Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 8

по курсу «Нейроинформатика»

Тема: Динамические сети

Студент: Куликов А.В.
Группа: М80-408Б-17
Преподаватель: Аносова Н.П.
Дата: 10 января 2021
Оценка:

Москва, 2020

Цель работы: исследование свойств некоторых динамических нейронных сетей, алгоритмов обучения, а также применение сетей в задачах аппроксимации функций и распознавания динамических образов.

Основные этапы работы:

1. Использовать сеть прямого распространения с запаздыванием для предсказания значений временного ряда и выполнения многошагового прогноза.
2. Использовать сеть сеть прямого распространения с распределенным запаздыванием для распознавания динамических образов.
3. Использовать нелинейную авторегрессионную сеть с внешними входами для аппроксимации траектории динамической системы и выполнения многошагового прогноза.

Оборудование:

Процессор: AMD Ryzen 5 Mobile 3550H

Объем оперативной памяти: 8 Гб

Программное обеспечение:

Python 3.8.5, MATLAB r2020

Сценарий выполнения работы:

Задание №1

```
clear;
clc;

% Считывание датасета из файла
fd = fopen('data.txt','r');
formatSpec = '%f %f';
sizeData = [1 Inf];
data = fscanf(fd, formatSpec, sizeData).';
fclose(fd);

% Сглаживание данных
x = smooth(data, 12);

% График исходные/сглаженные данные
figure;
hold on;
grid on;
plot(data, '-b');
plot(x, '-r');

% Формирование обучающего множества
D = 5;
ntrain = 500;
nval = 100;
ntest = 50;

trainInd = 1 : ntrain; % 1..500
valInd = ntrain + 1 : ntrain + nval; % 501..600
testInd = ntrain + nval + 1 : ntrain + nval + ntest; % 601..650
```

```
x_train = x(trainInd);
x_val = x(valInd);
x_test = x(testInd);

% Создание и конфигурация сети
hiddenSizes = 10;
net = timedelaynet(1:D, hiddenSizes, 'trainlm'); %1:10 delay, hid. l. size.
net.divideFcn = '';

x_train_seq = con2seq(x_train');
x_val_seq = con2seq(x_val');
x_test_seq = con2seq(x_test');

net = configure(net, x_train_seq, x_train_seq);
net = init(net);

net.trainParam.epochs = 2000;
net.trainParam.max_fail = 2000;
net.trainParam.goal = 1.0e-5;

% Обучение сети
[Xs, Xi, Ai, Ts] = preparets(net, x_train_seq, x_train_seq);
net = train(net, Xs, Ts, Xi, Ai);

% Рассчет выхода сети для обучающего множества
Y = sim(net, Xs, Xi);

% График выхода сети на обучающем множестве
figure;
hold on;
grid on;
plot(x_train, '-b');
plot([cell2mat(Xi) cell2mat(Y)], '-r');

% График ошибки на обучающем множестве
figure;
hold on;
grid on;
plot([cell2mat(Xi) cell2mat(Y)] - x_train', '-r');

% Рассчет выхода сети для контрольного множества
[Xs, Xi, Ai, Ts] = preparets(net, x_val_seq, x_val_seq);
Y = sim(net, Xs, Xi);

% График выхода сети на контрольном множестве
figure;
hold on;
grid on;
plot(x_val, '-b');
plot([cell2mat(Xi) cell2mat(Y)], '-r');
```

Структура сети и ее обучение

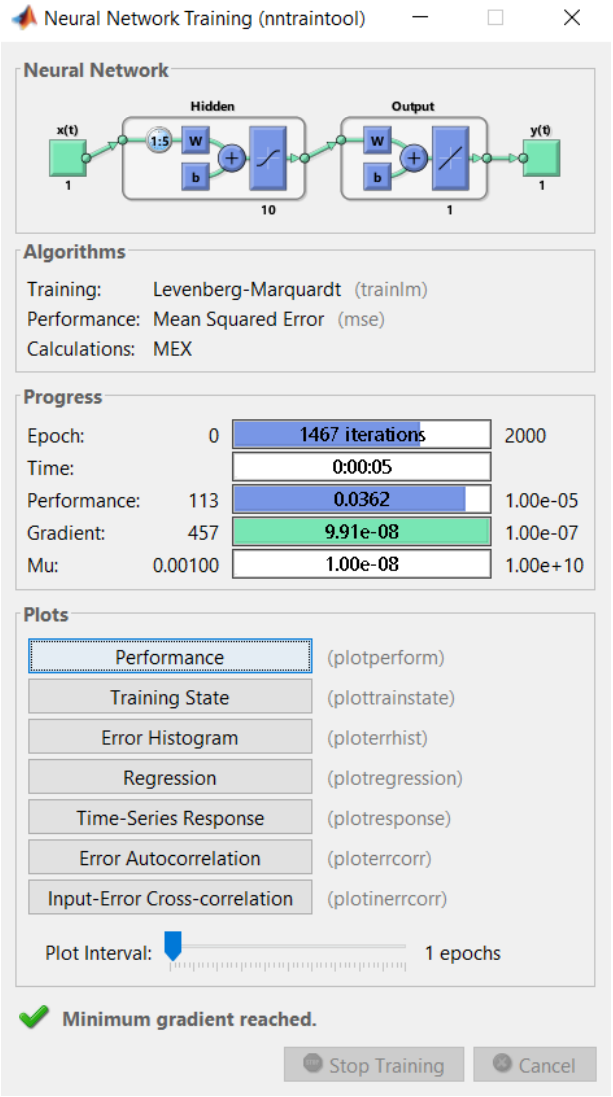


График исходные/сглаженные данные

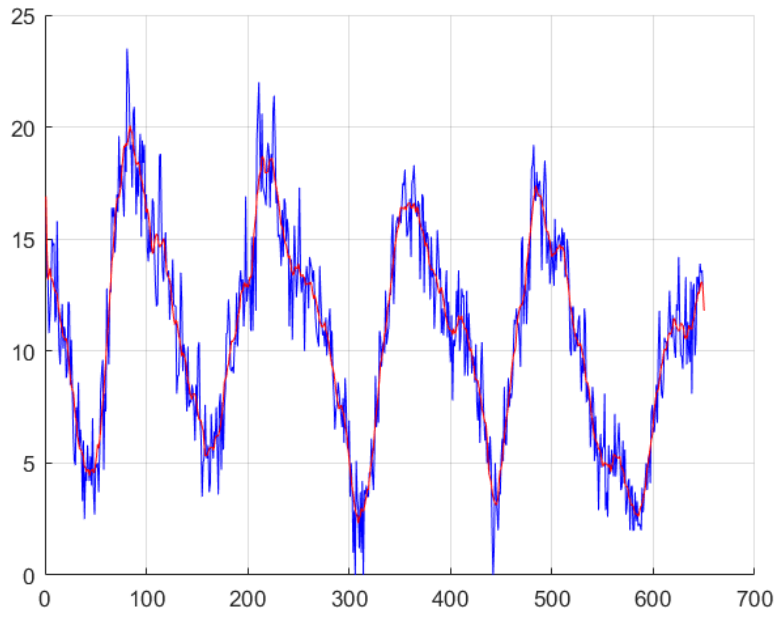
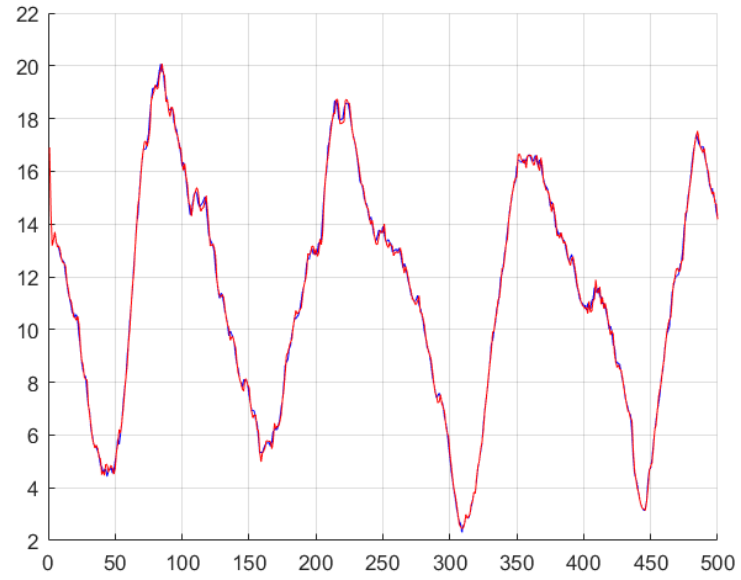


График выхода сети на обучающем множестве



Красная линия – выход сети. Синяя – обучающее множество.

График ошибки на обучающем множестве

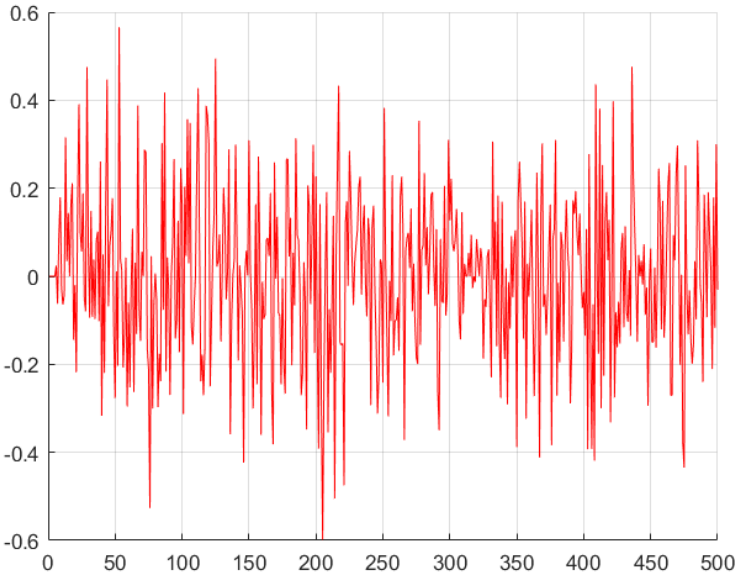
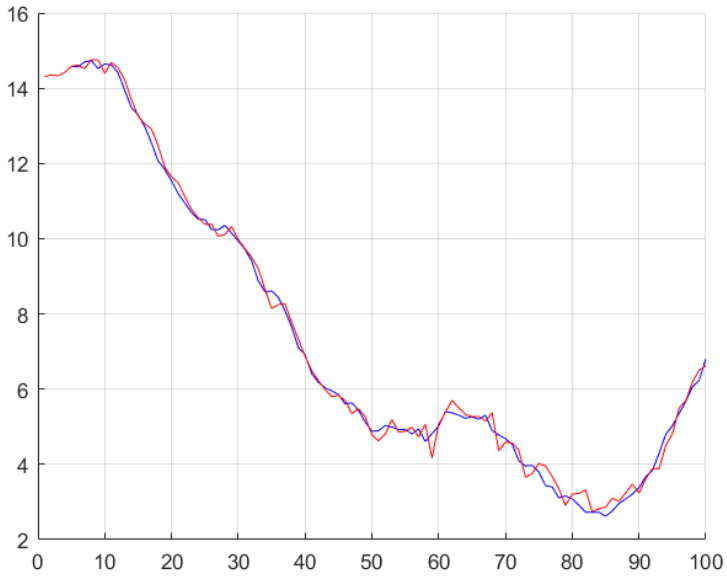


График выхода сети на контрольном множестве



Красная линия – выход сети. Синяя – контрольное множество.

Задание №2

```
clear;
clc;

% Создание обучающего множества
k1 = 0 : 0.025 : 1;
p_1 = sin(4 * pi * k1);
t_1 = -ones(size(p_1));

k2 = 2.38 : 0.025 : 4.1;
g = @(k)cos(cos(k) .* k .* k + 5*k);

p_2 = g(k2);
t_2 = ones(size(p_2));
R = {1; 3; 5};
P = [repmat(p_1, 1, R{1}), p_2, repmat(p_1, 1, R{2}), p_2, repmat(p_1, 1,
R{3}), p_2];
T = [repmat(t_1, 1, R{1}), t_2, repmat(t_1, 1, R{2}), t_2, repmat(t_1, 1,
R{3}), t_2];
Pseq = con2seq(P);
Tseq = con2seq(T);

% Создание и конфигурация сети
D = 4;

net = distdelaynet({0 : D, 0 : D}, 8, 'trainoss');
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
net.divideFcn = '';

net = configure(net, Pseq, Tseq);

[Xs, Xi, Ai, Ts] = preparets(net, Pseq, Tseq);

net.trainParam.epochs = 100;
net.trainParam.goal =  1.0e-5;
```

```
% Обучение сети
net = train(net, Xs, Ts, Xi, Ai);

% Рассчет выхода сети для обучающего множества
Y = sim(net, Xs, Xi, Ai);

% График обучающего множества и выход сети до порогового элемента
figure;
hold on;
grid on;
plot(cell2mat(Ts), '-b');
plot(cell2mat(Y), '-r');

Yc = sign(cell2mat(Y));

fprintf('Correctly recognized (train set): %d\\%d\n', nnz(Yc == T(D+1 :
end)), length(T)-D-1);

% Формирование тестового множества
R = {1; 4; 5};
P = [repmat(p_1, 1, R{1}), p_2, repmat(p_1, 1, R{2}), p_2, repmat(p_1, 1,
R{3}), p_2];
T = [repmat(t_1, 1, R{1}), t_2, repmat(t_1, 1, R{2}), t_2, repmat(t_1, 1,
R{3}), t_2];

Pseq = con2seq(P);
Tseq = con2seq(T);

[Xs, Xi, Ai, Ts] = preparets(net, Pseq, Tseq);

% Рассчет выхода для тестового множетсва
Y = sim(net, Xs, Xi, Ai);

Yc = sign(cell2mat(Y));

fprintf('Correctly recognized (test set): %d\\%d\n', nnz(Yc == T(D+1 : end)),
length(T)-D-1);

% График тестового множества и выход сети до порогового элемента
figure;
hold on;
grid on;
pLine = plot(cell2mat(Ts), 'b');
rLine = plot(cell2mat(Y), 'r');

legend([rLine, pLine], 'Target', 'Predicted');
```

Структура сети и ее обучение

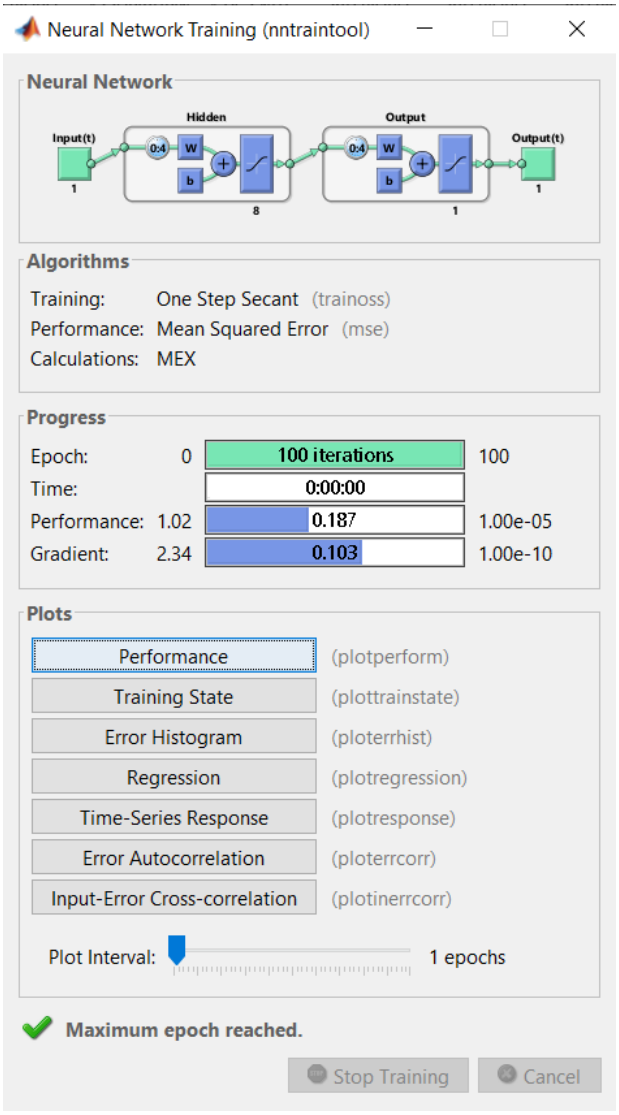
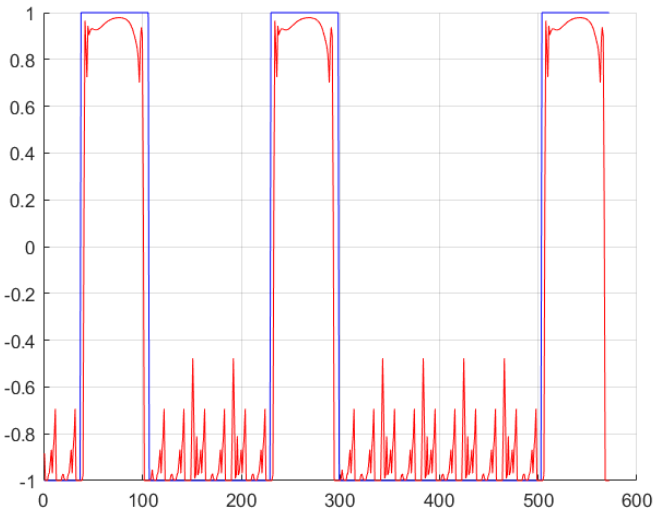
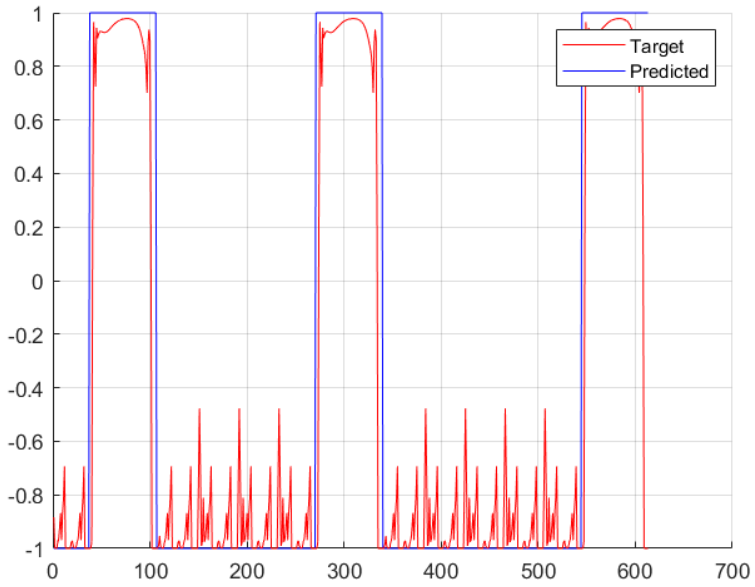


График обучающего множества и выход сети до порогового элемента



Синяя линия – обучающее множество. Красная – выход сети.

График тестового множества и выход сети до порогового элемента



Синяя линия – истинное значение функции. Красная – выход сети.

Задание №3

```
clear;
clc;

% Создание обучающего множества
t0 = 0;
tn = 10;
dt = 0.01;

n = (tn - t0) / dt + 1;

f = @(k)sin(k.^2 - 2 * k + 3);
f1 = @(y, u)y ./ (1 + y.^2) + u.^3;

u = zeros(1, n);

u(1) = f(0);
x = zeros(1, n);
for i = 2 : n
    t = t0 + (i - 1) * dt;
    x(i) = f1(x(i - 1), u(i - 1));
    u(i) = f(t);
end

% График управляющего и целевого сигнала
figure
subplot(2,1,1)
plot(t0:dt:tn, u, '-b'),grid
ylabel('control')
subplot(2,1,2)
plot(t0:dt:tn, x, '-r'), grid
ylabel('state')
xlabel('t')
```

```
D = 3;
ntrain = 700;
nval = 200;
ntest = 97;

trainInd = 1 : ntrain;
valInd = ntrain + 1 : ntrain + nval;
testInd = ntrain + nval + 1 : ntrain + nval + ntest;

% Создание и конфигурация NARX-сети
net = narxnet(1 : D, 1, 8);
net.trainFcn = 'trainlm';

net.divideFcn = '';
net.trainParam.epochs = 20000;
net.trainParam.max_fail = 2000;
net.trainParam.goal = 1.0e-5;

u_train = u(trainInd);
u_val = u(valInd);
u_test = u(testInd);

x_train = x(trainInd);
x_val = x(valInd);
x_test = x(testInd);

[Xs, Xi, Ai, Ts] = preparets(net, con2seq(u_train), {}, con2seq(x_train));

% Обучение сети
net = train(net, Xs, Ts, Xi, Ai);

% Рассчет выхода сети для обучающего множества
Y = sim(net, Xs, Xi);

t = t0:dt:tn;
train_range = t(1 : ntrain);

% График управляющего сигнала, истинного значения функции, выхода сети и
% ошибки на обучающем отрезке
figure
subplot(3,1,1)
plot(train_range, u(1:ntrain), '-b'),grid
ylabel('control')
subplot(3,1,2)
plot(train_range, x(1:ntrain), '-b', train_range, [x(1:D) cell2mat(Y)], '-
r'), grid
ylabel('state')
subplot(3,1,3)
plot(train_range(D+1:end), x(D+1:ntrain) - cell2mat(Y)), grid
ylabel('error')
xlabel('t')

[Xs, Xi, Ai, Ts] = preparets(net, con2seq([u_val(end-D+1:end) u_test]), {},
con2seq([x_val(end-D+1:end) x_test]));

% Рассчет выхода сети для тестового множества
Y = sim(net, Xs, Xi);

t = t0:dt:tn;
test_range = t(end-ntest-D+1:end);

% График управляющего сигнала, истинного значения функции, выхода сети и
% ошибки на тестовом отрезке
```

```
figure
subplot(3,1,1)
plot(t(end-ntest+1:end), u(end-ntest+1:end), '-b'),grid
ylabel('control')
subplot(3,1,2)
plot(t(end-ntest+1:end), x_test, '-b', t(end-ntest+1:end), cell2mat(Y), '-
r'), grid
ylabel('state')
subplot(3,1,3)
plot(t(end-ntest+1:end), x_test - cell2mat(Y)), grid
ylabel('error')
xlabel('t')
```

Структура сети и ее обучение

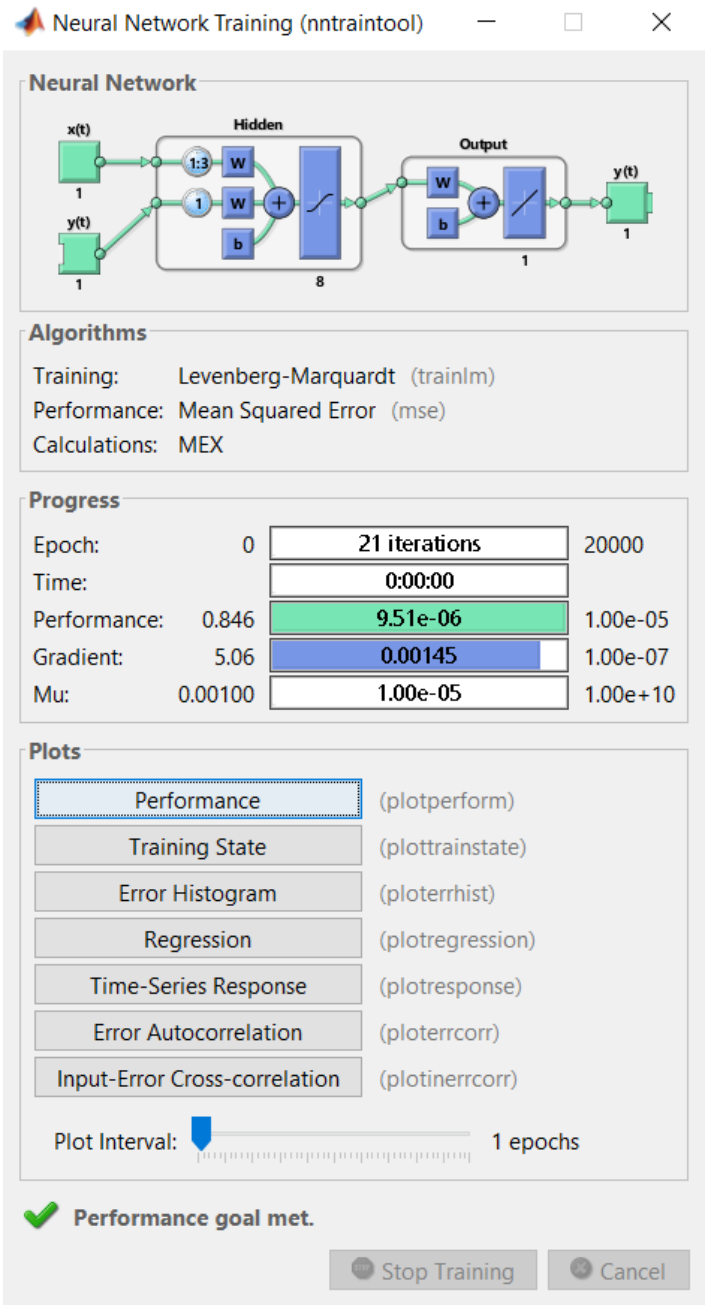


График управляющего и целевого сигнала

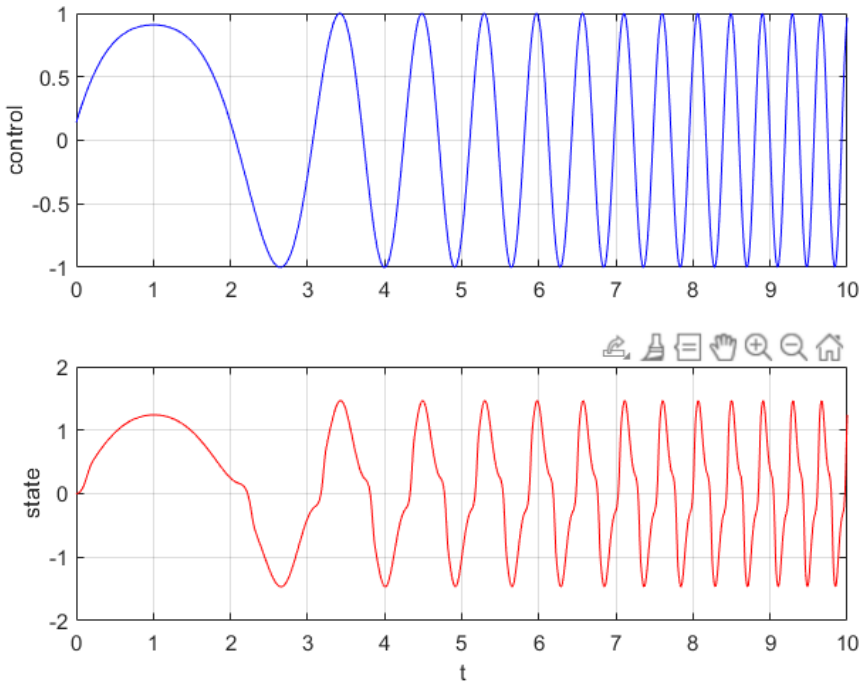


График управляющего сигнала, истинного значения функции, выхода сети и ошибки на обучающем отрезке

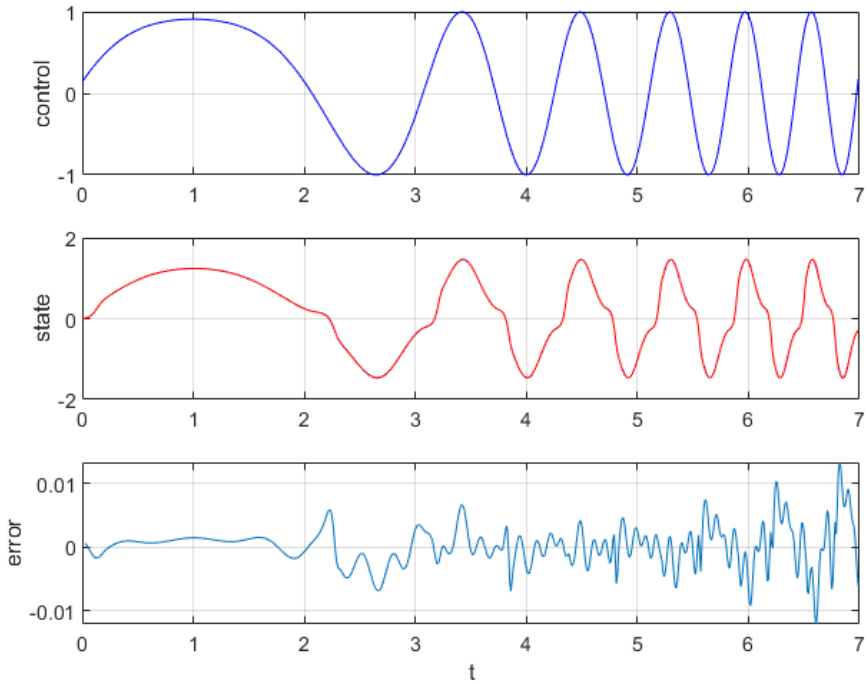
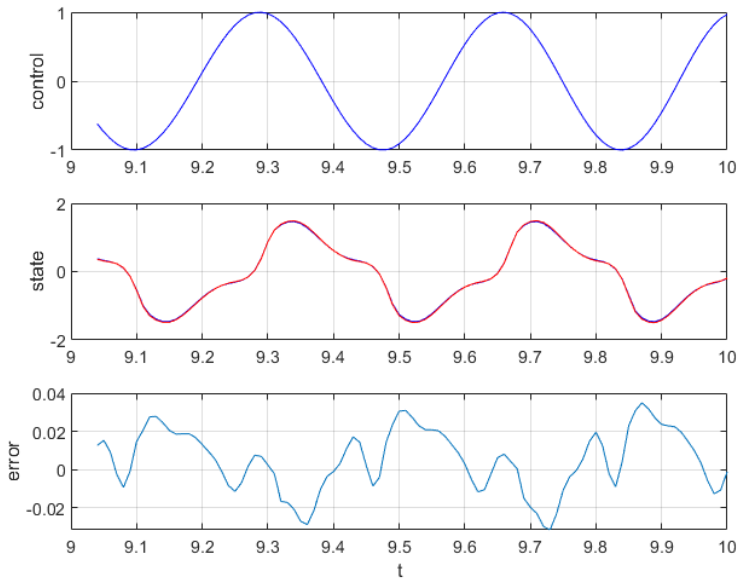


График управляющего сигнала, истинного значения функции, выхода сети и ошибки на тестовом отрезке



Выводы:

Динамические нейронные сети – сети, выход которых зависит не только от текущего входа сети, но и от ее предшествующих (по времени) входов, выходов и/или состояний.

Использованные в этой лабораторной сеть прямого распространения с запаздыванием (FTDNN), сеть прямого распространения с распределенным запаздыванием (DTDNN) и NARX-сети как раз такими и являются.

Динамические нейронные сети позволяют решать задачу многошагового прогноза, распознавания динамических образов, аппроксимации траектории динамической системы.