МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Кафедра вычислительной математики и программирования

Лабораторная работа №2
по спецкурсу «Нейроинформатика»

Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Выполнил: Днепров И.С.
Группа: М8О-407Б, вариант 10
Преподаватели: Тюменцев Ю.В.

Москва, 2020

Цель работы

Исследование свойств линейной нейронной сети и алгоритмов её обучения, применение сети в задачах аппроксимации и фильтрации.

Основные этапы работы

1. Использовать линейную нейронную сеть с задержками для аппроксимации функции. В качестве метода обучения использовать адаптацию.
2. Использовать линейную нейронную сеть с задержками для аппроксимации функции и выполнения многошагового прогноза.
3. Использовать линейную нейронную сеть в качестве адаптивного фильтра для подавления помех. Для настройки весовых коэффициентов использовать метод наименьших квадратов.

Оборудование

Процессор: 2,4 GHz Intel Core 2 Duo

Оперативная память: 8 ГБ 1067 MHz DDR3

Программное обеспечение

Matlab R2020b, 64-bit.

Сценарий выполнения работы

1. Задана временная последовательность $x(n)$. Построить и обучить линейную сеть с задержками, которая будет выполнять одношаговый прогноз для первой функции из варианта задания:

$$\hat{x}(n+1) = \sum_{i=1}^{D} w_i x(n-i+1) + b$$

где D задаёт глубину погружения временного ряда (*delays*), $\{w_i, b\}$ – весовые коэффициенты.

1.1. Построить обучающее множество: в качестве входного множества использовать значения первого входного сигнала на заданном интервале; преобразовать входное множество к последовательности входных образцов с помощью функции *con2seq*; эталонные выходы сети формируются из входной последовательности, чтобы сеть выполняла одношаговый прогноз.

$$x = \sin\left(-2t^2 + 7t\right) - \frac{1}{2}\sin(t), \quad t \in [0,4.5], h = 0.025$$

1.2. Создать сеть с помощью функции *newlin*. Задать задержки от 1 до $D = 5$. Задать скорость обучения равной 0.01.

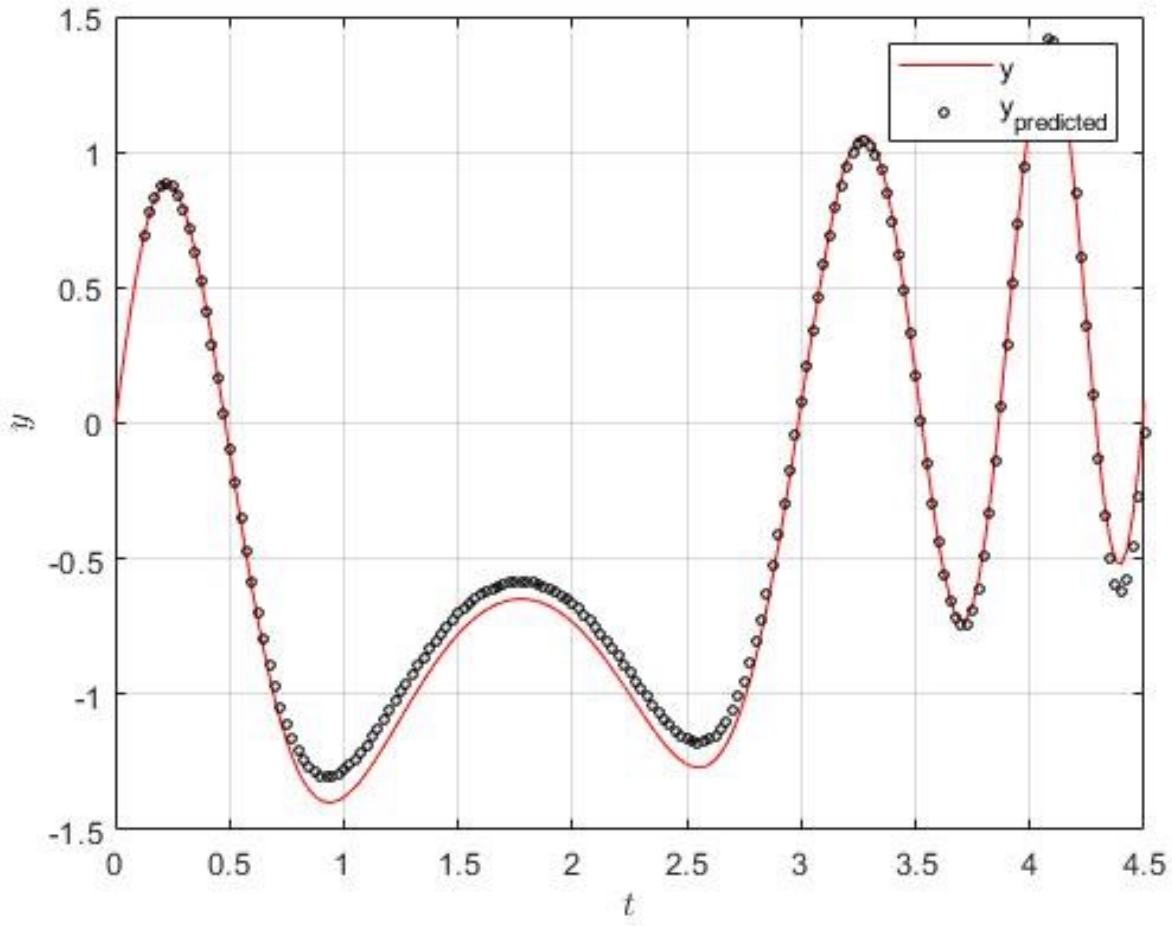
1.3. Инициализировать сеть случайными значениями.

1.4. Выполнить адаптацию с числом циклов равным 50. Занести в отчет величину ошибки обучения с помощью *sqrt*(*mse*). Поскольку сеть имеет задержки, то в функцию адаптации необходимо отдельно передать первые 5 элементов входной последовательности для инициализации задержек (входной параметр *Pi*). В противном случае задержки будут инициализированы нулями, что приведет к увеличению ошибки обучения при выполнении адаптации. В дальнейшем использовать входную и выходную последовательности, начиная с 6 элемента.

```
|iteration: 1 | error: 0.629274|
|iteration: 2 | error: 0.037495|
|iteration: 3 | error: 0.036122|
|iteration: 4 | error: 0.035642|
|iteration: 5 | error: 0.035379|
|iteration: 6 | error: 0.035201|
|iteration: 7 | error: 0.035067|
|iteration: 8 | error: 0.034959|
|iteration: 9 | error: 0.034869|
```

```
|iteration: 10 | error: 0.034792|
. . . . . . . . . . . . . . . . . . . .
|iteration: 40 | error: 0.033577|
|iteration: 41 | error: 0.033541|
|iteration: 42 | error: 0.033506|
|iteration: 43 | error: 0.033471|
|iteration: 44 | error: 0.033435|
|iteration: 45 | error: 0.033400|
|iteration: 46 | error: 0.033365|
|iteration: 47 | error: 0.033330|
|iteration: 48 | error: 0.033295|
|iteration: 49 | error: 0.033260|
|iteration: 50 | error: 0.033225|
```

1.5. Отобразить на графике эталонные значения и предсказанные сетью. График занести в отчет.



2. Для временной последовательности из задания 1 обучить линейную сеть с задержками (линейный адаптивный фильтр) и выполнить многошаговый прогноз.

   2.1. Построить обучающее множество: в качестве входного множества использовать значения первого входного сигнала на заданном интервале; преобразовать входное множество к последовательности входных образцов с помощью функции *con2seq*; эталонные выходы сети формируются из входной последовательности, чтобы сеть выполняла одношаговый прогноз.

   2.2. Создать сеть с помощью функции *newlin*. Задать задержки от 1 до $D == 3$. Задать скорость обучения с помощью функции *maxlinlr*(*cell2mat*(*P*),'*bias*').

   2.3. Инициализировать сеть случайными значениями.

   2.4. Задать параметры обучения: число эпох обучения (*net.trainParam.epochs*) равным равными 600, предельное значение критерия обучения (*net.trainParam.goal*) равным 10^{-6}. Также необходимо проинициализировать задержки *Pi*. Выполнить обучение сети с помощью функции *train*.

   2.5. Занести в отчет весовые коэффициенты и смещение. Занести в отчет окно *Performance* и *Neural Network Training*. Отразить структуру сети и проведенное обучение в отчете, заполнив таблицу 1.

Функция создания сети: newline

Входной слой: 1

Скрытый слой: нет

Выходной слой: 1

Активационные функции: линейная

Динамика: минимизация mae

Функция разделения обучающего множества: задержки

Число примеров в подмножествах: не задавал

Метод обучения: train

Параметры обучения:

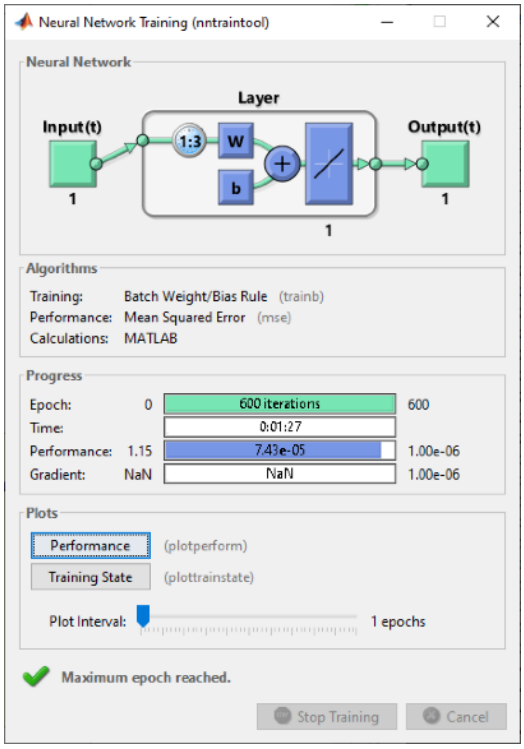
```
network.trainParam.epochs = 600

network.trainParam.goal = 1e-6
```

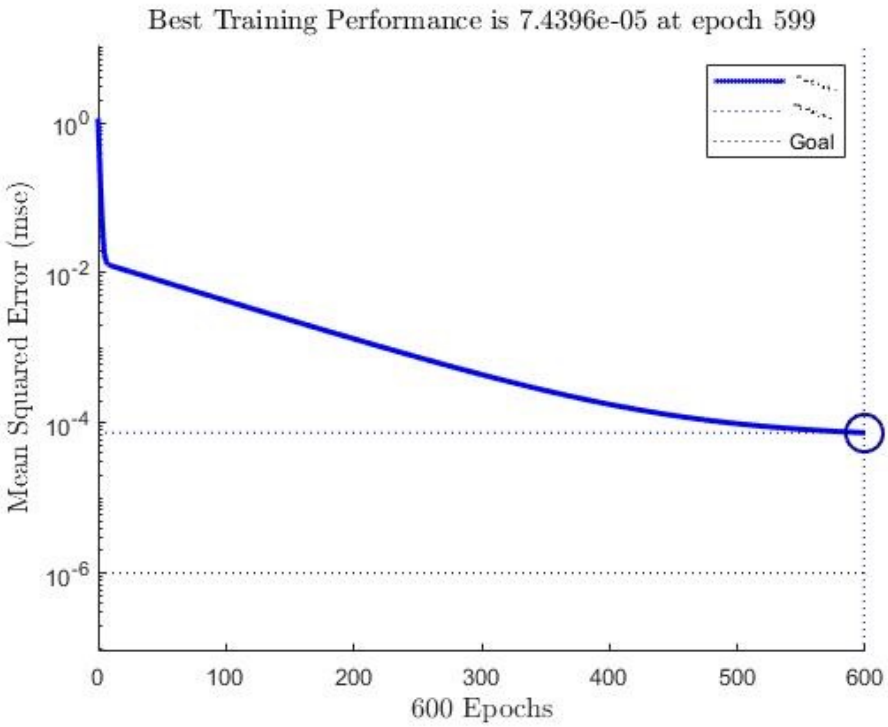
Метод инициализации сети: `rands`

Критерий окончания обучения: `epochs = 600 | goal = 1e-6`

Причина окончания обучения: `epochs = 600`

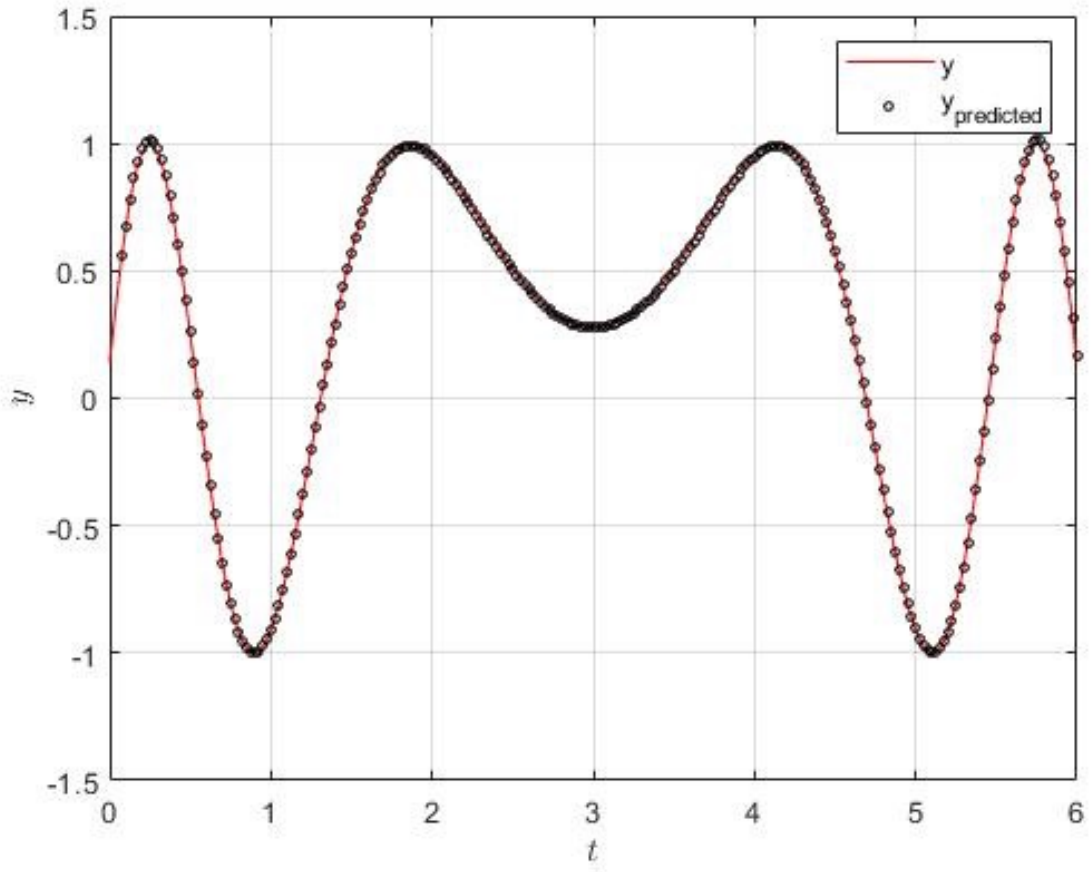
Число эпох обучения: 600





2.6. Рассчитать выход сети (*sim*) для обучающего множества. Сравнить выход сети с соответствующим эталонным множеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью, а также ошибку обучения. Графики занести в отчет.

```
R^2: 0.432275

MSE: 0.023011

RMSE: 0.151695

Относительная СКО: 24.418777%

MAE: 0.124227

min abs err: 0.010371

max abs err: 0.273873

MAPE: 17.053237

Доля с ошибкой менее 5%: 10.000000%

Доля с ошибкой от 5% до 10%: 20.000000%

Доля с ошибкой от 10% до 20%: 30.000000%

Доля с ошибкой от 20% до 30%: 30.000000%

Доля с ошибкой более 30%: 10.000000%
```



2.7. Сформировать набор данных для выполнения прогноза: продлить временную последовательность с заданным шагом на 10 отсчетов. Использовать полученный набор данных для выполнения прогноза: рассчитать выход сети (*sim*) для полученного набора. Сравнить выход сети с соответствующим куском исходной временной последовательности: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью, а также ошибку обучения. Графики занести в отчет.

R^2: 0.432275

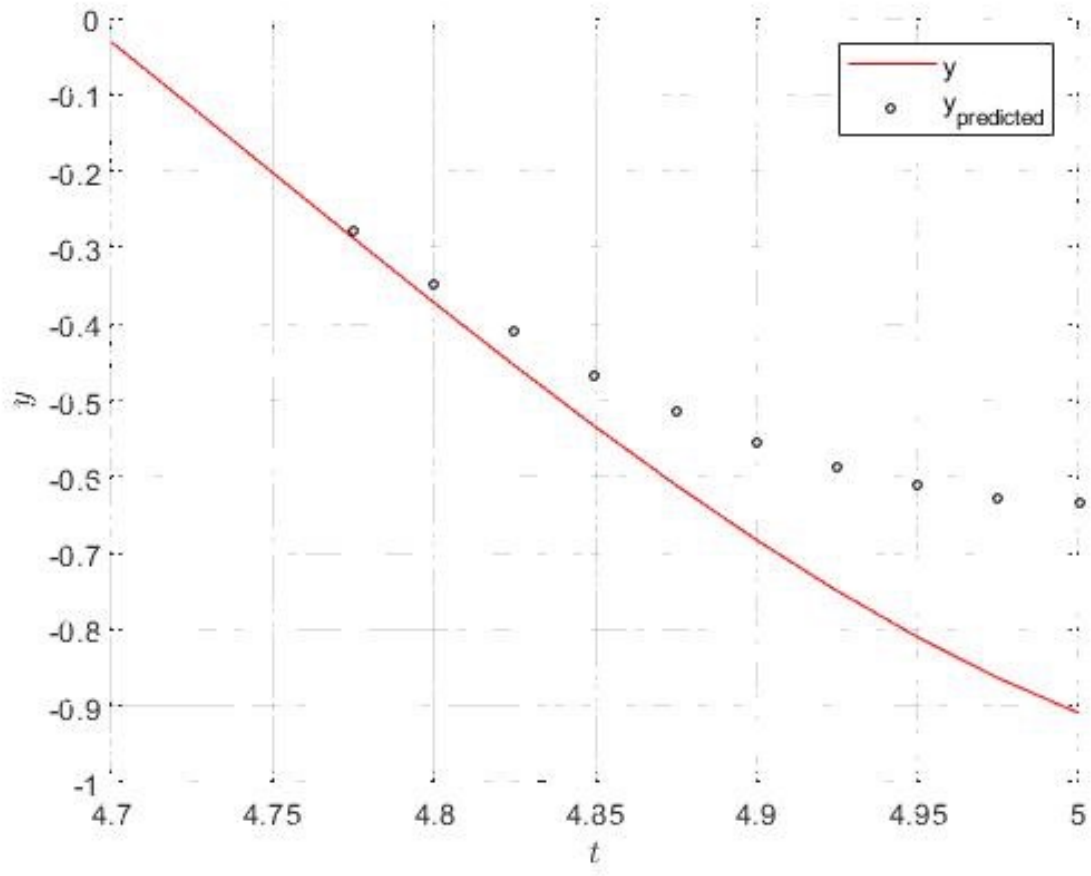
MSE: 0.023011

RMSE: 0.151695

Относительная СКО: 24.418777%

MAE: 0.124227

```
min abs err: 0.010371
max abs err: 0.273873
MAPE: 17.053237
Доля с ошибкой менее 5%: 10.000000%
Доля с ошибкой от 5% до 10%: 20.000000%
Доля с ошибкой от 10% до 20%: 30.000000%
Доля с ошибкой от 20% до 30%: 30.000000%
Доля с ошибкой более 30%: 10.000000%
```



3. Построить и обучить линейную сеть, которая является адаптивным линейным фильтром. Задачей фильтра является моделирование источника шума, чтобы в последующем удалить помехи из полезного сигнала. Фильтр должен аппроксимировать отображение:

$$\hat{y}(n+1) = \sum_{i=1}^{D} w_i x(n-i+1) + b$$

Вместо задержек использовать погружение временного ряда.

3.1. Построить обучающее множество: в качестве входного множества использовать значения второго входного сигнала на заданном интервале; эталонными выходами сети являются значения второй эталонной функции на заданном интервале. Эталонный выходной сигнал соответствует входному сигналу, измененному по амплитуде и смещенному по фазе, поэтому диапазон значений и шаг для сигналов совпадают.

$$x = \sin\left(t^2 - 6t + 3\right), \quad t \in [0,6], h = 0.025$$

$$y = \frac{1}{3}\sin\left(t^2 - 6t - \frac{\pi}{6}\right)$$

3.2. Вместо задержек необходимо расширить входное множество по формуле

$$P = zeros(D, Q)$$

$$P(i, i{:}Q) = x(1{:}Q - i + 1), i = 1,\ldots, D,$$

где Q – количество элементов. Задать глубину погружения ряда D равной 4.

3.3. Создать сеть с помощью функции *newlind*. Занести в отчет весовые коэффициенты и смещение.

```
W = [-1.23349539043816 0.996735826822112
 0.99723889333458 -1.0481119270546]

b = -0.0233789785203087
```

3.4. Рассчитать выход сети (sim) для обучающего множества. Сравнить выход сети с эталонным множеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью, а также ошибку обучения. Графики занести в отчет.

R^2: 0.599160

MSE: 0.020762

RMSE: 0.144090

Относительная СКО: 21.613455%

MAE: 0.129449

min abs err: 0.000368

max abs err: 0.364117

MAPE: 325.783706

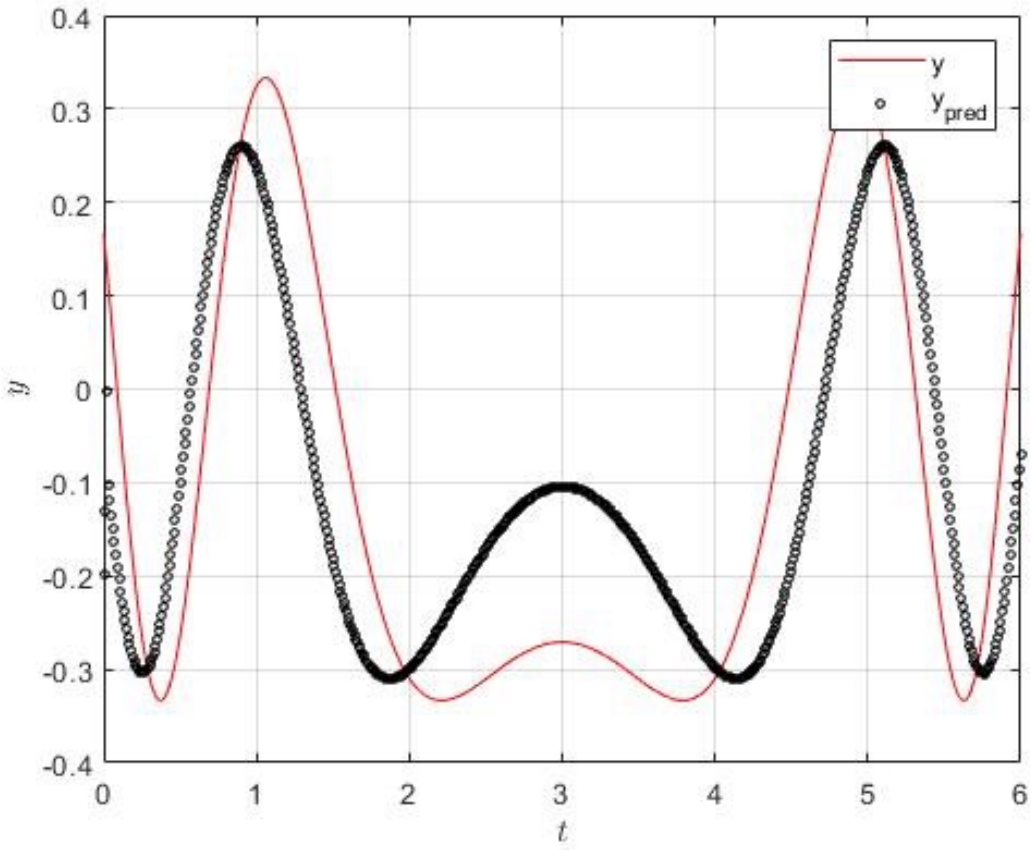
Доля с ошибкой менее 5%: 4.326123%

Доля с ошибкой от 5% до 10%: 4.326123%

Доля с ошибкой от 10% до 20%: 8.652246%

Доля с ошибкой от 20% до 30%: 9.317804%

Доля с ошибкой более 30%: 73.377704%



Код программы

Lab2.m

```
set(0, 'DefaultTextInterpreter', 'latex');
% Задание 1
% Входные данные
signalFunction = @(t) sin(-2 .* t .^ 2 + 7 .* t) - 0.5 .* sin(t);
t = 0:0.025:4.5;
signal = signalFunction(t);
D = 1:5; % задержки
x = con2seq(signal(D(end)+1:end)); % образцы
y = con2seq(signal(D(end)+1:end)); % цели

% Создаем сеть и инициализируем ее случайными значениями
network = newlin([-1 1], 1, D, 0.01);
network.inputweights{1}.initFcn = 'rands';
network.biases{1}.initFcn = 'rands';
network = init(network);

% Обучаем ее с помощью adapt

for i = 1:50
    [network, ~, err, ~] = adapt(network, x, y, con2seq(signal(D)));
    fprintf('|iteration: %d | error: %f|\n', i, sqrt(mse(err)));
end

%% Результаты обучения
predictedSignal = cell2mat(network([con2seq(signal(D)) x]));
p = plot(t, signal, t(D(end)+1:end), predictedSignal(D(end)+1:end),
'o');
```

```
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];

grid on

xlabel('$t$');
ylabel('$y$');
legend('y', 'y_{predicted}');
%% Задание 2
% Входные данные
signalFunction = @(t) sin(t .^ 2 - 6 .* t + 3);
t = 0:0.025:6;
signal = signalFunction(t);
D = 1:3;  % задержки
x = con2seq(signal(D(end)+1:end)); % образцы
y = con2seq(signal(D(end)+1:end)); % цели

% Создаем сеть и инициализируем ее случайными значениями
network = newlin([-1 1], 1, D, maxlinlr(cell2mat(x), 'bias'));
network.inputweights{1}.initFcn = 'rands';
network.biases{1}.initFcn = 'rands';
network = init(network);

% Обучаем ее с помощью train
network.trainParam.epochs = 600;
network.trainParam.goal = 1e-6;
network = train(network, x, y, con2seq(signal(D)));
```

```
%% Результаты обучения
predictedSignal = cell2mat(network([con2seq(signal(D)) x]));
p = plot(t, signal,   t(D(end)+1:end), predictedSignal(D(end)+1:end),
'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
grid on
xlabel('$t$');
ylabel('$y$');
legend('y', 'y_{predicted}');

% Данные для таблицы
display(dataForTable(cell2mat(y), predictedSignal(D(end)+1:end)));

%% Вычисляем прогноз
t = 4.7:0.025:5;
signal = signalFunction(t);  %% len = 13
predictedSignal = [signal(1:3) zeros(1, 10)];
for i = 4:13
    tmp = cell2mat(network(con2seq(predictedSignal(i-3:i))));
    predictedSignal(i) = tmp(end);
end

% Строим график
p = plot(t, signal, t(4:end), predictedSignal(4:end), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
grid on
```

```
xlabel('$t$');
ylabel('$y$');
legend('y', 'y_{predicted}');

% Данные для таблицы
display(dataForTable(signal(4:end), predictedSignal(4:end)));

%% Задание 3
% Входные данные
t = 0:0.01:6;
x = sin(t .^ 2 - 6 * t + 3);
y = 1 / 3 * sin(t .^ 2 - 6 * t + pi / 6);
%x = cos(-5 * t .^ 2 + 10 * t - 5);            % входной сигнал
%y = 1 / 8 * cos(-5 * t .^ 2 + 10 * t);          % выходной сигнал

D = 4;          % глубина
Q = length(x); % число образцов

P = zeros(D, length(x));
for i = 1:D
    P(i,i:Q) = x(1:Q-i+1);
end

%% Создание сети + график + характеристики

network = newlind(P, y);

fprintf('W = %s\nb = %s\n\n', mat2str(network.IW{1}),
mat2str(network.b{1}));
```

```
display(dataForTable(y, network(P)));


p = plot(t, y, t, network(P), 'o');


p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];


grid on
xlabel('$t$');
ylabel('$y$');
legend('y', 'y_{pred}');
```

dataForTable.m

```
function res =  dataForTable(y, yp)
    R2 = 1 - sum((y - yp) .^ 2)/sum((y - mean(y)) .^ 2);
    MSE = mse(y - yp);
    RMSE = sqrt(MSE);
    CKO = RMSE / (max(y) - min(y)) * 100;
    MAE = mae(y - yp);
    MinAbsErr = min(abs(y - yp));
    MaxAbsErr =  max(abs(y - yp));
    MAPE = mean(abs((y - yp) ./ y)) * 100;
    errors = abs((y - yp) ./ y) * 100;
    Under5PersentPortion = sum(errors < 5) / length(y) * 100;
    Under10PersentPortion = sum(5 <= errors & errors < 10) / length(y)
* 100;
        Under20PersentPortion = sum(10 <= errors & errors < 20) /
length(y) * 100;
        Under30PersentPortion = sum(20 <= errors & errors < 30) /
length(y) * 100;
```

```
        Over30PersentPortion = sum(errors >= 30) / length(y) * 100;


        res = sprintf(['R^2: %f\n' ...
                       'MSE: %f\n' ...
                       'RMSE: %f\n' ...
                       'Относительная СКО: %f%%\n' ...
                       'MAE: %f\n'...
                       'min abs err: %f\n' ...
                       'max abs err: %f\n' ...
                       'MAPE: %f\n' ...
                       'Доля с ошибкой менее 5%%: %f%%\n' ...
                       'Доля с ошибкой от 5%% до 10%%: %f%%\n'...
                       'Доля с ошибкой от 10%% до 20%%: %f%%\n' ...
                       'Доля с ошибкой от 20%% до 30%%: %f%%\n' ...
                       'Доля с ошибкой более 30%%: %f%%\n'], ...
                           R2, MSE, RMSE, CKO, MAE, MinAbsErr, MaxAbsErr,
MAPE,   Under5PersentPortion, Under10PersentPortion, Under20PersentPortion,
Under30PersentPortion, Over30PersentPortion);


    end
```

Выводы

В данной лабораторной работе использована линейные нейронные сети с задержками для аппроксимации функции, многошаговой и адаптивной фильтрции. Я имел возможность сравнить применить различные методы обучения линейных нейросетей и сравнить результаты. Лабораторная работа оказалась интересна тем, что в ней решалось сразу несколько задач нейроинформатики при помощи линейных нейросетей.