САМОСТІЙНА РОБОТА

з дисципліни

“Нейроінформаційні мережі керування біотехнічними обєктами”

(Модуль 1)

Тривалість роботи – 40 годин.

1. Створення і використання нейронних мереж за допомогою командного рядка

1.1. Нейронні мережі для апроксимації функцій

Створим узагальнено-регресійну НМ з ім'ям *a* для апроксимації функції вигляду $y = x^2$ на відрізку [-1, 1], використовуючи наступні експериментальні дані:

x = [-1 - 0,8 - 0,5 -0,2 0 0,1 0,3 0,6 0,9 1],

y = [1 0,64 0,25 0,04 0 0,01 0,09 0,36 0,81 1].

Процедура створення і використання даної НМ описується таким чином:

```
» P = [-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9 1]; % Задання вхідних значень
» T = [1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1]; % Задання вихідних значень

» Y = sim(a,[-0.9-0.7-0.3 0.4 0.8])% Опитування НМ
Y = 0.8200    0.6400    0.0400    0.0900    0.8100
```

Як видно, точність апроксимації в даному випадку отримали не дуже високою.

Можна спробувати поліпшити якість апроксимації за рахунок підбору величини відхилення, але в умовах прикладу необхідний результат легко досягається шляхом застосування мережі з радиальнобазисними елементами:

```
» a = newrbe(P,T);
»Y = sim(a,[-0.9-0.7-0.3 0.4 0.8]) % Опитування НМ
 Y= 0.8100    0.4900    0.0900    0.1600    0.6400
```

Створену мережу можна зберегти для подальшого використання набором в командному рядку **save('a');** при цьому буде створений файл **a.mat**, тобто файл з ім'ям НМ і розширенням **mat**. У наступних сеансах роботи збережену мережу можна завантажити, використовуючи функцію **load('a').** Природно, допустимі всі інші форми запису операторів **save** і **load**.

Розглянемо тепер аналогічне завдання, але з використанням лінійної НМ.

Хай експериментальна інформація задана значеннями:

x = [+1,0 +1,5 +3,0 -1,2],

y = [+0,5 +1,1 +3,0 -1,0].

Процес створення, навчання і використання лінійної НМ з ім'ям *b* ілюструється приведеними функціями (рис. 1.1).

```
» P = [+1.0 +1.5 +3.0-1.2];
» T = [+0.5 +1.1 +3.0-1.0];
» maxlr = maxlinlr(P,'bias'); % Визначення величини коефіцієнта навчання
» b = newlin([-2 2],1,[0],maxlr); % Створення лінійної НМ з ім'ям b
» b.trainParam.epochs = 15; % Задання кількості циклів навчання
»b = train(b,P,T); % Навчання НМ
TRAINWB, Epoch 0/15, MSE 2.865/0.
TRAINWB, Epoch 15/15, MSE 0.0730734/0.144
TRAINWB, Maximum epoch reached.
» p = -1.2;
» y = sim(b,p)% Опитування мережі
y= -0.8161
```

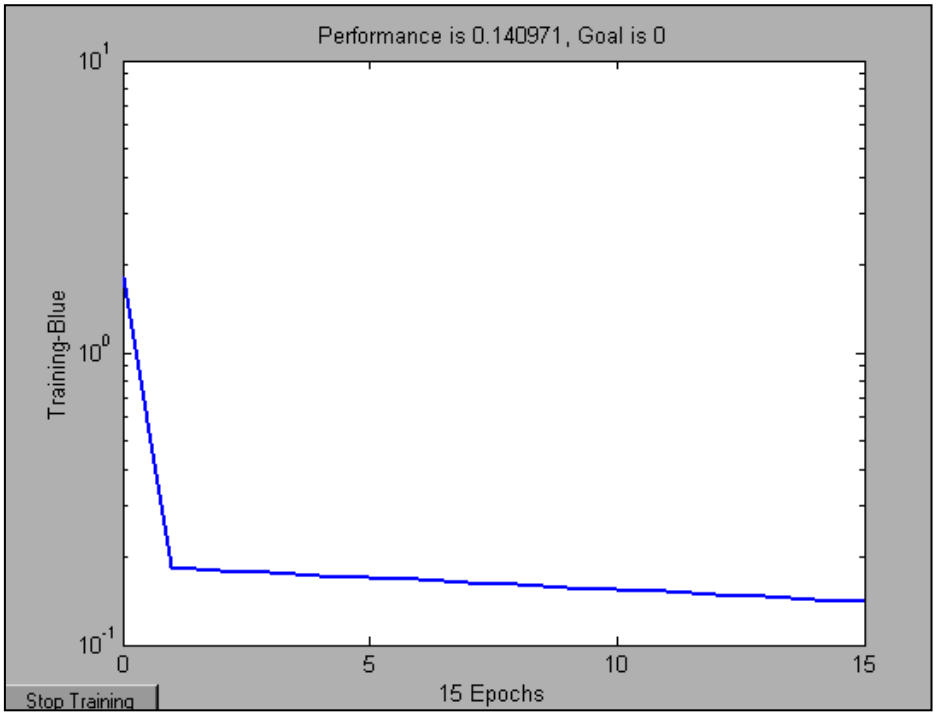


Рис. 1.1. Зміна помилки мережі в процесі її навчання

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклади нейромережевої апроксимації.
2. Згідно завдання викладача провести нейромережеву апроксимацію та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.2. Прогнозування значень процесу

Розглянемо тепер такий приклад. Припустимо, що є сигнал (функція часу), що описується співвідношенням **x(t) = sin (47 π t)**, який піддається дискретизації з інтервалом 0,025с.

Побудуємо лінійну нейронну мережу, яка дозволяє прогнозувати майбутнє значення подібного сигналу за 5 попередніми.

```
» t = 0:0.025:5; % Задання діапазону часу від 0 до 5 секунд
» x = sin(t*4*pi); % Сигнал, що передбачається
» Q = length (x); % Створення вхідних векторів

» P = zeros(5,Q); % Створення нульової матриці P
» P(1,2:Q)=x(1,1:(Q-1));
» P(2,3:Q)=x(1,1:(Q-2));
» P(3,4:Q)=x(1,1:(Q-3));
» P(4,5:Q)=x(1,1:(Q-4));
» P(5,6:Q)=x(1,1:(Q-5));
» s = newlind(P,x); % Створення нової НМ з ім'ям s
» y = sim(s,P); % Розрахунок прогнозованих значень

» % Створення графіків початкового сигналу і прогнозу

» plot(t,y,t,x,'+');
» xlabel('Time');
» ylabel('Predict - Signal ‘+’);
» title ( 'Output neural network ');

» % Розрахунок і створення графіка помилки прогнозу

» e = x-y;
» plot(t,e)
» hold on
»plot([min(t) max(t)],[0 0],':r');
» hold off
» xlabel ('Time');
» ylabel('Mistake');
» title('Mistakes signal');
```

У такому випадку мережа створювалася за допомогою функції **newlind,** при якій не вимагається додаткового навчання. Судячи із графічних результатів (рис. 1.2, 1.3) точність прогнозу з використанням лінійної НМ можна вважати достатньою.

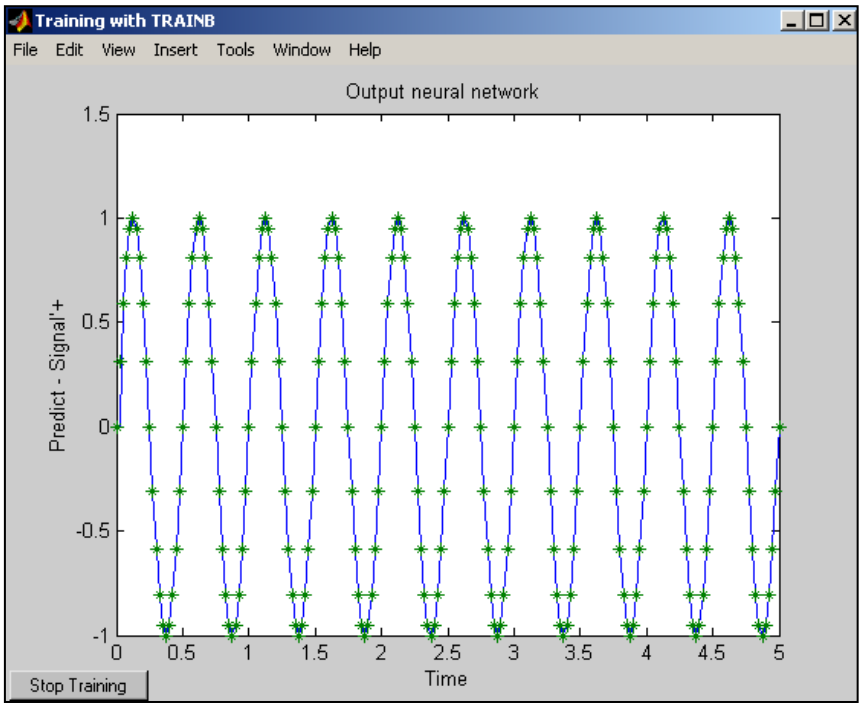


Рис. 1.2. Початковий сигнал і прогноз

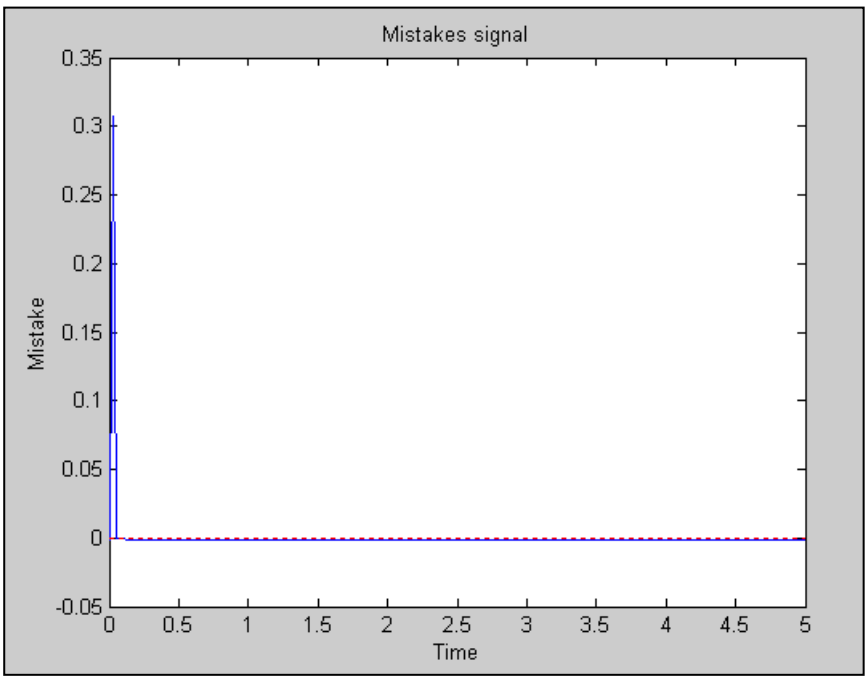


Рис.1.3. Помилка прогнозу

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклади нейромережевого прогнозування.
2. Згідно завдання викладача провести нейромережеве прогнозування та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.3. Використання шару Кохонена

Розглянемо завдання автоматичного виявлення (у режимі навчання без вчителя) центрів кластерів входів для двовимірного випадку з використанням шару Кохонена (шару нейронів, що «змагаються»). Вирішення такої задачі приведене нижче (рис. 2.4).

```
» X = [0 1; 0 1]; % Задання діапазонів можливого положення центрів кластерів
» % Задання параметрів для моделювання початкових даних
» % що належать 8 класам (кластерам)

» clusters = 8;

» points = 10;

» std_dev = 0.05;

» P = nngenc(X,clusters,points,std_dev); % Моделювання вхідних даних
» h = newc([0 1;0 1],8,0.1); % Створення шару Кохонена
» h.trainParam.epochs = 500; % Задання кількості циклів навчання

» h = init(h); % Ініціалізація мережі
» h = train(h,P); % Навчання мережі
» w = h.IW{1};

» %Виведення графіка початкових даних і виявлених центрів кластерів

» plot(P(1,:),P(2,:),'+r');
» hold on; plot(w(:,1),w(:,2),'ob');
» xlabel('p(1)');
» ylabel('p(2)');
» p = [0; 0.2]; % Задання нового вхідного вектора

» y = sim(h,p)% Опитування мережі

» y= (8,1)      1
```

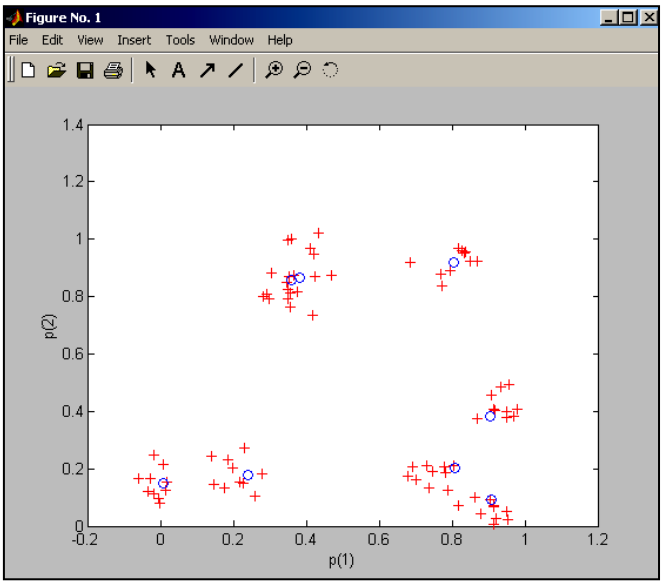


Рис. 1.4. Початкові дані і виявлені центри кластерів

Роботу і результат опитування НМ (видаються у формі розрідженої матриці) ілюструє рисунок 1.4 . В умовах прикладу пред'явлений вектор віднесений до восьмого класу (кластеру).

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклади кластеризації із використанням шару Кохонена.
2. Згідно завдання викладача провести кластеризацію із використанням шару Кохонена та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.4. Мережа Хопфілда з двома нейронами

Розглянемо мережу Хопфілда, що має два нейрони і два стійкі стани, які відображаються векторами [1 -1] і [-1 1]. Представимо ці вектори за допомогою рисунку 1.5, що виводиться програмою:

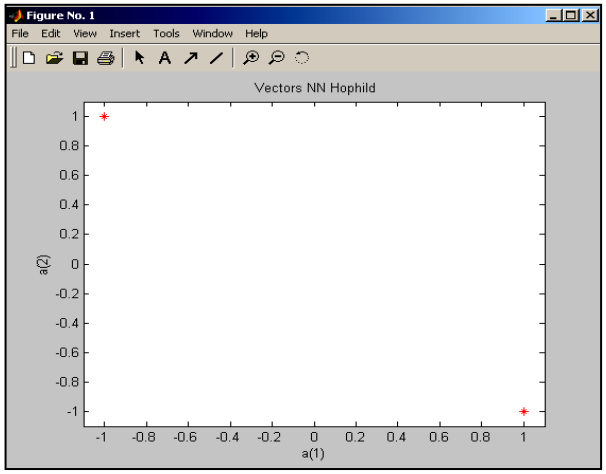


Рис. 1.5. Стійкі точки мережі Хопфілда

```
» T = [+1 -1;-1 +1];
» plot(T(1,:),T(2,:),'r +')
» axis([-1.1 1.1 -1.1 1.1]);
» title(‘ Vectors NN Hophild’);
» xlabel('a(1)'); » ylabel('a(2)');
```

Створимо мережу Хопфілда (з ім'ям H) і перевіримо її роботу, подавши на вхід вектори, відповідні стійким точкам. Якщо мережа працює правильно, вона повинна виробити ці ж вектори без яких-небудь змін.

```
» H = newhop(T); % Створення НМ Хопфілда
» [Y,Pf,Af] = sim(H,2,[ ],T);Y % Опитування мережі Хопфілда
  Y = 1     -1
      -1     1
```

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклади налаштування мережі Хопфілда.
2. Згідно завдання викладача провести налаштування мережі Хопфілдата та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.5. Класифікація за допомогою персептрона

Наступний приклад ілюструє рішення задачі класифікації за допомогою персептрона (рис. 2.6). Початкові вхідні вектори (при їх належності до одного з двох класів) і результат налаштування персептрона (з ім'ям My_net) представлені на рисунку.

```
» % Задання вхідних векторів з вказівкою їх належності
» % одному з двох класів
» P = [-0.5 -0.5 +0.3 -0.1;-0.5 +0.5 -0.5 +1.0];
» T=[1 1 0 0];
» plotpv(P,T); % Графічне представлення початкових векторів

» % Створення персептрона з вказівкою меж змін входів і 1 нейроном
» My_net=newp([-1 1;-1 1],1);
» E=1;
» My_net =init(My_net); % Ініціалізація персептрона

» % Організація циклу адаптивного налаштування персептрона
» % з виведенням графіка розділяючої лінії
» while (sse(E))
  [My_net,Y,E] = adapt(My_net,P,T);
  linehandle = plotpc(My_net.IW{1},My_net.b{1});
  drawnow;
  end;
```

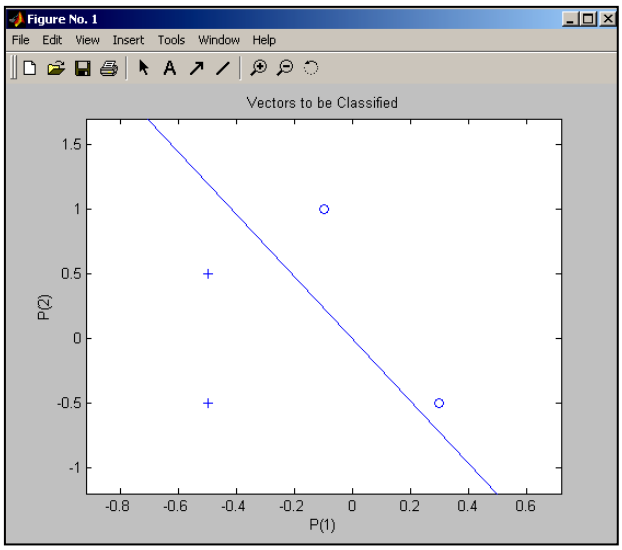


Рис. 1.6. Початкові вхідні вектори і розділяюча лінія

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений прикладу використання персептрона.
2. Згідно завдання викладача провести налаштування персептрона та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.6. Адаптивний лінійний прогноз

У попередньому прикладі налаштування НМ проводилось адаптивно. Відмінність такої настройки від виконаної, наприклад, за допомогою методу зворотного розповсюдження помилки, полягає в тому, що вектори навчальної вибірки поступають на вхід мережі не всі «одночасно», а послідовно, поодинці, при цьому після пред'явлення чергового вектора проводиться коректування вагів і зсувів і може бути проведене опитування мережі, потім все повторюється. Адапація налаштування особливо зручна при роботі НМ у «реальному» режимі часу.

Розглянемо приклад задання з прогнозуванням значень сигналу (за 5-ма попередніми значеннями) з використанням вказаного налаштування.

Припустимо, що початковий сигнал визначений на інтервалі часу від 0 до 6 с., при 0 < t < 4 с. він описується співвідношенням $x(t)= sin(47 \pi t)$, а при 4 < t < 6 с. співвідношенням $x(t)= sin(8 \pi t)$. Графік такого сигналу приведений на рисунку 1.7:

```
» time1 = 0:0.05:4; % від 0 до 4 секунд
» time2 = 4.05:0.024:6; % від 4 до 6 секунд
» time= [time1 time2];

» % T визначає початковий сигнал:
» T = con2seq([sin(time1*4*pi) sin(time2*8*pi)]);
```

» % Графік початкового сигналу:
» plot(time,cat(2,T{:}))
» xlabel('Time');
» ylabel (' Start signal ');
» title('Predict signal ');

Для прогнозу значень сигналу створимо лінійну НМ.
» % Вхідний і цільовий прогнозований сигнал однакові:
» P=T;
» % Задання коефіцієнта навчання
» lr = 0.1;
» % Для прогнозу використовуються 5 попередніх значень
» delays = [1 2 3 4 5];

» % Створення і налаштування лінійної НМ
» net = newlin(minmax(cat(2,P{:})),1,delays,lr); % Створення НМ
» [net,y,e] = adapt(net,P,T); % Адаптивне налаштування мережі

» % Графіки початкового сигналу і прогнозу
»plot(time,cat(2,y{:}),time,cat(2,T{:}),'--')
» xlabel('Time');
» ylabel (' Predict signal ');
» title ('Start signal and predict '); (Рис 1.8)

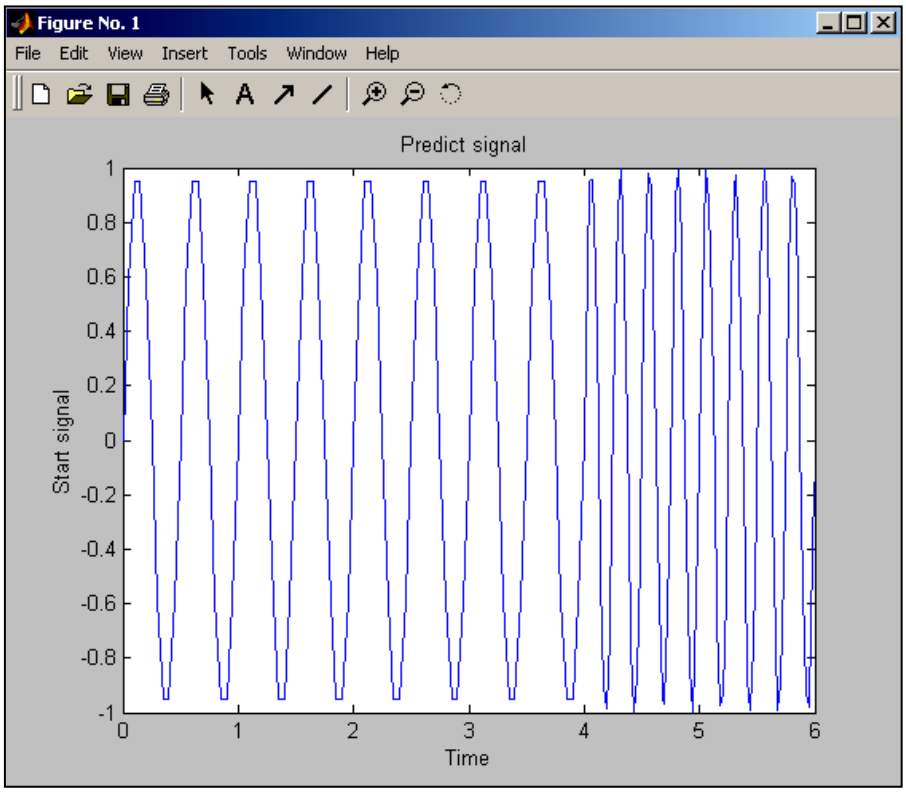
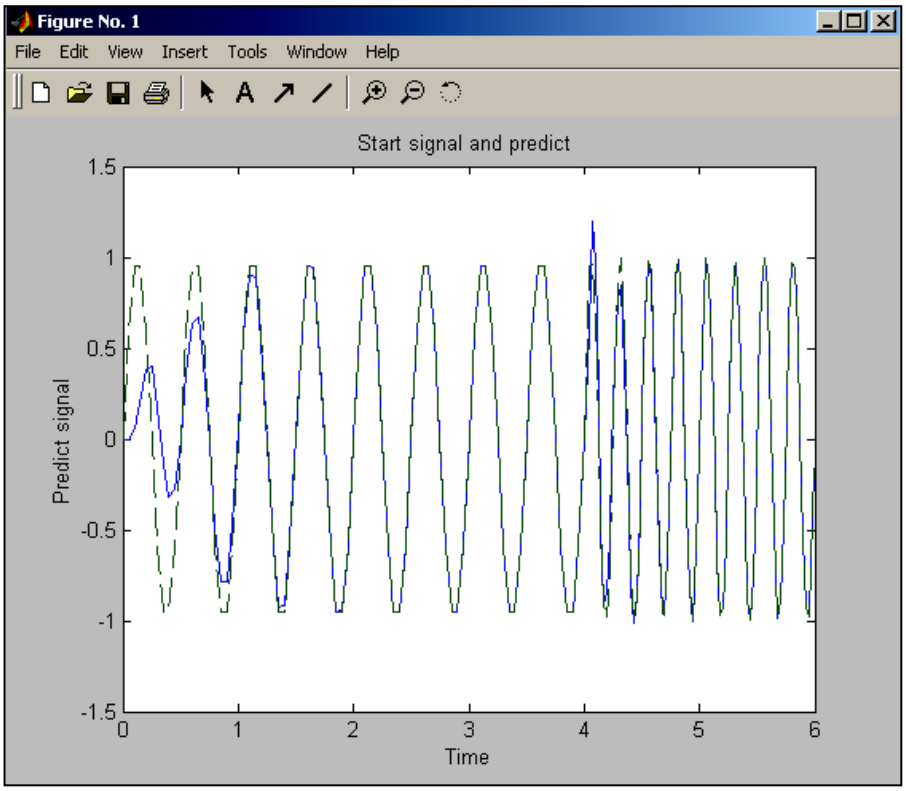


Рис. 1.7. Графік прогнозованого сигналу



Мал. 1.8. Початковий сигнал і прогноз

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклад адаптивного лінійного прогнозу.
2. Згідно завдання викладача провести адаптивний лінійний прогноз та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.7. Мережі Елмана

Розглянемо завдання відновлення форми сигналу з використанням рекурентної мережі Елмана. Нехай є два синусоїдальні сигнали – один з одиничною амплітудою, інший – з амплітудою, яка дорівнює двом:

```
p1 = sin(1:20);
p2 = sin(1:20)*2.
```

Нехай цільовим сигналом буде сигнал, складений з їх амплітудних значень:

```
t1 = ones(1,20);
t2 = ones(1,20)*2;
```

при цьому дані амплітуди чергуються, так що вхідні і цільові значення можуть бути представлені у формі:

```
p=[p1p2p1p2];
t = [t1 t2 t1 t2].
```

Перетворимо ці значення в послідовності:

Pseq = con2seq(p);

Tseq = con2seq(t).

Після цього можна безпосередньо перейти до проектуванню НМ. У ній запроектуємо один вхід і один вихід, тобто мережа матиме один вхідний елемент і один вихідний нейрон. Число нейронів у прихованому шарі може бути будь-яким (воно залежить від складності завдання); приймемо, що цей шар містить 10 нейронів. Подальше вирішення завдання ілюструється нижчим, при цьому на рисунку 1.9 приведено графік зміни помилки мережі в процесі її навчання, а на рисунку. 1.10 результати тестування мережі.

```
» % Задання початкових даних
» p1 = sin(1:20);
» t1 = ones(1,20);
» p2 = sin(1:20)*2;
» t2 = ones(1,20)*2;
» p = [p1 p2 p1 p2];
» t = [t1 t2 t1 t2];
» Pseq = con2seq(p);
» Tseq = con2seq(t);
»% Створення мережі Елмана з діапазоном входу [-2, 2] та 10 нейронами
» % прихованого шару, одним вихідним нейроном,
» % функцією активації у вигляді гіперболічного тангенса
» % для нейронів прихованого шару, лінійною функцією активації
» % для вихідного нейрона, функцією навчання з адаптацією
» % коефіцієнта навчання
» net = newelm([-2 2],[10 1],{ 'tansig', 'purelin'},'traingdx');
» % Задання параметрів навчання:
» net.trainParam.epochs = 500; % Число циклів навчання
» net.trainParam.goal= 0.01; % Цільове значення функції помилки
» net.performFcn = 'sse'; % Задання вигляду функції помилки

» % Навчання мережі. За умовчанням проміжні результати навчання
» % виводяться через 25 циклів
» [net,tr] = train(net,Pseq,Tseq);

» % Побудова графіка функції помилки
» semilogy(tr.epoch,tr.perf);
» title( 'The sum of squares of the errors Elman’s network');
» xlabel(' Cicles');
» ylabel( ' The sum of squares of the errors');

» % Тестування мережі
» a = sim(net,Pseq);
» time= 1:length(p);
» time= 1:length(p);
```

```
» plot(time,t,'-',time,cat(2,a{:}))
» title ( 'Testing results');
» xlabel( 'Time');
» ylabel( 'Preset values- - Neural networks ---')
```

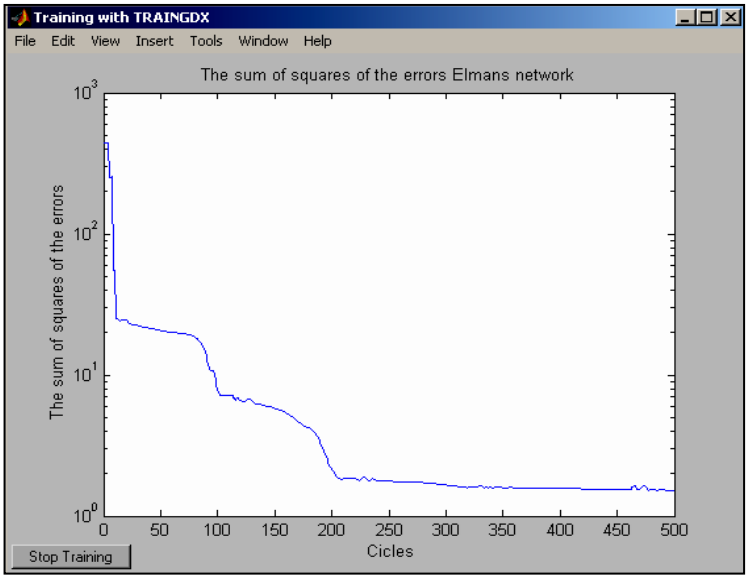


Рис. 1.9. Зміна помилки мережі в процесі навчання

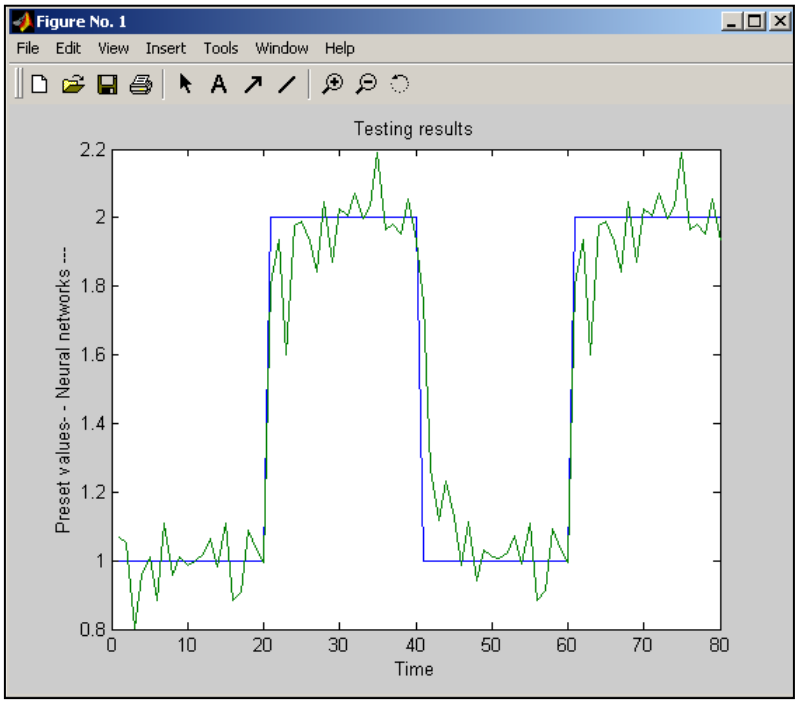


Рис. 1.10. Результати тестування мережі

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклад використання мережі Елмана.
2. Згідно завдання викладача реалізувати НМ Елмана та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

1.8. Мережі зустрічного розповсюдження

Припустимо, поставлено наступне завдання класифікації: задано набір з 10 векторів, представлених у вигляді стовпців-матриць, також задано вектор-рядок, який вказує належність кожного вектора до одного з двох класів:

Tc = [1 1 1 2 2 2 2 1 1 1].

Потрібно: побудувати автоматичний класифікатор подібних векторів, використовуючи наведені дані як навчальну вибірку.

Рішення подібної задачі проведемо із застосуванням мережі зустрічного розповсюдження:

```
» P = [-3 -2 -2 0 0 0 0 +2 +2 +3; 0 +1 -1 +2 +1 -1 -2 +1 -1 0];
» C = [1 1 1 2 2 2 2 1 1 1];
» T = ind2vec(C); % Перетворення вектора C в матрицю T з двома рядками

» % Створення нової мережі зустрічного розповсюдження вимагає 4-х па-раметрів:
» % 1) матриці мінімальних і максимальних значень вхідних елементів
» % 2) числа прихованих нейронів
» % 3) вектор з елементами, які вказують частку кожного з класів
» % 4) величини коефіцієнта навчання

» net = newlvq(minmax(P),4,[.6 .4],0.1); % Створення мережі
» net.trainParam.epochs = 150; % Задання числа циклів навчання
»net.trainParam.show = Inf; % Заборона на видачу проміжних результатів
» net = train(net,P,T); % Навчання НМ

TRAINWB1, Epoch 150/150
TRAINWB1, Maximum epoch reached.

» Y = sim(net,P)% Тестування мережі

Y=
     1   1   1   0   0   0   0   1   1   1
     0   0   0   1   1   1   1   0   0   0
```

Як видно з результатів тестування, класифікація елементів навчальної вибірки проведена точно (три перших і три останні вектори віднесено до першого класу, останні – до другого).

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведений приклад застосування НМ зустрічного розповсюдження.
2. Згідно завдання викладача реалізувати НМ зустрічного розповсюдження та нарисувати блок-схему роботи НМ з розшифровкою функцій та їх аргументів.

2. Створення і використання нейронних мереж у середовищі Simulink

Пакет Neural Network Toolbox містить ряд блоків, які або можуть бути безпосередньо використані для побудови нейронних мереж в середовищі *Simulink*, або застосовуватися разом з розглянутою вище функцією **gensim**.

Для виклику цього набору блоків, в командному рядку необхідно набрати команду **neural**, після виконання який з'являється відповідне робоче вікно (2.1).

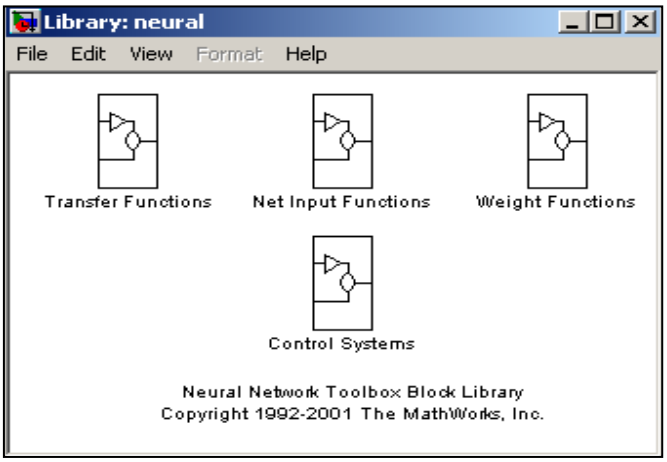


Рис 2.1. Основні нейромережеві блоки Simulink

Кожен з представлених на рисунку 2.1 блоків у свою чергу є набором (бібліотекою) деяких блоків. Розглянемо їх.

Блоки функцій активації (Transfer Functions). Подвійний клік лівої кнопки миші на блоці Transfer Functions призводить до появи бібліотеки функцій активації (рис. 2.2). Кожен з блоків даної бібліотеки перетворить поданий на нього вектор у відповідний вектор тієї ж розмірності.

Блоки перетворення входів мережі. Проводячи аналогічну розглянутій операцію, але з блоком *Net Input Functions*, прийдемо до бібліотеки блоків представлених на рисунку 2.3. Блоки даної бібліотеки реалізують розглянуті вище функції перетворення входів мережі.

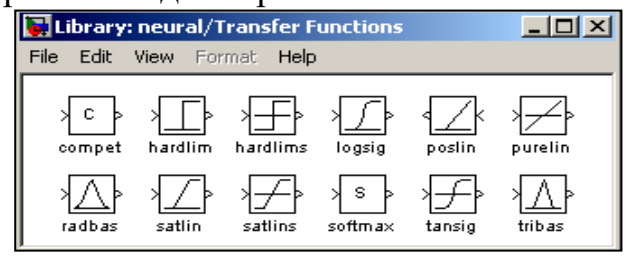


Рис. 2.2. Бібліотека функцій активації

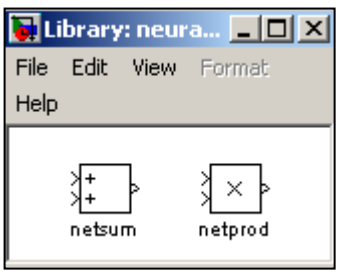


Рис. 2.3. Блоки бібліотеки перетворення входів мережі

Інша бібліотека – бібліотека блоків, що реалізовують функції вагів і зсувів.

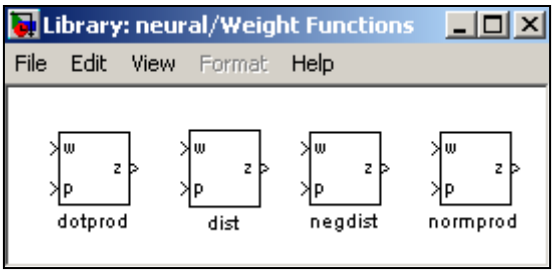


Рис. 2.4. Блоки бібліотеки функцій вагів та зсувів

Відзначимо, що при задані конкретних числових значень, під час операції із всіма приведеними блоками зважаючи на особливості Simulink вектори необхідно представляти як стовпці, а не як рядки (як це було до цих пір).

Основною функцією для формування нейромережевих моделей в *Simulink* є функція **gensim**, яка записується у формі: **gensim(net,st)**, де: **net** – ім'я створеної НМ, **st** – інтервал дискретизації (якщо НМ не має затримок, що асоціюються з її входами або шарами, значення даного аргументу встановлюється рівним -1).

Приклад

Хай вхідний і цільовий вектори мають вигляд:

p=[1 2 3 4 5];

t=[1 3 5 7 9].

Створимо лінійну НМ і протестуємо її:

```
» p = [1 2 3 4 5];
» t = [1 3 5 7 9];
» net = newlind(p,t);
» y = sim(net,p)
   y= 1.0000     3.0000     5.0000     7.0000     9.0000
```

Потім запустимо Simulink командою » **gensim(net, -1)** (рис. 2.5).

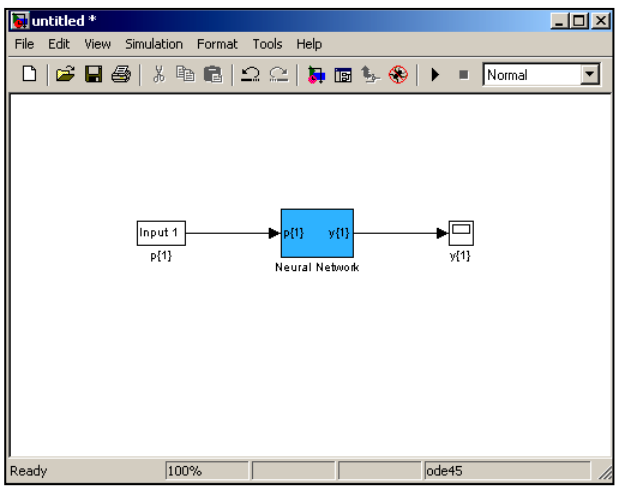


Рис. 2.5. Створена нейромережева модель Simulink

Для проведення тестування моделі клікнемо двічі по лівій іконі (Input 1), що приведе до відкриття діалогового вікна (рис. 2.6).

В даному випадку блок Input 1 є стандартним блоком задання константи (Constant). Змінимо значення за умовчанням на 2 і натиснемо кнопку OK. Потім натиснемо кнопку Start в меню моделювання. Розрахунок нового значення мережею проводиться практично миттєво. Для його виводу необхідно двічі клацнути мишею по правій іконі (блок y(1)). Результат обчислень відображається на інтерфейсному дисплеї, результат дорівнює 3, як і потрібно.

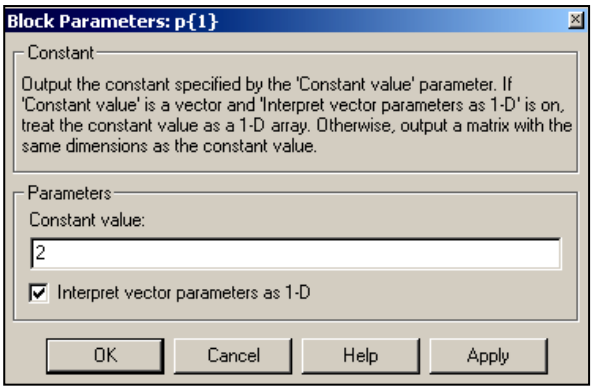


Рис. 2.6. Діалогове вікно задання входу НМ

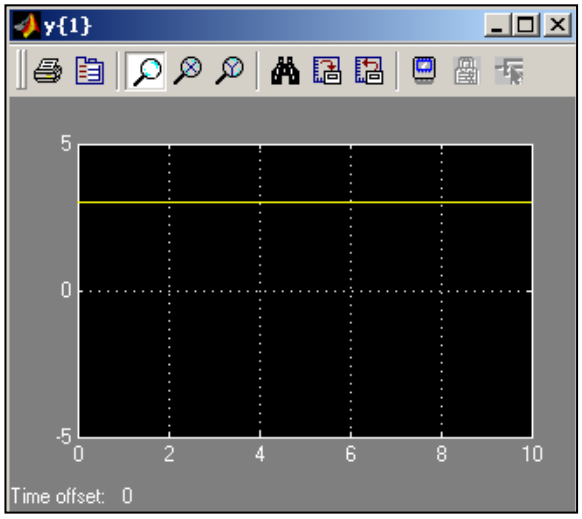


Рис. 2.6. Вікно з виходом НМ

Відзначимо, що, двічі клікаючи лівою кнопкою миші по блоку **Neural Network**, потім – по блоку **Layer 1**, можна отримати детальну графічну інформацію щодо архітектури мережі (рис. 2.7).

Із створеною мережею можна проводити різні операції, можливі в середовищі Simulink; взагалі за допомогою команди **gensim** здійснюється інтеграція створених нейромереж у блок-діаграми цього пакету з використанням інструментів моделювання різних систем, що є при цьому (наприклад, вбудовувати нейромережевий регулятор в систему управління і моделювання).

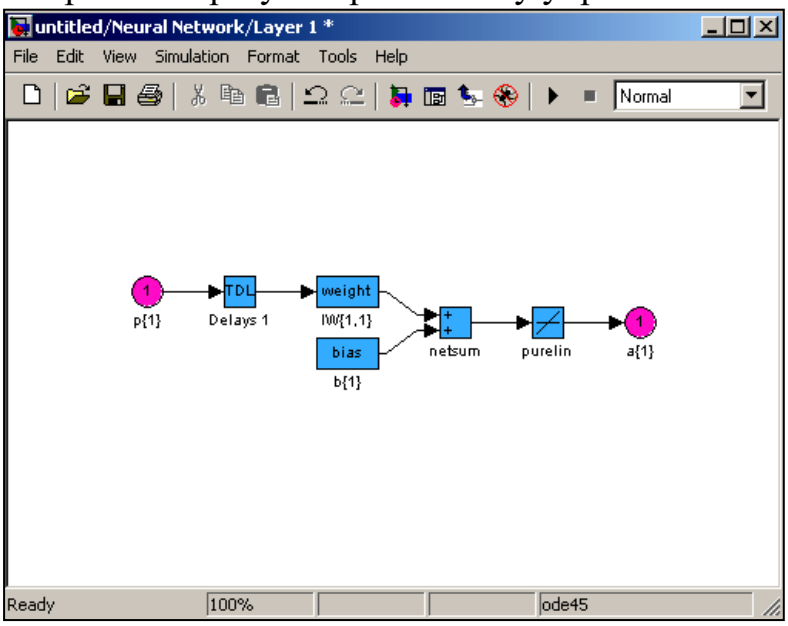


Рис. 2.7. Архітектура створеної НМ

Завдання

1. Реалізувати у *Simulink* наведений приклад застосування лінійної НМ.
2. Згідно завдання викладача реалізувати лінійну НМ у *Simulink*.