|  |  |
| --- | --- |
| mai |  |

**Институт** №8 «Информационные технологии и прикладная математика»

**Кафедра**  811

Лабораторная Работа №2

по курсу “Нейросетевые технологии искусственного интеллекта”

Тема: Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Москва, 2021

# Цель работы:

Исследование свойств линейной нейронной сети и алгоритмов ее 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.

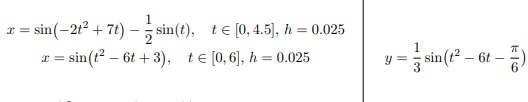
**Оборудование:** Intel Core i5-6200U

**Программное обеспечение:** MATLAB 9.2 R2017a

# Сценарий выполнения работы:

**Номер варианта: 2N**, где N - номер студента по действующему списку. N = 5

Вариант №10:



* + *Часть 1. Задана временная последовательность*

*x*(*n*) *. Построить и*

*обучить линейную сеть с задержками, которая будет выполнять одношаговый прогноз для первой функции из варианта задания:*

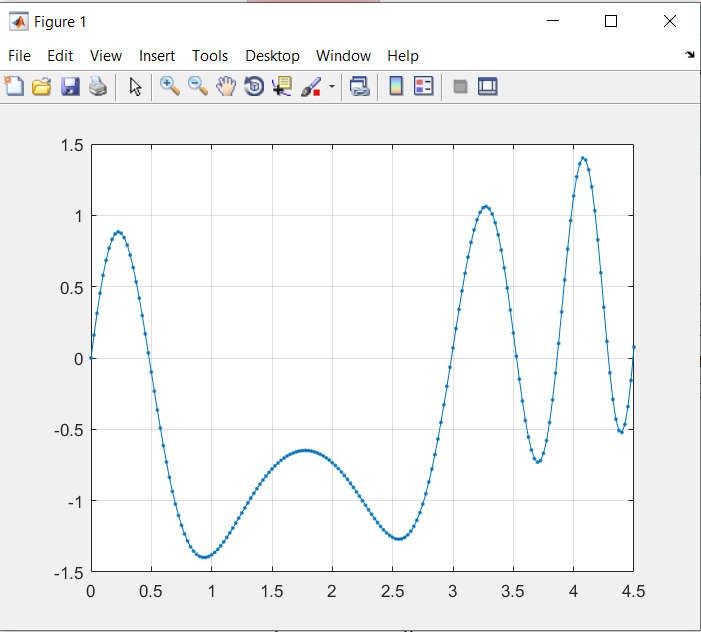
*D*

*x*ˆ(*n*  1)  *i x*(*n*  *i*  1)  *b , где D задает глубину погружения*

*i*1

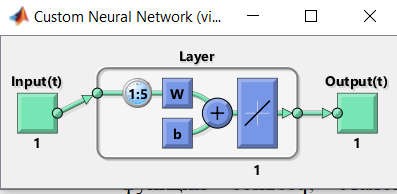
*временного ряда (delays),* *ш* ,*b* *— весовые коэффициенты.*

* 1. Строим обучающее множество: в качестве входного множества используем значения первого входного сигнала на заданном интервале; преобразовываем входное множество к последовательности входных образцов с помощью функции con2seq; эталонные выходы сети формируем из входной последовательности, чтобы сеть выполняла одношаговый прогноз. На рис. 1 представлен график входного сигнала.



**Рисунок 1.** График функции

* 1. Создаем сеть с помощью функции newlin. Задаем задержки от 1 до D = 5 . Задаем скорость обучения равной 0.01. На рис. 2 отображена структура сети с помощью функции (display).



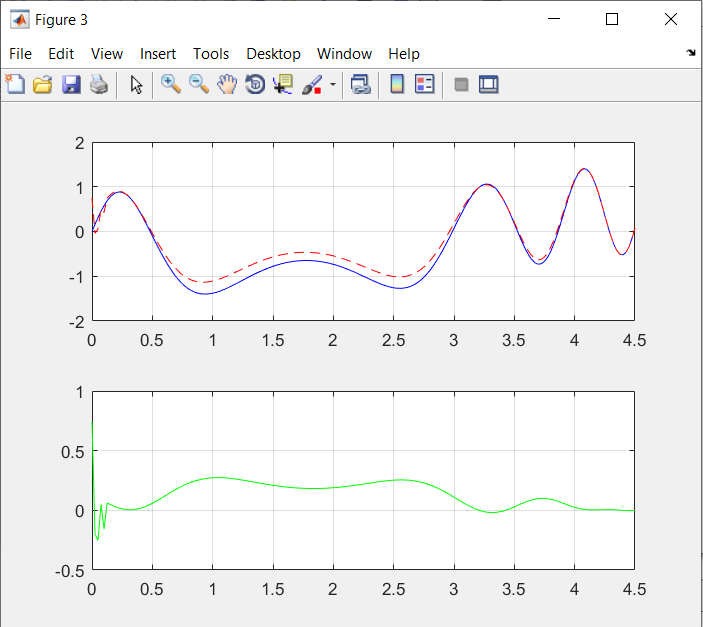
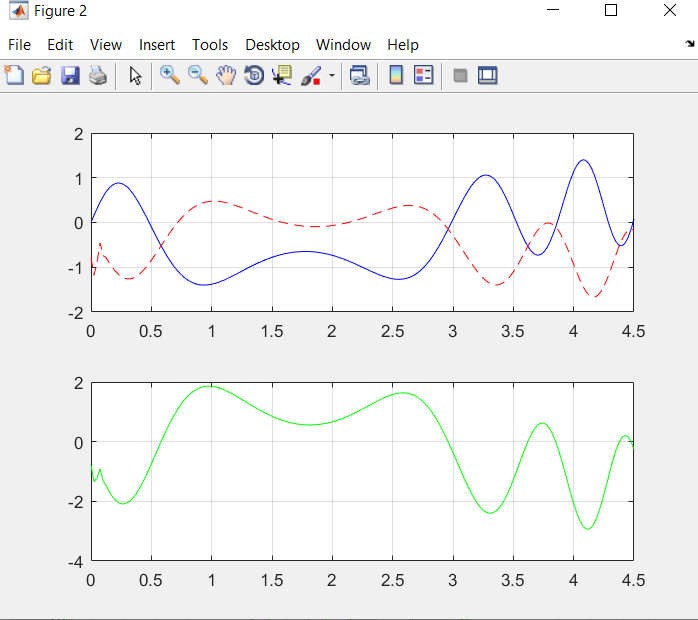
**Рисунок 2.** Структура сети

* 1. Инициализируем сеть случайными значениями.
  2. Выполняем адаптацию с числом циклов равным 50. Величину ошибки обучения с помощью sqrt(mse) можно увидеть ниже. В функцию адаптации необходимо отдельно передаем первые 5 элементов входной последовательности для инициализации задержек (входной параметр Pi). В дальнейшем использовать входную и выходную последовательности, начиная с 6 элемента.

M1 = 0.8515

* 1. На рис. 3,а представлен график эталонного значения, а на рис. 3,б представлен график предсказанный сетью. Величина ошибки обучения:

M2 = 0.2679



**Рис 3,а Рис 3,б**

**Рисунок 3.** Результат работы

Листинг программы:

clear all;

x = @ (t) sin(-2\*t.^2 + 7\*t)-0.5\*sin(t); t1 = 0:0.025:4.5;

P=x(t1);

plot(t1,P,'.-'); grid; for i=1:5

Pi(i)=P(i);

end;

for i=1:size(P,2)-5 PM(i)=P(i+5);

end;

net=newlin([0,5],1,[1 2 3 4 5],0.2);

display(net); view(net);

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

IW=net.IW{1,1}

b=net.b{1} M1=sqrt(mse(PM-net(PM))) Pi=con2seq(Pi); PM=con2seq(PM); P=con2seq(P);

Y=sim(net,P,Pi);

Y = seq2con(Y); Y=Y{1}; P = seq2con(P); P=P{1}; E = Y - P;

figure; subplot(211)

plot(t1,P,'b',t1,Y,'r--'); grid; subplot(212)

plot(t1,E,'g'); grid; net.adaptParam.passes = 500; [net]=adapt(net,PM,PM,Pi); t2=0:0.025:4.5;

P=x(t2);

P=con2seq(P); Y=sim(net,P,Pi);

Y = seq2con(Y); Y=Y{1}; P = seq2con(P); P=P{1}; E = Y - P;

M2=sqrt(mse(Y-P)) figure; subplot(211)

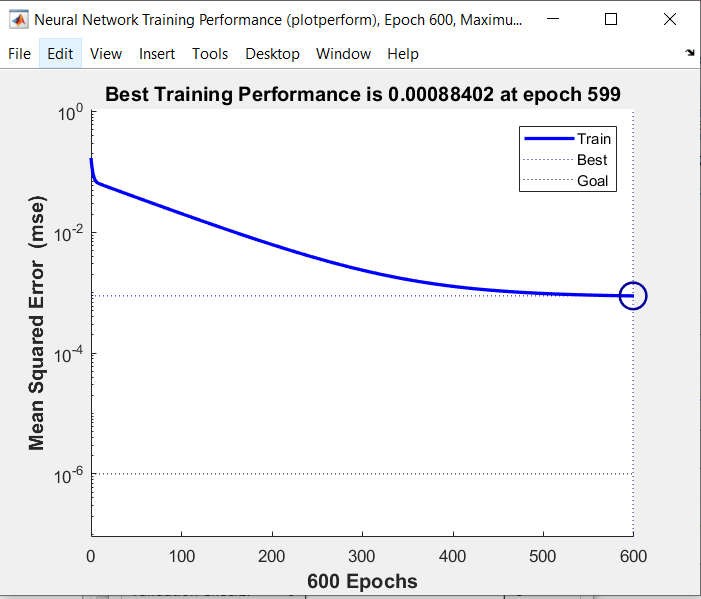
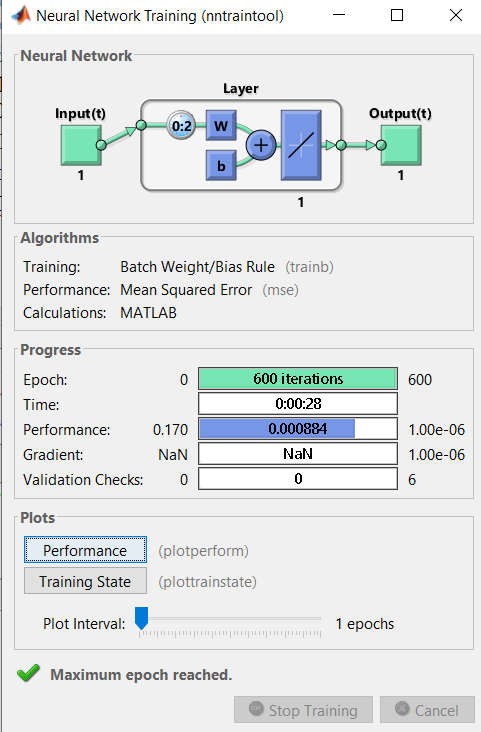
plot(t2,P,'b',t2,Y,'r--'); grid; subplot(212)

plot(t2,E,'g'); grid;

* *Часть 2. Для временной последовательности из задания 1 обучить линейную сеть с задержками (линейный адаптивный фильтр) и выполнить многошаговый прогноз.*
  1. Строим обучающее множество: в качестве входного множества используем значения первого входного сигнала на заданном интервале; преобразовываем входное множество к последовательности входных образцов с помощью функции con2seq; эталонные выходы сети формируем из входной последовательности, чтобы сеть выполняла одношаговый прогноз.
  2. Создаем сеть с помощью функции newlin. Задаем задержки от 1 до D = 3. Задаем скорость обучения с помощью функции maxlinlr(cell2mat(P),’bias’). Значение maxlinlr можно увидеть ниже.

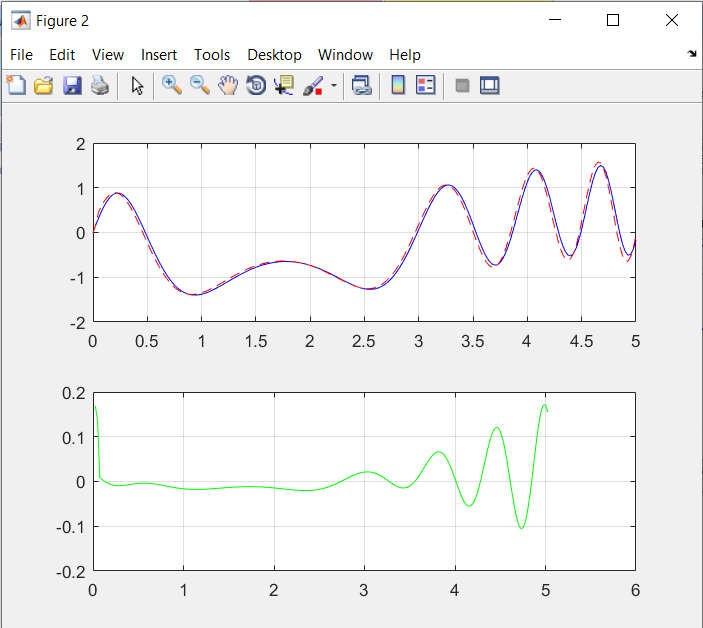
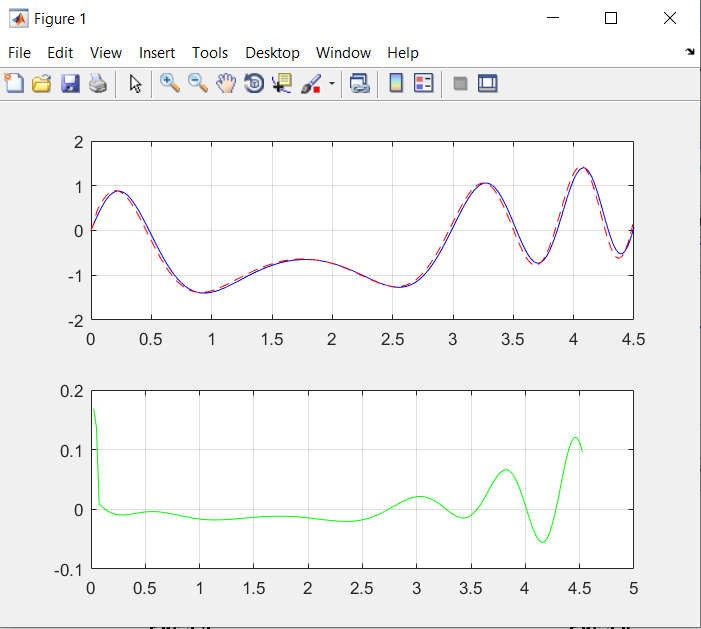
lr=maxlinlr(P,'bias')=0.0044

* 1. Инициализируем сеть случайными значениями.
  2. Задаем параметры обучения: число эпох обучения (net.trainParam.epochs) равным 600, предельное значение критерия обучения (net.trainParam.goal) равным 10Е−6. Также проинициализируем задержки Pi. Выполняем обучение сети с помощью функции train. На рис. 4,а продемонстрировано окно Neural Network Training, на рис. 4.б представлен график сходимости значения ошибки.



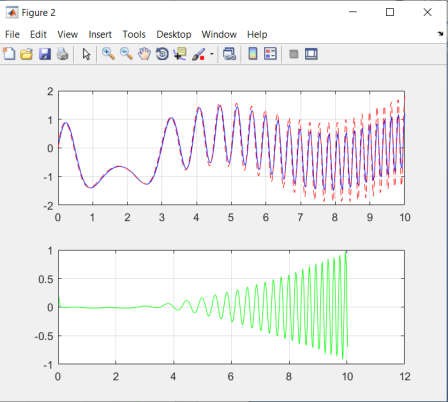
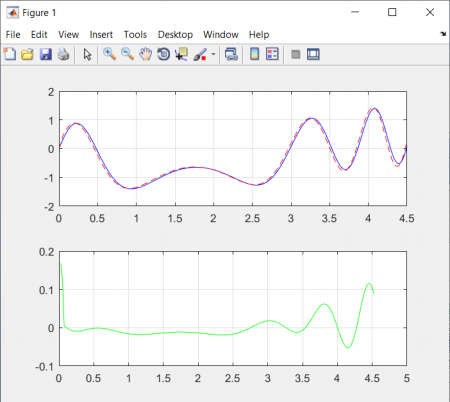
**Рис 4,а Рис 4,б**

**Рисунок 4.** Результат работы функции train

* 1. Отобразим на графике эталонные значения и предсказанные сетью, а также ошибку обучения. На рис. 5,а можно увидеть график эталонного значения, а на рис. 5,б можно увидеть график, предсказанный сетью.

**Рис 5,а Рис 5,б**

**Рисунок 5.** Результат работы предсказания функции train

* 1. Сформируем набор данных для выполнения прогноза: продлим временную последовательность с заданным шагом на 10 отсчетов. Используем полученный набор данных для выполнения прогноза: рассчитаем выход сети (sim) для полученного набора. Сравним выход сети с соответствующим куском исходной временной последовательности. Отобразим на графике эталонные значения и предсказанные сетью, а также ошибку обучения. На рис. 6,а можно увидеть график эталонного значения, а на рис. 6,б можно увидеть график, предсказанный сетью. Величина ошибки указана ниже:

M3 = 0.2962

**Рис 6,а Рис 6,б**

**Рисунок 6.** Результат работы предсказания функции train

Листинг программы:

clear all;

x = @ (t) sin(-2\*t.^2 + 7\*t)-0.5\*sin(t); t1 = 0:0.025:4.5;

P=x(t1);

% plot(t1,P,'.-'); grid; for i=1:2

Pi(i)=P(i);

end;

for i=1:size(P,2)-3 PM(i)=P(i+2);

end;

for i=1:size(P,2)-3 PM1(i)=P(i+3);

end; lr=maxlinlr(P,'bias');

net=newlin([-1,1],[-1,1],[0 1 2],0.002);

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

IW=net.IW{1,1}

b=net.b{1} M1=sqrt(mse(PM-net(PM))) Pi=con2seq(Pi); PM=con2seq(PM); PM1=con2seq(PM1); P=con2seq(P);

net.trainParam.goal=1E-6; net.trainParam.epochs=600; net=train(net,PM,PM1,Pi); Y=net(P);

Y = seq2con(Y); Y=Y{1};

PM = seq2con(PM); PM=PM{1}; P = seq2con(P); P=P{1}; t3=0.025:0.025:4.525; X=x(t3);

E = X - Y;

M2=sqrt(mse(Y-X)); figure; subplot(211)

plot(t1,P,'b',t1,Y,'r--'); grid; subplot(212)

plot(t3,E,'g'); grid; t4=0:0.025:10; P1=x(t4);

P2=con2seq(P1); Y1=sim(net,P2);

Y1 = seq2con(Y1); Y1 = Y1{1}; t5=0.025:0.025:10.025; X1=x(t5);

E1=X1-Y1;

M3=sqrt(mse(Y1-X1)) figure; subplot(211)

plot(t4,P1,'b',t4,Y1,'r--'); grid; subplot(212)

plot(t5,E1,'g'); grid;

* *Часть 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

*D*

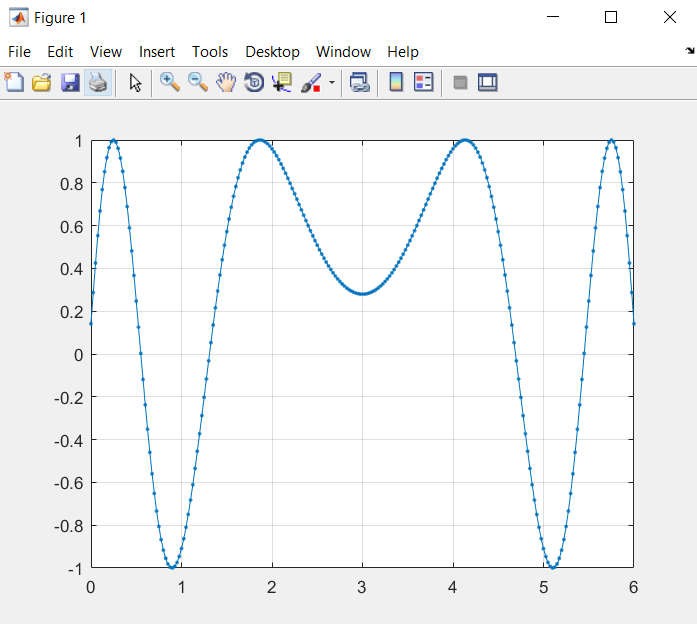
отображение:

*x*ˆ(*n*  1)  *i x*(*n*  *i*  1)  *b*

*i*1

*Вместо задержек использовать погружение временного ряда.*

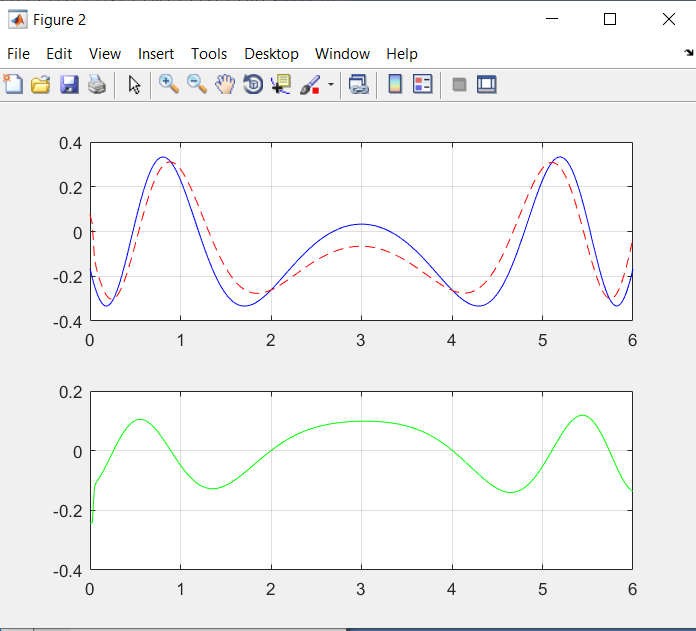
* 1. Построим обучающее множество: в качестве входного множества используем значения второго входного сигнала на заданном интервале; эталонными выходами сети являются значения второй эталонной функции на заданном интервале. Эталонный выходной сигнал соответствует входному сигналу, измененному по амплитуде и смещенному по фазе, поэтому диапазон значений и шаг для сигналов совпадают. На рис. 7 представлен график входного сигнала.



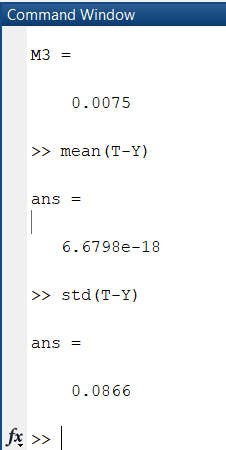
**Рисунок 7.** График функции

* 1. Создаем сеть с помощью функции newlind.
  2. Рассчитаем выход сети (sim) для обучающего множества. Сравним выход сети с эталонным множеством. Отобразим на графике эталонные значения и предсказанные сетью, а также ошибку обучения. На рис. 8 можно увидеть результат работы фильтра. Величину ошибки обучения можно увидеть ниже:

M3 = 0.0075



**Рисунок 8.** Результат работы фильтра

Среднее квадратическая ошибка и средняя ошибка представлена ниже:

Листинг программы:

clear all;

%3.1 Построение обучающего множества

%Входное множество

x = @ (t) sin(t.^2 - 6\*t + 3);

%Эталонный выход

y = @ (t) (1/3)\*sin(t.^2 - 6\*t - pi/6); t1=0:0.025:6;

P=x(t1);

plot(t1,P,'.-'); grid;

%3.2 Расширение входного множества P1=zeros(4,size(P,1));

for i=1:3

P2(i)=0;

end;

for i=1:size(P,2) P2(i+3)=P(i);

end;

for i=1:4

P3(i,1:size(P,2))=P2(i:size(P,2)+i-1);

end; T=y(t1);

T=con2seq(T); P=con2seq(P3);

%3.3 Создаем сеть net=newlind(P,T); Y=net(P);

Y = seq2con(Y);

Y = Y{1};

T = seq2con(T);

T = T{1};

M3=mse(T-Y)

figure; subplot(211)

plot(t1,T,'b',t1,Y,'r--'); grid; subplot(212)

plot(t1,T-Y,'g'); grid;

**Вывод:** В данной лабораторной работе исследованы свойства линейной нейронной сети и алгоритмов ее обучения, применение сети в задачах аппроксимации и фильтрации. В ходе лабораторной работы мне удалось выделить некоторые наблюдения в работе сети, а именно недостатки: данный способ решения позволяет предсказывать недалекое будущее. Во 2-ом этапе лабораторной работы можно увидеть, что при увеличении времени (например, до 10) ошибка точности увеличивается; достоинства: линейная нейроная сеть с правилом Уидроу-Хоффа позволяет использовать сеть для аппроксимации функции, выполнения прогноза и в качестве адаптивного фильтра для фильтрации помех. Стоит отметить метод *newlind*. Данный метод точнее решает задачу, оценка точности схожа с аналитическим решением.