**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 2**

по курсу «Нейроинформатика»

Тема: Линейная нейронная сеть. Правило обучения Уидроу-Хоффа.

Студент: Дубинин А. О.

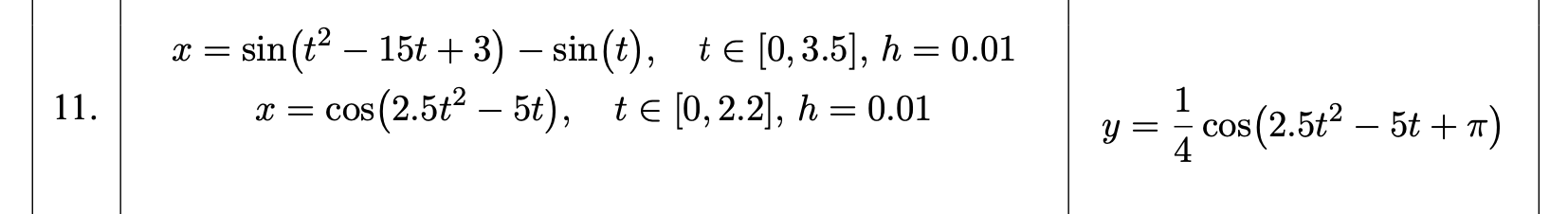
Группа: 8О-407Б-17

Преподаватель: Аносова Н.П.

Оценка:

Москва, 2021

Вариант 11.



**Выполнение работы:**

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.

*%1.1 Построение обучающего множества  
%Шаг*h = 0.01;  
  
*%Задаем координаты*X = 0:h:3.5;  
Y = sin((X.\*X) - 15\*X + 3) - sin(X);  
  
*%Последовательность входных образцов*Pn = con2seq(Y);

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

*%1.2  
%Построение сети*delays = [1 2 3 4 5];  
net = newlin([-1 1], 1, delays, 0.01);

1.3 Инициализировать сеть случайными значениями.

net.inputweights{1,1}.initFcn='rands'; net.biases{1}.initFcn='rands'; net=init(net);

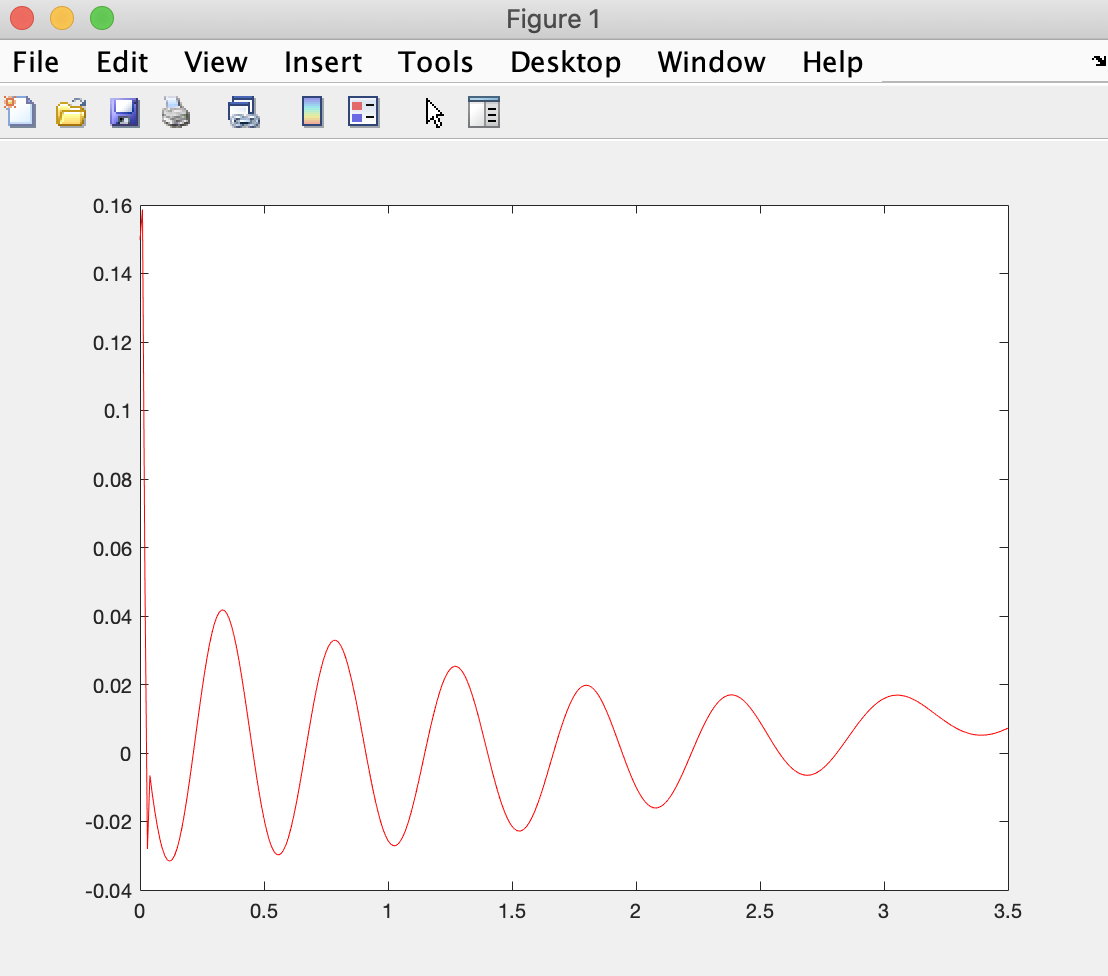
1.4 Выполнить адаптацию с числом циклов равным 50. Занести в отчет величину ошибки обучения с помощью sqrt(mse). Поскольку сеть имеет задержки, то в функцию адаптации необходимо отдельно передать первые 5 элементов входной последовательности для инициализации задержек (входной параметр Pi). В противном 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.

*%1.4 Адаптация сети*Pi = con2seq(Y(1:5));  
P = Pn(6:**end**);  
T = Pn(6:**end**);  
  
**for** i = 1:50  
 [net, Y, E] = adapt(net, P, T, Pi);  
 *%display(net.IW{1,1});***end**;  
  
*%Ошибка обучения*err = sqrt(mse(E));  
display(err);  
  
*%График ошибки*figure  
  
E = cell2mat(Pn) - cell2mat(sim(net, Pn));  
plot(X, E, **'r'**);

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

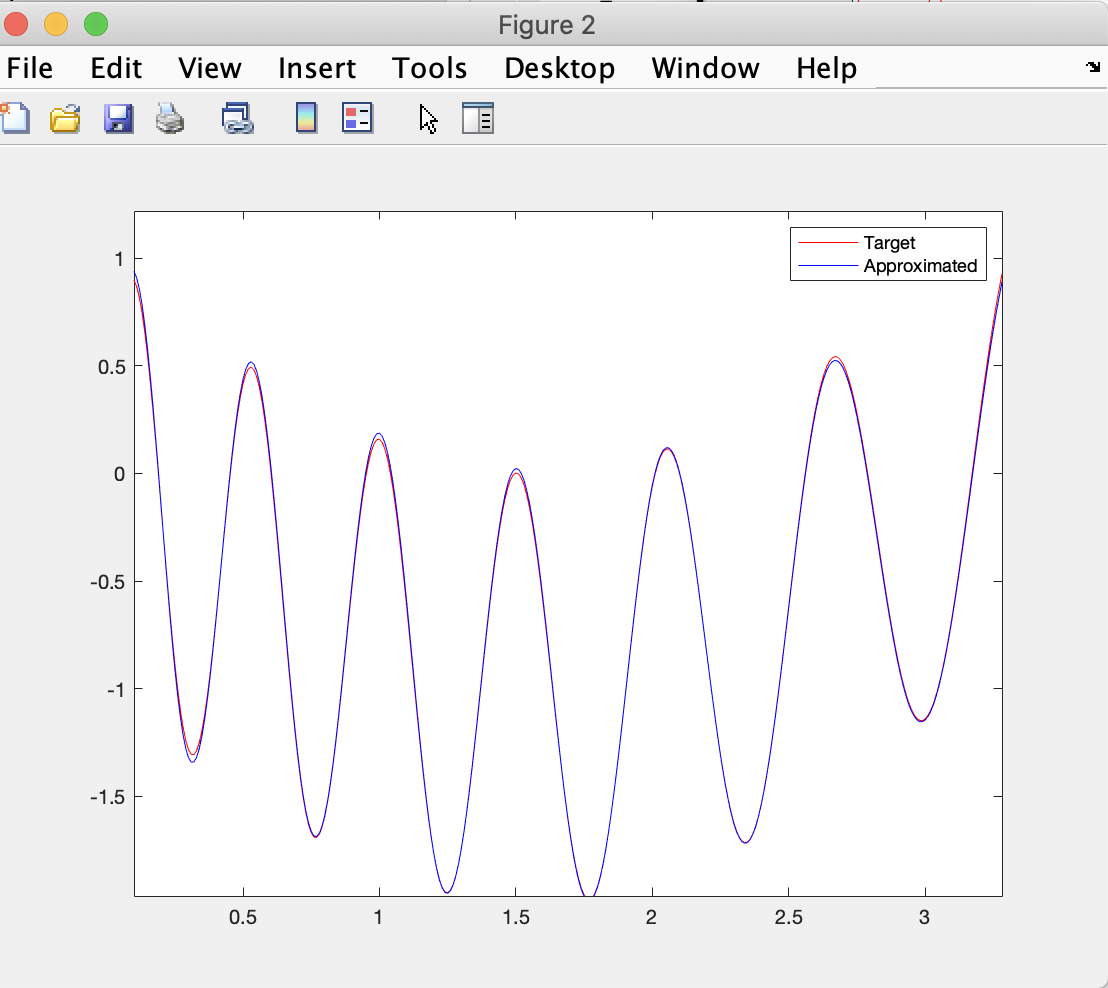
err =

0.0107

****

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

*%1.5 Графики  
%Эталон*figure  
referenceLine = plot(X(6:**end**), cell2mat(T), **'r'**);  
hold on;  
  
*%Предсказанное*approximationLine = plot(X(6:**end**), cell2mat(Y), **'b'**);  
legend([referenceLine,approximationLine],**'Target'**, **'Approximated'**);  
hold off;

****

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.

*%1.1 Построение обучающего множества  
%Шаг*h = 0.01;  
  
*%Задаем координаты*X = 0:h:3.5;  
Y = sin((X.\*X) - 15\*X + 3) - sin(X);  
  
*%Последовательность входных образцов*Pn = con2seq(Y);

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’).

*%2.2 Построение сети*delays = [1 2 3];  
net = newlin([-1 1], 1, delays, maxlinlr(cell2mat(Pn),**'bias'**));

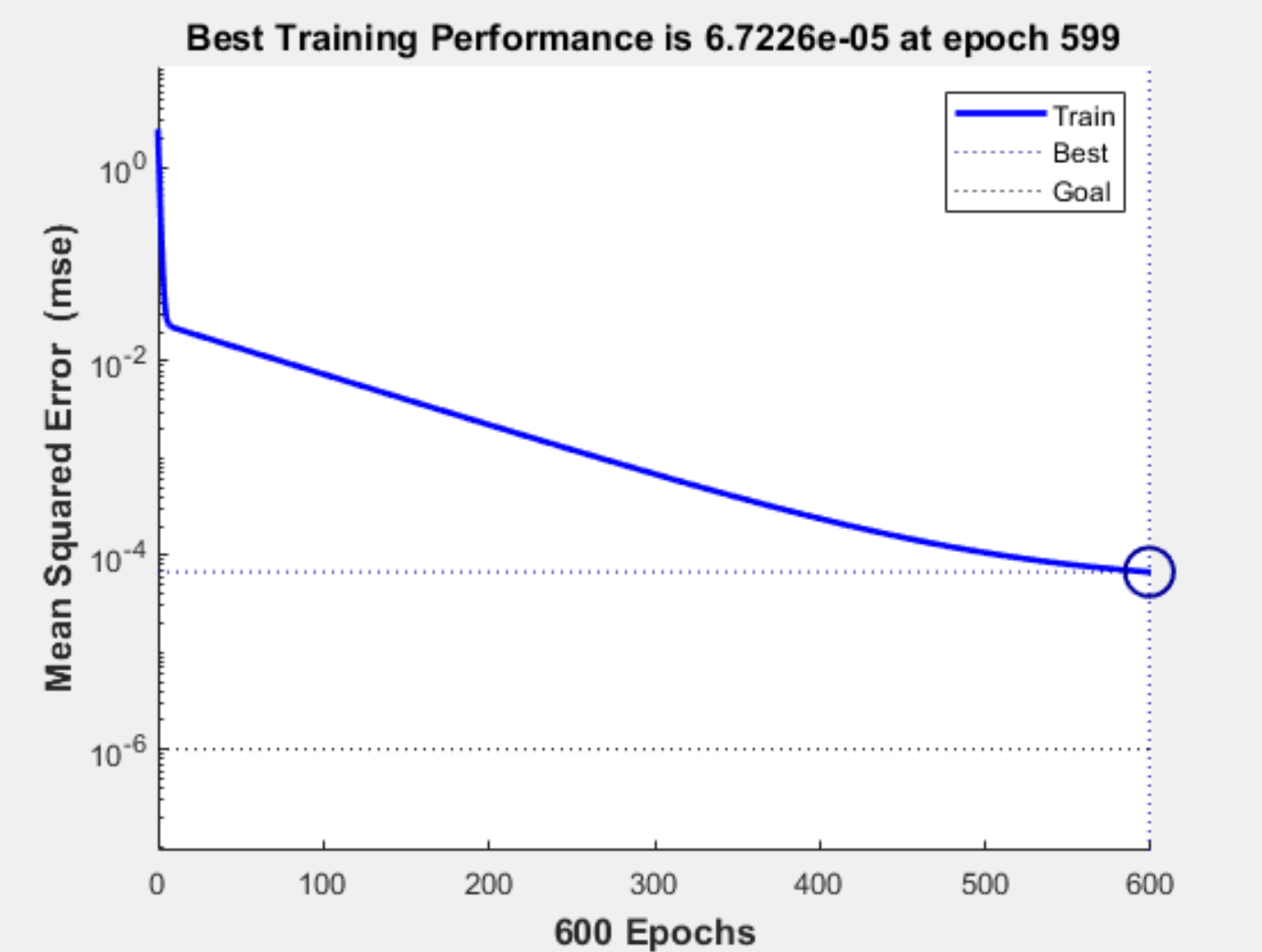
2.3 Инициализировать сеть случайными значениями.

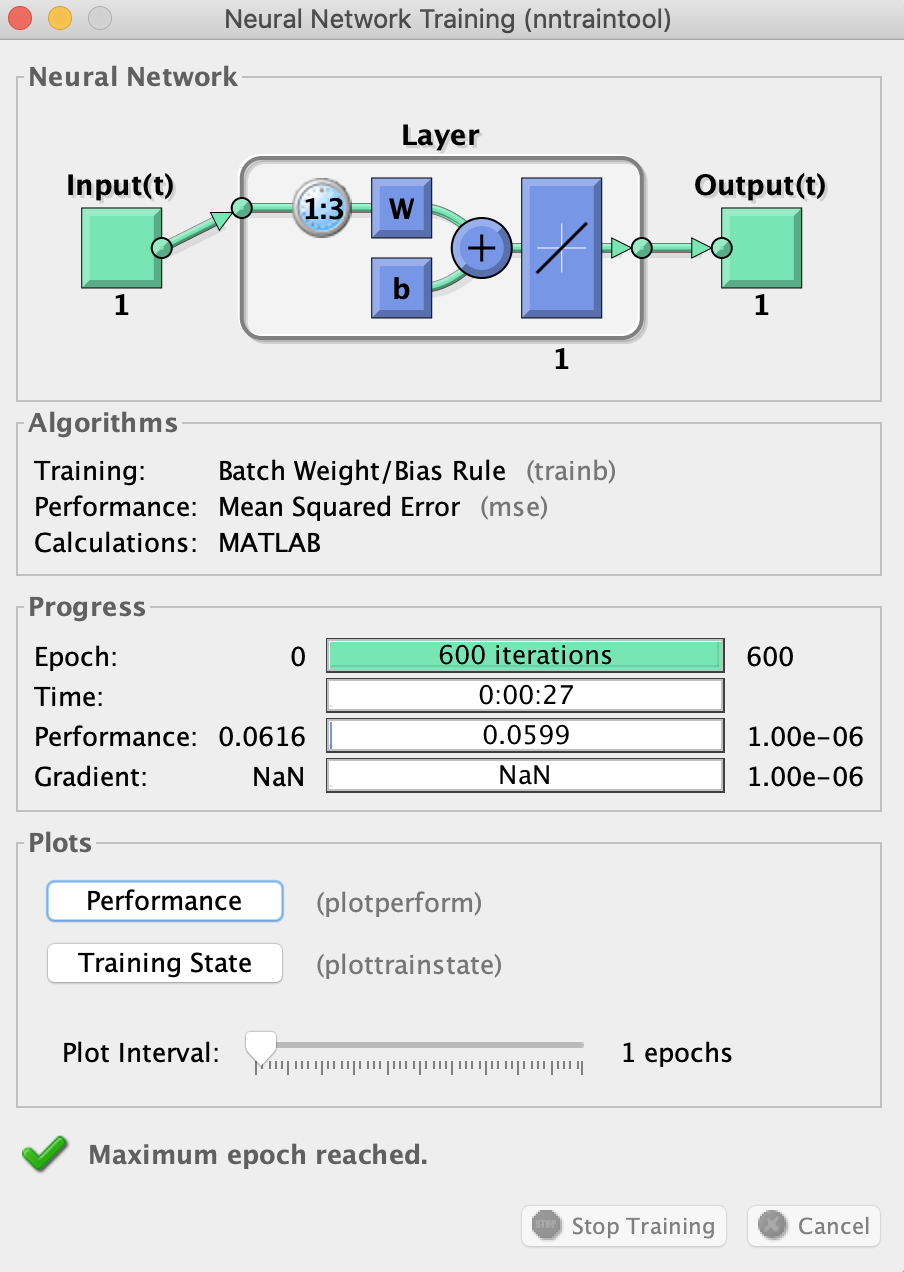
*%2.3  
%Инициализация*net.inputweights{1,1}.initFcn = **'rands'**;  
net.biases{1}.initFcn = **'rands'**;  
net = init(net);

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.

*%2.4 Задание параметров обучения*Pi = con2seq(Y(1:3));  
P = Pn(4:**end**);  
T = Pn(4:**end**);  
  
net.trainParam.epochs = 600;  
net.trainParam.goal = 0.000001;  
*%Обучение*net = adapt(net, P, T, Pi);  
net = train(net, P, T);

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.

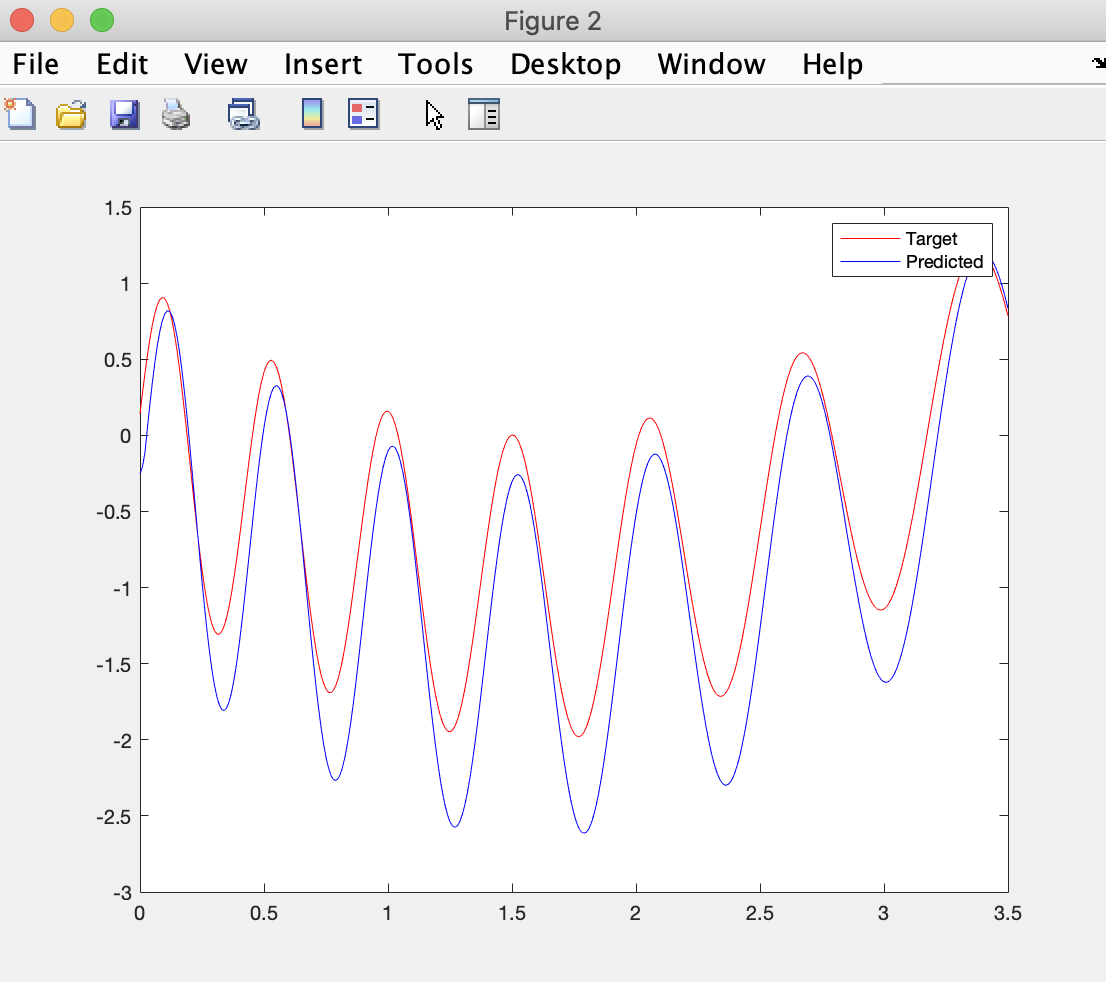
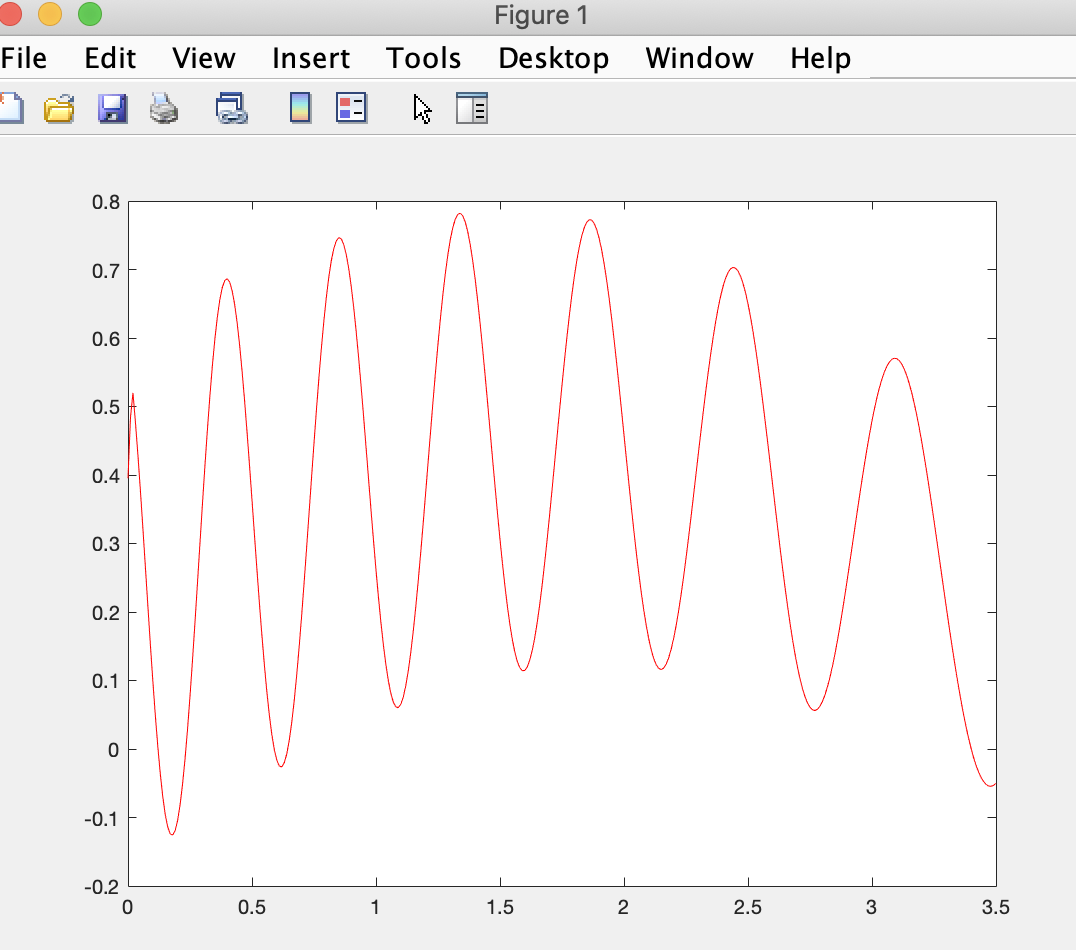




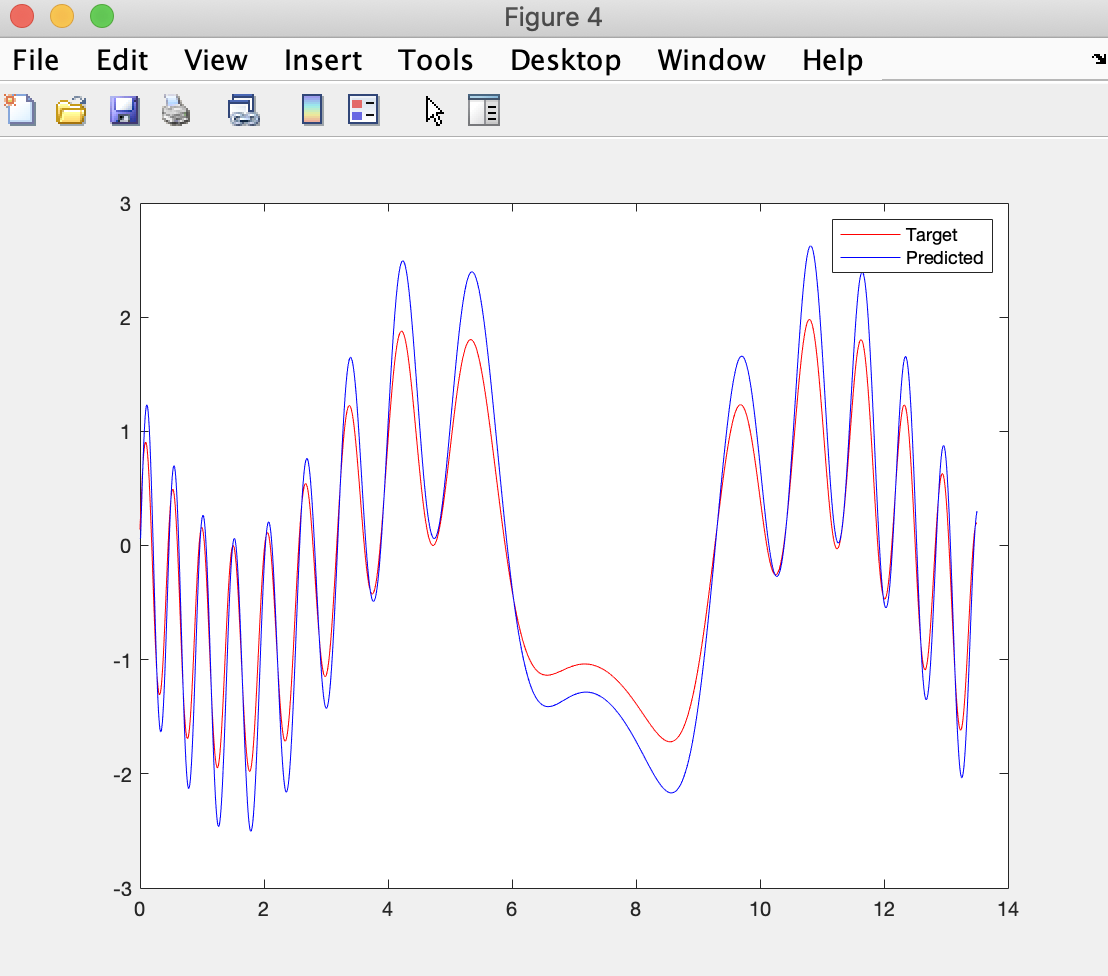
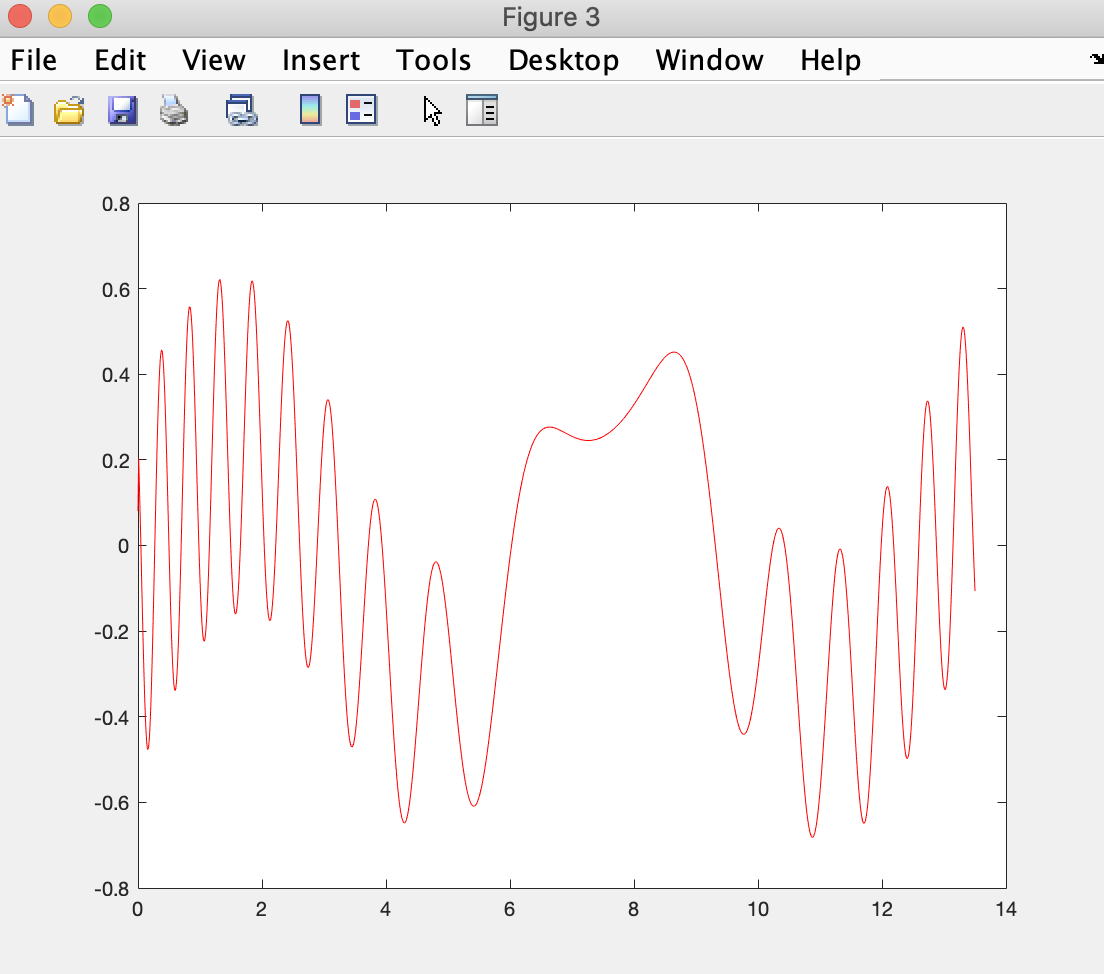
|  |  |
| --- | --- |
| Функция создания сети | newlin |
| Входной слой | 1 |
| Скрытый слой | 0 |
| Выходной слой | 1 |
| Активационные функции | purelin |
| Динамика | Задержки 1,2,3 |
| Число примеров в подмножествах | 238 |
| Метод обучения | trainb |
| Параметры обучения | lr = 0.0036 |
| Метод инициализации сети | rands |
| Критерий окончания обучения | epochs = 600, goal = 1E-6 |
| Причина окончания обучения | Достигнуто максимальное значение количества эпох |
| Число эпох обучения | 600 |

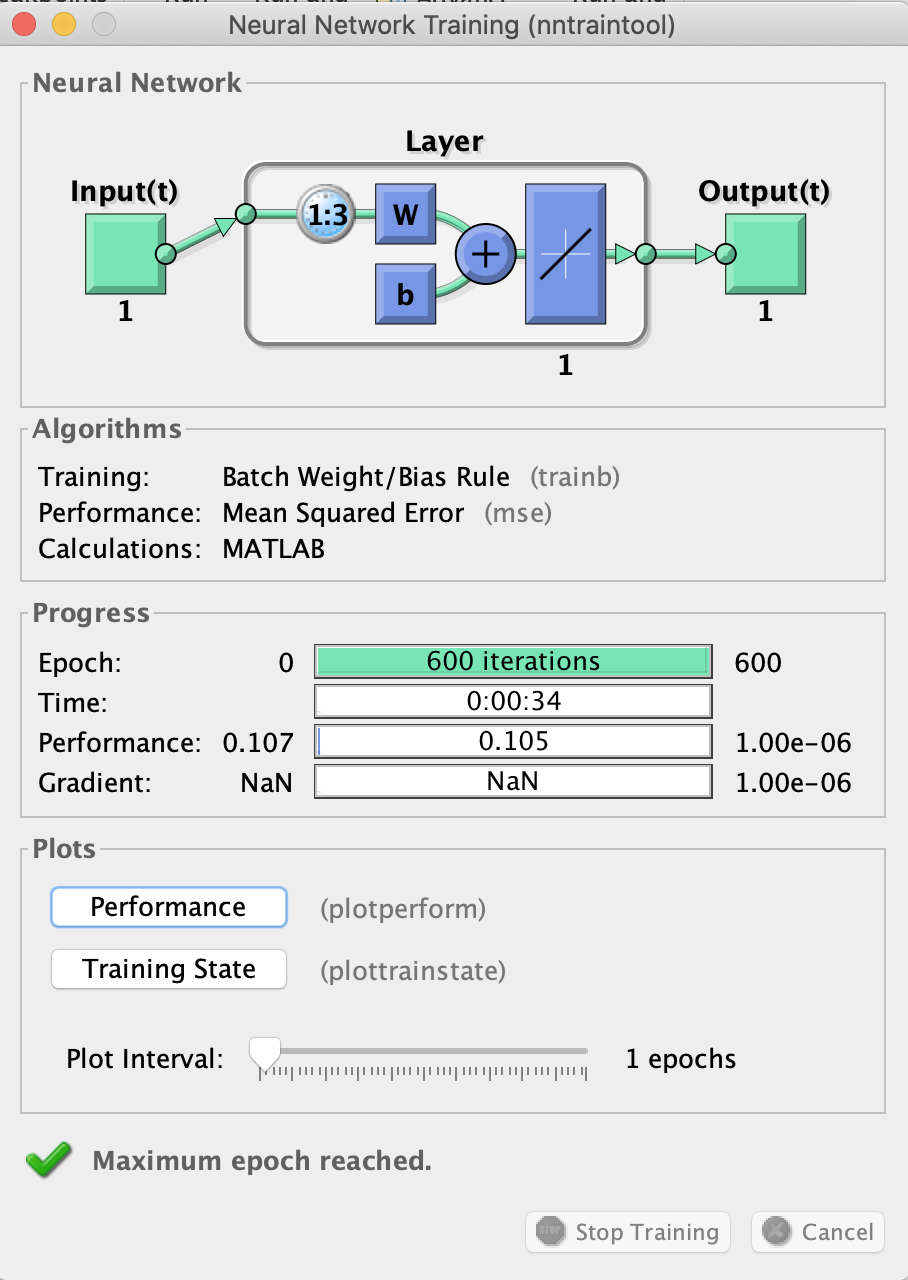
2.6 Рассчитать выход сети (sim) для обучающего множества. Сравнить выход сети с соответствующим эталонным множеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью, а также ошибку обучения. Графики занести в отчет.

E = cell2mat(Pn) - cell2mat(sim(net, Pn));  
  
figure  
plot(X, E, **'r'**);  
  
figure  
referenceLine = plot(X, Y, **'r'**);  
hold on;  
  
predictionLine = plot(X, cell2mat(sim(net, Pn)), **'b'**);  
  
legend([referenceLine,predictionLine],**'Target'**, **'Predicted'**);  
hold off;



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.

****

****

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.

*%3.1 Построение обучающего множества  
%Шаг*h = 0.01;  
X = 0:h:2.2;  
  
*%Входное множество*x = cos(2.5\*X.\*X - 5\*X);  
  
*%Эталонный выход*Y = 0.25\*cos(2.5\*X.\*X - 5\*X + pi);

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,...,D

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

*%3.2 Расширение входного множества  
  
%Задаем глубину погружения временного ряда*D = 4;  
Q = length(x);  
P = zeros(D, Q);  
  
**for** i=1:D  
 P(i, i:Q) = x(1:Q - i + 1);  
**end**;

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

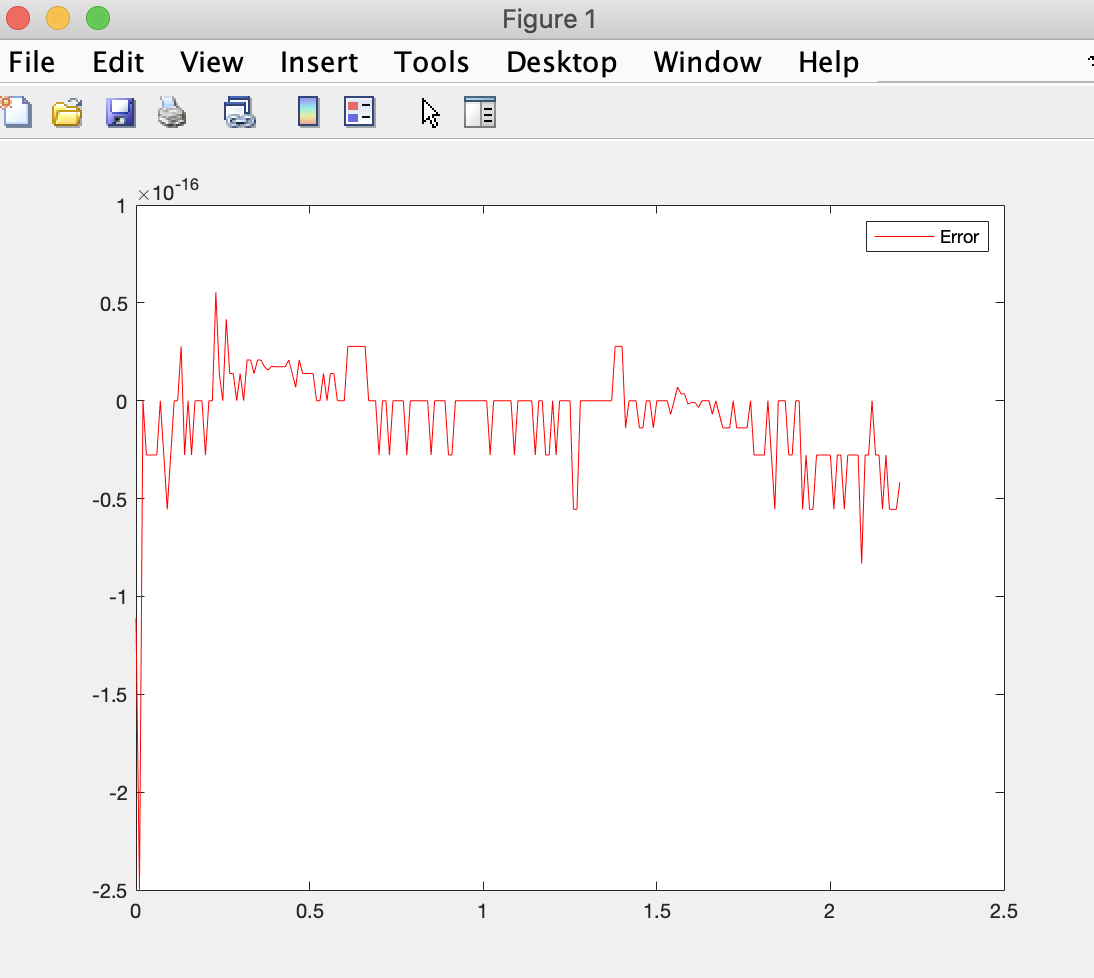
*%3.3 Создаем сеть*net = newlind(P, Y);  
  
*%Отображаем полученные веса и смещение*display(net.IW{1,1});  
display(net.b{1});

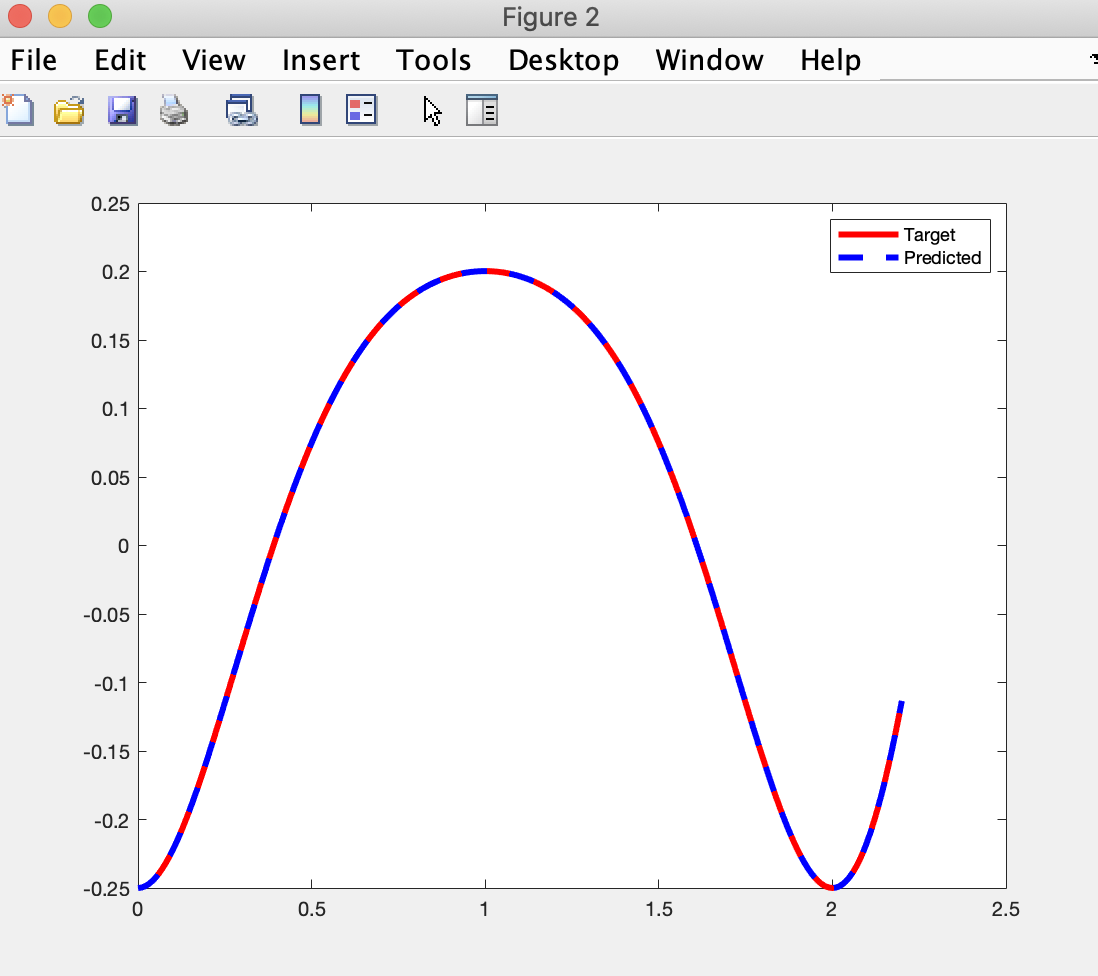
**-0.2500 0.0000 -0.0000 -0.0000**

**2.2406e-17**

3.4 Рассчитать выход сети (*sim*) для обучающего множества. Сравнить выход сети с эталон- ным 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.

*%3.4*T = sim(net, P);  
*%Погрешность*E = Y - T;  
  
figure  
err = plot(X, E, **'r'**);  
hold on;  
*%set(err, 'linewidth', 1);*legend(err,**'Error'**);  
hold off;  
  
*%Эталон*figure  
referenceLine = plot(X, Y, **'r'**);  
set(referenceLine, **'linewidth'**, 3);  
hold on;  
  
*%*approximationLine = plot(X, T, **'--b'**);  
set(approximationLine, **'linewidth'**, 3);  
  
legend([referenceLine,approximationLine],**'Target'**, **'Predicted'**);  
hold off;





Вывод:

Алгоритм LMS можно использовать для аппроксимации нелинейной функции, а также для прогнозирования значений функции, при нелинейной зависимости.