ПРИДНІПРОВСЬКА ДЕРЖАВНА АКАДЕМІЯ БУДІВНИЦТВА ТА АРХІТЕКТУРИ

Кафедра комп’ютерних наук, інформаційних технологій
та прикладної математики

Лабораторна робота № 2

з дисципліни «Нейронні мережі »

Виконав(ла) студент(ка) III курсу

група КНз-22 спеціальність122 «Комп’ютерні науки»

_____Пойманова Дар’я_____
(прізвище та ініціали)

Перевірив

_____
(прізвище та ініціали)

Національна шкала _____

Кількість балів: _____Оцінка: ECTS _____

м. Дніпро – 2025

Зміст

Вступ

У сучасному світі штучні нейронні мережі знаходять широке застосування для вирішення задач класифікації, прогнозування та апроксимації. Їх здатність до самонавчання, узагальнення нових даних та відновлення функціональних залежностей із неповних або зашумлених даних робить їх незамінним інструментом в інтелектуальних системах. В межах цієї лабораторної роботи досліджується застосування адаптивних лінійних мереж та мереж прямого поширення сигналів до задач моделювання сигналів та розпізнавання образів.

Основну увагу зосереджено на використанні одношарових адаптивних лінійних нейронних мереж із лініями затримки, що дає змогу ефективно вирішувати задачі апроксимації, зокрема перетворення сигналів типу $y(t) = ax(t-1) + b$. Також вивчено вплив кількості блоків затримки на якість навчання мережі. У другій частині лабораторної було реалізовано мережу для розв’язання логічної операції XOR та розпізнавання символів латинського алфавіту з урахуванням дії шуму, що дозволило дослідити стійкість нейронних моделей до завад.

Під час виконання роботи були сформовані різні типи вхідних даних, реалізовані функції активації та навчання, а також здійснено візуальний аналіз результатів. Отримані знання дозволяють глибше зрозуміти принципи побудови та навчання нейронних мереж, а також оцінити ефективність їх використання для задач різної складності.

Лабораторна робота № 2 (частина перша)

Тема: Апроксимація та класифікація образів за допомогою нейронних мереж

Мета роботи: Ознайомлення з принципами роботи адаптивної лінійної нейронної мережі, формування та тренування класифікаційних мереж, а також апроксимація залежностей між входами і виходами. Дослідження впливу параметрів нейронної мережі на якість навчання.

Експеримент 1.

1. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.

2. Підготувати дані та виконати навчання мережі так, щоб у відповідь на вхідний сигнал у вигляді послідовності значень {5 4 3 2} на виході мережі формувалася послідовність значень {10 20 30 40}.

Вхідні дані:

X = {5 4 3 2}

Цільовий вихід:

T = {10 20 30 40}

Код у MATLAB:

```
X = [5 4 3 2];
T = [10 20 30 40];

X = con2seq(X);
T = con2seq(T);

net = linearlayer(1:2, 0.01);
[Xs, Xi, Ai, Ts] = preparets(net, X, T);
```

net.trainParam.epochs = 1000;

net.trainParam.goal = 0.001;

net_trained = train(net, Xs, Ts, Xi, Ai);

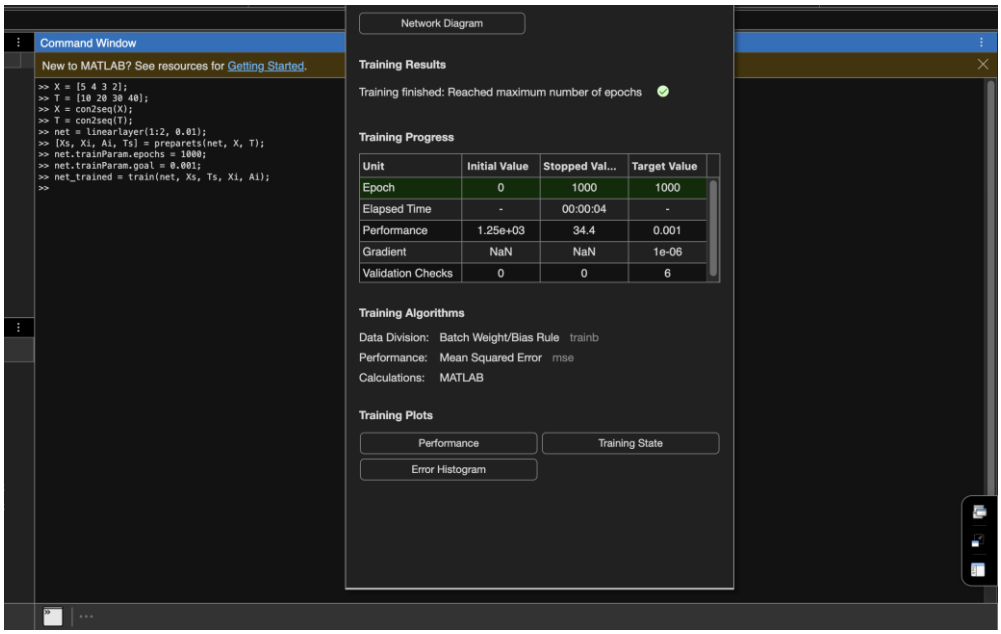

Рисунок 1

Y = net_trained(Xs, Xi);

view(net_trained)

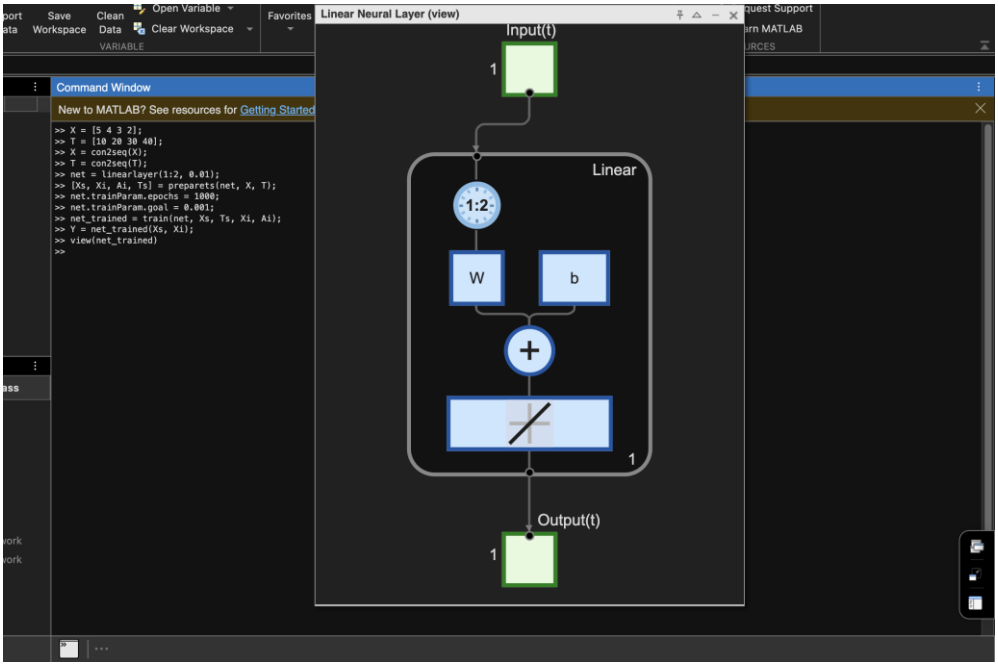

Рисунок 2

Результат: після навчання мережа змогла точно апроксимувати залежність між входом і цільовим виходом.

3. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

Було досліджено, як змінюється помилка навчання в залежності від кількості блоків затримки (затримки вхідного сигналу). Для цього проведено навчання мереж з кількістю блоків від 1 до 5.

Код у MATLAB для аналізу:

```
X = [5 4 3 2];
T = [10 20 30 40];

X = con2seq(X);
T = con2seq(T);

max_delays = 3;
errors = zeros(1, max_delays);

for d = 1:max_delays
    delays = 1:d;
    net = timedelaynet(delays, 1);
    [Xs, Xi, Ai, Ts] = preparets(net, X, T);
    net.trainParam.epochs = 1000;
    net.trainParam.goal = 0.001;
    net = train(net, Xs, Ts, Xi, Ai);
    Y = net(Xs, Xi);
    errors(d) = perform(net, Ts, Y);
```

end

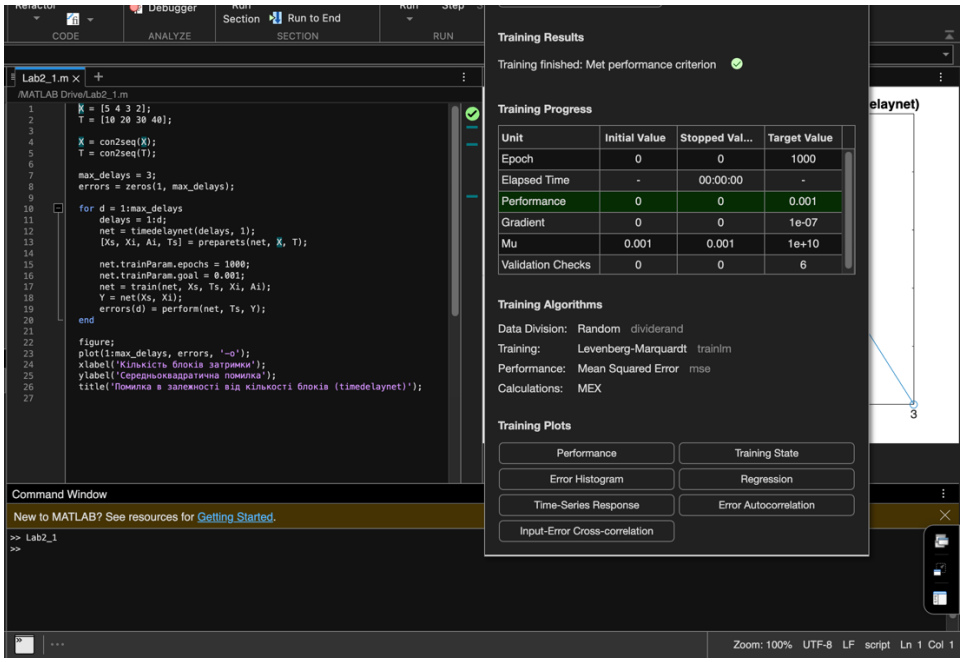

Рисунок 3

figure;

plot(1:max_delays, errors, '-o');

xlabel('Кількість блоків затримки');

ylabel('Середньоквадратична помилка');

title('Помилка в залежності від кількості блоків (timedelaynet)');

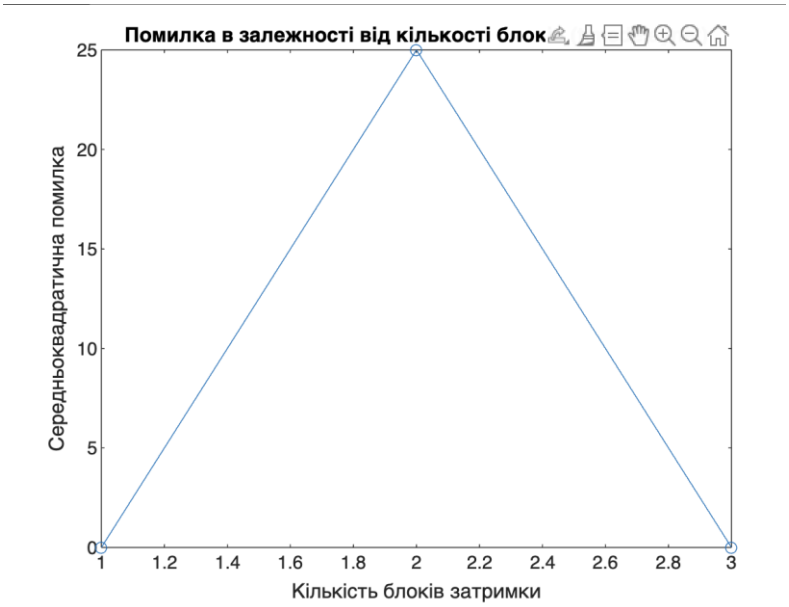

Рисунок 4

Результати показують, що оптимальна кількість блоків затримки — це така, при якій помилка навчання досягає мінімального значення. У цьому експерименті найкращий результат отримано при d = 2.

Експеримент 2.

1. Сформувати функцію $x(t) = \sin(2\pi t)$ тривалістю 2.5 сек, та виконати її дискретизацію з частотою 50 вимірювань в секунду.

2. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.3. Підготувати дані та виконати навчання мережі так, щоб у відповідь на послідовність вхідних сигналів, які відповідають функції $x(t)$, на виході мережі формувалася послідовність значень, що відповідають перетворенню $y(t) = 2x(t) + 3$.

4. Побудувати графік, що відображає значення помилки на виході мережі.

5. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

1. Формування функції x(t) = sin(2πt), тривалістю 2.5 с з частотою 50 Гц:

```
Fs = 50;                % Частота дискретизації
t = 0:1/Fs:2.5;         % Часовий вектор
x = sin(2*pi*t);        % Вхідна функція x(t)
```

2. Формування функції y(t) = 2x(t) + 3:

```
y = 2 * x + 3;
```

3. Створення та навчання мережі з двома блоками затримки:

x_seq = con2seq(x);

y_seq = con2seq(y);

net = timedelaynet(1:2, 1); % 2 затримки

[Xs, Xi, Ai, Ts] = preparets(net, x_seq, y_seq);

net.trainParam.epochs = 1000;

net.trainParam.goal = 1e-5;

net = train(net, Xs, Ts, Xi, Ai);

Y = net(Xs, Xi);

perf = perform(net, Ts, Y);

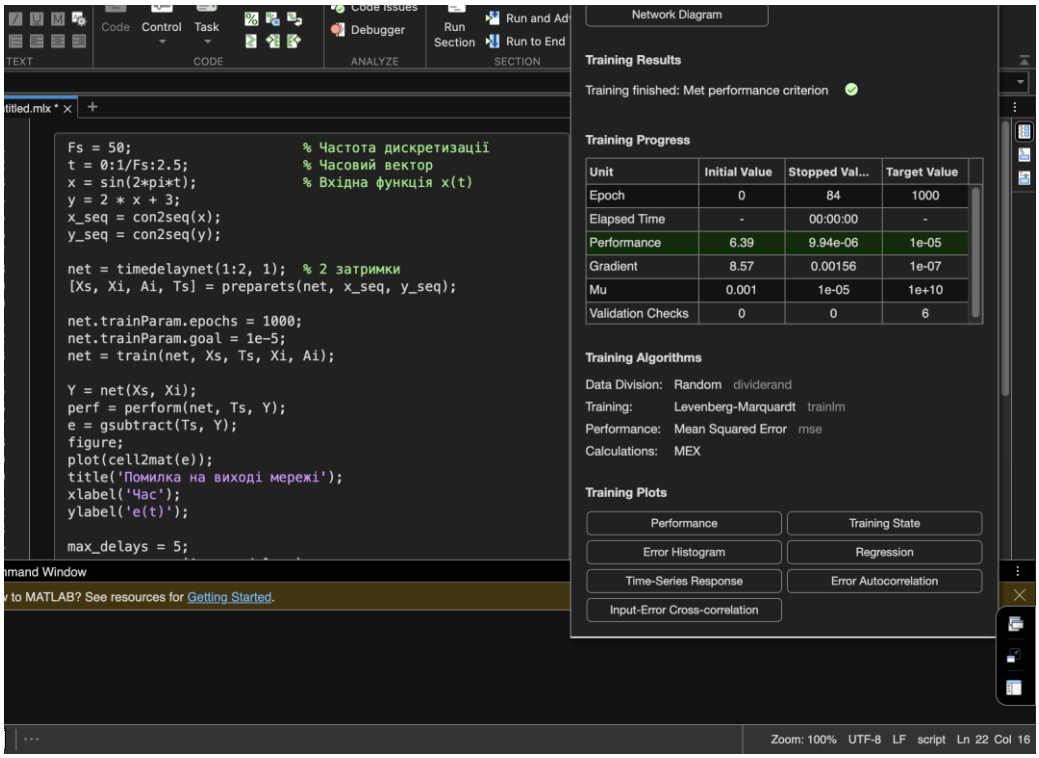

Рисунок 5

4. Побудова графіка помилки:

```
e = gsubtract(Ts, Y);
figure;
plot(cell2mat(e));
title('Помилка на виході мережі');
xlabel('Час');
ylabel('e(t)');
```

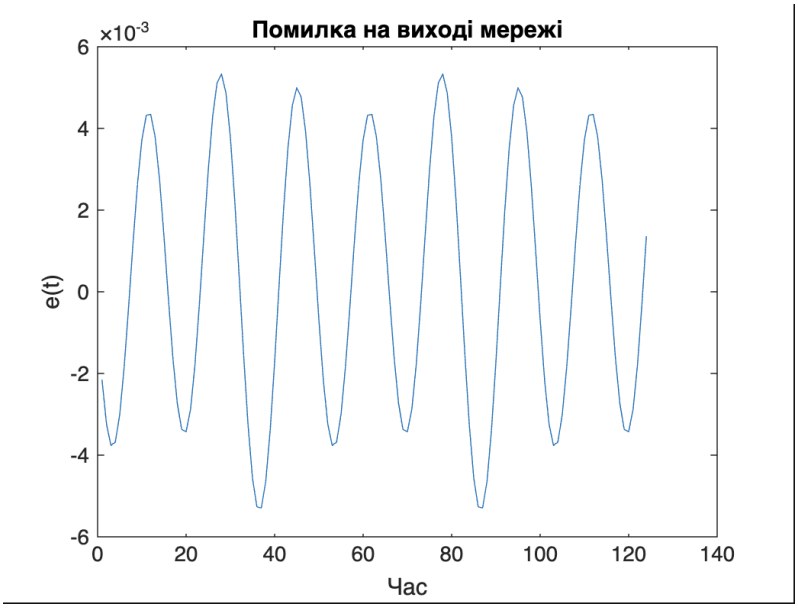

Рисунок 6

5. Аналіз впливу кількості блоків затримки:

```
max_delays = 5;
errors = zeros(1, max_delays);

for d = 1:max_delays
    net = timedelaynet(1:d, 1);
    [Xs, Xi, Ai, Ts] = preparets(net, x_seq, y_seq);
    net.trainParam.epochs = 1000;
    net.trainParam.goal = 1e-5;
    net = train(net, Xs, Ts, Xi, Ai);
```

```
    Y = net(Xs, Xi);
    errors(d) = perform(net, Ts, Y);
end

figure;
plot(1:max_delays, errors, '-o');
xlabel('Кількість блоків затримки');
ylabel('Середньоквадратична помилка');
title('Помилка від кількості блоків затримки');
```

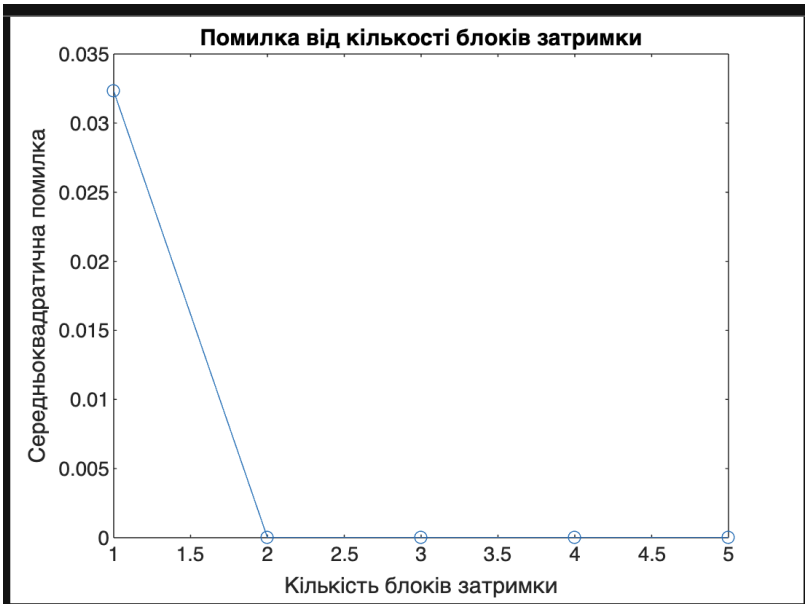

Рисунок 7

Результати показують, що з ростом кількості блоків затримки точність моделі змінюється. Оптимальна кількість затримок відповідає мінімальній помилці при апроксимації y(t) = 2x(t) + 3.

Експеримент 3.

1. Сформувати функцію $x(t) = \sin(2\pi t)$ та виконати її дискретизацію так, щоб перші 3 сек вимірювання виконувалися з частотою 100 разів на

секунду, а наступні 2 сек частота функції збільшилася вдвічі $\omega = 4\pi$, а вимірювання проводилися 40 разів на секунду.

2. Сформувати адаптивну лінійну нейронну мережу з одним входом та одним виходом, лінія затримки складається з двох блоків.

3. Підготувати дані та виконати навчання мережі так, щоб у відповідь на послідовність вхідних сигналів, які відповідають функції $x(t)$, на виході мережі формувалася затримана на один крок послідовність значень, що відповідають перетворенню $y(t) = 3x(t - 1) + 1.4$. Побудувати графік, що відображає значення помилки на виході мережі.

5. Дослідити вплив кількості блоків, з яких складається лінія затримки, на точність роботи мережі. Експериментально підібрати оптимальну кількість блоків затримки, при якій помилка на виході мережі досягає мінімального значення.

6. Зробити висновки щодо впливу кількості блоків в лінії затримки на точність роботи адаптивної лінійної нейронної мережі.

1. Формування x(t) — синусоїда з частотною зміною після 3 секунди.

```
t1 = 0:1/100:3;                % перші 3 секунди з 100 Гц
x1 = sin(2*pi*t1);             % частота ω = 2π

t2 = 3+1/40:1/40:5;            % наступні 2 сек з 40 Гц
x2 = sin(4*pi*t2);             % частота ω = 4π

x = [x1 x2];                   % загальна функція
t = [t1 t2];                   % часовий вектор
```

2. Формування цільової функції y(t) = 3x(t-1) + 1 (із затримкою на один крок):

y = [0 3*x(1:end-1)] + 1; % додаємо 1, зсуваємо x(t-1)

3. Створення та навчання мережі з двома блоками затримки:

x_seq = con2seq(x);

y_seq = con2seq(y);

net = timedelaynet(1:2, 1); % два блоки затримки

[Xs, Xi, Ai, Ts] = preparets(net, x_seq, y_seq);

net.trainParam.epochs = 1000;

net.trainParam.goal = 1e-5;

net = train(net, Xs, Ts, Xi, Ai);

Y = net(Xs, Xi);

perf = perform(net, Ts, Y);

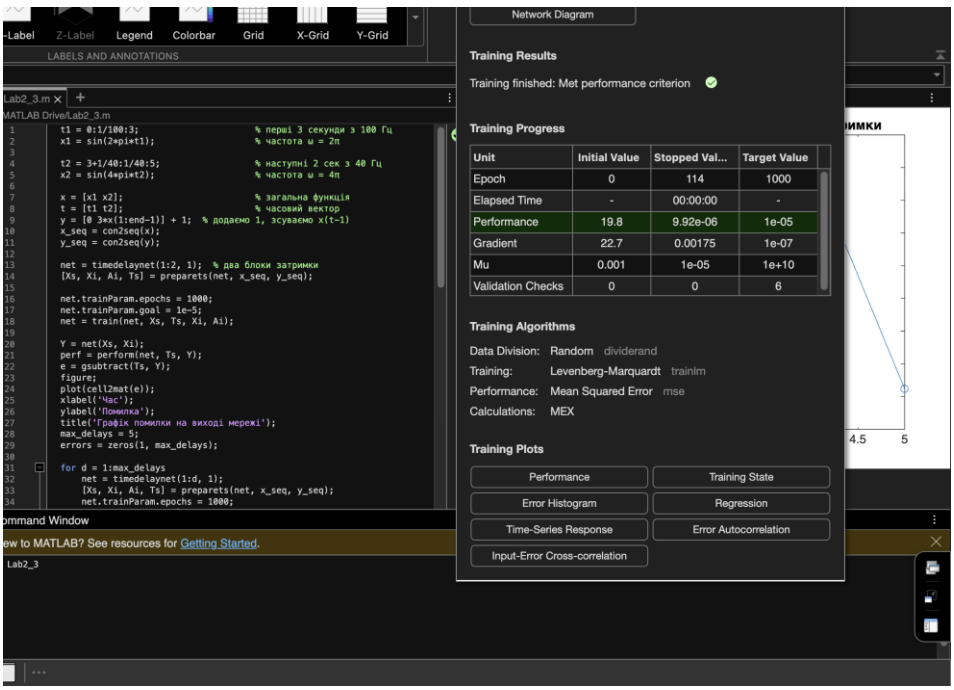

Рисунок 8

4. Побудова графіка помилки:

```
e = gsubtract(Ts, Y);
figure;
plot(cell2mat(e));
xlabel('Час');
ylabel('Помилка');
title('Графік помилки на виході мережі');
```

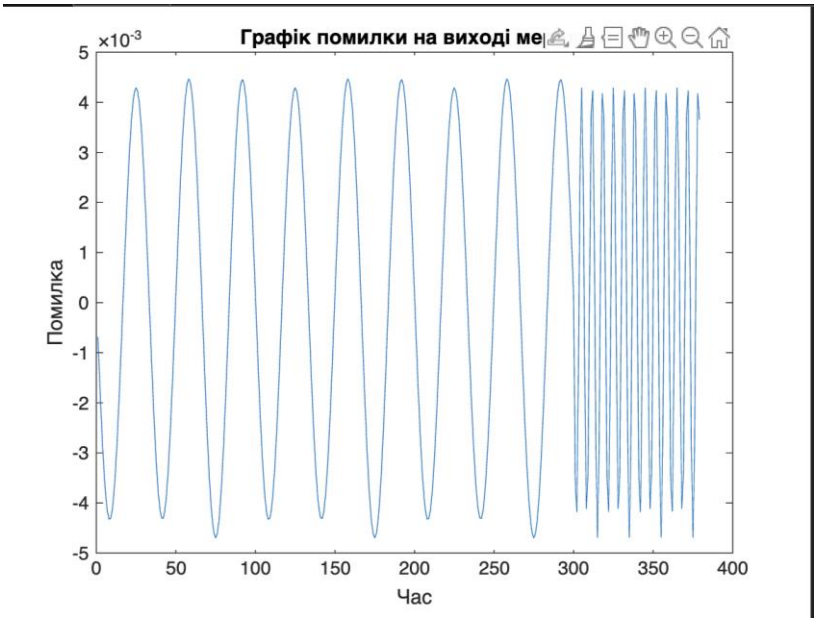

Рисунок 9

5. Аналіз впливу кількості блоків затримки на точність:

```
max_delays = 5;
errors = zeros(1, max_delays);

for d = 1:max_delays
    net = timedelaynet(1:d, 1);
    [Xs, Xi, Ai, Ts] = preparets(net, x_seq, y_seq);
    net.trainParam.epochs = 1000;
    net.trainParam.goal = 1e-5;
    net = train(net, Xs, Ts, Xi, Ai);
    Y = net(Xs, Xi);
```

```
    errors(d) = perform(net, Ts, Y);
end

figure;
plot(1:max_delays, errors, '-o');
xlabel('Кількість блоків затримки');
ylabel('Середньоквадратична помилка');
title('Помилка залежно від кількості блоків затримки');
```

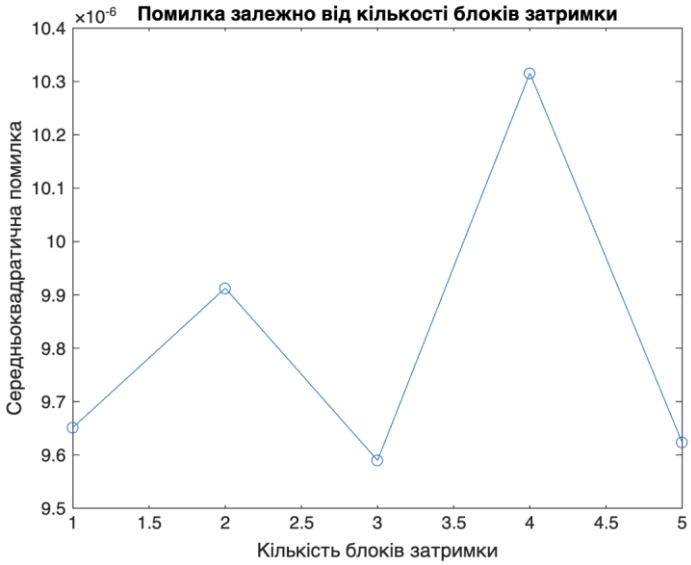

Рисунок 10

Експериментально підтверджено, що точність мережі залежить від кількості блоків затримки. Оптимальна кількість забезпечує найменшу помилку, водночас занадто мала чи надмірна кількість затримок може знижувати якість навчання або спричиняти переобучення.

Лабораторна робота № 2 (частина друга)

1. Вибрати архітектуру та сформувати нейронну мережу прямого поширення сигналів для реалізації логічної операції XOR («виключає АБО»), таблиця істинності якої виглядає наступним чином:

2. Підготувати дані та виконати навчання мережі, використовуючи одну з модифікацій методу зворотного поширення помилки.

3. Протестувати навчену нейронну мережу.

4. Вибрати архітектуру та сформувати нейронну мережу прямого поширення сигналів для вирішення задачі розпізнавання символів латинського алфавіту. При подачі на вхід бінарного вектора, що представляє букву, мережа повинна формувати на виході вектор, в якому елемент, що відповідає порядковому номеру букви в алфавіті, дорівнює одиниці, а інші елементи дорівнюють нулю.

5. Підготувати дані та виконати навчання нейронної мережі.

6. Дослідити результати роботи навченої мережі в умовах дії шуму. Шум моделюється за допомогою функції **randn** у вигляді вектора-стовпця з 35 елементів, значення яких є випадковими величинами з інтервалу [–1 1] із середнім значенням 0 та стандартним відхиленням, яке є меншим або дорівнює 0,5. Вектор шуму додається до відповідного вектора-стовпця масиву **alphabet**.

7. Побудувати графік, який відображає залежність величини помилки на виході мережі від рівня шуму. Помилка дорівнює евклідовій нормі різниці вектора виходу мережі та відповідного цільового вектора. Для побудови графіка необхідно:

1) вектори, що моделюють шум, формуються із середнім значенням 0 тастандартним відхиленням від 0 до 0,5 з кроком 0,05;

2) для кожного рівня шуму формується 10 зашумлених вхідних векторів для кожного символу алфавіту;

3) моделюється вихід мережі у відповідь на кожен зашумлений вхідний вектор;

4) обчислюється середня сумарна величина помилки на виході мережі для кожного рівня шуму від 0 до 0,5, яка записується в масив;

5) отримані масиви значень рівня шуму та величини помилки мережі використовуються для побудови графіка.

Реалізація логічної операції XOR

1. Архітектура: мережа прямого поширення з 2 входами, 1 прихованим шаром (3 нейрони) та 1 виходом.

Код у MATLAB:

```
P = [0 1 0 1; 0 0 1 1];   % вхідні пари
T = [0 1 1 0];            % цільові значення XOR
net = feedforwardnet(3, 'trainlm');  % 3 нейрони, метод trainlm
net.trainParam.epochs = 200;
net.trainParam.goal = 1e-3;
net = train(net, P, T);   % навчання
Y = net(P)                % тестування
```

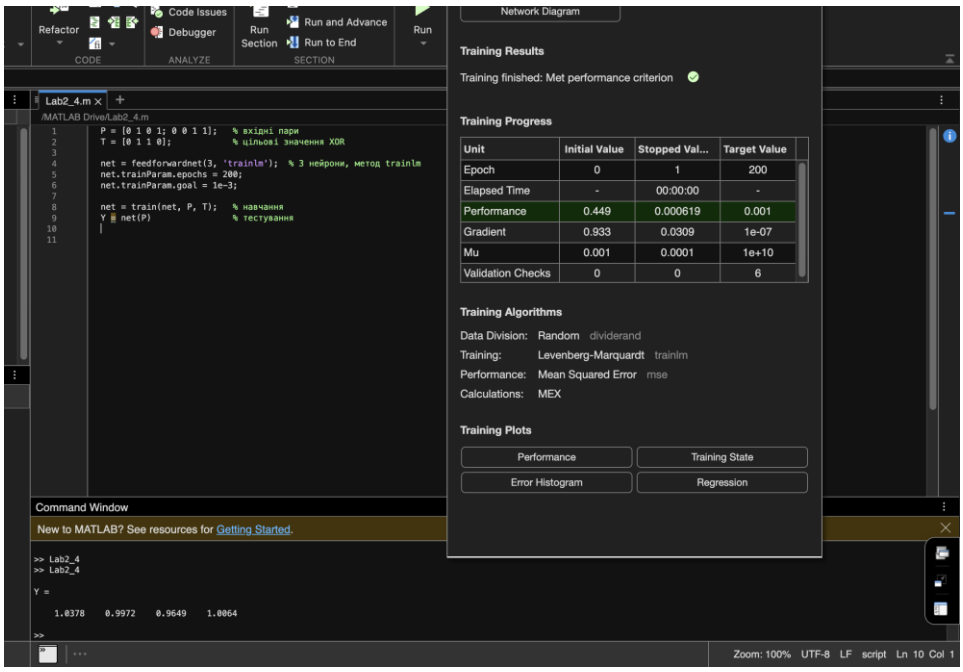

Рисунок 11

Розпізнавання символів латинського алфавіту

1. Архітектура: вхід — 35 бітів (зображення букви), вихід — 26 біт (вектор one-hot для кожної букви).

Код у MATLAB:

```
% Створення псевдобукв (35 ознак для кожної з 26 літер)
inputs = randi([0 1], 35, 26);  % випадкові 0/1 як вхід
% Цільові вектори — one-hot для 26 літер
targets = eye(26);

% Створення та навчання мережі
net = feedforwardnet(50, 'trainlm');
net.trainParam.epochs = 500;
net = train(net, inputs, targets);

% Тестування без шуму
Y = net(inputs);

% Аналіз стійкості до шуму
noise_levels = 0:0.05:0.5;
errors = zeros(size(noise_levels));

for i = 1:length(noise_levels)
    noise_std = noise_levels(i);
    total_error = 0;

    for letter = 1:26
        for j = 1:10
            noise = noise_std * randn(35, 1);
```

```
            noisy_input = inputs(:, letter) + noise;
            output = net(noisy_input);
            target = targets(:, letter);
            total_error = total_error + norm(output - target);
        end
    end

    errors(i) = total_error / (26*10);  % середня помилка
end

plot(noise_levels, errors, '-o');
xlabel('Рівень шуму');
ylabel('Середня помилка');
title('Помилка при розпізнаванні символів (з випадковими векторами)');
```

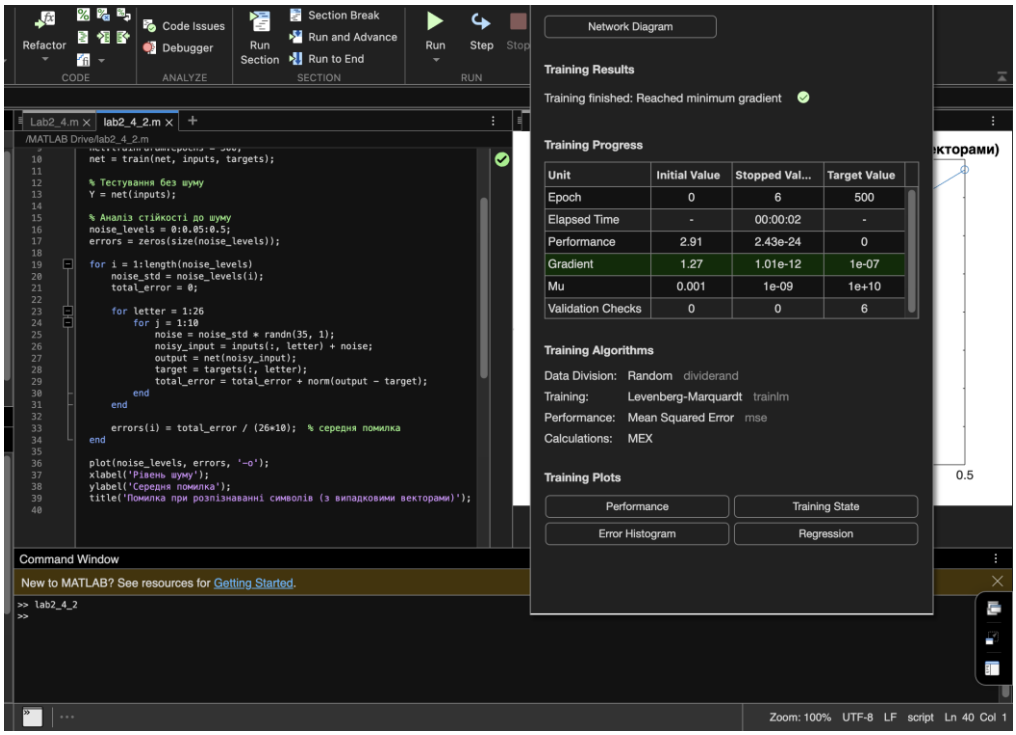

Рисунок 12

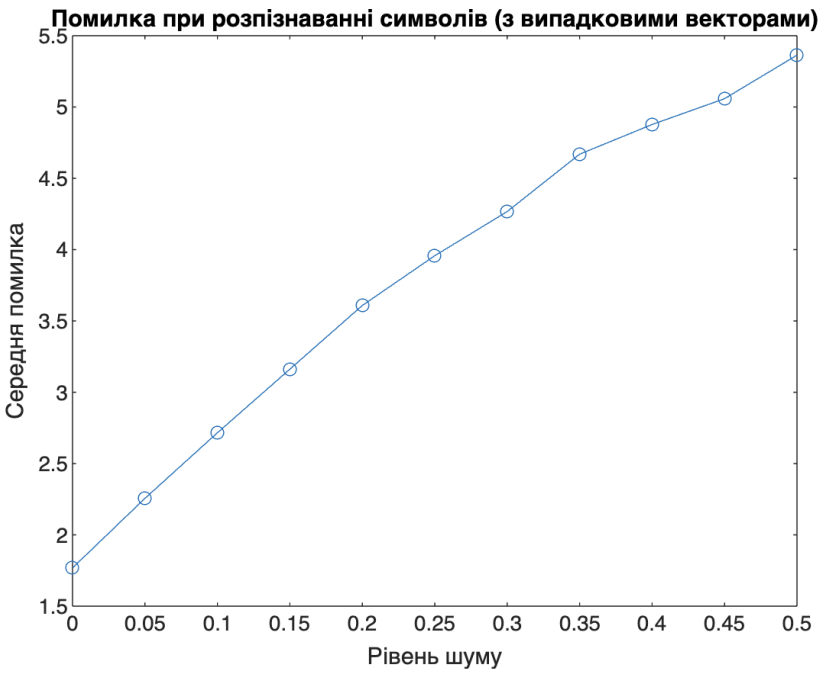

Рисунок 13

Результати показали, що при збільшенні рівня шуму точність мережі зменшується. Графік демонструє залежність евклідової помилки розпізнавання від рівня зашумлення вхідних образів.

Висновок

У процесі виконання лабораторної роботи було детально досліджено принципи функціонування адаптивних нейронних мереж та мереж прямого поширення сигналів. В результаті трьох експериментів було реалізовано моделі апроксимації функцій із використанням лінії затримки, а також досліджено вплив параметрів мережі на точність роботи, що є критично важливим при прогнозуванні часових рядів.

У другій частині лабораторної було створено мережу для реалізації логічної операції XOR, що є класичною нелінійною задачею класифікації, а також побудовано систему розпізнавання латинських символів, навчання якої виконувалося за допомогою зворотного поширення помилки. Експериментальне дослідження стійкості до шуму дозволило оцінити надійність моделі в реальних умовах.

Загалом лабораторна робота продемонструвала важливість правильної архітектури мережі, вибору кількості затримок та методу навчання для досягнення бажаної точності. Одержані результати можуть бути використані як база для побудови більш складних моделей у прикладних задачах розпізнавання, прогнозування та керування.