


НЕЙРОННІ МЕРЕЖІ В СИСТЕМАХ АВТОМАТИЗАЦІЇ

Методичні вказівки до виконання практичних робіт з
дисципліни

«Нейронні мережі в системах автоматизації»

(для здобувачів вищої освіти спеціальності
174 «Автоматизація, комп'ютерно-інтегровані технології та
робототехніка»
усіх форм навчання другого (магістерського)
рівня вищої освіти)

*Рекомендовано Науково-методичною радою
ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«МЕТІНВЕСТ ПОЛІТЕХНІКА»
(протокол № 6 від «24» травня 2024 р.)
Обов'язково до розміщення в репозиторії*



Нейронні мережі в системах автоматизації: методичні вказівки до виконання практичних робіт з дисципліни «Нейронні мережі в системах автоматизації» (для здобувачів вищої освіти спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» усіх форм навчання другого (магістерського) рівня вищої освіти). / Уклад. О.В. Разживін. Запоріжжя: ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2024. 139 с.

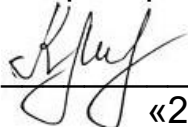
Практичні роботи присвячено роботі нейронних мереж в системах автоматизації які працюють у реальному часі з метою вироблення керуючих рішень у темпі надходження нової інформації, тому обчислювальні ресурси традиційних комп'ютерів швидко виявляються вичерпаними, якщо реалізується досить складна штучних нейронних мереж. Така ситуація характерна для систем аналізу та стиснення зображень, при управлінні технологічними процесами та рухомими об'єктами. У зв'язку з цим велике практичне значення набувають питання використання спеціалізованих обчислювальних засобів, що допускають паралельну обробку інформації.

Рекомендовано для здобувачів вищої освіти спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» усіх форм навчання другого (магістерського) рівня вищої освіти).

Самостійне електронне текстове мережеве видання

Затверджено на засіданні кафедри автоматизації, електро- та робототехнічних систем
Протокол № 8 від «30» квітня 2024 р.

Узгоджено:
Секретар Редакційної ради


Малій Х. В.
«21» травня 2024 р.

© ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«МЕТІНВЕСТ ПОЛІТЕХНІКА», 2024



ЗМІСТ

	Стор.
1 СТВОРЕННЯ І НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ З ДОПОМОГИ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА GUI	4
2 СТВОРЕННЯ, АДАПТАЦІЯ І НАВЧАННЯ ЛІНІЙНОЇ НЕЙРОННОЇ МЕРЕЖІ У КОМАНДНОМУ ВІКНІ MATLAB	19
3 РОЗРОБКА РАДІАЛЬНОЇ БАЗИСНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ АПРОКСИМАЦІЇ ФУНКЦІЙ	28
4 РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ СТАЦІОНАРНИХ СИГНАЛІВ	41
5 РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ СТАЦІОНАРНОГО ФІЛЬТРУ	47
6 СИНТЕЗ НЕЛІНІЙНОГО НЕЧІТКОГО РЕГУЛЯТОРА ПД-ТИПУ	52
7 СИНТЕЗ НЕЧІТКОГО РЕГУЛЯТОРА ПІ-ТИПУ У КОВЗАЮЧОМУ РЕЖИМІ	67
8 СИНТЕЗ НЕЧІТКОГО СУПЕРВІЗОРА	79
9 СУПЕРВІЗОРНЕ УПРАВЛІННЯ З ІНКРЕМЕНТАЛЬНОЮ ЗМІНОЮ КОЕФІЦІЄНТІВ РЕГУЛЯТОРА	92
10 СУПЕРВІЗОРНЕ УПРАВЛІННЯ З КОРЕКЦІЄЮ КОЕФІЦІЄНТІВ ПІД-РЕГУЛЯТОРА НА ПІДСТАВІ ІНФОРМАЦІЇ ПРО ПОМИЛКУ ТА ЇЇ ПОХІДНУ	108
ПЕРЕЛІК ЛІТЕРАТУРИ	123
ДОДАТОК А. Налаштування коефіцієнтів передаточної функції ПІД-регулятора	124
ДОДАТОК Б. Демонстраційні приклади ППП NNT	130
ДОДАТОК В. Функції пакета «Neural Network Toolbox» системи MATLAB для роботи з нейронними мережами	131

1 СТВОРЕННЯ І НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ З ДОПОМОГОЮ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА GUI

Мета роботи: освоїти методику розробки та навчання нейронної мережі із застосуванням графічного інтерфейсу користувача (GUI) системи MATLAB.

1.1 GUI- інтерфейс для пакету прикладних програм nnt

Графічний інтерфейс користувача (**Graphical User Interface**) дозволяє, не звертаючись до командного вікна системи MATLAB, виконати створення, навчання та моделювання, а також експорт та імпорт нейронних мереж.

Виклик GUI-інтерфейсу пакета прикладних програм **Neural Network Toolbox** можливий чи командою **nntool** з командного рядка або з вікна запуску додатків **Launch Pad** за допомогою опції **NNTool**. Після виклику з'являється вікно (див. рис. 1.1) **Network/Data Manager** (Управління мережею/даними).

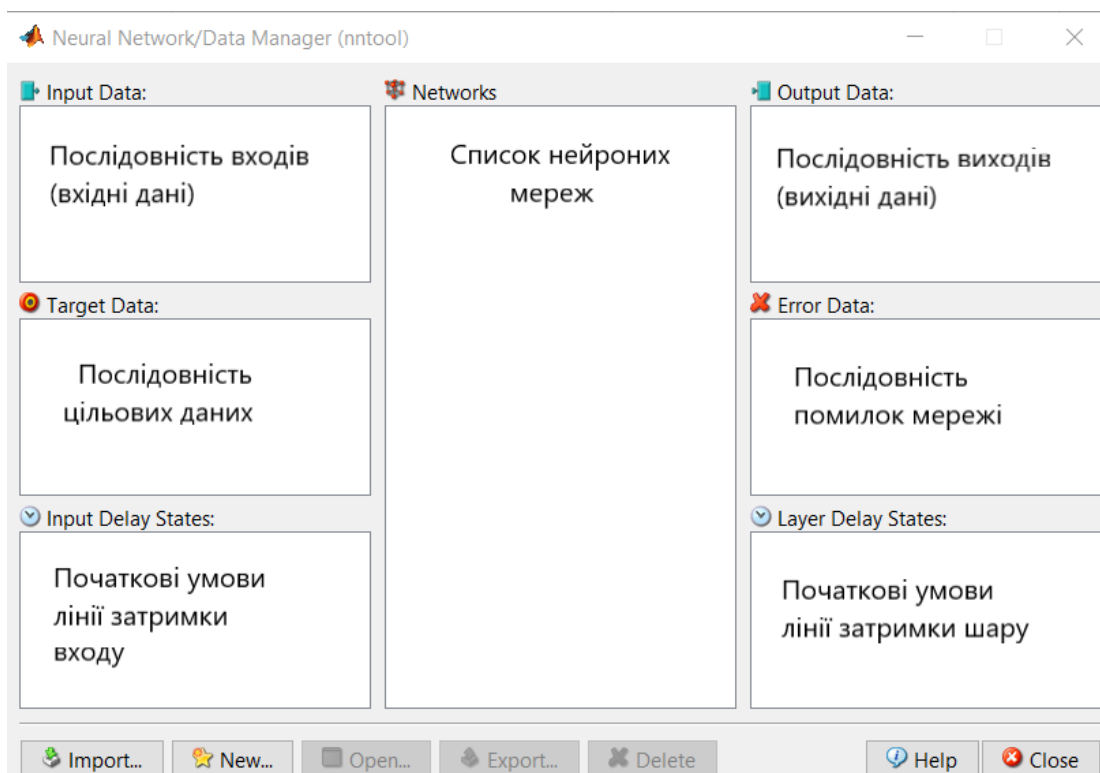


Рисунок 1.1 – Вікно керування мережею та даними

- Щоб створити нейронну мережу, потрібно виконати такі операції:
- кнопкою **Import** викликати вікно форматування даних **Import from Network/Data Manager** (рис. 1.2) та сформуванати послідовність входів (Input Data) та цілей (Target Data);
 - зачинити вікно кнопкою **Close**

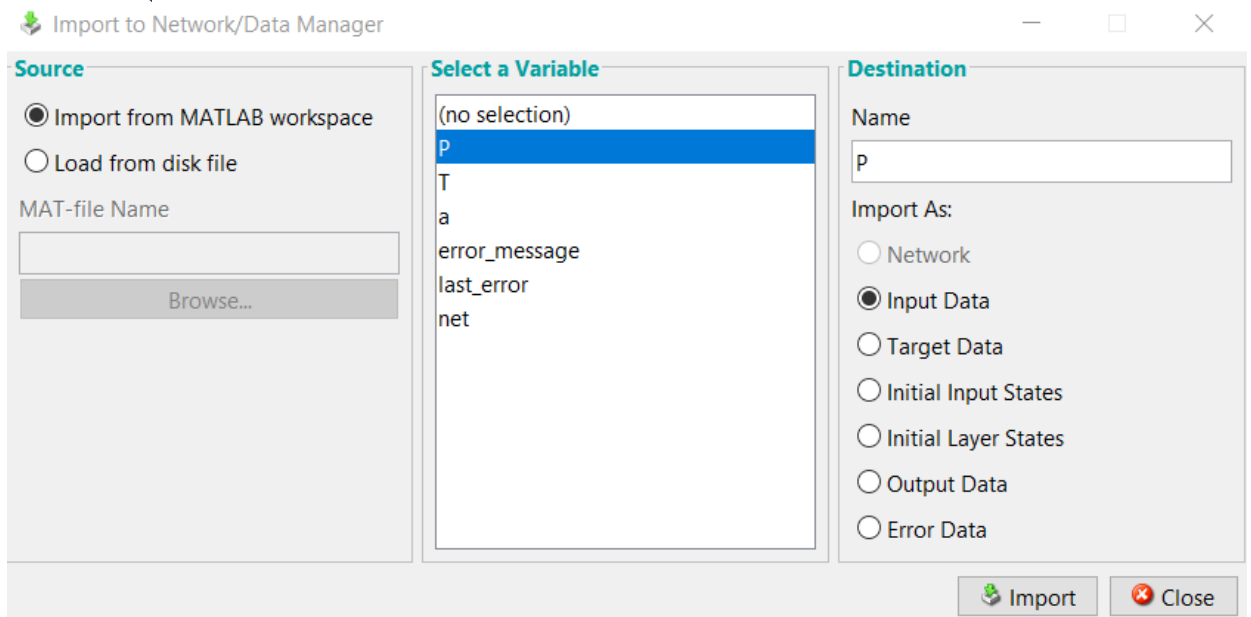


Рисунок 1.2 – Вікно вводу даних

В вікні вводу даних **Import from Network/Data Manager** можна вказати один із шести типів даних:

- **Inputs Data** (входи) – послідовність значень входів;
- **Targets Data** (цілі) – послідовність значень цілей;
- **Initial Input States** (початкові стани входів) – початкові умови лінії затримки на вході;
- **Initial Layer States** (початкові стани шару) – початкові умови лінії затримки у шарі;
- **Outputs Data** (виходи) – послідовність значень виходу мережі;
- **Errors Data** (помилка) - різниця значень цілей та виходів.

Зазвичай користувач задає лише послідовності входу та мети.

- в вікні **Creating Network or Data** (рис. 1.3), яке викликається кнопкою **New Network**, з вікна **Network/Data Manager** (рис 1.1) створити нову мережу;

- Вікно **Create New Network**, показане рисунку 1.3, включає поля для завдання параметрів створюваної мережі. Кількість та назви полів змінюються залежно від типу мережі.

- Поле **Network Type** призначений для вибору типу мережі. Типи мереж, доступних до роботи з інтерфейсом NNTool, наведено у таблиці 1.1.

Слід врахувати, що цей інтерфейс дозволяє створювати нейронні мережі лише з одним або двома шарами, кількість шарів **Layer** задається у полі **Number of Layer** . Параметри, що навчаються, є ваговими матрицями входу. **IW** (Input Weight) та вихода **LW** (Layer Weight), а також зміщення **b**. Мережі з двома шарами зазвичай мають послідовну структуру, коли вихід першого шару є входом другого шару.

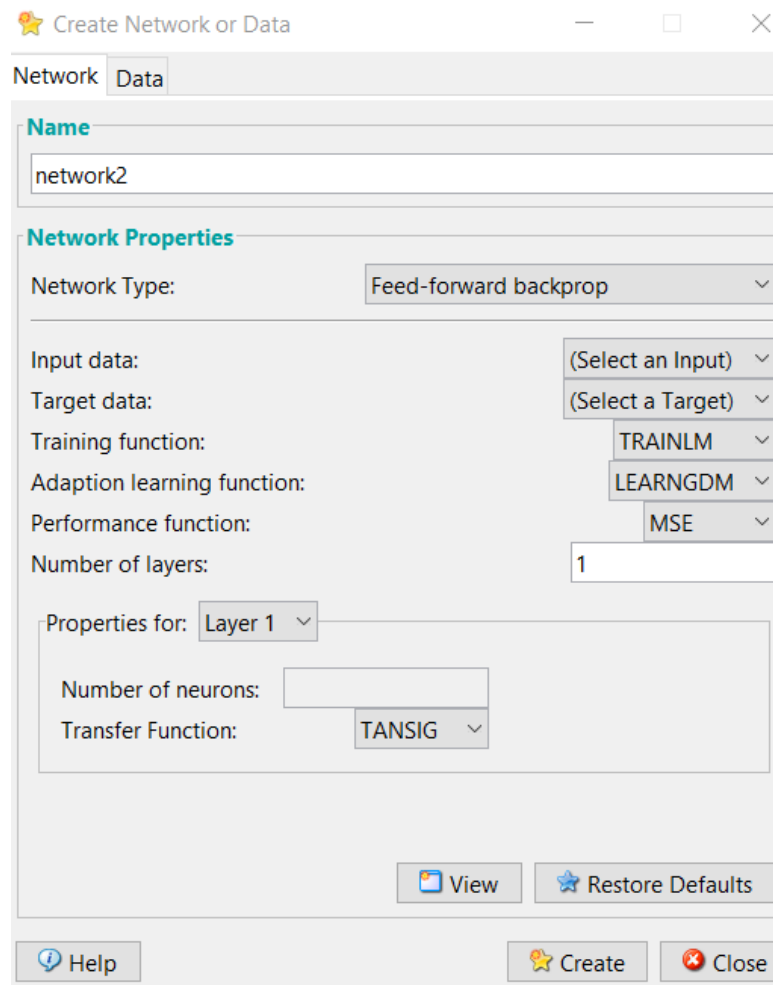


Рисунок 1.3 – Вікно створення мережі **Creating Network or Data**

Таблиця 1.1 – Список типів мереж, доступних до роботи з інтерфейсом NNTool

№ п/п	Тип мережі	Назва мережі	Кількість шарів	Параметр навчання
1	Competitive	Конкуруюча мережа	1	$IW[1,1], b[1]$
2	Cascade forward backprop	Каскадна мережа	2	Перегляд не забезпечується
3	Elman backprop	Мережа Елмана	2	Перегляд не забезпечується
4	Feed-forward backprop	Мережа з прямим розповсюдженням сигналу	2	$IW[1,1], b[1]$ $LW[2,1], b[2]$
5	Time delay backprop	Мережа запізненням	3	2 $IW[1,1], b[1]$ $LW[2,1], b[2]$
6	Generalized regression	Узагальнена регресійна мережа	2	$IW[1,1], b[1]$ $LW[2,1]$

Продовження табл 1.1

№ п/п	Тип мережі	Назва мережі	Кількість шарів	Параметр навчання
7	Hopfield	Мережа Хопфілда	1	Перегляд не забезпечується
8	Linear layer (design)	Лінійний шар (створення)	1	$IW[1,1]$, $b[1]$
9	Linear layer (train)	Лінійний шар (навчання)	1	$IW[1,1]$, $b[1]$
10	LVQ	Мережа класифікації векторів	2	$IW[1,1]$, $LW[2,1]$
11	Perceptron	Персептрон	1	$IW[1,1]$, $b[1]$
12	Probabalistic	Ймовірнісна мережа	2	$IW[1,1]$, $b[1]$ $LW[2,1]$
13	Radial basis (exact fit)	Радіальна базова мережа з нульовою помилкою	2	$IW[1,1]$, $b[1]$ $LW[2,1]$
14	Radial basis (fewer neurons)	Радіальна базова мережа з мінімальним числом нейронів	2	$IW[1,1]$, $b[1]$ $LW[2,1]$, $b[2]$
15	Self-organizing map	Самоорганізована карта Кохонена	1	$IW[1,1]$

Поле **Input Data** (діапазони входу) дозволяє задати допустимі межі входів (масиви вхідних даних).

Поле **Targets Data** (діапазони цілей) дозволяє задати допустимі межі цілій навчання (масиви даних цілей).

Поле **Training function** (функція навчання) дозволяє задати алгоритм навчання мережі.

Adaption learning function – виводить список функцій налаштувань режиму адаптації;

Performance function – виводить список функцій оцінки якості навчання;

Number of layers – завдання кількості шарів нейронної мережі;

Інші поля призначені для завдання наступних функцій:

Properties for – служить для завдання властивостей шару;

Number of neurons – задає кількість нейронів у шарі;

Transfer function – задає функцію активації шару.

Кнопкою **View** можливо викликати структуру мережі (рис. 1.4).

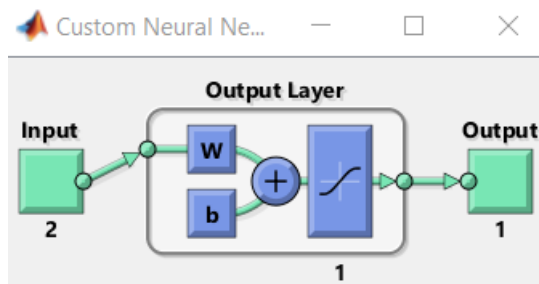


Рисунок 1.4 – Структура нейронної мережі

В вікні **Creating Network or Data** натисканням кнопки **Create** виконується ініціалізація мережі.

Надалі у діалоговому вікні **Network/Data Manager** подвійним кліком мишки по назві нейронної мережі викликається діалогова панель **Network: network1** (див. рис. 1.5).

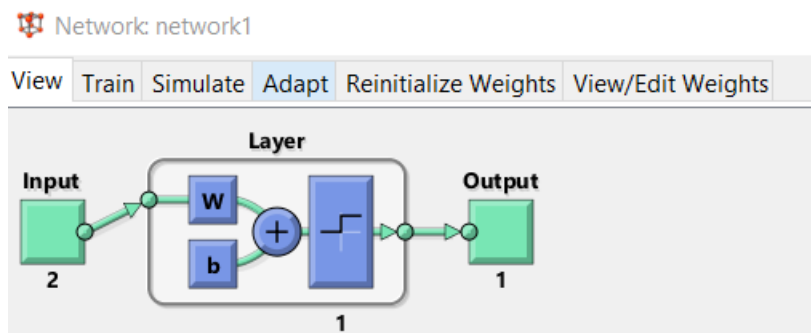


Рисунок 1.5 - Діалогова панель **Network: network1**

В полі **Reinitialize Weights** діалогової панелі **Network: network1** необхідно встановити діапазони входів мережі (наприклад для двох входів мережі [-3 2; -3 2]), потім натиснути кнопку **Set Input Ranges**. Далі, натиснувши кнопку **Initialize Weights**, необхідно так само встановити ваги входів **p1** та **p2**.

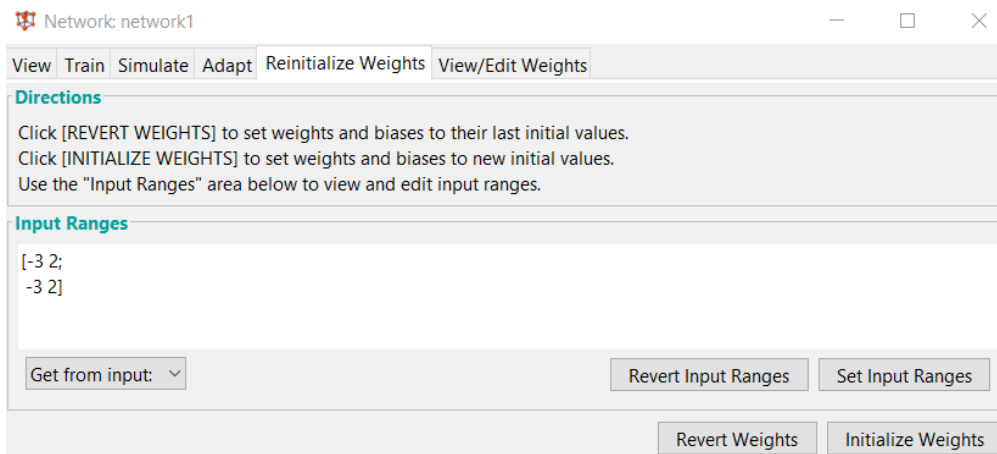


Рисунок 1.6 – Вікно ініціалізації мережі **network1**

Виконаємо адаптацію та навчання мережі, викликаючи вікна діалогу кнопками **Adapt** (див. рис. 1.7) або **Train** (див. рис. 1.8). У вікнах вказуються імена входу та цілі, а при виборі команди **Adaptation Parameters** в полі **epochs** задається кількість циклів адаптації.

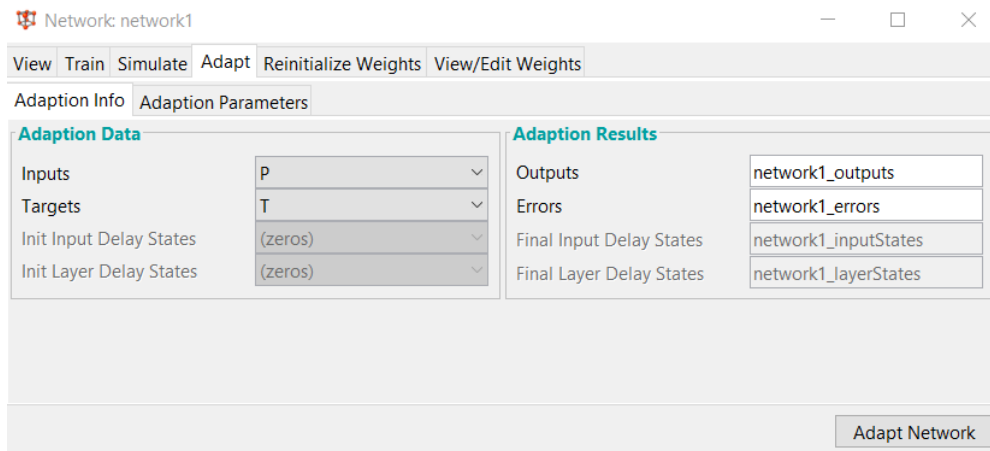


Рисунок 1.7 – Вікно налаштування параметрів адаптації мережі

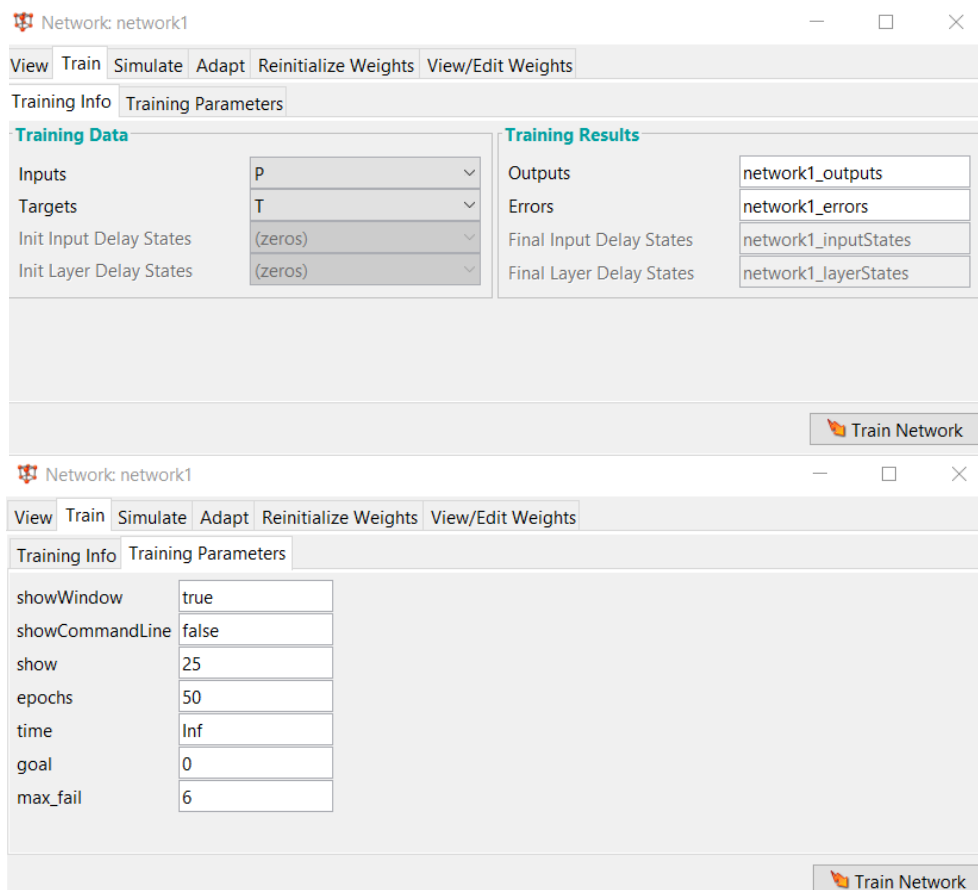


Рисунок 1.8 – Вікно завдання параметрів навчання мережі

Результати навчання можна побачити, обравши команду **View/Edit Weights** , по якій відкривається вікно з полем **Select the**

weight or bias to view для вибору ваги входів, що встановилися **IW** та зміщення **b** (див. рис. 1.9).



Рисунок 1.9 – Результат навчання мережі $IW\{1,1\}=[-1.5 -1.5]$

Раскрыв закладку **Select the weight or bias to view**, можно увидеть значение смещения: **b=[1]**.

Таким чином, лінія перемикування, що розділяє класи об'єктів у площині (p1; p2), записується так:

$$L: -1.5p_1 - 1.5p_2 + 1 = 0.$$

Перейшовши у вікно **Network/Data Manager**, можна переглянути значення сигналів на виході та помилку мережі. Для цього в полі **Outputs Data:** виділимо сигнал на виході мережі *network1_outputs* та виберемо команду **View** (подвійний клік мишки), по якій відкриється вікно **Data: network1_outputs** (див. рис. 1.10). Аналогічну процедуру виконаємо і для виведення помилки *network1_errors*.

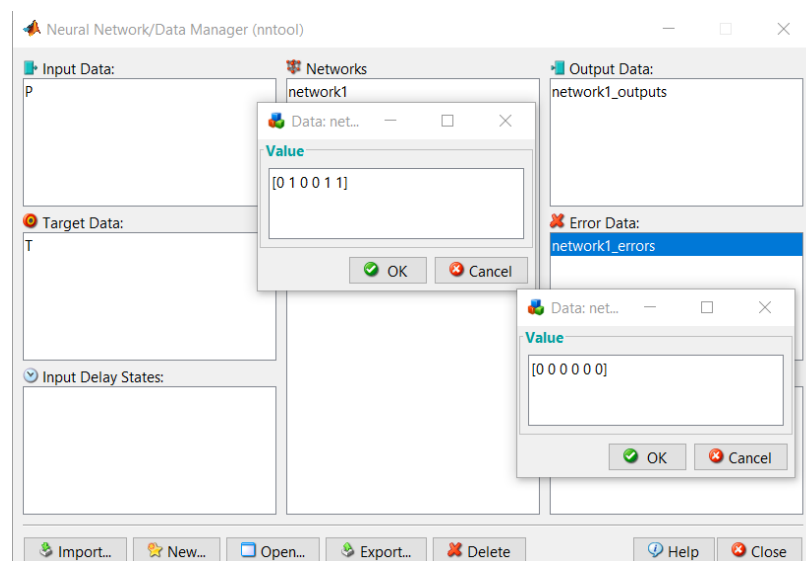


Рисунок 1.10 – Вікно для перегляду вихідних сигналів мережі

Кнопками **Import**, **Export** вікна управління мережею **Network/Data Manager** можна відкрити вікна **Import or Load to Network/Data Manager** или **Export or Save from Network/Data Manager**. Ці вікна дозволяють завантажити дані з робочої області MATLAB до робочої області GUI-інтерфейсу, або, навпаки, передати дані до робочої області MATLAB.

1.2 Методика виконання роботи

Нехай, наприклад, необхідно створити нейронну мережу у вигляді перцептрона, для поділу векторів входу на два класи, позначених як 0 та 1.

Процес створення мережі включатиме наступні операції.

1. Задаємо навчальну послідовність у вигляді двох масивів осередків: масиву входів $P = \{[2; 2] [1; 2] [-2; 2] [-1; 1] [1; -2]\}$ та масиву цілей $T = \{0 \ 0 \ 1 \ 1 \ 1\}$. Масив цілей визначає приналежність кожного вектору входу до певного класу. Виконаємо цю операцію у робочій області системи MATLAB.

```
>> nntool
>> P=[2;2] [1;2] [-2;2] [-1;1] [1;-2]
P =
Columns 1 through 4
[2x1 double] [2x1 double] [2x1 double] [2x1 double]
Column 5
[2x1 double]
>> T={0 0 1 1 1}
T =
[0] [0] [1] [1] [1]
```

2. Командою **nntool** викликаємо вікно (див. рис. 1.11) **Network/Data Manager** (Управління мережею/даними).

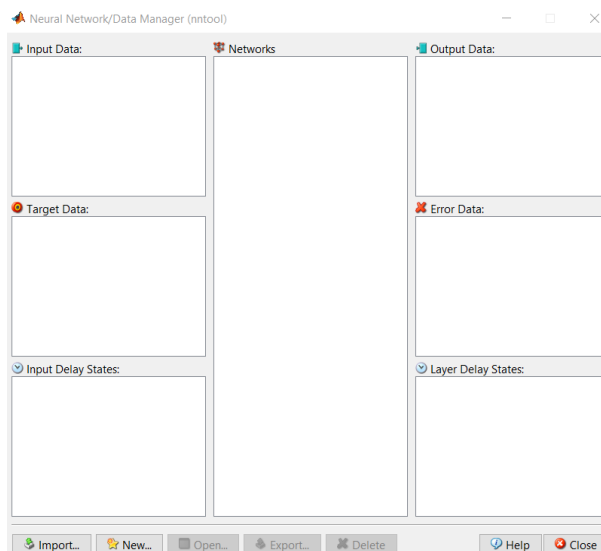


Рисунок 1.11 – Вікно керування мережею та даними

Кнопкою **Import** викликаємо вікно форматування даних **Import from Network/Data Manager** (див. рис. 1.12) в якому завантажуюємо послідовність входів **P** (Input Data) та цілей **T** (Target Data);

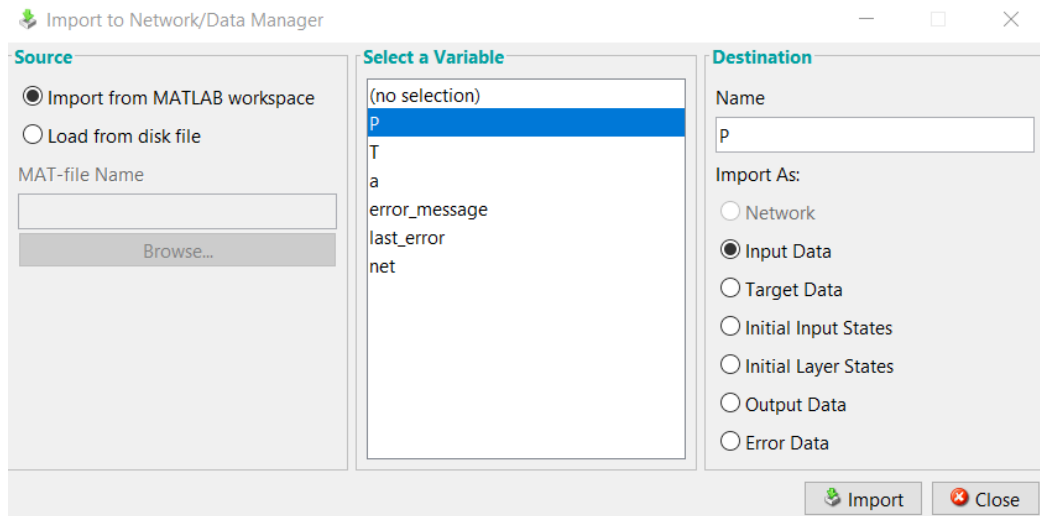


Рисунок 1.12 – Вікно вводу даних

В вікні **Creating Network or Data** (див. рис. 1.13), яке викликається кнопкою **New Network**, з вікна **Network/Data Manager** (див. рис. 1.11) створюємо нову мережу;

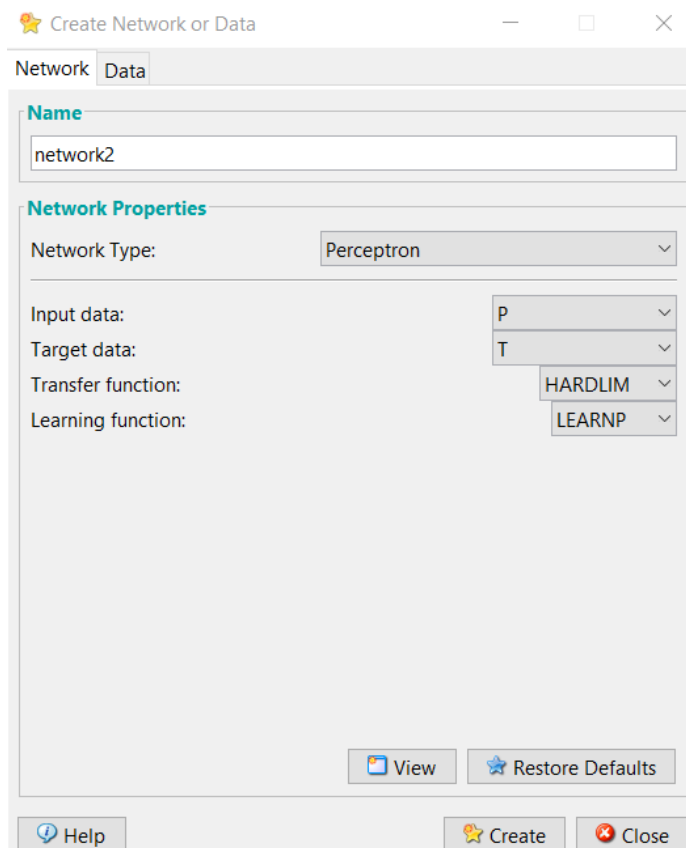


Рисунок 1.13 – Вікно створення мережі **Creating Network or Data**

Задаємо перцептрон з одним нейроном на два входи з діапазоном $[-2;2]$, застосовуємо функцію активації HARDLIM та правило налаштування LEARNP. Натисканням кнопки **Create** виконується ініціалізація мережі.

Надалі у діалоговому вікні **Network/Data Manager** подвійним кліком мишки по назві нейронної мережі викликаємо діалогову панель **Network: network1** (див. рис. 1.14).

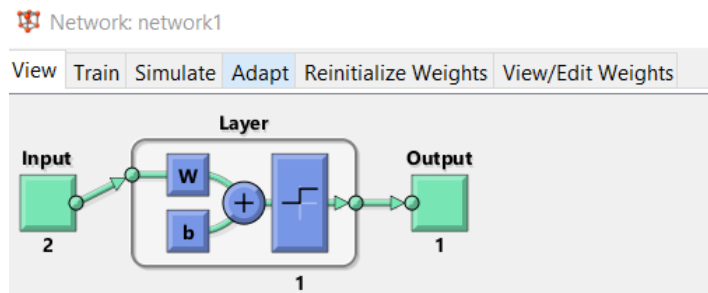


Рисунок 1.14 - Діалогова панель **Network: network1**

В полі **Reinitialize Weights** (див. рис 1.15) діалогової панелі **Network: network1** встановлюємо діапазони входів мережі (наприклад для двох входів мережі $[-2\ 2; -2\ 2]$), потім натиснути кнопку **Set Input Ranges**. Далі, натиснувши кнопку **Initialize Weights**, необхідно так само встановити ваги входів **p1** та **p2**.

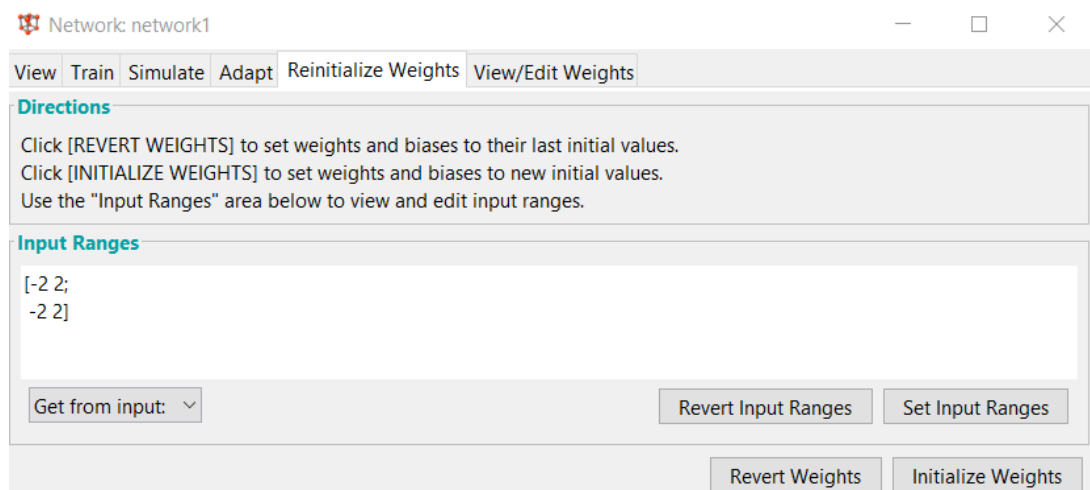


Рисунок 1.15 – Вікно ініціалізації мережі **network1**

Виконаємо адаптацію та навчання мережі, викликаючи вікна діалогу кнопками **Adapt** (див. рис. 1.16) або **Train** (див. рис. 1.17). У вікнах вказуються імена входу та цілі, а при виборі команди **Adaptation Parameters** в полі **epochs** задається кількість циклів адаптації. Для виконання даного завдання достатньо 5...7 епох навчання.

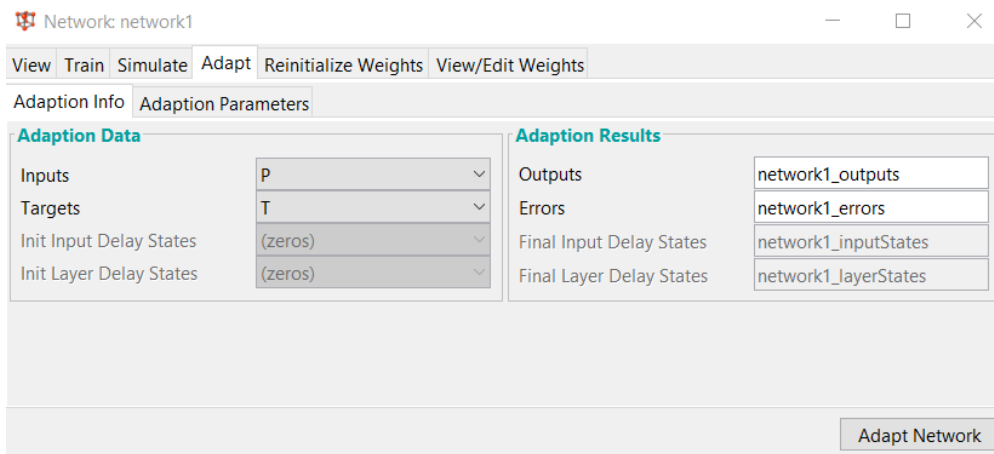


Рисунок 1.16 – Вікно налаштування параметрів адаптації мережі

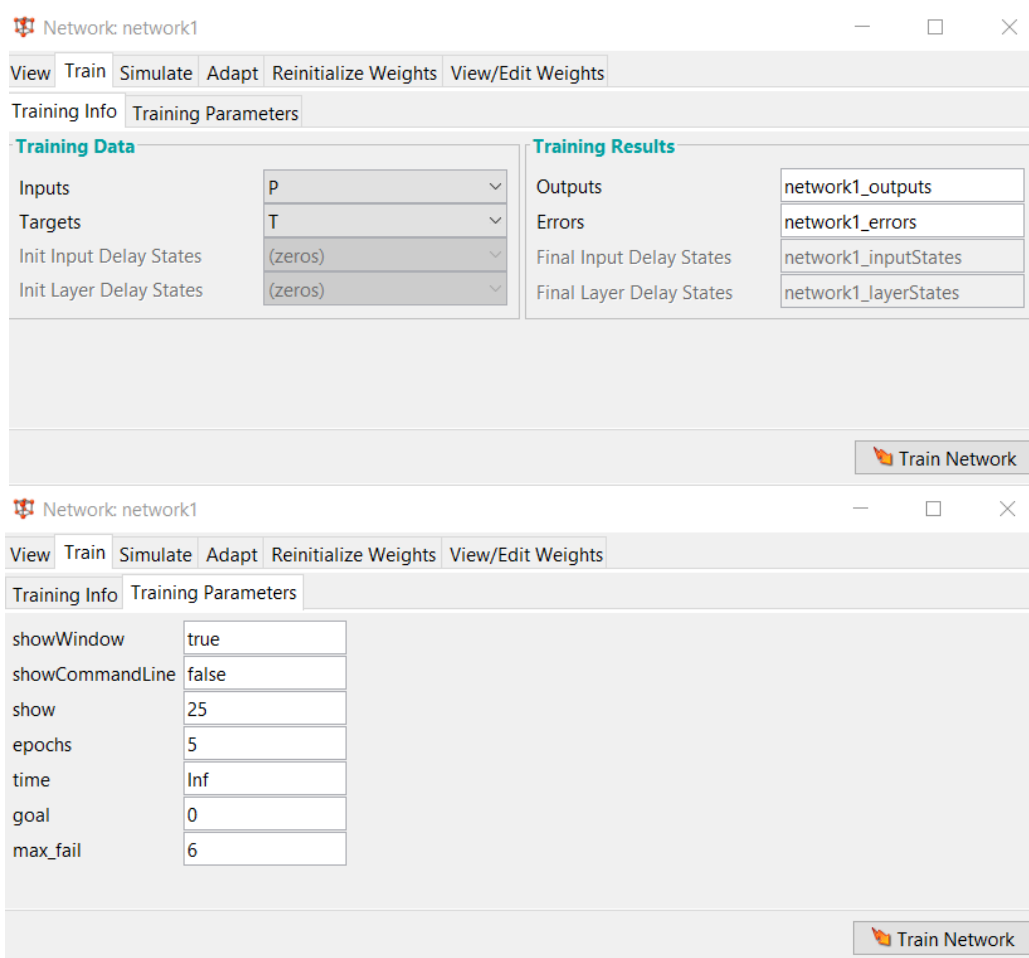


Рисунок 1.17 – Вікно завдання параметрів навчання мережі

Визначимо результати навчання можна побачити, обравши команду **View/Edit Weights**, по якій відкривається вікно з полем **Select the weight or bias to view** для вибору ваги входів, що встановилися **IW** та зміщення **b** (див. рис. 1.18)

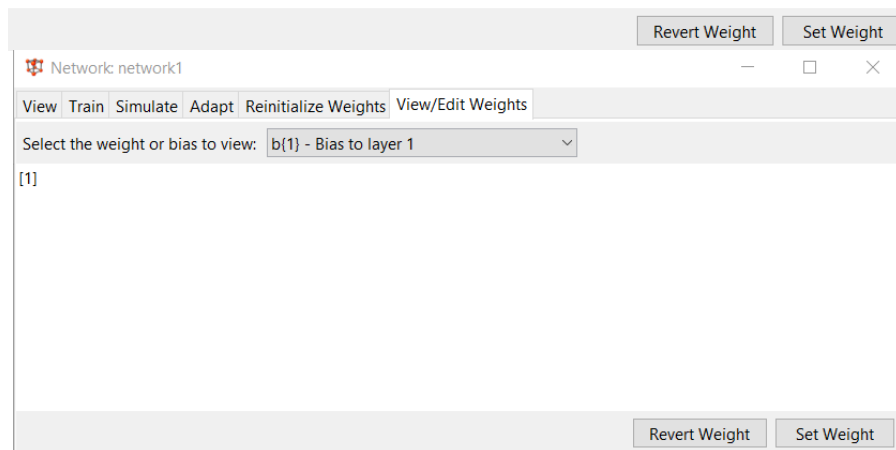
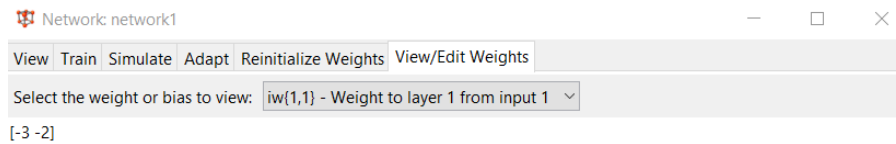


Рисунок 1.18 – Результат навчання мережі

Таким чином, лінія перемикання, що розділяє класи об'єктів у площині (p_1 ; p_2), записується так:

$$L: -3p_1 - 2p_2 + 1 = 0.$$

У вікні **Network/Data Manager**, переглянемо значення сигналів на виході та помилку мережі (див. рис. 1.19).

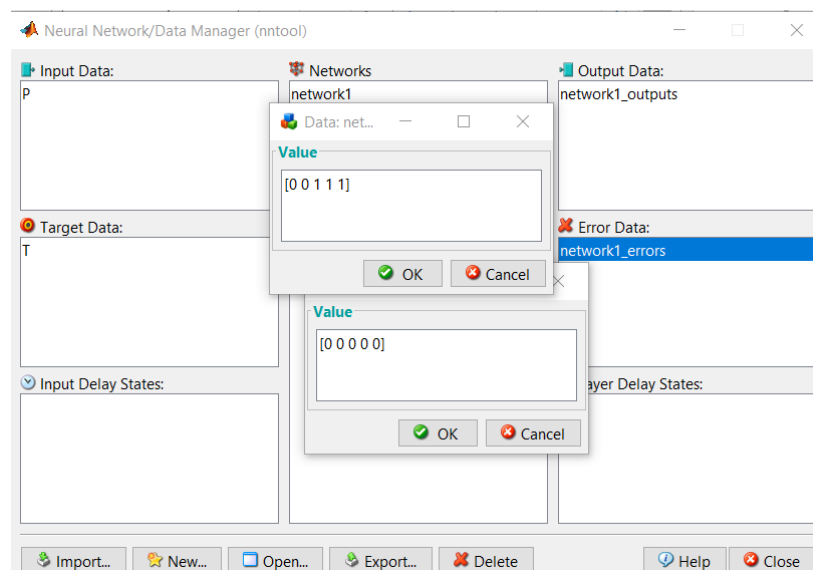


Рисунок 1.19 – Вікно для перегляду вихідних сигналів мережі

```

3. Для виведення графіка лінії перемикання потрібно у вікні команд
MATLAB ввести вихідні дані в іншому форматі. Після цього налаштувати
та адаптувати мережу. Програма матиме такий вигляд:>> P=[2 1 -2 -1
1;2 2 2 1 -2];
P=[2 1 -2 -1 1;2 2 2 1 -2];
T=[0 0 1 1 1];
net=newp([-2 2;-2 2],1);
net.adaptParam.passes=3;
net=train(net,P,T);
hold on;
plotpv(P,T);
plotpc(net.IW{1},net.b{1});
hold off
a=sim(net,P)

```

Результати виконання програми моделювання нейронної мережі

```

>> Lab1
a =
    0    0    1    1    1

```

В результаті програмного моделювання побудовано графік, на якому в координатах $p(1)$ – $p(2)$ нанесена лінія, яка розділяє класи вхідних векторів (див. рис. 1.20).

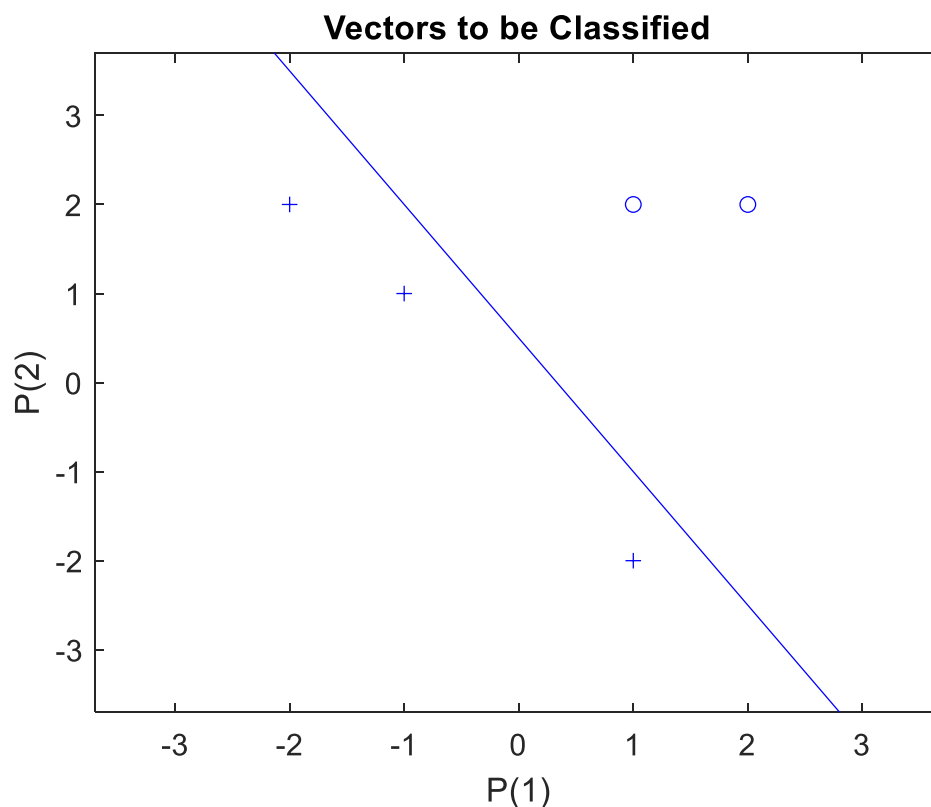


Рисунок 1.20 – Графік поділу класів об'єктів

1.3 Зміст звіту

Для виконання роботи студенти отримують індивідуальні завдання, варіанти яких наведені у таблиці 1.2.

Звіт по роботі повинен містити завдання, структуру мережі, параметри налаштування та результати навчання мережі.

При захисті звіту студент повинен відповісти на питання щодо методики роботи з графічним інтерфейсом користувача GUI, а також особливостей налаштування нейронних мереж.

Таблиця 1.2 - Варіанти індивідуальних завдань

№ вар.	Вхідна послідовність	Діапазон	Послідовність цілей
1	$p_1 = [-2 \ -3 \ +5 \ +1 \ -6 \ +3 \ -5]$ $p_2 = [0 \ -4 \ +2 \ +3 \ +1 \ +6 \ -3]$	-6...+6	[0 0 1 1 0 1 0]
2	$p_1 = [+2 \ -3 \ +5 \ +1 \ -6 \ +3 \ -5]$ $p_2 = [0 \ -4 \ +2 \ +3 \ +1 \ -6 \ -3]$	-5...+5	[0 1 0 0 0 1 1]
3	$p_1 = [-1 \ +5 \ +4 \ -5 \ +2 \ -3]$ $p_2 = [-3.5 \ +1.5 \ -5 \ +2.5 \ +1 \ -2]$	-5...+5	[0 1 0 1 1 0]
4	$p_1 = [-10 \ +8 \ +4 \ +12 \ -7 \ -5]$ $p_2 = [+2 \ +3.5 \ -2 \ -10 \ +7 \ -6]$	-12...+12	[0 1 1 1 0 0]
5	$p_1 = [-1 \ +2 \ +1.5 \ -0.5 \ +1.5 \ +2]$ $p_2 = [+1 \ +1.5 \ +0.5 \ +2 \ -1 \ +0.5]$	-2...+2	[0 1 0 0 1 1]
6	$p_1 = [+1 \ -2 \ +0.5 \ +1.5 \ 0 \ -2.5]$ $p_2 = [-3 \ -1 \ +1 \ -0.5 \ -2 \ +0.5]$	-3...3	[0 1 1 0 0 1]
7	$p_1 = [5 \ 4 \ 5 \ 3 \ 4 \ 2]$ $p_2 = [1 \ 3.5 \ 1 \ 2 \ 3 \ 0.5]$	0...5	[0 1 1 0 1 0]
8	$p_1 = [0.5 \ 1.5 \ 2 \ 1 \ 0 \ 1]$ $p_2 = [1 \ 0.5 \ 1 \ 2 \ 1.5 \ 2]$	0...2	[1 0 0 1 1 0]
9	$p_1 = [-0.5 \ -0.5 \ +0.3 \ -0.1 \ +0.2 \ -0.7]$ $p_2 = [-0.5 \ +0.5 \ -0.3 \ +1.0 \ +0.8]$	-1...+1	[1 1 0 0 0 1]
10	$p_1 = [+2 \ +1.5 \ +3 \ +0.5 \ -0.5 \ -1]$ $p_2 = [-1 \ +0.5 \ +2 \ +0.5 \ +2 \ +1]$	-1...+3	[0 1 1 0 1 0]
11	$p_1 = [+2 \ +4.5 \ +3 \ +2.5 \ -0.5 \ -1]$ $p_2 = [-3 \ +0.5 \ +2 \ +1.5 \ +2 \ -4]$	-4...+5	[0 1 1 0 0 0]
12	$p_1 = [+2 \ -3 \ +5 \ +1 \ -6 \ +3]$ $p_2 = [-1 \ +3.5 \ -4 \ +2 \ +3 \ -0.5]$	-6...+5	[0 1 0 1 1 0]
13	$p_1 = [+1 \ -2 \ +0.5 \ +1.5 \ -2.5 \ -0.5]$ $p_2 = [-0.5 \ +0.5 \ -0.3 \ +1.0 \ +0.8 \ -2]$	-3...+2	[1 0 1 1 0 0]
14	$p_1 = [-10 \ +8 \ +4 \ +2 \ -7 \ -5]$ $p_2 = [+2 \ +3.5 \ -2 \ -10 \ +7 \ -6]$	-10...+10	[0 1 1 1 0 0]

Продовження табл. 1.2

№ вар.	Вхідна послідовність	Діапазон	Послідовність цілей
15	$p_1 = [0.5 \ 1.5 \ 2 \ 1 \ 0 \ 1]$ $p_2 = [1 \ 0.5 \ 1 \ 2 \ 1.5 \ 2]$	0...2	[0 1 1 0 0 1]
16	$p_1 = [-2 \ -3 \ +5 \ +1 \ -6 \ +3 \ -5]$ $p_2 = [0 \ -4 \ +2 \ +3 \ +1 \ +6 \ -3]$	-6...+6	[1 1 0 0 1 0 1]
17	$p_1 = [+3 \ +4 \ +1 \ +4 \ -1 \ -0.5]$ $p_2 = [+3 \ +1 \ +0.5 \ -2 \ +2 \ -2]$	-3...+5	[1 0 1 0 1 0]
18	$p_1 = [+2 \ -3 \ +5 \ +1 \ -6 \ +3]$ $p_2 = [-1 \ +3.5 \ -4 \ +2 \ +3 \ -0.5]$	-6...+6	[1 0 1 0 0 1]
19	$p_1 = [0.5 \ 1.5 \ 2 \ 1 \ 0 \ 1]$ $p_2 = [1 \ 0.5 \ 1 \ 0.2 \ 1.5 \ 2]$	0...2	[0 1 0 0 0 1]
20	$p_1 = [+1 \ -2 \ +0.5 \ +1.5 \ -2.5 \ -0.5]$ $p_2 = [-0.5 \ +0.5 \ -0.3 \ +1.0 \ +0.8 \ -2]$	-3...+2	[0 1 0 0 1 1]

1.3 Контрольні питання

- 1 Призначення та функціональні можливості GUI-інтерфейсу.
- 2 Правила ініціалізації, адаптації та навчання мережі.
- 3 Оцінка якості роботи мережі.

2 СТВОРЕННЯ, АДАПТАЦІЯ ТА НАВЧАННЯ ЛІНІЙНОЇ НЕЙРОННОЇ МЕРЕЖІ У КОМАНДНОМУ ВІКНЕ MATLAB

Мета роботи: освоїти методику та придбати навички створення, адаптації та навчання лінійної нейронної мережі із застосування пакету прикладних програм (ППП) Neural Network Toolbox системи програмування MATLAB.

2.1 Методика створення, адаптації та навчання мережі в ППП NEURAL NETWORK TOOLBOX

Формування та ініціалізація мережі. Нейронна мережа створюється оператором **net**, а лінійний шар – оператором **newlin**. Наприклад, лінійна мережа з двоелементним вектором входу, значення якого знаходяться в діапазоні **[-1 1]**, одним шаром, без лінії затримки на вході (тип 0) та параметром швидкості налаштування 0.2 формується наступною командою:

```
>> net=newlin([-1 1;-1 1],1,0,0.2)
```

Така мережа функціонує відповідно до лінійної залежності виду:

$$a = \text{purelin}(Wp + b),$$

де a – вихід мережі, purelin – лінійна функція активації нейрона, W – вагова матриця входів, p – вектор входів, b – ваговий коефіцієнт зміщення нейрона.

Модель цієї нейронної мережі показано на рисунку 2.1.

Нехай, наприклад, вагова матриця має бути задана вектором $W = [1 \ 2]$, а ваговий коефіцієнт зміщення нейрона $b = 0$. Тоді ці параметри мережі в описі її структури будуть представлені таким чином:

```
net.IW{1,1}=[1,2];  
net.b{1}=0;
```

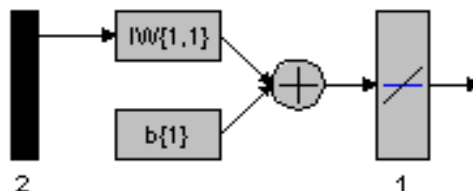


Рисунок 2.1 – Модель лінійної нейронної мережі з двоелементним вектором входу

При формуванні мережі оператором **newlin** ініціалізація мережі

проводиться за умовчанням, тому застосовувати метод `net=init(net)` не слід.

Адаптація (налаштування) нейронної мережі. Процедура адаптації здійснюється при введенні навчальної вибірки послідовним чи груповим способом. Для адаптації мережі необхідно встановити вектори входу P і цілі T , а також встановити початкові значення ваг IW та зміщення b .

Процес адаптації розглянемо з прикладу формування лінійної залежності виду:

$$t = 2p_1 + p_2.$$

При **послідовному способі** подання навчальної послідовності вектори задаються у вигляді масивів комірок формату `cell`:

```
>> net=newlin([-1 1;-1 1],1,0,0);
>> P=[-1;1] [-1/3;1/4] [1/2;0] [1/6;2/3];
>> T=[-1 -5/12 1 1];
>> P1=[P{:}], T1=[T{:}]
P1 =
-1.0000 -0.3333 0.5000 0.1667
 1.0000 0.2500 0 0.6667
T1 =
-1.0000 -0.4167 1.0000 1.0000
```

Спочатку поставимо мережу з нульовими значеннями початкових ваг і зміщень:


```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
```

Щоб забезпечити можливість вибирати довільні функції для налаштування ваги та зсувів, скористаємося М-функцією `adapt`. Для лінійних мереж, створюваних за допомогою методу `newlin` за умовчанням встановлюється функція `adaptwb` налаштування режиму та функція `learnwh` налаштування параметрів мережі за алгоритмом WH (правило Уїдроу-Хоффа).

Learnwh обчислює величину зміни ваги входу даного нейрона dw від вкладу нейрона pn' , помилки e та швидкості налаштування lr згідно Widrow-Hoff правила:

$$dw = lr * e * pn'.$$

Функції налаштування ваги та зсувів задаються у форматі `net.inputWeights{i,j}.learnFcn` та `net.biases{i,j}.learnFcn`, відповідно.



Виконаємо перший цикл адаптації з нульовим параметром швидкості налаштування та визначимо значення виходів (**a**) та помилки (**e**):

```
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
    0    0
a =
    [0] [0] [0] [0]
e =
    [-1] [-0.4167] [1] [1]
```

Як видно з наведеного фрагмента, ваги не модифікуються, виходи мережі залишаються рівними нулю та адаптація не відбувається, тому що параметр швидкості налаштування не заданий.

Задаємо початкові значення ваг входів та зміщення, а також встановимо функції та параметр швидкості налаштування. При цьому вважаємо, що в функції виходу не повинна бути постійною складовою, а значить, для параметра швидкості налаштування зміщення встановлюємо нульове значення:

```
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputWeights{1,1}.learnParam.lr=0.2;
>> net.biases{1,1}.learnParam.lr=0;
>> [net1,a,e]=adapt(net,P,T);
>> net1.IW{1,1},a,e
ans =
    0.3454 -0.0694
a =
    [0] [-0.1167] [0.1100] [-0.0918]
e =
    [-1] [-0.3000] [0.8900] [1.0918]
```

Звідси видно, що процес адаптації почався, але досягнення необхідної точності, наприклад 0,015, одного циклу недостатньо.

Виконаємо послідовну адаптацію мережі протягом 30 циклів. Для обчислення середньоквадратичної помилки використовуємо оператор `mse`, а для конвертування масиву осередків `cell` в масив чисел подвоєної точності `double` застосуємо функцію `cell2mat`:

```
>> net=newlin([-1 1;-1 1],1, 0, 0);
>> net.IW{1}=[0 0];
>> net.b{1}=0;
>> net.inputWeights{1,1}.learnParam.lr=0.2;
>> net.biases{1,1}.learnParam.lr=0;
>> P=[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]];
>> T={-1 -5/12 1 1};
>> for i=1:30,
```

```

[net, a{i}, e{i}]=adapt(net, P, T);
W(i,:)=net.IW{1,1};
end
>> mse(cell2mat(e{30}))
ans =
    0.0017
>> W(30,:)
ans =
    1.9199    0.9250
>> cell2mat(e{30})
ans =
   -0.0056  -0.0081    0.0434    0.0699

```

Після 30 циклів адаптації мережі за чотирма навчальними векторами входу та мети відбулася зміна значень ваг входів та помилок.

Побудуємо графіки цих змін до функцій числа циклів (ітерацій) процесу адаптації. Розташуємо їх в одній фігурі одна над одною, застосовуючи для цього функцію subplot:

```

>> subplot(2,1,1)
>> for i=1:30, E(i)=mse(e{i}); end
>> semilogy(1:30, E, '+k')
>> xlabel('Епох'), ylabel('Помилка'), grid
>> subplot(2,1,2)
>> plot(0:30,[[0 0]; W], 'k');
>> xlabel('Епох'), ylabel('Вага входів w(i)'),grid

```

Графіки функцій представлені рисунку 2.2.

Не важко переконалися, що процес адаптації зажадав 12 кроків – помилка навчання досягла цьому етапі значення $1,489e-3$. Це свідчить про те, що навчальна вибірка була представницькою і мережа відповідає задачі, що вирішується.

При **груповому способі** подання навчальної послідовності вектори входу та цілі задаються масивами у форматі double.

Основний цикл адаптації мережі організується так:

```

net=newlin([-1 1;-1 1],1,0,0);
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
net=newlin([-1 1;-1 1],1,0,0.2);
net.IW{1}=[0 0];
net.b{1}=0;
net.inputWeights{1,1}.learnParam.lr=0.2;
P=[-1 -1/3 1/2 1/6; 1 1/4 0 2/3];
T=[-1 -5/12 1 1];
EE=10; i=1;
while EE>0.0017176

```

```

[net,a{i}, e{i}, pf]=adapt(net,P,T);
W(i,:)=net.IW{1,1};
EE=mse(e{i});
ee(i)=EE;
i=i+1;
end

```

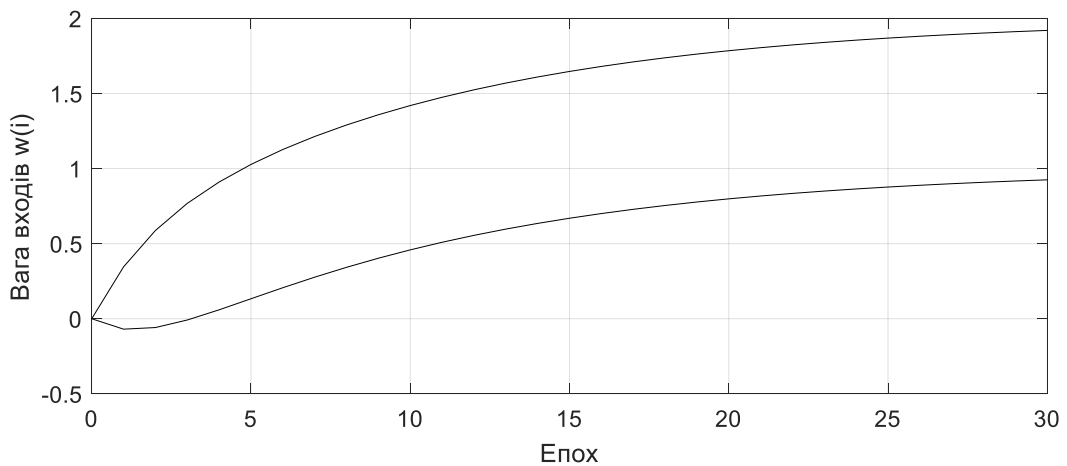
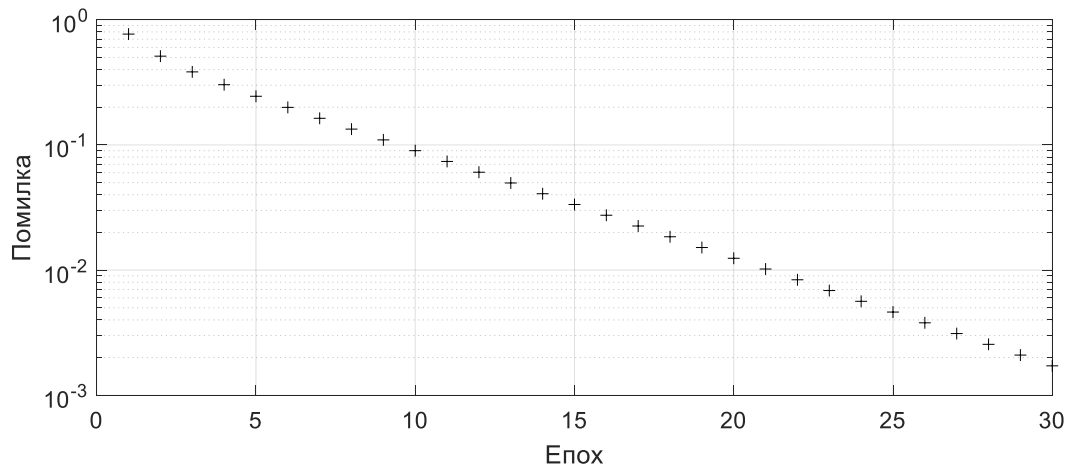


Рисунок 2.2 – Графіки зміни значень помилки та ваги входів у процесі адаптації

Результатом адаптації є наступні значення коефіцієнтів лінійної залежності (ваг входів), значень виходів мережі та середньоквадратичної похибки адаптації:

```

W(63,:)
ans =
    1.9114    0.8477
cell2mat(a(63))
ans =
    -1.0030   -0.3624    1.0172    0.9426
EE=mse(e{63})

```



EE =
0.0016

Наведемо процедуру адаптації нейронної мережі в динаміці на графіках. Об'єднаємо в одну фігуру графіки функцій виходів, ваг входів та помилки:

```
subplot(3,1,1)  
plot(0:63,[zeros(1,4); cell2mat(a)],'k')  
xlabel(''),ylabel('Виходи a(i)'),grid  
subplot(3,1,2)  
plot(0:63,[[0 0];W],'k')  
xlabel(''),ylabel('Вага входів w(i)'),grid  
subplot(3,1,3)  
semilogy(1:63, ee,'+k')  
xlabel('Епох'),ylabel('Помилка'),grid
```

Графіки функції наведені на рисунку 2.3.

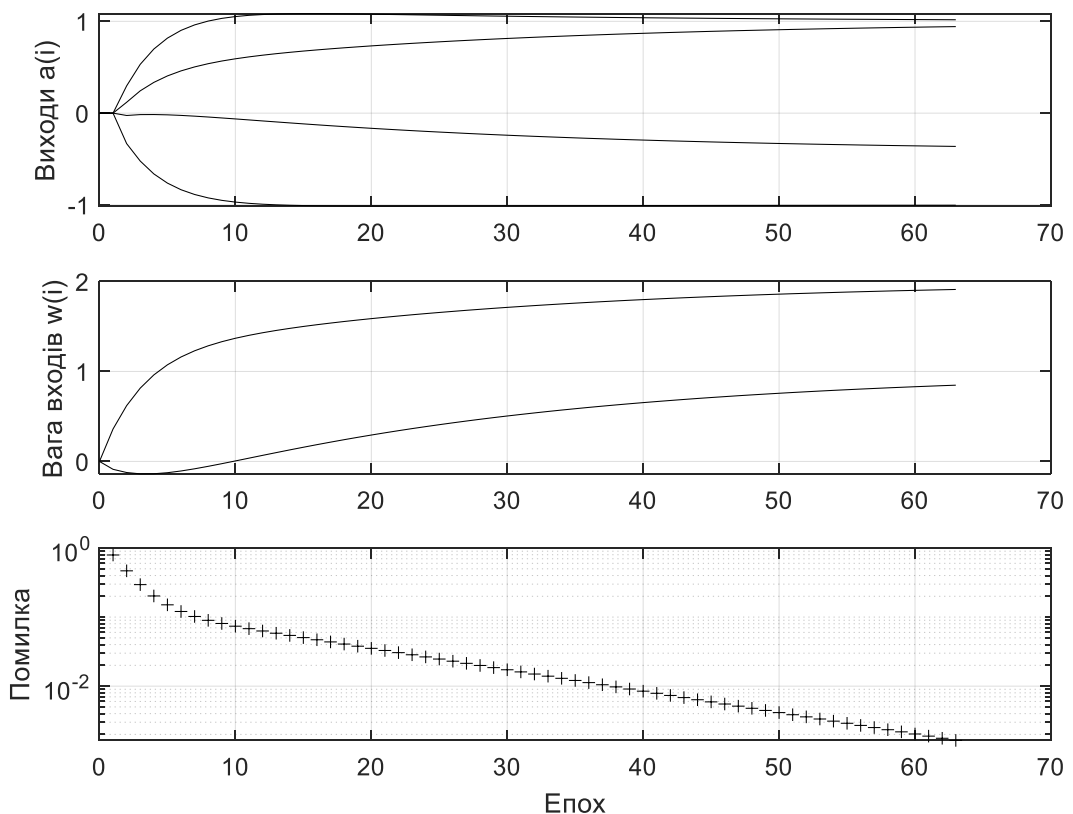


Рисунок 2.3 – Графіки вихідних сигналів мережі, ваг входів та помилки при адаптації з груповим поданням навчальної вибірки

Як впливає з аналізу графіків, для досягнення необхідної точності адаптації тут також знадобилося 12 епох.

Слід врахувати, що у лінійних динамічних мережах природним є

лише послідовний спосіб, оскільки ці мережі мають затримки сигналів.

Навчання мережі. Для навчання та налаштування параметрів мережі використовуються функції **trainwb** та **learnwh**, відповідно.

Після формування мережі, завдання навчальної вибірки (послідовним способом), введення параметрів швидкості налаштування та кількості циклів навчання (епох) отримаємо значення ваги входів та графік залежності величини помилки від числа циклів навчання, який представлений на рисунку 2.4.

```
net=newlin([-1 1;-1 1],1, 0, 0);
net.IW{1}=[0 0];
net.b{1}=0;
P=[-1;1] [-1/3; 1/4] [1/2; 0] [1/6; 2/3]];
T={-1 -5/12 1 1};
net.inputWeights{1,1}.learnParam.lr=0.2;
net.biases{1,1}.learnParam.lr=0;
net.trainParam.epochs=30;
net1=train(net,P,T);
W=net1.IW{1}
```

TRAINB, Epoch 0/30, MSE 0.793403/0.
 TRAINB, Epoch 25/30, MSE 0.00373997/0.
 TRAINB, Epoch 30/30, MSE 0.00138167/0.
 TRAINB, Maximum epoch reached.
 W =

1.9214 0.9260

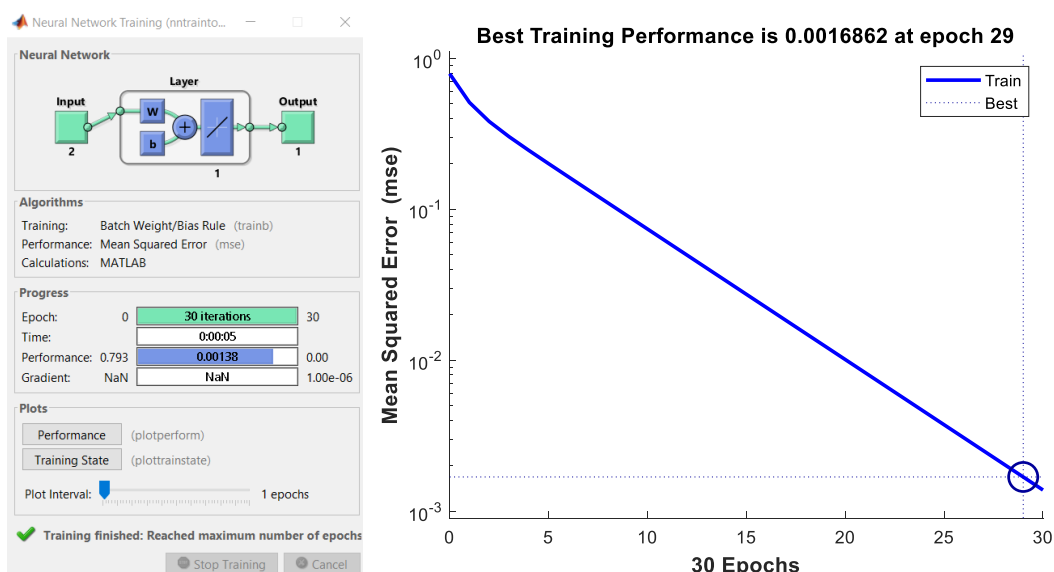


Рисунок 2.4 – Результати навчання НМ та графік залежності помилки навчання від числа циклів

2.2 Методика виконання роботи

Для виконання роботи студенти отримують індивідуальні завдання, варіанти яких наведені у таблиці 2.1.

У процесі створення мережі необхідно виконати.

1. Вибрати структуру мережі та провести її ініціалізацію (задати початкові значення ваги та зміщення).
2. Вибрати критерій якості та налаштування мережі.
3. Задати кількість циклів адаптації, провести адаптацію мережі.
4. Побудувати графіки функцій виходу мережі, ваг коефіцієнтів та критерію якості адаптації.

Таблиця 2.1 – Варіанти індивідуальних завдань

Варіант	Вхід	Значення вхідних сигналів навчальної послідовності $P1/P2$ (по порядку)						Значення функції цілі T					
		1	2	3	4	5	6	1	2	3	4	5	6
1	1	1/2	-1/2	1	-1/4	1/4	-1/2	0,9	-0,5	1,5	-1	1	-1
	2	0	-1/4	1/4	1/2	-3/4	1/6						
2	1	-1/2	-1	3/4	1/2	-1	-3/4	-0,5	-1,8	1	0,8	1	0
	2	1/5	3/4	-2/5	-1/4	-1	-1/4						
3	1	1/4	1	1/2	-1	-1/2	0	0,3	0,8	0	-0,7	0,2	0,5
	2	-1/4	-1/2	3/4	1/4	0	1						
4	1	1/2	-1/2	-1	1	3/4	-3/4	1,5	1,5	0	-3	-2	0,8
	2	1	1/2	-1/2	-1	-2/3	0						
5	1	-3/4	1/2	-1	0	1	3/4	1,5	0,7	1,5	-1	-2	-2,2
	2	0	1/4	-1/2	-1	0	-3/4						
6	1	3/4	-1	0	1/2	-1/4	-3/4	2	-1	1,5	1,2	-1	-2,3
	2	-3/4	0	-1	-1/2	1/2	1						
7	1	-3/4	1	1/4	1/2	-1	0	-1,6	1,8	-0,1	1	-1,5	-0,2
	2	1	-1/2	1	-1	0	1/2						
8	1	1	1/2	-2/3	3/4	-1	3/4	-0,3	2,4	0	0,2	-1,3	2,3
	2	1	-1	-1/2	1/2	0	-3/4						
9	1	3/4	-1	1/4	-1/4	1	-1/5	1,5	-2	-0,4	1	1	-1
	2	-3/4	1	1/2	-1	-1/4	3/4						
10	1	1/2	-1/2	-1	1	1/2	0	-0,3	0,7	-2,8	3,5	2	-1
	2	3/4	-1	1/2	-1	-2/3	2/3						



2.3 Зміст звіту

Звіт по роботі повинен містити завдання, структуру мережі, програму та результати адаптації мережі.

При захисті звіту студент повинен відповісти на питання щодо методики створення та адаптації лінійної нейронної мережі в пакеті прикладних програм.

2.4 Контрольні питання

1. У якому форматі записується оператор створення лінійної мережі?
2. Якою залежністю описується лінійна нейронна мережа?
3. Як представляється структура моделі одношарової лінійної мережі з R входами?
4. Як здійснюється ініціалізація мережі?
5. Як здійснюється адаптація мережі?
6. У яких форматах вводяться навчальні значення?
7. Який критерій зазвичай застосовується для налаштування лінійної мережі?
8. Яку значущість має параметр швидкості налаштування мережі?
9. Чим відрізняються послідовний та груповий способи завдання навчальних вибірок?

3 РОЗРОБКА РАДІАЛЬНОЇ БАЗИСНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ АПРОКСИМАЦІЇ ФУНКЦІЙ

Мета роботи: придбати навички створення, налаштування та моделювання радіальних базисних мереж для апроксимації функцій.

3.1 Методика формування, навчання та моделювання радіальної базисної мережі

Принцип функціонування мережі. Мережі радіальних базисних функцій (RBF) відрізняються від лінійних нейронних мереж тим, що містять шар прихованих радіально-симетричних прихованих нейронів (шаблонний шар). Для забезпечення радіально-симетричних властивостей необхідно:

- наявність центру, представленого вектором;
- наявність способу виміру відстані від центру до вхідного вектора;
- наявність спеціальної функції, яка відобразить функцію відстані.

У радіальних базисних мережах всі умови забезпечуються застосуванням радіального базисного нейрона з функцією активації виду:

$$a = \text{radbas}(n) = e^{-n^2},$$

де n – вхід функції активації, що визначається як модуль різниці вектору ваги і вектору входу, помножений на зсув: $n = \|\mathbf{p} - \mathbf{w}\| b$.

Практично це означає, що вихідний сигнал шаблонного нейрона – це функція відстані між вхідним вектором та збереженим центром. Що ближче вхідний вектор до центру, то більший вихідний сигнал нейрона (див. рис. 3.1). При цьому радіальний базисний нейрон діє як індикатор, який формує значення 1 коли вхід \mathbf{p} ідентичне вектору ваг \mathbf{w} . Зміщення b дозволяє коригувати чутливість радіального базисного нейрона.

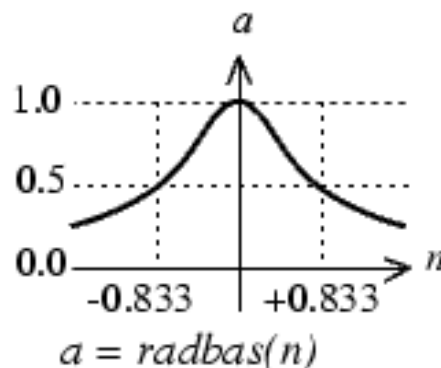


Рисунок 3.1 – Графік функції активації

Структура мережі показано рисунку 3.2. Мережа містить радіальний базисний шар та лінійний шар.

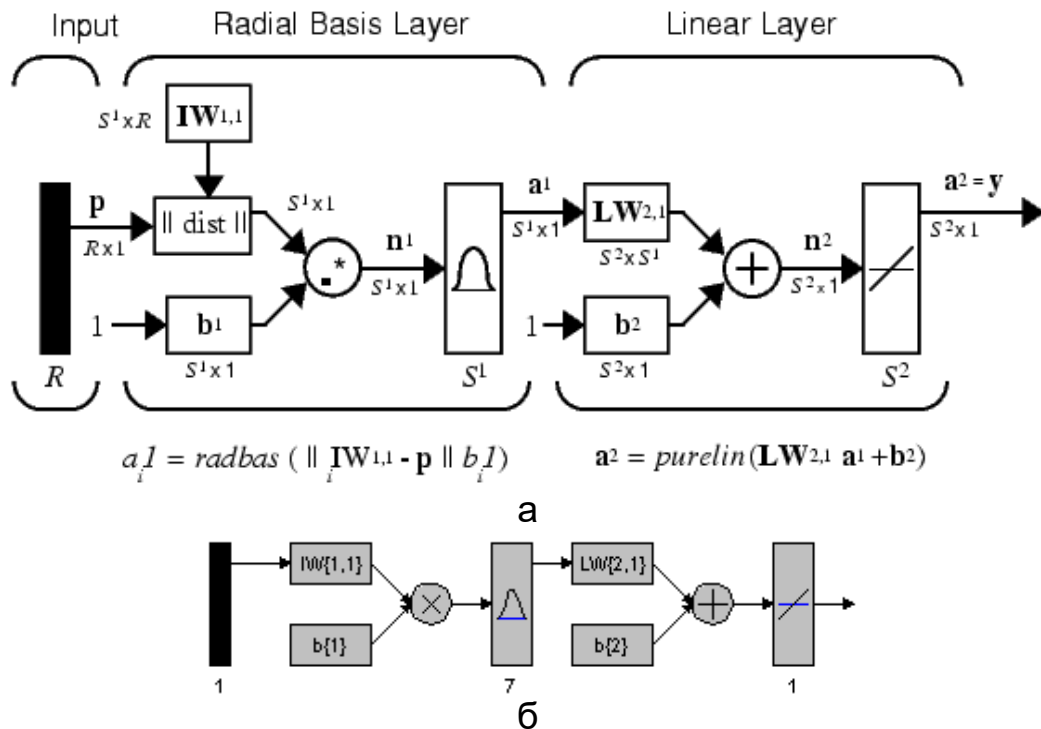


Рисунок 3.2 – Повна (а) та спрощена (б) структури моделі радіальної базисної мережі

Прихований радіальний базовий шар має S^1 нейронів, а вихідний шар - S^2 нейронів. Радіальний базовий шар містить спеціальний блок dist, на вхід якого подаються вхідний вектор \mathbf{p} та матриця ваг \mathbf{IW} . Виходом блоку є вектор, що складається з S^1 елементів, які визначаються відстанями між i -м вектором входу та i -м вектор-рядком \mathbf{IW} матриці ваг. Вихід блоку dist множиться по-елементна на вектор зміщення \mathbf{b} і формує вхід функції активації. Як функція активації використовується зазвичай функція Гауса.

Побудова мережі. Для створення радіальної базової мережі призначені М-функції newrbe та newrb. Перша дозволяє побудувати мережу з нульовою помилкою, друга дозволяє керувати кількістю нейронів прихованого шару. Недолік функції newrbe полягає в тому, що вона формує мережу з числом нейронів у прихованому шарі, рівним числу елементів навчальної множини. Якщо таких елементів багато, що притаманно реальних випадків, то прийнятне рішення досягається важко.

Створимо мережу з нульовою помилкою:

```
net=newrbe(P,T,SPREAD);
```

Вхідними аргументами функції newrbe є масиви вхідних векторів \mathbf{P} та векторів цілі \mathbf{T} , а також параметр впливу SPREAD. Функція встановлює ваги першого шару рівними \mathbf{P} . Зміщення встановлюються

рівними $0,8326/ \text{SPREAD}$. Це означає, що рівень перекриття радіальних базисних функцій дорівнює 0,5 та всі входи в діапазоні $\pm \text{SPREAD}$ вважаються значущими.

Ваги другого шару можуть бути знайдені шляхом моделювання виходів першого шару ($A\{1\}$) та подальшого розв'язання системи лінійних рівнянь алгебри:

$$[W\{2,1\} \ b\{2\}] * [A\{1\}; \text{ones}] = T$$

Оскільки входи другого шару $A\{1\}$ і цілі T відомі, а функція активації лінійна, то обчислення ваги і зміщень другого шару досить вирішити систему рівнянь:

$$Wb = T / [P; \text{ones}(1, \text{size}(P,2))]$$

Розглянемо процедуру створення та моделювання радіальної базисної мережі для довільної функції.

Значення входів змінюються в діапазоні від -1 до 1. Навчальна вибірка складена для точки 21 функції через 0,1. Графік функції представлений рисунком 3.3.

$$P = -1:0.1:1;$$

$$T = [-.9602 \ -.5770 \ -.0729 \ .3771 \ .6405 \ .6600 \ .4609 \\ .1336 \ -.2013 \ -.4344 \ -.5000 \ -.3930 \ -.1647 \ .0988 \\ .3072 \ .3960 \ .3449 \ .1816 \ -.0312 \ -.2189 \ -.3201];$$

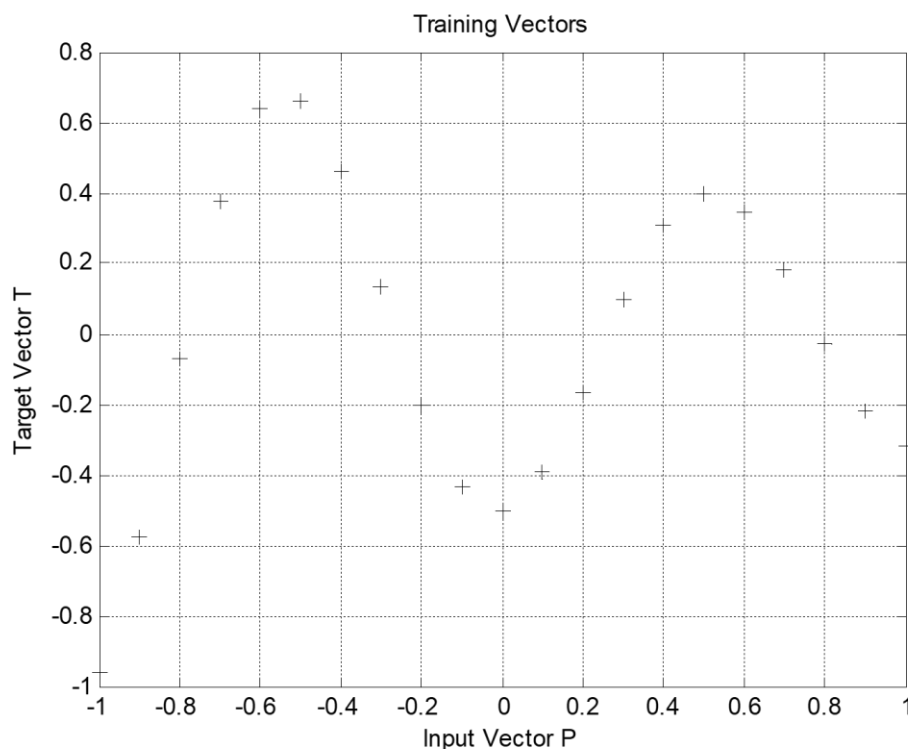


Рисунок 3.3 – Графік навчальної функції (вектор T)

```

plot(P,T,'+');
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');grid

```

%Збережемо поточний графік та сформуємо мережу:

```

hold on
net=newrbe(P,T);
net.layers{1}.size
Warning: Rank deficient, rank = 13, tol = 2.2386e-014.
> In newrbe>designrbe at 120
   In newrbe at 103
ans =
    21    % Кількість нейронів прихованого шару дорівнює 21

```

Зробимо моделювання мережі:

```

V=sim(net,P);
plot (P,V,'ob','MarkerSize',5,'LineWidth',2)
p=[-0.75 -0.25 0.25 0.75];
v=sim(net,p);
plot(p,v,'+k','MarkerSize',10,'LineWidth',2)

```

Результати моделювання наведено на рисунку 3.4.

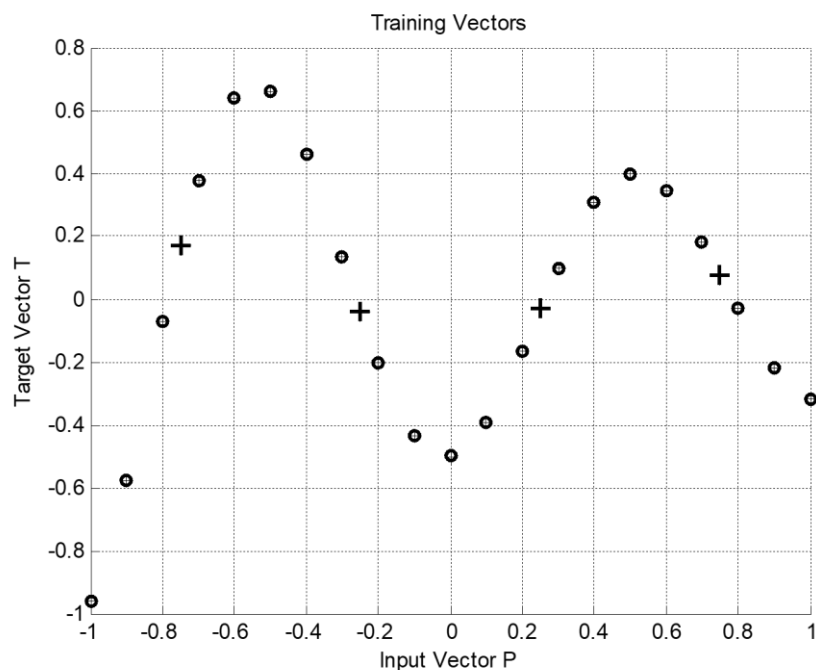


Рисунок 3.4 – Результати моделювання мережі під час подачі на вхід чотирьох нових значень

Як видно з рисунку 3.4, вектор T для нових значень входу визначено з високою точністю.

Створимо мережу з меншою кількістю нейронів, застосувавши функцію `newrb`. Нову мережу навчимо колишньою послідовністю цілей:

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
      .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
      .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','MarkerSize',4,'LineWidth',2)
hold on
GOAL=0.01; % Допустиме значення помилки
net=newrb(P,T,GOAL);
net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69051
ans =
    6 % Кількість нейронів прихованого шару дорівнює 6
v=sim(net,p);
p=[-0.75 -0.25 0.25 0.75];
v=sim(net,p);
plot(p,v,'+k','MarkerSize',10,'LineWidth',2)
```

Як очевидно з М-файлу, число нейронів першого шару дорівнює 6. У цьому забезпечується середньоквадратична помилка менше 0,01. Результати моделювання наведено на рисунку 3.5.

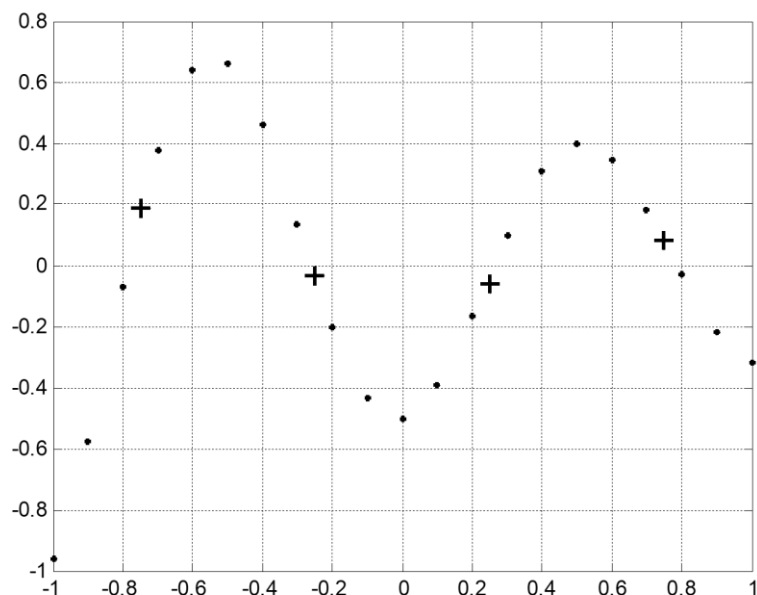


Рисунок 3.5 – Результати моделювання мережі з шістьма нейронами

Апроксимації функції. При застосуванні радіальних базисних мереж для апроксимації функцій використовується ідея розкладання довільної функції на ряд радіальних базисних функцій наступним чином:

$$f(x) = \sum_{i=1}^N a_i \phi_i(x) ,$$

де $\phi_i(x)$ – радіальна базисна функція.

Розглянемо процес апроксимації трьох радіальних базисних функцій, заданих на інтервалі [-3 3]:

```
p = -3:.1:3;
a1 = radbas(p);
a2 = radbas(p-1.5);
a3 = radbas(p+2)*0.5;
a4 = a1 + a2 + a3;
plot(p,a1,'b-',p,a2,'b--',p,a3,'b--',p,a4,'m-')
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input p');
ylabel('Output a');
```

Результати апроксимації представлені рисунку 3.6.

З рисунка видно, що при розкладанні довільної функції радіальними функціями базисними забезпечується необхідна гладкість.

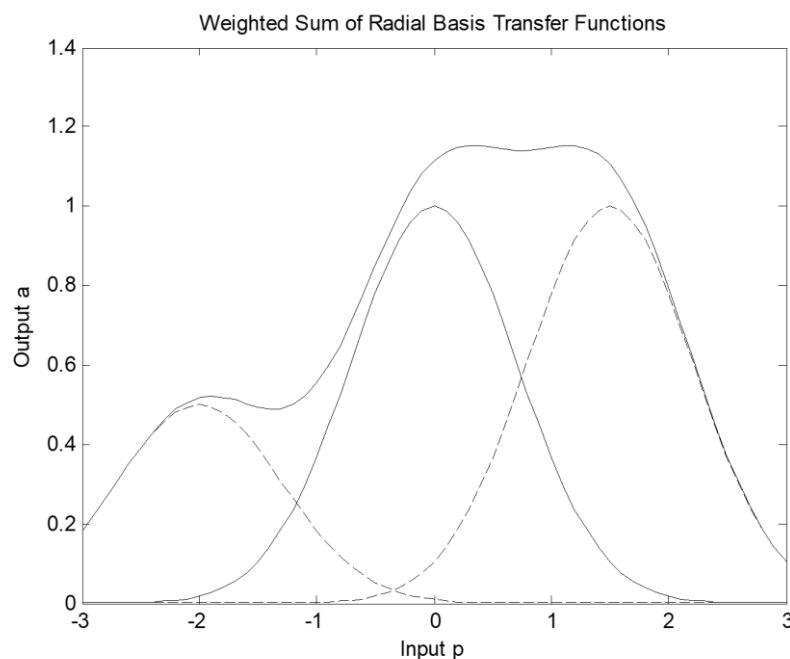


Рисунок 3.6 – Графіки базисних функцій та результат їх апроксимації

Ступінь гладкості та точність апроксимації пов'язані з параметром впливу SPREAD. Як відомо, цей параметр визначає діапазон значних входів (див. вище).

Розглянемо, у чому проявляється цей вплив.

Використовуючи попередній приклад навчальної вибірки, задаватимемо значення SPREAD, рівними 0,01; 1 та 20.

Введемо М-файл зі значенням SPREAD=0,01 та отримаємо результат (рис. 3.7).

```
P = -1:1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
     .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988 ...
     .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
GOAL=0.01;
SPREAD=0.01;
net=newrb(P,T,GOAL,SPREAD);
net.layers{1}.size
NEWRB, neurons = 0, SSE = 2.758
```

```
ans =
    19    % Кількість нейронів прихованого шару дорівнює 19
```

```
X=-1:0.01:1;
Y=sim(net,X);
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P');
ylabel('Target Vector T');
```

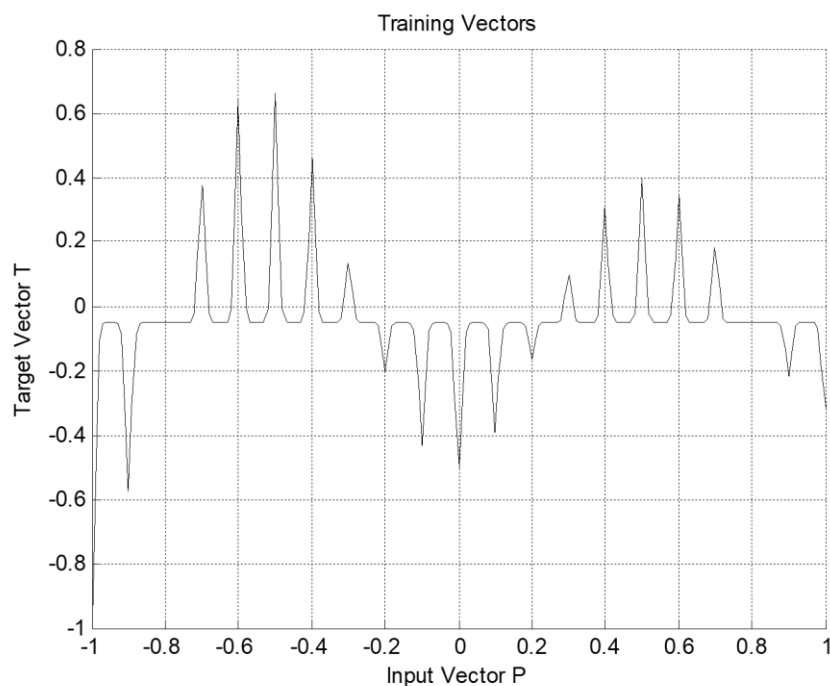


Рисунок 3.7 – Результат апроксимації при SPREAD=0,01

Як очевидно з результатів моделювання, мережа складається з 19 нейронів і забезпечує гладкості функції апроксимації.

Розглянемо процес апроксимації при SPREAD=1.

```
P = -1:.1:1;
```

```
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
```

```
    .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
```

```
    .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
```

```
GOAL=0.01;
```

```
>> SPREAD=1;
```

```
>> net=newrb(P,T,GOAL,SPREAD);
```

```
net.layers{1}.size
```

```
NEWRB, neurons = 0, SSE = 3.69051
```

```
ans =
```

```
    6
```

```
>> X=-1:.01:1;
```

```
Y=sim(net,X);
```

```
hold on;
```

```
plot(X,Y);
```

```
title('Training Vectors');
```

```
xlabel('Input Vector P');
```

```
>> grid
```

Результати моделювання мережі представлені на рис. 3.8.

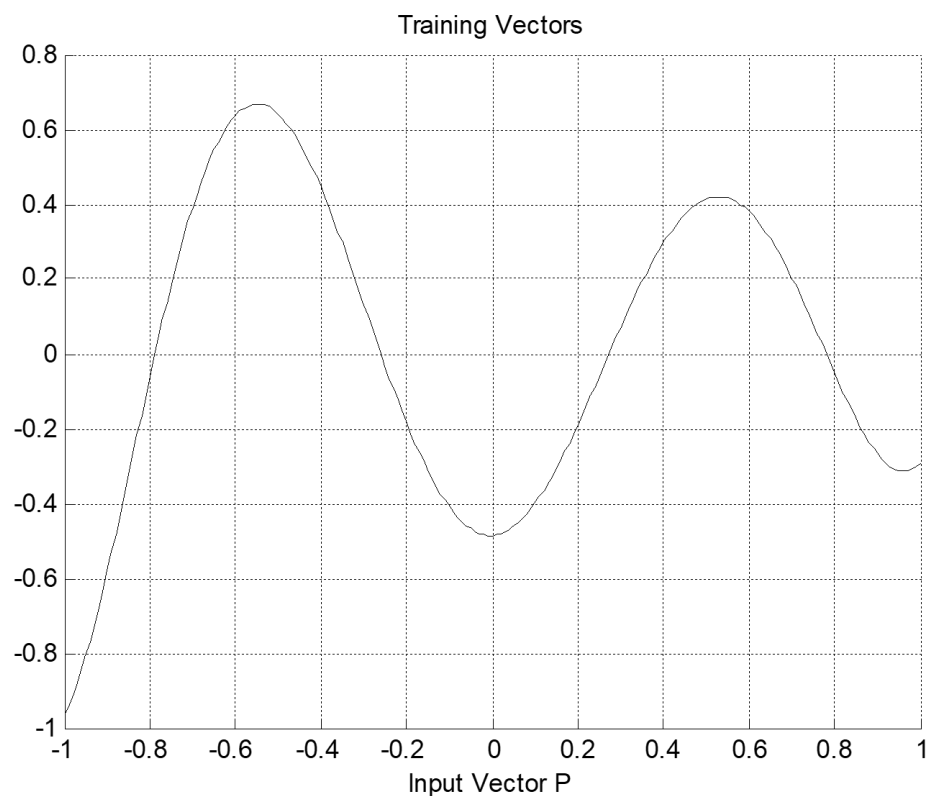


Рисунок 3.8 – Графік функції апроксимації при значенні SPREAD=1

Ця мережа складається з 6 нейронів прихованого шару і добре апроксимує нелінійну залежність, задану безліччю з 21 точки.

Встановимо параметр впливу SPREAD=20.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609
     .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988
     .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
GOAL=0.01;
SPREAD=20;
net=newrb(P,T,GOAL,SPREAD);
net.layers{1}.size
NEWRB, neurons = 0, SSE = 3.69972
ans =
    21      % Кількість нейронів прихованого шару в мережі
X=-1:.01:1;
Y=sim(net,X);
hold on;
plot(X,Y);
title('Training Vectors');
xlabel('Input Vector P'); grid
```

Результат моделювання наведено на рисунку 3.9.

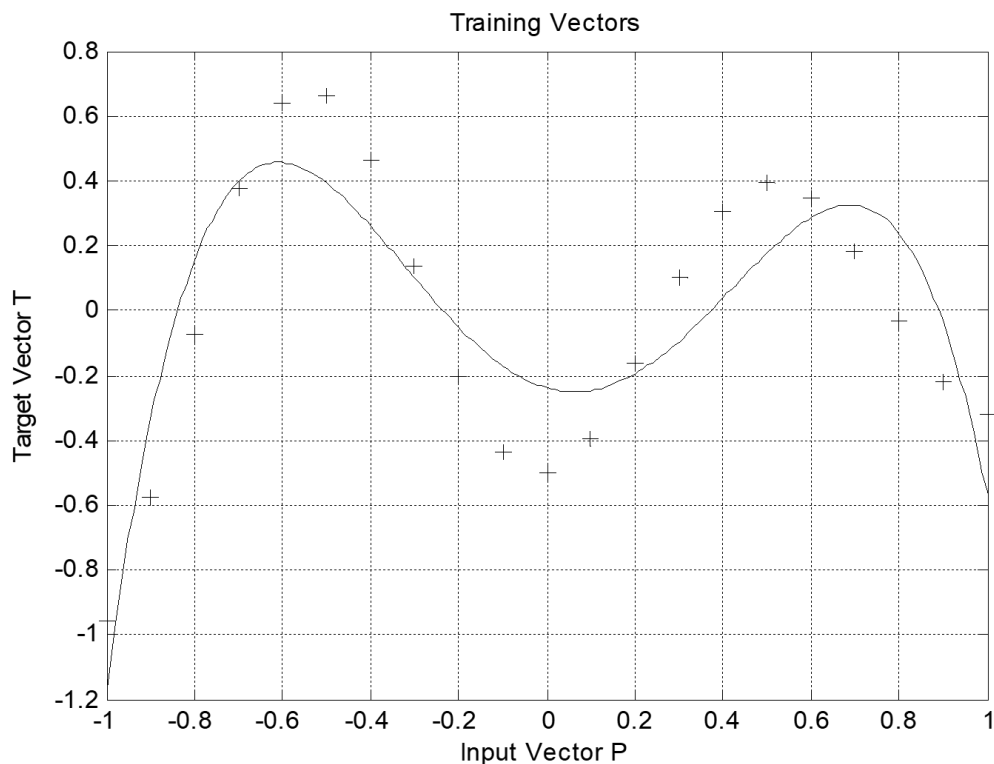



Рисунок 3.9 – Графік функції апроксимації при значенні SPREAD=20



З рисунку 3.9 видно, що при великому значенні параметра впливу SPREAD і великій кількості нейронів прихованого шару (21 нейрон) функції активації перекриваються, і кожен базисний нейрон для всіх значень входу генерує вихідні значення, близькі до 1. Це призводить до того, що мережа не може забезпечити необхідну точність через обчислювальні проблеми.

Таким чином, за результатами виконаних досліджень можна зробити висновок про те, що параметр впливу SPREAD слід брати більшим, ніж крок розбиття функції навчальної послідовності, але меншим за інтервал її розбиття. Для розглянутого прикладу цей діапазон становить від 0,1 до 2.

3.2 Методика виконання роботи

Для виконання роботи студенти отримують індивідуальні завдання, варіанти яких наведені у таблиці 3.1.

У процесі виконання роботи потрібно виконати таке.

1. Із застосуванням функції `newrb` створити структуру мережі з нульовою помилкою та зробити її навчання.

2. Із застосуванням функції `newrb` створити структуру мережі та провести її навчання за різних значень параметра впливу SPREAD.

3. Побудувати графіки функцій виходу мережі та зробити висновок про якість апроксимації.

3.3 Зміст звіту

Звіт по роботі повинен містити завдання, структуру мережі, програми та результати роботи мережі.

При захисті звіту студент повинен відповісти на питання щодо методики створення та застосування радіальної базисної нейронної мережі..

3.4 Контрольні питання

1. Чим відрізняється структура радіальних базисних мереж від структури лінійних нейронних мереж?

1. 2. Якими властивостями повинна мати радіальна базова мережа?

3. Які функції активації застосовують у радіальних базисних мережах?

4. Навіщо призначений прихований шаблонний шар?

5. Які М-функції використовуються для створення радіальних базисних мереж і в чому їхня відмінність?

6. Як задається навчальна вибірка для апроксимації функцій?

7. Яка ідея використовується для апроксимації функцій радіальної

базової мережі?

8. Від чого залежить кількість нейронів прихованого шару?

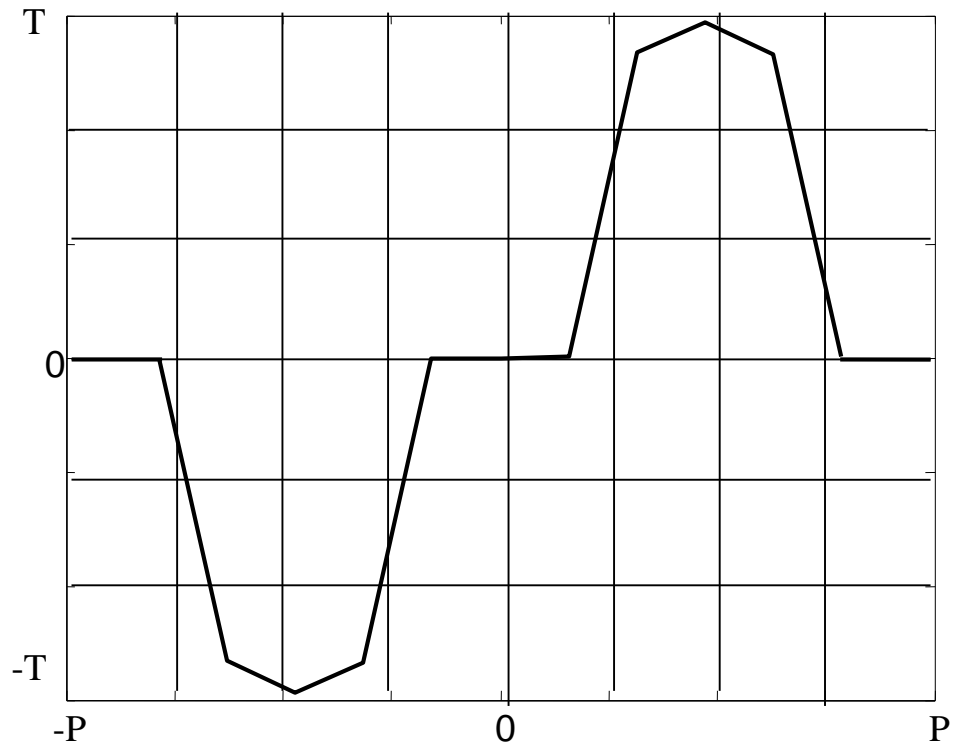
9. Як впливає параметр SPREAD на точність і гладкість функції апроксимації. У якому діапазоні він має бути?

3.5 Індивідуальні завдання

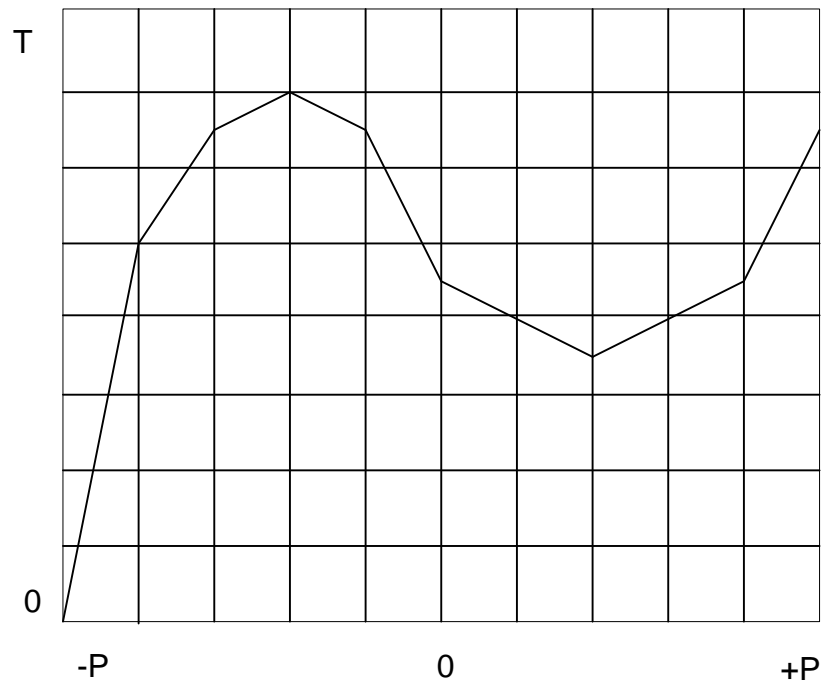
Варіант індивідуального завдання визначається номером рисунка 3.10 і рядком таблиці. Наприклад, варіант "2-а" означає, що вихідні дані для індивідуального завдання наведено на рис. 3.10,а та у рядку 2 таблиці 3.1.

Таблиця 3.1 – Варіанти завдань

Варіант	Діапазон вхідного вектору Р		Діапазон цілі Т	
	от	до	от	до
1-а	-1	+1	0	+1
2-б	-2	+2	0	+2
3-в	-3	+3	-1	+1
4-г	-4	+4	-2	+2
5-а	-5	+5	-3	+3
6-б	-1	+1	0	+1
7-в	-2	+2	0	+2
8-г	-3	+3	-1	+1
9-д	-4	+4	-2	+2
Варіант	Діапазон вхідного вектору Р		Діапазон цілі Т	
	от	до	от	до
10-б	-5	+5	-3	+3
11-в	-1	+1	0	+1
12-г	-2	+2	0	+2
13-а	-3	+3	-1	+1
14-б	-4	+4	-2	+2
15-в	-5	+5	-3	+3



а



б

Рисунок 3.10 – Лист 1

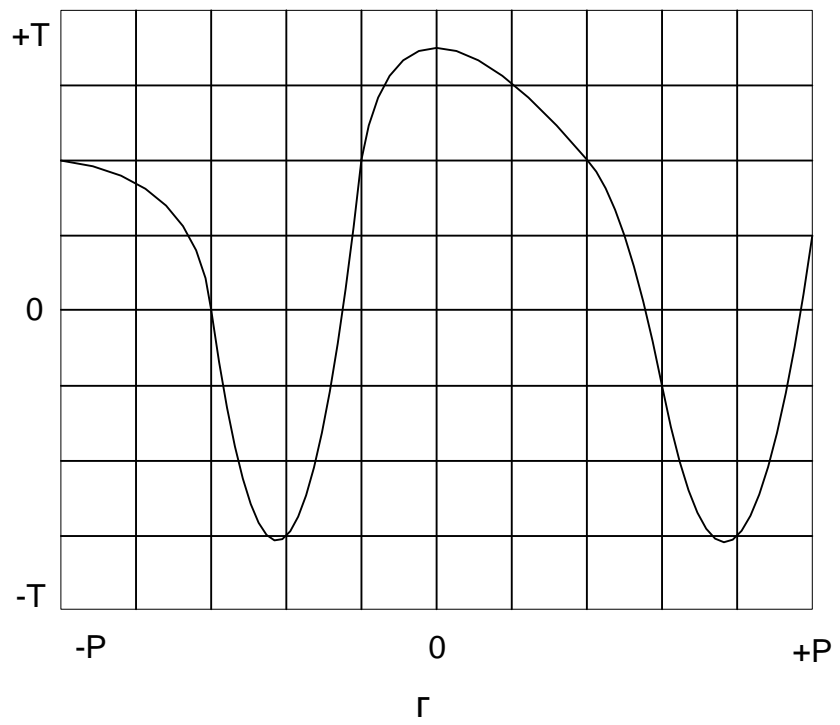
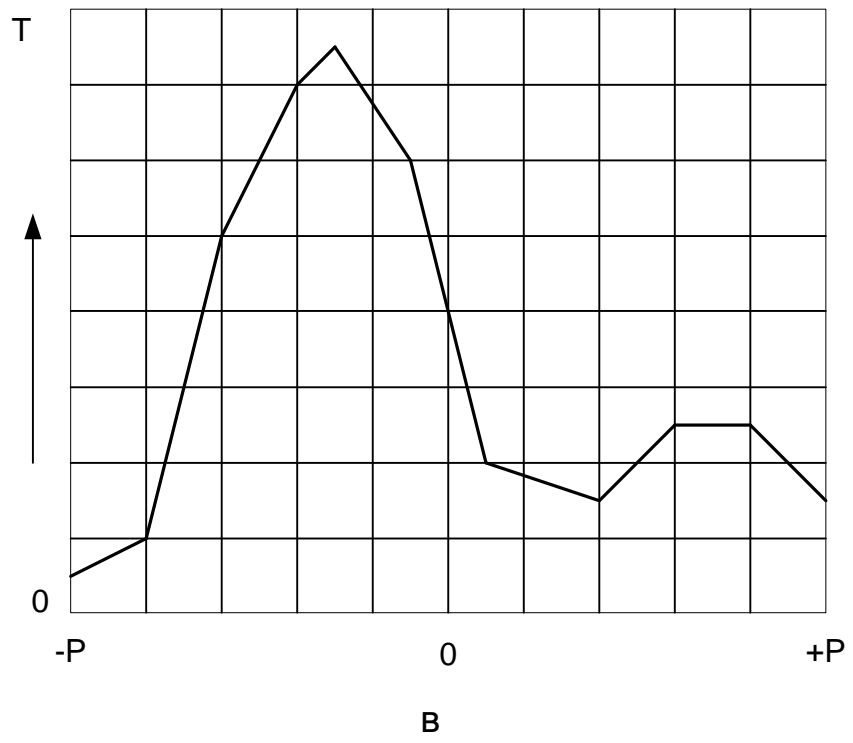


Рисунок 3.10 – Лист 2

4 РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ МОДЕЛИРОВАНИЯ РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ СТАЦІОНАРНИХ СИГНАЛІВ

Мета роботи – придбати вміння та навички у створенні та навчанні нейронних мереж для ідентифікації параметрів стаціонарних сигналів.

4.1 Постановка завдання та синтез мережі для моделювання стаціонарних сигналів

Постановка задачі передбачення стаціонарного сигналу. У цифрових системах керування безперервні сигнали видаються дискретними сигналами шляхом застосування операції квантування. Велике значення має можливість прогнозувати значення сигналу на кілька кроків уперед. Час кроку визначається періодом дискретності цифрового сигналу.

Нехай, наприклад, потрібно передбачити значення сигналу y_k на виході мережі на момент часу t_k , використовуючи для цього 5 останніх значень сигналів T . У математичному сенсі – це завдання екстраполяції функції, яка може бути сформульована в такий спосіб:

$$\begin{cases} p_{k-1} = T_{k-1}, & i = 1, \dots, 5; \\ y_k = \sum_{i=1}^5 w_i p_{k-1}, \end{cases}$$

де w_i – вагові коефіцієнти мережі.

Задаємо гармонійний сигнал y_k , діапазон спостереження $0 \dots 5$ с та період дискретності (такт квантування) $h = 0,025$ с:

$$\begin{aligned} y_k &= \sin(4\pi t_k) = \sin(4\pi kn), \\ t_k &= t_0 : h : t_f = 0 : 0.025 : 5. \end{aligned}$$

З позиції теорії нейронних мереж завдання екстраполяції перетворюється на завдання налаштування параметрів та навчання мережі.

Синтез нейронної мережі для передбачення значень сигналу. Оскільки сигнал стаціонарний та співвідношення між минулими та майбутніми значеннями незмінні, можна скористатися лінійною моделлю нейронної мережі. Для аналізу 5-ти сигналів мережа повинна складатися з одного нейрона з 5 входами та містити лінію затримки з п'ятьма блоками запізнення. Структура мережі наведена на рисунку 4.1.

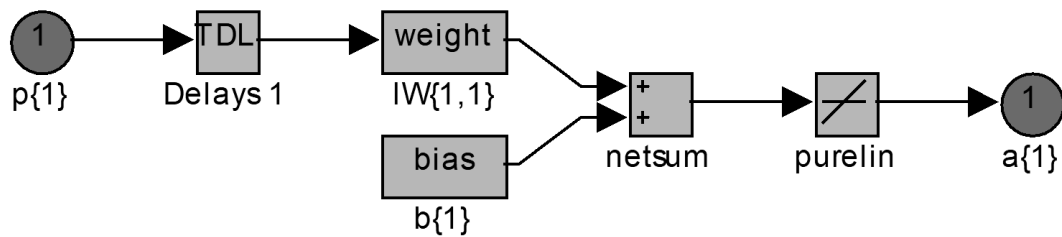


Рисунок 4.1 – Структура мережі для екстраполяції стаціонарного сигналу

Вхідну послідовність виберемо на інтервалі від 0 до 1 с, а контрольні підмножині сформуємо із значень сигналів в інтервалі від 3 до 5 с. Вхідна послідовність матиме довжину $Q_1 = 1/h$, кожен вектор входу повинен бути розкладений на компоненти, які будуть відповідати запізненням сигналу.

Сформуємо сигнал, задаємо вхідний та цільовий вектори, а також задаємо структуру мережі:

```

clf; % Очищення графічного вікна
tf=5;
h=0.025
figure(gcf)
echo on
time = 0: h:tf; % від 0 до tf секунд
T = sin(time*4*pi); % Завдання сигналу
Q = length(T); % Завдання довжини вектору
P = zeros(5,Q);
P(1,2:Q) = T(1,1:(Q-1));
P(2,3:Q) = T(1,1:(Q-2));
P(3,4:Q) = T(1,1:(Q-3));
P(4,5:Q) = T(1,1:(Q-4));
P(5,6:Q) = T(1,1:(Q-5));
figure(1)
plot(time,T) % Побудова графіка
xlabel('Time');
ylabel('Target Signal');
title('Signal to be Predicted');
grid;
net = newlind(P,T); % Завдання структури мережі

```

Графік вхідного сигналу для завдання передбачення показано на рисунку 4.2.

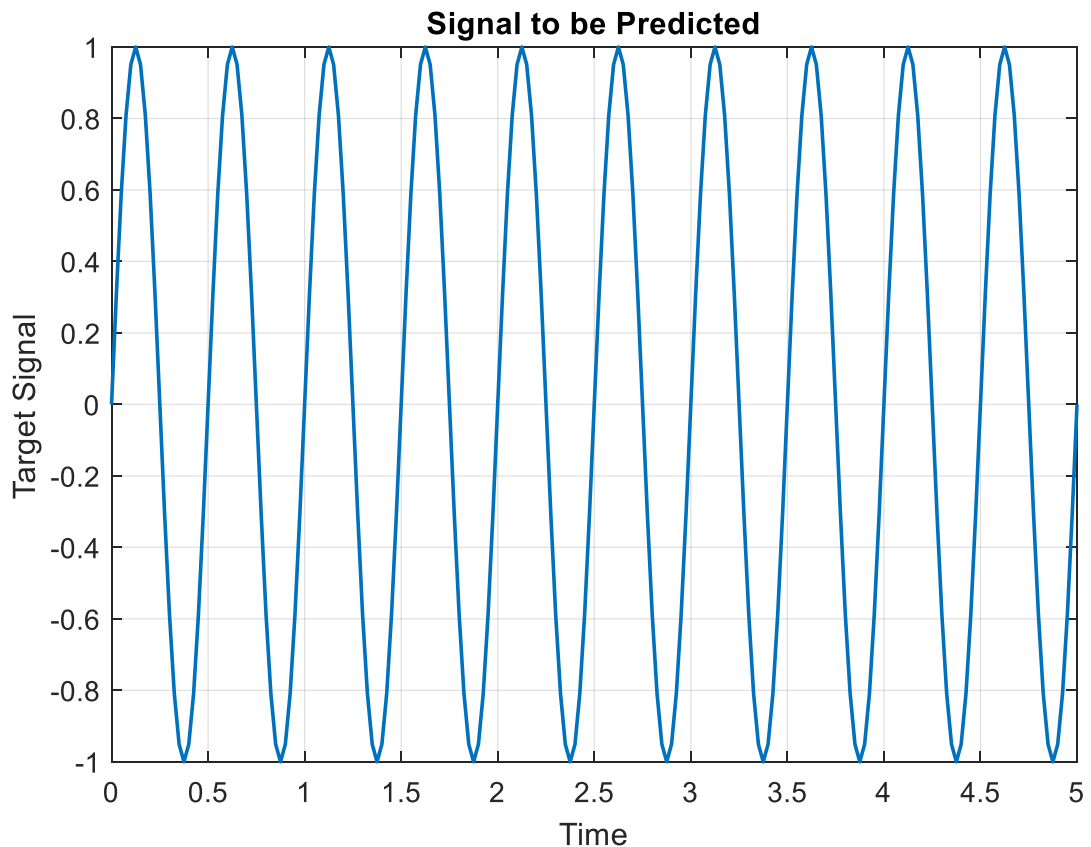


Рисунок 4.2 – Графік стаціонарного дискретного сигналу

Виконаємо перевірку мережі, використовуючи дві множини – навчальна та контрольна.

```

Q1=1/h; % Визначення довжини навчальної множини (Q1=1/h)
P1=zeros(5,Q1); % Обнуління матриці вхідного вектору
P1(1,1:Q1)=T(1,1:Q1); % Зіставлення векторів входу та цілі для 5
тактів
P1(2,2:Q1)=T(1,1:(Q1-1));
P1(3,3:Q1)=T(1,1:(Q1-2));
P1(4,4:Q1)=T(1,1:(Q1-3));
P1(5,5:Q1)=T(1,1:(Q1-4));
T1=T(1,6:(Q1+5)); % Формування вектору цілі з 6 по Q1+5 такти
T2=T(1,3/h:Q);
net = newlind(P1,T1);
a=sim(net,P1(:,1:Q1)); % Моделювання мережі
t1=6:Q1+5; %
figure(2)
plot(time(t1),a,'*r',time(1:Q1+5),T(1,1:Q1+5))
xlabel('Time,c');
grid;

```

Результати моделювання представлені на графіку рисунка 4.3.

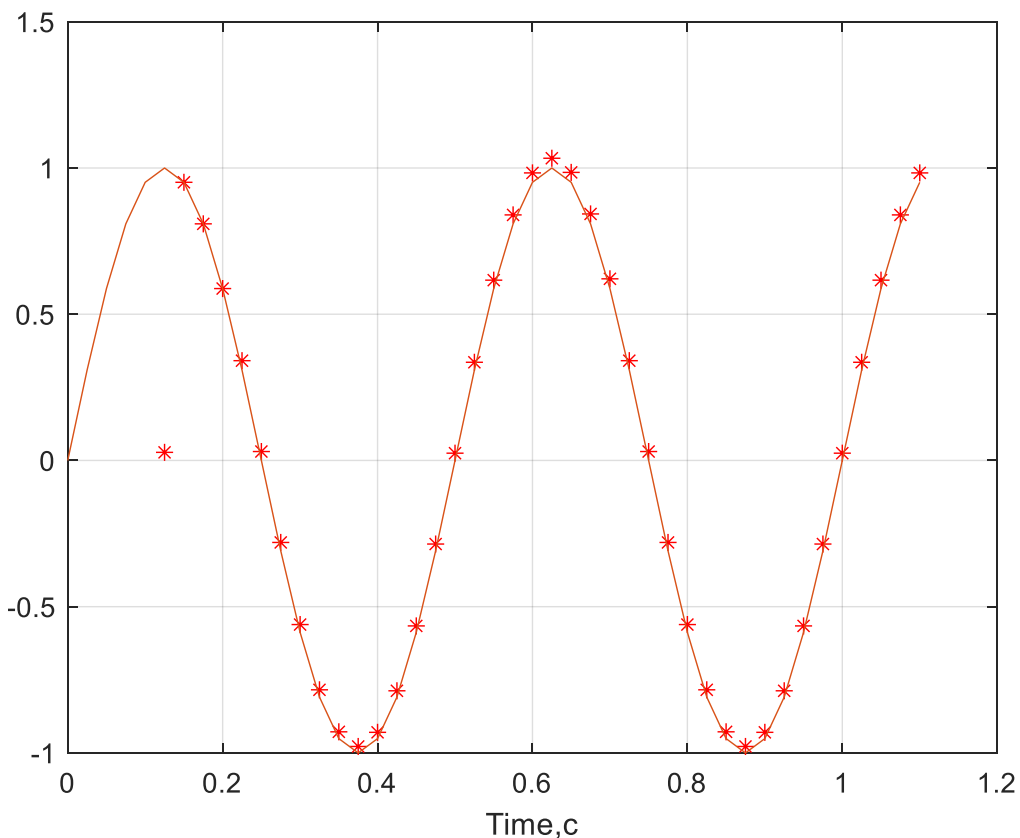


Рисунок 4.3 – Графіки зміни модельованого (зірочки) та фактичного сигналів

З рисунку 4.3 видно, що нейронна мережа забезпечує досить високу точність відстеження сигналу.

Тепер перевіримо роботу мережі, використовуючи контрольну множину. Задаємо довжину вхідної послідовності $N1$.

```

N1=20;
Tt=T2(1,1:N1);
P2(1,:)=Tt(1,:);
P2(2,2:end)=Tt(1,1:end-1);
P2(3,3:end)=Tt(1,1:end-2);
P2(4,4:end)=Tt(1,1:end-3);
P2(5,5:end)=Tt(1,1:end-4);
a=sim(net,P2); % Моделювання мережи
figure(3),clf
h1=plot(time(1:size(P2, 2)-5), a(1:end-5),'*');
hold on
h2=plot(time(1:size(P2, 2)-5), Tt(6:end), 'r');
grid;

```

Використовуючи властивості графічного об'єкта, обчислимо похибки екстраполяції у функції довжини навчальної вибірки:

```

y1=get(h1,'YData'); y2=get(h2,'YData')
minlength=min(length(y1), length(y2));
e=y1(1:minlength)-y2(1:minlength); % Розрахунок помилки
nre=sqrt(mse(e)); % Обчислення квадратного кореня
plot(time(1:size(P2, 2)-5),e);
legend('a','T','e');
xlabel('Time,c');
ylabel('a,T,e')

```

Результати моделювання мережі та обчислення похибки екстраполяції представлені на рисунку 4.4.

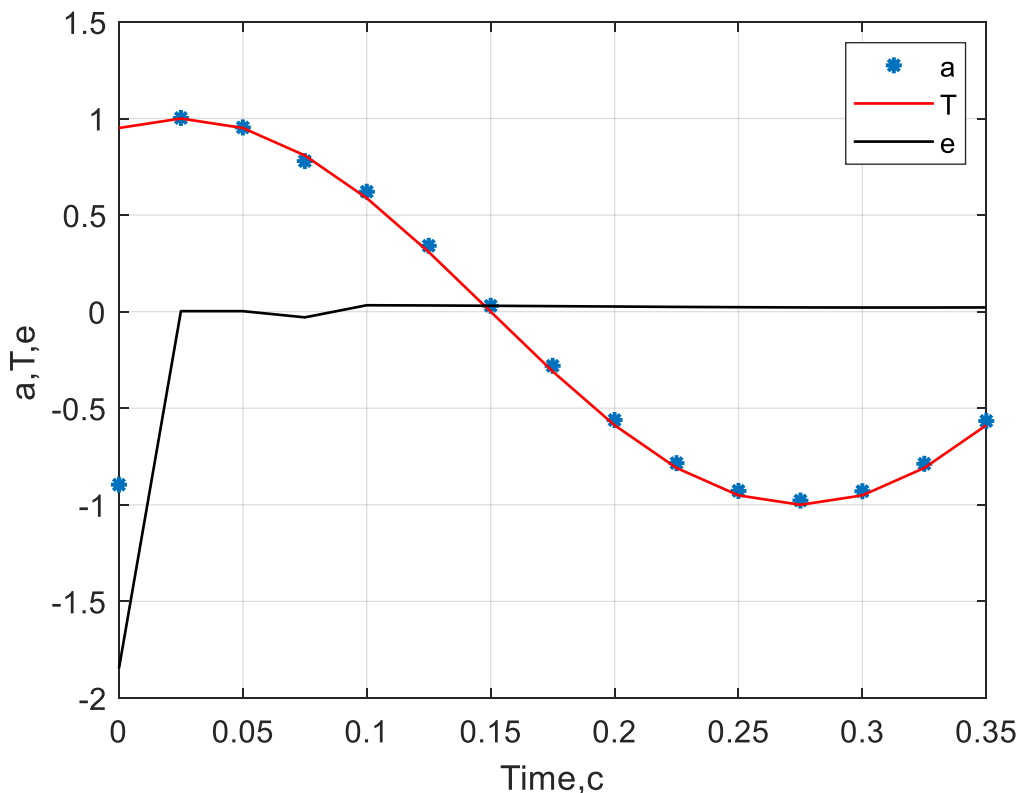


Рисунок 4.4 – Графік реакції мережі на контрольну множину

З аналізу малюнка 4.4 можна дійти невтішного висновку у тому, що нейронна мережа забезпечує достатню точність екстраполяції – помилка дорівнює $e = 0.0220$.

4.2 Методика виконання роботи

Для виконання роботи студенти отримують індивідуальні завдання, варіанти яких наведені у таблиці 4.1.

У процесі виконання роботи потрібно виконати:

1. Визначити довжину вхідного вектора.
2. Сформулювати навчальну послідовність (вектор мети).

3. Здійснити синтез нейронної мережі та її навчання.

4. Побудувати графіки функцій виходу мережі та зробити висновок про якість екстраполяції сигналу.

Таблиця 4.1 – Варіанти індивідуальних завдань для апроксимації функції

№ вар.	Функція	t_f	h	№ вар.	Функція	t_f	h
1	$y = \sin(2\pi t_k)$	10	0,05	11	$y = \sin(2\pi t_k)$	8	0,04
2	$y = \sin(2,5\pi t_k)$	7,5	0,05	12	$y = \sin(2,5\pi t_k)$	5	0,04
3	$y = \sin(3\pi t_k)$	6	0,02	13	$y = \sin(3\pi t_k)$	7,5	0,05
4	$y = \sin(3,5\pi t_k)$	5	0,02	14	$y = \sin(3,5\pi t_k)$	6	0,05
5	$y = \sin(4\pi t_k)$	4,5	0,02	15	$y = \sin(4\pi t_k)$	4	0,05
6	$y = \sin(4,5\pi t_k)$	4,25	0,01	16	$y = \sin(4,5\pi t_k)$	3	0,02
7	$y = \sin(5\pi t_k)$	4	0,01	17	$y = \sin(5\pi t_k)$	3	0,02
8	$y = \sin(5,5\pi t_k)$	3	0,01	18	$y = \sin(5,5\pi t_k)$	2,5	0,02
9	$y = \sin(6\pi t_k)$	2	0,005	19	$y = \sin(6\pi t_k)$	1,5	0,01
10	$y = \sin(1,5\pi t_k)$	8	0,04	20	$y = \sin(1,5\pi t_k)$	10	0,05

4.3 Зміст і захист звіту

Звіт по роботі повинен містити завдання, структуру мережі, програми та результати роботи мережі.

При захисті звіту студент повинен відповісти на питання щодо методики створення та застосування нейронної мережі для екстраполяції функцій.

4.4 Контрольні питання

1. У чому полягає сутність завдання екстраполяції функцій?
2. Що включає структура мережі для екстраполяції сигналів?
3. Як задається довжина вектора навчальної послідовності під час вирішення завдання екстраполяції сигналу?
4. Як перевіряється якість роботи нейронної мережі під час передбачення значень стаціонарного сигналу?

5 РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ МОДЕЛИРОВАНИЯ СТАЦИОНАРНОГО ФИЛЬТРА

Мета роботи – придбати вміння та навички у створенні та навчанні нейронних мереж для моделювання стаціонарних фільтрів.

5.1 Постановка задачі та синтез мережі для моделювання стаціонарного фільтра

Завдання полягає у створенні мережі, яка в процесі навчання визначає параметри фільтра, а потім як модель фільтра використовується для аналізу вихідних сигналів при довільних значеннях вхідних сигналів. Перша частина цього завдання відома як завдання ідентифікації.

Розглянемо лінійний стаціонарний фільтр, що описується рекурентним виразом:

$$y[n] = -0.5y[n-1] + 1.5y[n-2] + r[n].$$

Цей цифровий фільтр другого порядку може бути представлений дискретною функцією передачі:

$$y[n] = \frac{1}{1 + 0.5z^{-1} - 1.5z^{-2}} \cdot r[n].$$

В MATLAB такий фільтр описується М-функцією:

$$Y = \text{filter}([1 \ 0.5 \ -1.5], 1, R);$$

Допустимо, що на вхід фільтра подається сигнал: $r(t) = \sin(10 * \sin(t) * t)$.

Задаємо сигнал масивом значень з періодом дискретності 0,025 на інтервалі 5 с. Вхідну послідовність **P** задаємо на основі поточного та двох попередніх значень входу **X**.

```
time = 0:0.025:5;
```

```
X = sin(10*sin(time).*time);
```

```
Q=size(X,2);
```

```
P = zeros(3,Q);
```

```
P(1,1:Q) = X(1,1:Q);
```

```
P(2,2:Q) = X(1,1:(Q-1));
```

```
P(3,3:Q) = X(1,1:(Q-2));
```

```
T = filter([1 0.5 -1.5],1,X);
```

```
%Виведемо графіки вхідного та вихідного сигналів фільтра
```

```
figure(1)
```

```

subplot(2,1,1)
plot(time,X)
axis([0 5 -2 2]);
xlabel('Time');
ylabel('Input Signal');
grid;
title('Input Signal to the System');
subplot(2,1,2)
plot(time,T)
axis([0 5 -2 2]);
xlabel('Time');
ylabel('Output Signal');
title('Output Signal of the System');
grid;

```

Графік вихідного сигналу фільтра наведено на рисунку 5.1

Завдання нейронної мережі – визначити у процесі навчання параметри фільтра, та був використовувати їх задля моделювання при довільних значеннях входу.

Оскільки фільтр має один вихід, то нейронна мережа має складатися з одного нейрона. Щоб отримати та обробити одне поточне та два значення запізнення, нейрон повинен мати три входи.

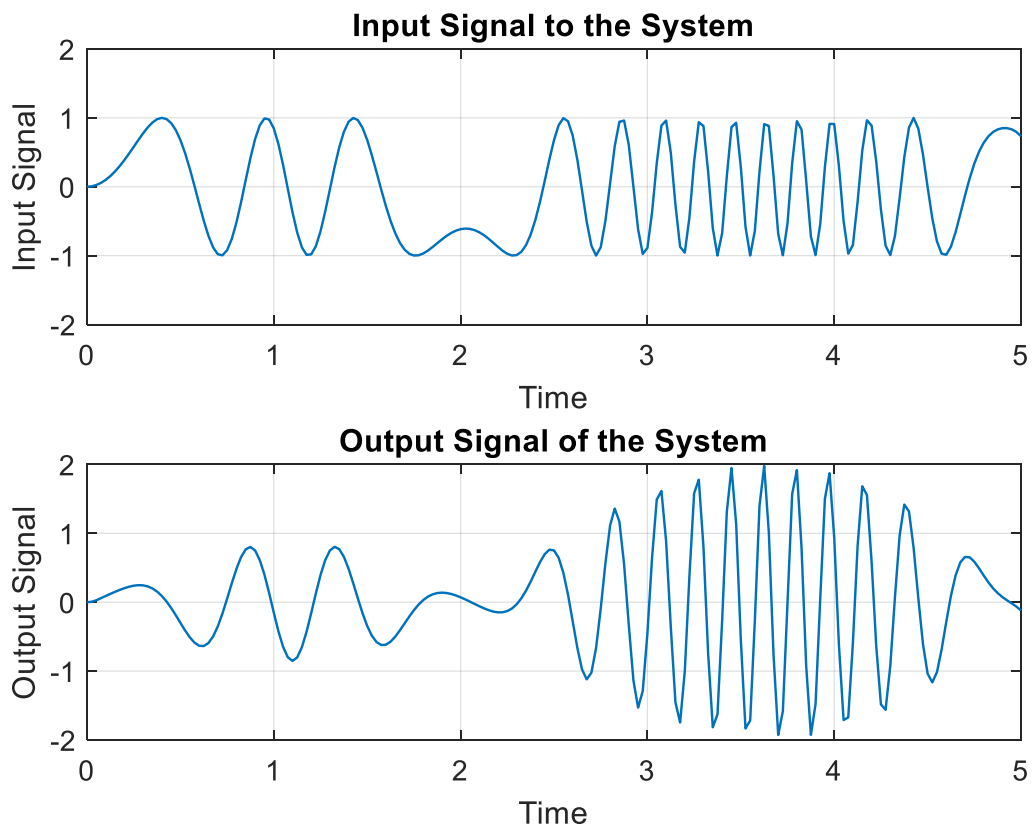


Рисунок 5.1 – Графіки вхідного та вихідного сигналів фільтра при моделюванні фільтра засобами MATLAB

Як цільовий вектор \mathbf{T} приймаємо масив виходу \mathbf{Y} , а вхідний послідовності \mathbf{P} зіставимо поточне значення та два наступних значення входів \mathbf{R} . Щоб отримати два значення, що запізнюються, введемо М-функцію затримки `newlind`.

Структура мережі для моделювання фільтра представлена на рисунку 5.2.

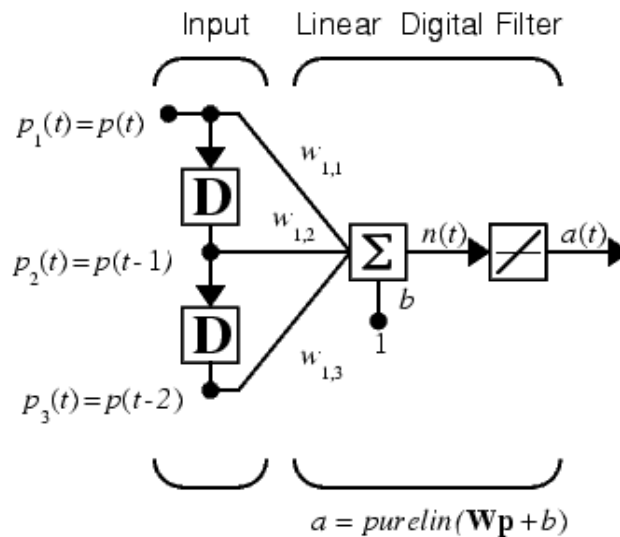


Рисунок 5.2 – Структура мережі для моделювання фільтра другого порядку

Введемо сигнали навчання та зробимо налаштування мережі:

```
R=X;
Q=size(R,2);
P=zeros(3,Q);
P(1,1:Q)=R(1,1:Q);
P(2,2:Q)=R(1,1:(Q-1));
P(3,3:Q)=R(1,1:(Q-2));
net=newlind(P,T);
net.IW{1}, net.b;
```

```
ans =
    1.0000    0.5000   -1.5000
ans =
    1×1 cell array
    {[-1.4869e-17]}
```

Для перевірки функціонування мережі порівняємо вихід \mathbf{a} із цільовою послідовністю \mathbf{T} :

```
a=sim(net,P);
figure(2),clf
```

```

subplot(2,1,1)
plot(time,T,'ok'),hold on
plot(time,a,'k'), grid on
axis([0 5 -2 2]);
ylabel('a- вихід НМ, T - цільова ф-ція');
% Тут же обчислимо та побудуємо графік похибки моделювання:
subplot(2,1,2)
e=T-a;
plot(time,e)
xlabel('Time, c')
ylabel('Помилка')
grid

```

Результати моделювання представлені на рисунку 5.3.

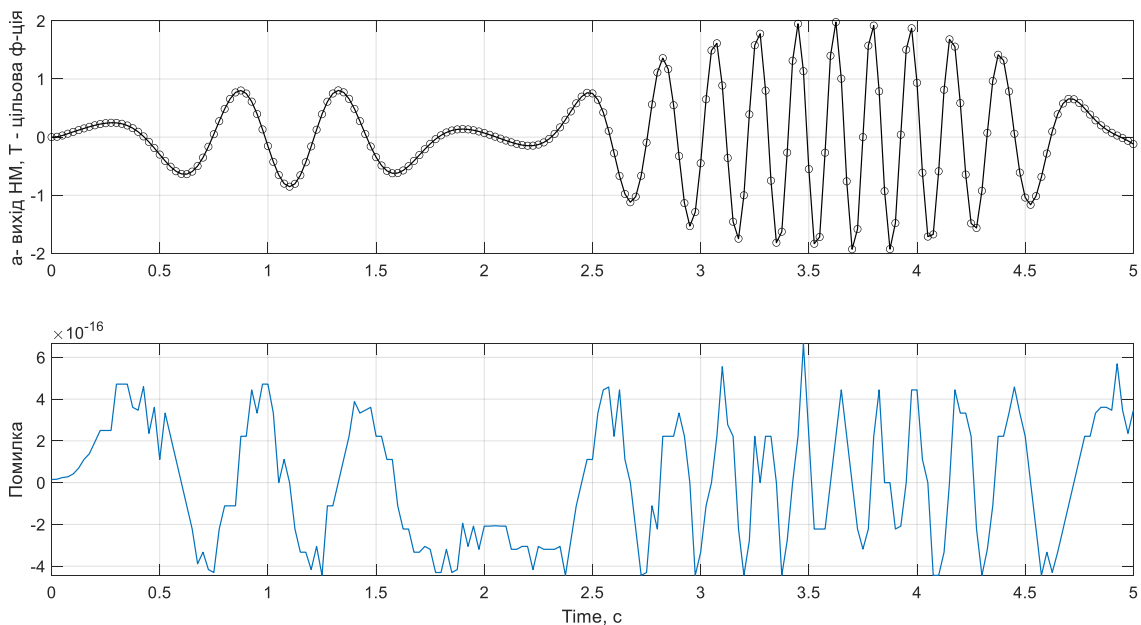


Рисунок 5.3 – Результати моделювання фільтра мережею

Результати моделювання показують, що мережа забезпечує вирішення поставленої задачі з високою точністю. Похибка моделювання знаходиться в межах точності обчислень (10^{-15}).

5.2 Методика виконання роботи

Для виконання роботи студенти отримують індивідуальні завдання, варіанти яких наведені у таблиці 5.1.

При вирішенні задачі моделювання стаціонарного фільтра необхідно виконати наступне:

1. Здійснити моделювання фільтра засобами MATLAB.
2. Зробити синтез нейронної мережі для моделювання фільтра.

3. Здійснити навчання та моделювання мережі.
4. Вивести графіки результатів моделювання та оцінити якість роботи мережі.

Таблиця 5.1 – Варіанти завдань для моделювання фільтра

№ вар.	Функція вхідного сигналу	№ вар.	Функція вхідного сигналу
1	$r(t) = \sin(2 * \sin(t) * t)$	11	$r(t) = \sin(2 * \sin(t) * t)$
2	$r(t) = \sin(3 * \sin(t) * t)$	12	$r(t) = \sin(3 * \sin(t) * t)$
3	$r(t) = \sin(4 * \sin(t) * t)$	13	$r(t) = \sin(4 * \sin(t) * t)$
4	$r(t) = \sin(5 * \sin(t) * t)$	14	$r(t) = \sin(5 * \sin(t) * t)$
5	$r(t) = \sin(6 * \sin(t) * t)$	15	$r(t) = \sin(6 * \sin(t) * t)$
6	$r(t) = \sin(7 * \sin(t) * t)$	16	$r(t) = \sin(7 * \sin(t) * t)$
7	$r(t) = \sin(8 * \sin(t) * t)$	17	$r(t) = \sin(8 * \sin(t) * t)$
8	$r(t) = \sin(9 * \sin(t) * t)$	18	$r(t) = \sin(9 * \sin(t) * t)$
9	$r(t) = \sin(10 * \sin(t) * t)$	19	$r(t) = \sin(10 * \sin(t) * t)$
10	$r(t) = \sin(12 * \sin(t) * t)$	20	$r(t) = \sin(12 * \sin(t) * t)$
$y[n] = \frac{1}{1 + 0.5z^{-1} - 1.5z^{-2}} \cdot r[n]$		$y[n] = \frac{1}{1 + 1.5z^{-1} - 3.5z^{-2}} \cdot r[n]$	

5.3 Зміст і захист звіту

Звіт по роботі повинен містити завдання, структуру мережі, програми та результати роботи мережі.

При захисті звіту студент повинен відповісти на питання щодо методики створення та застосування нейронної мережі для моделювання стаціонарного фільтра.

5.4 Контрольні питання

1. У чому полягає суть завдання моделювання стаціонарного фільтра?
2. Як надаються параметри цифрового фільтра?
3. Що включає структура нейронної мережі для моделювання стаціонарного фільтра?

6 СИНТЕЗ НЕЛІНІЙНОГО НЕЧІТКОГО РЕГУЛЯТОРА ПД-ТИПУ

Мета роботи: засвоєння методики синтезу нелінійного нечіткого регулятора ПД-типу з використанням редактора системи нечіткого висновку

6.1 Аналітичний опис нечіткого логічного регулятора ПД-типу

Запишемо закон управління ПД-регулятора у вигляді:

$$u = k_p \left(e(t) + \frac{k_d}{k_p} \frac{de(t)}{dt} \right)$$

Тоді класичний ПД-регулятор можна зобразити у вигляді показаному на рис. 6.1.

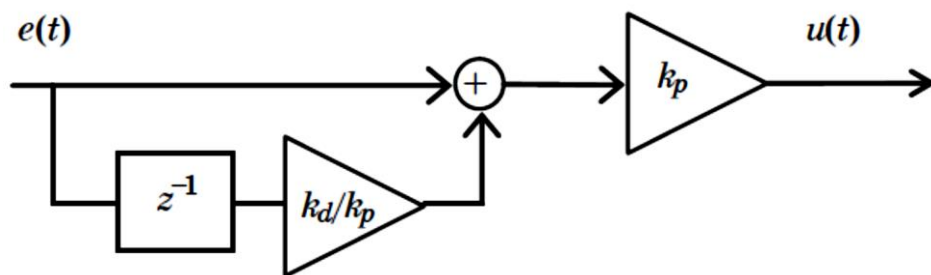


Рисунок 6.1 – Варіант опису ПД-регулятора

Використання похідної дозволяє зменшити перерегулювання, що може виникати при пропорційному управлінні. Однак при цьому можливе небажане збільшення часу перехідного процесу. Методика проектування має узгодити ці два суперечливі ефекти. Тому процес налаштування k_d та k_p передбачає такі кроки:

- 1) коефіцієнт k_d отримує по можливості невелике значення для досягнення найбільш швидкої реакції системи;
- 2) пропорційний коефіцієнт k_p повинен бути завжди більшим, ніж k_d .

6.2 Синтез нечіткого регулятора ПД-типу

НЛР_ПД відрізняється від класичного ПД-регулятора тим, що може реалізувати нелінійний закон управління, який описуватиметься деякою поверхнею в тривимірному просторі.

При цифровій реалізації НЛР_ПД замість похідної помилки управління часто розглядається її збільшення $\Delta e(t)$, так що поведінку НЛР_ПД можна описати виразом

$$u(t) = F_{\pi}(e(t), \Delta e(t)),$$

де F_{π} - лінгвістичний закон управління, заданий набором правил.

За певних умов НЛР_ПД може бути еквівалентний звичайному ПД-регулятору. Для забезпечення вищої якості роботи НЛР_ПД по відношенню до класичного регулятора потрібно правильно описати керуючі правила та правильно вибрати розташування термів лінгвістичних змінних.

Управляючі правила можна вибрати на основі якісного аналізу перехідного процесу.

Помилка управління обчислюється за формулою

$$e(t) = g(t) - y(t).$$

Негативне значення помилки $e(t)$ означає, що вихідний сигнал $y(t)$ більше уставки $g(t)$, і його потрібно зменшувати, позитивне значення $e(t)$ означає, що вихідний сигнал менший за уставку, і його потрібно збільшувати. Нульове значення $e(t)$ відповідає, очевидно, рівності $g(t)$ та $y(t)$.

Розглянемо збільшення помилки управління:

$$\begin{aligned} \Delta e(t) &= e(t) - e(t-1) = g(t) - y(t) - g(t) + y(t-1) = y(t-1) - y(t) \\ &= -\Delta y(t) \end{aligned}$$

Позитивний знак $\Delta e(t)$ означає, що вихідний сигнал зменшується. Негативне значення $\Delta e(t)$ означає, що вихідний сигнал збільшується. Нульове значення $\Delta e(t)$ відповідає постійному вихідному сигналу.

На рис. 6.2 показано поведінку помилки та збільшення помилки при нормальному перебігу перехідного процесу. Як показує рисунок, наближення вихідного сигналу системи до заданого значення відбувається тоді, коли $e(t)$ та $\Delta e(t)$ мають різні знаки. У цьому випадку керування на вхід об'єкта можна не подавати. В інших ситуаціях потрібна корекція.

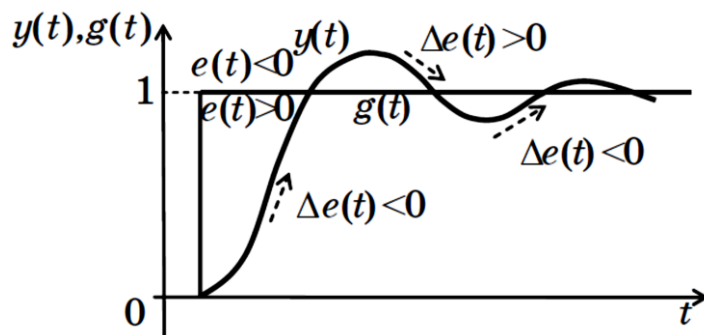


Рисунок 6.2 – Зміна помилки при нормальному перебігу перехідного процесу

Зроблені зауваження дозволяють класифікувати всі, що виникають при управлінні ситуації (див. табл. 6.1).

Таблиця 6.1 – Семантичний опис процесу управління

Ситуація		Опис ситуації	Характеристика ситуації	Управління
Знак $e(t)$	Знак $\Delta e(t)$			
-	+	$y(t) > g(t), \Delta y(t) < 0$	Норма	Нульове (0)
-	-	$y(t) > g(t), \Delta y(t) > 0$	Потрібна корекція	Негативне (-)
-	0	$y(t) > g(t), \Delta y(t) = 0$	Потрібна корекція	Негативне (-)
0	-	$y(t) = g(t), \Delta y(t) > 0$	Потрібна корекція	Негативне (-)
0	+	$y(t) = g(t), \Delta y(t) < 0$	Потрібна корекція	Позитивне (+)
0	0	$y(t) = g(t), \Delta y(t) = 0$	Норма	Нульове (0)
+	+	$y(t) < g(t), \Delta y(t) < 0$	Потрібна корекція	Позитивне (+)
+	-	$y(t) < g(t), \Delta y(t) > 0$	Норма	Нульове (0)
+	0	$y(t) < g(t), \Delta y(t) = 0$	Потрібна корекція	Позитивне (+)

У табл. 6.1 розглядаються лише знаки помилки управління та її збільшення, тому вважатимуться, що кількість термів для лінгвістичних змінних на входах і виході НЛР_ПД дорівнює 3, і закон управління можна сформулювати як таблиці лінгвістичних правил (ТЛП) (див. табл.6.2).

Таблиця 6.2 – Лінгвістичні правила НЛР_ПД

Таблиця правил		e^*			
		Н	0	П	
Δe^*	Н	Н	Н	0	u*
	0	Н	0	П	
	П	0	П	П	

Табл. 6.2 містить 9 правил управління (лінгвістичне значення управління знаходиться на перетині рядка та стовпця). Ці правила мають універсальний характер, і можуть бути використані для будь-якого об'єкта управління невисокого порядку.

Відповідно до (9.4), для П-регулятора знак управління збігається зі знаком помилки. Наприклад, при 5 термах справедлива табл. 6.3.



Таблиця. 6.3 – Лінгвістичний закон управління П-регулятора

e^*	НВ	НМ	0	ПМ	ПВ
u^*	НВ	НМ	Н	ПМ	ПВ

Входи та виходи ПД-регулятора нормалізовані, тому виконується:

$$u^* = e^* + \Delta e^*.$$

Припустимо, що u^* , e^* та Δe^* мають однакові множини термів, наприклад: НВ, НС, НМ, 0, ПМ, ПС, ПВ. Тоді, враховуючи обмежені розміри базової шкали, можна записати 49 варіантів обмеженої суми:

НВ+НВ=НВ; НВ+НС=НВ; НВ+НМ=НВ; НВ+0=НВ;
 НВ+ПМ=НС; НВ+ПС=НМ; НВ+ПВ=0;
 НС+НВ=НВ; НС+НС=НВ; НС+НМ=НВ; НС+0=НС;
 НС+ПМ=НМ; НС+ПС=0; НС+ПВ=ПМ;
 ...
 ПВ+НВ=0; ПВ+НС=ПМ; ПВ+НМ=ПС; ПВ+0=ПВ;
 ПВ+ПМ=ПВ; ПВ+ПС=ПВ; ПВ+ПВ=ПВ

Лінгвістичний закон управління набуває вигляду, показано у табл. 6.4. Табл. 6.4 можна модифікувати за допомогою додаткових евристичних міркувань. Наприклад, можна вимагати, щоб при великому значенні помилки сигнал керування також був великим незалежно від значення збільшення помилки (див. табл. 6.5).

Таблиця 6.4 – Табличні лінгвістичні правила

Таблиця правил		e^*						
		НВ	НС	НМ	0	ПМ	ПС	ПВ
Δe^*	НВ	НВ	НВ	НВ	НВ	НС	НМ	0
	НС	НВ	НВ	НВ	НС	НМ	0	ПМ
	НМ	НВ	НВ	НС	НМ	0	ПМ	ПС
	Н	НВ	НС	НМ	0	ПМ	ПС	ПВ
	ПМ	НС	НМ	0	ПМ	ПС	ПВ	ПВ
	ПС	НМ	0	ПМ	ПС	ПВ	ПВ	ПВ
	ПВ	0	ПМ	ПС	ПВ	ПВ	ПВ	ПВ



Таблиця 6.5 – Модифіковані табличні лінгвістичні правила

Таблиця правил		e*						
		НВ	НС	НМ	0	ПМ	ПС	ПВ
Δe*	НВ	НВ	НВ	НВ	НВ	НС	НМ	ПВ
	НС	НВ	НВ	НВ	НС	НМ	ПМ	ПВ
	НМ	НВ	НВ	НС	НМ	0	ПС	ПВ
	Н	НВ	НС	НМ	0	ПМ	ПС	ПВ
	ПМ	НВ	НС	0	ПМ	ПС	ПВ	ПВ
	ПС	НВ	НМ	ПМ	ПС	ПВ	ПВ	ПВ
	ПВ	НВ	Н	ПС	ПВ	ПВ	ПВ	ПВ

Використання модифікованої таблиці лінгвістичних правил має зменшити час перехідного процесу у системі.

Евристичний закон керування, заданий табл. 6.4 та 6.5, є універсальним для об'єктів з одним входом та одним виходом. Налаштування закону управління для конкретного об'єкта передбачає вибір коефіцієнтів нормалізації та денормалізації, а також центрів термів лінгвістичних змінних на відповідних базових шкалах.

6.3 Приклад синтезу нечіткого регулятора ПД-типу

Нехай як базова використовується структура, показана на рис. 9.15. При заміні регулятора на НЛР_ПД, вона набуває вигляду, показаного на рис. 6.3.

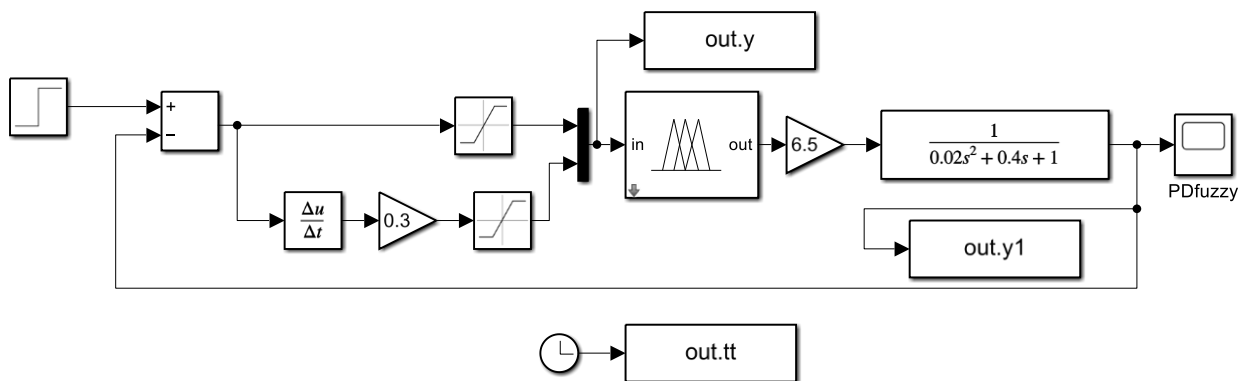


Рисунок 6.3 – Математична модель системи управління з НЛР_ПД

Вирішимо дане завдання згідно описаного алгоритму з використанням редактора системи нечіткого висновку.

Будуємо у MatLab Simulink математичну модель об'єкту керування з послідовним включення Fuzzy Logic Controller до класичного регулятора (див. рис. 11.71). Fuzzy Logic Controller надаємо Fis name: PDfis7 (ім'я може бути будь-яке).

Викликається редактор системи нечіткого висновку (Fuzzy Inference System – FIS) у MatLab командою

```
>> fuzzy
```

Дале проводиться налаштування головного вікна редактора нечіткої логічної системи (див. рис. 6.4.).

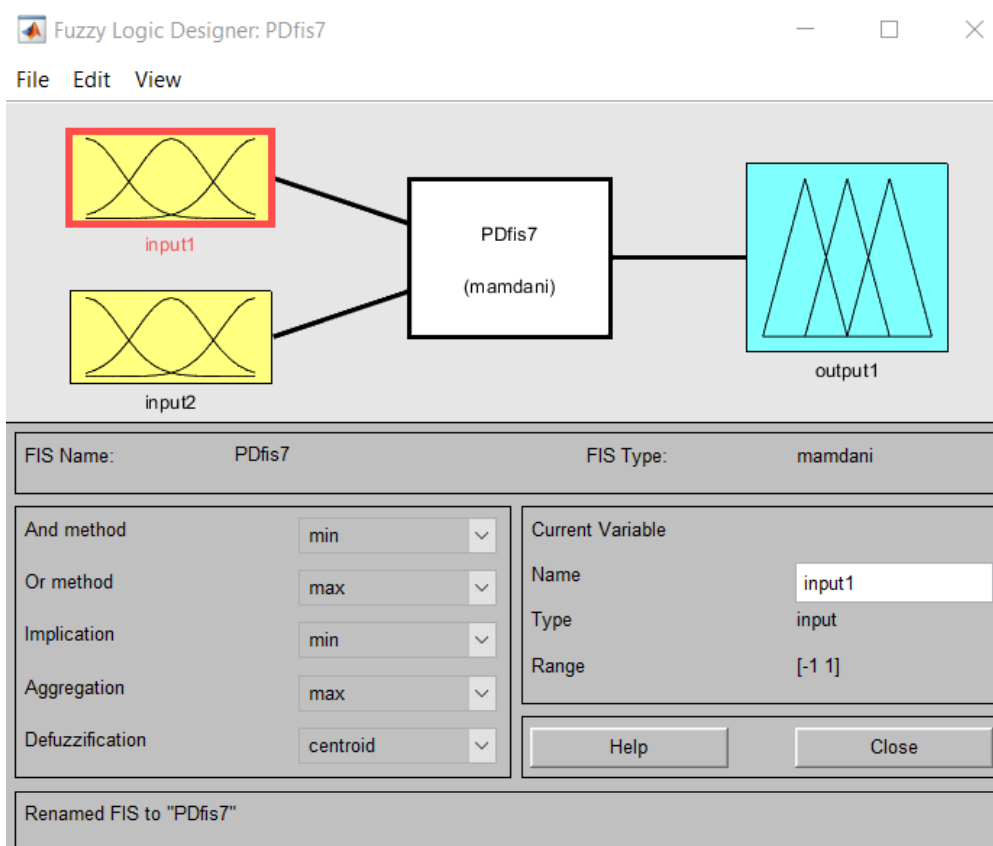


Рисунок 6.4 – Налаштування інтерфейсу FIS editor

У FIS editor задається опис систему нечіткого логічного висновку Mamdani. Для створюваної системи обирається вид логічного зв'язку (*And method – min*) та (*Or method – max*), вид імплікації (*Implication – min*), спосіб агрегування висновків правил (*Aggregation – max*) та метод дефазифікації (*Defuzzification – centroid*).

У меню *Edit* послідовно додаємо дві вхідних змінних з розміром базової шкали (*Range= [-1 1]*) та вихідну змінну з розміром базової шкали (*Range= [-1 1]*).

Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної змінної

трикутну функцію приналежності.

Нехай терми лінгвістичної змінної «Помилка управління» розміщуються відповідно до таблиці 6.6 згідно налаштуванням НЛР_П регулятора з параметрами:

Таблиця 6.6 – Закон управління проектованим НЛР_П

	НВ (негативне велике)	НС (негативне середнє)	НМ (негативне мале)	Н (нульове)	ПМ (позитивне мале)	ПС (позитивне середнє)	ПВ (позитивне велике)
<i>e</i>	-1	-0,33	-0,033	0	0,033	0,33	1
<i>u</i>	-1	-0,66	0,33	0	0,33	0,66	1

Name = "NB"; Type = "trimf"; Param = [-1.33 -1 -0.33];

Name = "NM"; Type = "trimf"; Param = [-1 -0.33 -0.03];

Name = "NS"; Type = "trimf"; Param = [-0.33 -0.033 0];

Name = "Z"; Type = "trimf"; Param = [-0.1 0 0.1];

Name = "PS"; Type = "trimf"; Param = [0 0.033 0.33];

Name = "PM"; Type = "trimf"; Param = [0.033 0.33 1];

Name = "PB"; Type = "trimf"; Param = [0.33 1 1.33].

Результат налаштування функцій приналежності вхідних змінних «Помилка управління» наведено на рис. 6.5.

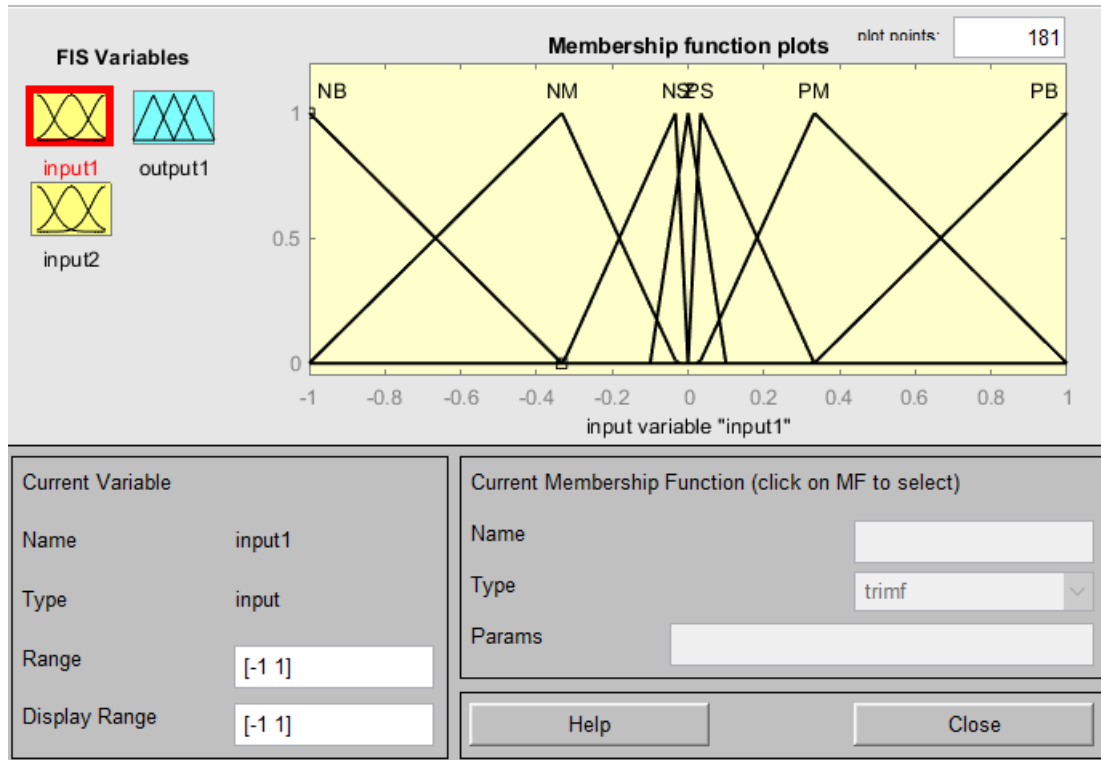


Рисунок 6.5 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Терми вхідної лінгвістичної змінної «Похідна помилки управління» формуємо рівномірно. Для опису вхідних логічних змінних «Похідна помилки управління» у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. З параметрами:

Name = "NBe"; Type = "trimf"; Param = [-1.33 -1 -0.66];
 Name = "NMe"; Type = "trimf"; Param = [-1 -0.66 -0.33];
 Name = "NSe"; Type = "trimf"; Param = [-0.66 -0.33 0];
 Name = "Ze"; Type = "trimf"; Param = [-0.33 0 0.33];
 Name = "PSe"; Type = "trimf"; Param = [0 0.33 0.66];
 Name = "PMe"; Type = "trimf"; Param = [0.33 0.66 1];
 Name = "PBe"; Type = "trimf"; Param = [0.66 1 1.33].

Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління» на рис. 6.6.

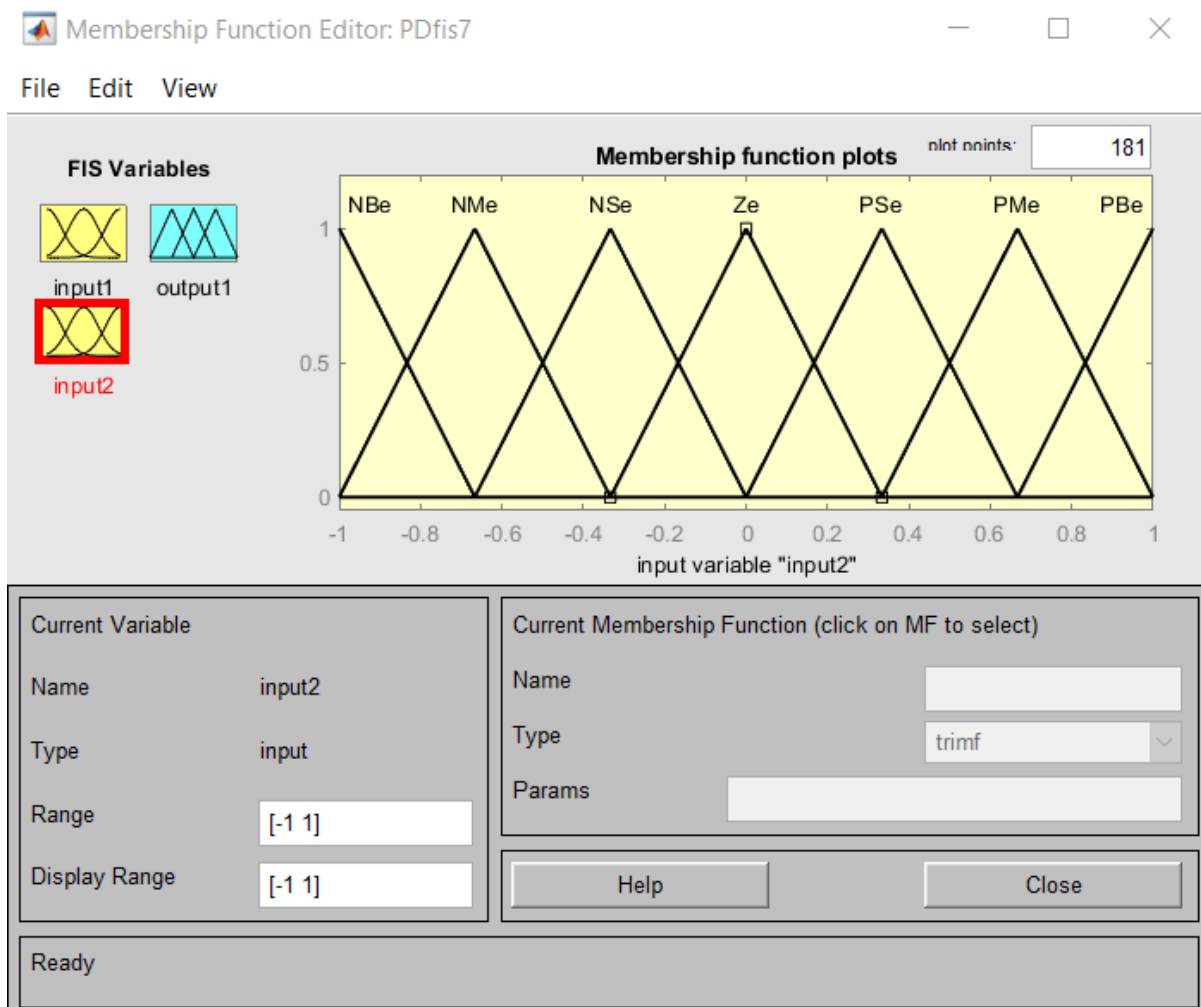


Рисунок 6.6 – Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління»

Для опису вихідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. З параметрами:

Name = "NBu"; Type = "trimf"; Param = [-1.33 -1 -0.66];
 Name = "NMu"; Type = "trimf"; Param = [-1 -0.66 -0.33];
 Name = "NSu"; Type = "trimf"; Param = [-0.66 -0.33 0];
 Name = "Zu"; Type = "trimf"; Param = [-0.33 0 0.33];
 Name = "PSu"; Type = "trimf"; Param = [0 0.33 0.66];
 Name = "PMu"; Type = "trimf"; Param = [0.33 0.66 1];
 Name = "PBu"; Type = "trimf"; Param = [0.66 1 1.33].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 6.7.

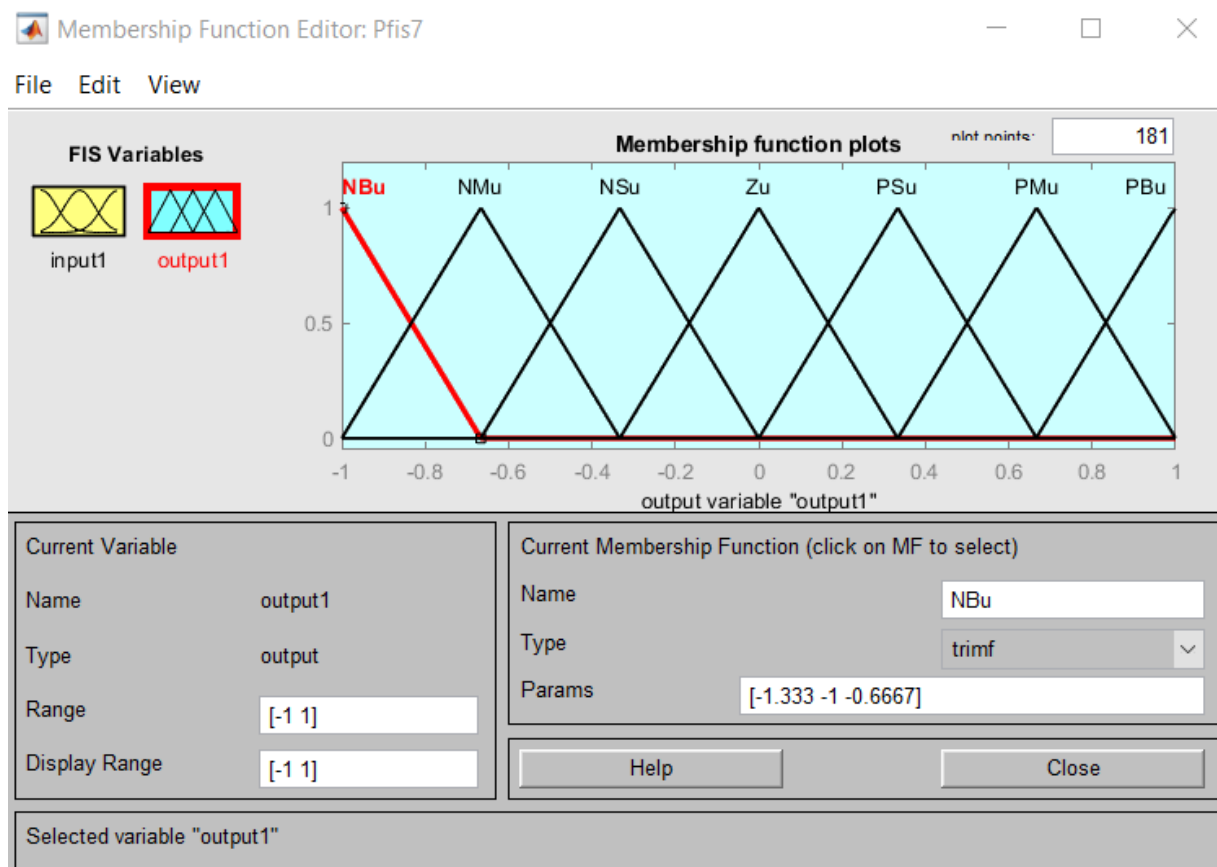



Рисунок 6.7 – Результат налаштування функцій приналежності вихідних змінних

Після опису вхідні та вихідні лінгвістичні змінні, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку дорівнює 49. Управляючі правила сформуємо згідно з табл. 9.5:

1. if (input1 is NB) and (input2 is NBe) then (output1 is NBu) (1)

- 
2. if (input1 is NB) and (input2 is NMe) then (output1 is NBu) (1)
 3. if (input1 is NB) and (input2 is NSe) then (output1 is NBu) (1)
 4. if (input1 is NB) and (input2 is Ze) then (output1 is NBu) (1)
 5. if (input1 is NB) and (input2 is PSe) then (output1 is NBu) (1)
 6. if (input1 is NB) and (input2 is PMe) then (output1 is NBu) (1)
 7. if (input1 is NB) and (input2 is PBe) then (output1 is NBu) (1)
 8. if (input1 is NM) and (input2 is NBe) then (output1 is NBu) (1)
 9. if (input1 is NM) and (input2 is NMe) then (output1 is NBu) (1)
 10. if (input1 is NM) and (input2 is NSe) then (output1 is NBu) (1)
 11. if (input1 is NM) and (input2 is Ze) then (output1 is NMu) (1)
 12. if (input1 is NM) and (input2 is PSe) then (output1 is NMu) (1)
 13. if (input1 is NM) and (input2 is PMe) then (output1 is NSu) (1)
 14. if (input1 is NM) and (input2 is PBe) then (output1 is Zu) (1)
 15. if (input1 is NS) and (input2 is NBe) then (output1 is NBu) (1)
 16. if (input1 is NS) and (input2 is NMe) then (output1 is NBu) (1)
 17. if (input1 is NS) and (input2 is NSe) then (output1 is NMu) (1)
 18. if (input1 is NS) and (input2 is Ze) then (output1 is NSu) (1)
 19. if (input1 is NS) and (input2 is PSe) then (output1 is Zu) (1)
 20. if (input1 is NS) and (input2 is PMe) then (output1 is PSu) (1)
 21. if (input1 is NS) and (input2 is PBe) then (output1 is PMu) (1)
 22. if (input1 is Z) and (input2 is NBe) then (output1 is NBu) (1)
 23. if (input1 is Z) and (input2 is NMe) then (output1 is NMu) (1)
 24. if (input1 is Z) and (input2 is NSe) then (output1 is NSu) (1)
 25. if (input1 is Z) and (input2 is Ze) then (output1 is Zu) (1)
 26. if (input1 is Z) and (input2 is PSe) then (output1 is PSu) (1)
 27. if (input1 is Z) and (input2 is PMe) then (output1 is PMu) (1)
 28. if (input1 is Z) and (input2 is PBe) then (output1 is PBu) (1)
 29. if (input1 is PS) and (input2 is NBe) then (output1 is NMu) (1)
 30. if (input1 is PS) and (input2 is NMe) then (output1 is NSu) (1)
 31. if (input1 is PS) and (input2 is NSe) then (output1 is Zu) (1)
 32. if (input1 is PS) and (input2 is Ze) then (output1 is PSu) (1)
 33. if (input1 is PS) and (input2 is PSe) then (output1 is PMu) (1)
 34. if (input1 is PS) and (input2 is PMe) then (output1 is PBu) (1)
 35. if (input1 is PS) and (input2 is PBe) then (output1 is PBu) (1)
 36. if (input1 is PM) and (input2 is NBe) then (output1 is NMu) (1)
 37. if (input1 is PM) and (input2 is NMe) then (output1 is PSu) (1)
 38. if (input1 is PM) and (input2 is NSe) then (output1 is PMu) (1)
 39. if (input1 is PM) and (input2 is Ze) then (output1 is PMu) (1)
 40. if (input1 is PM) and (input2 is PSe) then (output1 is PBu) (1)
 41. if (input1 is PM) and (input2 is PMe) then (output1 is PBu) (1)
 42. if (input1 is PM) and (input2 is PBe) then (output1 is PBu) (1)
 43. if (input1 is PB) and (input2 is NBe) then (output1 is PBu) (1)
 44. if (input1 is PB) and (input2 is NMe) then (output1 is PBu) (1)
 45. if (input1 is PB) and (input2 is NSe) then (output1 is PBu) (1)
 46. if (input1 is PB) and (input2 is Ze) then (output1 is PBu) (1)

- 47. if (input1 is PB) and (input2 is PSe) then (output1 is PBU) (1)
- 48. if (input1 is PB) and (input2 is PMe) then (output1 is PBU) (1)
- 49. if (input1 is PB) and (input2 is PBe) then (output1 is PBU) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 6.8).

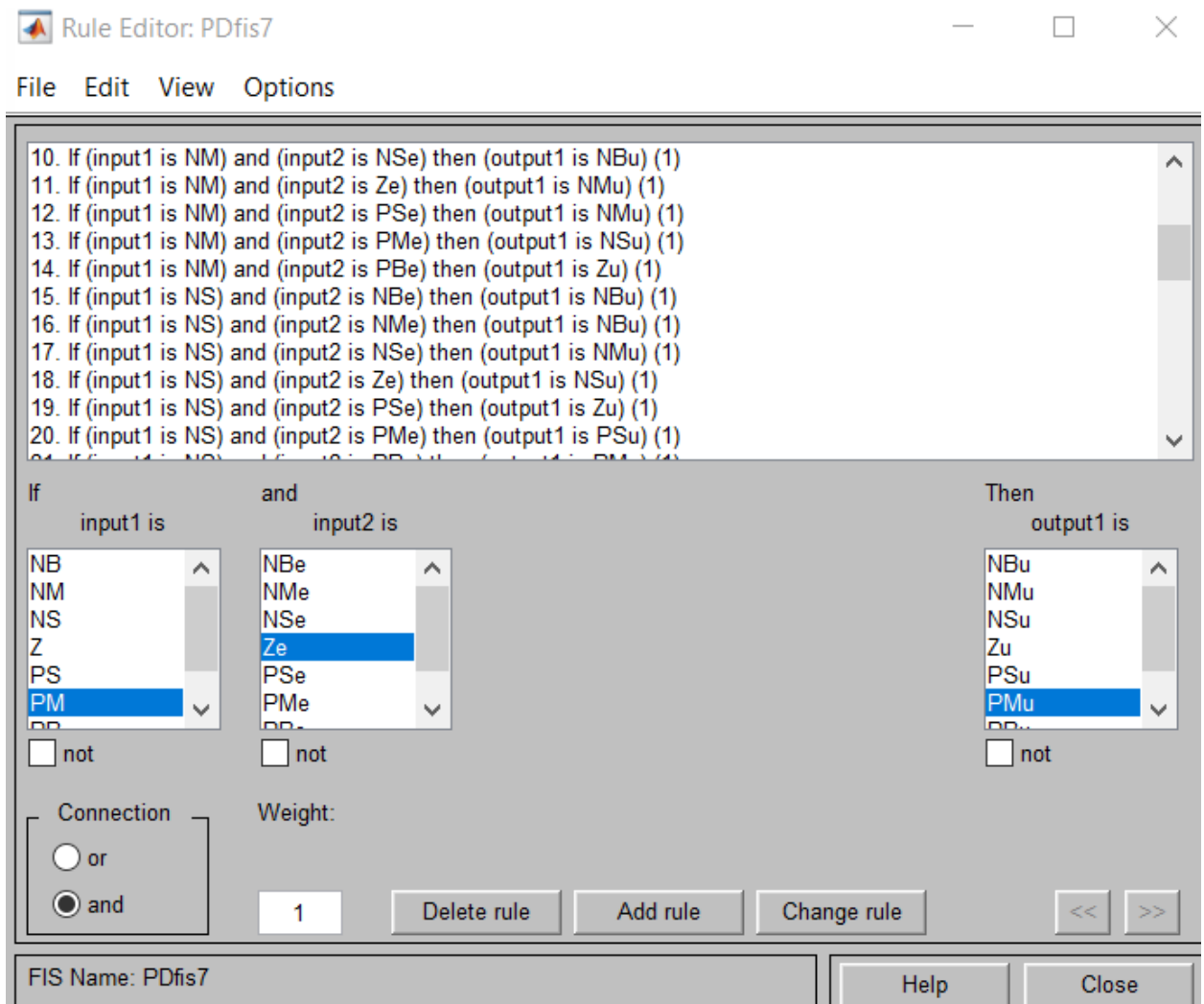


Рисунок 6.8 - Налаштовування опису правил функціонування НЛР_ПД

Посилки у правилах пов'язані (connection) за допомогою операції *and*. Введене правило забезпечене ваговим коефіцієнтом (*Weight=1*).

Після опису правил можливо за допомогою - інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 6.9);

- інтерфейсу *Surface Viewer* можливо переглянути графічне подання закону управління (рис. 6.10).

Після процедур налаштування НЛР_ПД у редакторі системи нечіткого висновку здійснює запис результатів налаштування у Fuzzy Logic Controller. Для цього у вкладці *File* здійснюємо *Export* в

математичну модуль нечіткого контролера From Workspase вказавши ім'я Fuzzy Logic Controller.

Після експорту закону функціонування НЛР_ПД на математичній моделі системи управління (див. рис. 6.3) получено перехідний процес для синтезованого НЛР_ПД, який задовольняє поставленому завданню проектування

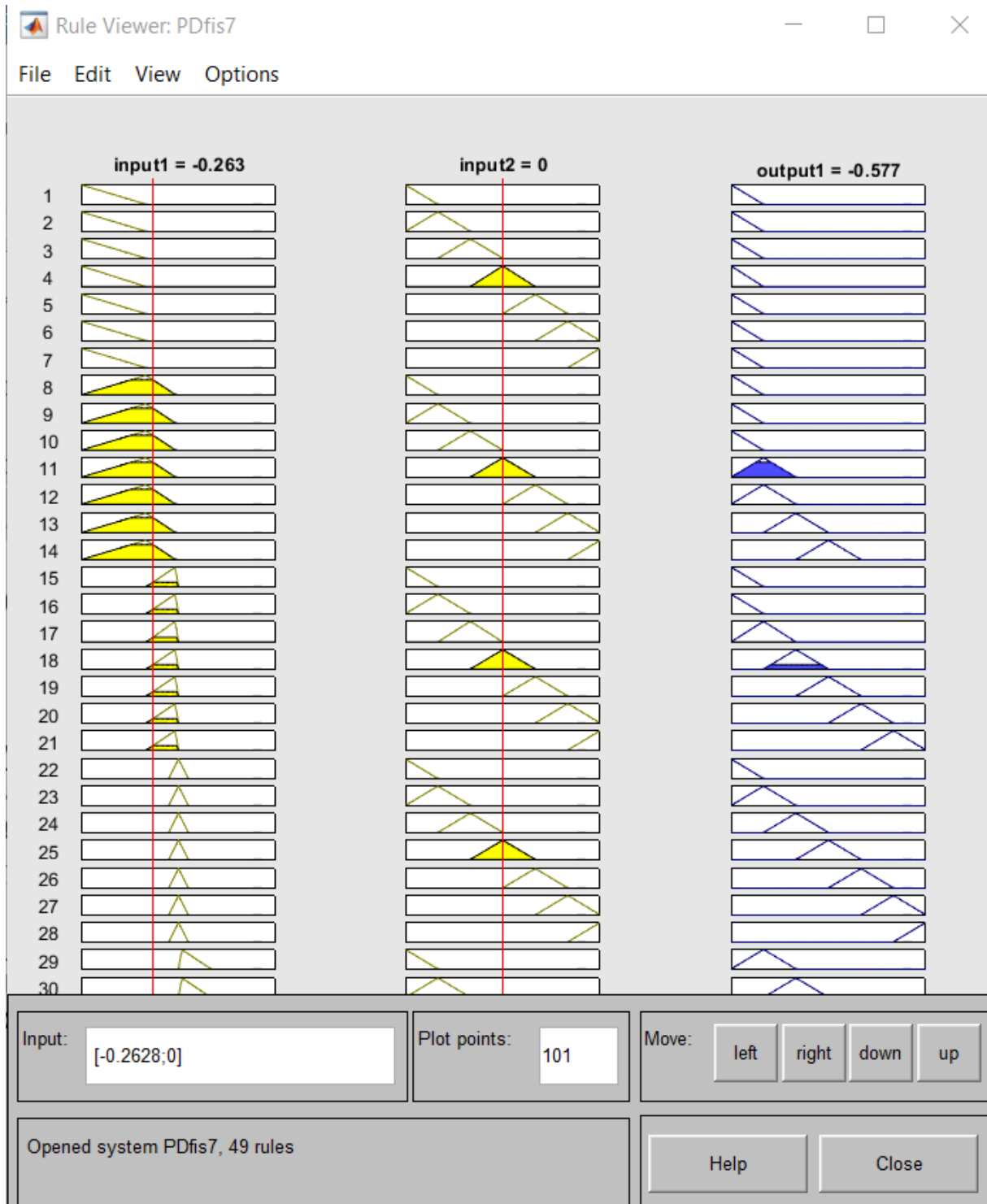


Рисунок 6.9 - Робота системи НЛР_ПД

На рис. 6.11 показаний перехідний процес для синтезованого НЛР_ПД. Як свідчить рис. 6.11, перерегулювання зведено до мінімуму.

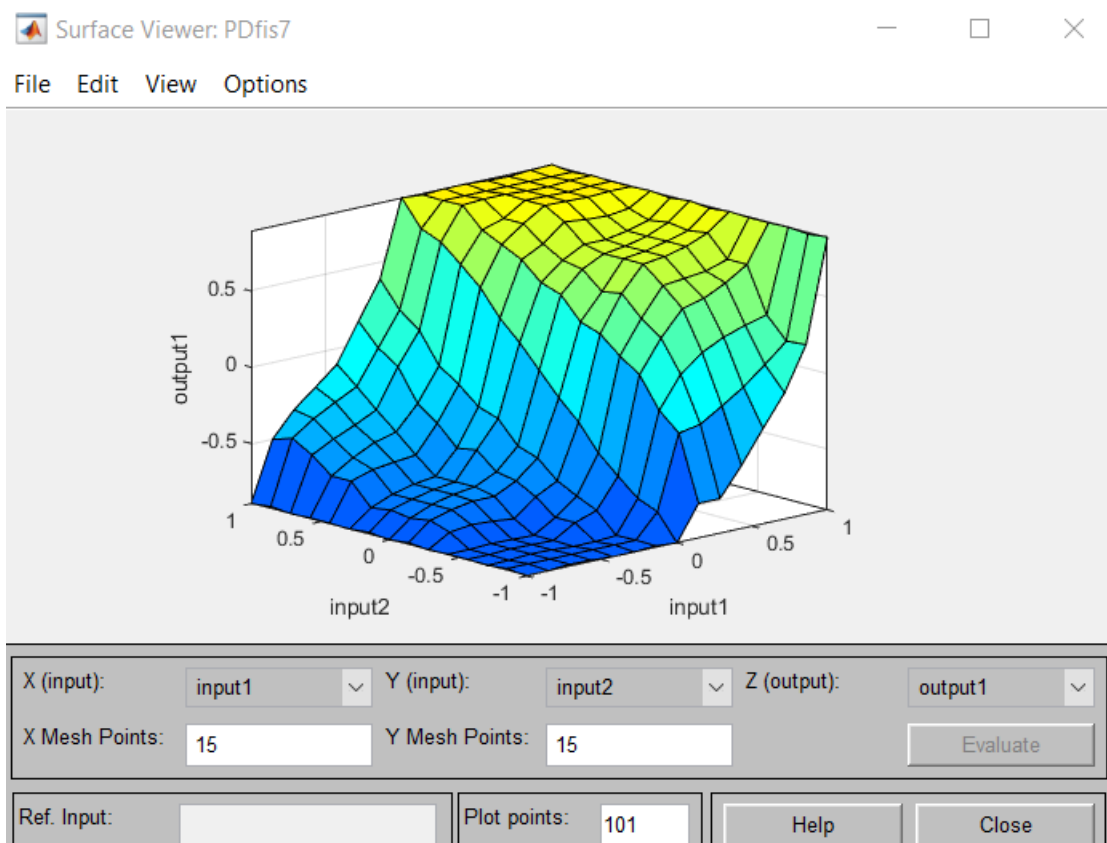


Рисунок 6.10 – Поверхня управління системою нечіткого висновку

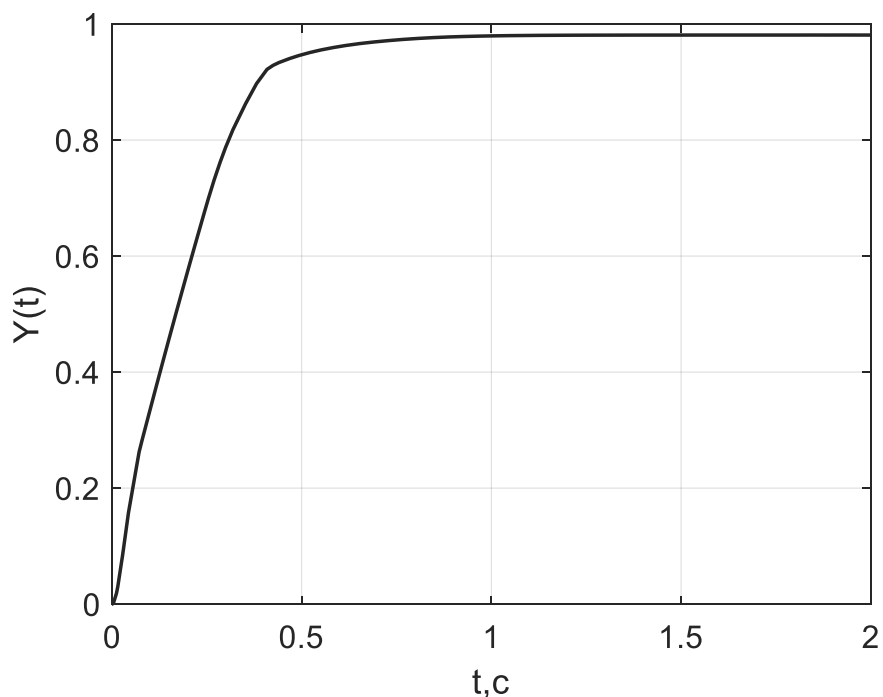


Рисунок 6.11 - Перехідний процес для НЛР_ПД із сімома правилами

Таким чином, вихідними даними для синтезу НЛР_ПД є опис

об'єкта управління та вимоги до перехідного процесу: час наростання, перерегулювання, помилка, що встановилася.

6.4 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 6.6.

За вказаними у табл. 6.7 потрібно синтезувати нелінійний нечіткий регулятор ПД-типу з використанням системи нечіткого висновку:

Таблиця 6.7 - Варіанти індивідуальних завдань

№ вар	Передаюча функція об'єкту управління
1	$W(p) = \frac{2}{2p^2 + p + 1}$
2	$W(p) = \frac{3}{0.2p^2 + 0.1p + 1}$
3	$W(p) = \frac{4}{2p^2 + 0.5p + 1}$
4	$W(p) = \frac{1}{0.5p^2 + 0.25p + 1}$
5	$W(p) = \frac{6}{0.8p^2 + 0.4p + 1}$
6	$W(p) = \frac{3}{p^2 + 0.8p + 1}$
7	$W(p) = \frac{7}{5p^2 + 2.5p + 1}$
8	$W(p) = \frac{5}{2p^2 + 1.1p + 1}$
9	$W(p) = \frac{2.5}{6.2p^2 + 3p + 1}$
10	$W(p) = \frac{3.6}{2p^2 + p + 1}$
11	$W(p) = \frac{4.5}{3p^2 + 1.5p + 1}$
12	$W(p) = \frac{10}{0.5p^2 + 0.25p + 1}$
13	$W(p) = \frac{8.8}{8p^2 + 4p + 1}$
14	$W(p) = \frac{3}{1.5p^2 + 0.8p + 1}$
15	$W(p) = \frac{4.7}{5.6p^2 + 3p + 1}$

№ вар	Передаюча функція об'єкту управління
16	$W(p) = \frac{6.3}{5.2p^2 + 2.1p + 1}$
17	$W(p) = \frac{2.4}{2.7p^2 + 1.3p + 1}$
18	$W(p) = \frac{3}{0.2p^2 + 0.11p + 1}$
19	$W(p) = \frac{4}{2p^2 + 1.3p + 1}$
20	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

6.5 Контрольні питання

1. Як виходить аналітичний опис НЛР_ПД?
2. Викладіть евристичні міркування, що використовуються при синтезі НЛР ПД-типу.
3. Який вид має таблиця лінгвістичних правил НЛР_ПД?
4. Скільки кроків включає синтез нелінійного НЛР_ПД? Які це кроки?

7 СИНТЕЗ НЕЧІТКОГО РЕГУЛЯТОРА ПІ-ТИПУ У КОВЗАЮЧОМУ РЕЖИМІ

Мета роботи: засвоєння методики синтезу нелінійного нечіткого регулятора ПІ-типу у ковзаючому режимі роботи з використанням редактора системи нечіткого висновку

7.1. Ковзаючий режим нечіткого регулятора

Розглянемо відображення таблиці лінгвістичних змінних на фазову площину при 5 термах, що описують кожну змінну (рис. 7.1).

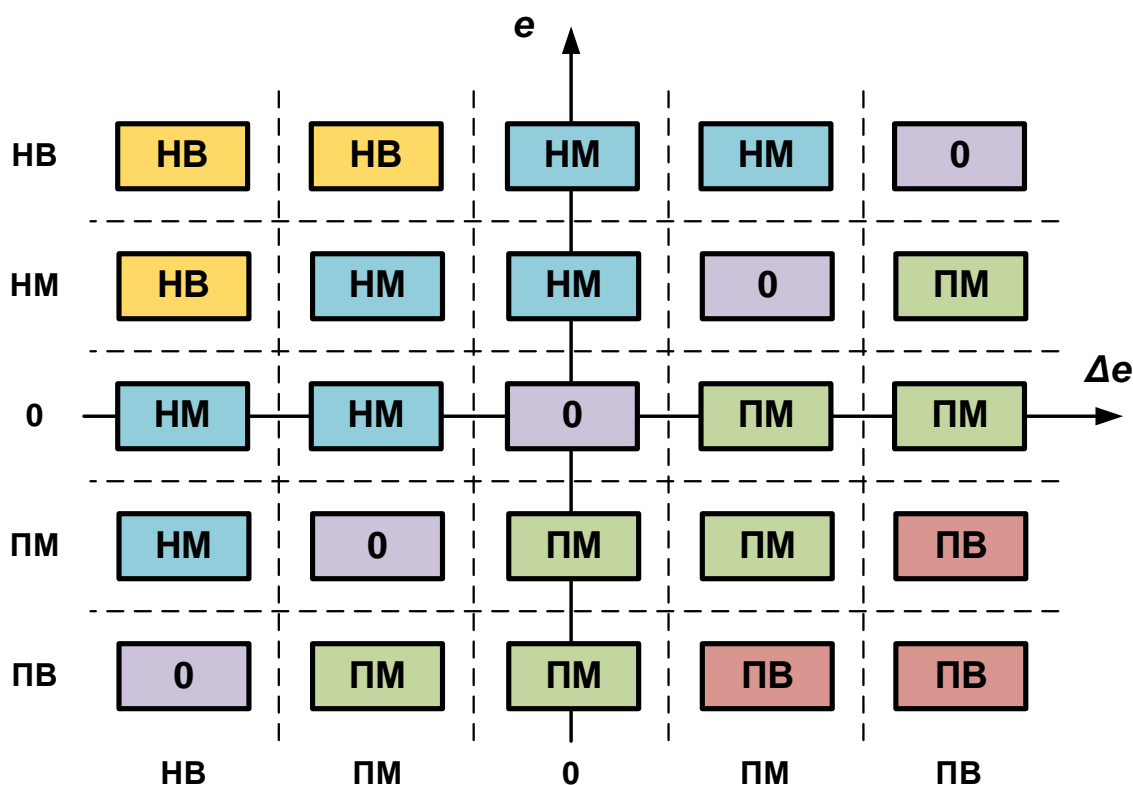


Рисунок 7.1 – Відображення закону управління фазову площину

Виділимо на фазовій площині області, де управління однаково (див. рис. 7.2).

Як свідчить рис. 7.2 величина модуля сигналу управління пропорційна відстані від прямої $e = \Delta e$. На використанні цієї особливості заснований так званий ковзний режим нечіткого регулятора.

Ковзаючий режим управління – робастний метод управління нелінійними системами в умовах невизначеності, який застосовується для об'єктів без істотного запізнення.

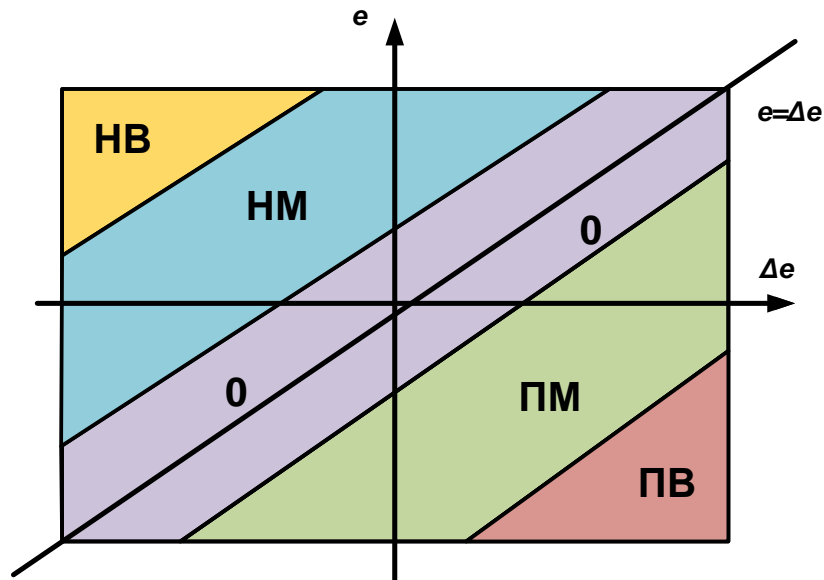


Рисунок 7.2 - Різні області сигналу керування на фазовій площині

Розглянемо нелінійний об'єкт з одним входом та одним виходом (SISO):

$$x^{(n)} = f(X) + u;$$

$$X = (x, \dot{x}, \dots, x^{(n)})^T,$$

де $u \in R$ - керуючий вхід; $x \in R$ - вихід, $X \in R^n$ - вектор стану. Невизначеність моделі описується функцією

$$f(X) = \hat{f}(X) + \Delta f(X);$$

$$\Delta f(X) \leq F(X)$$

де оцінка $\hat{f}(X)$ та $F(X)$ - відомі; $\Delta f(X)$ - невідома величина.

Мета управління полягає у виробленні такого сигналу зворотного зв'язку $u = u(X)$, щоб стан системи X наближався до бажаного стану X_d , тобто.

$$e = X - X_d \rightarrow 0.$$

Визначимо скалярну функцію

$$S(X, t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} e = e^{(n-1)} + C_{n-1}^1 \lambda e^{(n-2)} + C_{n-1}^2 \lambda^2 e^{(n-3)} + \dots + \lambda^{(n-1)} e$$



де λ – позитивна константа. Рівняння

$$s(X, t) = 0 \quad (7.1)$$

визначає залежну від часу перемикальну поверхню в просторі станів R^n .

Для системи другого порядку ($n = 2$) поверхня ковзання $S(t)$ визначається виразом

$$S(X, t) = \dot{e} + \lambda e = \dot{x} + \lambda x - \dot{x}_d - \lambda x_d = 0$$

Рівняння має єдине рішення $e(t) = 0$ за нульових початкових умов $e(0) = 0$.

Проблема керування траєкторією зводиться до завдання забезпечення (7.1).

Це може бути забезпечено при виконанні умови ковзання:

$$\frac{1}{2}(s)^2 \leq -\eta|s|$$

де η - позитивна константа.

Інакше кажучи, для того щоб у системі існував ковзний режим, необхідно сформулювати управління таким чином, щоб точка, що зображає, один раз потрапивши на поверхню ковзання, вже не могла зійти з неї і далі рухалася тільки вздовж цієї поверхні. Нехай у деякий момент часу $S > 0$. Тоді похідна за часом dS/dt (швидкість зміни S) має бути негативною. У цьому випадку S (відхилення від поверхні ковзання) буде зменшуватися до нуля, тобто зображувальна точка буде рухатися у напрямку до S . Нехай, далі точка перетину поверхню $S = 0$ і тепер $S < 0$. Відповідно, повинно змінитися і значення похідної, а саме виконати умову $dS/dt > 0$. У цьому випадку зображувальна точка буде рухатися назад у напрямку до поверхні ковзання.

Відзначимо лише, що режим ковзання вимагає опису фазової траєкторії бажаної системи (лінії ковзання). Якщо реальна фазова траєкторія перетинає бажану фазову траєкторію, сигнал управління перемикається. Перемикання можуть відбуватися з дуже великою (теоретично нескінченною) частотою, утримуючи при цьому перехідний процес на лінії ковзання. При зміні властивостей об'єкта змінюється лише частота перемикання сигналу управління, а траєкторія, вздовж якої проходить рух, залишається незмінною.

Таким чином, можна вважати, що задана лінія перемикання, де значення сигналу управління дорівнює нулю. Вище та нижче цієї лінії сигнал керування має різні знаки, а його модуль пропорційний відстані від цієї лінії (рис. 7.3):

$$S: \Delta e + \lambda e = 0.$$

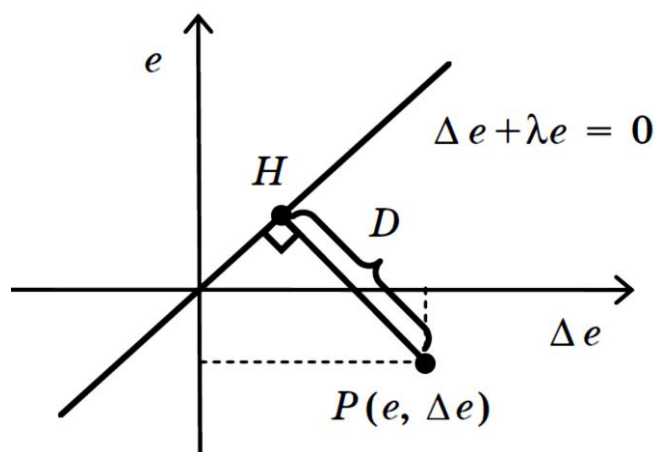


Рисунок 7.3 - Відстань до лінії перемикання

Розглянемо розрахунок дистанції від точки у фазовому просторі до лінії перемикання (рис. 7.3 де H – точка перетину лінії перемикання і перпендикуляра, проведеного до цієї лінії з точки P).

Відстань розраховується за відомими формулами (замість похідної тут використано збільшення):

$$d = \sqrt{(e - e_1)^2 + (\Delta e - \Delta e_1)^2} = \frac{|\Delta e_1 + \lambda \Delta e_1|}{\sqrt{1 + \lambda^2}}.$$

З урахуванням знаку відстань можна описати формулою

$$D = \operatorname{sgn}(S) \frac{|\Delta e_1 + \lambda \Delta e_1|}{\sqrt{1 + \lambda^2}} = \frac{\Delta e_1 + \lambda \Delta e_1}{\sqrt{1 + \lambda^2}},$$

де

$$\operatorname{sgn}(S) = \begin{cases} 1, & \text{якщо } S > 0; \\ -1, & \text{якщо } S < 0. \end{cases}$$

Таким чином, за рахунок використання відстані до лінії перемикання зі знаком можна описати нечіткий закон управління одномірною таблицею (табл. 7.1).

Таблиця 7.1 – Одномірний нечіткий закон управління

D	x_1	x_2	x_3	x_4	x_5
D^*	НВ	НМ	0	ПМ	ПВ
U^*	НВ	НМ	0	ПМ	ПВ

У табл. 7.1 величина D^* визначає лінгвістичне значення відстані, а

U^* – лінгвістичне значення сигналу управління. Число правил, що описують нечіткий закон управління, скорочується таким чином у п'ять разів (порівняно з табл. 10.5). У цьому полягає основна перевага використання ковзного режиму НЛР. Справді, величини x_i , зазначені у табл. 7.1 є центрами відповідних термів, і нечіткий закон управління вимагає завдання всього 5 параметрів, а не 25.

Лінію перемикавання можна описати і у випадку, коли на вході регулятора подається більше двох вхідних змінних. В загальному випадку перемикальна поверхня задається рівнянням

$$S = e^{(n-1)} + \lambda_{n-1} \lambda e^{(n-1)} + \dots + \lambda_2 \dot{e} + \lambda_1 e = 0$$

Відповідно, відстань до лінії перемикавання визначається за формулою

$$D = \frac{e^{(n-1)} + \lambda_{n-1} \lambda e^{(n-1)} + \dots + \lambda_2 \dot{e} + \lambda_1 e}{\sqrt{1 + \lambda_{n-1}^2 + \dots + \lambda_2^2 + \lambda_1^2}}$$

Отже, змінюється лише формула обчислення відстані, а алгоритм нечіткого управління залишається тим самим, і завдання проектування НЛР завжди значно спрощується.

7.2 Приклад моделювання роботи НЛР у режимі спостереження

Нехай об'єкт управління заданий передавальною функцією

$$W = \frac{6,5}{0,02p^2 + 0,4p + 1}$$

На рис. 7.4 представлена схема моделювання в MatLab Simulink.

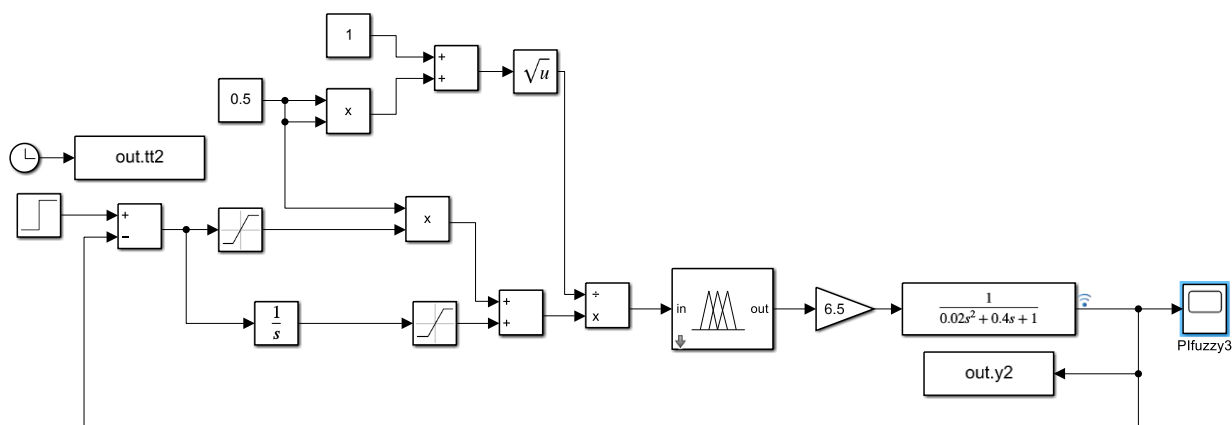


Рисунок 7.4 – Математична модель НЛР в режимі спостереження

Вхідний сигнал НЛР формується відповідно до (10.4). Нечіткий

закон управління відповідає табл. 10.5. При рівномірному розташуванні термів вхідно-вихідних лінгвістичних змінних завдання конструювання НЛР зводиться до правильного вибору коефіцієнта λ .

Синтезуємо Fuzzy Logic Controller НЛР в режимі спостереження надаємо Fis name: PISfis7 (ім'я може бути будь-яке).

Викликається редактор системи нечіткого висновку (Fuzzy Inference System – FIS) у MatLab командою

```
>> fuzzy
```

Дале проводиться налаштування головного вікна редактора нечіткої логічної системи (див. рис. 7.5.).

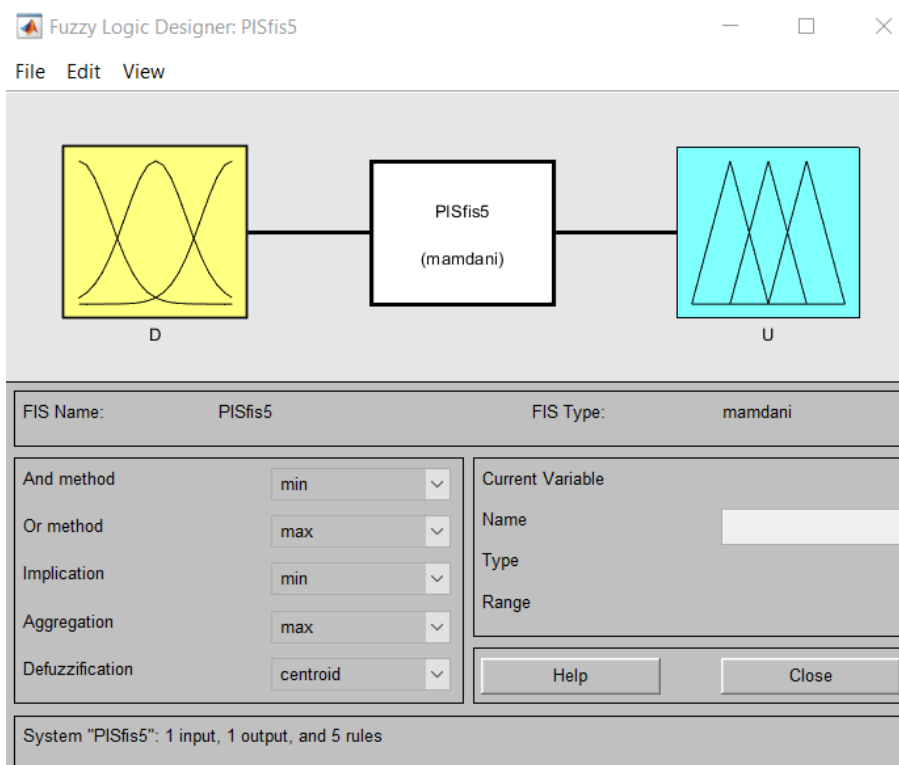


Рисунок 7.5 – Налаштування інтерфейсу FIS editor

У FIS editor задається опис систему нечіткого логічного висновку Mamdani. Для створюваної системи обирається вид логічного зв'язку (And method – min) та (Or method – max), вид імплікації (Implication – min), спосіб агрегування висновків правил (Aggregation – max) та метод дефазифікації (Defuzzification – centroid).

У меню Edit послідовно додаємо вхідну змінну з розміром базової шкали (Range= [-1 1]) та вихідну змінну з розміром базової шкали (Range= [-1 1]).

Для опису вхідних логічних змінних у редакторі функцій приналежності (Membership Function Editor), задаємо для кожної змінної трикутну функцію приналежності.

Терми лінгвістичної змінної «Помилка управління» розміщуються відповідно:

Name = "NB"; Type = "trimf"; Param = [-1.33 -1 -0.33];
 Name = "NM"; Type = "trimf"; Param = [-1 -0.33 -0.03];
 Name = "NS"; Type = "trimf"; Param = [-0.33 -0.033 0];
 Name = "Z"; Type = "trimf"; Param = [-0.1 0 0.1];
 Name = "PS"; Type = "trimf"; Param = [0 0.033 0.33];
 Name = "PM"; Type = "trimf"; Param = [0.033 0.33 1];
 Name = "PB"; Type = "trimf"; Param = [0.33 1 1.33].

Результат налаштування функцій приналежності вхідних змінних «Помилка управління» наведено на рис. 7.6.



Рисунок 7.6 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Для опису вихідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. З параметрами:

Name = "NBD"; Type = "trimf"; Param = [-1.5 -1 -0.5];
 Name = "NSD"; Type = "trimf"; Param = [-1 -0.5 0];
 Name = "ZD"; Type = "trimf"; Param = [-0.5 0 0.5];
 Name = "PSD"; Type = "trimf"; Param = [0 0.5 1];
 Name = "PBD"; Type = "trimf"; Param = [0.5 1 1.5].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 7.7.



Рисунок 7.7 – Результат налаштування функцій приналежності вихідних змінних

Після опису вхідні та вихідні лінгвістичні змінні, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку дорівнює 5. Управляючі правила сформуємо згідно з табл. 10.5:

1. if (input1 is NBD) then (output1 is NBu) (1)
2. if (input1 is NSD) then (output1 is NSu) (1)
3. if (input1 is ZD) then (output1 is Zu) (1)
4. if (input1 is PSD) then (output1 is PSD) (1)
5. if (input1 is PBD) then (output1 is PBU) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 7.8).

Посилки у правилах пов'язані (connection) за допомогою операції *and*. Введене правило забезпечене ваговим коефіцієнтом (*Weight=1*).

Після опису правил можливо за допомогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 7.9);
- інтерфейсу *Surface Viewer* можливо переглянути графічне подання закону управління (рис. 7.10).

1. If (input1 is NBD) then (output1 is NBU) (1)
 2. If (input1 is NSD) then (output1 is NSU) (1)
 3. If (input1 is ZD) then (output1 is ZDU) (1)
 4. If (input1 is PSD) then (output1 is PSU) (1)
 5. If (input1 is PBD) then (output1 is PBU) (1)

If input1 is

NBD
 NSD
 ZD
 PSD
 PBD
 none

not

Connection

or
 and

Weight: 1

Delete rule Add rule Change rule << >>

Then output1 is

NBU
 NSU
 ZDU
 PSU
 PBU
 none

not

FIS Name: PISfis5

Help Close

Рисунок 7.8 - Налаштовування опису правил функціонування НЛР

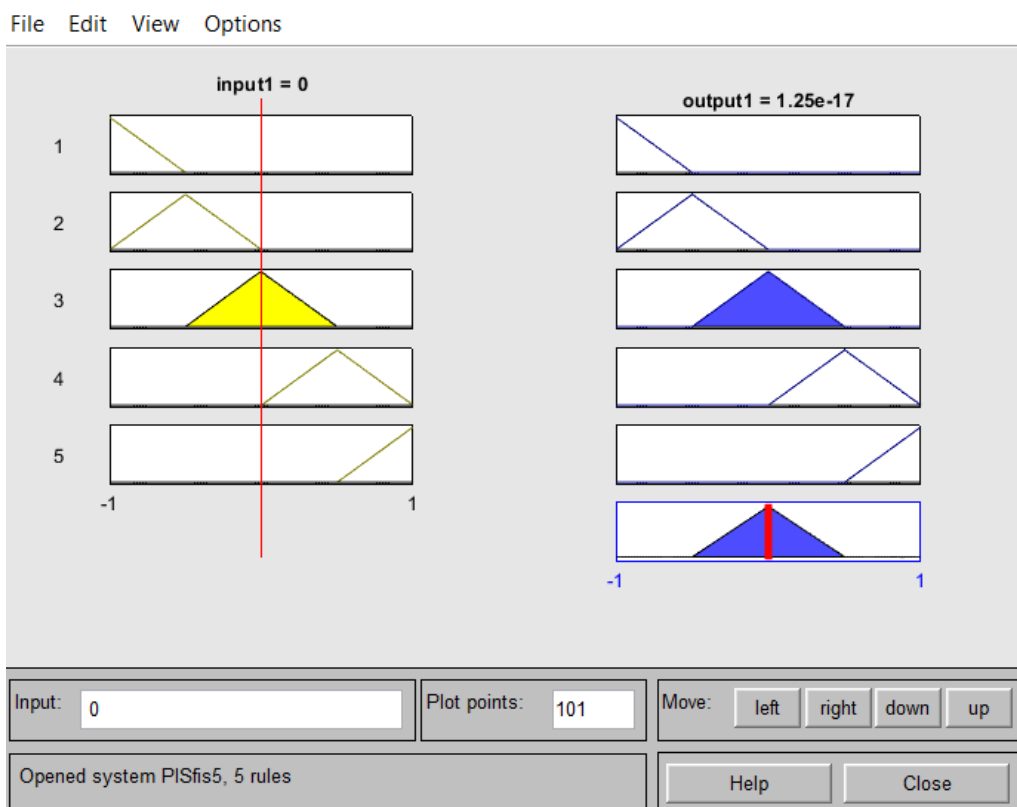


Рисунок 7.9 - Робота системи НЛР

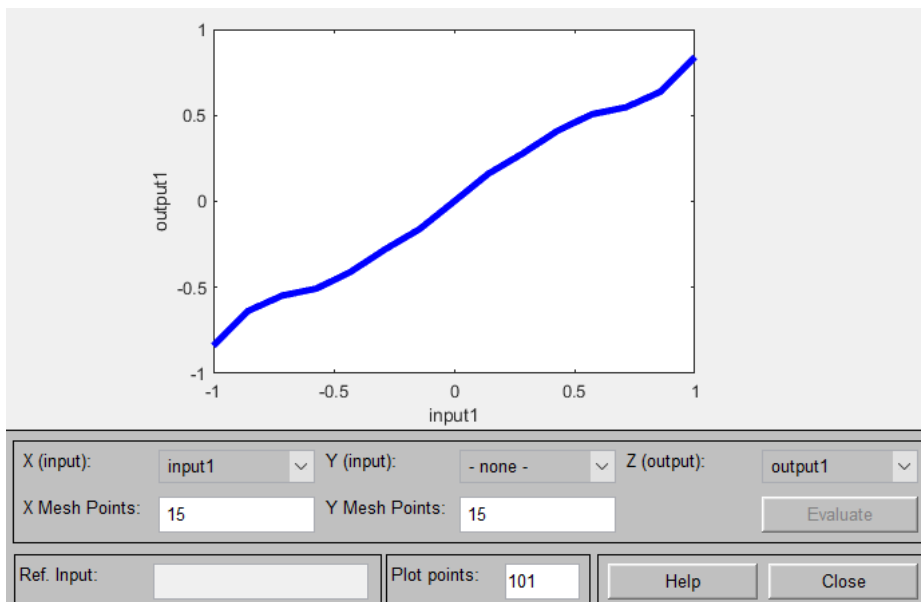


Рисунок 7.10 – Графік управління системою нечіткого висновку

Після процедур налаштування НЛР у редакторі системи нечіткого висновку здійснює запис результатів налаштування у Fuzzy Logic Controller. Для цього у вкладці File здійснюємо Export в математичну модуль нечіткого контролера From Workspase вказавши ім'я Fuzzy Logic Controller.

Після експорту закону функціонування НЛР на математичної моделі системи управління (див. рис. 7.4) получено перехідний процес для синтезованого НЛР при різних коефіцієнтах λ (див. рис. 7.11).

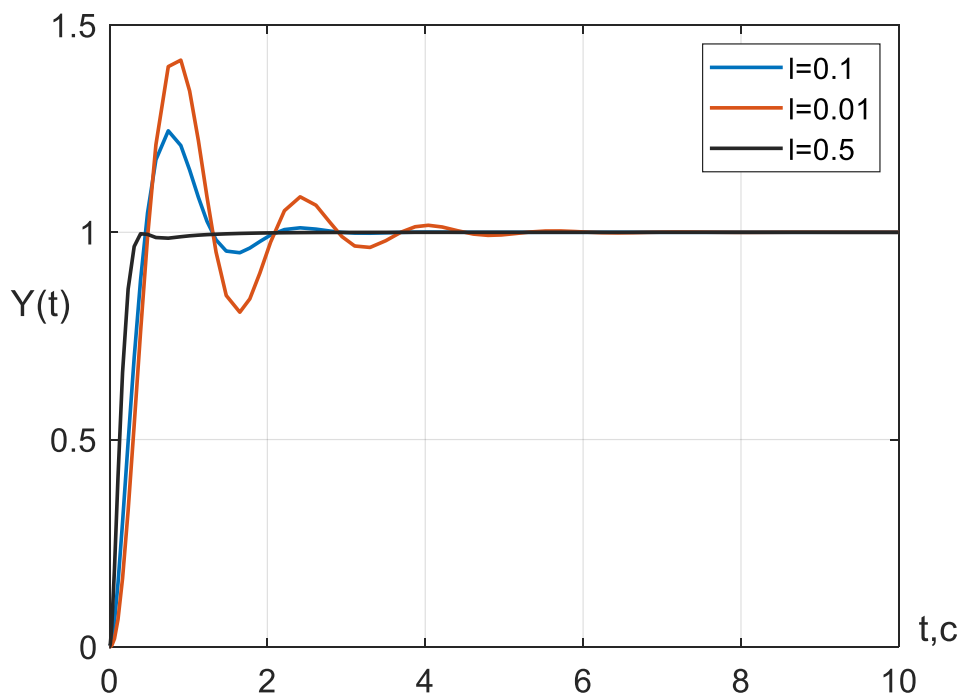


Рисунок 7.11 – Графік перехідного процесу для синтезованого НЛР при різних коефіцієнтах λ

7.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 7.2.

За вказаними у табл. 7.2 потрібно синтезувати нелінійний нечіткий регулятор ПІ-типу у режимі спостереження з використанням системи нечіткого висновку:

Таблиця 7.2 - Варіанти індивідуальних завдань

№ вар	Передаточна функція об'єкту управління
1	$W(p) = \frac{2}{2p^2 + p + 1}$
2	$W(p) = \frac{3}{0.2p^2 + 0.1p + 1}$
3	$W(p) = \frac{4}{2p^2 + 0.5p + 1}$
4	$W(p) = \frac{1}{0.5p^2 + 0.25p + 1}$
5	$W(p) = \frac{6}{0.8p^2 + 0.4p + 1}$
6	$W(p) = \frac{3}{p^2 + 0.8p + 1}$
7	$W(p) = \frac{7}{5p^2 + 2.5p + 1}$
8	$W(p) = \frac{5}{2p^2 + 1.1p + 1}$
9	$W(p) = \frac{2.5}{6.2p^2 + 3p + 1}$
10	$W(p) = \frac{3.6}{2p^2 + p + 1}$
11	$W(p) = \frac{4.5}{3p^2 + 1.5p + 1}$
12	$W(p) = \frac{10}{0.5p^2 + 0.25p + 1}$
13	$W(p) = \frac{8.8}{8p^2 + 4p + 1}$
14	$W(p) = \frac{3}{1.5p^2 + 0.8p + 1}$
15	$W(p) = \frac{4.7}{5.6p^2 + 3p + 1}$
16	$W(p) = \frac{6.3}{5.2p^2 + 2.1p + 1}$

Продовження табл. 7.2

№ вар	Передаюча функція об'єкту управління
17	$W(p) = \frac{2.4}{2.7p^2 + 1.3p + 1}$
18	$W(p) = \frac{3}{0.2p^2 + 0.11p + 1}$
19	$W(p) = \frac{4}{2p^2 + 1.3p + 1}$
20	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

7.4 Контрольні питання

1. Що таке ковзний режим нечіткого регулятора?
2. Як використовується фазова площина при описі ковзного режиму?
3. У чому основна перевага використання ковзного режиму НЛР?

8 СИНТЕЗ НЕЧІТКОГО СУПЕРВІЗОРА

Мета роботи: засвоєння методики синтезу нечіткого супервізора регулятора ПІД-типу з використанням редактора системи нечіткого висновку

8.1 Теоретичні основи синтезу нечіткого супервізора

Ідея нечіткого супервізора полягає в організації дворівневої системи, в якій на нижньому рівні знаходиться звичайний ПІД-регулятор, а на верхньому рівні – НЛР (рис. 8.1).

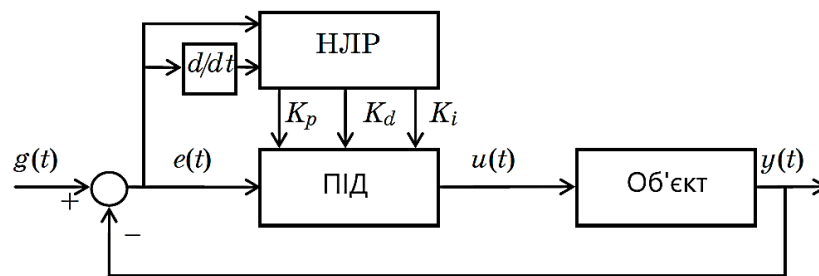


Рисунок 8.1 – Структурна схема нечіткого супервізора

Призначення НЛР полягає в тому, щоб автоматично змінювати коефіцієнти ПІД-регулятора на різних стадіях перехідного процесу. Можливі різні реалізації нечітких супервізорів, нижче розглядаються три схеми.

Перша схема безпосередньо реалізує структуру рис. 8.1. Як було показано вище, НЛР має змінний коефіцієнт посилення. У НЛР_ПІД є три вхідні змінні, тому вважатимуться, що його коефіцієнт посилення складається з трьох складових

$$K_{PID} = K_P + K_I + K_D.$$

Нехай помилка управління та її похідна описуються відповідно до рис. 8.2.

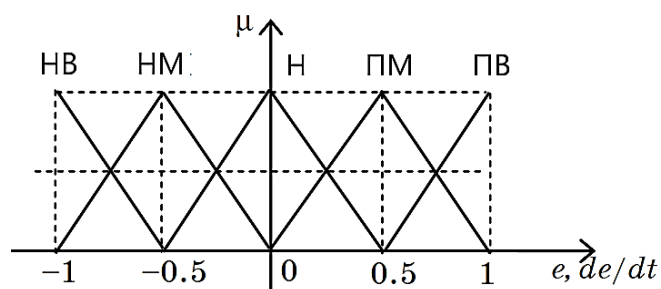


Рисунок 8.2 – Лінгвістичний опис вхідних змінних НЛР_ПІД

Лінгвістичні значення нормалізованих коефіцієнтів НЛР_ПІД можна описати за допомогою семи термів з найменуваннями «нульовий (Н)», «малий 1 (М1)», «малий 2 (М2)», «середній 1 (С1)», «середній 2 (С2)», «Великий 1 (В1)», «Великий 2 (В2)» (рис. 8.3).

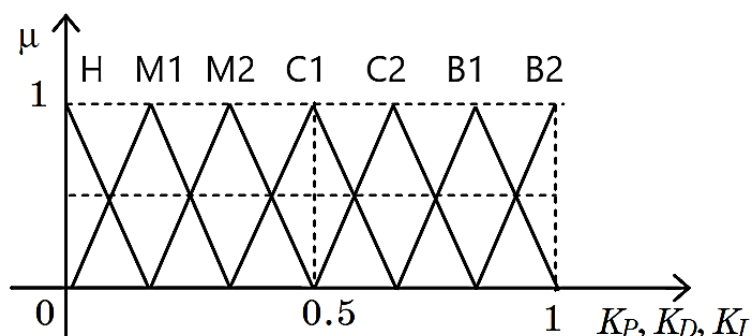


Рисунок 8.3 – Лінгвістичний опис коефіцієнтів підсилення

Для опису вимог до коефіцієнтів ПІД-регулятора розглянемо типовий графік перехідного процесу (рис. 8.4).

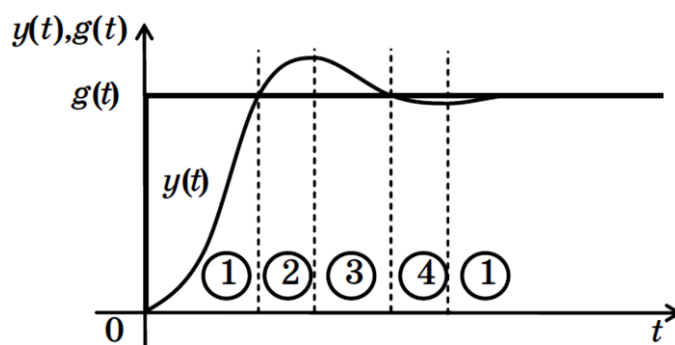


Рисунок 8.4 – Типовий перехідний процес

Розглядаючи знак помилки та її похідної, на графіку можна виділити 4 різні області:

- 1: $e(t) > 0, de/dt < 0$;
- 2: $e(t) < 0, de/dt < 0$;
- 3: $e(t) < 0, de/dt > 0$;
- 4: $e(t) > 0, de/dt > 0$.

Найбільше просто описати вимоги до коефіцієнта K_I . Він відповідає за помилку, що встановилася, і його зміну можна описати правилом: «Якщо встановилася помилка велика, то K_I великий». Оскільки розглядається встановлена помилка, значення похідної тут, не має значення. Однак, щоб гасити коливання в області нульової помилки, необхідно надавати K_I малі значення, коли похідна ненульова. У результаті виходить наступний опис змінного K_I (табл. 8.1).



Таблиця 8.1 – Лінгвістичні правила опису K_I

Таблиця правил		$(de/dt)^*$				
		НВ	НМ	Н	ПМ	ПВ
Δe^*	НВ	C1	C1	C1	C1	C1
	НМ	M1	M1	M1	M1	M1
	Н	M2	M2	Н	M2	M2
	НМ	M1	M1	M1	M1	M1
	Н	C1	C1	C1	C1	C1

Коефіцієнт K_P відповідає за помилку відстеження вхідного сигналу e , і вимогу до його значення можна описати правилом: "Якщо e велика, то K_P великий". При нульовій помилці з позитивною похідною слід дещо збільшувати K_P для зменшення часу перехідного процесу. Табл. 8.2 описує ці правила.

Таблиця 8.2 – Лінгвістичні правила опису K_P

Таблиця правил		$(de/dt)^*$				
		НВ	НМ	Н	ПМ	ПВ
Δe^*	НВ	B2	B2	B2	B2	C1
	НМ	B1	B1	B1	C2	B2
	Н	Н	Н	M2	M1	M1
	НМ	B1	B1	B1	C2	B2
	Н	B2	B2	B2	B2	B2

Коефіцієнт K_D відповідає за перерегулювання, вимогу для його значення можна описати правилом: «Якщо перерегулювання велике, то K_D великий». Перерегулювання виникає при наближенні $y(t)$ до $g(t)$,

коли помилка та її похідна мають різні знаки (області 1 та 3 на рис. 11.126). У цих ситуаціях K_D повинен мати велике значення, інакше пропорційно зменшується до нуля (див. табл. 8.3).

Таблиця 8.3 – Лінгвістичні правила опису K_D

Таблиця правил		$(de/dt)^*$				
		НВ	НМ	Н	ПМ	ПВ
Δe^*	НВ	Н	М1	С1	С2	В2
	НМ	М1	В1	С2	В2	В2
	Н	С1	С2	С2	В2	В2
	НМ	В1	В2	В2	М1	М1
	Н	В2	В2	В2	В2	В2

Табл.8.1-8.3 можуть бути зведені в одну табл. 8.4.

Таблиця 8.4 – Правила корекції коефіцієнтів НЛР_ПІД

№	e	de/dt	K_p	K_D	K_I	№	e	de/dt	K_p	K_D	K_I
1	НВ	НВ	В2	Н	С1	14	Н	ПМ	М1	В2	М2
2	НВ	НМ	В2	М1	С1	15	Н	ПВ	М1	В2	М2
3	НВ	Н	В2	С1	С1	16	ПМ	НВ	В1	В2	М1
4	НВ	ПМ	В2	С2	С1	17	ПМ	НМ	В1	В2	М1
5	НВ	ПВ	В2	В2	С1	18	ПМ	Н	В1	В2	М1
6	НМ	НВ	В1	М1	М1	19	ПМ	ПМ	С2	В2	М1
7	НМ	НМ	В1	В1	М1	20	ПМ	ПВ	В2	В2	М1
8	НМ	Н	В1	С2	М1	21	ПВ	НВ	В2	В2	С1
9	НМ	ПМ	С2	В2	М1	22	ПВ	НМ	В2	В2	С1
10	НМ	ПВ	В2	В2	М1	23	ПВ	Н	В2	В2	С1
11	Н	НВ	Н	В1	М2	24	ПВ	ПМ	В2	В2	С1
12	Н	НМ	Н	В2	М2	25	ПВ	ПВ	В2	В2	С1
13	Н	Н	М2	В2	Н						

8.2 Приклад синтезу нечіткого супервізора ПІД-регулятора

На рис. 8.5 показана схема моделювання НЛР_ПІД.

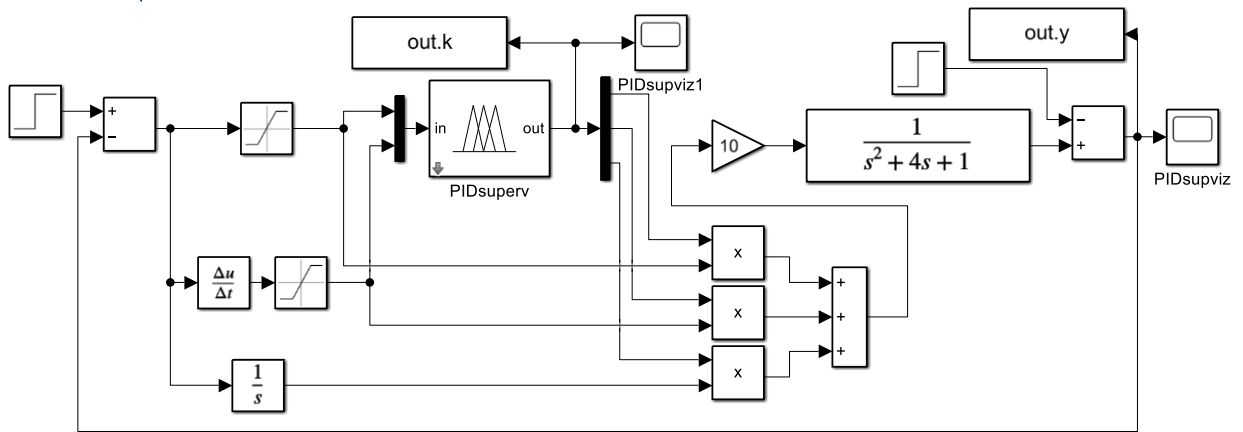


Рисунок 8.5 – Варіант математичного моделювання системи управління з нечітким супервізором ПІД регулятора

Відповідно до рис. 8.5 сформуємо в Simulink MatLab модель системи управління з нечітким супервізором ПІД регулятора при $x = 10$.

Fuzzy Logic Controller надаємо Fis name: PIDsupv1 (ім'я може бути будь-яке).

Викликається редактор системи нечіткого висновку (Fuzzy Inference System – FIS) у MatLab командою

`>> fuzzy`

Дале проводиться налаштування головного вікна редактора нечіткої логічної системи (див. рис. 8.6.).

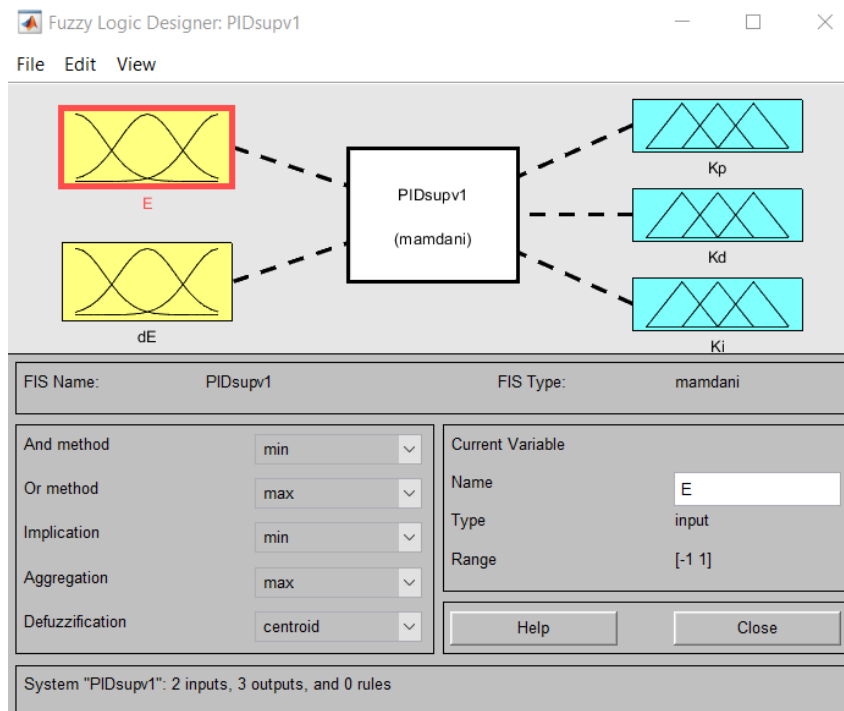


Рисунок 8.6 – Налаштування інтерфейсу FIS editor

У *FIS editor* задається опис систему нечіткого логічного висновку *Mamdani*. Для створюваної системи обирається вид логічного зв'язку (*And method – min*) та (*Or method – max*), вид імплікації (*Implication – min*), спосіб агрегування висновків правил (*Aggregation – max*) та метод дефазифікації (*Defuzzification – centroid*).

У меню *Edit* послідовно додаємо 2 вхідні змінні з розміром базової шкали (*Range= [-1 1]*) та 3 вихідні змінні з розміром базової шкали (*Range= [0 1]*).

Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної змінної трикутну функцію приналежності.

Нехай терми лінгвістичної змінної «Помилка управління» розміщуються відповідно до рис. 11.129 з 5 параметрами:

Name = "NBe"; Type = "trimf"; Param = [-1.5 -1 -0.5];

Name = "NSe"; Type = "trimf"; Param = [-1 -0.5 0];

Name = "Ze"; Type = "trimf"; Param = [-0.5 0 0.5];

Name = "PSe"; Type = "trimf"; Param = [0 0.5 1].

Name = "PBe"; Type = "trimf"; Param = [0.5 1 1.5].

Результат налаштування функцій приналежності вхідних змінних «Помилка управління» наведено на рис. 8.7.

Терми вхідної лінгвістичної змінної «Похідна помилки управління» формуємо рівномірно. Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. З параметрами:

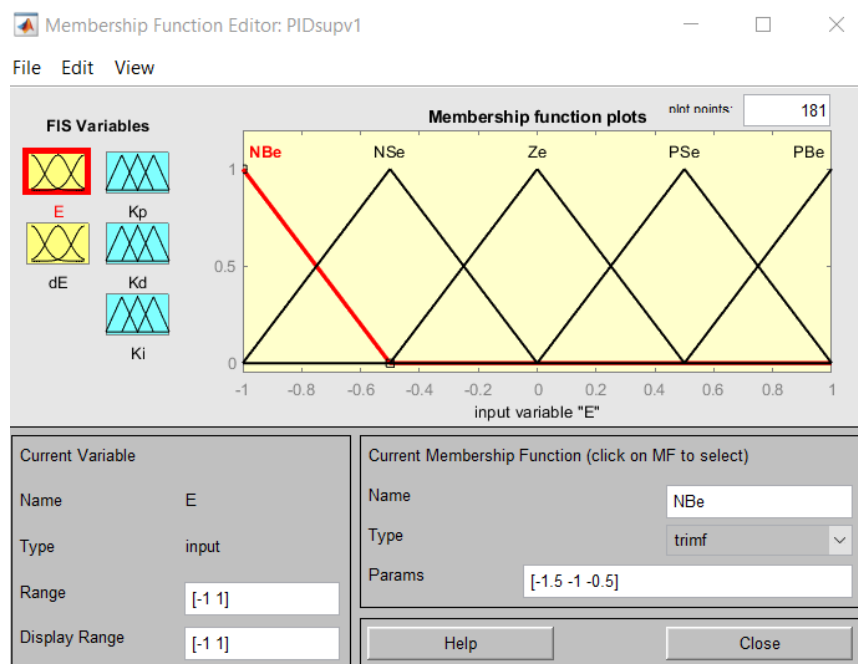


Рисунок 8.7 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Name = "NBde"; Type = "trimf"; Param = [-1.5 -1 -0.5];
 Name = "NSde"; Type = "trimf"; Param = [-1 -0.5 0];
 Name = "Zde"; Type = "trimf"; Param = [-0.5 0 0.5];
 Name = "PSde"; Type = "trimf"; Param = [0 0.5 1].
 Name = "PBde"; Type = "trimf"; Param = [0.5 1 1.5].

Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління» на рис. 8.8.

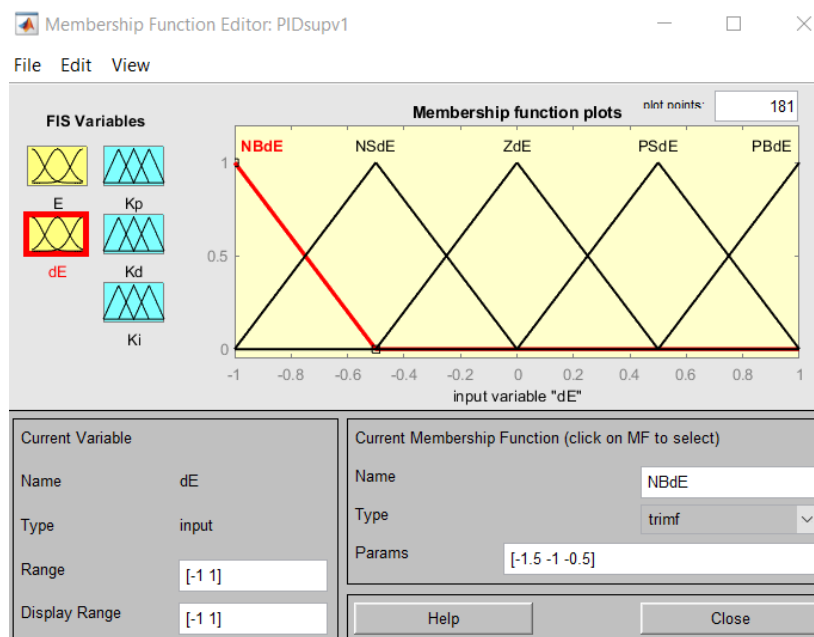


Рисунок 8.8 – Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління»

Для опису вихідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності.

- для K_p :

Name = "Zp"; Type = "trimf"; Param = [-0.166 0 0.166];
 Name = "M1p"; Type = "trimf"; Param = [0 0.166 0.33];
 Name = "M2p"; Type = "trimf"; Param = [0.166 0.33 0.5];
 Name = "C1p"; Type = "trimf"; Param = [0.33 0.5 0.66];
 Name = "C2p"; Type = "trimf"; Param = [0.5 0.66 0.83].
 Name = "B1p"; Type = "trimf"; Param = [0.66 0.83 1];
 Name = "B2p"; Type = "trimf"; Param = [0.833 1 1.166].

- для K_d :

Name = "Zd"; Type = "trimf"; Param = [-0.166 0 0.166];
 Name = "M1d"; Type = "trimf"; Param = [0 0.166 0.33];
 Name = "M2d"; Type = "trimf"; Param = [0.166 0.33 0.5];
 Name = "C1d"; Type = "trimf"; Param = [0.33 0.5 0.66];
 Name = "C2d"; Type = "trimf"; Param = [0.5 0.66 0.83].

Name = "B1d"; Type = "trimf"; Param = [0.66 0.83 1];
 Name = "B2d"; Type = "trimf"; Param = [0.833 1 1.166].

- для K_i :

Name = "Zi"; Type = "trimf"; Param = [-0.166 0 0.166];
 Name = "M1i"; Type = "trimf"; Param = [0 0.166 0.33];
 Name = "M2i"; Type = "trimf"; Param = [0.166 0.33 0.5];
 Name = "C1i"; Type = "trimf"; Param = [0.33 0.5 0.66];
 Name = "C2i"; Type = "trimf"; Param = [0.5 0.66 0.83].
 Name = "B1i"; Type = "trimf"; Param = [0.66 0.83 1];
 Name = "B2i"; Type = "trimf"; Param = [0.833 1 1.166].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 8.9.

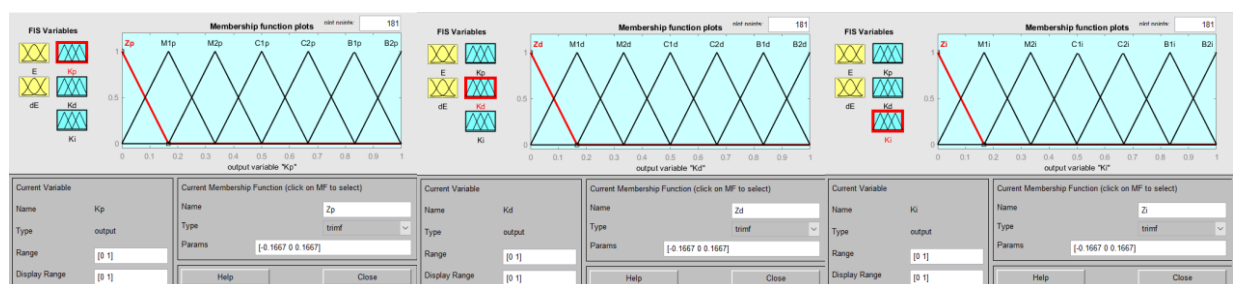


Рисунок 8.9 – Результат налаштування функцій приналежності вихідної змінної K_p , K_d , K_i

Після опису вхідні та вихідні лінгвістичних змінних, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку 25. Управляючі правила сформульовано згідно з табл. 8.4:

1. if (E is NBe) and (dE is NBdE) then (K_p is B2p)(K_d is Zd)(K_i is C1i) (1)
2. if (E is NBe) and (dE is NSdE) then (K_p is B2p)(K_d is M1d)(K_i is C1i) (1)
3. if (E is NBe) and (dE is ZdE) then (K_p is B2p)(K_d is C1d)(K_i is C1i) (1)
4. if (E is NBe) and (dE is PSdE) then (K_p is B2p)(K_d is C2d)(K_i is C1i) (1)
5. if (E is NBe) and (dE is PBdE) then (K_p is B2p)(K_d is B2d)(K_i is C1i) (1)
6. if (E is NSe) and (dE is NBdE) then (K_p is B1p)(K_d is M1d)(K_i is M1i) (1)
7. if (E is NSe) and (dE is NSdE) then (K_p is B1p)(K_d is B1d)(K_i is M1i) (1)
8. if (E is NSe) and (dE is ZdE) then (K_p is B1p)(K_d is C2d)(K_i is M1i) (1)
9. if (E is NSe) and (dE is PSdE) then (K_p is C2p)(K_d is B2d)(K_i is M1i) (1)
10. if (E is NSe) and (dE is PBdE) then (K_p is B2p)(K_d is B2d)(K_i is M1i) (1)
11. if (E is Ze) and (dE is NBdE) then (K_p is Zp)(K_d is B1d)(K_i is M2i) (1)
12. if (E is Ze) and (dE is NSdE) then (K_p is Zp)(K_d is B2d)(K_i is M2i) (1)
13. if (E is Ze) and (dE is ZdE) then (K_p is M2p)(K_d is B2d)(K_i is Zi) (1)
14. if (E is Ze) and (dE is PSdE) then (K_p is M1p)(K_d is B2d)(K_i is M2i) (1)
15. if (E is Ze) and (dE is PBdE) then (K_p is M1p)(K_d is B2d)(K_i is M2i) (1)
16. if (E is PSe) and (dE is NBdE) then (K_p is B1p)(K_d is B2d)(K_i is M1i) (1)
17. if (E is PSe) and (dE is NSdE) then (K_p is B1p)(K_d is B2d)(K_i is M1i) (1)
18. if (E is PSe) and (dE is ZdE) then (K_p is B1p)(K_d is B2d)(K_i is M1i) (1)

19. if (E is PSe) and (dE is PSdE) then (Kp is C2p)(Kd is B2d)(Ki is M1i) (1)
20. if (E is PSe) and (dE is PBdE) then (Kp is B2p)(Kd is B2d)(Ki is M1i) (1)
21. if (E is PBe) and (dE is NBdE) then (Kp is B2p)(Kd is B2d)(Ki is C1i) (1)
22. if (E is PBe) and (dE is NSdE) then (Kp is B2p)(Kd is B2d)(Ki is C1i) (1)
23. if (E is PBe) and (dE is ZdE) then (Kp is B2p)(Kd is B2d)(Ki is C1i) (1)
24. if (E is PBe) and (dE is PSdE) then (Kp is B2p)(Kd is B2d)(Ki is C1i) (1)
25. if (E is PBe) and (dE is PBdE) then (Kp is B2p)(Kd is B2d)(Ki is C1i) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 8.10).

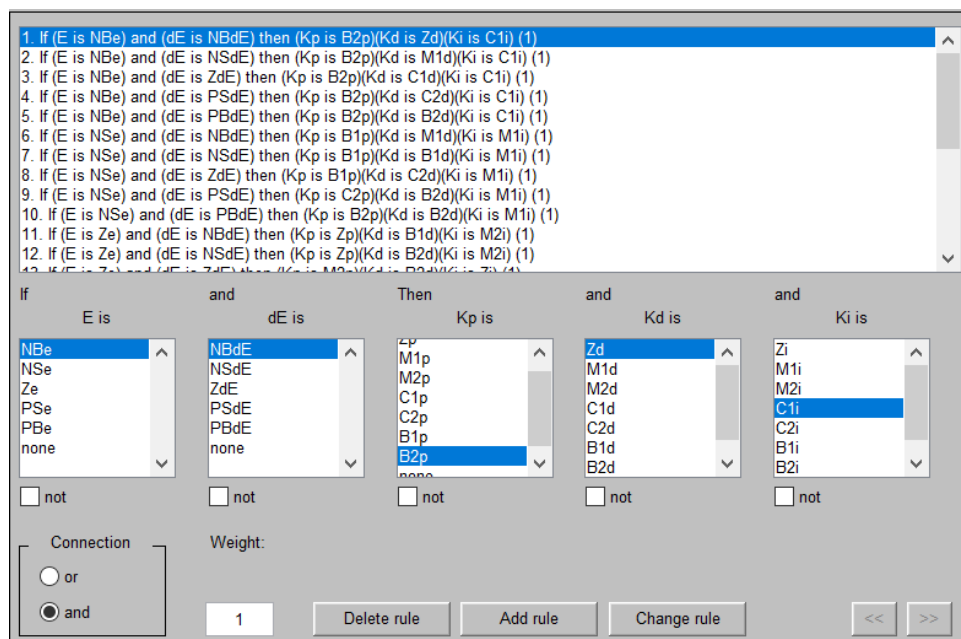


Рисунок 8.10 - Налаштовування опису правил функціонування нечіткого супервізора НЛР_ПІД

Посилки у правилах пов'язані (connection) за допомогою операції *and*. Введене правило забезпечене ваговим коефіцієнтом (*Weight=1*).

Після опису правил можливо за допомогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 8.11).

- інтерфейсу *Surface Viewer* можливо переглянути графічне подання закону управління (рис. 8.12).

Після процедур налаштування нечіткого супервізора НЛР_ПІД у редакторі системи нечіткого висновку здійснює запис результатів налаштування у *Fuzzy Logic Controller*. Для цього у вкладці *File* здійснюємо *Export* в математичну модуль нечіткого контролера *From Workspase* вказавши ім'я *Fuzzy Logic Controller*.

На рис. 8.13 представлені графіки зміни коефіцієнтів ПІД-регулятора під керуванням нечіткого супервізора протягом перехідного процесу.

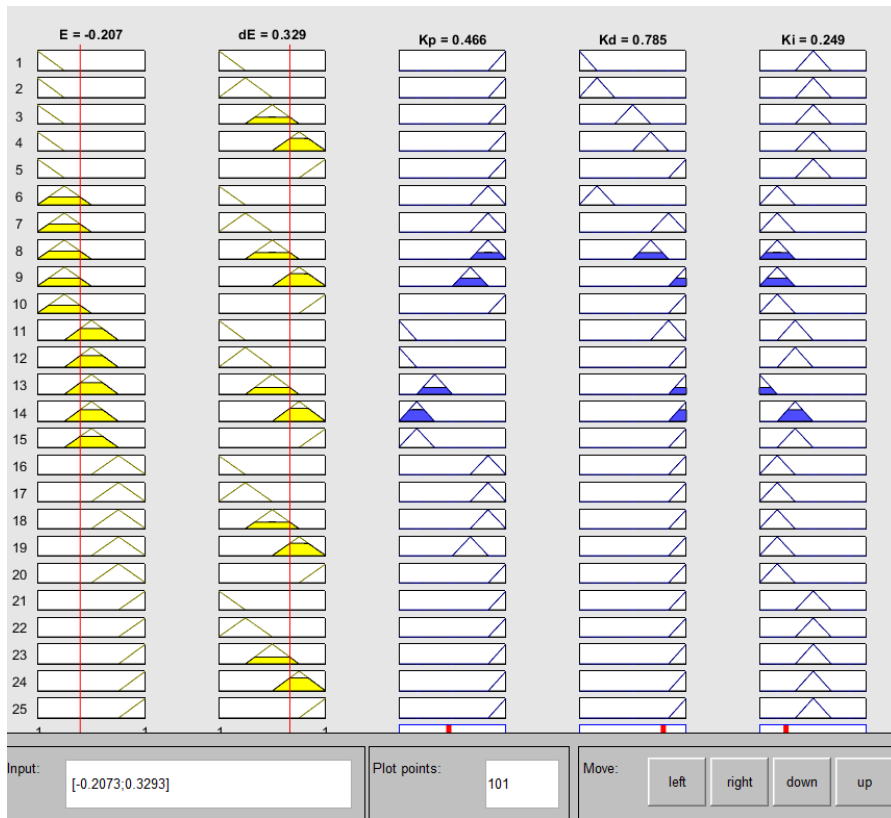


Рисунок 8.11 - Робота системи нечіткого супервізора НЛР_ПІД

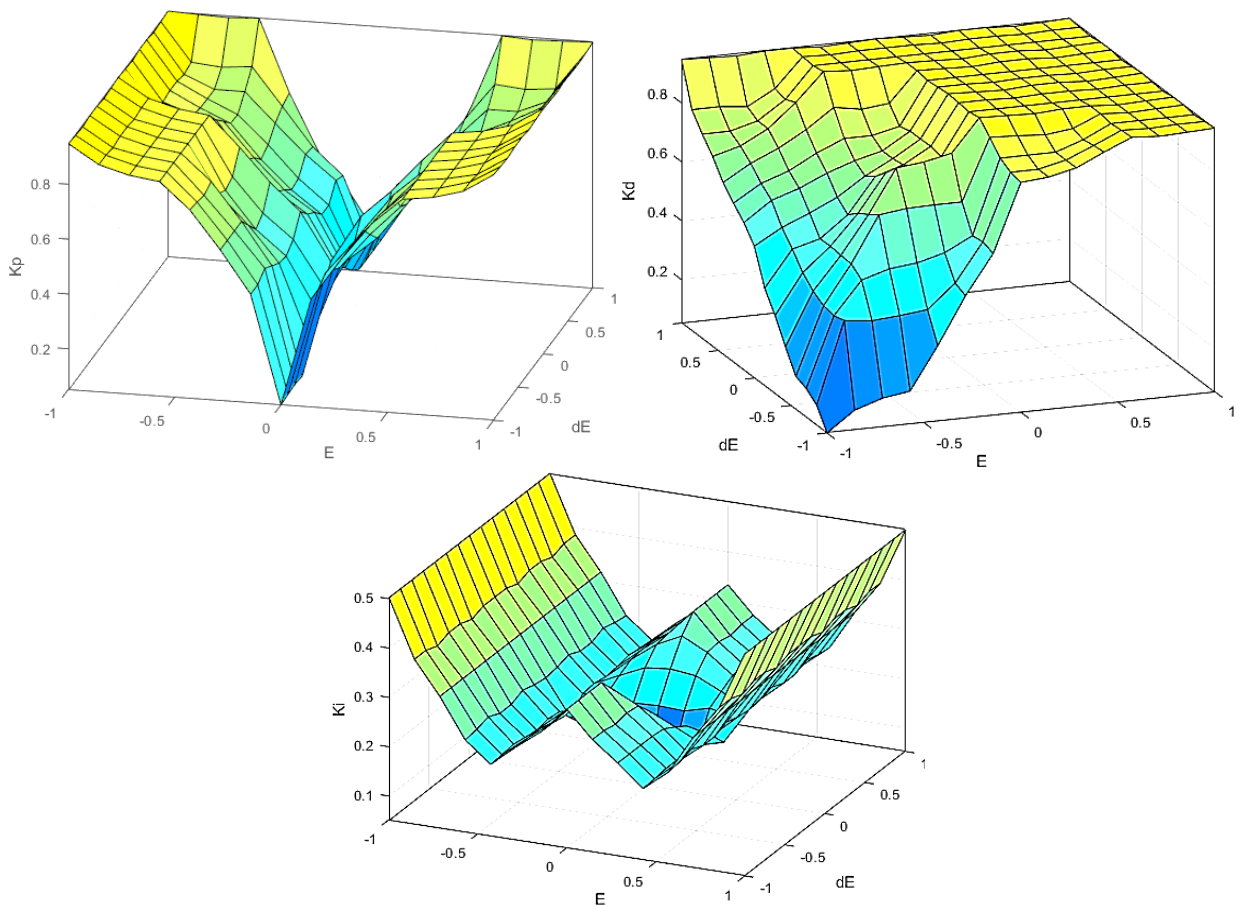


Рисунок 8.12 – Поверхні управління системою нечіткого висновку

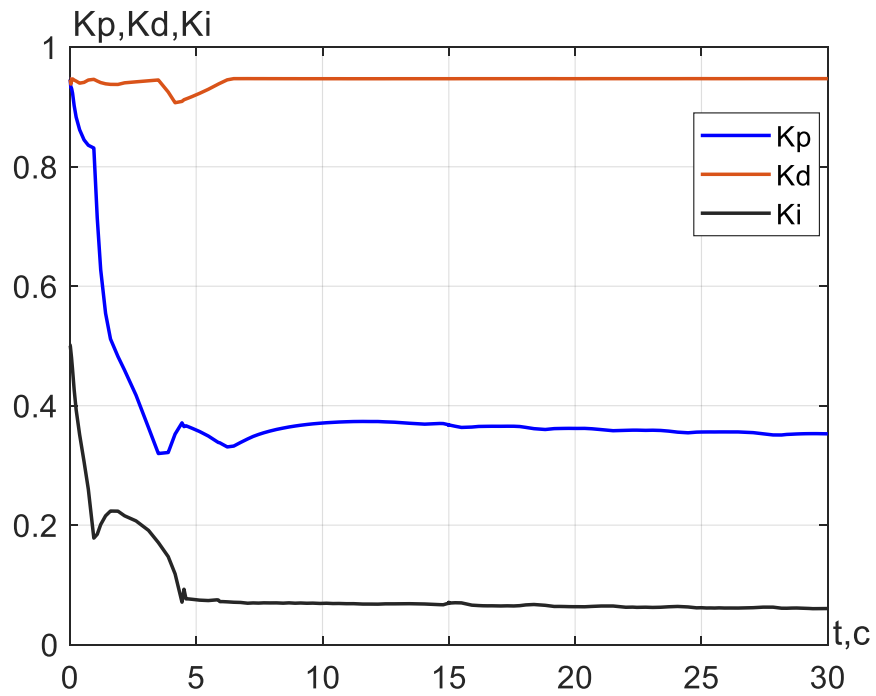


Рисунок 8.13 - Зміна коефіцієнтів ПІД-регулятора протягом перехідного процесу

На рис. 8.14 показаний перехідний процес у системі управління з нечітким супервізором.

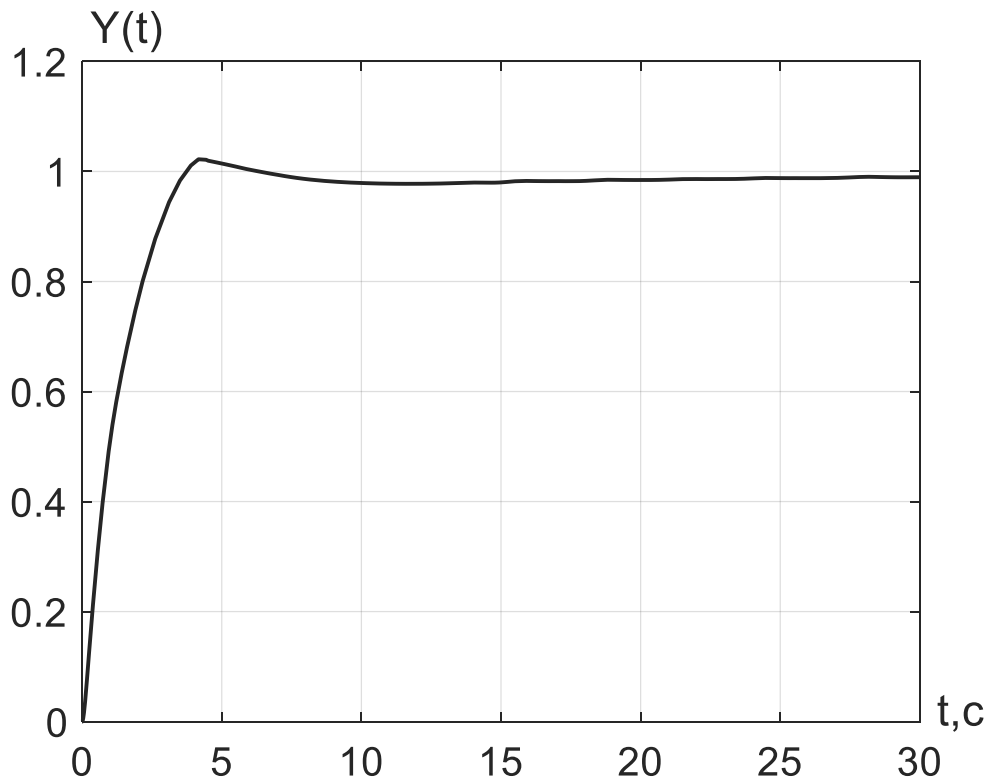


Рисунок 8.14 – Перехідний процес у системі управління з нечітким супервізором

8.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 8.5.

За вказаними у табл. 8.5 потрібно синтезувати нечіткий супервізор регулятора ПІД-типу.

Таблиця 8.5 - Варіанти індивідуальних завдань

№ вар	Передаточна функція об'єкту управління
1	$W(p) = \frac{1}{2p^2 + p + 1}$
2	$W(p) = \frac{1}{0.2p^2 + 0.1p + 1}$
3	$W(p) = \frac{1}{2p^2 + 0.5p + 1}$
4	$W(p) = \frac{1}{0.5p^2 + 0.25p + 1}$
5	$W(p) = \frac{1}{0.8p^2 + 0.4p + 1}$
6	$W(p) = \frac{1}{p^2 + 0.8p + 1}$
7	$W(p) = \frac{7}{5p^2 + 2.5p + 1}$
8	$W(p) = \frac{1}{2p^2 + 1.1p + 1}$
9	$W(p) = \frac{1}{6.2p^2 + 3p + 1}$
10	$W(p) = \frac{1}{2p^2 + p + 1}$
11	$W(p) = \frac{1}{3p^2 + 1.5p + 1}$
12	$W(p) = \frac{1}{0.5p^2 + 0.25p + 1}$
13	$W(p) = \frac{18}{8p^2 + 4p + 1}$
14	$W(p) = \frac{1}{1.5p^2 + 0.8p + 1}$
15	$W(p) = \frac{1}{5.6p^2 + 3p + 1}$
16	$W(p) = \frac{1}{5.2p^2 + 2.1p + 1}$

№ вар	Передаюча функція об'єкту управління
17	$W(p) = \frac{1}{2.7p^2 + 1.3p + 1}$
18	$W(p) = \frac{1}{0.2p^2 + 0.11p + 1}$
19	$W(p) = \frac{1}{2p^2 + 1.3p + 1}$
20	$W(p) = \frac{1}{7.2p^2 + 3.1p + 1}$

8.4 Контрольні питання

1. Що таке нечіткий супервізор?
2. На підставі яких міркувань відбувається синтез правил нечіткого супервізора?
3. Опишіть варіанти схем нечітких супервізорів.
4. Опишіть принципи опису дискретного НЛР ПД.
5. Опишіть принципи опису дискретного НЛР ПІ.

9 СУПЕРВІЗОРНЕ УПРАВЛІННЯ З ІНКРЕМЕНТАЛЬНОЮ ЗМІНОЮ КОЕФІЦІЄНТІВ РЕГУЛЯТОРА

Мета роботи: засвоєння методики синтезу супервізорного управління з інкрементальною зміною коефіцієнтів регулятора з використанням рекдактора системи нечіткого висновку

9.1 Основи супервізорного управління з інкрементальною зміною коефіцієнтів регулятора

Розглянемо другий варіант супервізорного управління, у якому відбувається інкрементальна зміна коефіцієнтів регулятора. Нехай розглядається ПД-регулятор, у якому нечіткий супервізор може зменшувати чи збільшувати коефіцієнти посилення залежно від характеру перехідного процесу (рис. 9.1).

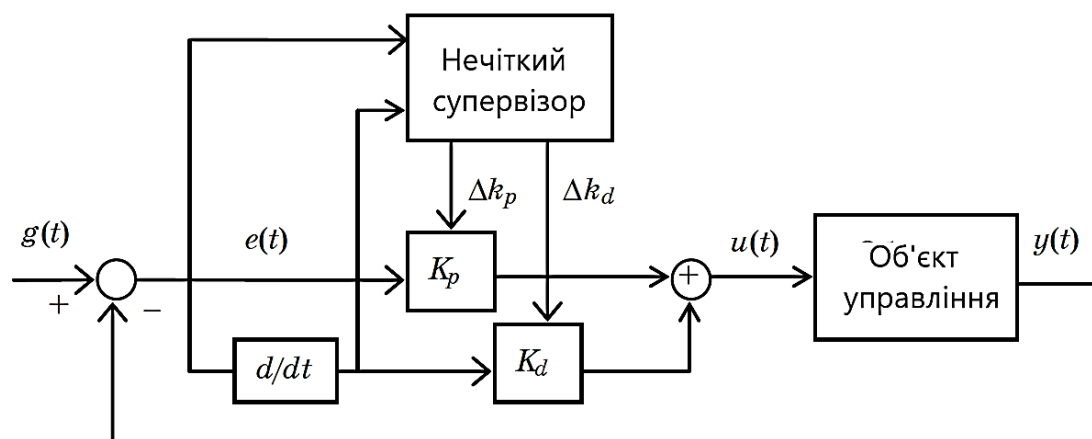


Рисунок 9.1 - Інкрементальний нечіткий супервізор

Таким чином, вихід ПД-регулятора описуватиметься формулою

$$u(t) = (k_p + \Delta k_p)e(t) + (k_d + \Delta k_d)de(t)/dt.$$

Нечітка система використовується для модифікації параметрів регулятора нижнього рівня (базового). Такий підхід може бути ефективним при керуванні нестационарним об'єктом, а також при неточно відомих параметрах об'єкта. Базовий регулятор може бути налаштований за стандартною методикою (наприклад, Зіглера – Ніколса), а нечіткий супервізор дозволяє варіювати коефіцієнти базового регулятора, покращуючи якість управління.

Основна ідея корекції коефіцієнтів ПД-регулятора у тому, щоб збільшувати k_p , коли керується помилка управління. Диференціальний коефіцієнт k_d змінюється пропорційно k_p , тому підтримується співвідношення k_d/k_p , отримане при початковому налаштуванні

регулятора. Такий закон корекції забезпечується формулами:

$$\Delta k_p = y(t)k_p;$$

$$\Delta k_d = y(t)k_d.$$

де $y(t)$ - сигнал, що виробляється нечітким супервізором.

Закон роботи нечіткого супервізора відрізняється від таблиці логічних змінних звичайного НЛР, його можна обґрунтувати, розглядаючи типовий графік перехідного процесу (рис. 8.4), який на фазовій площині набуває наступного вигляду (рис. 9.2).

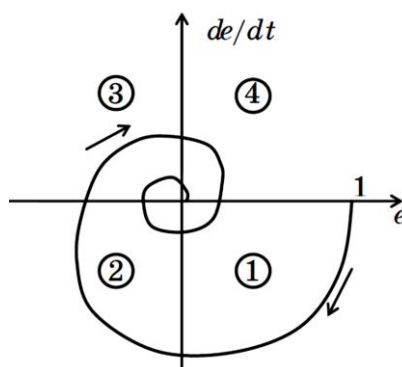


Рисунок 9.2 – Перехідний процес на фазовій площині

На початку перехідного процесу $e(t)$ велика позитивна та de/dt нульова. У регуляторі нижнього рівня використовуються номінальні параметри.

Потім $e(t)$ починає поступово зменшуватися, а de/dt зростає у негативному напрямку (зона 1). Вихід супервізора поступово збільшується від нуля до максимального значення, щоб зменшити час наростання.

У зоні 2 $e(t)$ негативна, і система повинна повільно рухатися вниз зменшення перерегулювання. Це досягається зменшенням контрольованого параметра, і вихід супервізора у цій галузі негативний.

Як тільки перерегулювання досягло пікового значення, знак de/dt стає позитивним (зона 3), і вихід супервізора повинен збільшувати своє значення, доки система не потрапить до регіону 4.

У зоні 4 $e(t)$ знову позитивна, і вихідний сигнал об'єкта наближається до стану, що встановився. Тут потрібно поступово зменшувати посилення, щоб обмежити перерегулювання.

Якщо $e(t)$ нульова, вихід супервізора протилежний за знаком значення de/dt .

Якщо de/dt нульова, то вихід супервізора збігається за знаком з $e(t)$, і має бути більшим у області малих $e(t)$ зменшення статичної помилки.

Зроблені зауваження дозволяють описати поведінку нечіткого

супервізора за допомогою таблиці лінгвістичних змінних, що показана на рис. 9.3.

	$e(t)$										
	NL	NB	NM	NS	ZE	PS	PM	PB	PL		
NL	NL	NL	NL	NL	PB	PB	PB	PM	PM	①	
NB	NL	NL	NL	NB	PB	PB	PB	PM	PM		
NM	NL	NL	NB	NM	PB	PB	PM	PM	PS		
NS	NL	NB	NM	NS	PL	PB	PM	PS	PS		
de/dt ZE	ZE	NS	NM	NB	PL	PB	PM	PS	ZE	②	
PS	PS	PS	PM	PB	NL	NS	NM	NB	NL		
PM	PS	PM	PM	PB	NB	NM	NB	NL	NL		④
PB	PM	PM	PB	PB	NB	NB	NL	NL	NL		
PL	PM	PM	PB	PB	NB	NL	NL	NL	NL		

Рисунок 9.3 – Таблиця лінгвістичних правил нечіткого супервізора (цифрами позначені регіони фазової площини)

При опису передбачається використання одноманітного лінгвістичного опису нормалізованих $e(t)$, de/dt та $y(t)$ у вигляді, показаному на рис. 9.4, де використані такі скорочення: NL – negative large (негативне дуже велике), NB – negative big (негативне велике), NM – negative medium (негативне середнє), NS – negative small (негативне мале), ZE – zero (нульове) PS – positive small (позитивне мале), PM – positive medium (позитивне середнє), PB – positive big (позитивне велике), PL – positive large (позитивне дуже велике).

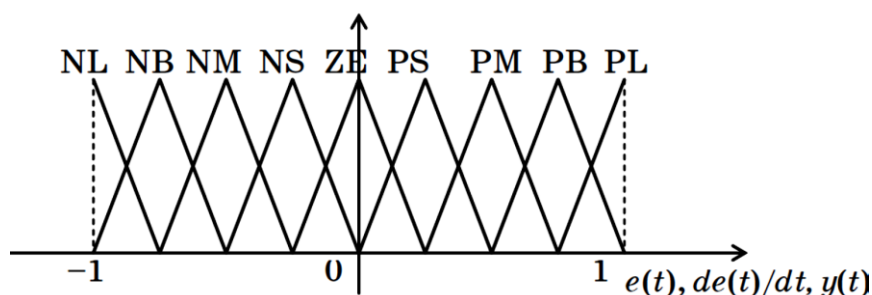


Рисунок 9.4 – Лінгвістичний опис помилки та її похідної

9.2 Приклад синтезу супервізорного управління з інкрементальною зміною коефіцієнтів регулятора

Розглянемо приклад використання нечіткого супервізора. Нехай є система управління з ПД-регулятором, коефіцієнти якого розраховані за методом Зіглера – Ніколса (див. рис. 9.5).

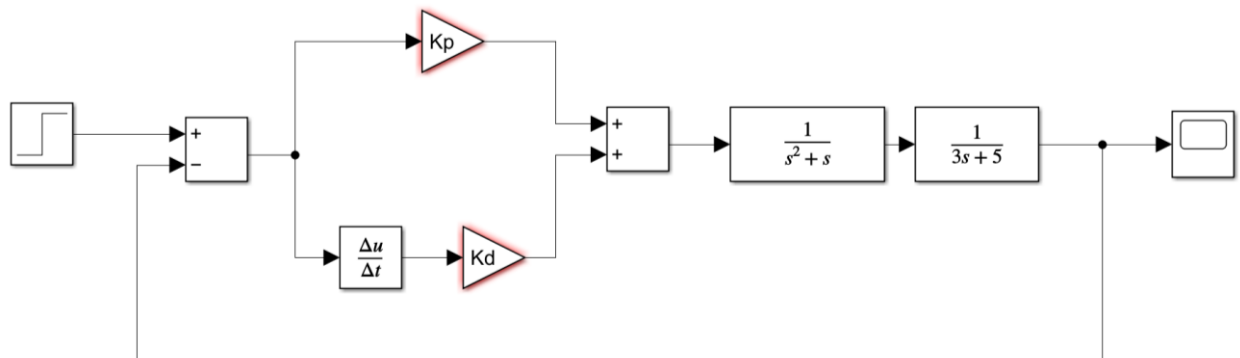


Рисунок 9.5 – Структурна схема математичної моделі системи управління з ПД-регулятором у MatLab Simulink

Налаштування регулятора методом Зіглера – Ніколса полягає в тому, щоб обчислити або підібрати оптимальні значення коефіцієнтів:

- K_P - коефіцієнт посилення пропорційної складової;
- K_I - коефіцієнт посилення інтегруючої складової;
- K_D - коефіцієнт посилення диференціюючої складової.

За великим рахунком, при використанні ПД, ПІ, ПІД-регулятора необхідно побудувати модель усієї системи в цілому та математично обчислити необхідні значення коефіцієнтів. У такому разі значення можна розрахувати точно.

Але на практиці математичний розрахунок коефіцієнтів - завдання далеко не тривіальне і вимагає глибоких знань теорії автоматичного управління, тому в більшості випадків використовуються інші, спрощені методи налаштування.

Метод полягає у послідовному виконанні наступних операцій:

- коефіцієнти регулятора дорівнюємо 0.
- встановлюємо певне цільове значення регульованого (вихідного) параметра.
- поступово починаємо збільшувати пропорційний коефіцієнт та стежимо за реакцією системи.
- за певного значення $K=K_P$ виникнуть незагасні коливання регульованої величини.
- фіксуємо це значення, а також період коливань системи.

У цьому практична частина методу закінчується. З отриманих значень розраховуємо коефіцієнти:

$$K_P = 0.6K;$$

$$K_I = \frac{2K_P}{T};$$

$$K_D = \frac{K_P T}{8}.$$

де K - коефіцієнт пропорційної складової, у якому виникли коливання, T – період цих коливань.

Налаштуємо коефіцієнти ПД регулятора методом Зиглера-Нікольса для даного випадку.

1. Задаємо на математичній моделі $K_D = 0$, а коефіцієнт пропорційної складової регулятора збільшуємо до значення при якому у системі виникає коливальний перехідний процес з однаковим періодом коливань. У даному випадку такі коливання у системі управління настають при значенні $K=K_P=13$. (див. рис. 9.6).

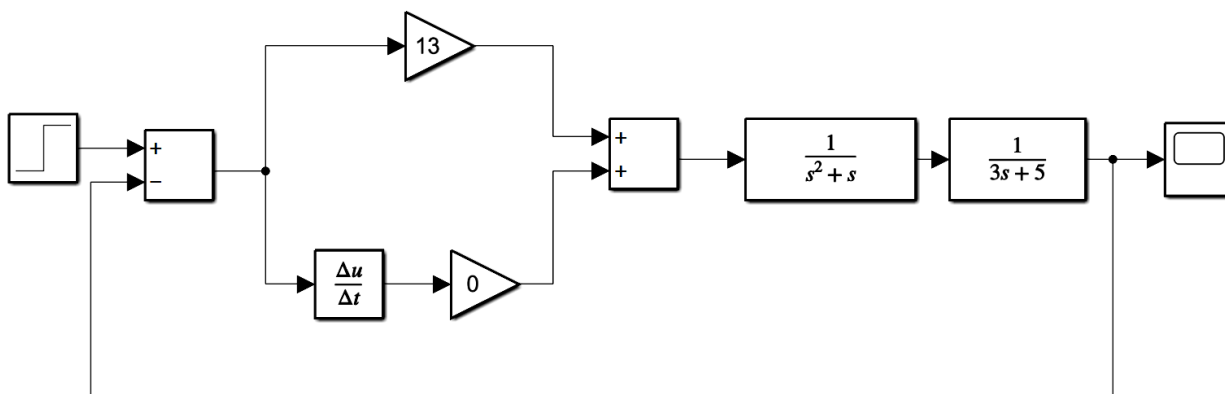


Рисунок 9.6 – Структурна схема налаштування регулятора математичної моделі системи управління у MatLab Simulink

2. На графіку перехідного процесу визначається період коливань $T=4,9$ с (див. рис. 9.7).

3. Розраховуємо коефіцієнти передачі ПД-регулятора

$$K_P = 0.6K = 0,6 \cdot 13 = 7,8;$$

$$K_D = \frac{K_P T}{8} = \frac{7,8 \cdot 4,9}{8} = 4,8.$$

Заносимо результати розрахунку коефіцієнтів ПД-регулятора до математичної моделі системи (див. рис 9.8.)

Перехідний процес для системи управління наведеної на рисунку 9.8 має вигляд який зображено на рис. 9.9.

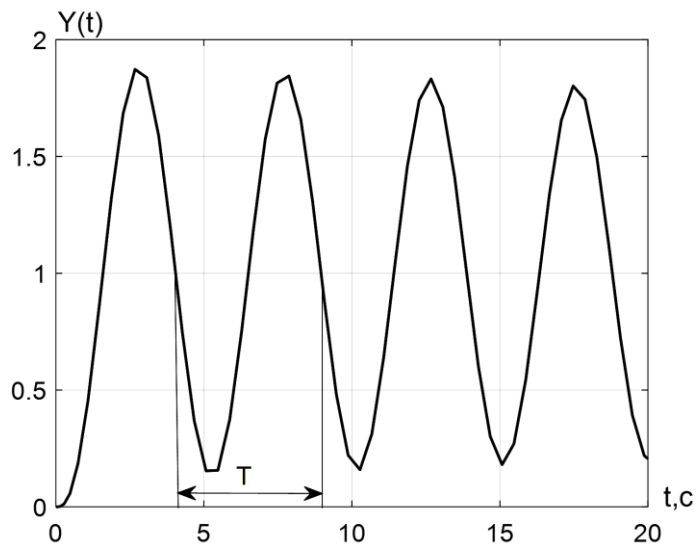


Рисунок 9.7 – Графік перехідного процесу системи управління при $K=K_P=13$

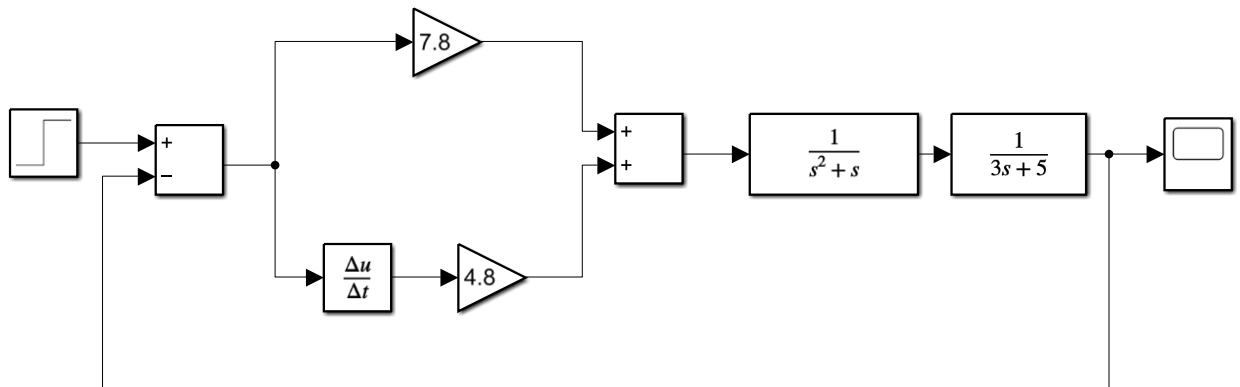


Рисунок 9.8 - Структурна схема математичної моделі системи управління з ПД-регулятором

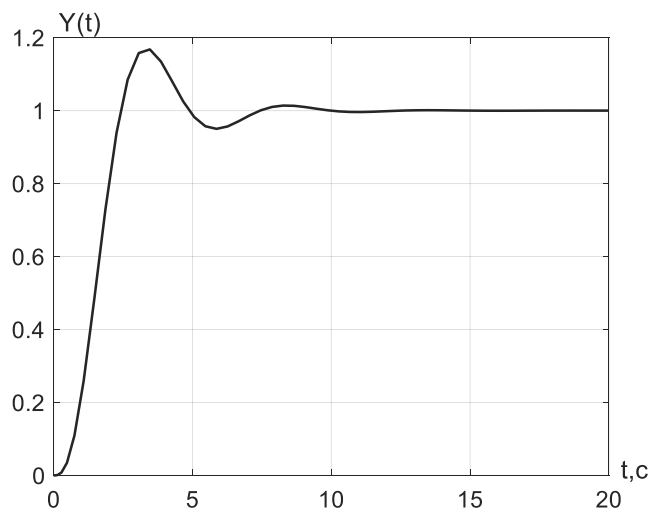


Рисунок 9.9 – Перехідний процес с системі управління з ПД-регулятором

Проведемо синтез ПД-регулятора з нечітким супервізором в якому відбувається інкрементальна зміна коефіцієнтів регулятора нечітким супервізором ПД регулятора. Сформуємо в Simulink MatLab модель системи управління (див. рис. 9.10).

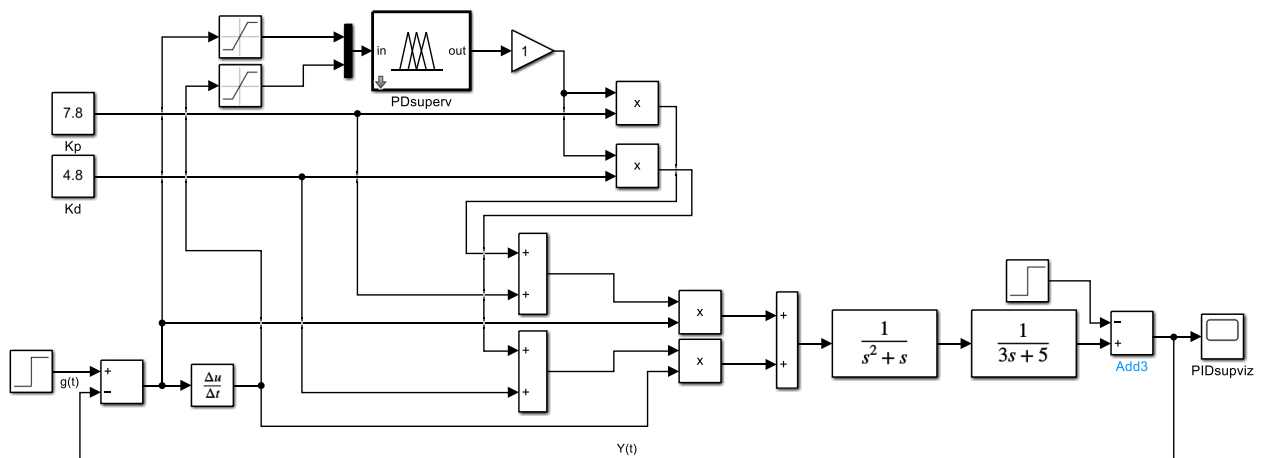


Рисунок 9.10 – Структурна схема математичної моделі системи управління з нечітким супервізором інкрементального типу

Fuzzy Logic Controller надаємо Fis name: PDsupv2 (ім'я може бути будь-яке).

Викликається редактор системи нечіткого висновку (Fuzzy Inference System – FIS) у MatLab командою

>> fuzzy

Дале проводиться налаштування головного вікна редактора нечіткої логічної системи (див. рис. 9.11).

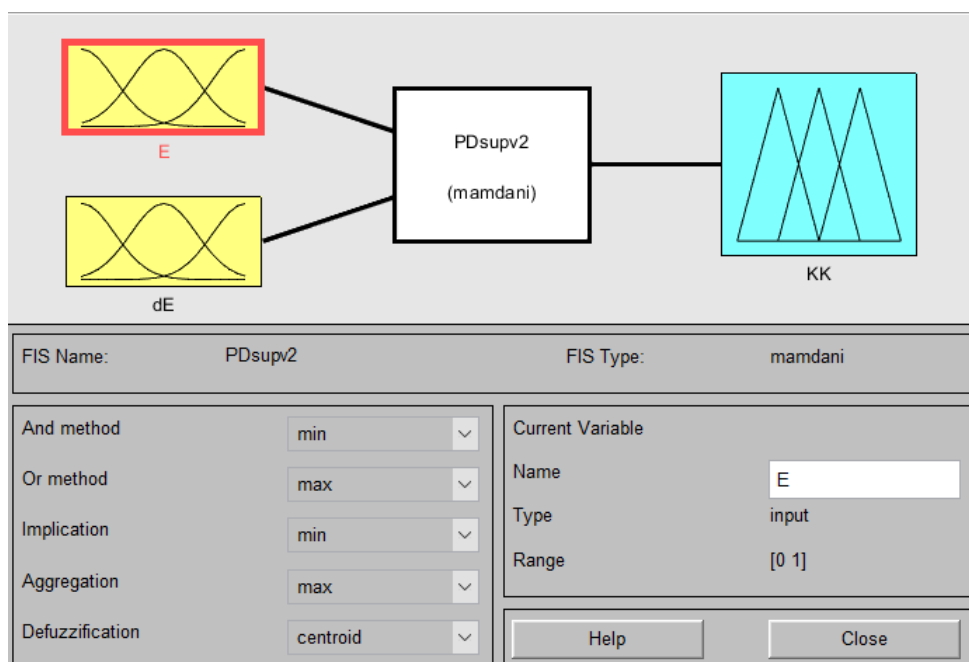


Рисунок 9.11 – Налаштування інтерфейсу FIS editor

У *FIS editor* задається опис систему нечіткого логічного висновку *Mamdani*. Для створюваної системи обирається вид логічного зв'язку (*And method – min*) та (*Or method – max*), вид імплікації (*Implication – min*), спосіб агрегування висновків правил (*Aggregation – max*) та метод дефазифікації (*Defuzzification – centroid*).

У меню *Edit* послідовно додаємо 2 вхідні змінні з розміром базової шкали (*Range= [-1 1]*) та вихідну змінну з розміром базової шкали (*Range= [0 1]*).

Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної змінної трикутну функцію приналежності.

Нехай терми лінгвістичної змінної «Помилка управління» розміщуються відповідно до рис. 9.12 з 9 параметрами функцій приналежності:

Name = "NLe"; Type = "trimf"; Param = [-1.125 -1 -0.75];
 Name = "NBe"; Type = "trimf"; Param = [-1 -0.75 -0.5];
 Name = "NMe"; Type = "trimf"; Param = [-0.75 -0.5 -0.25];
 Name = "NSe"; Type = "trimf"; Param = [-0.5 -0.25 0];
 Name = "Ze"; Type = "trimf"; Param = [-0.25 0 0.25];
 Name = "PSe"; Type = "trimf"; Param = [0 0.25 0.5].
 Name = "PMe"; Type = "trimf"; Param = [0.25 0.5 0.75].
 Name = "PBe"; Type = "trimf"; Param = [0.5 0.75 1].
 Name = "PLe"; Type = "trimf"; Param = [0.75 1 1.25].

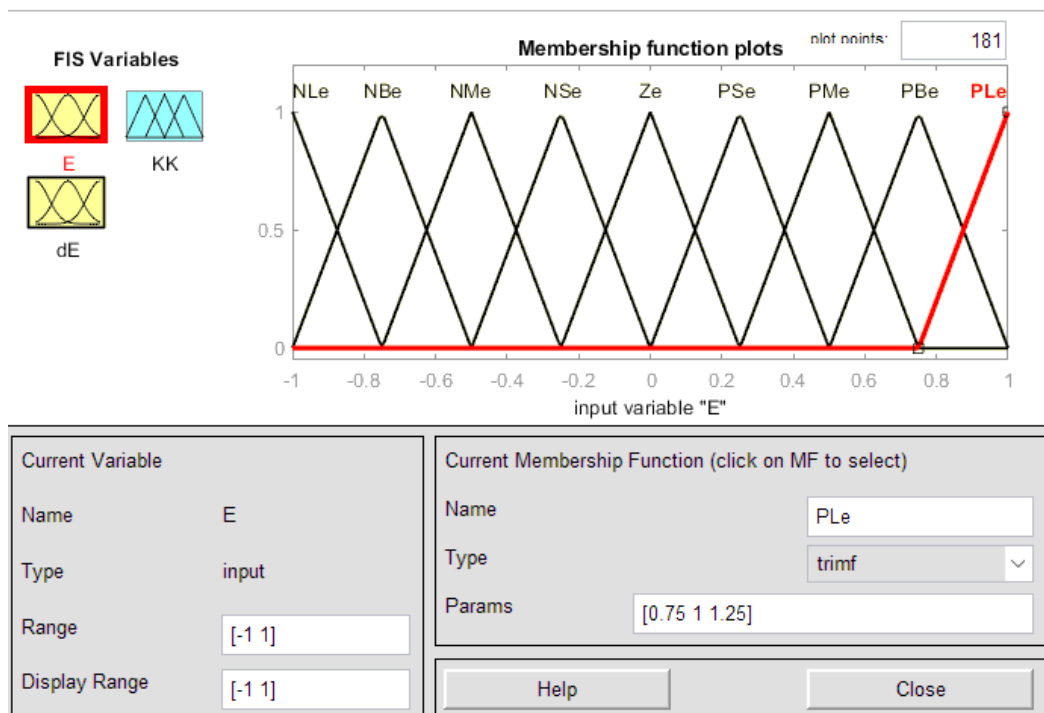


Рисунок 9.12 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Терми вхідної лінгвістичної змінної «Похідна помилки управління» формуємо рівномірно. Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. 3 параметрами:

Name = "NLdE"; Type = "trimf"; Param = [-1.125 -1 -0.75];
 Name = "NBdE"; Type = "trimf"; Param = [-1 -0.75 -0.5];
 Name = "NMdE"; Type = "trimf"; Param = [-0.75 -0.5 -0.25];
 Name = "NSdE"; Type = "trimf"; Param = [-0.5 -0.25 0];
 Name = "ZdE"; Type = "trimf"; Param = [-0.25 0 0.25];
 Name = "PSdE"; Type = "trimf"; Param = [0 0.25 0.5].
 Name = "PMdE"; Type = "trimf"; Param = [0.25 0.5 0.75].
 Name = "PBdE"; Type = "trimf"; Param = [0.5 0.75 1].
 Name = "PLdE"; Type = "trimf"; Param = [0.75 1 1.25].

Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління» на рис. 9.13.

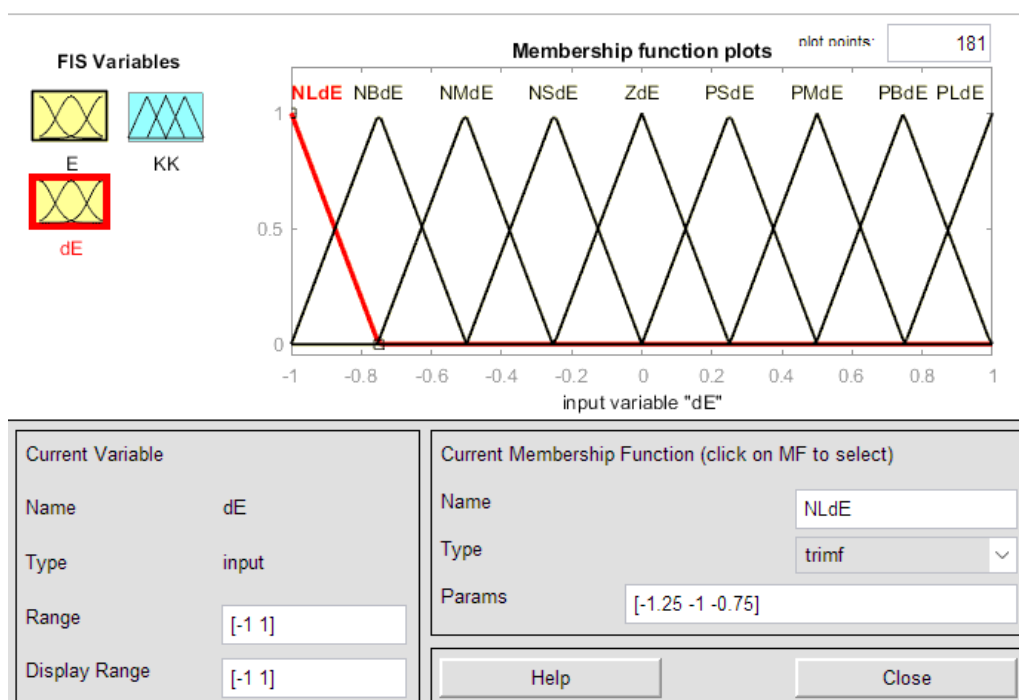


Рисунок 9.13 – Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління»

Для опису вихідної логічної змінної у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності.

Name = "NL"; Type = "trimf"; Param = [-1.125 -1 -0.75];
 Name = "NB"; Type = "trimf"; Param = [-1 -0.75 -0.5];
 Name = "NM"; Type = "trimf"; Param = [-0.75 -0.5 -0.25];

Name = "NS"; Type = "trimf"; Param = [-0.5 -0.25 0];
 Name = "Z"; Type = "trimf"; Param = [-0.25 0 0.25];
 Name = "PS"; Type = "trimf"; Param = [0 0.25 0.5].
 Name = "PM"; Type = "trimf"; Param = [0.25 0.5 0.75].
 Name = "PB"; Type = "trimf"; Param = [0.5 0.75 1].
 Name = "PL"; Type = "trimf"; Param = [0.75 1 1.25].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 9.14.

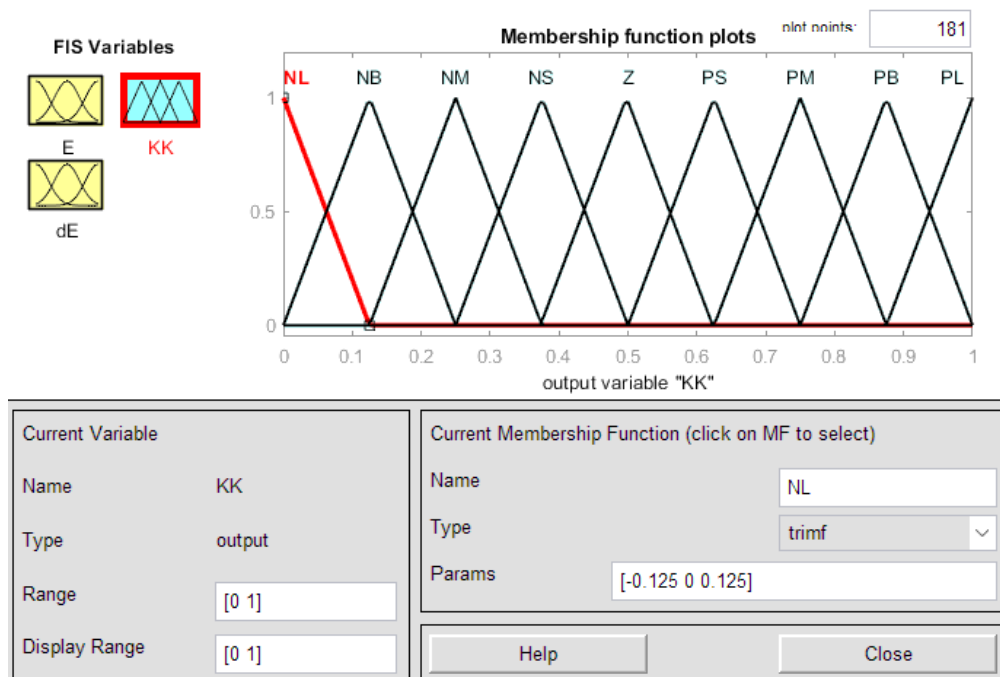


Рисунок 9.14 – Результат налаштування функцій приналежності вихідної змінної

Після опису вхідні та вихідні лінгвістичних змінних, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку 25. Управляючі правила сформульовано згідно з рис. 9.3:

1. if (E is NLe) and (dE is NLdE) then (KK is NL) (1)
2. if (E is NLe) and (dE is NBdE) then (KK is NL) (1)
3. if (E is NLe) and (dE is NMdE) then (KK is NL) (1)
4. if (E is NLe) and (dE is NSdE) then (KK is NL) (1)
5. if (E is NLe) and (dE is ZdE) then (KK is PB) (1)
6. if (E is NLe) and (dE is PSdE) then (KK is PB) (1)
7. if (E is NLe) and (dE is PMdE) then (KK is PB) (1)
8. if (E is NLe) and (dE is PBdE) then (KK is PM) (1)
9. if (E is NLe) and (dE is PLE) then (KK is PM) (1)
10. if (E is NBe) and (dE is NLdE) then (KK is NL) (1)
11. if (E is NBe) and (dE is NBdE) then (KK is NL) (1)
12. if (E is NBe) and (dE is NMdE) then (KK is NL) (1)

- 
13. if (E is NBe) and (dE is NSdE) then (KK is NB) (1)
 14. if (E is NBe) and (dE is ZdE) then (KK is PB) (1)
 15. if (E is NBe) and (dE is PSdE) then (KK is PB) (1)
 16. if (E is NBe) and (dE is PMdE) then (KK is PB) (1)
 17. if (E is NBe) and (dE is PBdE) then (KK is PM) (1)
 18. if (E is NBe) and (dE is PLE) then (KK is PM) (1)
 19. if (E is NMe) and (dE is NLdE) then (KK is NL) (1)
 20. if (E is NMe) and (dE is NBdE) then (KK is NL) (1)
 21. if (E is NMe) and (dE is NMdE) then (KK is NB) (1)
 22. if (E is NMe) and (dE is NSdE) then (KK is NM) (1)
 23. if (E is NMe) and (dE is ZdE) then (KK is PB) (1)
 24. if (E is NMe) and (dE is PSdE) then (KK is PB) (1)
 25. if (E is NMe) and (dE is PMdE) then (KK is PM) (1)
 26. if (E is NMe) and (dE is PBdE) then (KK is PM) (1)
 27. if (E is NMe) and (dE is PLE) then (KK is PS) (1)
 28. if (E is NSe) and (dE is NLdE) then (KK is NL) (1)
 29. if (E is NSe) and (dE is NBdE) then (KK is NB) (1)
 30. if (E is NSe) and (dE is NMdE) then (KK is NM) (1)
 31. if (E is NSe) and (dE is NSdE) then (KK is NS) (1)
 32. if (E is NSe) and (dE is ZdE) then (KK is PL) (1)
 33. if (E is NSe) and (dE is PSdE) then (KK is PB) (1)
 34. if (E is NSe) and (dE is PMdE) then (KK is PM) (1)
 35. if (E is NSe) and (dE is PBdE) then (KK is PS) (1)
 36. if (E is NSe) and (dE is PLE) then (KK is PS) (1)
 37. if (E is Ze) and (dE is NLdE) then (KK is Z) (1)
 38. if (E is Ze) and (dE is NBdE) then (KK is NS) (1)
 39. if (E is Ze) and (dE is NMdE) then (KK is NM) (1)
 40. if (E is Ze) and (dE is NSdE) then (KK is NB) (1)
 41. if (E is Ze) and (dE is ZdE) then (KK is PL) (1)
 42. if (E is Ze) and (dE is PSdE) then (KK is PB) (1)
 43. if (E is Ze) and (dE is PMdE) then (KK is PM) (1)
 44. if (E is Ze) and (dE is PBdE) then (KK is PS) (1)
 45. if (E is Ze) and (dE is PLE) then (KK is Z) (1)
 46. if (E is PSe) and (dE is NLdE) then (KK is PS) (1)
 47. if (E is PSe) and (dE is NBdE) then (KK is PS) (1)
 48. if (E is PSe) and (dE is NMdE) then (KK is PM) (1)
 49. if (E is PSe) and (dE is NSdE) then (KK is PB) (1)
 50. if (E is PSe) and (dE is ZdE) then (KK is NL) (1)
 51. if (E is PSe) and (dE is PSdE) then (KK is NS) (1)
 52. if (E is PSe) and (dE is PMdE) then (KK is NM) (1)
 53. if (E is PSe) and (dE is PBdE) then (KK is NB) (1)
 54. if (E is PSe) and (dE is PLE) then (KK is NL) (1)
 55. if (E is PMe) and (dE is NLdE) then (KK is PS) (1)
 56. if (E is PMe) and (dE is NBdE) then (KK is PM) (1)
 57. if (E is PMe) and (dE is NMdE) then (KK is PM) (1)

58. if (E is PMe) and (dE is NSdE) then (KK is PB) (1)
59. if (E is PMe) and (dE is ZdE) then (KK is NB) (1)
60. if (E is PMe) and (dE is PSdE) then (KK is NM) (1)
61. if (E is PMe) and (dE is PMdE) then (KK is NB) (1)
62. if (E is PMe) and (dE is PBdE) then (KK is NL) (1)
63. if (E is PMe) and (dE is PLE) then (KK is NL) (1)
64. if (E is PBe) and (dE is NLdE) then (KK is PM) (1)
65. if (E is PBe) and (dE is NBdE) then (KK is PM) (1)
66. if (E is PBe) and (dE is NMdE) then (KK is PB) (1)
67. if (E is PBe) and (dE is NSdE) then (KK is PB) (1)
68. if (E is PBe) and (dE is ZdE) then (KK is NB) (1)
69. if (E is PBe) and (dE is PSdE) then (KK is NB) (1)
70. if (E is PBe) and (dE is PMdE) then (KK is NL) (1)
71. if (E is PBe) and (dE is PBdE) then (KK is NL) (1)
72. if (E is PBe) and (dE is PLE) then (KK is NL) (1)
73. if (E is PLe) and (dE is NLdE) then (KK is PM) (1)
74. if (E is PLe) and (dE is NBdE) then (KK is PM) (1)
75. if (E is PLe) and (dE is NMdE) then (KK is PB) (1)
76. if (E is PLe) and (dE is NSdE) then (KK is PB) (1)
77. if (E is PLe) and (dE is ZdE) then (KK is NB) (1)
78. if (E is PLe) and (dE is PSdE) then (KK is NL) (1)
79. if (E is PLe) and (dE is PMdE) then (KK is NL) (1)
80. if (E is PLe) and (dE is PBdE) then (KK is NL) (1)
81. if (E is PLe) and (dE is PLE) then (KK is NL) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 9.15).

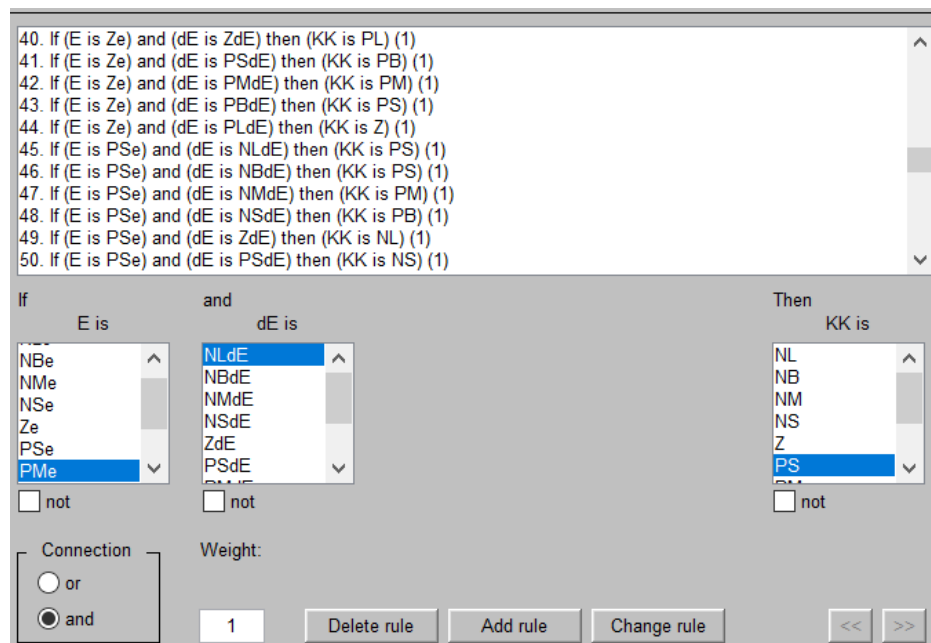


Рисунок 9.15 - Налаштовування опису правил функціонування нечіткого інкрементального супервізора НЛР_ПД

Посилки у правилах пов'язані (connection) за допомогою операції *and*. Введене правило забезпечене ваговим коефіцієнтом ($Weight=1$). Після опису правил можливо за допомогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 11.152).

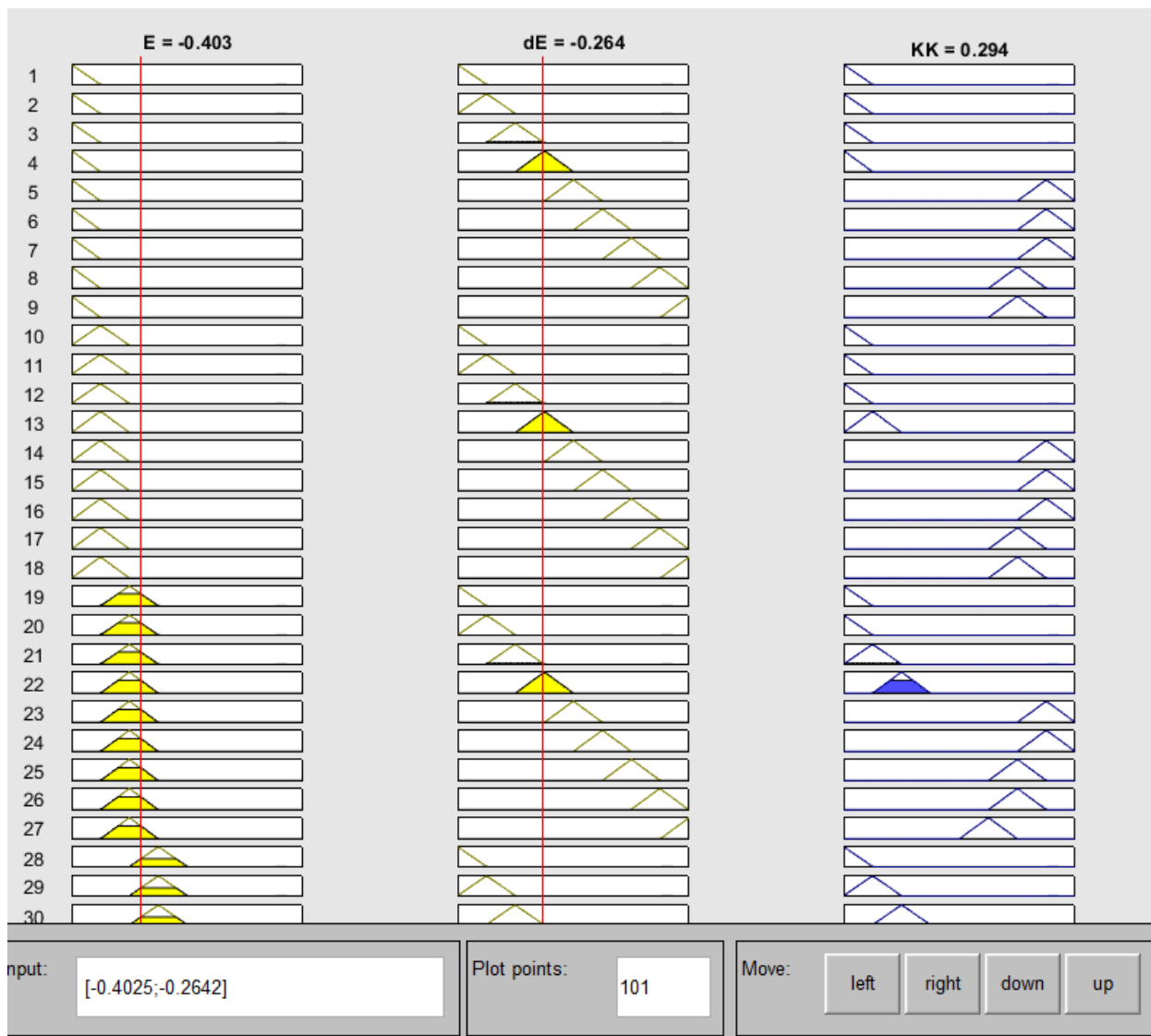


Рисунок 9.16 - Робота системи нечіткого інкрементального супервізора НЛР_ПД

- інтерфейсу *Surface Viewer* можливо переглянути графічне подання закону управління (рис. 9.17).

Після процедур налаштування нечіткого інкрементального супервізора НЛР_ПД у редакторі системи нечіткого висновку здійснює запис результатів налаштування у Fuzzy Logic Controller. Для цього у вкладці *File* здійснюємо *Export* в математичну модуль нечіткого контролера *From Workspase* вказавши ім'я *Fuzzy Logic Controller*.

На рис. 9.18 представлені графіки перехідного процесу у системі управління з нечітким інкрементальним супервізором НЛР_ПД. На

математичної моделі (див. рис. 9.10) у момент часу 15 с додано збурення. Як видно з графіку перехідного процесу, що спроектована система управління повністю компенсує збурюючі впливи.

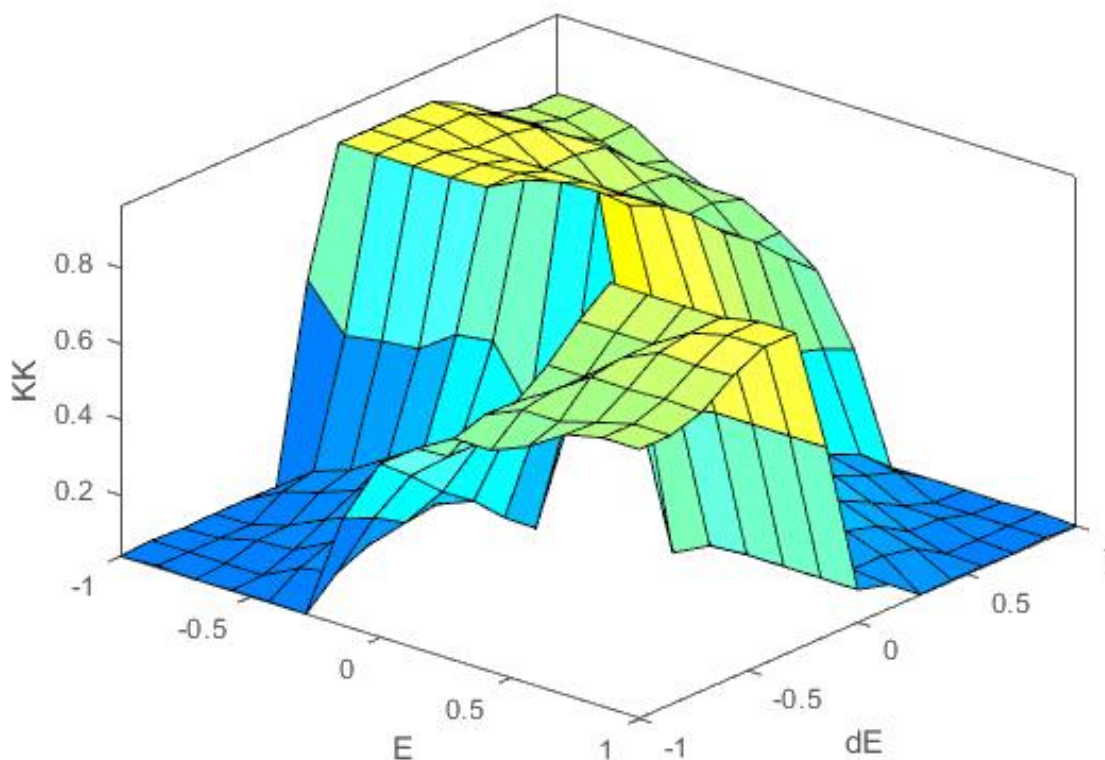


Рисунок 9.17 - Графічне подання закону управління

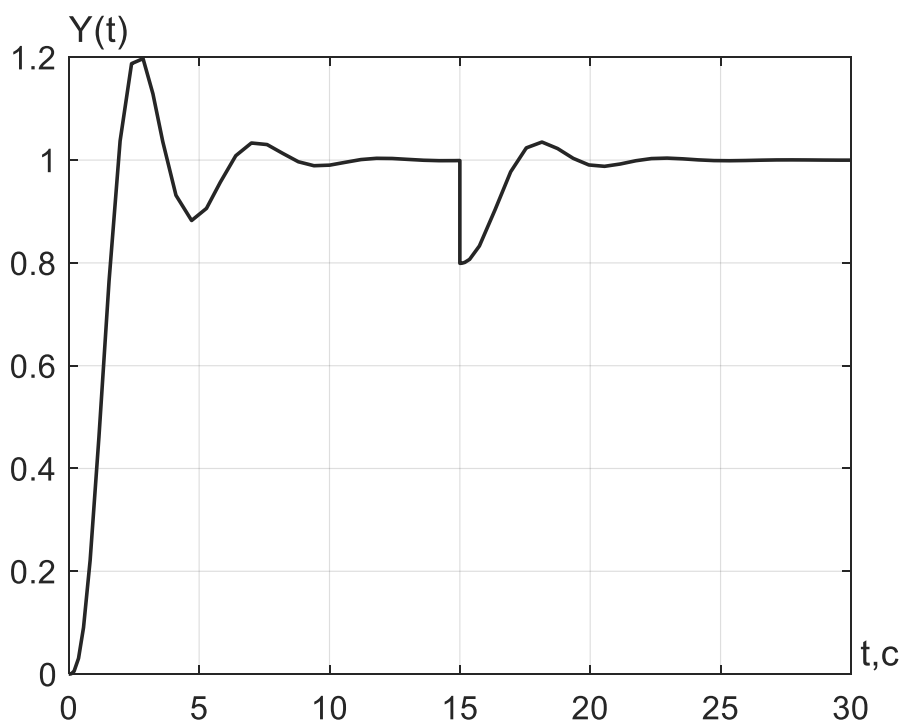


Рисунок 9.18 – Графік перехідного процесу у системі управління з нечітким інкрементальним супервізором ПД-типу

9.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 9.5.

За вказаними у табл. 9.5 потрібно синтезувати нечіткий інкрементальний супервізор регулятора ПД-типу.

Таблиця 9.5 - Варіанти індивідуальних завдань

№ вар	Передаюча функція об'єкту управління
1	$W_1(p) = \frac{1}{2p^2 + p}; W_2(p) = \frac{1}{3p + 1}$
2	$W_1(p) = \frac{1}{p^2 + p}; W_2(p) = \frac{1}{3p + 1}$
3	$W_1(p) = \frac{1}{3p^2 + p}; W_2(p) = \frac{1}{p + 1}$
4	$W_1(p) = \frac{1}{2,5p^2 + p}; W_2(p) = \frac{1}{1,5p + 1}$
5	$W_1(p) = \frac{1}{3p^2 + p}; W_2(p) = \frac{1}{2p + 1}$
6	$W_1(p) = \frac{1}{4p^2 + p}; W_2(p) = \frac{1}{3p + 1}$
7	$W_1(p) = \frac{1}{2p^2 + 0,2p}; W_2(p) = \frac{1}{3p + 1}$
8	$W_1(p) = \frac{1}{4p^2 + 0,4p}; W_2(p) = \frac{1}{3p + 1}$
9	$W_1(p) = \frac{1}{2p^2 + 0,1p}; W_2(p) = \frac{1}{2p + 1}$
10	$W_1(p) = \frac{1}{5p^2 + 0,5p}; W_2(p) = \frac{1}{p + 1}$
11	$W_1(p) = \frac{1}{0,2p^2 + 0,01p}; W_2(p) = \frac{1}{2p + 1}$
12	$W_1(p) = \frac{1}{3,5p^2 + 0,35p}; W_2(p) = \frac{1}{p + 1}$
13	$W_1(p) = \frac{1}{2,2p^2 + 0,02p}; W_2(p) = \frac{1}{0,1p + 1}$
14	$W_1(p) = \frac{1}{2,5p^2 + 0,25p}; W_2(p) = \frac{1}{4p + 1}$
15	$W_1(p) = \frac{1}{4p^2 + p}; W_2(p) = \frac{1}{2,5p + 1}$
16	$W_1(p) = \frac{1}{1,6p^2 + 0,36p}; W_2(p) = \frac{1}{p + 1}$

№ вар	Передаюча функція об'єкту управління
17	$W_1(p) = \frac{1}{4,4p^2 + 0,44p}; W_2(p) = \frac{1}{2,2p + 1}$
18	$W_1(p) = \frac{1}{1,5p^2 + 0,125p}; W_2(p) = \frac{1}{2p + 1}$
19	$W_1(p) = \frac{1}{3,2p^2 + 0,2p}; W_2(p) = \frac{1}{p + 1}$
20	$W_1(p) = \frac{1}{5p^2 + 0,25p}; W_2(p) = \frac{1}{0,2p + 1}$

9.4 Контрольні питання

1. Що таке нечіткий супервізор?
2. На підставі яких міркувань відбувається синтез правил нечіткого супервізора?
3. Опишіть варіанти схем нечітких супервізорів.
4. Опишіть принципи опису дискретного НЛР ПД.
5. Опишіть принципи опису дискретного НЛР ПІ.

10 СУПЕРВІЗОРНЕ УПРАВЛІННЯ З КОРЕКЦІЄЮ КОЕФІЦІЄНТІВ ПІД-РЕГУЛЯТОРА НА ПІДСТАВІ ІНФОРМАЦІЇ ПРО ПОМИЛКУ ТА ЇЇ ПОХІДНУ

Мета роботи: засвоєння методики синтезу супервізорного управління з корекцією коефіцієнтів ПІД-регулятора на підставі інформації про помилку та її похідну з використанням редактора системи нечіткого висновку

10.1 Основи організації нечіткого супервізора з корекцією коефіцієнтів ПІД-регулятора на підставі інформації про помилку та її похідну

Припустимо, що відомі діапазони, в яких можуть змінюватися коефіцієнти K_p та K_d :

$$K_p \in [K_{p.min}, K_{p.max}],$$

$$K_d \in [K_{d.min}, K_{d.max}]$$

При виборі діапазону зміни коефіцієнтів можна керуватися такими формулами

$$K_{p.min} = 0.32K_u; \quad K_{p.max} = 0.6K_u \\ K_{d.min} = 0.08K_uT_u; \quad K_{d.max} = 0.15K_uT_u$$

де K_u – мінімальне значення коефіцієнта посилення П-регулятора, у якому виникають автоколивання з періодом T_u .

Вважатимемо, що значення коефіцієнтів нормалізовані:

$$K_p' = \frac{K_p - K_{p.min}}{K_{p.max} - K_{p.min}}; \quad K_d' = \frac{K_d - K_{d.min}}{K_{d.max} - K_{d.min}}$$

відкіля

$$K_p = (K_{p.max} - K_{p.min})K_p' + K_{p.min};$$

$$K_d = (K_{d.max} - K_{d.min})K_d' + K_{d.min}.$$

Згідно з методикою Зіглера - Ніколса, постійна часу інтегрування залежить від постійного часу диференціювання:

$$T_i = \alpha T_d,$$



ТАКИМ ЧИНОМ

$$K_i = \frac{K_p}{\alpha T_d} = \frac{(K_p)^2}{\alpha K_d}$$

Супервізор містить безліч правил виду:

Якщо $(e(t) = A_i)$ та $(de(t)/dt = B_i)$, то $(K_p = C_i)$ і $(K_d = D_i)$ та $\alpha = \alpha_i$;
 $i = 1, 2, \dots, m$,

де A_i, B_i, C_i і D_i – нечіткі множини, α_i – константа.

Лінгвістичне опис $e(t)$ та $de(t)/dt$ показано на рис. 10.1.

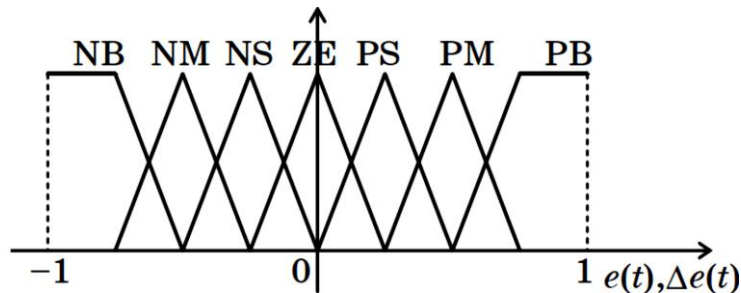


Рисунок 10.1 – Лінгвістичний опис помилки та її прирощення

Лінгвістичний опис K_p та K_d показано на рис. 10.2, де використано лише два терми: *Big* та *Small* («велике» і «мале»), для їх опису можуть бути використані дзвонові функції приналежності.

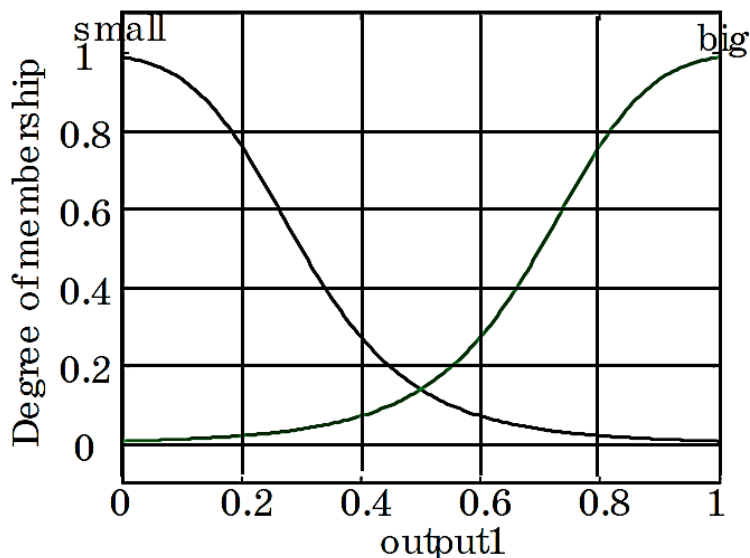


Рисунок 10.2 - Лінгвістичний опис K_p та K_d

Для опису правил зміни коефіцієнтів розглянемо характерні точки типового перехідного процесу (рис. 10.3).

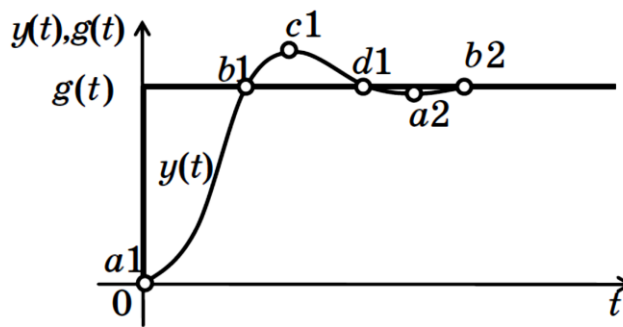


Рисунок 10.3 – Бажаний перехідний процес

На початку перехідного процесу (точка a_1) потрібно велике значення сигналу керування для отримання малого часу наростання. Тому ПІД-регулятор повинен мати велике значення K_p та K_i , а значення K_d має бути малим, тобто K'_p повинен мати значення *Big*, а K'_d – *Small*. Для забезпечення великого значення K_i (порівняно з вихідним значенням, отриманим за методом Зіглера – Ніколса) треба, щоб значення α було мало.

У методі Зіглера – Ніколса T_i задається вчетверо більше T_d ($\alpha = 4$). Тому якщо α набуває значення менше 4, то вплив інтеграла посилюється, таким чином, у точці a_1 справедливо правило виду:

Якщо ($e(k) = PB$) та ($\Delta e(k) = ZE$), то ($K'_p = Big$) та ($K'_d = Small$) та $\alpha = 2$.

Величина α може розглядатися як синглетон (рис. 10.4, де використані позначення *S-small*, *MS-medium small*, *M-medium*, *B-big*).

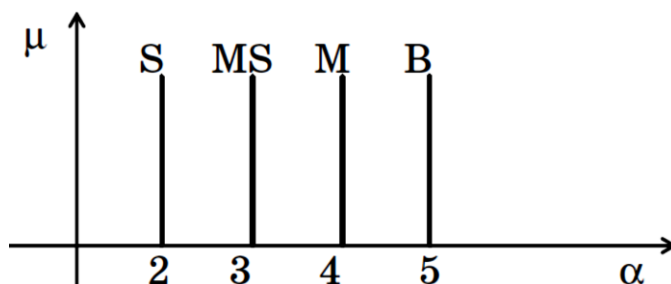


Рисунок 10.4 – Лінгвістичний опис варіантів значень α

В околиці точки b_1 (див. рис. 10.3) сигнал керування повинен бути малим для уникнення великого перерегулювання. Тому ПІД-регулятор повинен мати малі K_p та K_i і великий K_d . Ці умови описує правило:

Якщо ($e(k) = ZE$) та ($\Delta e(k) = NB$), то ($K'_p = Small$) та ($K'_d = Big$) та $\alpha = 5$.

Повторюючи наведені міркування, можна описати правила зміни K_p та K_d у вигляді табл. 10.1 та 10.2.

Таблиця 10.1 – Правила визначення K'_p

Таблиця правил		$de(t)/dt$						
		NB	NM	NS	ZE	PS	PM	PB
$e(t)$	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	ZE	S	S	S	S	S	S	S
	PS	S	S	S	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

K'_p

Таблиця 10.2 – Правила визначення K'_d

Таблиця правил		$de(t)/dt$						
		NB	NM	NS	ZE	PS	PM	PB
$e(t)$	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZE	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

K'_d

Закон зміни коефіцієнта α показаний у табл. 10.3.

Таблиця 10.3 – Правила визначення α

Таблиця правил		$de(t)/dt$						
		<i>NB</i>	<i>NM</i>	<i>NS</i>	<i>ZE</i>	<i>PS</i>	<i>PM</i>	<i>PB</i>
$e(t)$	<i>NB</i>	2	2	2	2	2	2	2
	<i>NM</i>	3	3	2	2	2	3	3
	<i>NS</i>	4	3	3	2	3	3	4
	<i>ZE</i>	5	4	3	3	3	4	5
	<i>PS</i>	4	3	3	2	3	3	4
	<i>PM</i>	3	3	2	2	2	3	3
	<i>PB</i>	2	2	2	2	2	2	2

Таблиці 10.1–10.3 можуть бути зведені в одну таблицю з 49 правил, що мають дві посилки та три висновки.

10.1 Приклад синтезу нечіткого супервізора з корекцією коефіцієнтів ПІД-регулятора на підставі інформації про помилку та її похідну

Приклад. На рис. 10.5 показана схема моделювання супервізора Simulink MatLab для об'єкта, заданого передатною функцією:

$$W(p) = \frac{27}{(p + 1)(p + 3)^3}$$

Fuzzy Logic Controller надаємо Fis name: PIDsupv3 (ім'я може бути будь-яке).

Викликається редактор системи нечіткого висновку (Fuzzy Inference System – FIS) у MatLab командою
 >> fuzzy

Дале проводиться налаштування головного вікна редактора нечіткої логічної системи (див. рис. 10.6).

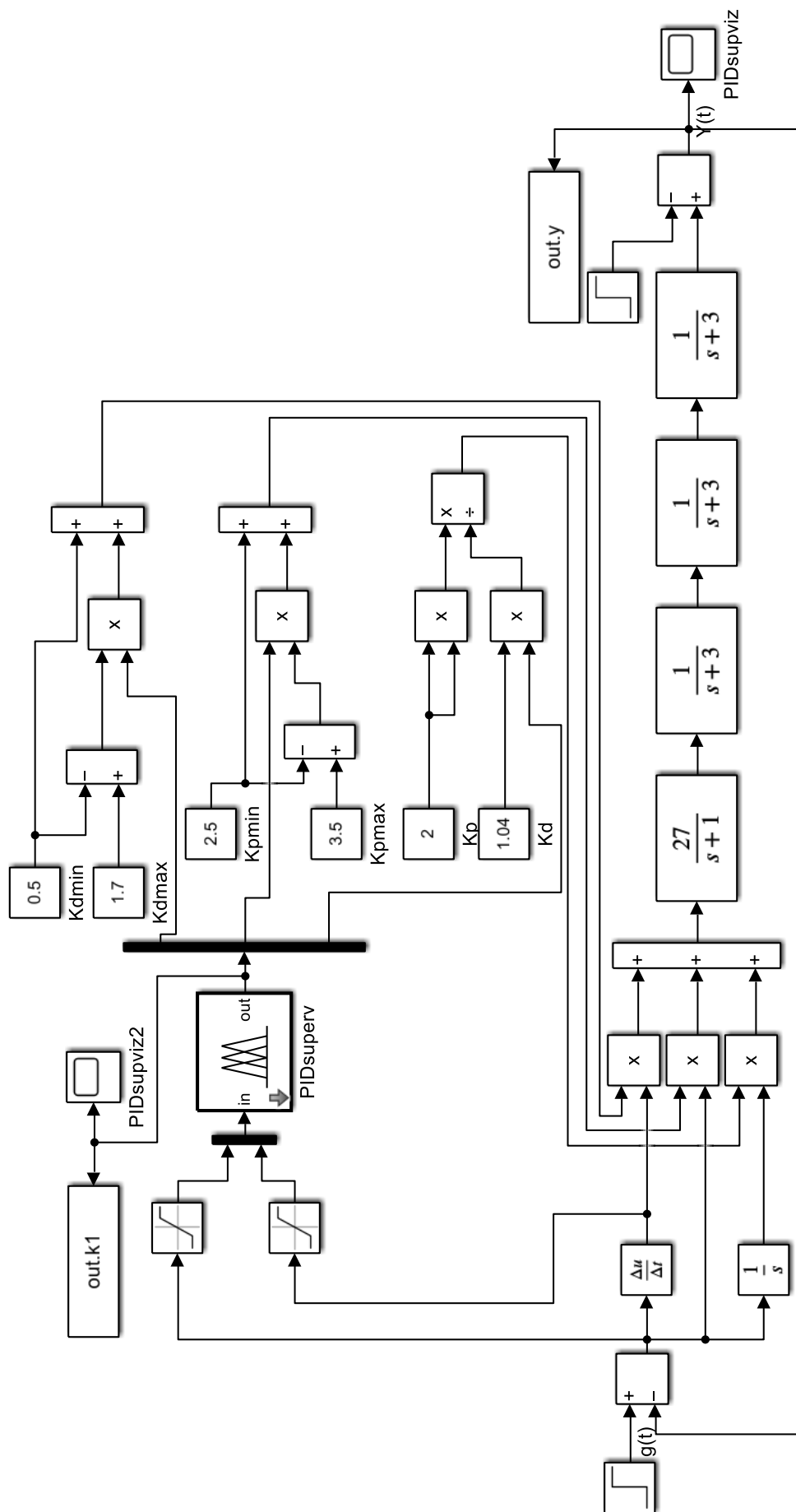


Рисунок 10.5 – Структурна схема математичної моделі системи управління

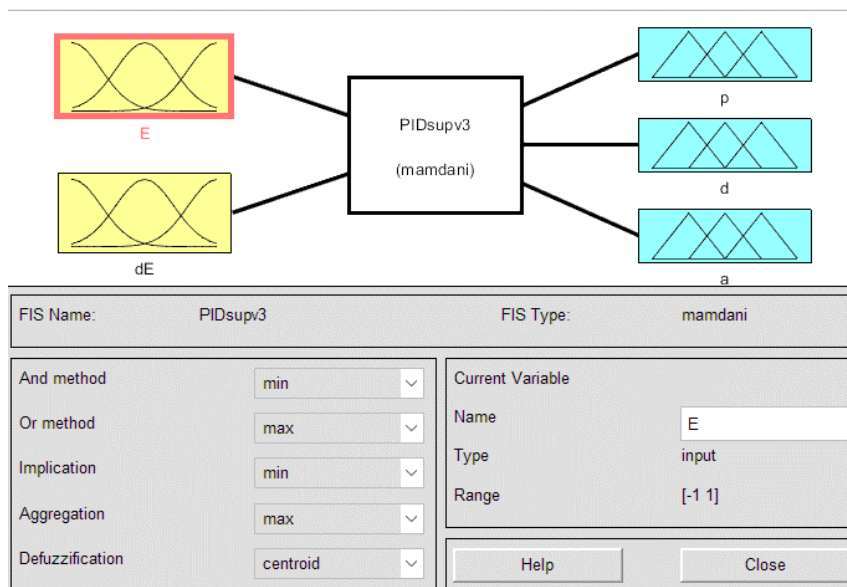


Рисунок 10.6 – Налаштування інтерфейсу FIS editor

У *FIS editor* задається опис систему нечіткого логічного висновку *Mamdani*. Для створюваної системи обирається вид логічного зв'язку (*And method* – *min*) та (*Or method* – *max*), вид імплікації (*Implication* – *min*), спосіб агрегування висновків правил (*Aggregation* – *max*) та метод дефазифікації (*Defuzzification* – *centroid*).

У меню *Edit* послідовно додаємо 2 вхідні змінні з розміром базової шкали (*Range*= [-1 1]) та 3 вихідні змінні з розміром базової шкали (*Range*= [0 1]).

Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної змінної трикутну функцію приналежності.

Нехай терми лінгвістичної змінної «Помилка управління» розміщуються відповідно до рис. 10.7 з 7 параметрами функцій приналежності:

Name = "NBe"; Type = "trimf"; Param = [-1.33 -1 -0.66];
Name = "NMe"; Type = "trimf"; Param = [-1 -0.66 -0.33];
Name = "NSe"; Type = "trimf"; Param = [-0.66 -0.33 0];
Name = "Ze"; Type = "trimf"; Param = [-0.33 0 0.33];
Name = "PSe"; Type = "trimf"; Param = [0 0.33 0.66].
Name = "PMe"; Type = "trimf"; Param = [0.33 0.66 1].
Name = "PBe"; Type = "trimf"; Param = [0.66 1 1.33].

Терми вхідної лінгвістичної змінної «Похідна помилки управління» формуємо рівномірно. Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності. 3 параметрами:

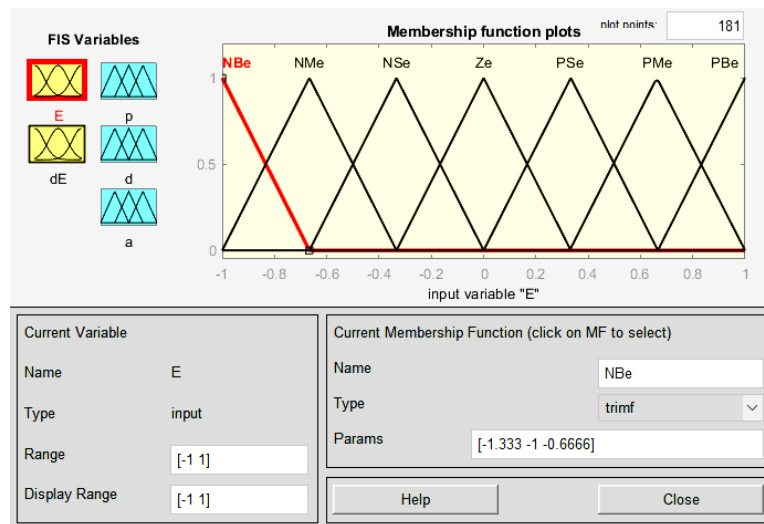


Рисунок 10.7 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Name = "NBde"; Type = "trimf"; Param = [-1.33 -1 -0.66];
 Name = "NMde"; Type = "trimf"; Param = [-1 -0.66 -0.33];
 Name = "NSde"; Type = "trimf"; Param = [-0.66 -0.33 0];
 Name = "Zde"; Type = "trimf"; Param = [-0.33 0 0.33];
 Name = "PSde"; Type = "trimf"; Param = [0 0.33 0.66].
 Name = "PMde"; Type = "trimf"; Param = [0.33 0.66 1].
 Name = "PBde"; Type = "trimf"; Param = [0.66 1 1.33].

Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління» на рис. 10.8.

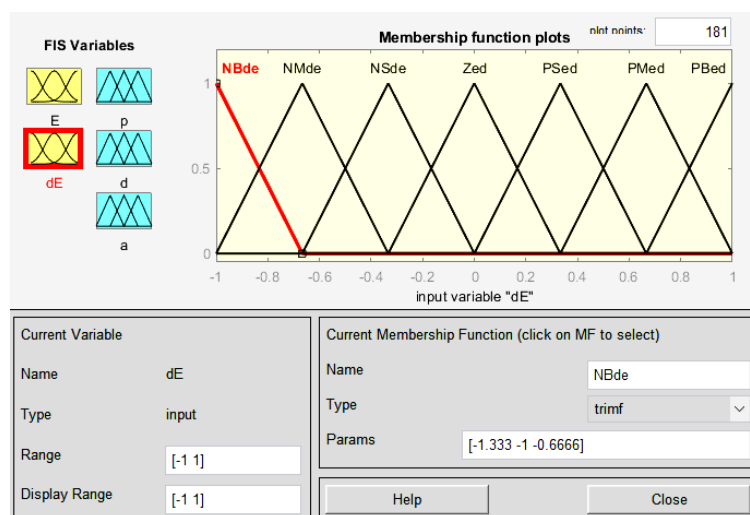


Рисунок 10.8 – Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління»

Для опису вихідної логічної змінної K_p у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної дзвоноподібну функцію приналежності.

Name = "Sp"; Type = "gbellmf"; Param = [0.5 2.5 0];
 Name = "Bp"; Type = "gbellmf"; Param = [0.5 2.5 1].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 10.9.

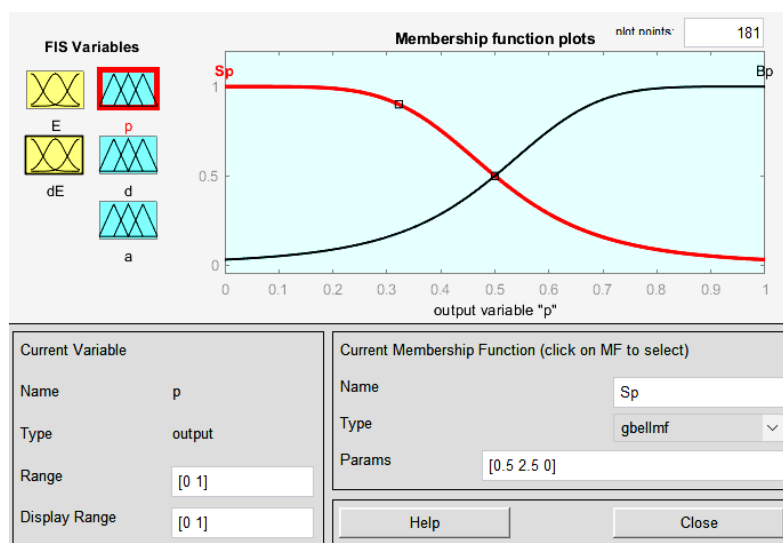


Рисунок 10.9 – Результат налаштування функцій приналежності вихідної змінної K'_p

Для опису вихідної логічної змінної K'_d у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної дзвоноподібну функцію приналежності.

Name = "Sd"; Type = "trimf "; Param = [0.5 2.5 0];
 Name = "Bd"; Type = "trimf "; Param = [0.5 2.5 1].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 10.10.

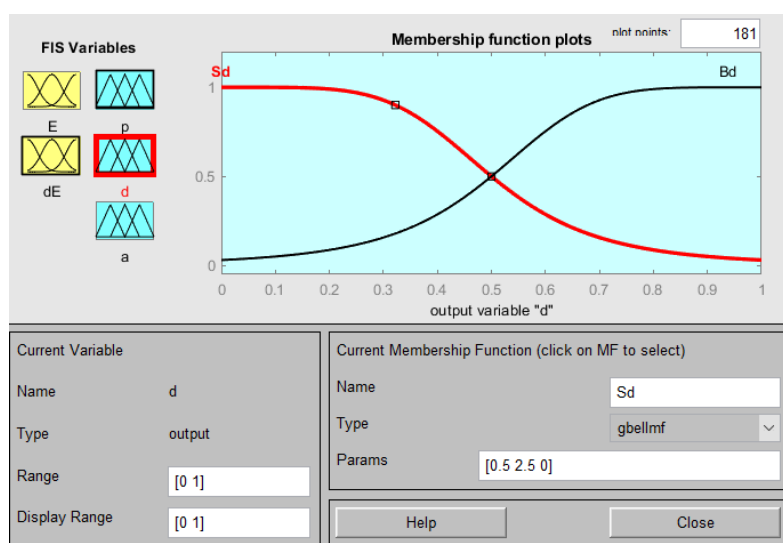


Рисунок 10.10 – Результат налаштування функцій приналежності вихідної змінної K'_d

Для опису вихідної логічної змінної α у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної вихідної змінної трикутну функцію приналежності.

Name = "Sa"; Type = "gbellmf"; Param = [0.5 2.5 0];

Name = "Ba"; Type = "gbellmf"; Param = [0.5 2.5 1].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 10.11.

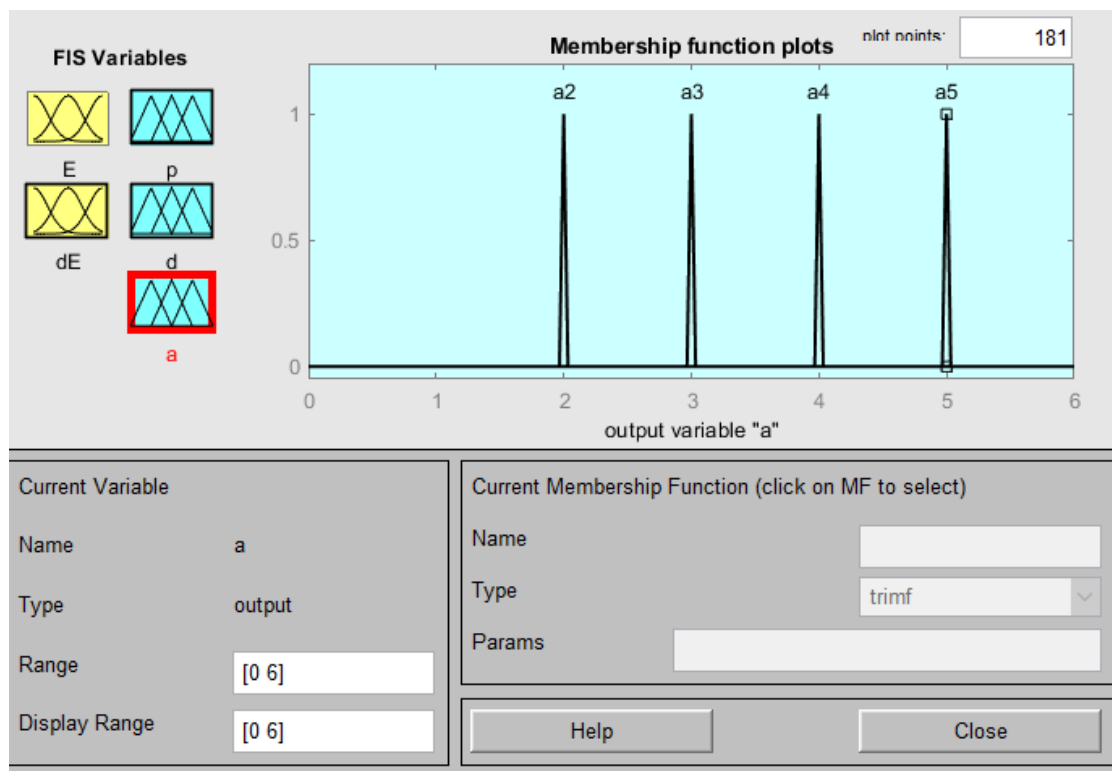


Рисунок 10.11 – Результат налаштування функцій приналежності вихідної змінної α

Після опису вхідні та вихідні лінгвістичних змінних, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку 25. Управляючі правила сформульовано згідно з рис. 11.166:

1. if (E is NBe) and (dE is NBdE) then (p is Bp) (d is Sd) (a is a2) (1)
2. if (E is NBe) and (dE is NMdE) then (p is Bp) (d is Sd) (a is a2) (1)
3. if (E is NBe) and (dE is NSdE) then (p is Bp) (d is Sd) (a is a2) (1)
4. if (E is NBe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
5. if (E is NBe) and (dE is PSdE) then (p is Bp) (d is Sd) (a is a2) (1)
6. if (E is NBe) and (dE is PMdE) then (p is Bp) (d is Sd) (a is a2) (1)
7. if (E is NBe) and (dE is PBdE) then (p is Bp) (d is Sd) (a is a2) (1)
8. if (E is NMe) and (dE is NBdE) then (p is Sp) (d is Bd) (a is a3) (1)
9. if (E is NMe) and (dE is NMdE) then (p is Bp) (d is Bd) (a is a3) (1)
10. if (E is NMe) and (dE is NSdE) then (p is Bp) (d is Sd) (a is a2) (1)
11. if (E is NMe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
12. if (E is NMe) and (dE is PSdE) then (p is Bp) (d is Sd) (a is a2) (1)

- 
13. if (E is NMe) and (dE is PMdE) then (p is Bp) (d is Bd) (a is a3) (1)
 14. if (E is NMe) and (dE is PBdE) then (p is Sp) (d is Bd) (a is a3) (1)
 15. if (E is NSe) and (dE is NBdE) then (p is Sp) (d is Bd) (a is a4) (1)
 16. if (E is NSe) and (dE is NMdE) then (p is Sp) (d is Bd) (a is a3) (1)
 17. if (E is NSe) and (dE is NSdE) then (p is Bp) (d is Bd) (a is a3) (1)
 18. if (E is NSe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
 19. if (E is NSe) and (dE is PSdE) then (p is Bp) (d is Bd) (a is a3) (1)
 20. if (E is NSe) and (dE is PMdE) then (p is Sp) (d is Bd) (a is a3) (1)
 21. if (E is NSe) and (dE is PBdE) then (p is Sp) (d is Bd) (a is a4) (1)
 22. if (E is Ze) and (dE is NBdE) then (p is Sp) (d is Bd) (a is a5) (1)
 23. if (E is Ze) and (dE is NMdE) then (p is Sp) (d is Bd) (a is a4) (1)
 24. if (E is Ze) and (dE is NSdE) then (p is Sp) (d is Bd) (a is a3) (1)
 25. if (E is Ze) and (dE is ZdE) then (p is Bp) (d is Bd) (a is a3) (1)
 26. if (E is Ze) and (dE is PSdE) then (p is Sp) (d is Bd) (a is a3) (1)
 27. if (E is Ze) and (dE is PMdE) then (p is Sp) (d is Bd) (a is a4) (1)
 28. if (E is Ze) and (dE is PBdE) then (p is Sp) (d is Bd) (a is a5) (1)
 29. if (E is PSe) and (dE is NBdE) then (p is Sp) (d is Bd) (a is a4) (1)
 30. if (E is PSe) and (dE is NMdE) then (p is Sp) (d is Bd) (a is a3) (1)
 31. if (E is PSe) and (dE is NSdE) then (p is Bp) (d is Bd) (a is a3) (1)
 32. if (E is PSe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
 33. if (E is PSe) and (dE is PSdE) then (p is Bp) (d is Bd) (a is a3) (1)
 34. if (E is PSe) and (dE is PMdE) then (p is Sp) (d is Bd) (a is a3) (1)
 35. if (E is PSe) and (dE is PBdE) then (p is Sp) (d is Bd) (a is a4) (1)
 36. if (E is PMe) and (dE is NBdE) then (p is Sp) (d is Bd) (a is a3) (1)
 37. if (E is PMe) and (dE is NMdE) then (p is Bp) (d is Bd) (a is a3) (1)
 38. if (E is PMe) and (dE is NSdE) then (p is Bp) (d is Sd) (a is a2) (1)
 39. if (E is PMe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
 40. if (E is PMe) and (dE is PSdE) then (p is Bp) (d is Sd) (a is a2) (1)
 41. if (E is PMe) and (dE is PMdE) then (p is Bp) (d is Bd) (a is a3) (1)
 42. if (E is PMe) and (dE is PBdE) then (p is Sp) (d is Bd) (a is a3) (1)
 43. if (E is PBe) and (dE is NBdE) then (p is Bp) (d is Sd) (a is a2) (1)
 44. if (E is PBe) and (dE is NMdE) then (p is Bp) (d is Sd) (a is a2) (1)
 45. if (E is PBe) and (dE is NSdE) then (p is Bp) (d is Sd) (a is a2) (1)
 46. if (E is PBe) and (dE is ZdE) then (p is Bp) (d is Sd) (a is a2) (1)
 47. if (E is PBe) and (dE is PSdE) then (p is Bp) (d is Sd) (a is a2) (1)
 48. if (E is PBe) and (dE is PMdE) then (p is Bp) (d is Sd) (a is a2) (1)
 49. if (E is NBe) and (dE is PBdE) then (p is Bp) (d is Sd) (a is a2) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 10.12).

Посилки у правилах пов'язані (connection) за допомогою операції *and*. Введено правило забезпечене ваговим коефіцієнтом (*Weight=1*).

Після опису правил можливо за допомогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 10.13).

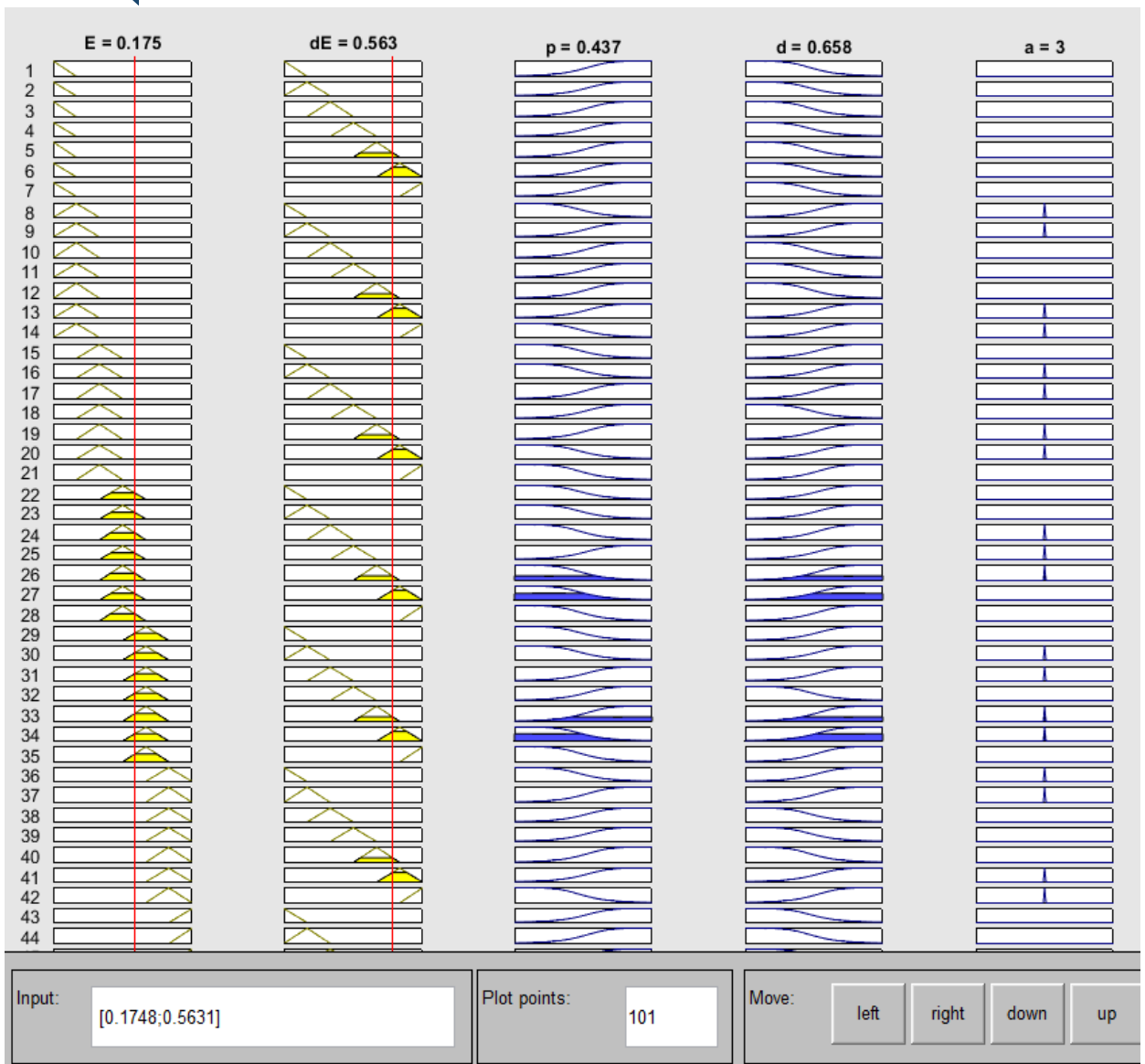


Рисунок 10.13 - Робота системи нечіткого супервізора НЛР_ПІД

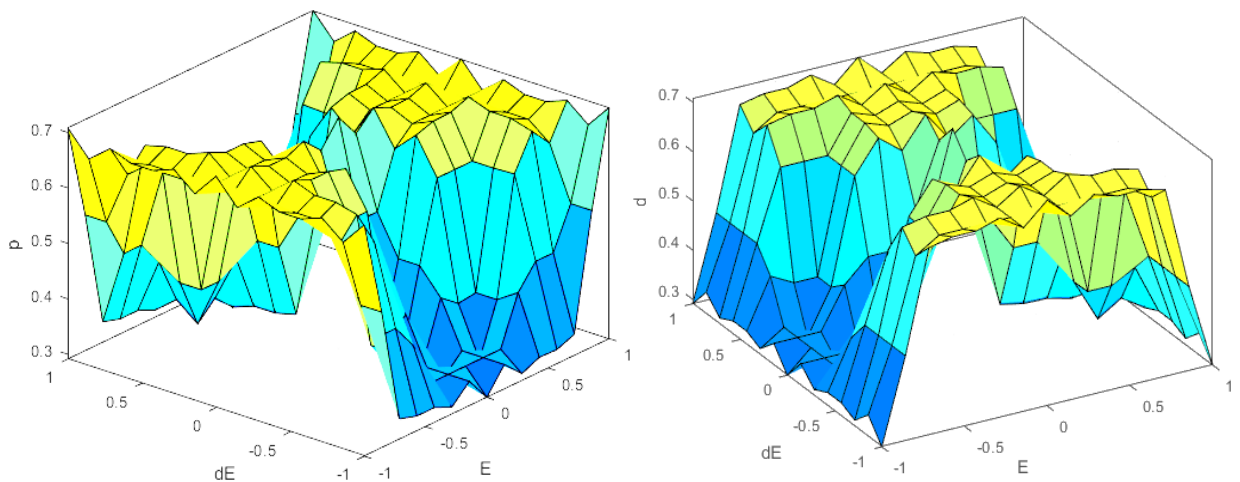


Рисунок 10.14 - Графічне подання закону управління K'_p, K'_d

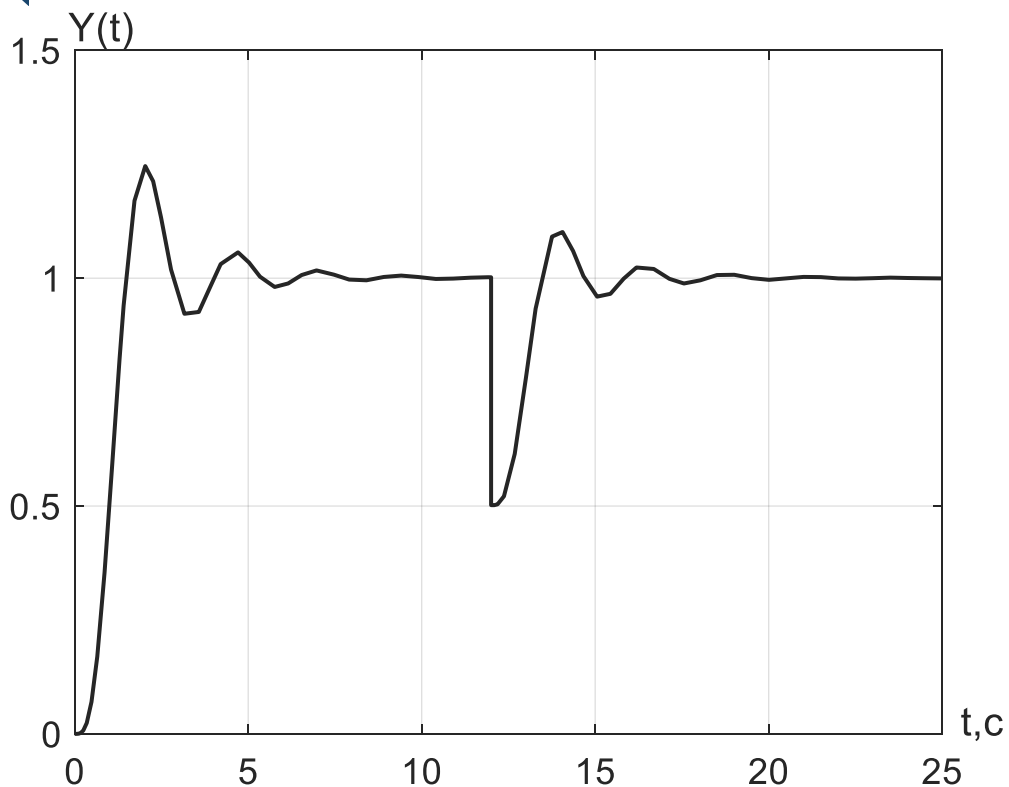


Рисунок 10.15 – Графік перехідного процесу у системі управління з нечітким супервізором ПІД-типу

10.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 10.4.

За вказаними у табл. 10.4 потрібно синтезувати нечіткий супервізор регулятора ПІД-типу.

Таблиця 10.4 - Варіанти індивідуальних завдань

№ вар	Передаточна функція об'єкту управління
1	$W(p) = \frac{1}{(3p + 1)(p + 1)^3}$
2	$W(p) = \frac{2}{(p + 1)(2p + 1)^3}$
3	$W(p) = \frac{3}{(3p + 1)(3p + 1)^3}$
4	$W(p) = \frac{4}{(2p + 1)(3p + 1)^3}$
5	$W(p) = \frac{5}{(p + 1)(3p + 1)^3}$
6	$W(p) = \frac{6}{(2p + 1)(2p + 1)^3}$

Продовження табл. 10.4

№ вар	Передаюча функція об'єкту управління
7	$W(p) = \frac{7}{(4p + 1)(3p + 1)^3}$
8	$W(p) = \frac{8}{(3p + 1)(p + 1)^3}$
9	$W(p) = \frac{9}{(p + 1)(2p + 1)^3}$
10	$W(p) = \frac{10}{(3p + 1)(3p + 1)^3}$
11	$W(p) = \frac{11}{(2p + 1)(3p + 1)^3}$
12	$W(p) = \frac{12}{(p + 1)(3p + 1)^3}$
13	$W(p) = \frac{13}{(2p + 1)(2p + 1)^3}$
14	$W(p) = \frac{14}{(4p + 1)(3p + 1)^3}$
15	$W(p) = \frac{15}{(3p + 1)(p + 1)^3}$
16	$W(p) = \frac{16}{(p + 1)(2p + 1)^3}$
17	$W(p) = \frac{17}{(3p + 1)(3p + 1)^3}$
18	$W(p) = \frac{18}{(2p + 1)(3p + 1)^3}$
19	$W(p) = \frac{19}{(p + 1)(3p + 1)^3}$
20	$W(p) = \frac{20}{(2p + 1)(2p + 1)^3}$

10.4 Контрольні питання

1. Що таке нечіткий супервізор?
2. На підставі яких міркувань відбувається синтез правил нечіткого супервізора?
3. Опишіть варіанти схем нечітких супервізорів.
4. Опишіть принципи опису дискретного НЛР ПД.
5. Опишіть принципи опису дискретного НЛР ПІ.



ПЕРЕЛІК ЛІТЕРАТУРИ

1. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators // *Neural Network*. 1989. Vol. 2. P. 359–366.256
2. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергеев. Харьков: Основа, 1997.
3. Hagan M. T., Demuth H. B. Neural networks for control // *Proc. 1999 American Control Conference*. San Diego: CA, 1999. P. 1642–1656.
4. Neural systems for control / O. Omidvar, D. L. Elliott eds. // New York: Academic Press. 1997. P. 272.
5. Галушкин А. И. Основы нейроуправления // Приложение к журналу «Информационные технологии». 2002. № 10. С. 2–16.
6. Чернодуб А. Н., Дзюба Д. А. Обзор методов нейроуправления // *Проблемы программирования*. 2011. № 2. С.79–94.
7. Soloway D., Haley P. J. Neural generalized predictive control // *Proc. 1996 IEEE International Symposium on Intelligent Control*. 1996. P. 277–281.
8. Chen S., Billings S. A. Representation of nonlinear systems: The NARMA model // *Int. J. Control*. 1989. Vol. 49(3). P. 1013–1032.
9. Narendra K. S., Mukhopadhyay S. Adaptive control using neural networks and approximate models // *IEEE Trans. Neural Networks*. 1997. Vol. 8. P. 475–485.
10. Broomhead D. S., Lowe D. Multivariable functional interpolation and adaptive networks // *Complex Systems*. 1988. N 2. P. 321–355.
11. Yager R., Filev D. *Essentials of fuzzy modeling and control*. New York: John Wiley & Sons. 1984.
12. Elman J. L. Finding structure in time // *Cognitive Sci. Ser.* 1990. N 14. P. 179–211.
13. Glover F. Future paths for integer programming and links to artificial intelligence // *Comput. Oper. Res.* 1986. Vol. 13(5). P. 533–549.

ДОДАТОК А

Налаштування коефіцієнтів передаточної функції ПІД-регулятора

Метод Зіглера – Ніколса може бути формульовано у двох варіантах – для замкнутої та розімкнутої системи. Розглядаються П, ПІ та ПІД-регулятори.

Перепишемо закон управління ПІД-регулятора у вигляді передаточної функції

$$H(p) = K_p \left(1 + T_d p + \frac{1}{T_i p} \right).$$

Розглянемо перший варіант (замкнута система):

1) коефіцієнти k_d і k_i встановлюються рівними нулю, а коефіцієнт k_p збільшується до тих, доки в системі не виникнуть автоколивання (рис. А.1а).

2) позначимо граничне значення k_p як P , а період автоколивань як T .

3) значення коефіцієнтів регулятора розраховуються відповідно до табл. А.1.

Таблиця А.1 – Розрахунок коефіцієнтів регулятора (варіант 1)

	k_p	T_i	T_d
П-регулятор	$0,5P$		
ПІ-регулятор	$0,45P$	$T/1,2$	
ПІД-регулятор	$0,6P$	$T/2$	$T/8$

Нехай об'єкт управління описується передаточною функцією

$$W(p) = \frac{1}{p^3 + 15p^2 + 2p + 1}.$$

Потрібно розрахувати параметри ПІД-регулятора.

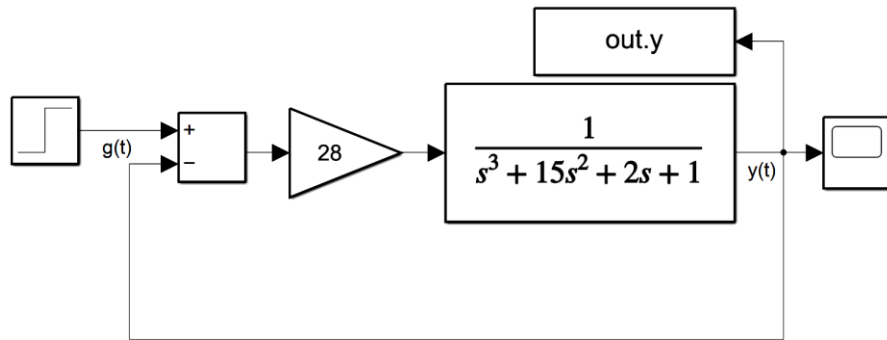
Схема моделювання у Simulink MatLab показана на рис. А.1.

Автоколивання у системі виникають при $k_p = 28$ (див. рис. А.2).

Таким чином, $P = 28$, $T \approx 5$. Відповідно до табл. А.1 отримуємо:

$$K_p \approx 17; T_d \approx 0.625; T_i \approx 2.5.$$

a)



б)

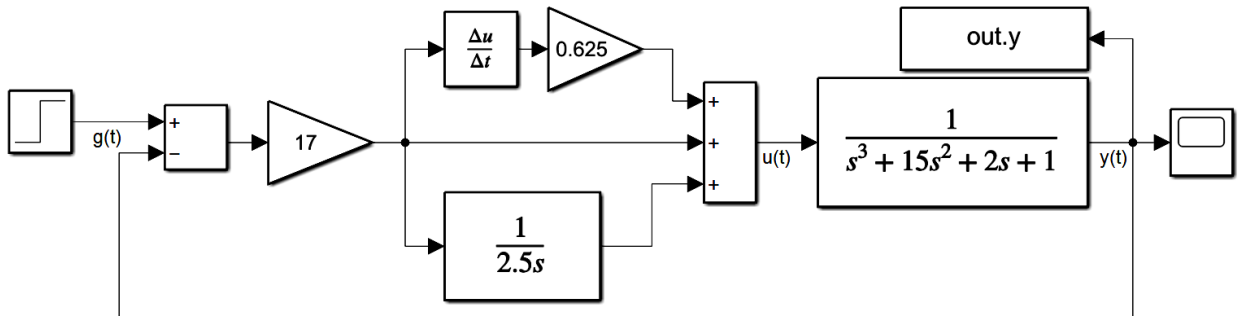


Рисунок А.1 – Математична модель роботи а) П-регулятора та б) ПІД-регулятора

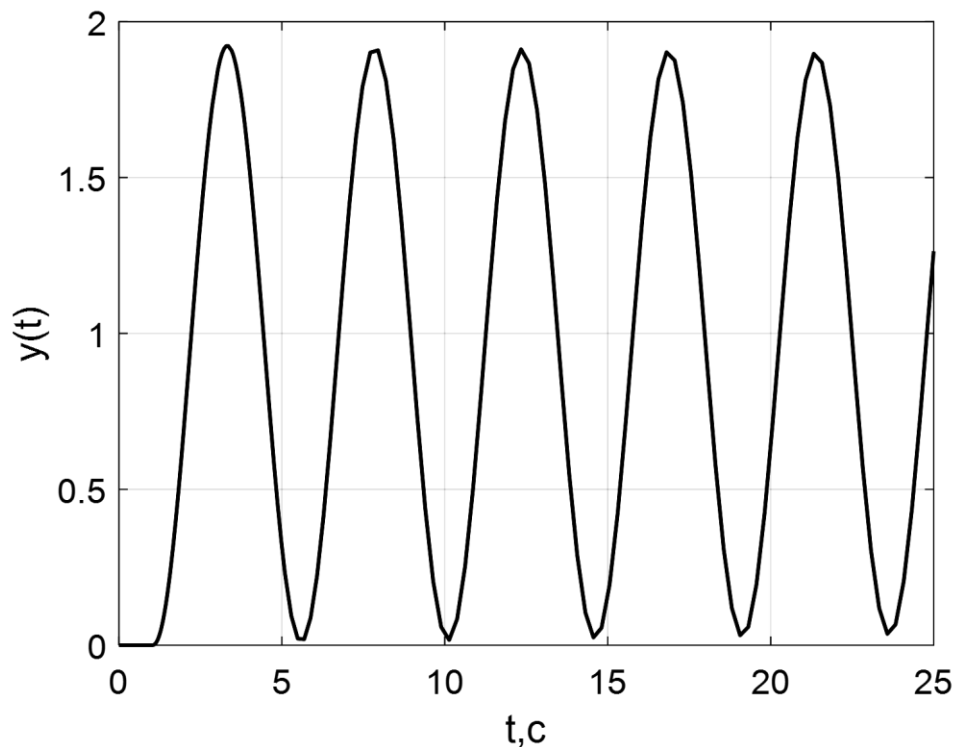


Рисунок А.2 - Незагасні коливання в системі управління

Перехідний процес для ПІД-регулятора (див. рис. А.1б) за розрахованими коефіцієнтами регулятора показано на рис. А.3.

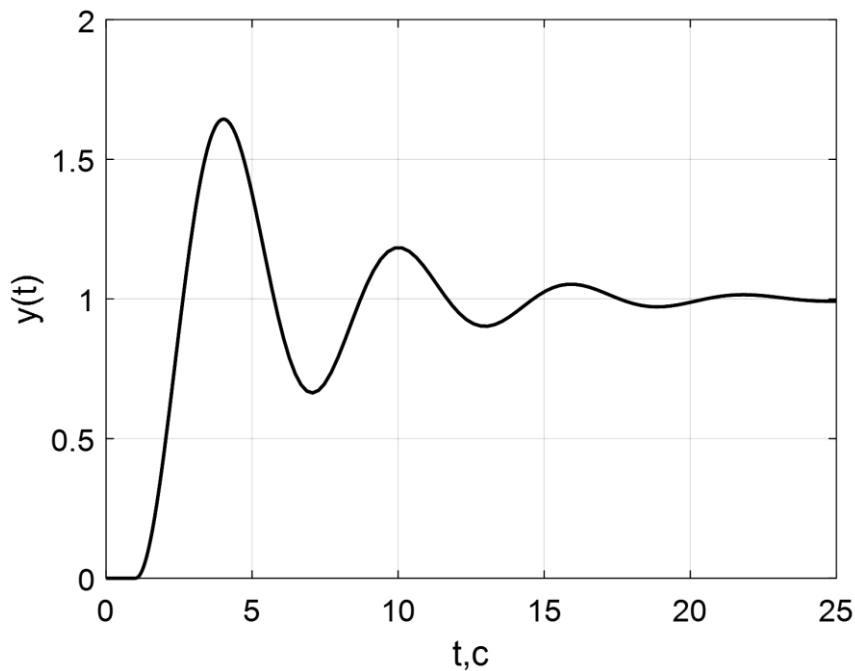


Рисунок А.3 – Перехідний процес у системі з ПІД-регулятором

Розглянемо другий варіант (розімкнута система).

На вхід системи подається одиничний стрибок. Незалежно від того, чи є система стійкою чи нестійкою, будується дотична до точки перегину кривої перехідного процесу, і визначаються два параметри: R та L (рис. А.);). Далі коефіцієнти обраного типу регулятора розраховуються відповідно до табл. А.2.

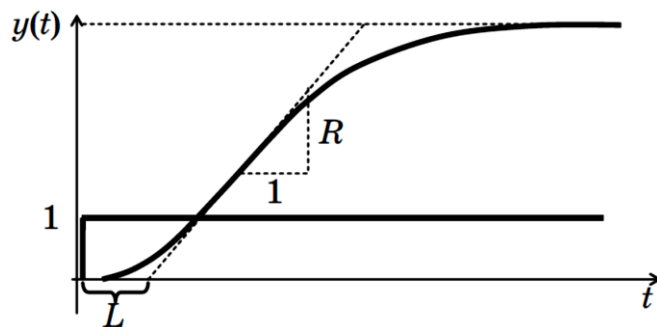


Рисунок А.4 – Визначення параметрів для стійкої системи

Таблиця А.2 – Розрахунок коефіцієнтів регуляторів (варіант 2)

	k_p	T_i	T_d
П	$1/(RL)$		
ПІ	$0,9/(RL)$	$3L$	
ПІД	$1,2/(RL)$	$3L$	$0,5L$

Нехай об'єкт управління описується передаточною функцією

$$W(p) = \frac{1}{15p^2 + 8p + 1}$$

Необхідно розрахувати параметри ПІД-регулятора. Математична модель об'єкта наведена на рисунку А.5.

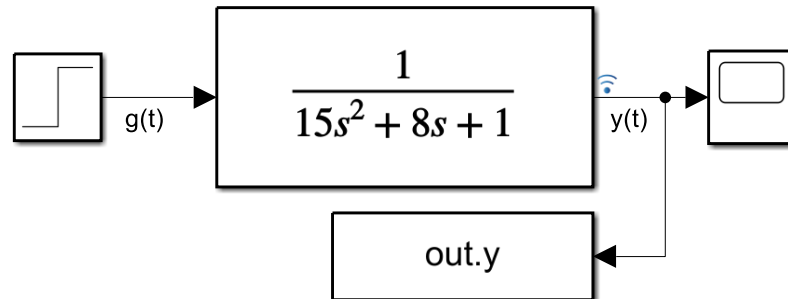


Рисунок А.5 – Математична модель об'єкта

Крива перехідного процесу для розімкненої системи показано на рис. А.6.

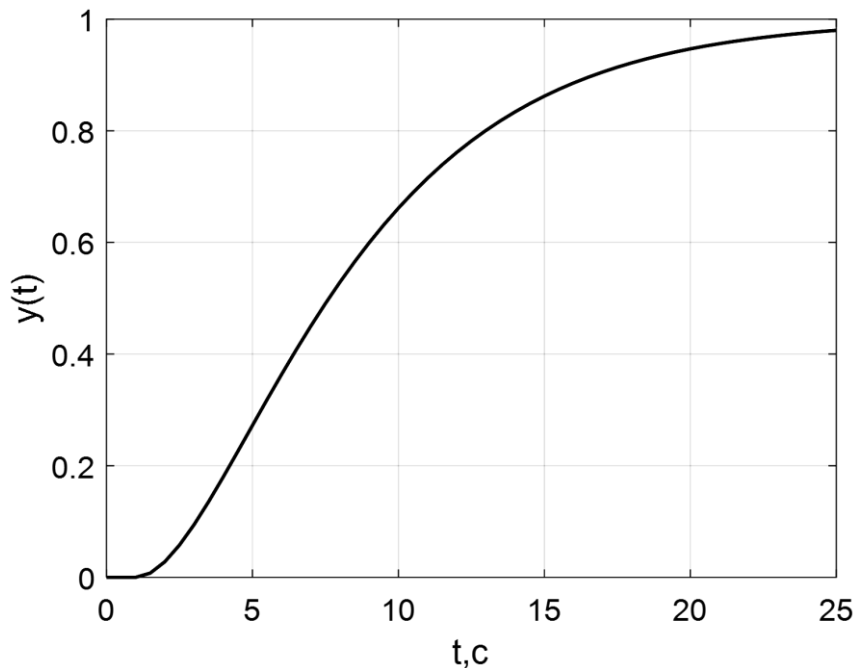


Рисунок А.6 – Графік перехідного процесу для системи без регулятора

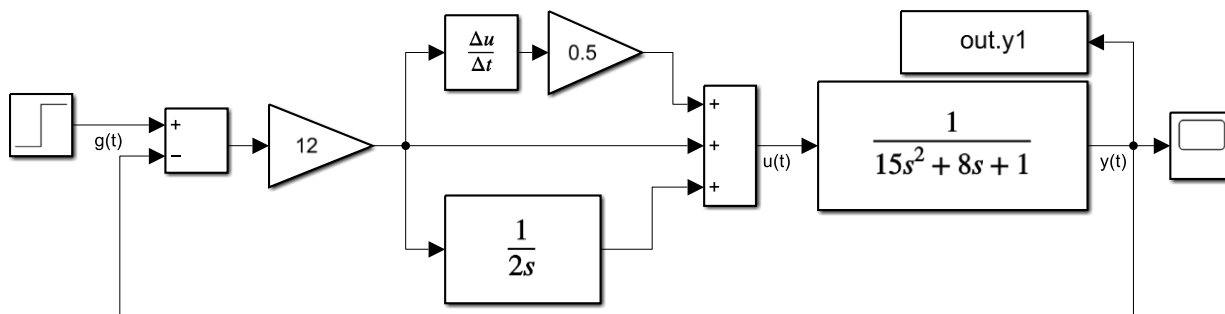
За кривою перехідного процесу, показаної на рис. А.6, визначаємо

$$L \approx 1; R \approx 0.1.$$

Потім за формулами табл. А.2 маємо

$$K_P \approx 12; T_d \approx 0.5; T_i \approx 2.$$

Математична модель роботи замкненої системи управління з ПІД-регулятором наведена на рис.А.7.



Перехідний процес для замкнутої системи показано на рис. А.7.

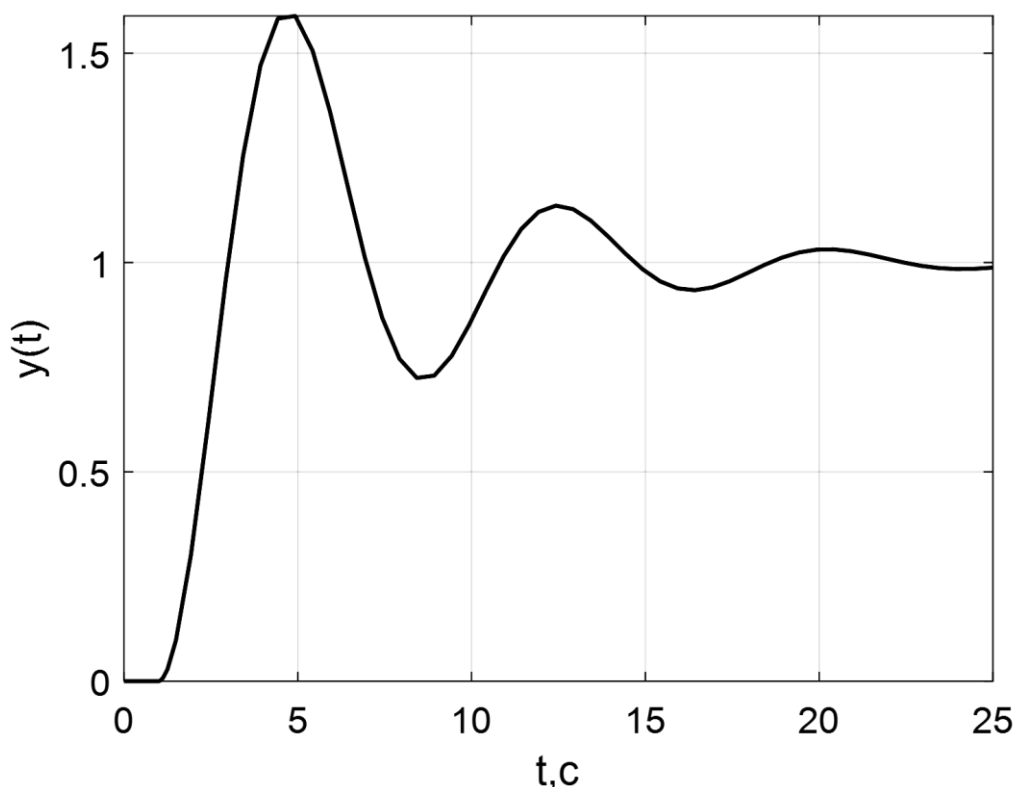



Рисунок А.7 – Графік перехідного процесу в замкненій системі з ПІД-регулятором (варіант 2)

Як свідчать рис. А.3 та А.7, для ПІД-регуляторів, розрахованих за методикою Зіглера – Ніколса, характерне велике перерегулювання (порядку 60%).

Нечіткі регулятори можуть розглядатися як природний розвиток ідей ПІД-управління, при правильному проектуванні вони забезпечують



кращі експлуатаційні показники, тобто можуть зменшити перерегулювання, час перехідного процесу і помилку, що встановилася. Для НЛР, як буде показано нижче, можна запропонувати досить прості експертні алгоритми налаштування.

ДОДАТОК Б
Демонстраційні приклади ППП NNT

Ім'я функції	Зміст прикладу
<i>Перцепторон</i>	
demop1	Перцептрон із двома входами
demop4	Формування вхідних векторів
demop5	Навчання з нормованою функцією налаштування
demop6	Приклад лінійно нероздільних векторів
<i>Лінійні мережі</i>	
demolin1	Приклад функціонування лінійної мережі
demolin2	Навчання лінійного нейрона
demolin4	Завдання лінійної апроксимації
demolin5	Завдання з неповними даними
demolin6	Завдання з лінійно залежними даними
demolin7	Оцінка впливу параметрів швидкості налаштування
demolin8	Адаптований лінійний шар
<i>Радіальні базисні мережі</i>	
demorb1	Приклад радіальної базисної мережі
<i>Рекурентна мережа Хопфілда</i>	
demohop1	Приклад двумірної мережі Хопфілда
<i>Застосування нейронних мереж</i>	
applin2	Передбачення нестационарного сигналу
appscr1	Розпізнавання символів
predcstr	Управління каталітичним реактором
nammaglev	Управління магнітною подушкою
mrefrobotarm	Управління ланкою робота

Перелік усіх демонстраційних прикладів, включених до пакета Neural Network Toolbox, можна отримати за командою `help nndemos`.

ДОДАТОК В

Функції пакета «Neural Network Toolbox» системи MATLAB для роботи з нейронними мережами

Функції створення нової мережі

- `network` – створення нейронної мережі користувача

Запис:

`net=network`

`net=network(numInputs, numLayers, biasConnect, inputConnect, layerConnect, outputConnect, targetConnect)` – функція повертає створену нейронну мережу з ім'ям `net` і з наступними характеристиками (у дужках дано значення за замовчуванням):

`numInputs` – кількість входів (0),

`numLayers` – кількість шарів (0),

`biasConnect` – булевський вектор з кількістю елементів, що дорівнює кількості шарів (нулі),

`inputConnect` – булевська матриця з числом рядків, рівним кількості шарів і числом стовпців, рівним кількості входів (нулі),

`layerConnect` – булевська матриця з числом рядків і стовпців, що дорівнює кількості шарів (нулі),

`outputConnect` – булевський вектор-рядок з числом елементів, що дорівнює кількості шарів (нулі),

`targetConnect` – вектор-рядок, такий самий, як попередній (нулі).

- `newsc` – створення конкурентного шару

`net=newsc(PR, S, KLR, CLR)` – функція створення шару Кохонена.

Аргументи функції:

`PR` – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,

`S` – число нейронів,

`KLR` – коефіцієнт навчання Кохонена (за замовчуванням 0.01),

`CLR` – коефіцієнт «справедливості» (за умовчанням 0.001).

- `newscf` – створення каскадної спрямованої мережі

`net=newscf(PR, [S1 S2...SNI], {TF1 TF2...TFNI}, BTF, BLF, PF)` – функція створення різновиду багатошарової нейронної мережі зі зворотним поширенням помилки – так званої каскадної нейронної мережі. Така мережа містить приховані $N1$ шари, використовує вхідні функції типу `dotprod` та `netsum`, ініціалізація мережі здійснюється функцією `initnw`.

Аргументи функції:

PR – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,
 S_i – розмір i -го прихованого шару, для $N1$ шарів,
 TF_i – функція активації нейронів i -го шару, за умовчанням 'tansig',
 BTF – функція навчання мережі, за умовчанням 'traingd',
 BLF – функція налаштування ваг та зміщень, за замовчуванням 'learngdm',
 PF – функція помилки, за умовчанням 'mse'.

- `newelm` – створення мережі зворотного розповсюдження Елмана (Elman)
`net=newelm(PR, [S1 S2...SNI], {TF1 TF2...TFNI}, BTF, BLF, PF)` – функція створення мережі Елмана. Аргументи такі самі, як і в попередньої функції.

- `newff` – створення односпрямованої мережі
`net=newff(PR, [S1 S2...SNI], {TF1 TF2...TFNI}, BTF, BLF, PF)` – функція створення «класичної» багатошарової нейронної мережі з навчанням за методом зворотного розповсюдження помилки.

- `newfftd` – створення односпрямованої мережі із вхідними затримками
`net=newfftd(PR, ID, [S1 S2...SNI], {TF1 TF2...TFNI}, BTF, BLF, PF)` – те, що й попередня функція, але з наявністю затримок по входах. Додатковий аргумент `ID` – вектор вхідних затримок.

- `newgrnn` – створення узагальненої регресійної нейронної мережі
`net=newgrnn(P, T, spread)` – функція створення узагальнено-регресійної мережі.

Аргументи функції:

P – $R \times Q$ матриця Q вхідних векторів,

T – $S \times Q$ матриця Q цільових векторів,

`spread` – відхилення (за замовчуванням 1.0).

- `newhop` – створення рекурентної мережі Хопфілда
`net=newhop(T)` – функція створення мережі Хопфілд. Використовує лише один аргумент

T – $R \times Q$ матриця Q цільових векторів (значення елементів мають бути +1 або -1).

- `newlin` – створення лінійного шару
`net=newlin(PR, S, ID, LR)` – функція створення шару лінійних нейронів.

Аргументи функції:

PR – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,

S – кількість елементів у вихідному векторі,

ID – вектор вхідної затримки (за умовчанням [0]),

LR – коефіцієнт навчання (за умовчанням 0.01).

Повертається новий лінійний шар.

При записі у формі `net=newlin(PR, S, 0, P)` використовується аргумент

P – матриця вхідних векторів, повертається лінійний шар з максимально можливим коефіцієнтом навчання при заданій P.

- `newlind` – конструювання лінійного шару

`net=newlind(P, T)` – функція проектування нового лінійного прошарку.

Дана функція за матрицями вхідних та вихідних векторів методом найменших квадратів визначає ваги та зміщення лінійної нейронної мережі.

- `newlvq` – створення квантованої мережі

`net=newlvq(PR, S1, PC, LR, LF)` – функція створення мережі зустрічного розповсюдження.

Аргументи функції:

PR – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,

S1 – кількість прихованих нейронів,

PC – S2 елементів вектора, що задають частки приналежності до різних класів,

LR – коефіцієнт навчання за замовчуванням 0.01,

LF – функція навчання, за умовчанням 'learnlv2'.

- `newp` – створення перцептрону

`net=newp(PR, S, TF, LF)` – функція створення перцептрону.

Аргументи функції:

PR – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,

S – кількість нейронів,


TF – функція активації, за замовчуванням 'hardlim',

LF – функція навчання, за замовчуванням 'learnp'.

- `newpnn` – конструювання імовірнісної нейронної мережі

`net=newpnn(P, T, spread)` – функція створення ймовірнісної нейронної мережі. Аргументи – як у функції `newgrnn`.

- `newrb` – конструювання мережі з радіальним базисом



`net=newrb(P, T, goal, spread)` – функція створення мережі з радіальними базовими елементами. Аргументи `P`, `T`, `spread` – такі ж, як у функції `newgrnn`; аргумент `goal` – задана середньоквадратична помилка.

- `newrbe` – конструювання точної мережі з радіальними базовими функціями
`net=newrbe (P, T, spread)` – функція створення мережі з радіальними базовими елементами з нульовою помилкою на навчальній вибірці.

- `newsom` – створення самоорганізованої карти
`net=newsom(PR, [D1, D2,...], TFCN, DFCN, OLR, OSTEPS, TLR, TND)` – функція створення карти, що самоорганізується, з аргументами:

`PR` – $R \times 2$ матриця мінімальних та максимальних значень для R вхідних елементів,

`I` – розміри i -го шару, за замовчуванням,

`TFCN` – топологічна функція, за замовчуванням 'hextop',

`DFCN` – функція відстані, за замовчуванням 'linkdist',

`OLR` – коефіцієнт навчання фази упорядкування, за замовчуванням 0.9,

`OSTEPS` – кількість кроків фази впорядкування, за замовчуванням 1000,

`TLR` – коефіцієнт навчання фази налаштування, за замовчуванням 0.02,

`TND` – відстань для фази налаштування, за замовчуванням 1.

Функції навчання

`[net, tr]=trainbfg(net, Pd, TI, Ai, Q, TS, VV, TV)` – функція навчання, що реалізує різновид квазиньютоновського алгоритму зворотного поширення помилки (*BFGS*).

Аргументи функції:

`net` - ім'я нейронної мережі, що навчається,

`Pd` – найменування масиву затриманих входів навчальної вибірки,

`TI` – масив цільових значень виходів,

`Ai` - матриця початкових умов вхідних затримок,

`Q` – кількість навчальних пар в одному циклі навчання (розмір «пачки»),

`TS` – вектор тимчасових інтервалів,

`VV` – порожній (`[]`) масив чи масив перевірочних даних,

`TV` – порожній (`[]`) масив або масив тестових даних.

Функція повертає навчену нейронну net мережу та набір записів tr для кожного циклу навчання (tr.epoch – номер циклу, tr.perf – поточна помилка навчання, tr.vperf – поточна помилка для перевірки, tr.tperf – поточна помилка для тестової вибірки).

Процес навчання відбувається відповідно до значень наступних параметрів (у дужках наведено значення за замовчуванням):

net.trainParam.epochs (100) – задану кількість циклів навчання,
net.trainParam.show (25) – кількість циклів для показу проміжних результатів,

net.trainParam.goal (0) – цільова помилка навчання,
net.trainParam.time (∞) – максимальний час навчання за секунди,
net.trainParam.min_grad (10^{-6}) – цільове значення градієнта,
net.trainParam.max_fail (5) – максимально допустима кратність перевищення помилки перевірконої вибірки порівняно з досягнутим мінімальним значенням,

net.trainParam.searchFcn ('srchcha') – ім'я одновимірного алгоритму оптимізації, що використовується.

Структури та розміри масивів:

$P_d - N_o \times N_i \times T_s$ – клітинний масив, кожен елемент якого $P\{i, j, ts\}$ є матриця $D_{ij} \times Q$,
 $T_I - N_1 \times T_S$ – клітинний масив, кожен елемент якого $P\{i, ts\}$ є матриця $V_i \times Q$,

$A_i - N_1 \times L_D$ – клітинний масив, кожен елемент якого $A_i\{i, k\}$ є матриця $S_i \times Q$,

де

$N_i = \text{net.numInputs}$ (кількість входів мережі),

$N_1 = \text{net.numLayers}$ (кількість її шарів),

$L_D = \text{net.numLayerDelays}$ (кількість шарів затримки),

$R_i = \text{net.inputs}\{i\}.\text{size}$ (розмір i-го входу),

$S_i = \text{net.layers}\{i\}.\text{size}$ (розмір i-го шару),

$V_i = \text{net.targets}\{i\}.\text{size}$ (розмір цільового вектору),

$D_{ij} = R_i * \text{length}(\text{net.inputWeights}\{i, j\}.\text{delays})$.

Якщо масив VV – непустий, то він повинен мати структуру, що визначається наступними компонентами:

$VV.PD$ – затримані значення входів перевірконої вибірки,

$VV.TI$ – цільові значення,


$VV.A_i$ - початкові вхідні умови,

$VV.Q$ – кількість перевірконих пар в одному циклі навчання,

$VV.TS$ – часові інтервали перевірконої вибірки.

Ці параметри використовуються для завдання зупинки процесу навчання у випадках, коли помилка для перевірки не зменшується або починає зростати.

Структуру, аналогічну структурі масиву VV , має масив TV .



Розглянута функція, задана у формі `trainbfg(code)`, повертає для значень аргументу, відповідно, 'phases' та 'pdefaults', імена параметрів навчання та їх значення за промовчанням.

Для використання функції в нейронних мережах, що визначаються користувачем, необхідно:

- 1) встановити параметр `net.trainFcn='trainbfg'` (при цьому параметри алгоритму будуть задані за умовчанням);
- 2) встановити необхідні значення параметрів (`net.trainParam`).

Процес навчання зупиняється у разі виконання будь-якої з наступних умов:

- перевищено задану кількість циклів навчання (`net.trainParam.epochs`),
- перевищено завданий час навчання (`net.trainParam.time`),
- помилка навчання стала менш завданої (`net.trainParam.goal`),
- градієнт став менш завданого (`net.trainParam.min_grad`),
- зростання помилки перевіркової вибірки в порівнянні з досягнутим мінімальним перевищило задане значення (`net.trainParam.max_fail`).

`[net, tr]=trainbr(net, Pd, TI, Ai, Q, TS, VV)` – функція, що реалізує так званий Байєсовський метод навчання, сутність якого полягає в підстроюванні ваг та зміщень мережі на основі алгоритму Левенберга-Марквардта. Даний алгоритм мінімізує комбінацію квадратів помилок і ваг з вибором найкращої такої (для отримання найкращих узагальнюючих властивостей мережі). Ця процедура відома як Байєсівська регуляризація, звідки слідує назва методу.

Аргументи, параметри, величини, що повертаються і використання - такі ж, як у попередньої функції. Сказане залишається чинним всім іншим функціям цієї групи.

`[net, tr]=traincgb(net, Pd, TI, Ai, Q, TS, VV)` – функція навчання нейронної мережі, що реалізує різновид алгоритму сполучених градієнтів (так званий метод Powell-Beale).

`[net, tr]=traincgf(net, Pd, TI, Ai, Q, TS, VV)` – функція навчання нейронної мережі, що реалізує різновид алгоритму зворотного розповсюдження помилки у поєднанні з методом оптимізації Флетчера-Поуелла.

`[net, tr]=traincgp(net, Pd, TI, Ai, Q, TS, VV)` – те саме, що й у попередньому випадку, але з використанням методу Polak-Ribiere.

`[net, tr] = traingd (net, Pd, TI, Ai, Q, TS, VV)` - функція, що реалізує "класичний" алгоритм зворотного поширення помилки.

[net, tr] = traingda (net, Pd, TI, Ai, Q, TS, VV) – те саме, що й у попередньому випадку, але з адаптацією коефіцієнта швидкості навчання.

[net, tr] = traingdm (net, Pd, TI, Ai, Q, TS, VV) - функція, що реалізує модифікований алгоритм зворотного розповсюдження помилки з введеною "інерційністю" корекції ваги і зміщення.

[net, tr] = traingdx (net, Pd, TI, Ai, Q, TS, VV) - функція, що реалізує комбінований алгоритм навчання, що поєднує особливості двох вищенаведених.

[net, tr] = trainlm (net, Pd, TI, Ai, Q, TS, VV) – дана функція повертає ваги та усунення нейронної мережі, використовуючи алгоритм оптимізації Левенберга-Марквардта.

[net, tr] = trainoss (net, Pd, TI, Ai, Q, TS, VV) - функція, що реалізує різновид алгоритму зворотного розповсюдження помилки з використанням методу посічених.

[net, tr] = trainrp (net, Pd, TI, Ai, Q, TS, VV) - функція, що реалізує різновид алгоритму зворотного поширення помилки, так званий пружний алгоритм зворотного поширення (resilient backpropagation algorithm, RPROP).

[net, tr]=trainscg(net, Pd, TI, Ai, Q, TS, VV) – дана функція повертає ваги та зміщення нейронної мережі, використовуючи алгоритм масштабованих сполучених градієнтів.

[net, tr]=trainwb(net, Pd, TI, Ai, Q, TS, VV) – дана функція коригує ваги та усунення нейронної мережі відповідно до заданої функції навчання нейронів.

[net, tr]=trainb1(net, Pd, TI, Ai, Q, TS, VV) – те саме, що й попередня функція, але одночасно на вхід мережі пред'являється лише один вектор входу.

[net, Ac, EI]=adaptwb(net, Pd, TI, Ai, Q, TS) – функція адаптації ваг та зміщень нейронної мережі. Використовується разом із функціями newp та newlin. Повертає масив виходів шарів Ac та масив помилок шарів EI.

Функції використання мережі

- sim – моделювання нейронної мережі

[Y, Pf, Af]=sim(net, P, Pi, Ai) – функція, що моделює роботу мережі.

Аргументи функції:

net – ім'я мережі, P – входи мережі, Pi – масив початкові умови вхідних затримок (за замовчуванням вони нульові), Ai – масив початкових умов затримок шару нейронів (за умовчанням вони нульові). Функція повертає значення виходів Y та масиви кінцевих умов затримок.

Аргументи P_i , A_i , P_f , A_f використовуються тільки у випадках, коли мережа має затримки по входах або шарах нейронів.

Структура даних аргументів:

P – масив розміру $N_i \times TS$, кожен елемент якого $P\{i, ts\}$ є матрицею розміру $R_i \times Q$.

P_i – масив розміру $N_i \times ID$, кожен елемент якого $P_i\{i, k\}$ (i -й вхід у момент $ts = k-ID$) є матрицею розміру $R_i \times Q$ (за замовчуванням – нуль).

A_i – масив розміру $N_i \times LD$, кожен елемент якого $A_i\{i, k\}$ (вихід i -го шару в момент $ts=k-LD$) є матрицею розміру $S_i \times Q$ (за замовчуванням – нуль).

Y – масив розміру $N_O \times TS$, кожен елемент якого $Y\{i, ts\}$ є матрицею розміру $U_i \times Q$.

P_f – масив розміру $N_i \times ID$, кожен елемент якого $P_f\{i, k\}$ (i -й вхід в момент $ts=TS+k-ID$) є матрицею розміру $R_i \times Q$.

A_f – масив розміру $N_i \times LD$, кожен елемент якого $A_f\{i, k\}$ (вихід i -го шару в момент $ts=TS+k-LD$) є матрицею розміру $S_i \times Q$,

при цьому

$N_i = \text{net.numInputs}$ (кількість входів мережі),

$N_l = \text{net.numLayers}$ (кількість її шарів),

$N_o = \text{net.numOutputs}$ (кількість виходів мережі),

$ID = \text{net.numInputDelays}$ (вхідні затримки),

$LD = \text{net.numLayerDelays}$ (затримки шару),

$TS = \text{Number of time steps}$ (число часових інтервалів),

$Q = \text{Batch size}$ (розмір набору векторів, що подаються),

$R_i = \text{net.inputs}\{i\}.\text{size}$ (розмір i -го вектора входу),

$S_i = \text{net.layers}\{i\}.\text{size}$ (розмір i -го шару),

$U_i = \text{net.outputs}\{i\}.\text{size}$ (розмір вектора виходу).

- `init` – ініціалізація нейронної мережі


`net=init(net)` – функція ініціалізує нейронну мережу з ім'ям `net`, встановлюючи ваги та зміщення мережі відповідно до установок `net.initFcn` и `net.initParam`.

- `adapt` – функція адаптації мережі

`[net, Y, E, Pf, Af]=adapt(net, P, T, Pi, Ai)` – функція адаптації мережі. Встановлює адаптацію мережі відповідно до установок `net.adaptFcn` та `net.adaptParam`., де E – помилка мережі, T – цільові значення виходів (за замовчуванням – нуль); решта аргументів – як у команди `sim`.

- `train` – тренування нейронної мережі

`[net, tr]=train(net, P, T, Pi, Ai)` – функція здійснює навчання мережі відповідно до установок `net.trainFcn` та `net.trainParam`, де `tr` –



інформація про виконання процесу навчання (кількість циклів та відповідна помилка навчання).

- `disp` – функція відображення властивостей нейронної мережі
`disp(net)` – функція повертає розгорнуту інформацію про структуру та властивості мережі.
`display` – функція відображення імен змінних та властивостей мережі.