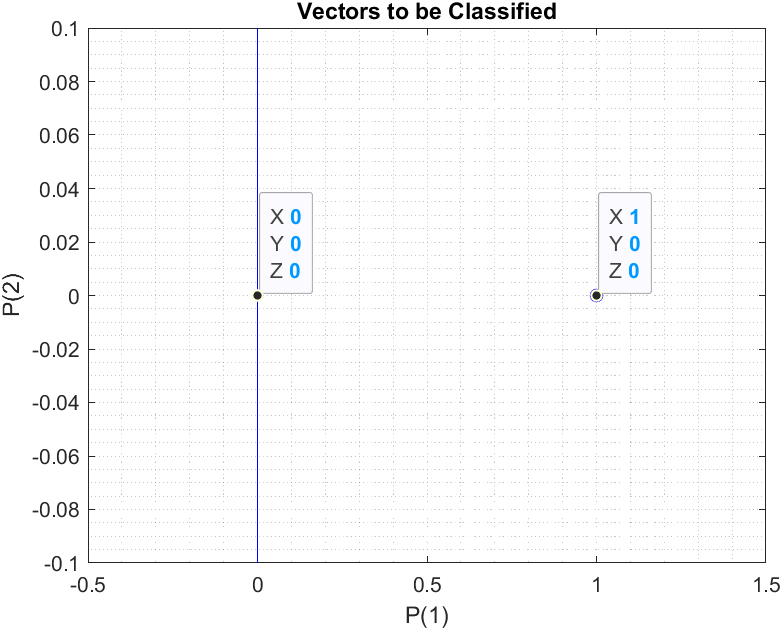
**

Рисунок 3 – Изображение разделяющей линии

1. *Реализации функции «исключающее «или»»*

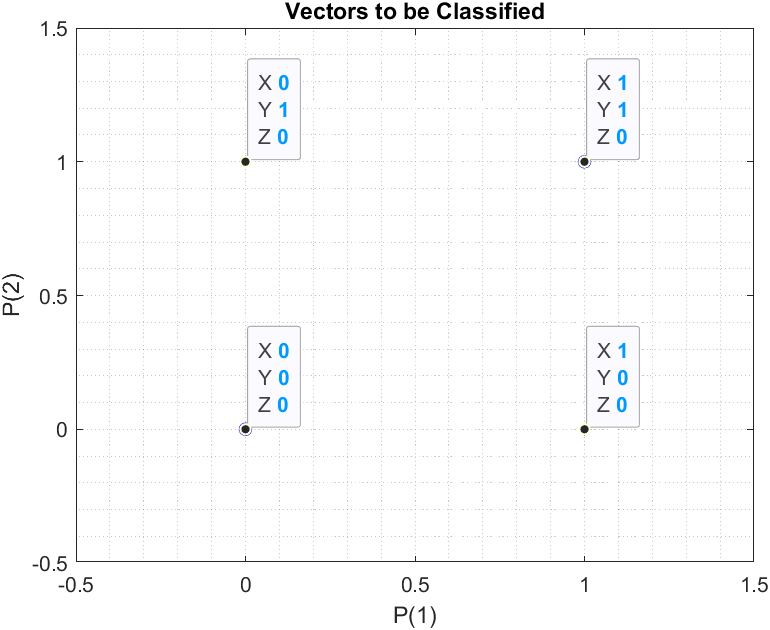


Рисунок 4 – Изображение разделяющей линии

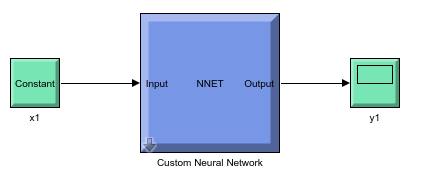
Как видно из графиков, при реализации функции «и» выходная функция равна нулю в трех случаях и только при x = 1 и y = 1 выходная функция будет равна единице. Таким образом, разделяющая линия делит плоскость на два участка: область, что выше, нейрон будет считать за единицу, а та, что ниже – за ноль.

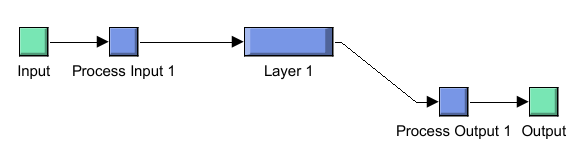
При реализации функции «или» выходная функция равна единице в трех случаях и только при x = 0 и y = 0 выходная функция будет равна нулю. Таким образом, разделяющая линия делит плоскость также на два участка по той же логике, что и в случае функции «и».

При реализации функции «не» выходная функция может принимать всего лишь два значения, являющиеся противоположными входным значениям. В этом случае разделяющая линия вертикальная. Область, которая слева, принимается нейроном за единицу, а та, которая справа – за ноль.

Также очевидно, что при реализации функции «исключающее «или» нейрон не обучается.

1. Формирование нейросетевой модели в Toolbox Simulink





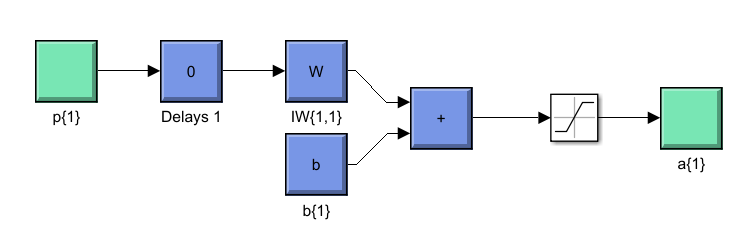


Рисунок 5 – Блок диаграмма

а) Моделирование функции «и»

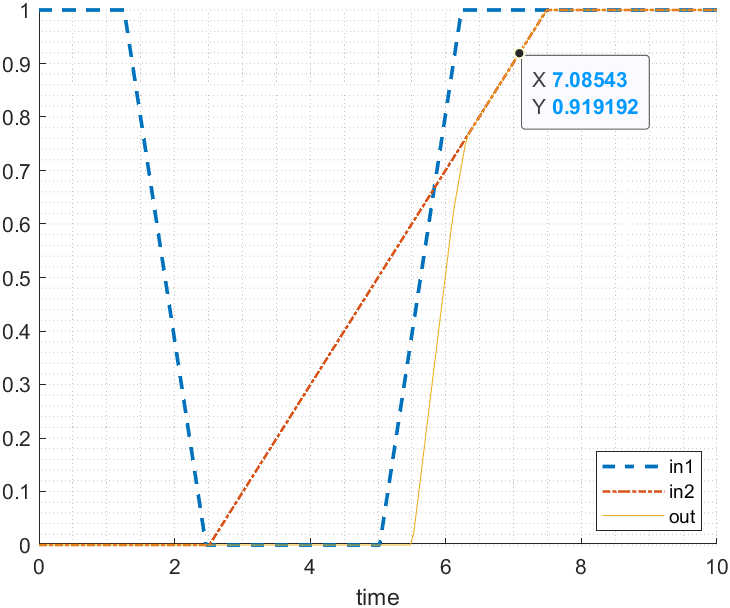
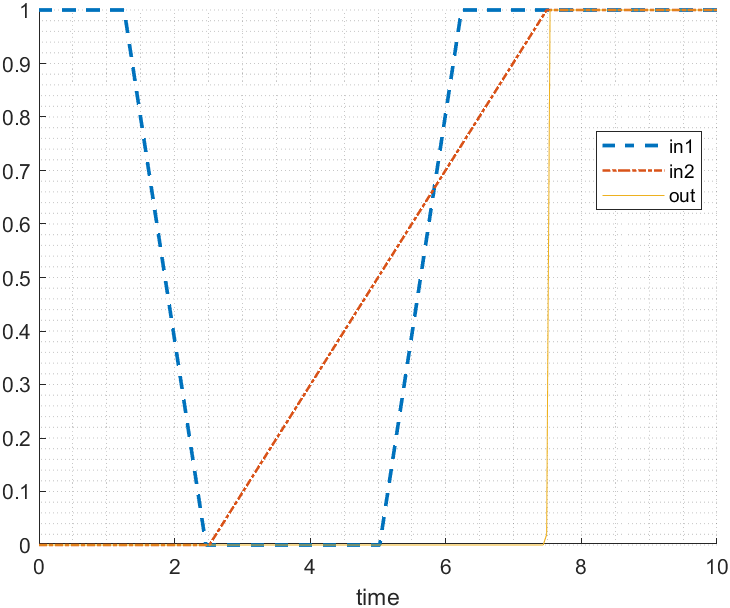


Рисунок 6 – Результат моделирования нейросетевой модели функции «и» (слева – ручное сравнение, справа – сравнение через Simulink-модель)

b) Моделирование функции «или»

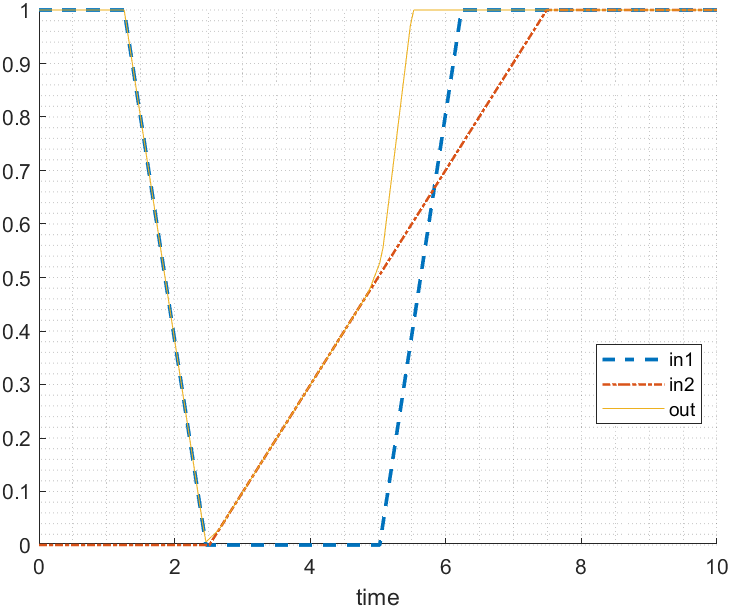
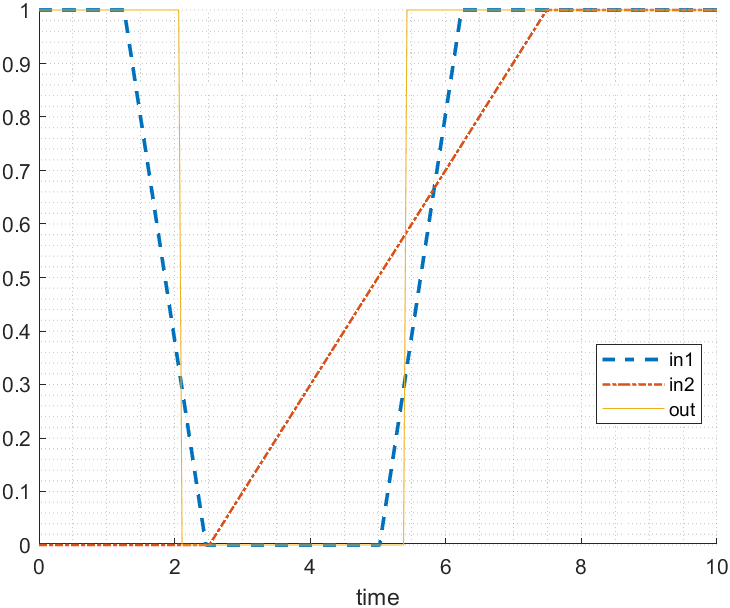


Рисунок 7 – Результат моделирования нейросетевой модели функции «или» (слева – ручное сравнение, справа – сравнение через Simulink-модель)

b) Моделирование функции «не»

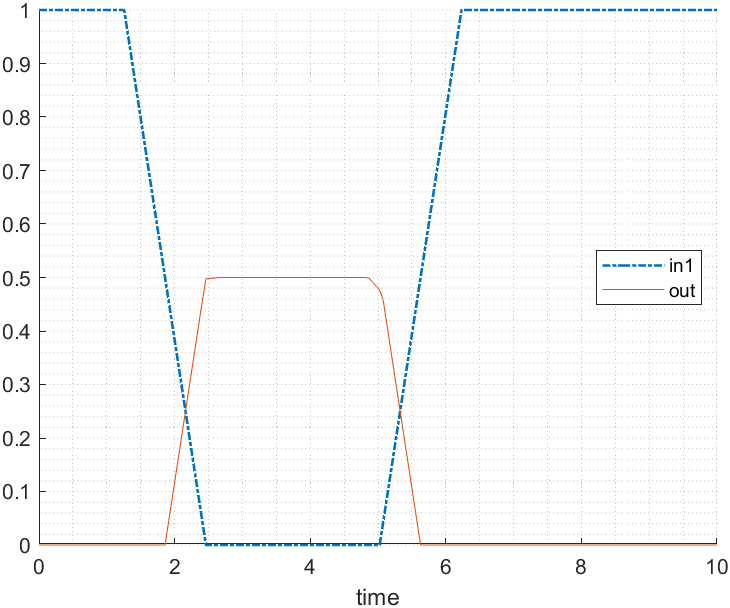
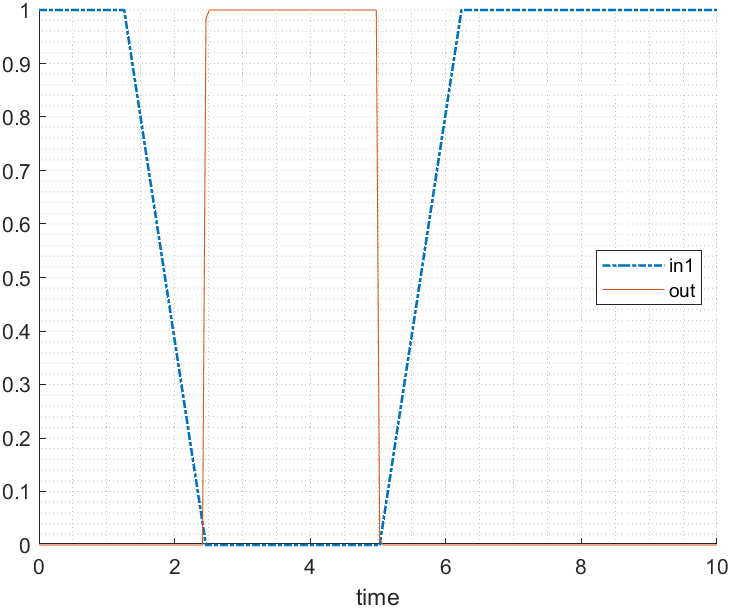


Рисунок 8 – Результат моделирования нейросетевой модели функции «не» (слева – ручное сравнение, справа – сравнение через Simulink-модель)

b) Моделирование функции «исключающее «или»»

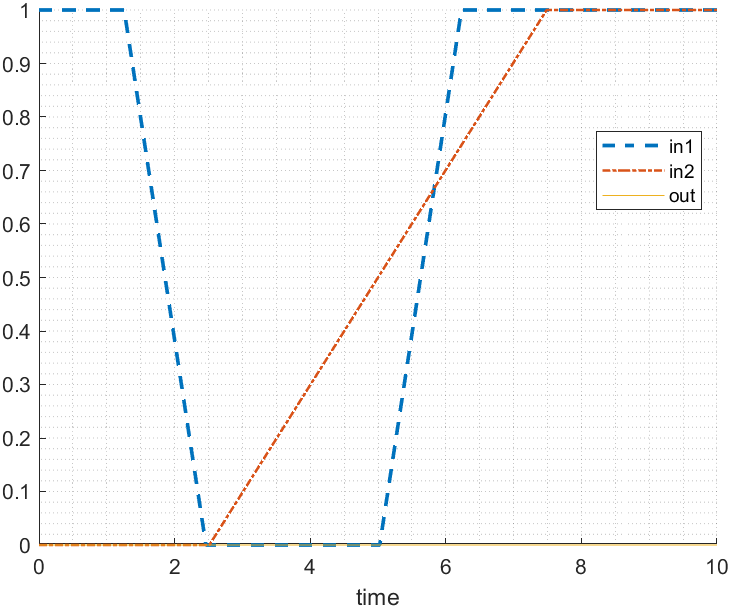
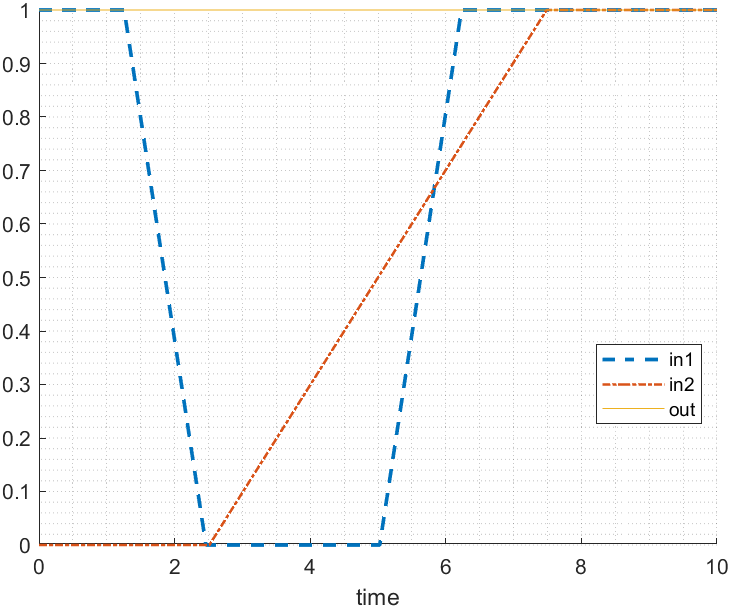


Рисунок 9 – Результат моделирования нейросетевой модели функции «исключающее «или»» (слева – ручное сравнение, справа – сравнение через Simulink-модель)

Из рисунков 6-9 можно увидеть, что результаты моделирования соответствуют ожиданиям.

**Вывод:**

В ходе лабораторной работы было проведено исследование искусственных нейронов (ИН) типа перцептрона, обучение ИН выполнению логических функций «не», «и», «или». Также были решены задачи классификации с помощью ИН в пакете *Neural Networks Toolbox*, смоделированы функций в *Toolbox Simulink.* Можно наглядно заметить, что нейрон обучается во всех случаях, кроме ситуации с «исключающим «или»».

**Приложение А**

**Скрипт лабораторной работы**

clc, clear, close all

P\_ = {

[0 1 0 1; 0 0 1 1],

[0 1 0 1; 0 0 1 1],

[0 1],

[0 1 0 1; 0 0 1 1]

};

T\_ = {

[0 0 0 1],

[0 1 1 1],

[1 0],

[0 1 1 0]

};

for\_newp = {

[0 1; 0 1],

[0 1; 0 1],

[0 1],

[0 1; 0 1]

};

sys\_name = {

"\_and",

"\_or",

"\_not",

"\_xor"

};

flag = 0;

for i = 1:4

% P = [0 1 0 1; 0 0 1 1]; % задание входных векторов;

P = P\_{i}; % задание входных векторов;

% T=[0 1 1 1]; % задание выходов нейрона;

% T = [0 0 0 1];

T = T\_{i};

figure, hold on, grid minor

plotpv(P,T); % графическое представление исходных векторов;

% net1 = newp([0 1; 0 1], 1); % создание перцептрона с 1 нейроном;

net1 = newp(for\_newp{i}, 1); % создание перцептрона с 1 нейроном;

E = 1;% присвоение начального значения ошибки;

net1 = init(net1); % инициализация перцептрона;

j = 0;

while(sse(E)) % организация цикла обучения перцептрона, классификация;

[net1,Y,E] = adapt(net1,P,T); % обучение нейрона на выборке [P,T];

% Получение управляющей структуры linehandle для изображения разделяющей

% линии в координатах весов (IW) и порога срабатывания нейрона (b)

if (j > 200)

disp('break')

break;

end

j = j + 1;

end

linehandle = plotpc(net1.IW{1}, net1.b{1});

drawnow;

if (flag)

[sysName,netName] = gensim(net1, 'InputMode', 'port',...

'OutputMode', 'WorkSpace', 'SolverMode', 'Discrete');

sys\_name{i} = sysName;

end

%%

N = 50;

A = 1.0;

x = [repelem(A, N/2) linspace(A, 0, N/2) repelem(0, N) linspace(0, A, N/2) repelem(A, N + N/2)];

t = linspace(0, 10, length(x));

if (size(P, 1)) == 2

y = [repelem(0, N) linspace(0, 1, 2\*N) repelem(1, N)];

ts = timeseries([x; y], t);

else

ts = timeseries(x, t);

end

% u = @ (t) timeseries([sin(t); sin(t)], t);

% out = sim(sysName,'ReturnWorkspaceOutputs', 'on', 'LoadExternalInput', ...

% 'on', 'ExternalInput', 'ts');

out = sim(("lr\_4\_model" + sys\_name{i} + ".slx"), ...

'LoadExternalInput', 'on', 'ExternalInput', 'ts')

% %%

% out.tout = 0:0.2:20

out.y1 = interp1(out.tout, out.y1, ts.Time);

figure, hold on, grid minor, xlabel('time')

if (size(P, 1)) == 2

plot(ts.Time, ts.Data(1, :), '--', 'LineWidth', 1.75)

plot(ts.Time, ts.Data(2, :), '-.', 'LineWidth', 1.25)

plot(ts.Time, out.y1) % !

legend('in1', 'in2', 'out', 'Location', 'best')

else

plot(ts.Time, ts.Data(1, :), '-.', 'LineWidth', 1.25)

plot(ts.Time, out.y1) % !

legend('in1', 'out', 'Location', 'best')

end

end

%%

function [ts] = tripuls\_(t)

A = 1;

% t=-5:0.001:5;

T1 = 4; % width A

T2 = 8; % width all impulse

s = A \* (T2 \* tripuls(t,T2) - T1 \* tripuls(t,T1)) / (T2 - T1);

numel(s)

% ts = timeseries([s; s], t+5);

ts = timeseries([s; s], t+5);

numel(ts.Data)

end