**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САУ**

отчет

**по лабораторной работе № 5**

**по дисциплине «Интеллектуальные системы управления»**

Тема: **РЕШЕНИЕ ЗАДАЧ АППРОКСИМАЦИИ СРЕДСТВАМИ**

**НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ**

**8 вариант**

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

Санкт-Петербург

2024

**Цель работы.**

Ознакомление с пакетом *Neural Networks Toolbox* системы (MATLAB); рассмотрение способов формирования и обучения сетей прямого распространения и RBF-сетей; исследование эффективности нейросетевого подхода для формирования заданных функций.

**Основные сведения**

Группы искусственных нейронов в сети образуют *слои*. Однослойные сети позволяют моделировать ряд функций, например, логические функции «и», «или», «не». Нереализуемые однослойной сетью функции (например, «исключающее или») называются *линейно неразделимыми*. Для моделирования подобных функций используют *многослойные сети*.

На рис. 5.1 многослойная сеть *прямого распространения* состоит из входного, промежуточного (скрытого) и выходного слоев. Обозначено: ***W*** 1 , ***W*** 2 – матрицы весовых коэффициентов скрытого и выходного слоев соответственно, – выход *i-*го ИН *j-*го слоя.ИН входного слоя служат для распределения сигналов между ИН скрытого слоя и не осуществляют преобразование входных сигналов. ИН каждого слоя не связаны между собой, выходы ИН *l*-го слоя поступают только на входы ИН *l*+1-го слоя. Функция активации принимается одинаковой для всех ИН скрытых слоев сети. Выходной слой, как правило, состоит из ИН типа адалина и называется в этом случае *мадалина*. *Радиальная базисная сеть* состоит из двух слоев: скрытый слой из радиальных базисных нейронов (описание дано в лабораторной работе 4), выходной слой – мадалина.

Описание функции создания многослойной нейронной сети прямого распространения *newff* : имя = newff (PR, [S1 S2… SN ], {TF1 TF2… TFN}, BTF, BLF, PF).

Аргументы функции: PR – R×2 – матрица минимальных и максимальных значений для R входных элементов; Si– число нейронов i-го слоя; TFi – функция активации i-го слоя, по умолчанию «*tansig*» (гиперболический тангенс); BTF – функция обучения, по умолчанию «*traingd*» (алгоритм обучения обратного распространения ошибки), BLF – функция настройки весов и смещений, по умолчанию «*learngdm*» (градиентный алгоритм оптимизации с инерционной составляющей), PF – функция ошибки, по умолчанию «*mse*» (среднеквадратичная ошибка).

Описание функции создания *RBF*-сети: имя = newrb (P, T, goal, spread).

Аргументы функции: P – матрица входных векторов; T – матрица целевых векторов; goal – заданная среднеквадратичная ошибка; spread – разброс функции, по умолчанию 1.0.

**Результаты работы**

1. Формирование заданной функции с точностью до 0.1% с помощью сети прямого распространения

Формирование заданной функции с помощью сети прямого распространения при исходных настройках (рисунок 1).

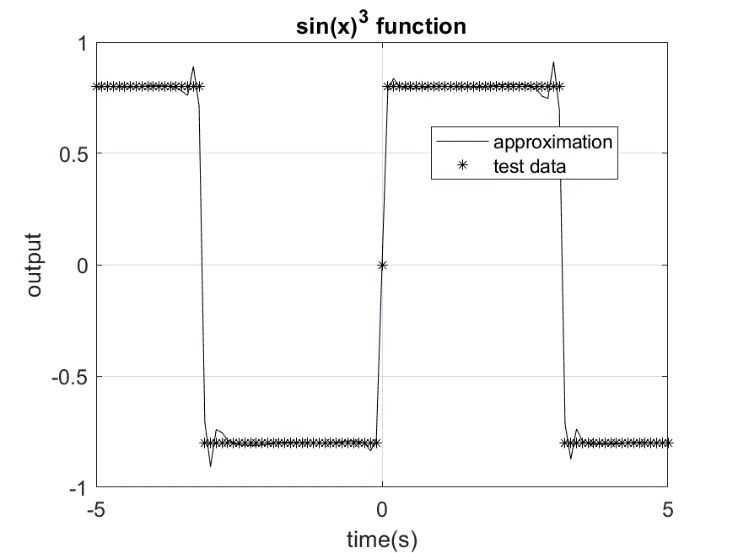
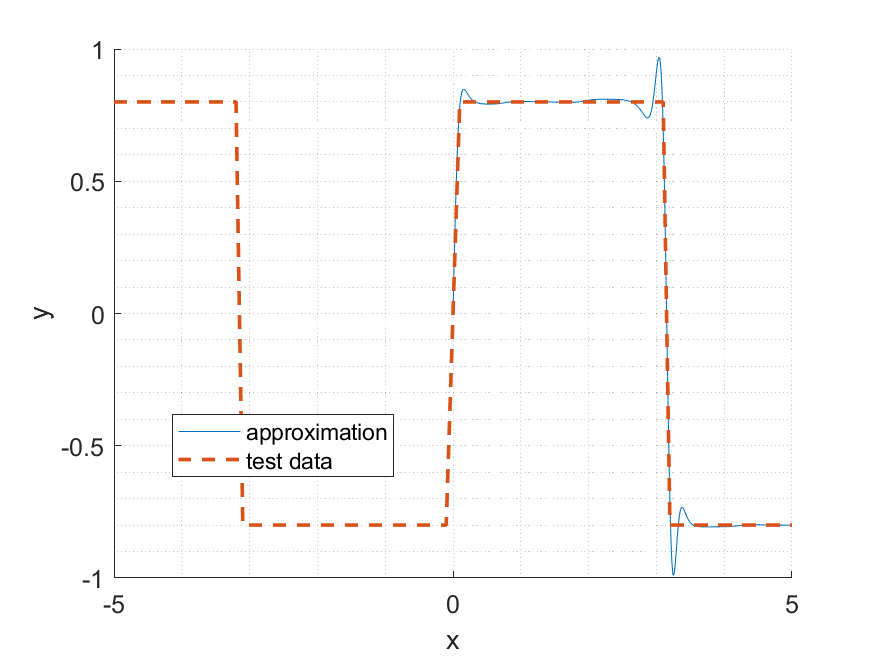
 

Рисунок 1 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

На рисунках 2-5 сформированы функции с точностью до 0.1% с помощью сети прямого распространения при изменении:

1. *Числа данных обучающей выборки*

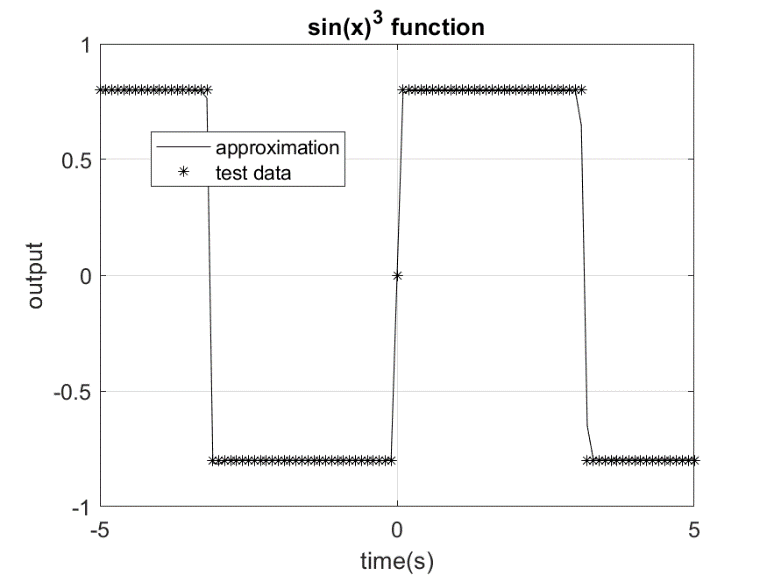
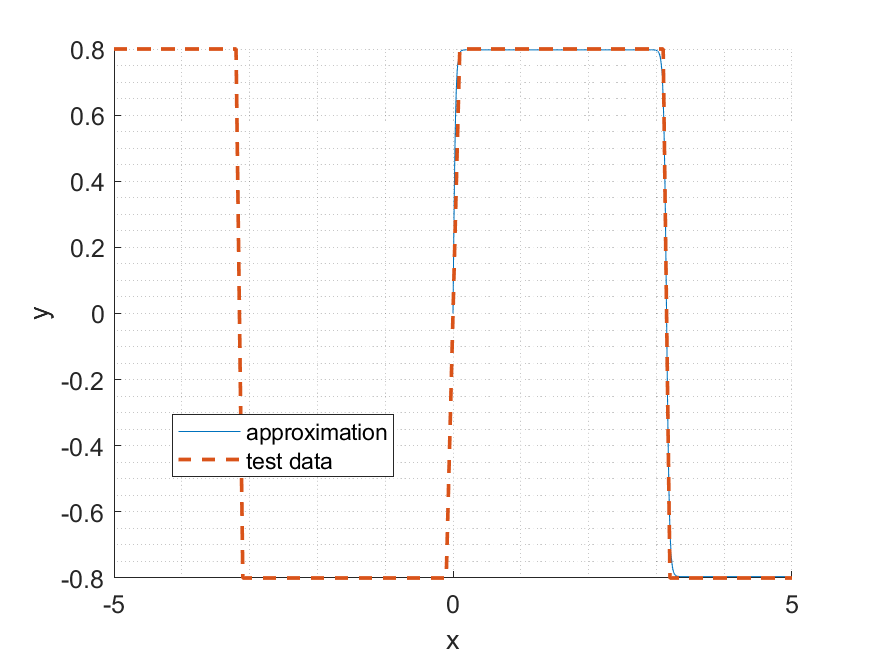
 

Рисунок 2 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. *Числа слоев сети*

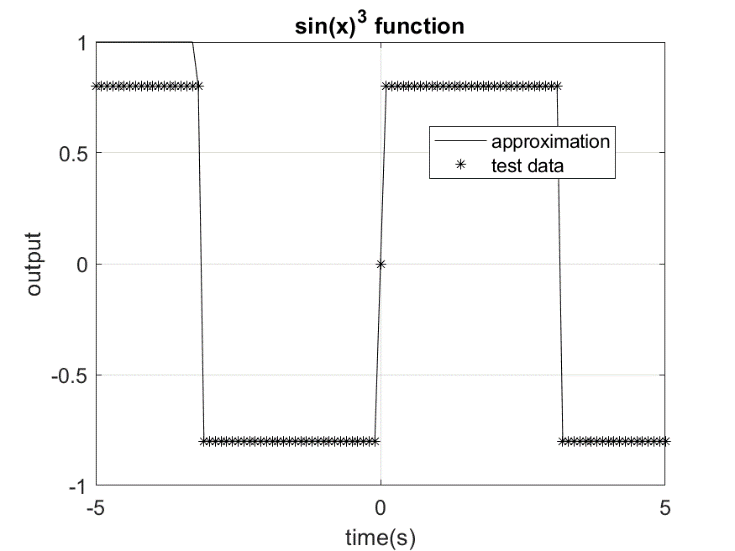
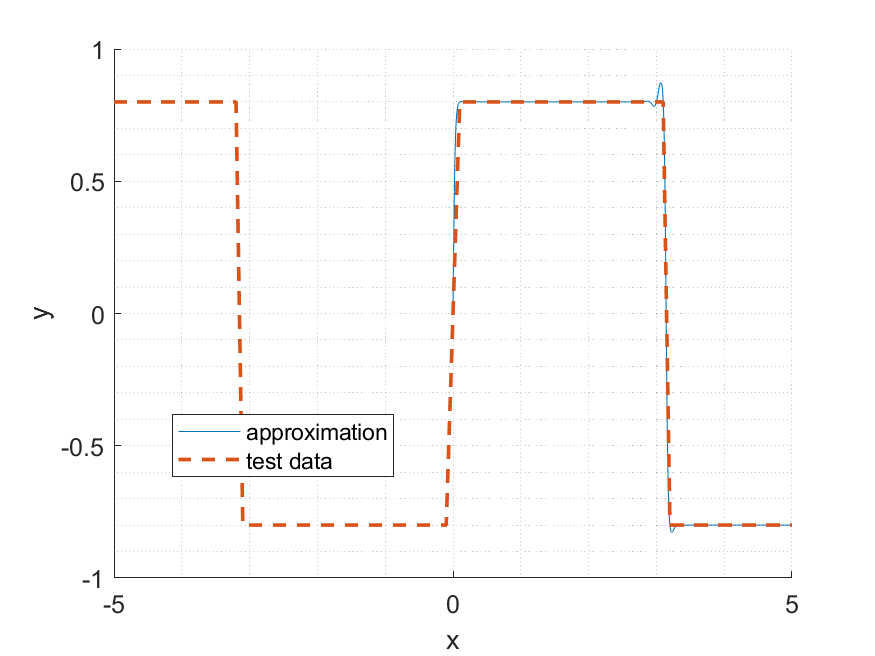
 

Рисунок 3 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. *Числа ИН в скрытом слое*

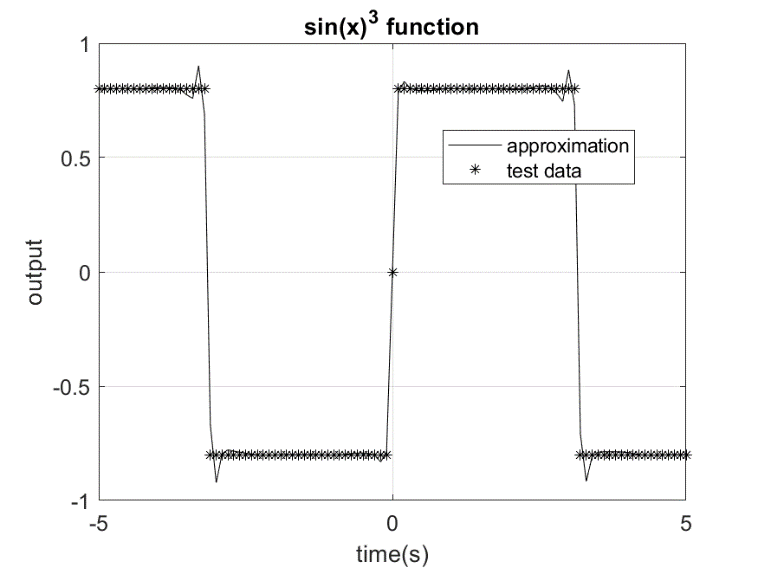
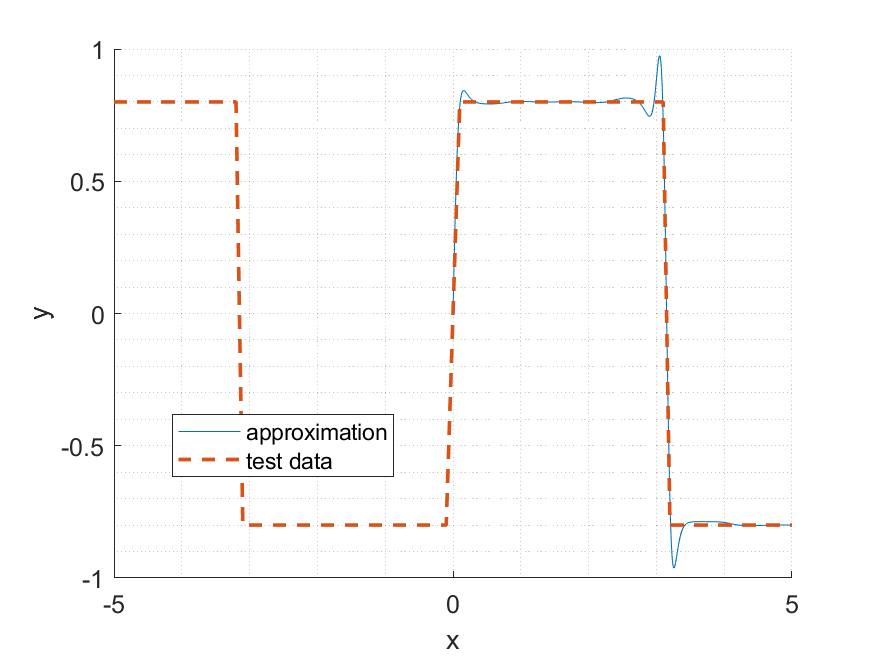
 

Рисунок 4 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. *Типа функция активации скрытого слоя*

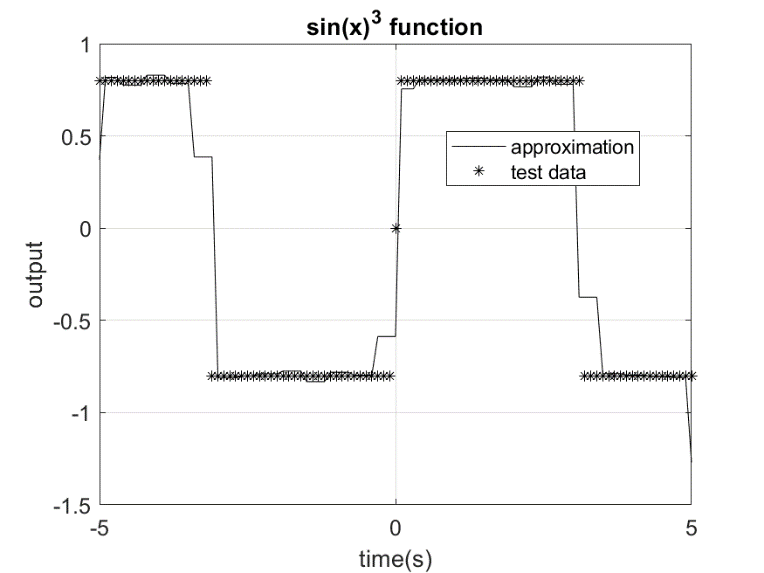
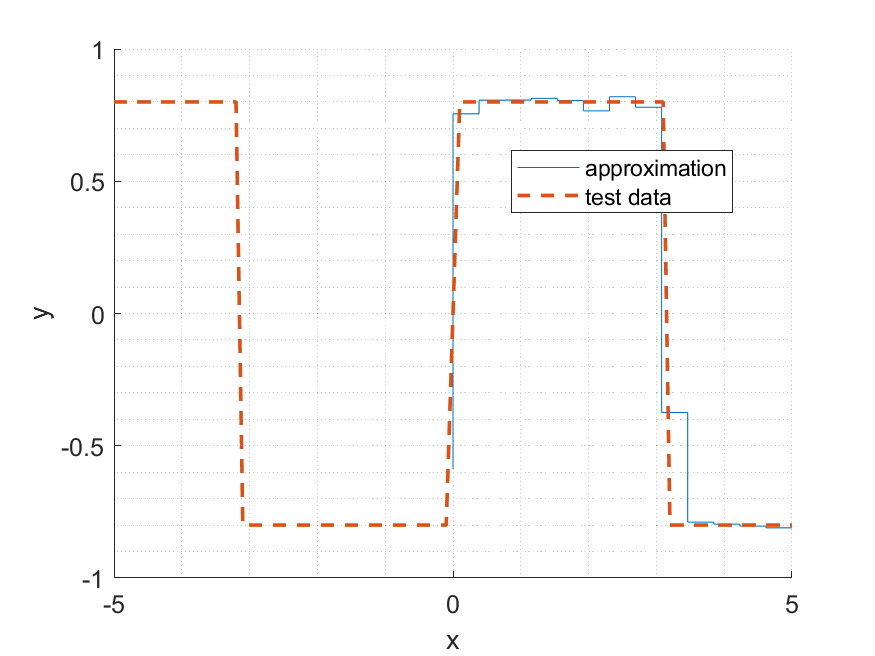
 

Рисунок 5 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. *Числа циклов обучения*

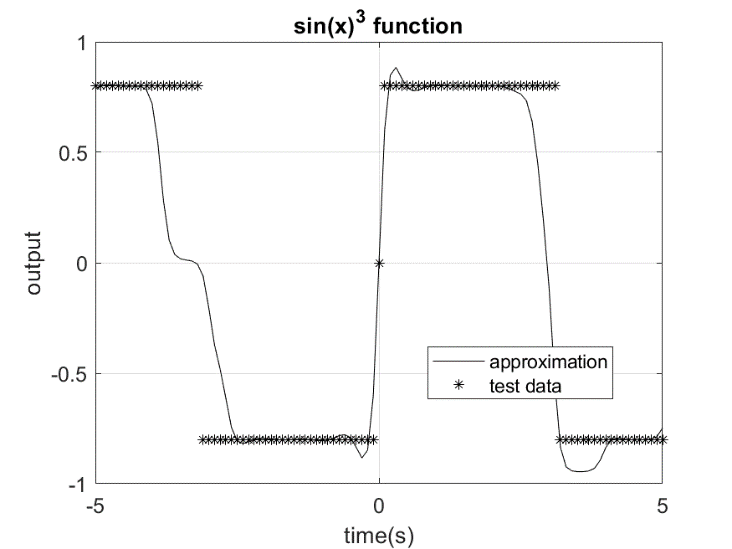
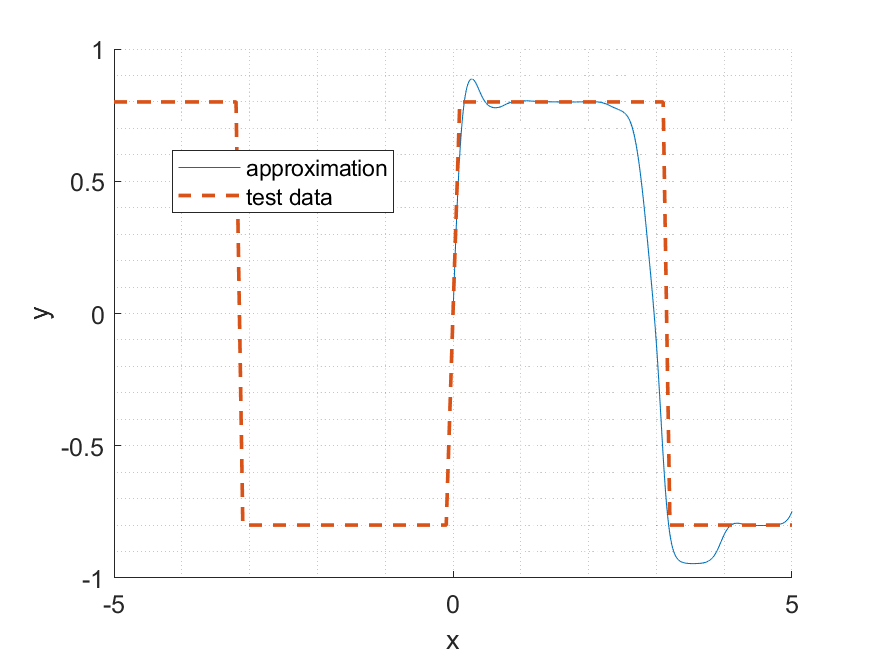
 

Рисунок 6 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. *Алгоритма обучения*

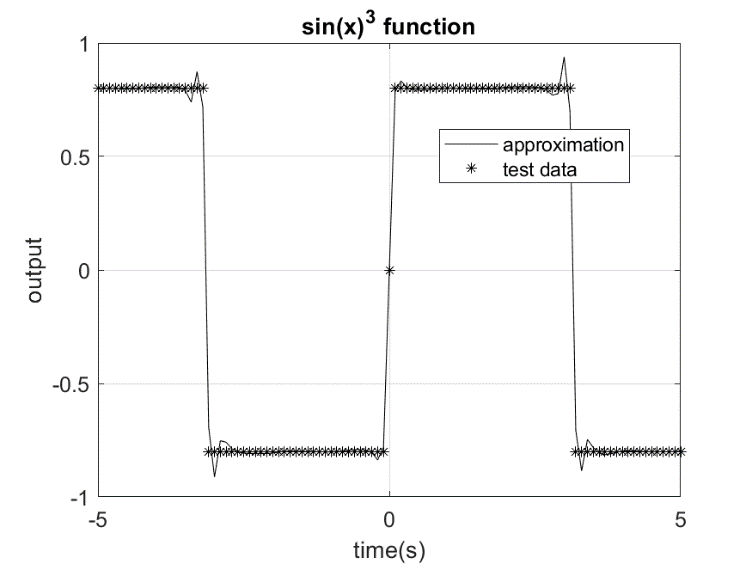
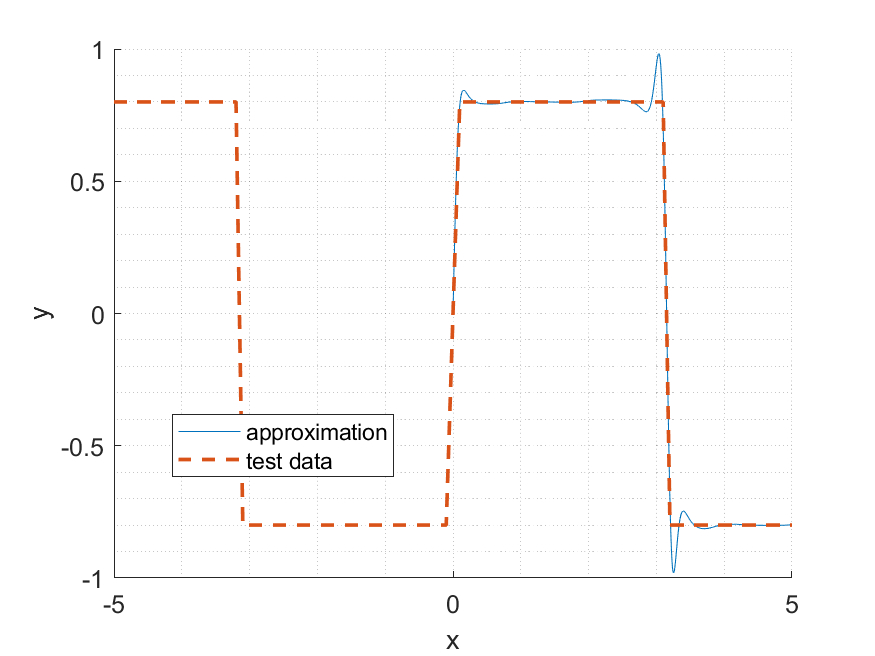
 

Рисунок 7 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

Как видно по графикам, сравнение сигналов через код MatLab и через модели в Simulink выглядят одинаково, что доказывает, что оба способа пригодны для анализа.

Видно, что при изменении алгоритма обучения, числа данных обучающей выборки, числа ИН в скрытом слое и числа слоев сети исходный и аппроксимируемый сигнал практически идентичны. Очевидно, что при изменении функции активации скрытого слоя (tansig hardlim) и уменьшения числа циклов обучения, аппроксимация происходит плохо.

1. Реализации логических функций «и», «или», «исключающее или» с помощью RBF-сети.
2. Реализации логической функций «и»

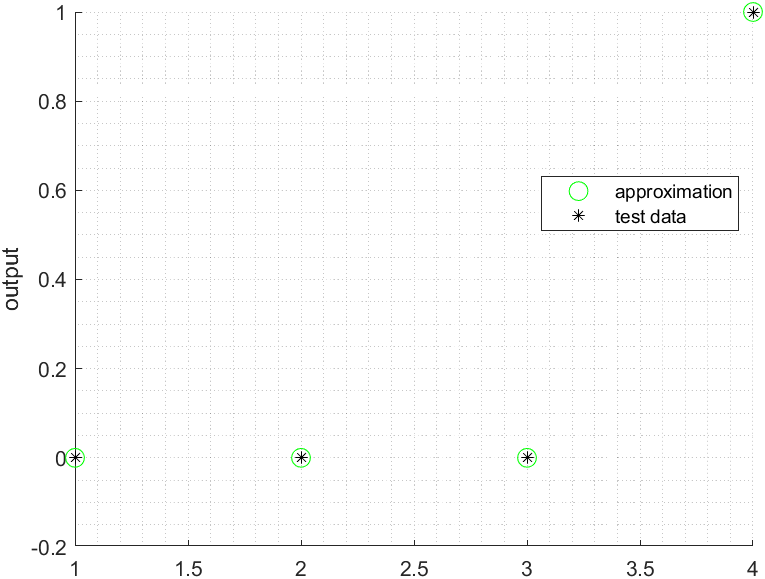
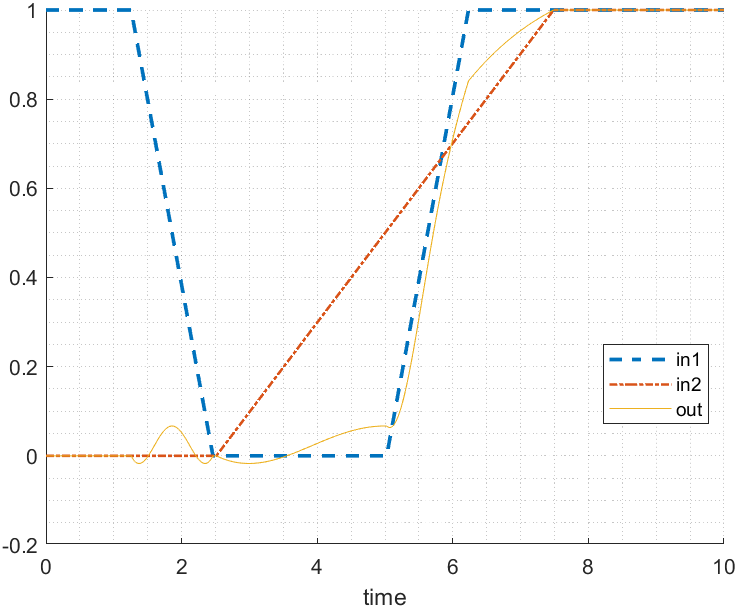


Рисунок 8 – Графики исходной и аппроксимированной функции (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. Реализации логической функций «или»

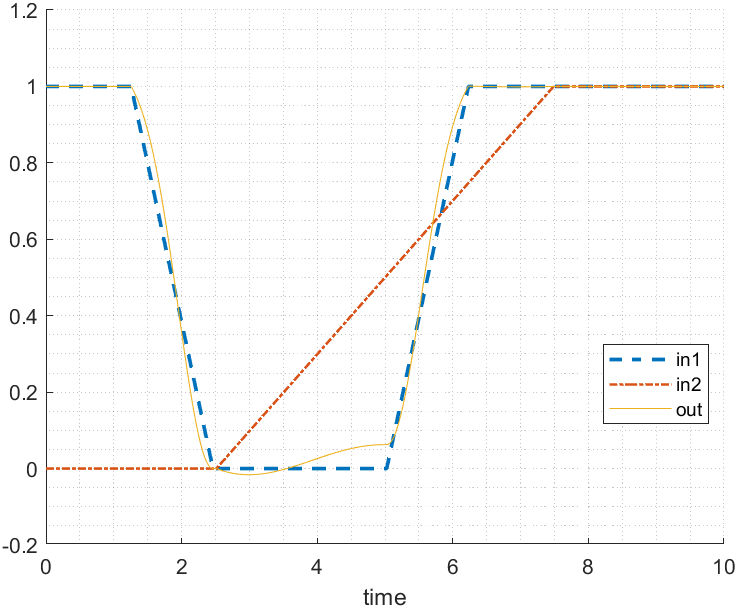
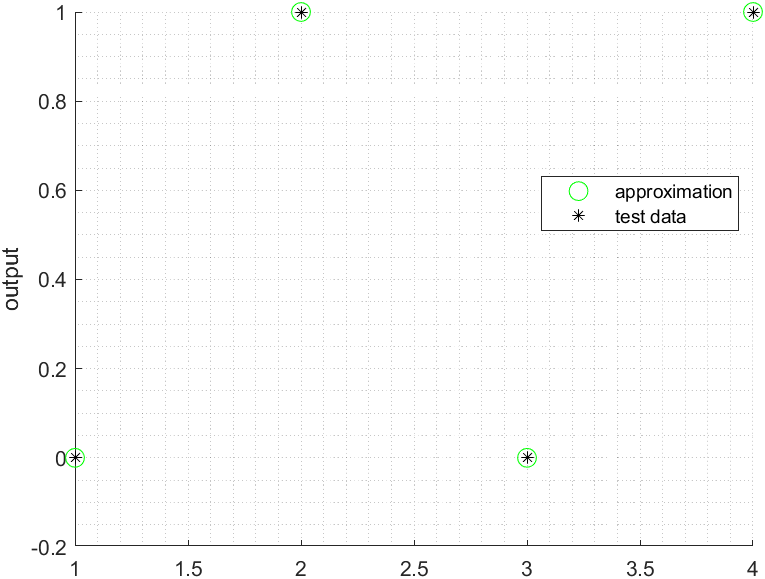


Рисунок 9 – Графики исходной и аппроксимированной функции (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. Реализации логической функций «исключающее или»

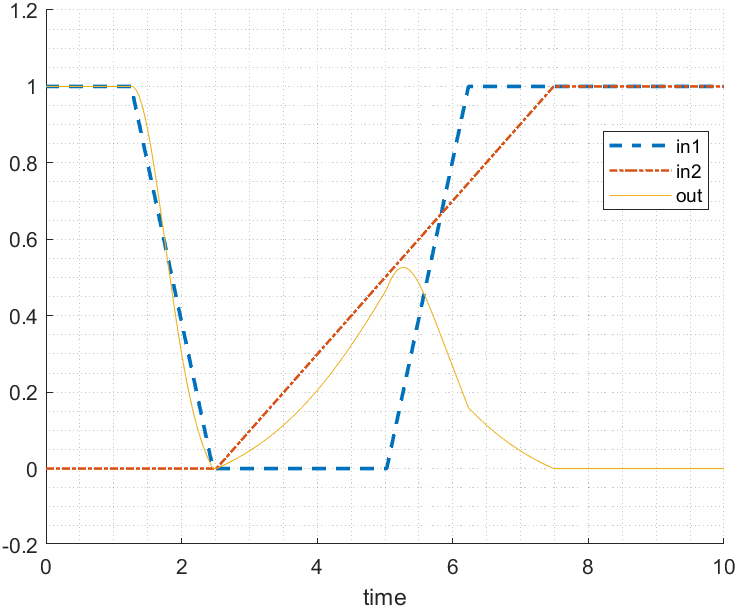
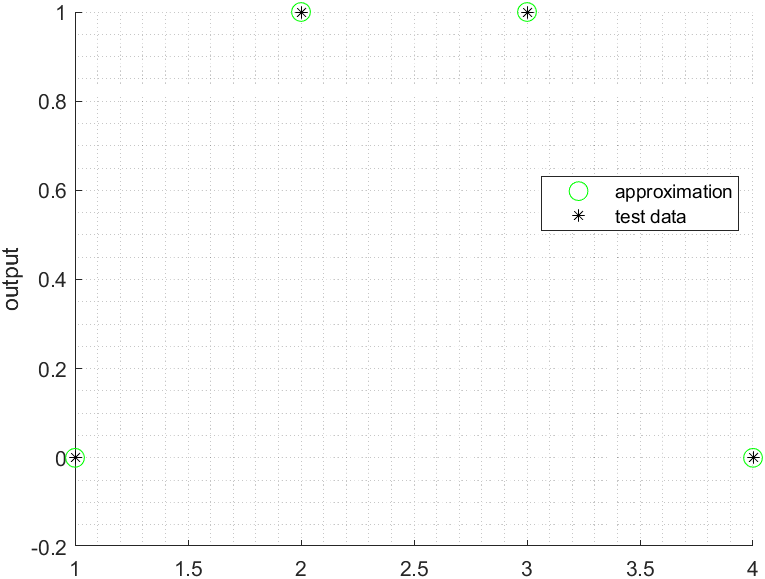


Рисунок 10 – Графики исходной и аппроксимированной функции (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. Формирование заданной функции с помощью RBF-сети, изменяя: число данных обучающей выборки, величину ошибки, разброс параметров.

Формирование заданной функции с помощью RBF-сети при исходных настройках (рисунок 11).

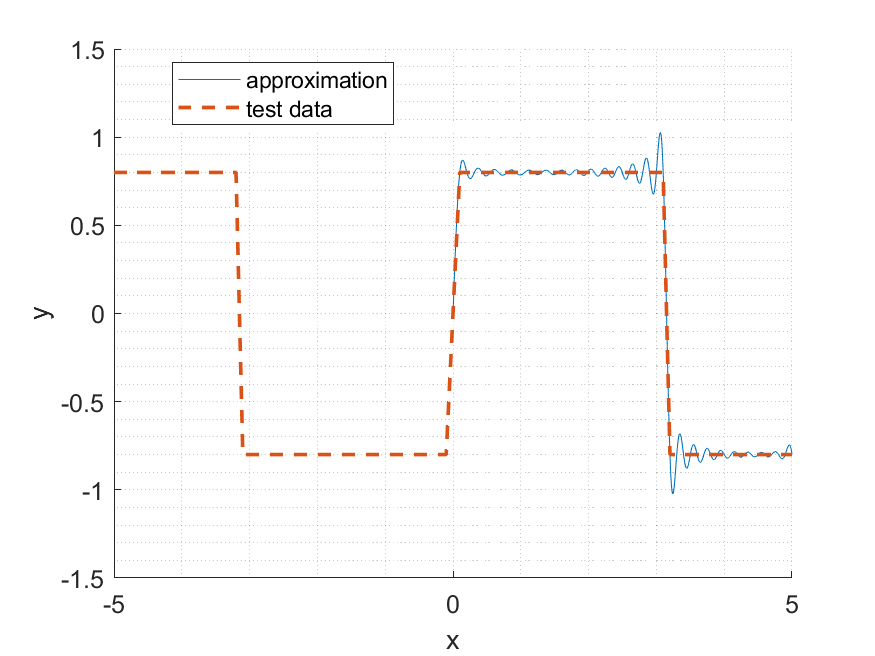
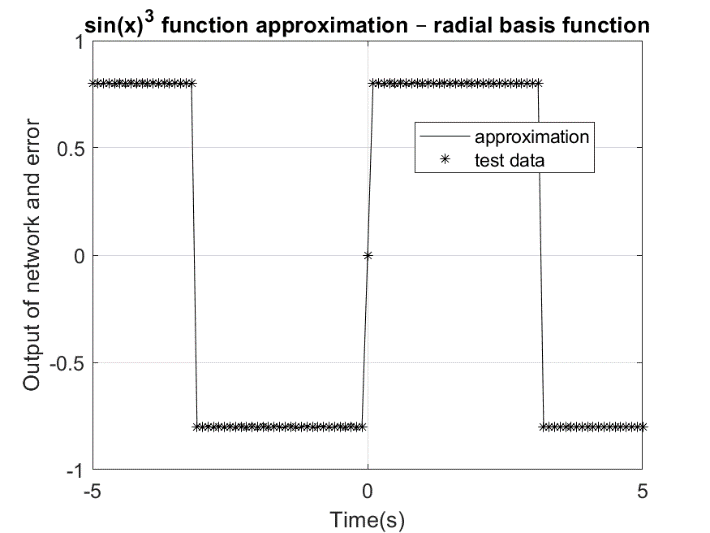


Рисунок 11 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

На рисунках 12-14 сформированы функции с помощью RBF-сети при изменении:

1. Числа данных обучающей выборки

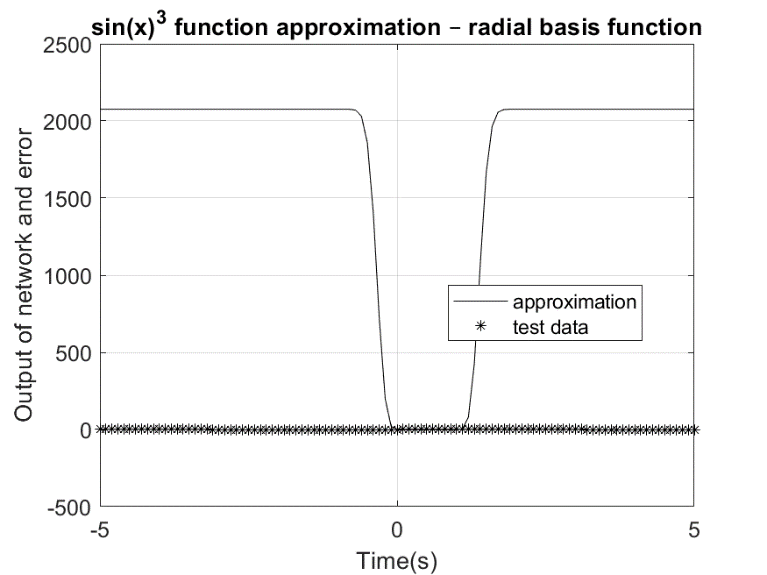
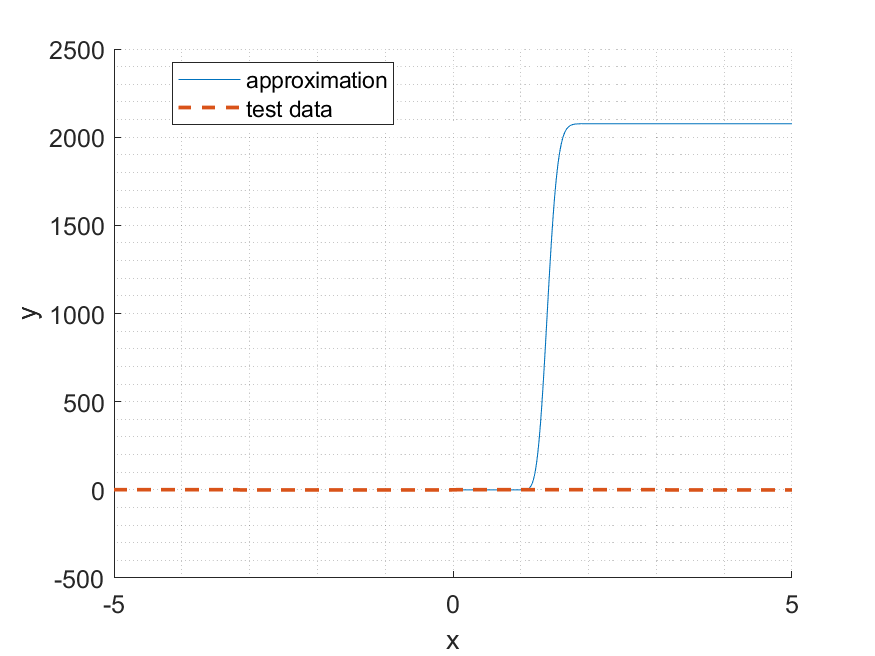
 

Рисунок 12 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

1. Величины ошибки

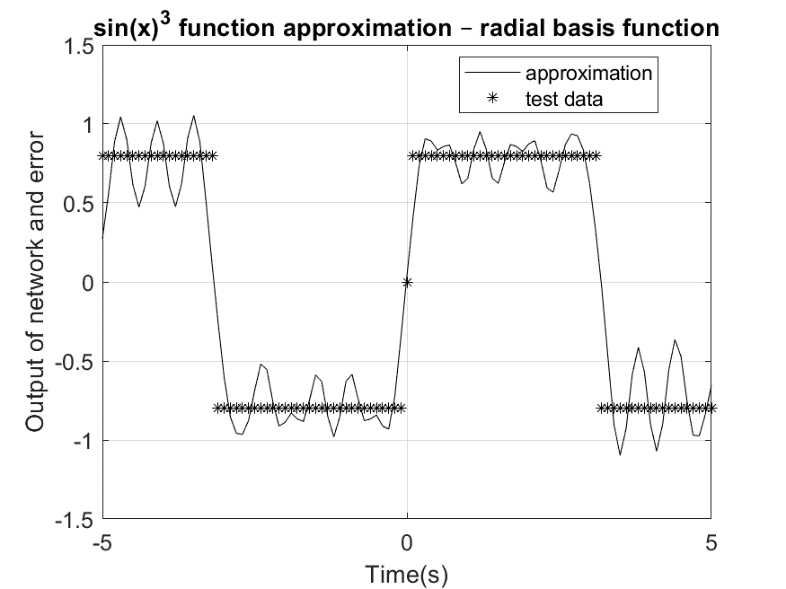
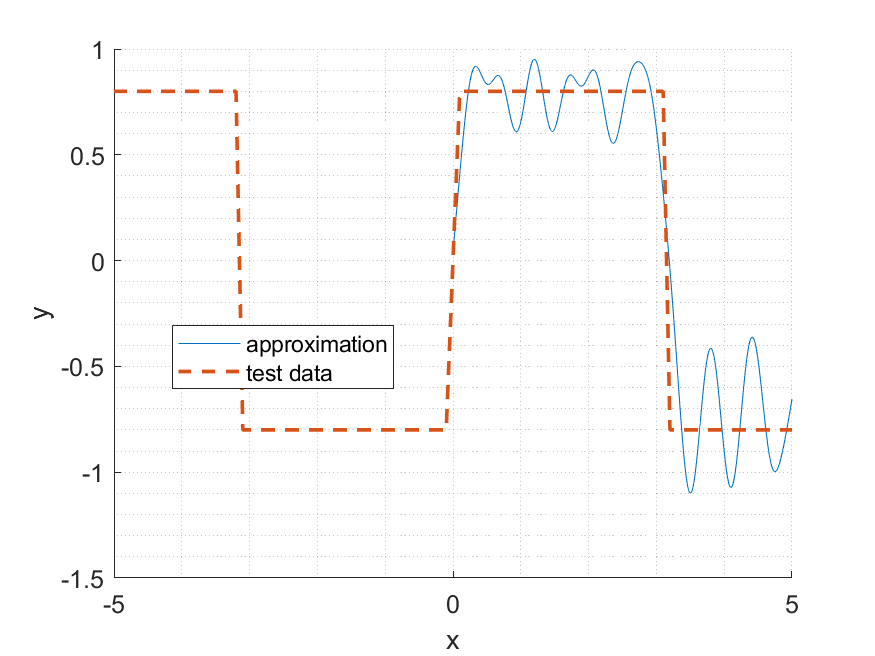
 

Рисунок 13 – Графики исходного и аппроксимированного сигнала

1. Разброса параметров

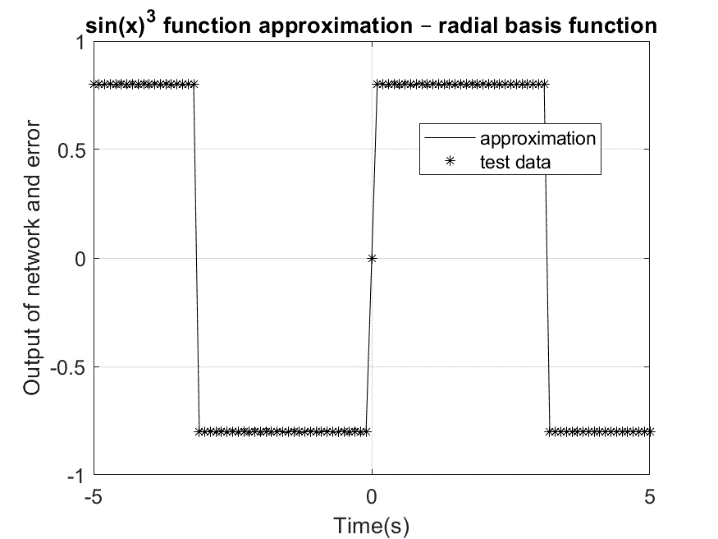
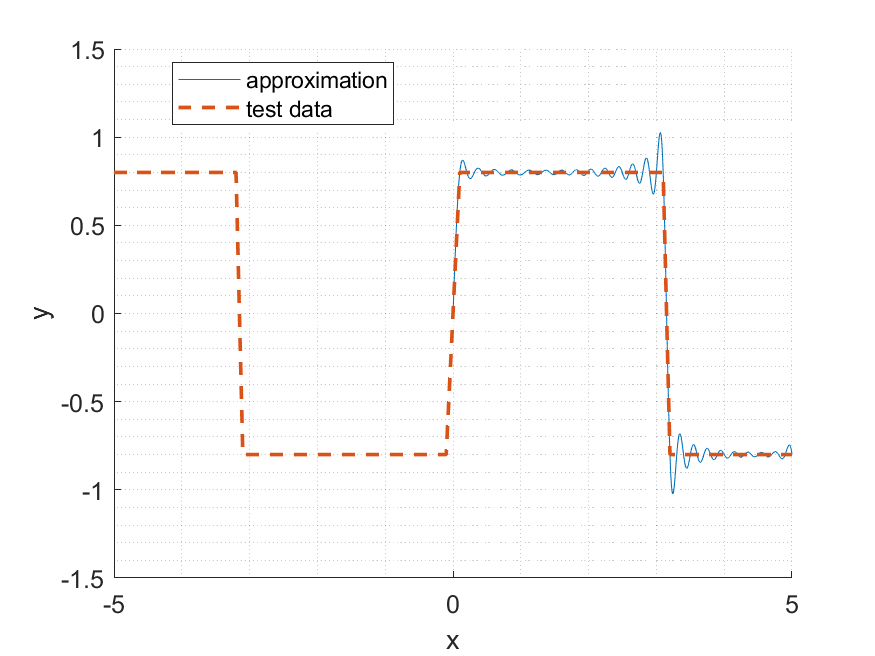
 

Рисунок 14 – Графики исходного и аппроксимированного сигнала (слева – ручное сравнение, справа – сравнение через Simulink-модель)

Из графиков видно, что формирование заданной функции с помощью RBF-сети, происходит плохо в случае уменьшения числа данных обучающей выборки и увеличения ошибки (0.0001 0.001).

Схемы слоя сети прямого распространения и RBF-сети представлены на рисунках 15 и 16.

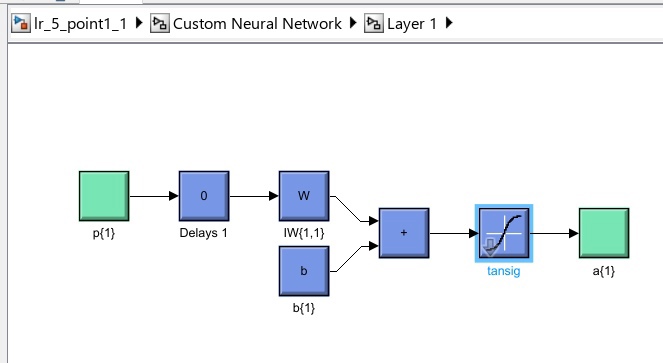


Рисунок 15 – Слой сети прямого распространения

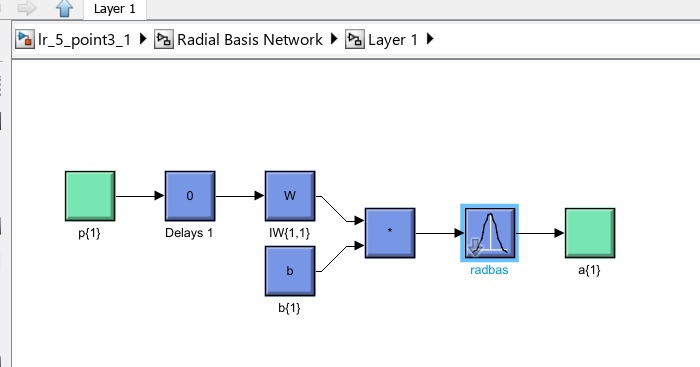


Рисунок 16 – Слой сети RBF

Как видно из рисунков, ключевым отличием является разная функция активации.

**Вывод:**

В ходе лабораторной работы был изучен пакет *Neural Networks Toolbox* системы (MATLAB); были рассмотрены способы формирования и обучения сетей прямого распространения и RBF-сетей; была исследована эффективность нейросетевого подхода для формирования заданных функций. Анализ показало, что использование функций Matlab показывает такие же результаты, как и моделирование в среде Simulink. Функции, разработанные с помощью RBF-сети, показывают, что при увеличении ошибки и уменьшения числа данных обучающей выборки качество обучения ухудшается.

В ходе работы было доказано, что при изменении алгоритма обучения, числа данных обучающей выборки, числа ИН в скрытом слое и числа слоев сети исходный и аппроксимируемый сигнал практически идентичны. При уменьшении числа циклов обучения аппроксимация происходит плохо.

**Приложение А**

Скрипт лабораторной работы

clc, clear, close all

flag = 0;

mkdir lr5\

fld = "lr5\";

version = 'R2020a';

sim\_mdl\_create = 1 % !!!!!!!!!!!!!!!!!!

% Аппроксимация функции

x = (-5:0.1:5); % данные входа

y = 0.8\*sign(sin(x)); % данные выхода

P = x; T = y;

model\_name\_no\_fld = @(point, i) "lr\_5\_point" + string(point) + "\_" + string(i);

model\_name = @(point, i) fld + model\_name\_no\_fld(point, i);

% return

%% 1. Формирование заданной функции с точностью до 0.1%

% close all, clc

% а) число данных обучающей выборки;

% б) число слоев сети;

% в) число ИН в скрытом слое;

% г) тип функция активации скрытого слоя:

% (tansig (гиперболический тангенс); hardlim (пороговая); logsig (сигмоидальная);

% hardlims (знаковая); satlin (полулинейная с насыщением); tribas (треугольная));

% purelin (линейная);

% д) число циклов обучения (net.trainParam.epochs);

% е) алгоритм обучения: traingd (алгоритм обучения обратного распространения ошибки);

% trainbr (байесовская регуляризация, алгоритм Левенберга-Марквардта);

% trainlm (алгоритм оптимизации Левенберга-Марквардта).

params.n\_data.P = P; % а

params.n\_data.T = T;

params.n\_neuron = 27; % в

params.n\_layer = [params.n\_neuron, 1]; % б, or [params.n\_neuron, params.n\_neuron, 1]

params.t\_fun = {'tansig', 'purelin'}; % г

params.epochs = 1000; % д

params.alg = 'trainbr'; % e

field = {

"default", ...

"n\_data", ...

["n\_layer", "t\_fun"], ...

"n\_neuron", ...

"t\_fun", ...

"epochs", ...

"alg"

};

value = {

"default", ...

struct('P', x(1:round(end/2)), 'T', y(1:round(end/2))), ... % а

{[params.n\_neuron, 1] {'tansig', 'tansig', 'purelin'}}, ... % б

15, ... % в

{'hardlim', 'purelin'}, ... % г

2, ... % д

'trainlm' % е

};

for i = 1:length(field)

% for i = 1

changed\_params = params;

if (length(field{i}) == 1)

changed\_params.(field{i}) = value{i};

else

for j=1:length(field{i})

changed\_params.(field{i}{j}) = value{i}{j};

end

end

changed\_params

% changed\_params.(field{i})

% Создание и тестирование нейронной сети

net = newff( ...

[changed\_params.n\_data.P(1) changed\_params.n\_data.P(end)], ...

[changed\_params.n\_layer], ...

changed\_params.t\_fun, ...

changed\_params.alg);

% newff– сеть прямого распространения(feedforward);

% [–1 2] – матрица минимальных и максимальных значений входных элементов;

% [5, 1] – матрица, указывающая на число ИН в слоях:

% 5 – число ИН в первом скрытом слое, 1 – число ИН в выходном слое;

% {'tansig', 'purelin'}

% 'tansig' – функция активации скрытого слоя(гиперболический тангенс).

% Варианты: tansig; hardlim; logsig; hardlims; satlin; tribas;

% 'purelin' – линейная функция активации выходного слоя;

% 'traingd' – алгоритм обучения обратного распространения ошибки;

% Варианты: 'trainbr'; 'trainlm';

net.trainParam.show = 400; % результаты выводить через 400 итераций;

net.trainParam.lr = 0.05; % скорость обучения;

net.trainParam.epochs = changed\_params.epochs; % количество циклов обучения;

net.trainParam.goal = 1e-3; % заданная ошибка обучения;

% Обучение сети

net1 = train(net, P, T);

% Тестирование сети

a = sim(net1, P);

% Создание графиков исходного и аппроксимированного сигнала

figure %

plot(P, a, 'k-'); grid; hold on;

xlabel('time(s)'); ylabel('output'); title('sin(x)^3 function')

plot(P, T, 'k\*')

legend('approximation', 'test data', 'Location', 'best')

print(fld + "plot\_point1\_1\_" + string(i), '-dpng');

if (sim\_mdl\_create)

% формирование нейросетевой модели в Toolbox Simulink;

create\_slx(flag, net1, model\_name(1, i), version)

ts = timeseries(P, P); %

out = sim((model\_name(1, i) + ".slx"), ...

'LoadExternalInput', 'on', ...

'ExternalInput', 'ts', 'StopTime', 'P(end)', ...

"MaxStep", "P(end)/2000")

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

figure, hold on, grid minor, xlabel('x'), ylabel('y')

plot(out.tout, out.y1) % !

plot(P, T, '--', 'LineWidth', 1.75);

legend('approximation', 'test data', 'Location', 'best');

print(fld + "plot\_point1\_2\_" + string(i), '-dpng');

end

end

% return

%% 2. RBF - сеть

% close all, clc

% реализация «и», «или», «исключающее или»

P\_ = {

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

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

[0 1 0 1; 0 0 1 1]

};

T\_ = {

[0 0 0 1], ...

[0 1 0 1], ...

[0 1 1 0]

};

% Реализация логических функций функции

% Создание и тестирование нейронной сети с радиальными базисными элементами;

goal = 0; % среднеквадратичная ошибка;

spread = 0.5; % разброс параметров;

% for i = 3

for i = 1:3

net = newrb(P\_{i}, T\_{i}, goal, spread);

% Тестирование сети;

a = sim(net, P\_{i})

figure, hold on, grid minor, ylabel('output') % !

plot(a, 'go', MarkerSize=9);

% xlabel('time(s)');

plot(T\_{i}, 'k\*')

legend('approximation', 'test data', 'Location', 'best')

print(fld + "plot\_point2\_1\_" + string(i), '-dpng');

if (sim\_mdl\_create)

% формирование нейросетевой модели в Toolbox Simulink;

create\_slx(flag, net, model\_name(2, i), version)

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));

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

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

out = sim(model\_name(2, i) + ".slx", ...

'LoadExternalInput', 'on', 'ExternalInput', 'ts', ...

'StopTime', 't(end)', ...

"MaxStep", "t(end)/2000")

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

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

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

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

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

plot(out.tout, out.y1) % !

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

print(fld + "plot\_point2\_2\_" + string(i), '-dpng');

end

end

% return

%% 3. Формирование заданной функции с помощью RBF-сети

% close all, clc

params\_RBF.n\_data.P = P; % а

params\_RBF.n\_data.T = T;

params\_RBF.goal = 0.0001;

params\_RBF.spread = 0.2;

% flag = 1

field = {

"default", ...

"n\_data", ... % число данных обучающей выборки,

"goal", ... % величину ошибки,

"spread" % разброс параметров

};

value = {

"deafault", ...

struct('P', x(1:round(end/2)), 'T', y(1:round(end/2))), ...

0.001, ...

0.9

};

for i = 1:length(field)

% for i = 1

changed\_params\_RBF = params\_RBF;

changed\_params\_RBF.(field{i}) = value{i};

% Создание и тестирование нейронной сети с радиальными базисными элементами

net = newrb( ...

changed\_params\_RBF.n\_data.P, ...

changed\_params\_RBF.n\_data.T, ...

changed\_params\_RBF.goal, ...

changed\_params\_RBF.spread);

%Тестирование сети

a = sim(net, P);

% Создание графиков исходного и аппроксимированного сигнала

figure % !

plot(P, a, 'k-'); grid; hold on;

plot(P, T, 'k\*');

xlabel('Time(s)'); ylabel('Output of network and error');

title ('sin(x)^3 function approximation – radial basis function')

legend('approximation', 'test data', 'Location', 'best')

print(fld + "plot\_point3\_1\_" + string(i), '-dpng');

if (sim\_mdl\_create)

% создание модели сети в Toolbox Simulink

% SSE Sum squared error performance function (суммарная квадратичная ошибка).

create\_slx(flag, net, model\_name(3, i), version);

ts = timeseries(P, P); % (y, x)

out = sim(model\_name(3, i) + ".slx", 'StopTime', 'P(end)', ...

'LoadExternalInput', 'on', 'ExternalInput', 'ts', ...

"MaxStep", "P(end)/2000")

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

figure, hold on, grid minor, xlabel('x'), ylabel('y')

plot(out.tout, out.y1) % !

plot(P, T, '--', 'LineWidth', 1.75)

legend('approximation', 'test data', 'Location', 'best')

print(fld + "plot\_point3\_2\_" + string(i), '-dpng');

end

end

%%

function [] = create\_slx(flag, net1, model\_name, version)

if (flag || ~exist(model\_name + ".slx", 'file'))

% "Name: " + model\_name

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

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

set\_param(sysName + "/y1", 'MaxDataPoints', '+inf')

if (1)

% close\_system("lr\_5\_point1\_1.slx")

save\_system(sysName, model\_name);

close\_system(model\_name + ".slx");

else

save\_system(sysName, "temp");

% open\_system(model\_name(3333, i))

% set\_param(model\_name\_no\_fld(3333, i),"SolverType","Variable-step")

Simulink.exportToVersion( ...

"temp", ...

model\_name + ".slx", ...

version);

close\_system("temp");

delete("temp" + ".slx");

end

end

end