ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ УПРАВЛІННЯ

Методичні вказівки до виконання лабораторних робіт з дисципліни

«Інтелектуальні системи управління»

(за освітньо-професійною програмою другого (магістерського) рівня освіти «Інтелектуальні системи управління та робототехнічні комплекси в гірничо-металургійному виробництві» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»)

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

Запоріжжя 2024

Інтелектуальні системи управління: методичні вказівки до виконання практичних робіт з дисципліни «Інтелектуальні системи управління» (за освітньо-професійною програмою другого (магістерського) рівня освіти «Інтелектуальні системи управління та робототехнічні комплекси в гірничо-металургійному виробництві» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»). / Уклад. О.В. Разживін. Запоріжжя: ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2024. 118 с.

Практичні роботи присвячено роботі інтелектуальний систем управління в 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 за освітньо-професійною програмою другого (магістерського) рівня освіти «Інтелектуальні системи управління та робототехнічні комплекси в гірничо-металургійному виробництві» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»).

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.

ЗМІСТ

Стор.

1 КЛАСТЕРНИЙ АНАЛІЗ СЕНСОРНИХ МЕРЕЖ

1.1 Кластеризація за допомогою алгоритму нечітких центрів

Мета роботи: Освоїти методику знаходження центрів кластерів простору розташування сенсорів.

Завдання*:* Знайти центри кластерів сенсорів бездротової мережі, використовуючи алгоритм нечітких центрів за допомогою програми *Clustering* (Кластеризація), що входить до *Fuzzy Logic Toolbox* математичного середовища MATLAB.

Основні теоретичні відомості:

Кластеризація – це об'єднання об'єктів у групи (кластери) на основі схожості ознак для об'єктів однієї групи та відмінностей між групами.

Кластер (від англійського cluster) – гроно, пучок, скупчення, група елементів, що характеризуються якоюсь загальною властивістю.

Кластеризація допомагає уявити неоднорідні дані у наочнішому вигляді й у подальшого, зручнішого, використання цих даних.

Кластеризація може бути використана для вирішення наступних завдань:

- обробка зображень;
- класифікація;
- тематичний аналіз колекцій документів;
- побудова репрезентативної вибірки;
- аналізу експериментальних даних;
- при побудові сенсорних мереж та інш.

Кластерний аналіз призначений для розбиття безлічі об'єктів на задану чи невідому число класів на підставі деякого математичного критерію якості класифікації. Критерій якості кластеризації тією чи іншою мірою відображає такі неформальні вимоги:

- усередині груп об'єкти мають бути тісно пов'язані між собою;
- об'єкти різних груп мають бути далекі один від одного.

Ці вимоги виражають стандартну концепцію компактності класів розбиття. Вузловим моментом у кластерному аналізі вважається вибір міри близькості об'єктів (метрики), від якого вирішальним чином залежить остаточний варіант розбиття об'єктів групи при заданому алгоритмі розбиття. У кожній конкретній задачі цей вибір проводиться по-своєму, з урахуванням основних цілей дослідження, фізичної та статистичної природи інформації, що використовується і т.д.

У кластерному аналізі так само важлива відстань між цілими групами об'єктів. Наведемо приклади найбільш поширених відстаней та заходів близькості, що характеризують взаємне розташування окремих груп об'єктів.

Нехай: ω_i – *i*-я група (клас, кластер) об’єктів;

Ni – кількість об'єктів, що утворюють групу ω_i, кожен об'єкт є крапкою в *n*- мірному просторі;

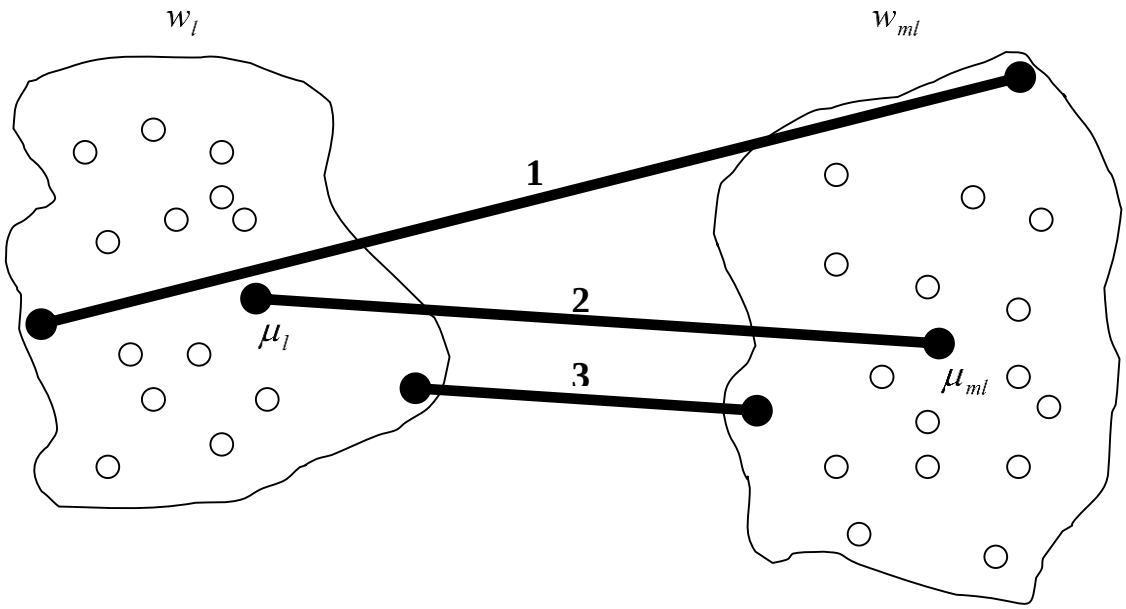
μ_i – середнє арифметичне об'єктів, що входять до ω_i (тобто. μ_i – «центр ваги» *i*-й групи);

p – число груп (кластерів);

x_i – *i*-й об’єкт кластера;

q (ω_l, ω_m) – відстань між групами ω_l та ω_m.

Різні способи визначення відстані між кластерами ω_l та ω_m (див. рис. 1.1.)



1 – по найдальших об'єктах, 2 – по центрам тяжіння, 3 – по найближчим об'єктам

Рисунок 1.1. Способи визначення відстані між кластерами ω_l та ω_m

Відстань далекого сусіда – відстань між найдальшими об'єктами кластерів (рис. 1.1 відрізок 1):

$$q_{\min}(\omega_l, \omega_m) = \max d(\mathbf{x}_i, \mathbf{x}_j), i = \overline{1, p}, j = \overline{1, p}.$$

Відстань центрів тяжіння дорівнює відстані між центральними точками кластерів (рис. 1.1 – відрізок 2):

$$q(\omega_l, \omega_m) = d(\mu_l, \mu_m).$$

Відстань найближчого сусіда є відстань між найближчими об'єктами кластерів (рис. 1.1 відрізок 3):

$$q_{\min}(\omega_l, \omega_m) = \min d(\mathbf{x}_i, \mathbf{x}_j), i = \overline{1, p}, j = \overline{1, p}.$$

Вибір тієї чи іншої міри відстані між кластерами впливає на вид геометричних угруповань об'єктів, що виділяються алгоритмами кластерного аналізу, в просторі ознак. Так, алгоритми, що ґрунтуються на відстані найближчого сусіда, добре працюють у разі угруповань, що мають складну, зокрема, ланцюжкову структуру. Відстань далекого сусіда застосовується, коли шукані угруповання утворюють у просторі ознак кулясті хмари. І проміжне місце займають алгоритми, що використовують відстані центрів тяжкості та середнього зв'язку, які найкраще працюють у разі угруповань еліпсоїдної форми.

Алгоритми кластерного аналізу відрізняються великою різноманітністю. Це можуть бути, наприклад, алгоритми, що реалізують повний перебір поєднань об'єктів або здійснюють випадкові розбиття безлічі об'єктів. У той же час більшість таких алгоритмів складається з двох етапів:

- на першому етапі задається початкове (можливо, штучне або навіть довільне) розбиття безлічі об'єктів на класи та визначається деякий математичний критерій якості автоматичної класифікації;
- • на другому етапі об'єкти переносяться з класу до класу, доки значення критерію не перестане покращуватися..

Таким чином, алгоритми кластеризації засновані на подобі образів та розміщують близькі образи в один кластер.

Виявлення центрів – це значний етап під час попередньої обробки даних, оскільки дозволяє порівняти з цими центрами функції приналежності змінних під час проектування системи нечіткого виводу.

Програма *Clustering* (Кластеризація) пакету *Fuzzy Logic Toolbox* математичного середовища MATLAB виявляє центри кластерів, тобто. точки в багатовимірному просторі даних, у яких групуються (накопичуються) експериментальні дані.

У програмі *Clustering* (Кластеризація) використовуються два алгоритми виявлення центрів кластерів: *Subtractive clustering* (що віднімає кластеризація) та *Fuzzy c-means* (алгоритм нечітких центрів).

В основі першого алгоритму лежить пропозиція, що кожна експериментальна точка може бути центром кластера, при цьому спочатку для кожної точки обчислюється міра правдоподібності даного припущення («потенціал точки»), заснована на щільності крапок у заданій околиці. Подальші обчислення відбуваються ітеративне:

1) точка з найбільшим потенціалом оголошується центром першого кластеру;

2) із зазначеної околиці цієї точки видаляються всі інші точки;

3) з точок, що залишилися, оголошується центр наступного кластера і т.д., поки не будуть розглянуті (виключені або оголошені центрами) всі точки.

Алгоритм Fuzzy c-means є більш точним, для його роботи потрібне завдання таких опцій як число кластерів і число ітерацій. Розглянемо його докладніше.

Кластеризація на основі алгоритму нечітких центрів.

На основі нечіткого c-means алгоритму виконується кластеризація даних. Цей алгоритм кластеризації запропонував Джеймс Бездек (James Bezdek) у 1981 році.

Існує безліч методів кластеризації, які можна класифікувати на чіткі та нечіткі. Чіткі методи кластеризації розбивають вихідне безліч об'єктів X на кілька підмножин, що не перетинаються. При цьому будь-який об'єкт із X належить лише одному кластеру. Нечіткі методи кластеризації дозволяють одному й тому об'єкту належати одночасно кількох (чи навіть усім) кластерам, але з різною мірою істинності. Нечітка кластеризація у багатьох ситуаціях «природніша», ніж чітка, наприклад, для об'єктів, розташованих на межі кластерів.

Завдання нечіткої кластеризації ставиться так:

Дано:

- $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, ..\mathbf{x}_n^T)^T$ – об'єкти, що підлягають кластеризації, де *n* – кількість об'єктів, T – символ транспонування. Кожен об'єкт $\mathbf{x}_k = (x_{k_1}, x_{k_2}, ..., x_{k_p})^T$, $k = \overline{1.n}$ являє собою точку в p-вимірному просторі ознак;
- *c* – кількість кластерів ($2 \le c < n$).

Необхідно кожному елементу множини поставити у відповідність до ступеня належності до класів.

Елементи одного кластера повинні бути такі близькі один до одного, як це тільки можливо, і, одночасно, кластери повинні бути на найбільшому віддаленні один від одного. Для забезпечення керованості процесу кластеризації необхідно використовувати міру близькості, якою зазвичай визначають відстань між двома об'єктами (точками в *p*-мірному просторі) $\mathbf{x}_k$ та $\mathbf{x}_l$ у вигляді речової функції $d : x \times x \to R^+$ такий що:

$$d(\mathbf{x}_k, \mathbf{x}_l) = d_{kl} \ge 0;$$
$$d_{kl} = 0 \Leftrightarrow \mathbf{x}_k = \mathbf{x}_l;$$
$$d_{kl} = d_{lk}.$$

Додатково, якщо функція d задовольняє правилу трикутника, тобто. $d_{kl} \le d_{kj} + d_{jl}$, тоді ця функція є метрикою, хоча виконання цієї властивості не завжди необхідне задач кластеризації.

Будь-яке розбиття безлічі $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, ...\mathbf{x}_n^T)^T$ на нечіткі підмножини S_i ($i = \overline{1.c}$) може бути повністю описано функцією приналежності $\eta_{S_i} : \mathbf{X} \to [0,1]$.

Позначимо через η_{ik} – ступінь належності об'єкта $\mathbf{x}_k = (x_{k_1}, x_{k_2}, ..., x_{k_p})^T$, к підмножиною S_i, тобто $\eta_{ik} \equiv \eta_{S_i}(\mathbf{x}_k)$, а через $\mathbf{V}_{cn}$ – безліч усіх дійсних матриць розміром $c \times n$. Тоді нечітким з-розбиттям (або матрицею ступенів приналежності) називається матриця $\mathbf{M} = [\eta_{ik}] \in V_{cn}$ при виконанні наступних умов:

$$\eta_{ik} \in [0,1], i = \overline{1.c}, k = \overline{1.n}; \quad (4.1)$$

$$\sum_{i=1}^{c} \eta_{ik} = 1, k = \overline{1.n}; \quad (4.2)$$

$$\sum_{k=1}^{n} \eta_{ik} \in (0, n), i = \overline{1.c}. \quad (4.3)$$

На відміну від чіткого, при нечіткому с-розбиття будь-який об'єкт одночасно належить до різних кластерів, але з різним ступенем. Умови (4.2) та (4.3) потребують лише, щоб сума ступенів належності об'єкта до всіх кластерів була нормалізована до 1, а також щоб кількість кластерів, до яких належить об'єкт не перевищувала *c*.

Позначимо центри кластерів, тобто. точки в *p*-мірному просторі, навколо яких сконцентровані відповідні об'єкти, через $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{ip})$, $i = \overline{1.c}$

При використанні евклідової відстані завдання нечіткої кластеризації полягає у знаходженні такої матриці ступенів належності **M** та таких координат центрів кластерів $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_c)$, які забезпечують мінімум наступного критерію:

$$\sum_{i=1}^{c} \sum_{k=1}^{n} (\eta_{ik})^m \cdot \|\mathbf{x}_k - \mathbf{v}_i\|^2 \to \min,$$

де $\mathbf{v}_i = \dfrac{\sum_{k=1}^{n} (\eta_{ik})^m \cdot \mathbf{x}_k}{\sum_{k=1}^{n} \eta_{ik}}$ – центр *i*-го кластера, $i = \overline{1.c}$, m – так звана експоненційна вага ($m \geq 1$). У математиці під позначенням $\|\bullet\|$ зазвичай розуміють норму, тобто. функцію, задану на векторному просторі та узагальнюючу поняття довжини вектору.

Значення експоненційної ваги встановлюється на початок кластеризації. Експонентна вага m впливає на матрицю ступенів належності **M**. Чим більше m, тим кінцева матриця *c*- розбиття стає більш «розмазаною», а при $m \to \infty$ все об'єкти належать всім кластерам

з однаковим ступенем приналежності, що дуже поганим рішенням. Експонентна вага дозволяє також при формуванні координат центрів кластерів посилити вплив об'єктів з великими значеннями ступенів належності та зменшити вплив об'єктів з малими значеннями ступенів належності. На даний момент не існує теоретично обґрунтованого правила вибору значення *m*. Зазвичай встановлюють *m=2.*

Аналітичного вирішення задачі знаходження оптимальних координат центрів кластерів та матриці ступенів належності не існує, тому вона вирішується чисельне. У середовищі MATLAB алгоритм нечітких центрів реалізовано у функції *fcm*.

Функція *fcm* може мати три вхідні аргументи:

1) **X** – матриця, що представляє дані, що підлягають кластеризації. Кожен рядок матриці відповідає одному об'єкту (образу);
2) *c* – кількість кластерів, яка має бути отримана в результаті виконання функції *fcm*. Кількість кластерів має бути більшою за 1 і менше числа образів, заданих матрицею **X.**
3) *options* – необов'язковий аргумент, що встановлює параметри алгоритму кластеризації:

- options(1) – значення експоненційної ваги (за замовчуванням – 2.0);
- options(2) – максимальна кількість ітерацій алгоритму кластеризації (за замовчуванням значення - 100);
- options(3) – мінімально допустиме значення покращення цільової функції за одну ітерацію алгоритму (за замовчуванням значення – 0.000001);
- options(4) – виведення проміжних результатів під час роботи функції fcm (значення за замовчуванням – 1).

Для використання значень за промовчанням можна ввести NaN як значення відповідної координати вектору *options*.

Алгоритм кластеризації зупиняється, коли виконано максимальну кількість ітерацій або коли покращення цільової функції за одну ітерацію менше вказаного мінімально допустимого значення.

Функція *fcm* має три вихідні аргументи:

1) **V** – матриця координат центрів кластерів, отриманих у результаті кластеризації. Кожен рядок матриці відповідає центру одного кластера;
2) **M** – матриця ступенів належності образів до кластерів. Кожен рядок матриці відповідає функції приналежності одного кластера;
3) **obj_fcn** – вектор значень цільової функції на кожній ітерації алгоритму кластеризації.

Модуль графічного інтерфейсу Findcluster:

GUI-модуль (Модуль графічного інтерфейсу) **Findcluster** дозволяє автоматично знаходити центри кластерів багатовимірних даних за допомогою нечіткого *c-means* алгоритму та алгоритму кластеризації, що віднімає (*subtractive clustering*). Завантаження модуля **Findcluster** здійснюється за командою **findcluster.** Основне графічне вікно модуля **Findcluster** із зазначенням призначення функціональних областей наведено на рис. 1.2.

Модуль **Findcluster** із зазначенням призначення функціональних областей наведено на рисотримачем 7 верхніх типових меню графічного вікна. 2 (**File (Файл), Edit (Редагування)**, **View (Огляд)**, **Insert (Вставка)**, **Tools (Інструменти)**, **Windows (Вікна)** та **Help (Допомога)**), область візуалізації, область завантаження даних, область кластеризації, область виведення поточної інформації, а також кнопки **Info (Інформація)** та **Close (Зачинити)**, які дозволяють викликати вікно довідки та закрити модуль, відповідно.

Область візуалізації

У цій галузі у двовимірному просторі виводяться експериментальні дані (образи) та знайдені центри кластерів. Для образів використовується маркер у вигляді червоного кола (o), а центрів кластерів – маркер як чорної точки ( ).

В області також розташовані меню вибору координатних осей **X-axis** та **Y-axis**, що дозволяють асоціювати ознаки образів з осями абсцис та ординат.

Область загрузки даних

У цій області, яка розташована у верхньому правому куті вікна, знаходиться кнопка **Load Data (Завантаження даних)**. Натискання цієї кнопки дозволяє завантажити дані для кластеризації, що зберігаються на диску. Після натискання кнопки **Load Data…** відкривається типове вікно відкриття файлу. У файлі дані повинні бути записані рядково, тобто кожному образу повинен відповідати один рядок файлу даних.

Область виведення поточної інформації

У цій галузі, яка розташована внизу графічного вікна, виводиться найважливіша поточна інформація, наприклад, стан модуля, номер ітерації алгоритму кластеризації, значення цільової функції тощо..

Область кластеризації

У цій галузі користувач може вибрати алгоритм кластеризації, встановити параметри алгоритму кластеризації, провести кластеризацію та зберегти координати центрів кластерів у вигляді файлу. В області розташовані наступні меню та кнопки.

Меню **Method (Метод)** дозволяє вибрати один із двох алгоритмів кластеризації:

- **subtractiv** – алгоритм віднімання кластеризації;
- **fcm -** нечіткий c-means алгоритм.

При виборі алгоритму кластеризації, що віднімає, графічне вікно модуля **Findcluster** має вигляд, показаний на рис. 1.2.

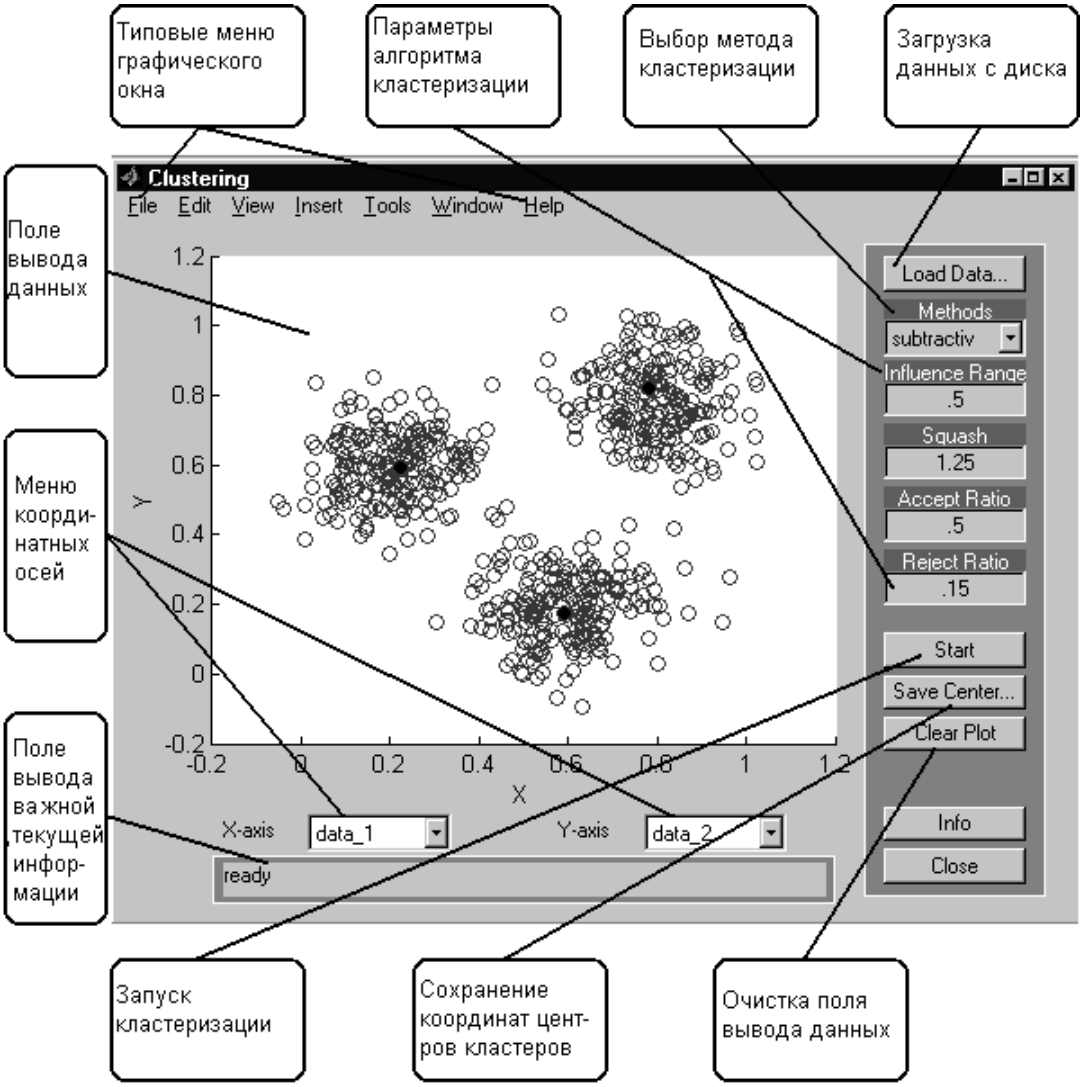


*Рисунок 1.2 - Головне вікно модуля **Findcluster***

У цьому випадку користувач має можливість встановити значення наступних параметрів алгоритму **Influence Range (Діапазон впливу)**, **Squash (Давка)**, **Accept Ratio (Коефіцієнт, що приймається) та Reject Ratio (Відхилений коефіцієнт)**, смысл которых объяснен в описании функции **subclust**.

При виборі нечіткого *c-means* алгоритму область кластеризації набуває вигляду, зображеного на рис. 1.3. У цьому випадку користувач має можливість встановити значення наступних параметрів:

- **Cluster Num. –** кількість кластерів;
- **Max Iteration # -** максимальна кількість ітерацій алгоритму;
- **Min –** мінімально допустиме значення покращення цільової функції за одну ітерацію алгоритму;
- **Exponent –** значення експоненційної ваги.

Рисунок 1.3 – Область кластеризації

Кнопка **Start –** запускає кластеризацію. У разі використання алгоритму fcm значення координат центрів кластерів виводяться у вікні візуалізації після кожної ітерації. При використанні алгоритму, що віднімає, відкривається додаткове вікно (див. рис. 1.3), що показує динаміку процес кластеризації. Координати центрів кластерів виводяться після закінчення алгоритму.

Кнопка **Clear Plot** запускає кластеризацію. У разі використання алгоритму fcm значення координат центрів кластерів виводяться у вікні візуалізації після кожної ітерації. При використанні алгоритму, що віднімає, відкривається додаткове вікно (див. рис. 1.3), що показує динаміку процес кластеризації. Координати центрів кластерів виводяться після закінчення алгоритму).

```
0.636308 0.211547 0.608300
0.595761 -0.018493 0.580148
0.591420 0.172150 0.487198
0.509358 0.193506 0.385195
0.380619 0.140240 0.577906
0.386364 0.235873 0.423574
0.611046 0.061674 0.638085
0.502620 0.091982 0.584804
0.625099 0.093233 0.324443
0.492908 0.131983 0.302384
0.717591 0.314972 0.868287
```

Рисунок 1.4 - Файл clusterdemo.dat

Приклад виконання:

1) загрузим файл даних MATLABR2021b\toolbox\fuzzy\fuzdemos\ clusterdemo.dat з допомогою кнопки «Load Data»;

2) виберемо алгоритм кластеризації *Fuzzy c-means (fcm),* з допомогою кнопки «Method»;

3) задаємо число кластерів рівне 3, за допомогою кнопки опції «Cluster num»;

4) поставимо число ітерацій рівне 100, за допомогою кнопки опції «Max Iteration#»;

5) натискаємо кнопку Start та отримуємо результат (див. рис. 1.5).

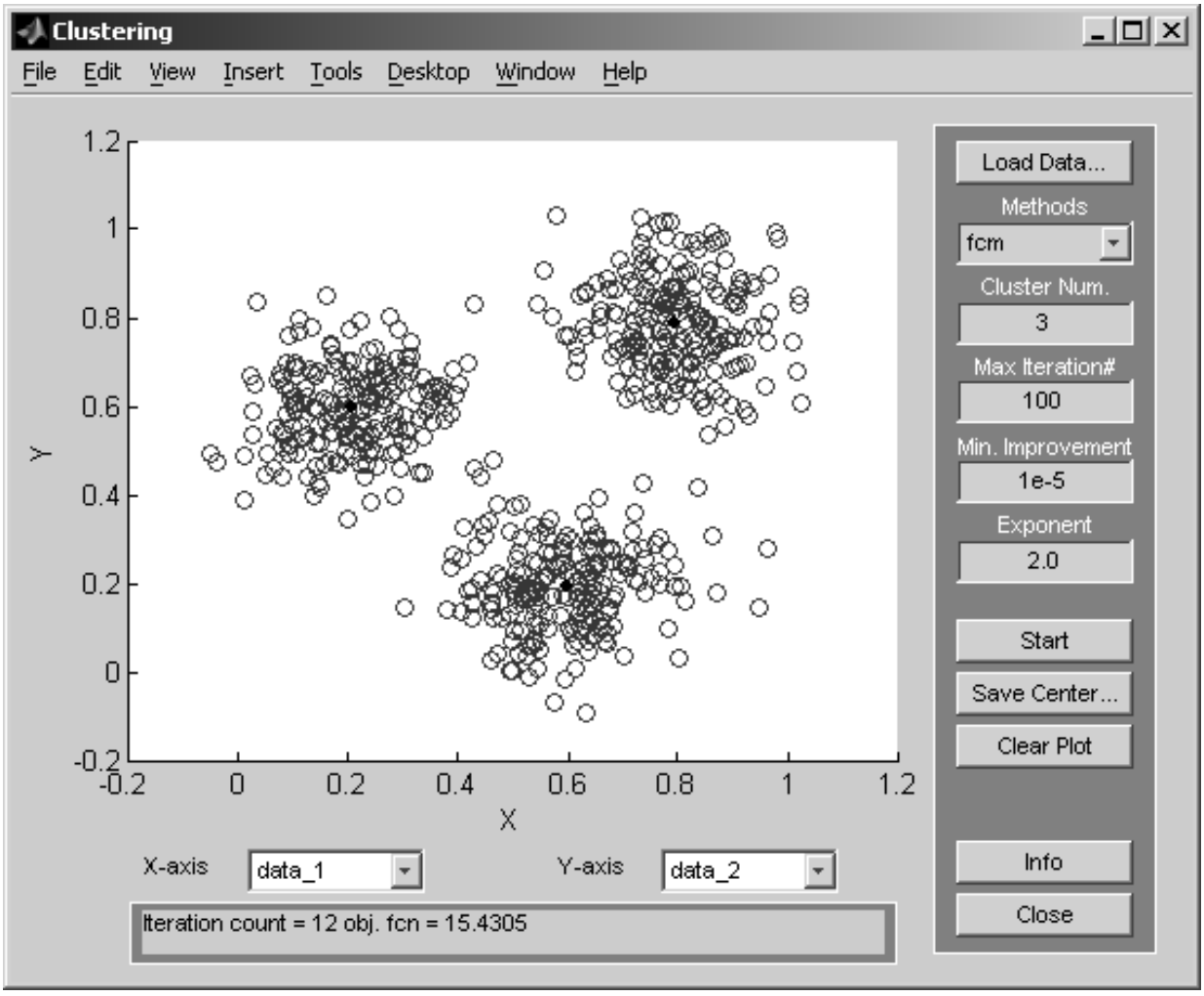


Рисунок 1.5 - Результат роботи програми Clustering (Центри кластерів пофарбовані в чорний колір).

При збільшенні кількості кластерів до 30 отримаємо результат, поданий на рис. 1.6.

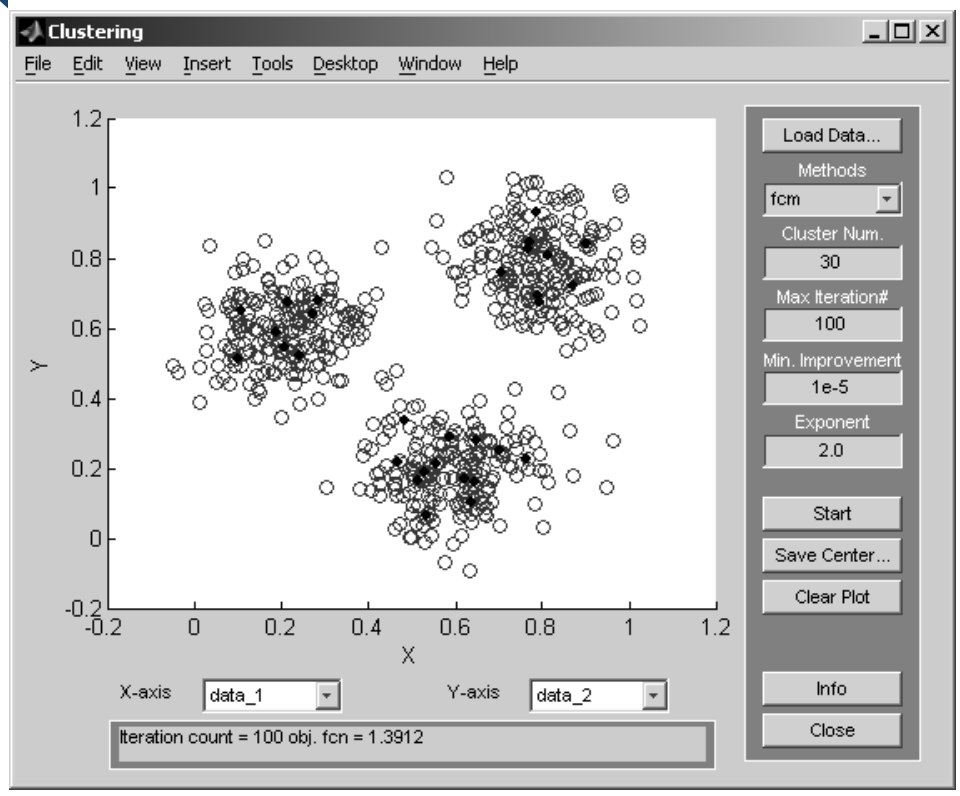


Рисунок 1.6 - Результат роботи програми Clustering (Центри кластерів пофарбовані в чорний колір).

Приклад.

Функція командного рядка fem призначена для вирішення задачі нечіткої кластеризації з використанням алгоритму FCM. Вона може бути викликана в одному з наступних форматів:

[center, U, obj_fcn) = fcm(data, cluster_n) або

[center, U, obj_fcn] = fcm(data, cluster_n, options)

Вхіднимн аргументами цієї функції є:

- **data**: матриця початкових даних X кластеризації, s-рядок якої представляє інформацію про один об'єкт нечіткої кластеризації $a_s \in A$ у формі векторе $x^s = (x^s_1, x^s_2, ..., x^s_N)$, x_{sj} – кількісне значення ознаки $p_j \in P$ для об'єкта даних $a_s \in A$, s=1, 2,..., S –кількість екземплярів кластеризації, N – кількість параметрів (ознак), що описують один екземпляр (або кластер);

- **cluster_n**: кількість шуканих нечітких кластерів $v \in V$ (більше одиниці).

Вихідними аргументами цієї функції є:

- center: матриця центрів шуканих нечітких кластерів $C = (C^1, C^2, ..., CV)^T$, v = 1, 2, ..., V, кожен рядок якої представляє собою координати центру одного з нечітких кластерів у формі вектор $C^v = (C^v_1, C^v_2, \ldots, C^v_N)$;

- U: матриця значень функцій приналежності шуканого нечіткого розбиття $u_s = (u^s_1, u^s_2, ..., u^s_V)$, $u^s_v \hat{I} [0,1]$;

obj_fcn: значення цільової функції J(x,u,C) на кожній з ітерацій роботи алгоритму FCM.

Функція fcm(data, cluster_n, options) може бути викликана з додатковими аргументами options, які призначені для управління процесом нечіткої кластеризації, а також для зміни критерію зупинки роботи алгоритму і відображення інформації на екрані монітора.

Ці додаткові аргументи мають наступні значення:

- options (1) : експоненціальна вага m для розрахунку матриці нечіткого розбиття U (за умовчанням це значення дорівнює т=2);
- options (2): максимальне число ітерацій k (за умовчанням це значення дорівнює k=100);
- options (3): параметр збіжності алгоритму e (за умовчанням це значення дорівнює e=0.00001);
- options (4): інформація про поточну ітерацію, що відображується на екрані монітора (за умовчанням це значення дорівнює 1).

Якщо будь-яке із значень додаткових аргументів дорівнює NAN (не число), то для цього аргументу використовується значення за умовчанням. Функція fcm закінчує свою роботу, коли алгоритм FCM виконає максимальну кількість ітерацій k, або коли різниця між значеннями цільових функцій J(x,u,C) на двох послідовних ітераціях буде менше заданого апріорі значення параметра збіжності алгоритму e.

Функція *fcm* реалізована у вигляді m-файлу і використовує, у свою чергу, три інші функції: функцію *initfcm* для формування матриці вихідного розбиття деяким випадковим чином, функцію *distfcm* розрахунку матриці відстаней між точками даних та центрами кластерів та функцію *stepf cm* для значень цільової функцій належності об'єктів нечітким кластерам на кожній ітерації роботи алгоритму FCM. Всі ці функції також реалізовані у вигляді m-файлів і знаходяться у папці C:\MATLAB\toolbox\fuzzy\fuzzy.

Розглянемо множину даних, що містяться в системі MATLAB і використовуються як тестова сукупність об’єктів нечіткої кластеризації. Ці дані представляють собою матрицю спостережень X розмірністю 140x2, яка міститься у файлі fcmdata.dat, що поставляється разом з системою MATLAB. В цьому випадку матриця спостережень X відповідає 140 об’єктам, для кожного з яких виконані виміри по двох ознаках, що є зручним для візуалізації вихідних даних і результатів нечіткої кластеризації в двовимірному просторі на площині. Графік координат точок на площині, відповідних об’єктам нечіткої кластеризації, представлений на рис. 1.7.

У лістингу 1 наводиться послідовність команд, які забезпечують рішення задачі нечіткої кластеризації множини даних X і візуалізацію отриманих результатів. В даному прикладі використовується перший формат запису функції fcm.

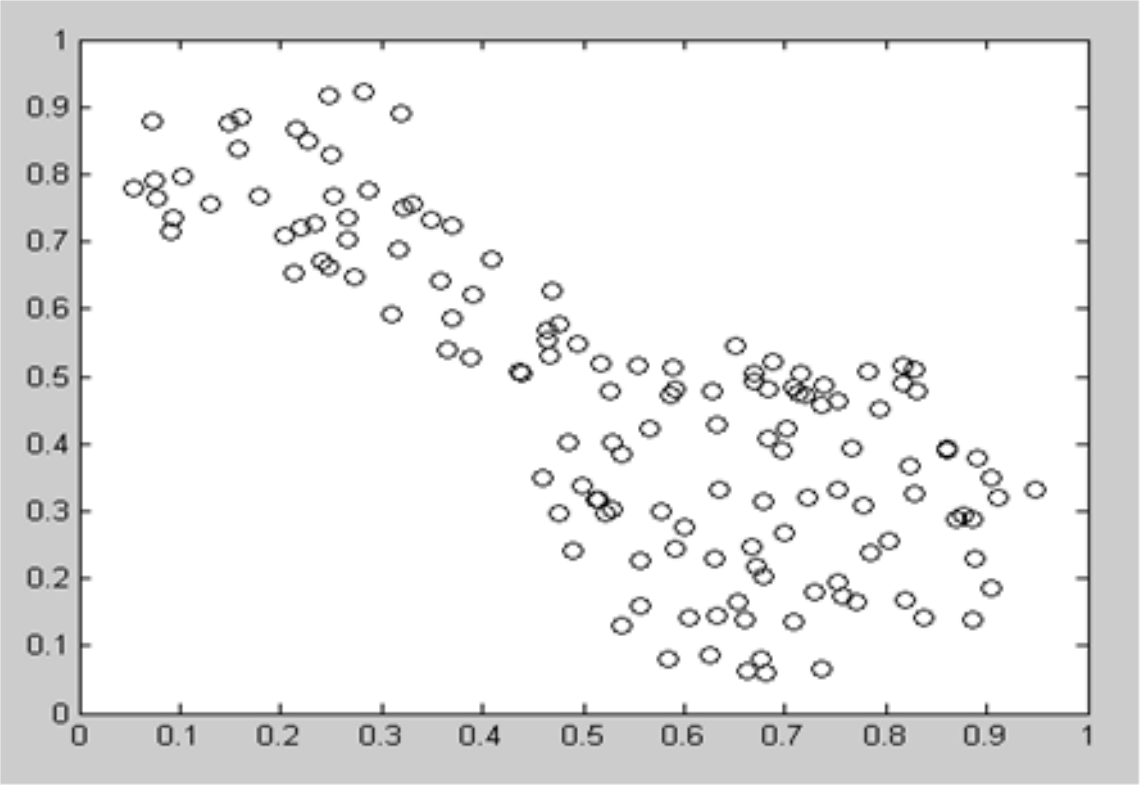


Рисунок 1.7 – Множина об'єктів нечіткої кластеризації

Лістинг 1

```
load fcmdata.dat
plot(fcmdata(:, 1), fcmdata(:,2), 'o', 'color', 'k')
[center, U, obj_fcm] = fcm(fcmdata, 2);
maxU =  max(U);
index1 = find(U(1, :) == maxU);
index2 = find(U(2, :)== maxU);
line(fcmdata(index1, 1), fcmdata(index1, 2), 'linestyle', 'none', 'marker', 'o',
'color', 'g');
line(fcmdata(index2, 1), fcmdata(index2, 2), 'linestyle', 'none', 'marker', 'x',
'color', 'r');
hold on
plot(center(1,1), center(1,2), 'ko', 'MarkerSize', 12, 'LineWidth', 2)
plot(center(2,1), center(2,2), 'kx', 'MarkerSize', 12, 'LineWidth', 2)
[center, U, obj_fcm] = fcm(fcmdata, 2)
```

Результат рішення задачі нечіткої кластеризації для двох нечітких кластерів з використанням вказаної послідовності команд може бути візуалізований (див. рис. 1.8).

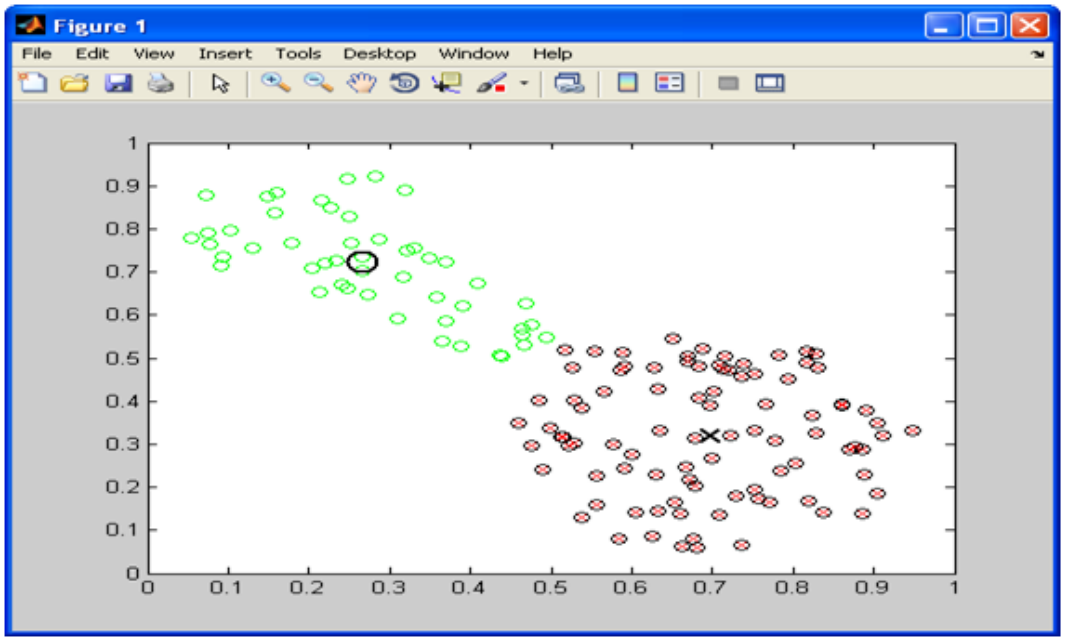


Рисунок 1.8 – Результат рішення задачі нечіткої кластеризації

Якщо після запису функції fcm в третьому рядку не вказувати крапку з комою (;), то у вікні команд відображуватимуться значення координат центрів нечітких кластерів, значення функцій приналежності об'єктів нечітким кластерам і значення цільової функції на кожній з ітерацій роботи алгоритму FCM (див. рис. 1.9).

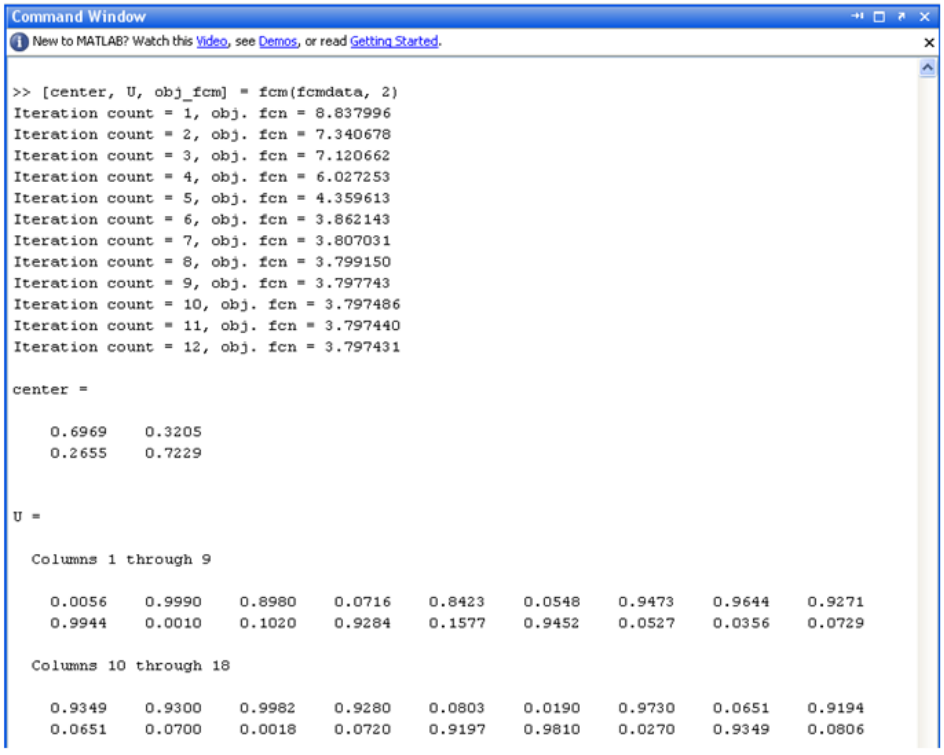


Рисунок 1.9 – Результат рішення задачі нечіткої кластеризації в командному вікні системи Matlab

Контрольні питання:

1. Яка програма використовувалася виявлення центрів кластерів?
2. Дайте визначення кластеризації.
3. Для яких завдань можна використовувати кластеризація?
4. Як можна визначити відстань між кластерами?
5. Назвіть етапи алгоритму кластерного аналізу.
6. Як визначаються центри кластерів?

1.2. Кластеризація за допомогою нейронних мереж

Цель роботи: освоїти основні принципи вирішення задачі кластеризації з використанням нейронних мереж з шаром Коханена і карт, що самоорганізуються.

Завдання: використовуючи вбудовані функції пакету нейронних мереж математичного середовища MATLAB вирішити обрану задачу кластеризації, а також розглянути використання карт, що самоорганізуються..

Поняття кластеризації. Завдання кластеризації (категоризації, класифікації " без вчителя ") – завдання розміщення вхідних векторів (образів) за категоріями (кластерам), те щоб близькі вектори (подібні образи) опинилися у однієї категорії. Відмінність завдання кластеризації від схожої неї завдання класифікації у тому, що набір категорій спочатку не заданий й у процесі навчання нейронної мережі. Прикладом завдання кластеризації є завдання стиснення інформації шляхом зменшення різноманітності даних.

Кластеризація може бути використана для вирішення наступних завдань:

- обробка зображення;
- класифікація;
- тематичний аналіз колекцій документів;
- побудова репрезентативної вибірки.

Методи кластеризації з допомогою нейронних мереж є розвитком класичних методів кластеризації. Наприклад, метод кластеризації векторів з допомогою мережі Кохонена містить у основі метод *K середніх*. У той же час нейронні мережі є набагато більш гнучким інструментом у застосуванні до даних, що мають великий об'єм та надмірну розмірність..

Приклад виконання:

Задача. Використовуючи вбудовані функції пакета нейронних мереж математичного середовища MATLAB, побудувати нейронну мережу з шаром Коханена, яка безліч вхідних даних (сенсорів)

розділить на кластери і виявить їх центри (місто встановлення шлюзу). На навчену мережу подати новий вхідний вектор і визначити до якого кластера він належить.

Для створення нейронної мережі із шаром Коханена скористаємося вбудованою в середу MATLAB функцією ***newc***:

```
X=[0 1;0 1];                    % Завдання діапазону зміни елементів
clusters=5;                     % Завдання кількість кластерів
points=5;                       % Завдання кількість точок у кластері
std_dev=0.01;
P=nngenc(X,clusters,points,std_dev);    % Моделювання вхідних даних
h=newc([0 1; 0 1],5,.1)                 % створення шару Коханена
h.trainParam.epochs=50;                 % Завдання кількості циклів навчання
h=train(h,P)
w=h.IW{1};
% виведення графіків вихідних даних та виявлених центрів кластерів
plot(P(1,:),P(2,:),'^r'),grid;
hold on; plot(w(:,1),w(:,2),'ob');
xlabel('p(1)');
ylabel('p(2)');
A=0.6
B=0.5
p=[A;B];                        % Завдання нового вхідного вектора
plot(A,B,'+k')
y=sim(h,p)                      % Опитування мережи
```

Результат роботи програми представлено на рис. 1.10. Крім того, його можна побачити у командному вікні:

y = (4,1) 1

Пред'явлений вектор віднесено до четвертого кластера.

Розглянемо застосування нейронної мережі із шаром Кохонена для кластеризації масиву даних із файлу clusterdemo.dat, вже використаного нами. Цей масив має розмірність 600x3. Використовуємо з файлу матрицю, що складається з перших двох стовпців і всіх рядків. Визначимо для цієї множини 9 кластерів, після чого подамо новий вектор і визначимо номер кластера, до якого він віднесений:

```
load clusterdemo.dat
P=[clusterdemo(:,1)'; clusterdemo(:,2)']*100
% створюємо НС Кохонена з 9 кластерами (нормальний росто-ваговий показник, надлишок ваги та недолік ваги)
```

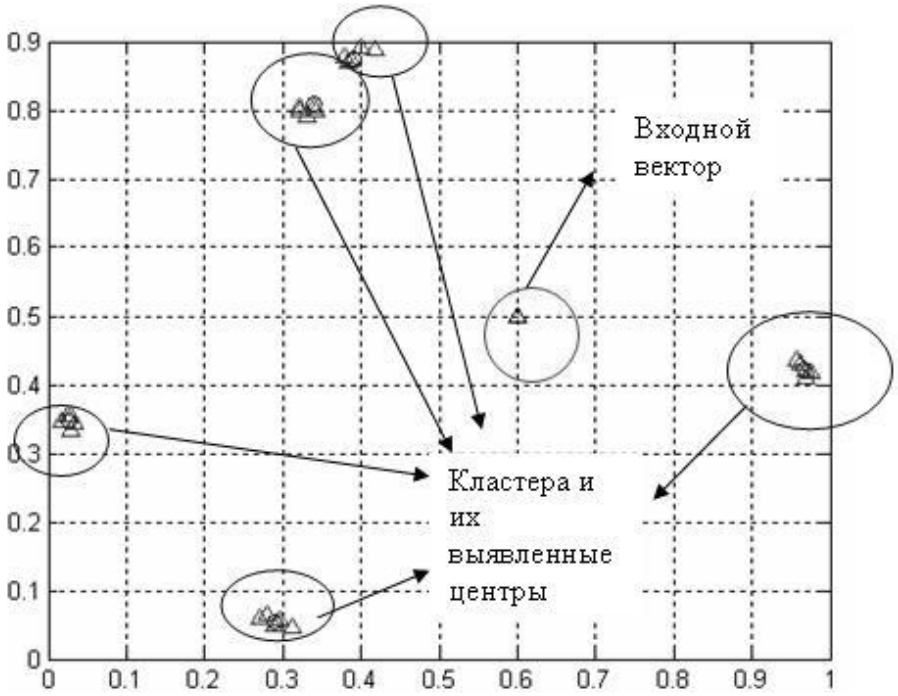


Рисунок 1.10 – Виявлені центри кластерів

```
h=newc([0 105; 0 105],9,.1)
h.trainParam.epochs=100; % Завдання кількості циклів навчання
h=train(h,P)
w=h.IW{1}
plot(P(1,:),P(2,:),'^r');
hold on; plot(w(:,1),w(:,2),'ob');
A=62
B=50
p=[A;B]; % Задание нового входного вектора
plot(A,B,'+k'),grid
y=sim(h,p)
```

Результат роботи програми представлено на рис. 1.11. Крім того, його можна побачити у командному вікні:

```
w =
 82.0621 89.6343
 60.3012 8.5841
 15.0290 66.3286
 46.3310 19.4929
 67.0821 27.2685
 86.8517 69.9695
 34.1376 64.4572
 14.8135 47.9045
 69.8167 73.3353
y = (5,1) 1
```

Пред'явлений вектор віднесено до п'ятого кластера, центр якого має координати. (*67.0821, 27.2685*).

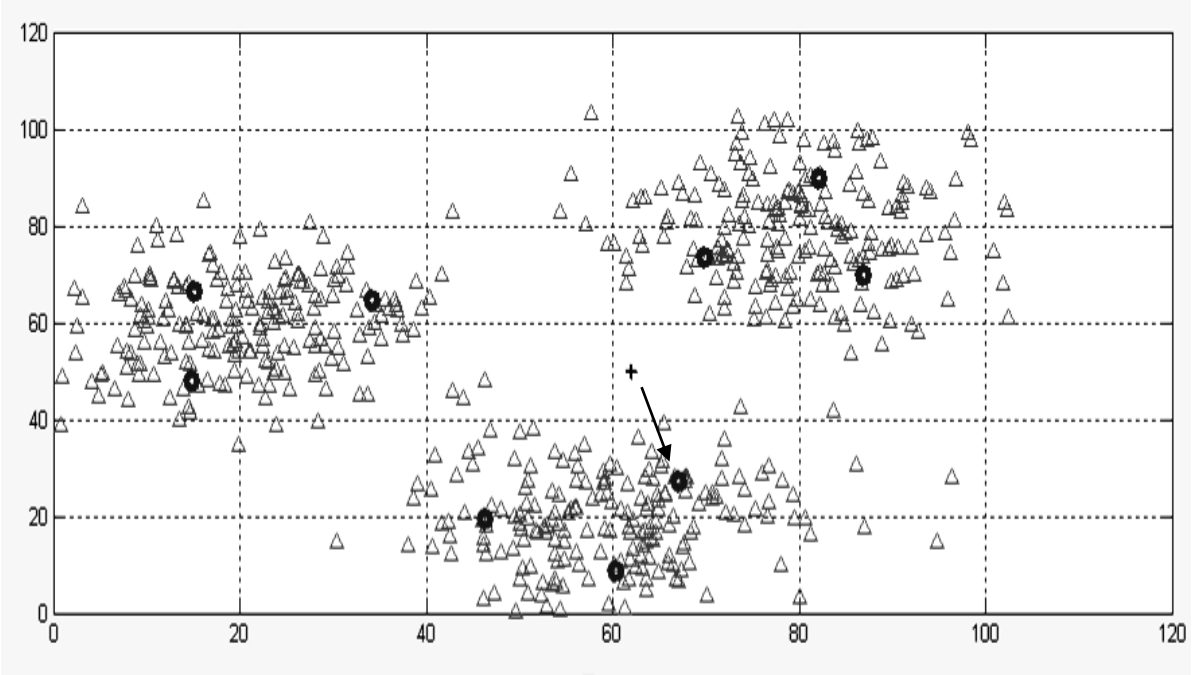


Рисунок 1.11 - Виявлені кластери (плюс на рисунку – вхідний вектор)

1.3. Самоорганізовані карти

Розглянемо поняття самоорганізованої карти. *Самоорганізовані карти* (Self Organizing Maps – SOM) – це один з різновидів нейромережевих алгоритмів. Основною відмінністю даної технології від розглянутих нами раніше нейромереж, які навчаються за алгоритмом зворотного поширення, і те, що з навчанні використовується метод навчання без вчителя, тобто результат навчання залежить від структури вхідних даних. Нейронні мережі даного типу часто застосовуються для вирішення найрізноманітніших завдань, від відновлення перепусток у даних до аналізу даних та пошуку закономірностей, наприклад, у фінансовій задачі.

Алгоритм функціонування карток, що самонавчаються, являє собою один з варіантів кластеризації багатовимірних векторів. Прикладом таких алгоритмів може бути алгоритм k-найближчих середніх (c-means). Важливою відмінністю алгоритму SOM є те, що в ньому всі нейрони (вузли, центри класів) упорядковані до певної структури (зазвичай двовимірної сітки). При цьому під час навчання модифікується не тільки нейрон-переможець, а й його сусіди, але меншою мірою. За рахунок цього SOM можна вважати одним із методів проектування багатовимірного простору у простір з більш низькою розмірністю. З використанням цього алгоритму вектора, схожі у вихідному просторі, виявляються поруч і отриманої карті.

Структура самоорганізованих карт. SOM передбачає використання впорядкованої структури нейронів. Зазвичай використовуються одне та двовимірні сітки. При цьому кожен нейрон є *n*-мірний вектор-стовбець $\mathbf{w} = [w_1, w_2, ..., w_n]^T$, де *n* - визначається розмірністю вихідного простору (розмірністю вхідних векторів). Застосування одна і двовимірних сіток пов'язано з тим, що виникають проблеми при відображенні просторових структур більшої розмірності (при цьому знову виникають проблеми зі зниженням розмірності до двовимірної монітора).

Зазвичай нейрони розташовуються у вузлах двомірної сітки з прямокутними або шестикутними осередками (рис. 1.12). При цьому, як було зазначено вище, нейрони також взаємодіють один з одним. Розмір цієї взаємодії визначається відстанню між нейронами на карті. На малюнку дано приклад відстані для шестикутної та чотирикутної сіток.

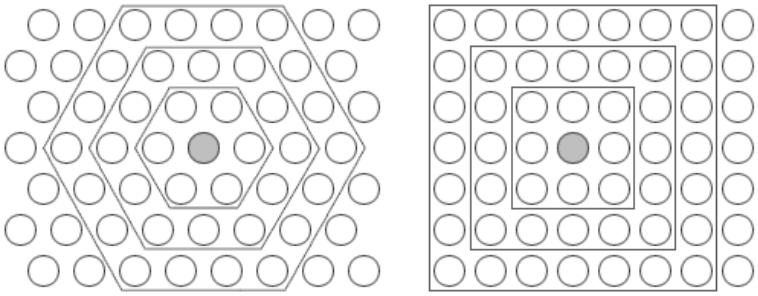


Рисунок 1.12 - Розташування нейронів у вузлах двовимірної сітки з чотирикутними та шестикутними осередками

Легко помітити, що з шестикутної сітки відстань між нейронами більше збігається з евклидовой відстанню, ніж чотирикутної сітки.

При цьому кількість нейронів у сітці визначає ступінь деталізації результату роботи алгоритму, і, зрештою, від цього залежить точність узагальнюючої здатності карти.

Початкова ініціалізація картки. При реалізації алгоритму SOM заздалегідь визначається конфігурація сітки (прямокутна або шестикутна), а також кількість нейронів в мережі. Деякі джерела рекомендують використовувати максимально можливу кількість нейронів у карті. При цьому початковий радіус навчання (neighborhood в англомовній літературі) значною мірою впливає на здатність узагальнення за допомогою отриманої картки. У разі коли кількість вузлів карти перевищує кількість прикладів у навчальній вибірці, то успіх використання алгоритму великою мірою залежить від вибору початкового радіусу навчання. Однак, у випадку, коли розмір карти становить десятки тисяч нейронів, то час, необхідний на навчання карти зазвичай буває занадто велике для вирішення практичних завдань,

таким чином, необхідно досягати допустимого компромісу при виборі кількості вузлів.

Перед початком навчання картки необхідно проініціалізувати вагові коефіцієнти нейронів. Вдало обраний спосіб ініціалізації може суттєво прискорити навчання, і призвести до отримання якісніших результатів. Існують три способи ініціювання початкових ваг.

- Ініціалізація випадковими значеннями, коли всі ваги дають малі випадкові величини.
- Ініціалізація прикладами, коли як початкові значення задаються значення випадково вибраних прикладів з навчальної вибірки
- Лінійна ініціалізація. В цьому випадку ваги ініціюються значеннями векторів, лінійно впорядкованих уздовж лінійного підпростору, що проходить між двома головними власними векторами вихідного набору даних. Власні вектори можуть бути знайдені, наприклад, за допомогою процедури Грама-Шмідта..

Навчання самоорганізованих карт. Навчання складається з послідовності корекцій векторів, що являють собою нейрони. На кожному кроці навчання з вихідного набору даним випадково вибирається один із векторів, а потім проводиться пошук найбільш схожого на нього вектора коефіцієнтів нейронів. При цьому вибирається нейрон-переможець, який найбільше схожий на вектор входів. Під схожістю в даній задачі розуміється відстань між векторами, що зазвичай обчислюється в евклідовому просторі. Після того, як знайдений нейрон-переможець, проводиться коригування ваг нейромережі. При цьому вектор, що описує нейрон-переможець і вектора, що описують його сусідів у сітці, переміщуються в напрямку вхідного вектора.

Наведемо приклад використання карти, що самоорганізується, на прикладі двовимірних векторів. Використовуючи карти, що самоорганізуються, двомірні вектори необхідно розбити на кластери, потім подати на вхід самоорганізуючої карти новий вектор і визначити кластер до якого він відноситься.

```
P=rands(2,100)          % Завдання випадкових двомірних вхідних
векторів
figure(1)
hold on
plot(P(1,:),P(2,:),'+r')        % візуальне зображення вхідних векторів
%Створення СР з 3*4 нейронами
%За умовчанням функція TFCN = 'hextop', тобто нейрони
розташовуються у вузлах двомірної сітки з шестикутними
осередками
net=newsom([0 1;0 1],[3 4]);
net.trainParam.epoch=1    % Завдання числа циклів налаштування
net=train(net,P)          % налаштування мережи
```

```
A=0.5
B=0.3
p=[A;B];                    % Завдання нового вхідного вектору
plot(A,B,'^k') % промальовування на малюнку вхідного вектору
(чорний трикутник)
figure(2)
plotsom(net.iw{1,1},net.layers{1}.distances)
a=sim(net,p)                % опитування мережи
```

Результат роботи програми представлено на рис. 1.13, 1.14.

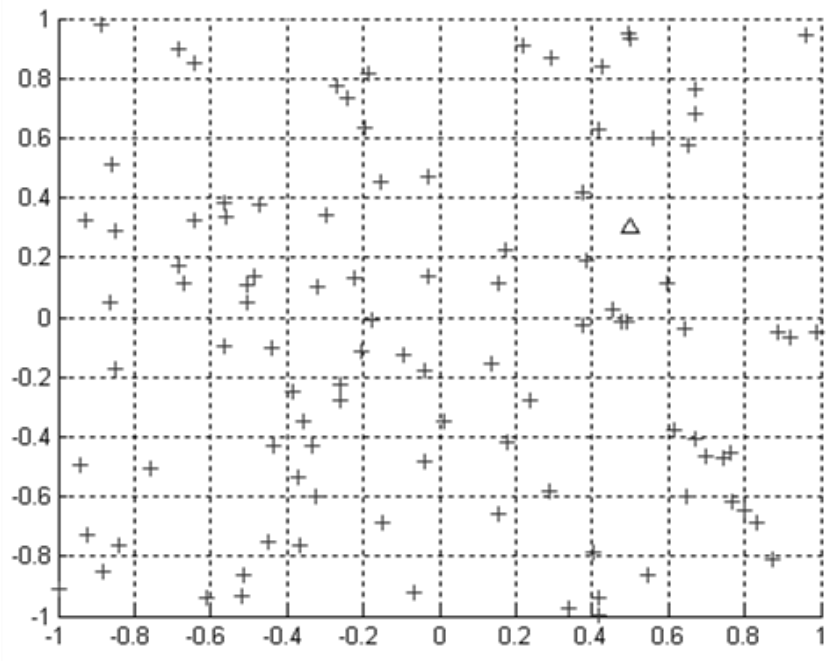


Рисунок 1.13 – Вхідні вектори

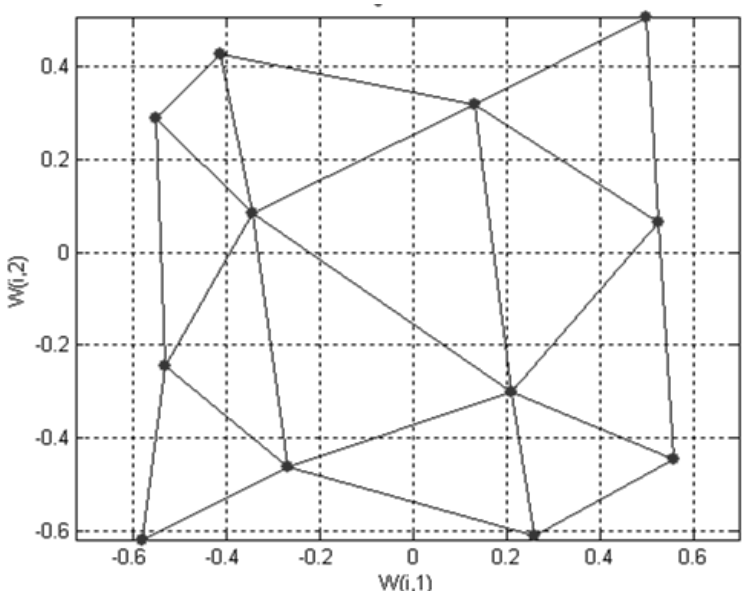


Рисунок 1.14 – Виявлені центри кластерів

Результат роботи програми можна побачити і у командному вікні:

a =

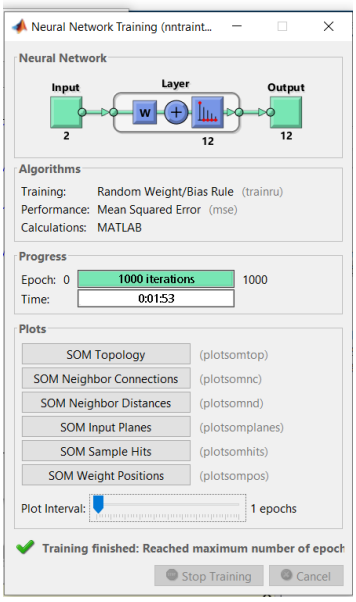
(12,1) 1

Пред'явлений вектор віднесено до дванадцятого кластера.

Приклад. Побудови мережі SOM/

```
load clusterdemo.dat
P=[clusterdemo(:,1)'; clusterdemo(:,2)']*100
figure(1)
hold on
plot(P(1,:),P(2,:),'+r')          % візуальне зображення вхідних векторів
%Створення СР з 3*4 нейронами
%За умовчанням функція TFCN = 'hextop', тобто нейрони розташовуються у вузлах двомірної сітки з шестикутними осередками
net=newsom([0 1;0 1],[3 4]);
net.trainParam.epoch=1      % Завдання числа циклів налаштування
net=train(net,P)            % налаштування мережи
A=0.5
B=0.3
p=[A;B];                    % Завдання нового вхідного вектору
plot(A,B,'^k') % промальовування на малюнку вхідного вектору (чорний трикутник)
figure(2)
plotsom(net.iw{1,1},net.layers{1}.distances)
a=sim(net,p)                % опитування мережи
```

Результат виконання програми наведено на рисунках 1.15 – 1.17



Рисунко 1.15 – Результати навчання мережі SOM

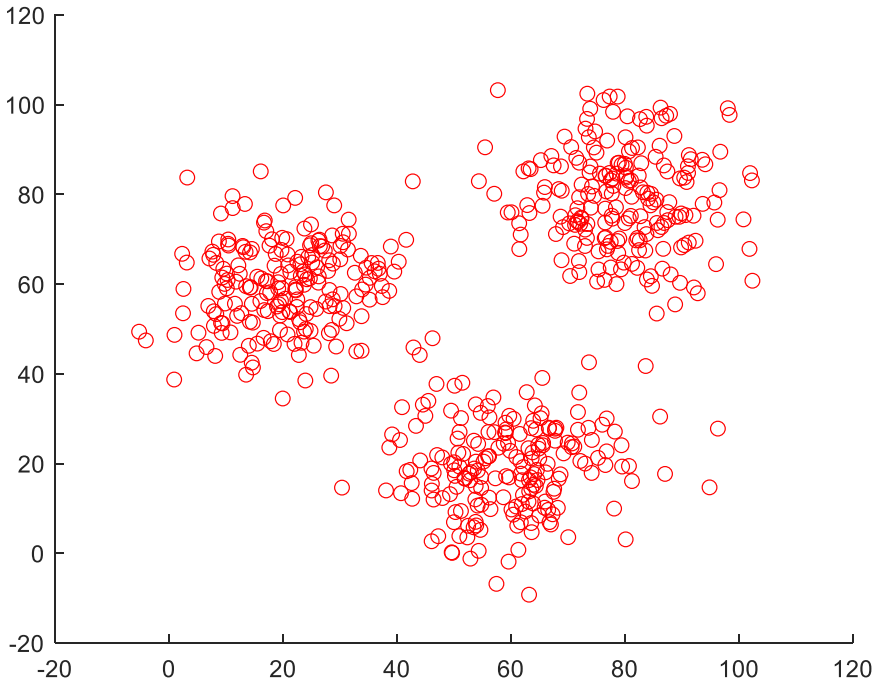


Рисунок 1.16 – Вхідні вектори

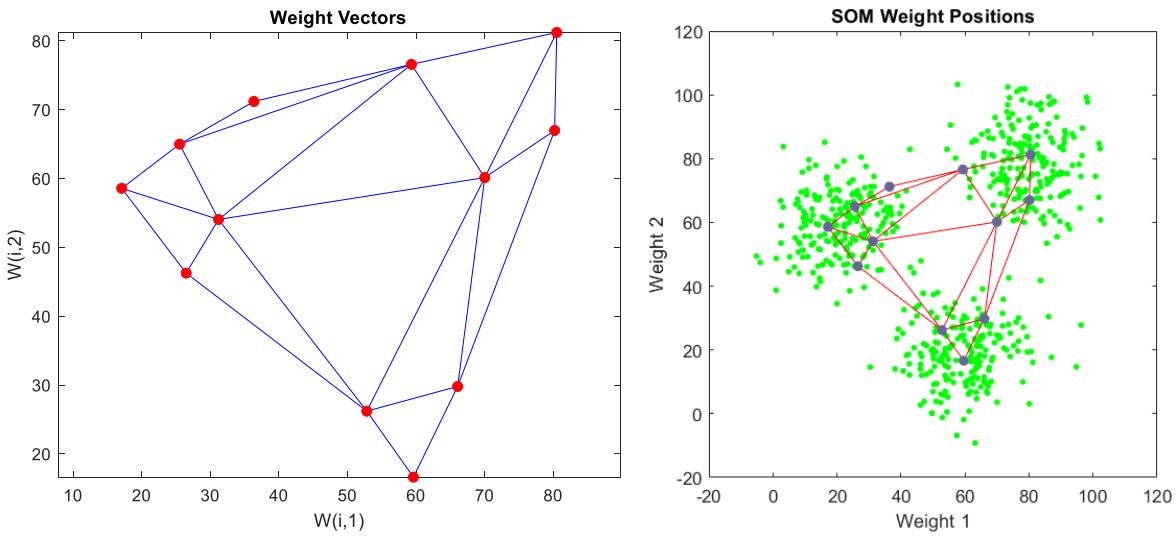


Рисунок 1.17 – Виявлені центри кластерів та дистанція міжними

Контрольні питання:

1. Що розуміємо під кластеризацією?
2. Опишіть вбудовані оператори MATLAB для кластеризації.
3. Опишіть мережу Кохонена.
4. Навіщо використовуються карти, що самоорганізуються.
5. Опишіть відмінність мережі Кохонена від SOM.

2 РОБОТА З НЕЧІТКИМИ МНОЖИНАМИ В MATLAB

Мета роботи: освоїти методику опису функцій приналежності у MatLab та виконання арифметичних обчислення над нечіткими множинами.

2.1 Опис функцій приналежності

У системі MatLab реалізовано набір команд для опису функцій приналежності (ФП) (див. табл. 2.1).

Таблиця 2.1 – Команди для опису функцій приналежності у MatLab

Команда	Функція
dsigmf	ФП у вигляді різниці між двома сигмоїдами
evalmf	Обчислення значень довільної ФП
evalmmf	Розрахунок ступенів приладдя для кількох ФП
gauss2mf	Двостороння гауссівська ФП
gaussmf	Гауссівська ФП
gbellmf	Узагальнена Дзвоноподібна ФП
pimf	π-подібна ФП
psigmf	Добуток двох сигмоїдних ФП
sigmf	Сигмоїдна ФП
smf	*s*-подібна ФП
trapmf	Трапецієподібна ФП
trimf	Трикутна ФП
zmf	*z*-подібна ФП
fuzarith	Калькулятор для виконання арифметичних операцій складання, віднімання, множення та поділу над нечіткими числами

Розглянемо приклади опису функцій приналежності (ФП) у MatLab. Для опису трикутної ФП використовується команда

```
>> y = trimf(x, [ABC]);
```

де *x* – базова шкала, де описується ФП; *A*, *B* та *C* – координати вершин трикутника на базовій шкалі. Наприклад:

```
x = (0:0.2:10);
y = trimf(x, [3 4 5]);
plot(x, y);
grid;
ylabel('y');
```

```
xlabel('x')
```

Отриманий графік показано на рис. 2.1.

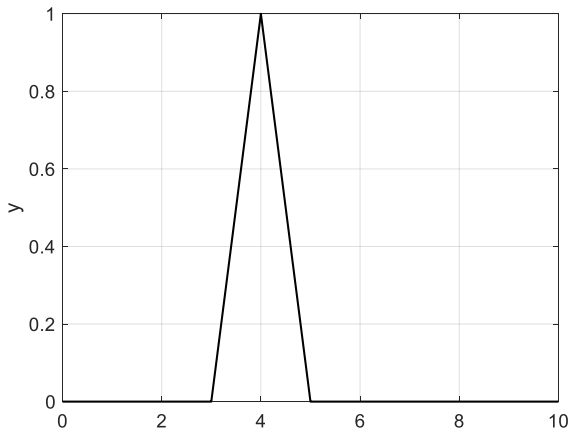


Рисунок 2.1 – Трикутна функція приналежності

Для опису трапецеподібної ФП використовується команда виду

```
>> y = trapmf(x, [A B C D]);
```

де x - Базова шкала; A, B, C та D – координати вершин трапеції на базовій шкалі.
Наприклад:

```
x = (0:0.2:10);
y = trapmf(x, [3 4 6 8]); plot(x, y)
grid;
ylabel('y');
xlabel('x')
```

Отриманий графік показано на рис. 2.2.

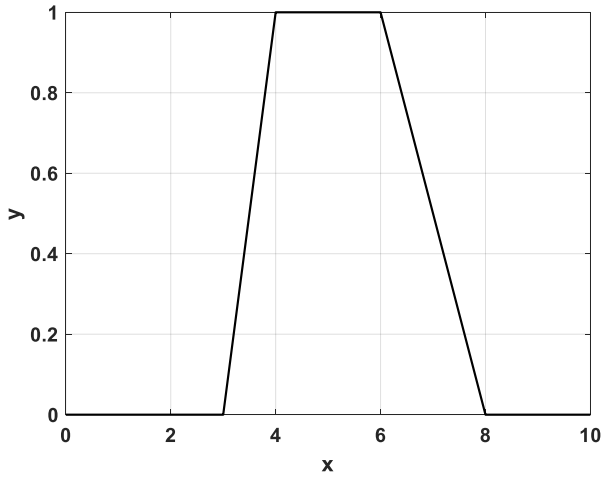


Рисунок 2.2 – Трапецієподібна функція приналежності

Для опису гаусової ФП використовується команда виду

```
>> y = gaussmf(x, [AB]);
```

Наприклад:

```
x = (0:0.2:10);
y = gaussmf(x, [1 5]); plot(x, y)
grid;
ylabel('y');
xlabel('x')
```

Отриманий графік показано на рис. 2.3.

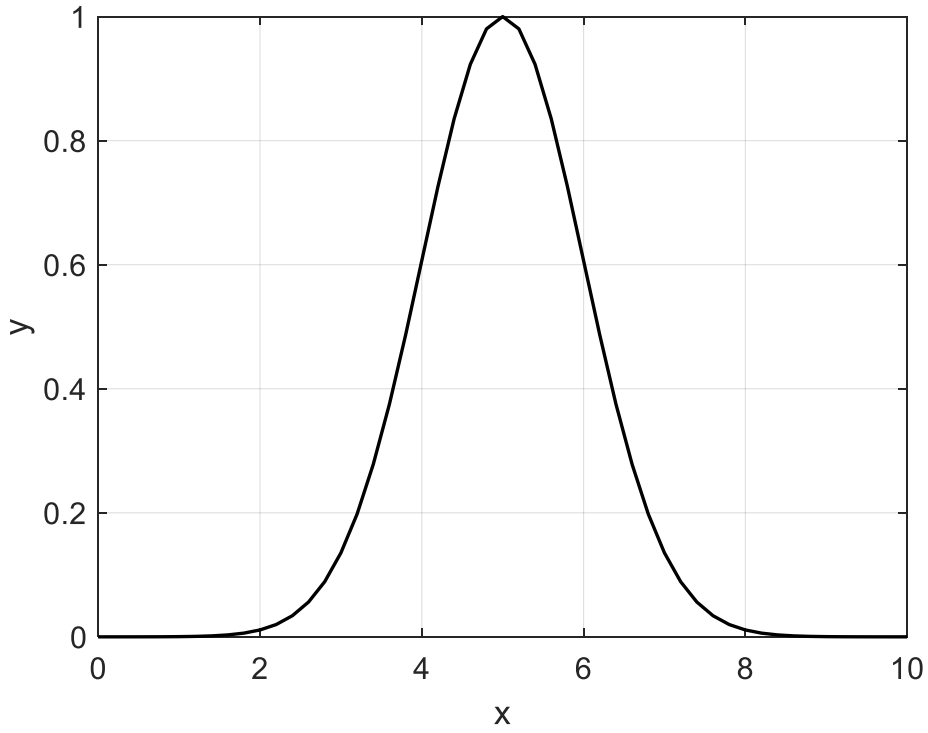


Рисунок 2.3 – Гаусовська функція приналежності

У MatLab передбачено варіант комбінації двох гаусових функцій, так що в проміжку між їхніми центрами значення ФП дорівнює 1.

```
>> y = gauss2mf(x,[B1 A1 B2 A2]);
```

Наприклад:

```
x = (0:0.2:10);
y = gauss2mf(x, [1 5 1.5 6]);
plot(x, y);
grid;
ylabel('y');
xlabel('x')
```

Отриманий графік показано на рис. 2.4.

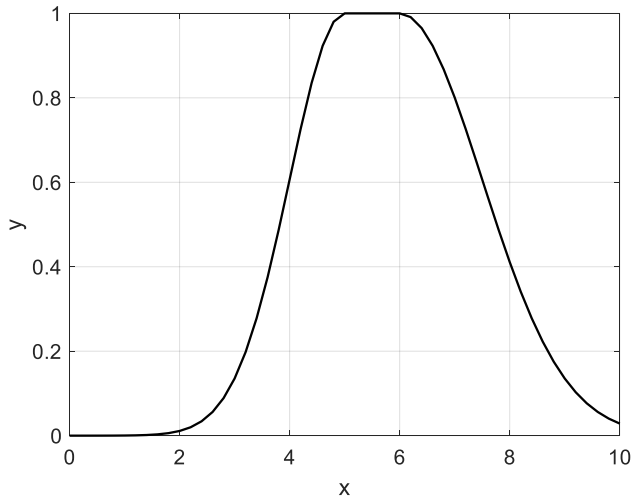


Рисунок 2.4 – Двостороння Гаусовська функція приналежності

Дзвоноподібна ФП описується командою:

```
>> y = gbellmf(x, [ABC]);
```

де A і B - параметри; C – центр ФП.

Розглянемо вплив параметра B:

```
x = (0:0.2:10);
y1 = gbellmf(x, [1 1 5]);
y2 = gbellmf(x, [1 2 5]);
y3 = gbellmf(x, [1 3 5]);
plot (x, y1, x, y2, x, y3);
grid
ylabel('y');
xlabel('x')
```

Сімейство графіків, що вийшло, показано на рис. 2.5.

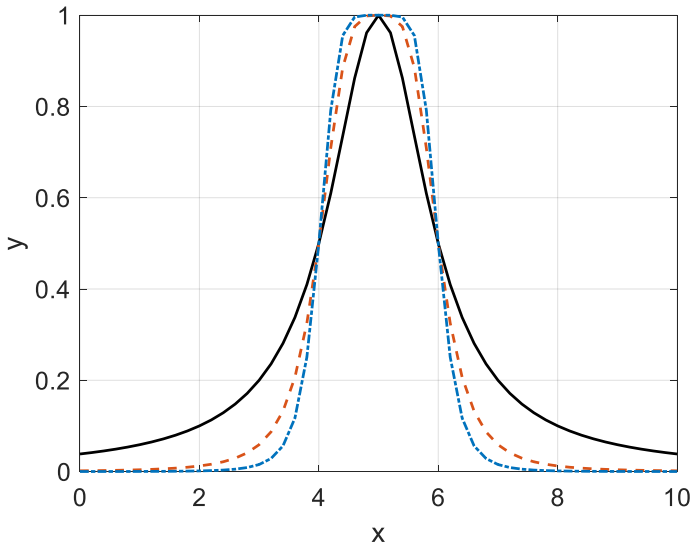


Рисунок 2.5 - Зміна «різкості» дзвоноподібної функції приналежності

Розглянимо вплив параметра *A*:

```
x = (0:0.2:10);
y1 = gbellmf(x, [1 1 5]);
y2 = gbellmf(x, [2 1 5]);
y3 = gbellmf(x, [3 1 5]);
plot(x,y1,x,y2,x,y3);
grid;
ylabel('y');
xlabel('x')
```

Сімейство графіків, що вийшло, показано на рис. 2.6.

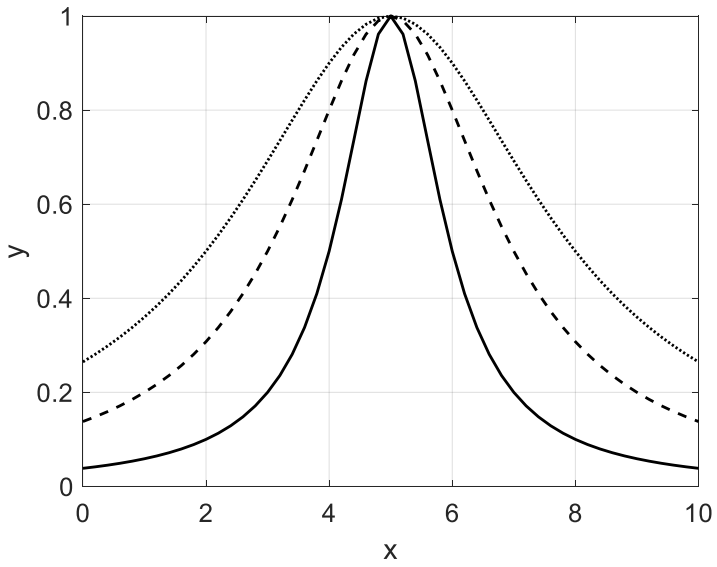


Рисунок 2.6 - Зміна «ширини» дзвоноподібної функції приналежності

Для визначення сигмоїдної ФП використовується команда виду

```
>> y = sigmf(x, [AB]);
```

Для з'ясування впливу параметрів A та B побудуємо сімейство графіків:

```
x = (0:0.2:10);
y1 = sigmf(x, [3 5]);
y3 = sigmf(x, [1 5]);
y2 = sigmf(x, [2 5]);
plot(x,y1,x,y2,x,y3)
grid;
ylabel('y');
xlabel('x')
```

Як свідчить рис. 2.7, параметр *B* визначає координату x, в якою функція приналежності перетинає значення 0,5. Параметр *A* визначає "розмитість" сигмоїдної функції.

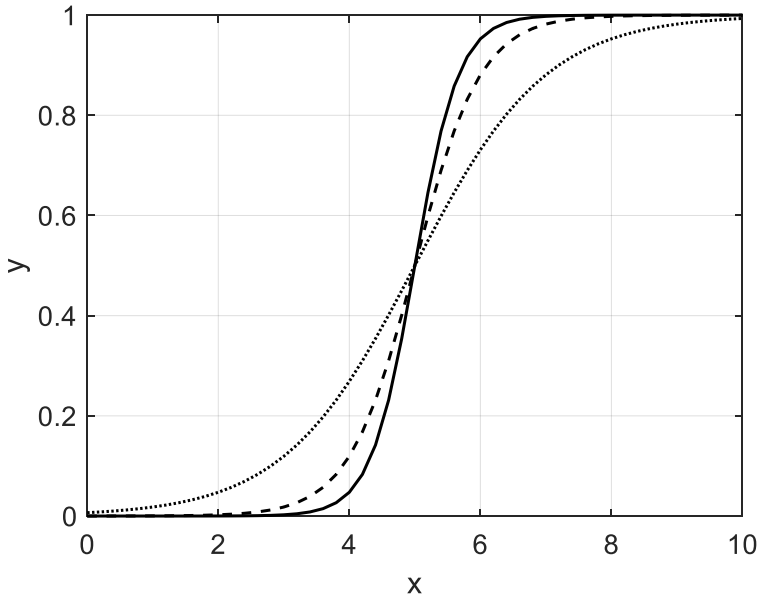


Рисунок 2.7 – Вплив параметрів на опис сигмоїдної функції

Вибираючи негативне значення *A*, можна отримати правосторонню сигмоїдну функцію (див. рис. 2.8):

```
x = (0:0.2:10);
y1 = sigmf(x, [-3 5]);
y2 = sigmf(x, [-4 5]);
y3 = sigmf(x, [-1 5]);
plot(x,y1,x,y2,x,y3)
grid;
ylabel('y');
xlabel('x')
```

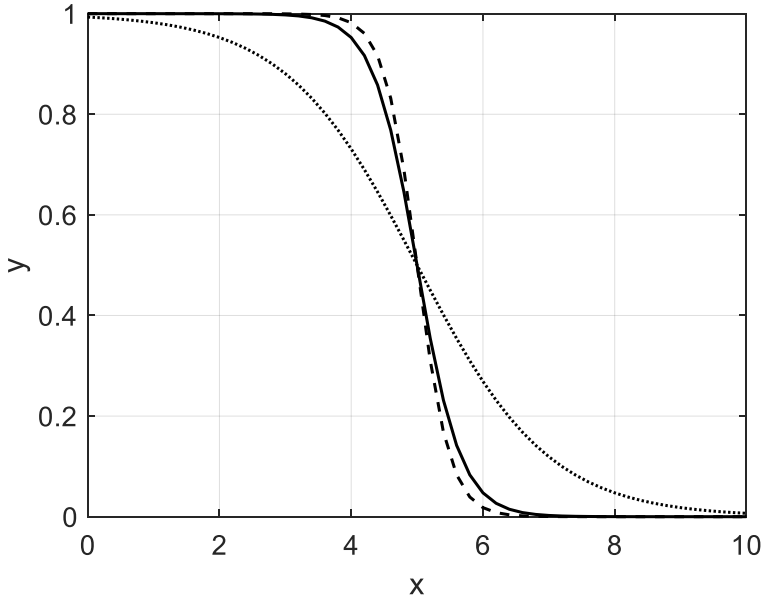


Рисунок 2.8 – Правостороння сигмоїдна функція

За допомогою функції *psigmf* може бути використаний добуток «лівосторонньої» та «правосторонньої» сигмоїдних функцій, наприклад:

```
x = (0:0.2:10);
params1 = [2 3];
params2 = [-5 8];
y = psigmf(x, [params1 params2]);
plot(x,y);
grid;
ylabel('y');
xlabel('x')
```

Графік добутоку сигмоїдних функцій показаний на рис. 2.9.

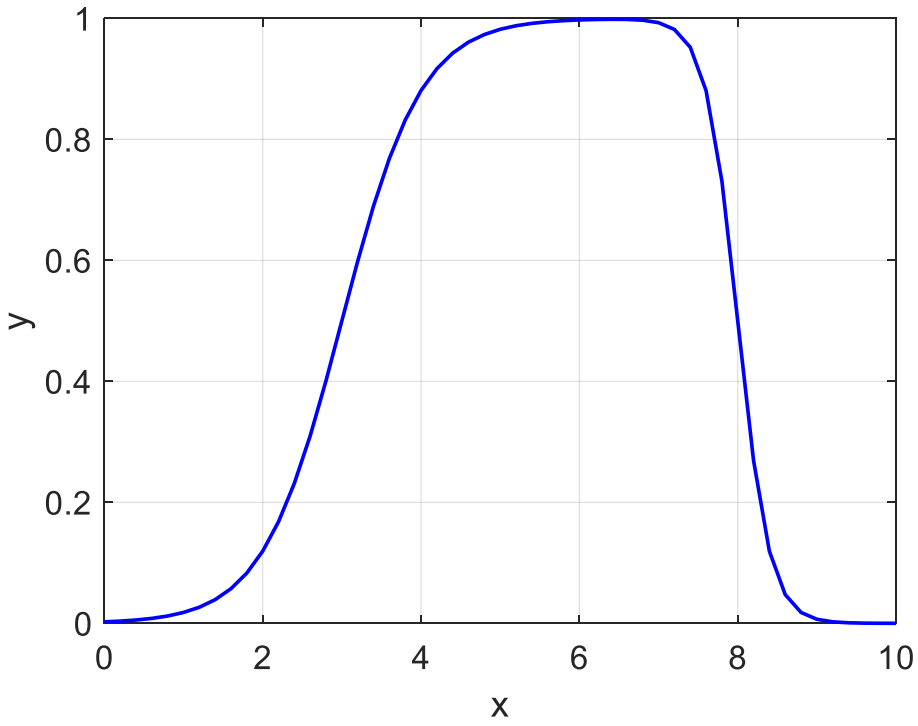


Рисунок 2.9 – Добуток сигмоїдних функцій

Різниця сигмоїдних функцій задається командою виду:

```
>> dsigmf(x,[A1 B1 A2 B2])
```

Наприклад:

```
x = (0:0.2:10);
y=dsigmf(x,[5 2 5 7]);
plot(x,y);
grid;
ylabel('y');
xlabel('x')
```

Ця команда відповідає наступній групі команд:

```
x = (0:0.2:10);
y1 = sigmf(x, [5 2]);
y2 = sigmf(x, [5 7]);
y3 = y1-y2;
plot(x,y3);
grid;
ylabel('y');
xlabel('x')
```

Графік ФП показано на рис. 2.10.

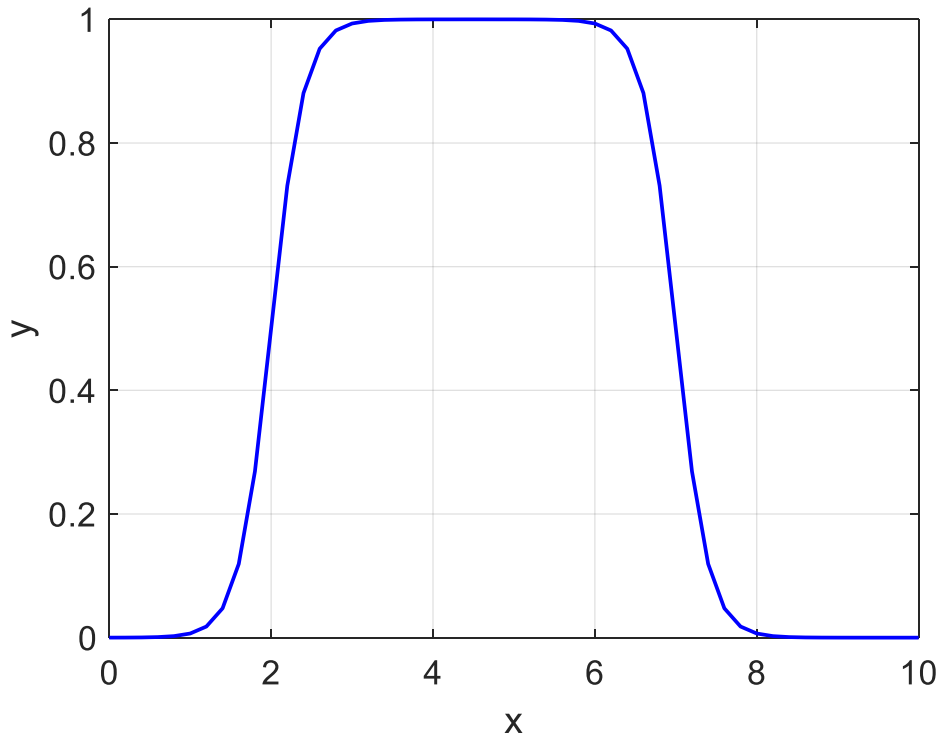


Рисунок 2.10 – Різниця сигмоїдних функцій

На основі сплайн-апроксимації побудована *Z*-подібна ФП, яка дозволяє описати плавне зменшення приналежності від *A* до *B*:

```
>> zmf(x,[a b])
```

Наприклад:

```
x = (0:0.2:10);
plot(x, zmf(x, [2 8]));
grid;
ylabel('y');
xlabel('x')
```

Графік цієї функції показано на рис. 2.11.

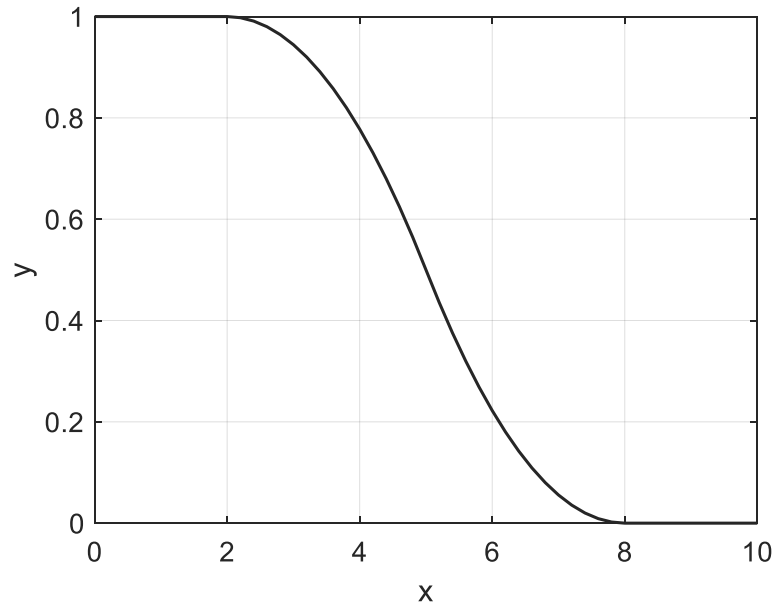


Рисунок 2.11 – Графік Z-подібна функції приналежності

S-подібна ФП є «парною» до Z-подібної функції:

```
>> smf(x,[a b])
```

Наприклад:

```
x = (0:0.2:10);
y=smf(x,[1 8]);
plot(x,y);
grid;
ylabel('y');
xlabel('x')
```

Графік Z-подібної функції показано на рис. 2.12.

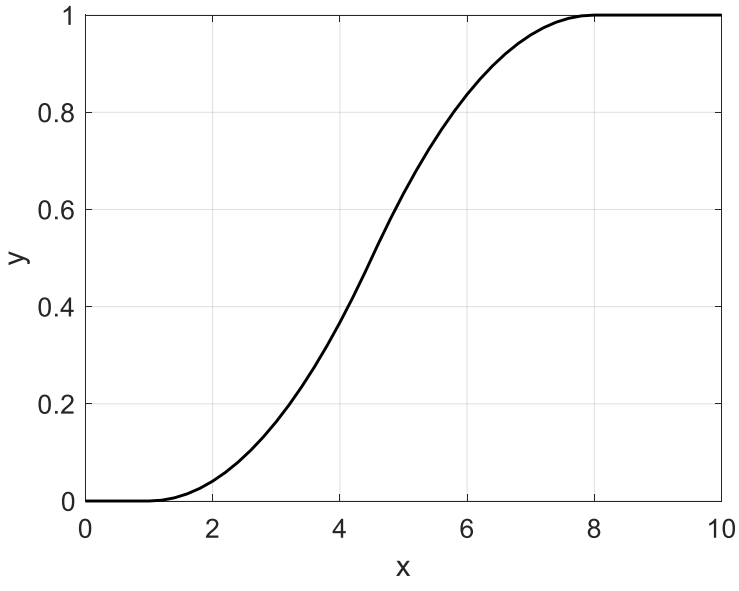


Рисунок 2.12 – Графік Z-подібної функції приналежності

Для опису π-подібної функції приналежності використовуються чотири параметри:

```
x = (0:0.2:10);
y = pimf(x, [2 5 8 9]);
plot(x,y);
grid;
ylabel('y');
xlabel('x')
```

Графік, що вийшов, показаний на рис. 2.13.

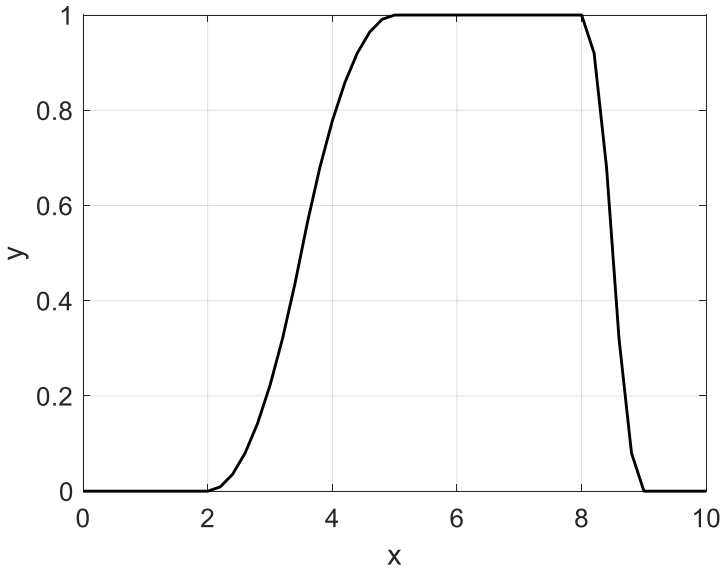


Рисунок 2.13 – Графік π-подібної функції приналежності

За допомогою функції evalmf можна оцінити ступінь приналежності елементів заданого вхідного вектора до нечіткої множини. Функція має формат:

y = evalmf (x, params, type),

де *x* - вектор, для координат якого необхідно розрахувати ступеня належності; *params* - вектор параметрів функції приналежності, порядок завдання яких визначається її типом; *type* – тип функції приналежності, який може бути заданий ім'ям функції або її кодом: 1 – *trimf*; 2 - *'trapmf'*; 3 - *'gaussmf'*; 4 - *'gauss2mf'*; 5 - *'sigmf'*; 6 - *'dsigmf'*; 7 - *'psigmf'*; 8 - *'gbellmf'*; 9 - *'smf'*; 10 - *'zmf'*; 11 - 'pimf'.

При заданні іншого типу функції приналежності передбачається, що її визначено користувачем і задана відповідним m-файлом.
Розглянемо приклад

```
x = [3 4 5];
y = evalmf (x, [2 5 8 9], 'pimf')
```

```
y =
0.2222 0.7778 1.0000
```

Обчислити значення кількох функцій приналежності нечітких множин, заданих на одній й тій же універсальній множині, можна з допомогою функції

y = evalmmf(x, params, types),

де *x* - вектор, для координат якого необхідно розрахувати ступеня належності; *params* – матриця властивостей функції приналежності. Перший рядок матриці визначає параметри першої функції приладдя, другий рядок – параметри другої функції приналежності тощо; *types* – матриця типів функції приналежності. Перший рядок матриці визначає тип першої функції приналежності, другий рядок – тип другої функції приналежності (див. опис команди *evalmf*).
Розглянемо приклад.

```
mf = [fismf(@gaussmf,[1.5 5]) fismf(@trapmf,[3 4 6 7])];
x = (-2:0.1:12)';
y = evalmmf(mf,x);
plot(x,y); grid;
xlabel('Universe of discourse (x)'),ylabel('Membership value (y)')
legend('gaussmf, P=[1.5 5]','trapmf, P=[3 4 6 7]')
```

Результат обчислення функцій приналежності наведено на рисунку 2.14

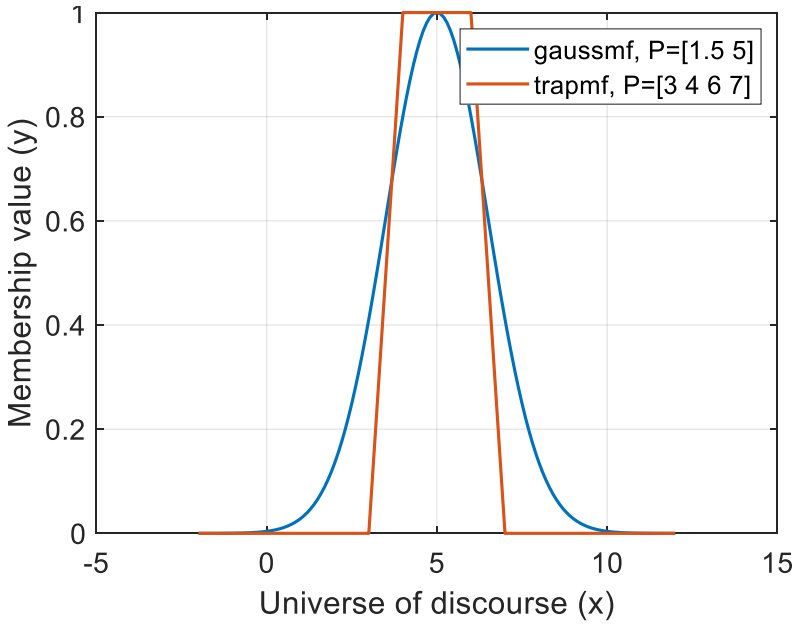


Рисунок 2.14 – Графіки обчислення функцій приналежності гаусовської та трапецеїдальної

Розглянемо приклади виконання операцій над нечіткими множинами MatLab. Нехай дано базова множина X = [-5,20]:

```
N = 501;
minX = -5;
maxX = 20;
x = linspace(minX,maxX,N);
mf1 = gaussmf(x,[1 5]);
mf2 = gaussmf(x,[1 7]);
% обчислення max та побудова графіку обчислення
mf = max(mf1,mf2);
figure(1)
plot(x,mf,'LineWidth',3)
% обчислення min та побудова графіку обчислення
figure(2)
mf3 = min(mf1,mf2);
plot(x,mf3,'LineWidth',3)
% обчислення об’єднання та побудова графіку обчислення
figure(3)
mf4 = probor([mf1;mf2])% об’єднання
plot(x,mf4,'LineWidth',3)
% обчислення перетин та побудова графіку обчислення
figure(4)
mf4 = prod([mf1;mf2])
plot(x,mf4,'LineWidth',3)% перетин
```

На рис. 2.15 та 2.16 показано виконання операцій об'єднання та перетину при використанні відповідно операторів *max* та *min*.

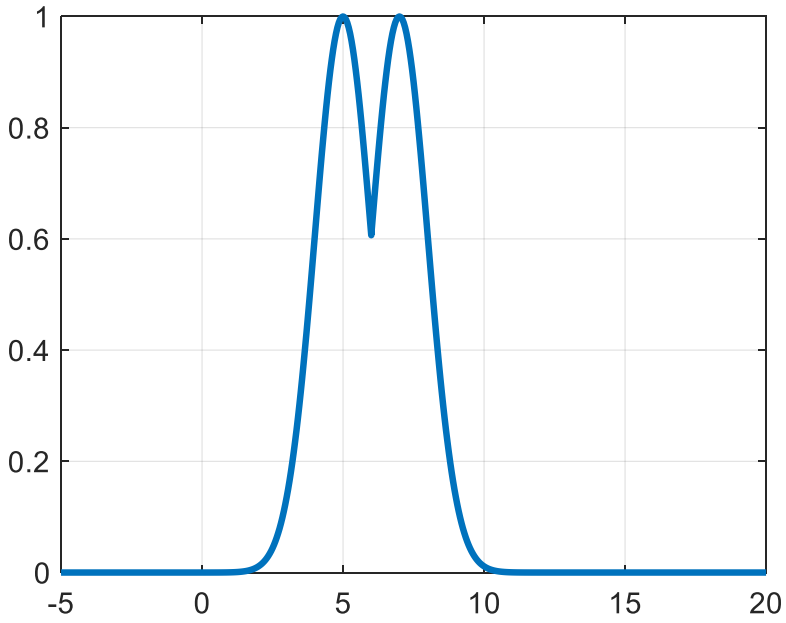


Рисунок 2.15 – Графіки об'єднання нечітких множин (операція max)

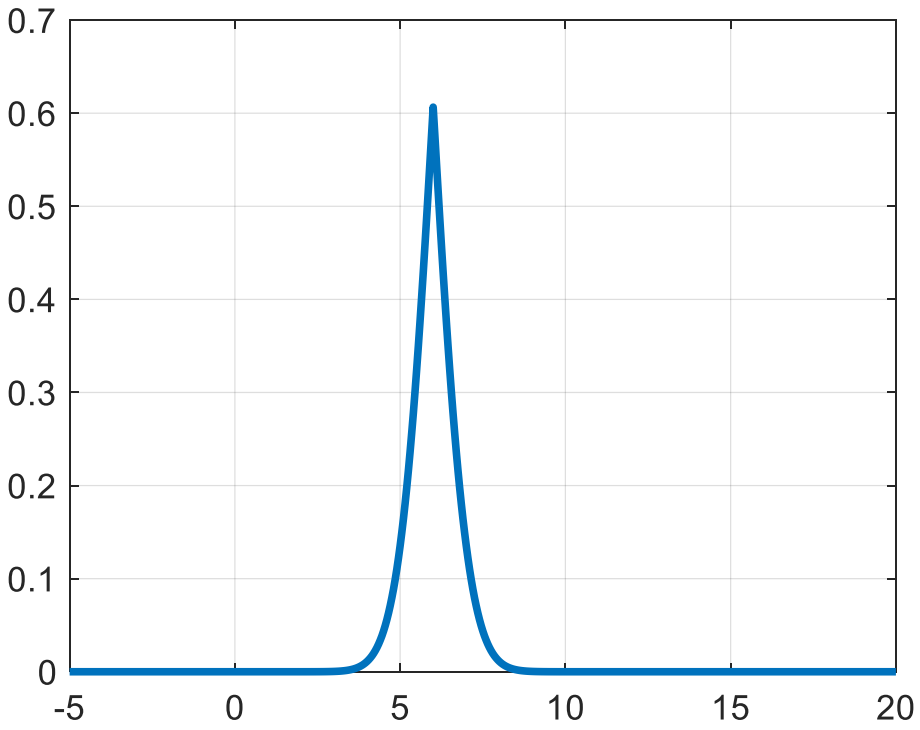


Рисунок 2.16 – Графіки перетину нечітких множин (операція min)

На рис. 2.17 та 2.18 показано виконання операцій об'єднання та перетину при використанні відповідно операторів обмеженої суми та додатку.

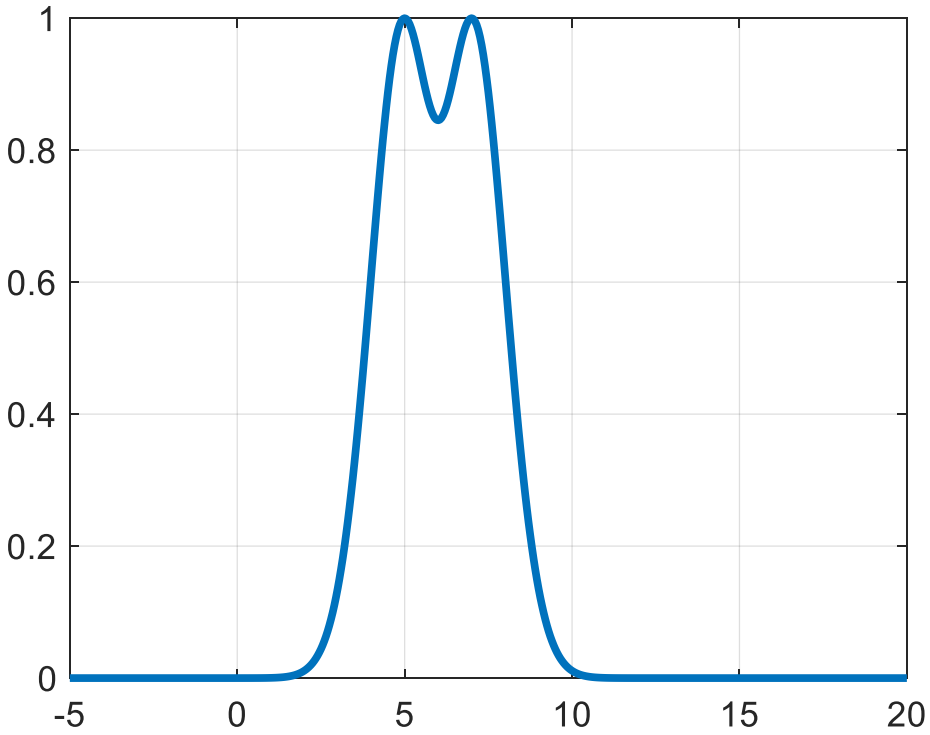


Рисунок 2.17 – Графіки об'єднання нечітких множин (операція обмежена *сума)*

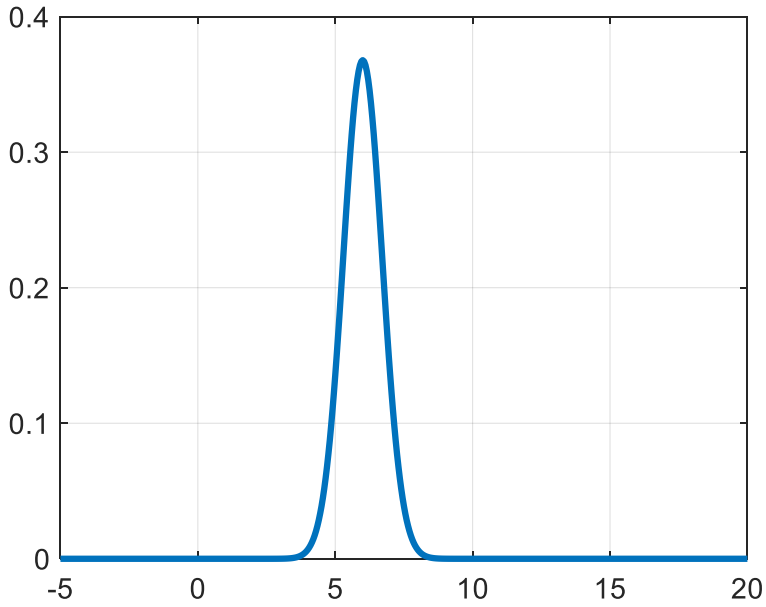


Рисунок 2.18 – Графіки перетину нечітких множин (операція добуток)

Для виконання арифметичних операцій складання, віднімання, множення та поділу над нечіткими числами нечіткої множини у складі MatLab є калькулятор fuzarith. Для його використання необхідно задати рядок виду

c = fuzarith(x, a, b, operator),

де *x* – універсальна множина, на якій задані нечіткі числа; *a* – вектор ступенів приналежності елементів універсальної множини першої нечіткої множини; *b* – вектор ступенів приналежності елементів універсальної множини другої нечіткої множини; *operator* - одна з допустимих арифметичних операцій: *'sum'* - додавання; *'sub'* - віднімання; *'prod'* – множення; *'div'* – ділення .

Вихідною змінною функцією *fuzarith* є вектор ступенів приналежності елементів універсальної множини *x* результату виконання нечіткої арифметичної операції. Розмірності векторів *x*, *a*, *b* та *c* повинні бути однаковими.

Приклад виконання арифметичних операцій складання, віднімання, множення та ділення над нечіткими числами нечіткої множини у MatLab:

```
N = 501;
minX = -20;
maxX = 20;
x = linspace(minX,maxX,N);
A = trapmf(x,[-10 -2 1 3]);
B = gaussmf(x,[2 5]);
```

```
%Evaluate the sum, difference, product, and quotient of A and B.
Csum = fuzarith(x,A,B,'sum');
Csub = fuzarith(x,A,B,'sub');
Cprod = fuzarith(x,A,B,'prod');
Cdiv = fuzarith(x,A,B,'div');
%Plot the addition and subtraction results.
figure(5)
subplot(2,1,1)
plot(x,A,'--',x,B,':',x,Csum,'c')
title('Fuzzy Addition, A+B')
legend('A','B','A+B')
subplot(2,1,2)
plot(x,A,'--',x,B,':',x,Csub,'c')
title('Fuzzy Subtraction, A-B')
legend('A','B','A-B')
figure(6)
subplot(2,1,1)
plot(x,A,'--',x,B,':',x,Cprod,'c')
title('Fuzzy Multiplication, A*B')
legend('A','B','A*B')
subplot(2,1,2)
plot(x,A,'--',x,B,':',x,Cdiv,'c')
title('Fuzzy Division, A/B')
legend('A','B','A/B')
```

На рис. 2.19 та показаний результат арифметичного обчислення нечітких множин

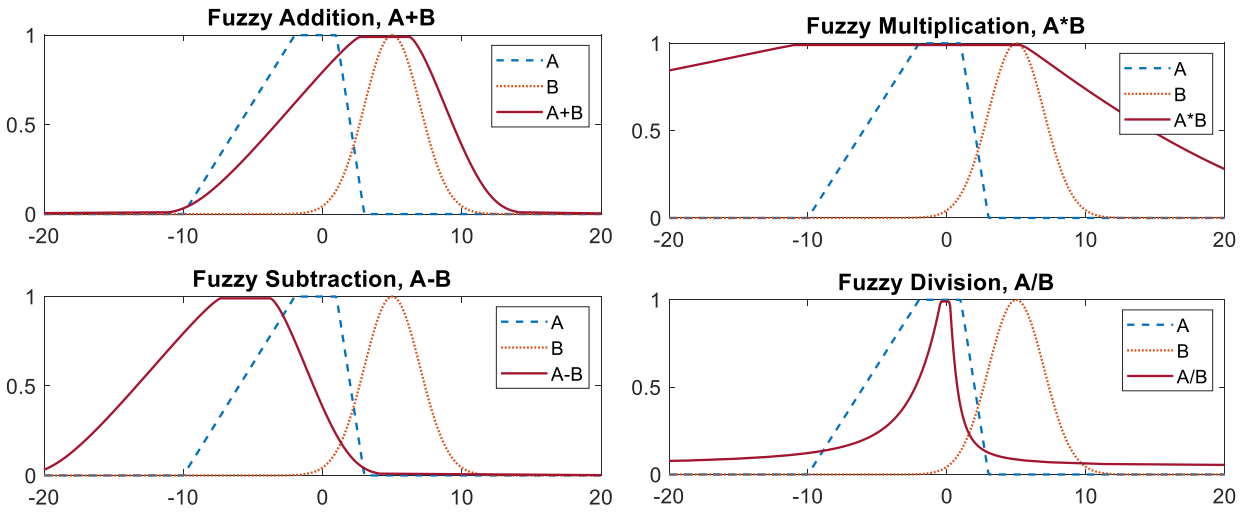


Рисунок 2.19 – Графіки результатів арифметичного обчислення нечітких множин

Нечіткі арифметичні операції виконуються за таким алгоритмом:

– перетворення нечітких чисел-операндів в α-рівневі нечіткі множини;

– виконання арифметичної операції для кожного α-рівня відповідно до принципу розширення;

– перетворення результуючого нечіткого числа з α-рівневого уявлення до традиційного виду.

Кількість α-рівнів може вибиратися користувачем.

2.2 Зміст і захист звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 2.2.

За вказаними даними у табл. 2.2 необхідно виконати:

1. операцій max, min, об’єднання, перетин над нечіткими множинами у MatLab

2. арифметичні операцій складання, віднімання, множення та ділення над нечіткими числами нечіткої множини у MatLab

Звіт по роботі повинен містити завдання, програму обчислення арифметичних операції над функціями приналежності, графіки результатів обчислень

Таблиця 2.2 - Варіанти індивідуальних завдань

№ вар.	Діапазон базової множини *X*	Функція приналежності 1	Функція приналежності 1
1	[0,20]	*‘trimf’*	*'gaussmf'*
2	[-20,20]	*'trapmf'*	*'gauss2mf'*
3	[-5,20]	*'gaussmf'*	*'dsigmf'*
4	[20,40]	*'gauss2mf'*	*'psigmf'*
5	[10,50]	*'sigmf'*	*‘trimf’*
6	[-10,20]	*'dsigmf'*	*'trapmf'*
7	[-15,15]	*'psigmf'*	*'gaussmf'*
8	[-10,30]	*'gbellmf'*	*'gauss2mf'*
9	[0,20]	*'smf'*	*'sigmf'*
10	[-20,20]	*'zmf'*	*'dsigmf'*
11	[-5,20]	'pimf'	*'psigmf'*
12	[20,40]	*‘trimf’*	*'gbellmf'*
13	[10,50]	*'trapmf'*	*'smf'*
14	[-10,20]	*'gaussmf'*	*'zmf'*
15	[-15,15]	*'gauss2mf'*	'pimf'
16	[-10,30]	*'sigmf'*	*‘trimf’*
17	[-5,20]	*'dsigmf'*	*'smf'*
18	[20,40]	*'psigmf'*	*'zmf'*
19	[10,50]	*'gbellmf'*	'pimf'
20	[-10,20]	*'smf'*	*‘trimf’*

2.3 Контрольні питання

1. Опишіть основні варіанти операції об'єднання нечітких множин.
2. Опишіть основні варіанти операції перетину нечітких множин.
3. Опишіть основні варіанти операції доповнення нечітких множин.
4. Опишіть операції різниці та симетричної різниці нечітких множин.
5. Опишіть операції твору алгебри та алгебраїчної суми нечітких множин.
6. Опишіть операції розтягування та стиснення нечітких множин.

3 АПРОКСИМАЦІЯ ЗАЛЕЖНОСТІ З ВИКОРИСТАННЯ СИСТЕМИ НЕЧІТКОГО ВИСНОВКУ

Мета роботи: отримати у редакторі нечіткого висновку апроксимаційну криву яка відповідає графіку завданої вихідної функції

3.1 Методика виконання роботи

Допустимо, потрібно підібрати такі параметри системи нечіткого висновку, при яких виходить апроксимація заданої кривої (див. рис. 3.1).

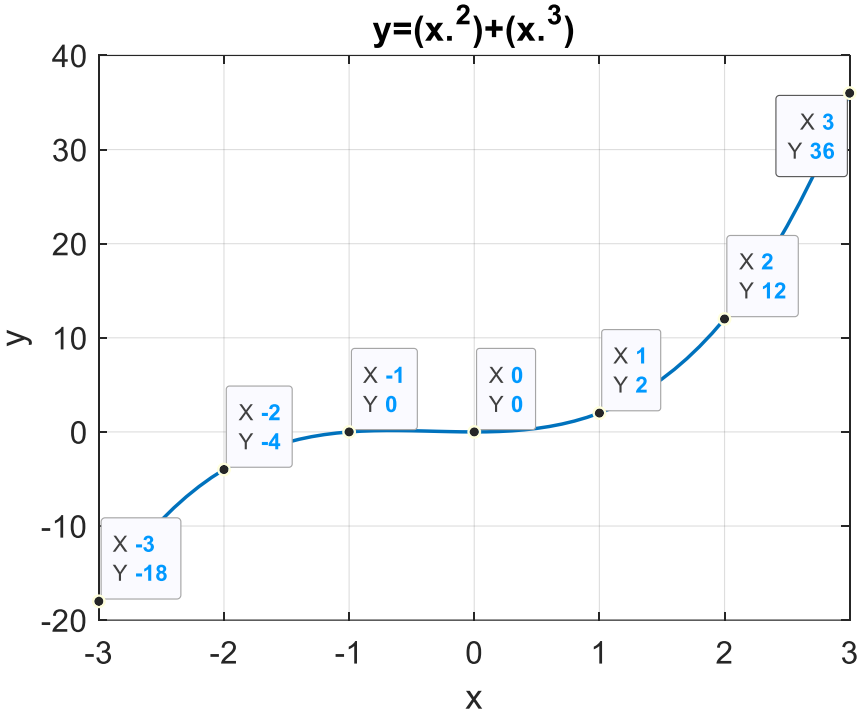


Рисунок 3.1 – Графік функції $y = x^2 + x^3$

Для виконання апроксимації потрібно вибрати набір точок (див. табл. 3.1).

Таблиця 3.1 – Набір точок для апроксимації функції

№ з/п	1	2	3	4	5	6	7
x	-3	-2	-1	0	1	2	3
y	-17	-4	0	0	3	12	36

1 варіант вирішення поставленого завдання шляхом програмної реалізації.

Програмна реалізація для апроксимації функції з використанням нечіткої системи:

```
% Побудова графіку залежності, що завдана
xx = -3:0.1:3;
```

```
yy=(xx.^2)+(xx.^3);
figure (1)
plot(xx,yy); grid;
title('y=(x.^2)+(x.^3)');
xlabel('x');
ylabel('y');
fis = mamfis('Name','Graffuzzy');
% Додається та налаштується вхідна змінна.
fis.Inputs = fisvar;
fis.Inputs.Name = "x";
fis.Inputs.Range = [-3 3];
% Задаються функції приналежності вхідної змінної.
% Для кожного MF створюється об'єкт fismf і встановлюються
властивості
% за допомогою покрокового опису.
% Виберемо центри термів X, що описуються гаусовими функціями,
відповідно до табл. 3.1.
fis.Inputs.MembershipFunctions = fismf;
fis.Inputs.MembershipFunctions.Name = "x1";
fis.Inputs.MembershipFunctions.Type = "gaussmf";
fis.Inputs.MembershipFunctions.Parameters = [0.4247 -3];
fis.Inputs.MembershipFunctions(2) = fismf;
fis.Inputs.MembershipFunctions(2).Name = "x2";
fis.Inputs.MembershipFunctions(2).Type = "gaussmf";
fis.Inputs.MembershipFunctions(2).Parameters = [0.4247 -2];
fis.Inputs.MembershipFunctions(3) = fismf;
fis.Inputs.MembershipFunctions(3).Name = "x3";
fis.Inputs.MembershipFunctions(3).Type = "gaussmf";
fis.Inputs.MembershipFunctions(3).Parameters = [0.4247 -1];
fis.Inputs.MembershipFunctions(4) = fismf;
fis.Inputs.MembershipFunctions(4).Name = "x4";
fis.Inputs.MembershipFunctions(4).Type = "gaussmf";
fis.Inputs.MembershipFunctions(4).Parameters = [0.4247 0];
fis.Inputs.MembershipFunctions(5) = fismf;
fis.Inputs.MembershipFunctions(5).Name = "x5";
fis.Inputs.MembershipFunctions(5).Type = "gaussmf";
fis.Inputs.MembershipFunctions(5).Parameters = [0.4247 1];
fis.Inputs.MembershipFunctions(6) = fismf;
fis.Inputs.MembershipFunctions(6).Name = "x6";
fis.Inputs.MembershipFunctions(6).Type = "gaussmf";
fis.Inputs.MembershipFunctions(6).Parameters = [0.4247 2];
fis.Inputs.MembershipFunctions(7) = fismf;
fis.Inputs.MembershipFunctions(7).Name = "x7";
fis.Inputs.MembershipFunctions(7).Type = "gaussmf";
fis.Inputs.MembershipFunctions(7).Parameters = [0.4247 3];
```

```
% Додається та налаштовується вихідна змінна
% та її функція приналежності.
fis.Outputs = fisvar([-17 36],'Name',"y");
% Вказується вихідні функції приналежності
% за допомогою вектора об’єктів fismf.
% Виберемо центри термів У, що описуються гаусовими функціями,
відповідно до табл. 3.1
mf1 = fismf("gaussmf",[3.751 -17],'Name',"y1");
mf2 = fismf("gaussmf",[3.751 -4],'Name',"y2");
mf3 = fismf("gaussmf",[3.751 0],'Name',"y3");
mf4 = fismf("gaussmf",[3.751 0],'Name',"y4");
mf5 = fismf("gaussmf",[3.751 3],'Name',"y5");
mf6 = fismf("gaussmf",[3.751 12],'Name',"y6");
mf7 = fismf("gaussmf",[3.751 36],'Name',"y7");
fis.Outputs.MembershipFunctions = [mf1 mf2 mf3 mf4 mf5 mf6 mf7];
% Створюються правила для нечіткої системи.
rule1 = fisrule([1 1 1 1],1);
rule2 = fisrule([2 2 1 1],1);
rule3 = fisrule([3 3 1 1],1);
rule4 = fisrule([4 4 1 1],1);
rule5 = fisrule([5 5 1 1],1);
rule6 = fisrule([6 6 1 1],1);
rule7 = fisrule([7 7 1 1],1);
rules = [rule1 rule2 rule3 rule4 rule5 rule6 rule7];
rules = update(rules,fis);
fis.Rules = rules;
% Перевірка введених правил
showrule(fis,[1 2 3 4 5 6 7],'symbolic')
% Графіки опису термів
figure(2)
plotmf(fis,'input',1); grid
figure(3)
plotmf(fis,'output',1);grid
% Перевірка правил
showrule(fis,[1 2 3 4 5 6 7],'symbolic')
% Побудова графіку апроксимації ф-ції
figure(4);
gensurf(fis); grid
% Графічний опис властивості описаної нечіткої системи
figure(5)
plotfis(fis);
```

Графічне уявлення опису термів вхідної змінної наведено на рисунку 3.2, а вихідної змінної на рису. 3.3. Терми описано гаусовським функціями приналежності, центри термів співпадають з значеннями

наведеними у таблиці 3.1.

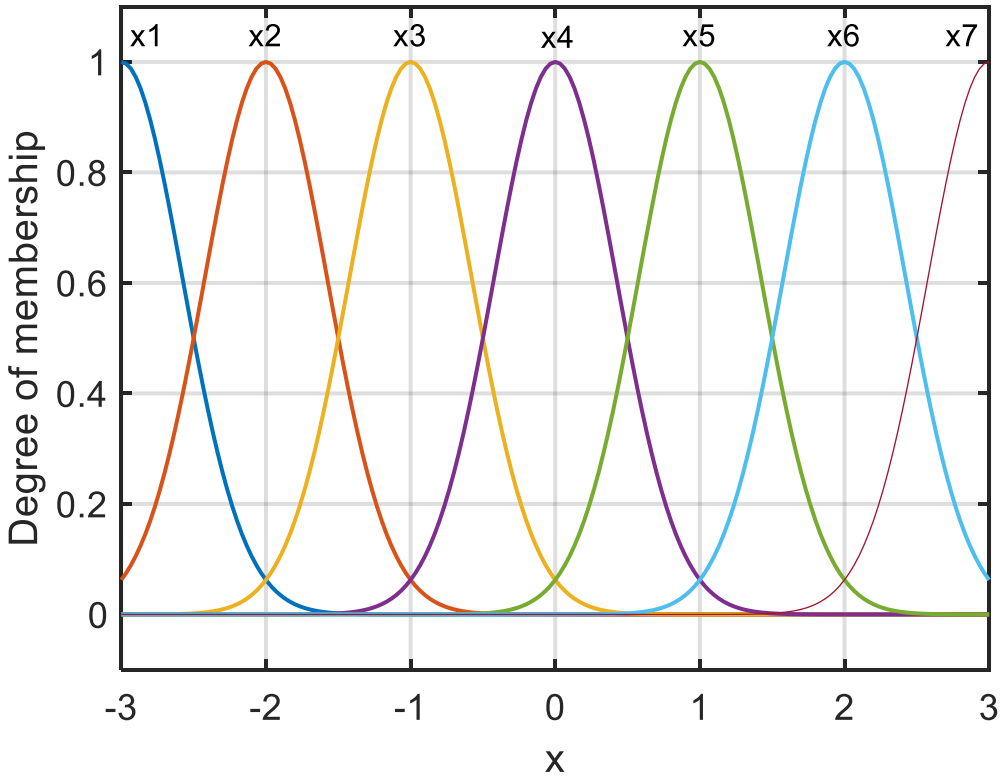


Рисунок 3.2 - Графічне уявлення опису термів вхідної змінної

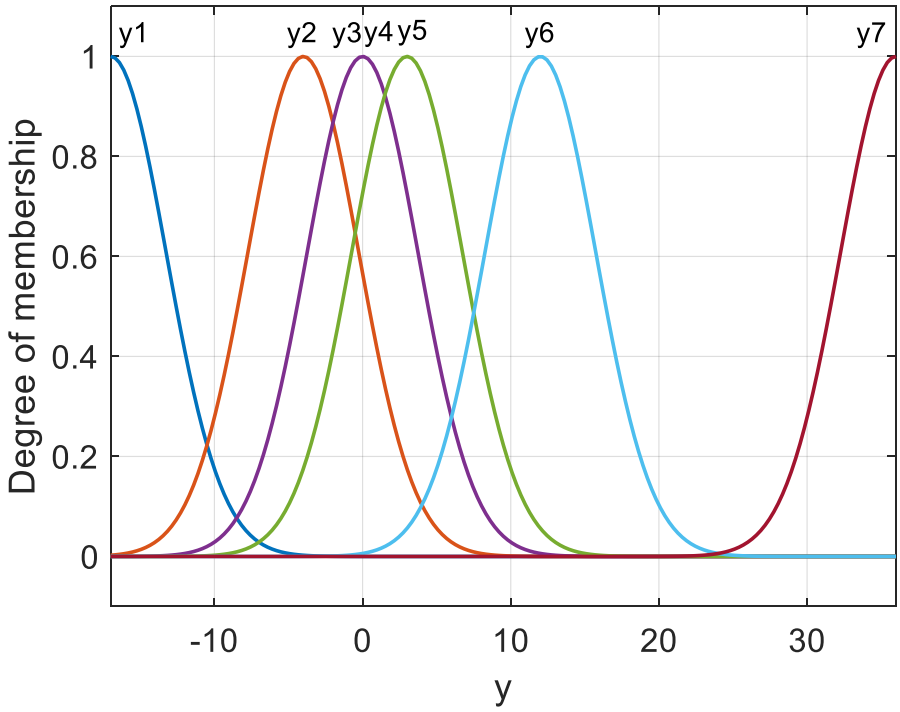


Рисунок 3.3 - Графічне уявлення опису термів вихідної змінної

На рис. 3.4 показаний результат апроксимації вихідної функції за допомогою системи нечітких правил.

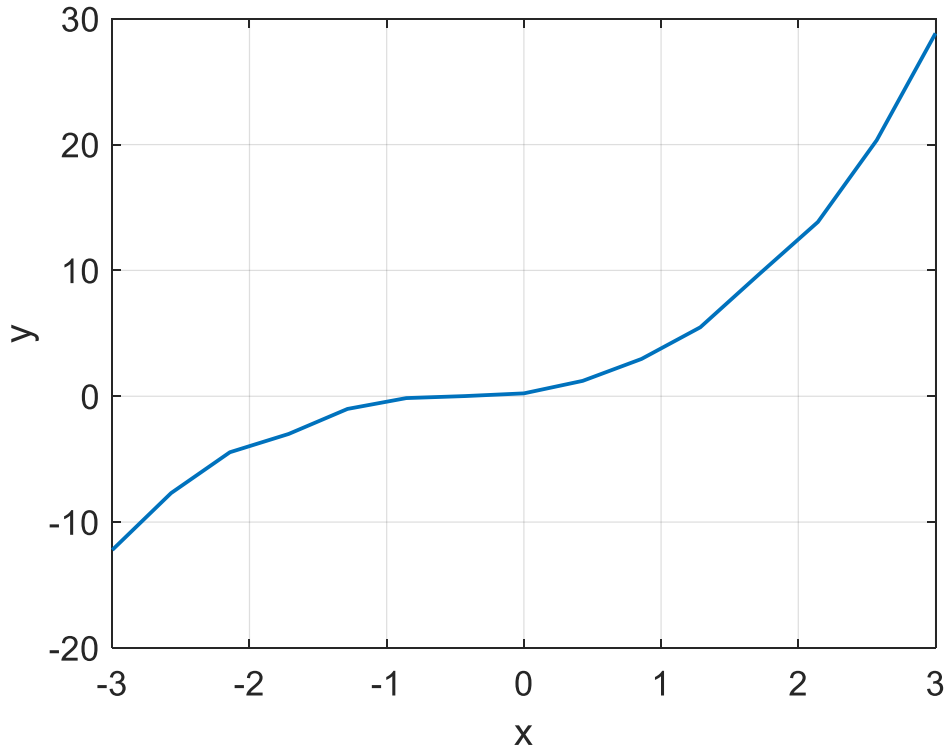


Рисунок 3.4 – Результат апроксимації вихідної функції за допомогою системи нечітких правил

На рисунку 3.5 зображено графічний опис властивості описаної нечіткої системи.

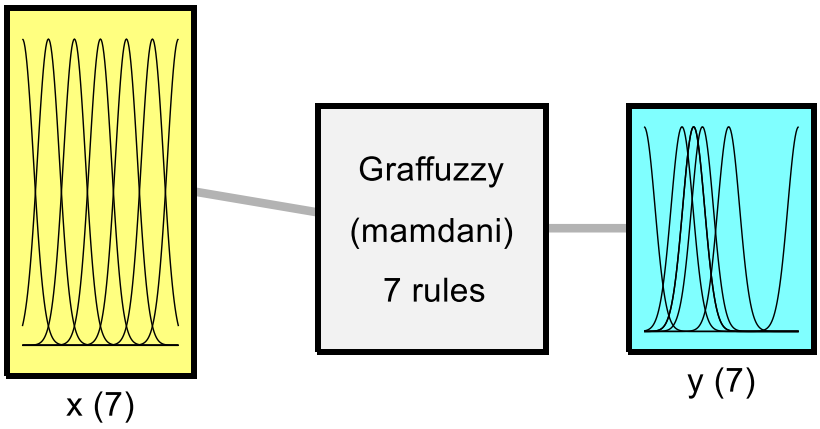


Рисунок 3.5 – Графічний опис властивості описаної нечіткої системи

2 варіант вирішення завдання з використанням редактора системи нечіткого висновку.

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

>> fuzzy

Дале проводиться налаштовування головне вікно редактора нечіткої логічної системи (див. рис. 3.6.).

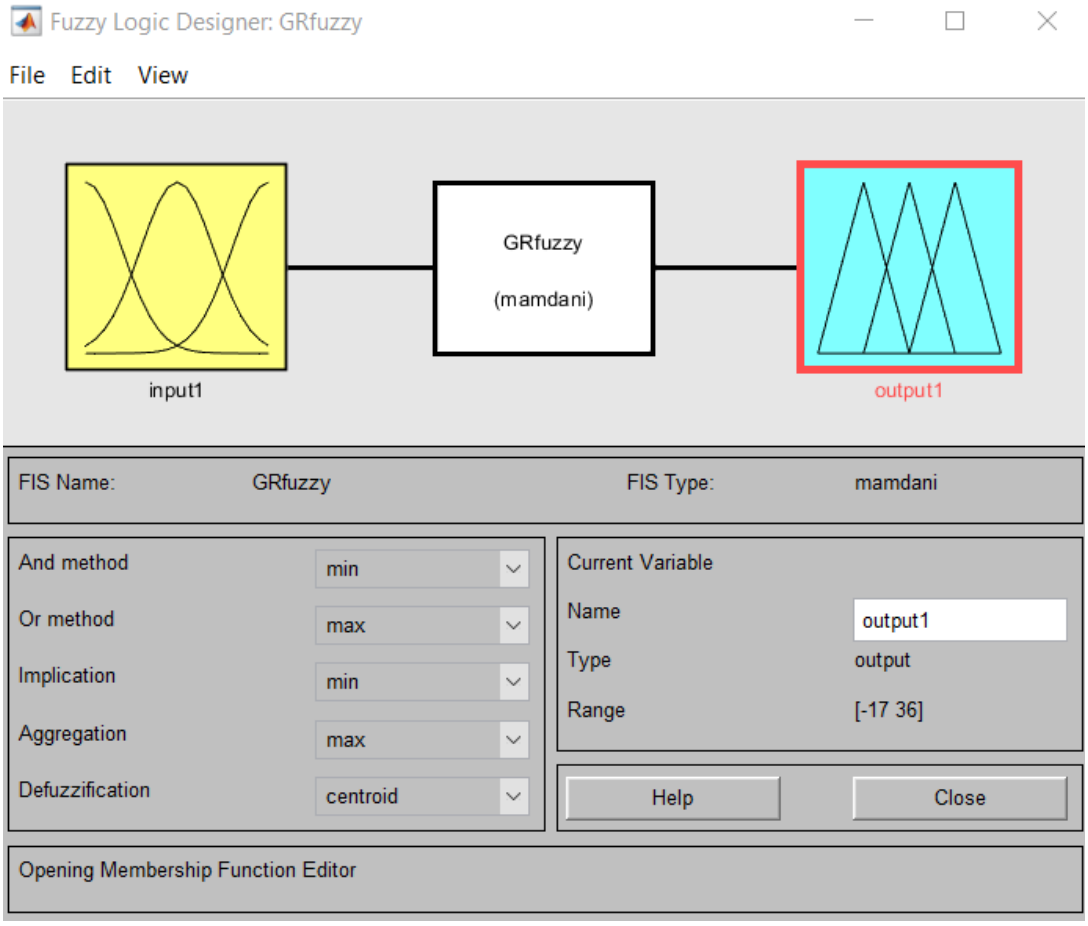


Рисунок 3.6 - Інтерфейс FIS editor

У *FIS editor* задається опис систему нечіткого логічного висновку *Mamdani*. Для створюваної системи обирається вид логічного зв'язоку (*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 7 вхідних з розміром базової шкали (*Range*= [-3 3]) та 7 вихідних змінних з розміром базової шкали (*Range*= [-17 36]).

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

Name = "x1"; Type = "gaussmf"; Param = [0.4247 -3];
Name = "x2"; Type = "gaussmf"; Param = [0.4247 -2];
Name = "x3"; Type = "gaussmf"; Param = [0.4247 -1];
Name = "x4"; Type = "gaussmf"; Param = [0.4247 0];
Name = "x5"; Type = "gaussmf"; Param = [0.4247 1];
Name = "x6"; Type = "gaussmf"; Param = [0.4247 2];
Name = "x7"; Type = "gaussmf"; Param = [0.4247 3];

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

вихідної змінної *y* гаусовську функцію приналежності. З параметрами:
Name = "y1"; Type = "gaussmf"; Param = [3.751 -17];
Name = "y2"; Type = "gaussmf"; Param = [3.751 -4];
Name = "y3"; Type = "gaussmf"; Param = [3.751 0];
Name = "y4"; Type = "gaussmf"; Param = [3.751 0];
Name = "y5"; Type = "gaussmf"; Param = [3.751 3];
Name = "y6"; Type = "gaussmf"; Param = [3.751 12];
Name = "y7"; Type = "gaussmf"; Param = [3.751 36];

Примітка. У якості центрів термів, що описуються гаусовими функціями, обрані значення відповідно до табл. 3.1.

Приклад налаштування головного вікна редактора *Membership Function Editor* показано на рис. 3.7.

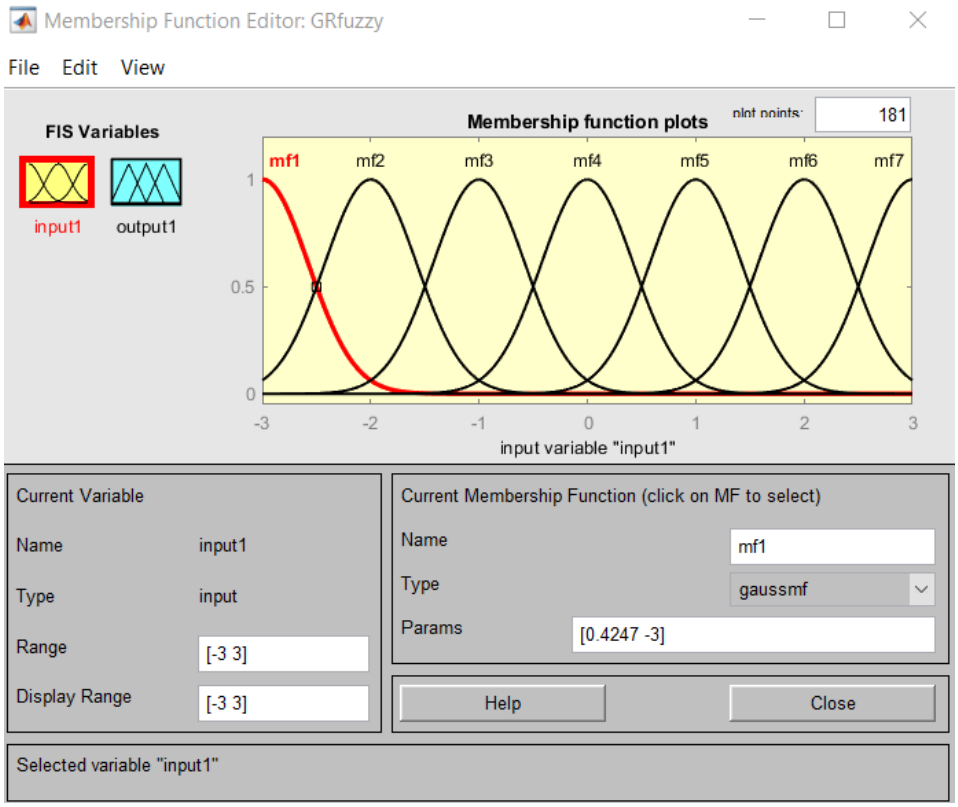


Рисунок 3.7 - Приклад налаштування головного вікна редактора Membership Function Editor

Після опису вхідні та вихідні лінгвістичні змінні, здійснюється опис правил у вікні редактора Rule Editor. Налаштовування правил для даного випадку наведено на рисунку (рис. 3.8).

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

Після опису правил можливо за допамогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого логічного висновку за різних вхідних даних (див. рис. 3.9);

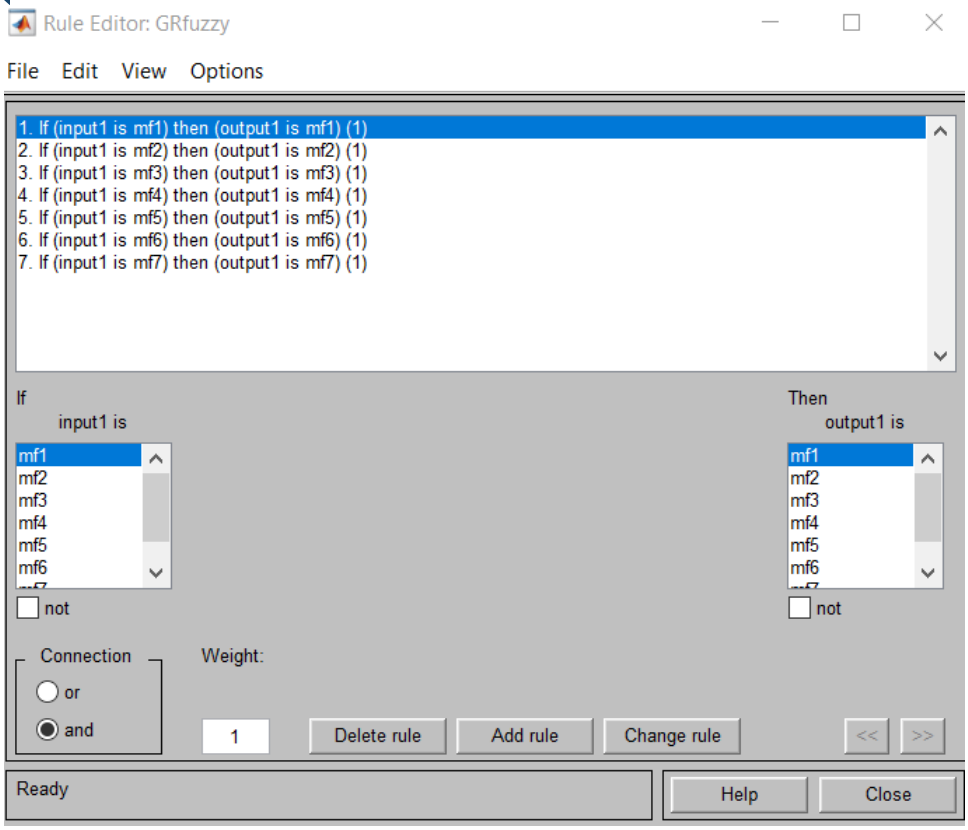


Рисунок 3.8 – Інтерфейс ruleedit

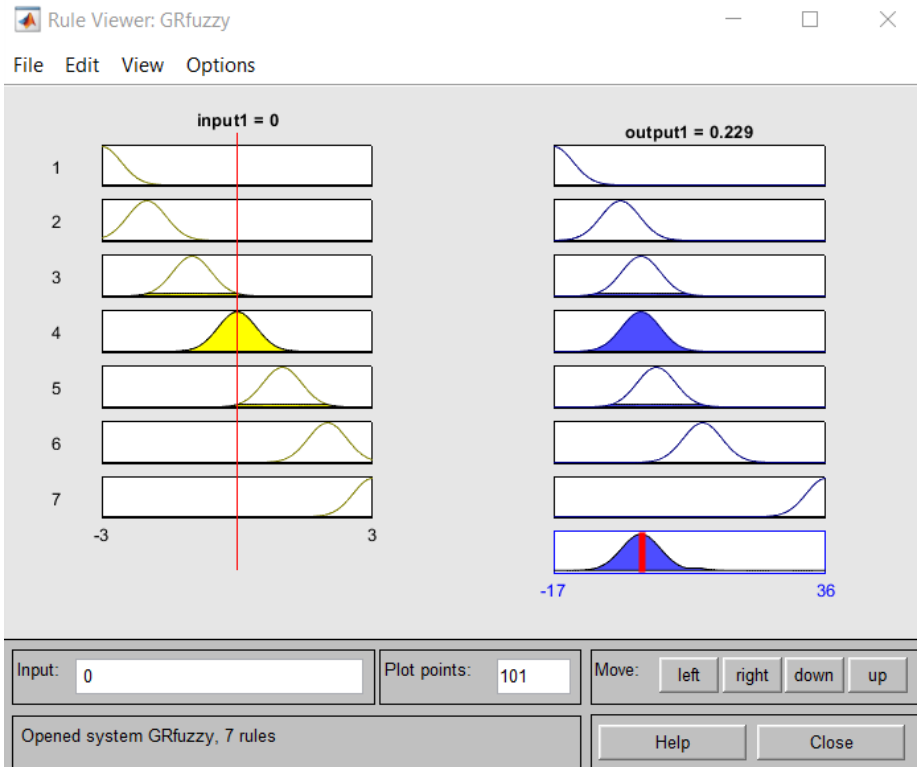


Рисунок 3.9 - Робота системи нечіткого логічного висновку

– інтерфейсу *Surface Viewer* можливо переглянути керуючу поверхню нечіткої логічної системи, яка виходить при подачі на вхід

системи різних допустимих значень (рис. 3.10).

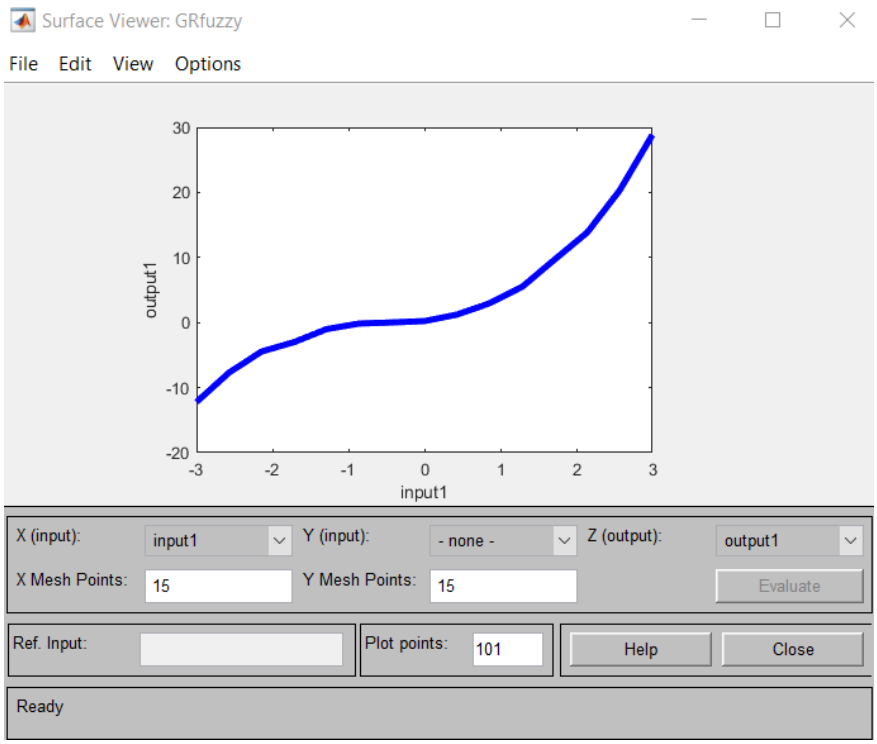


Рисунок 3.10 – Апроксимаційна крива побудована системою нечіткого логічного висновку

Отримана крива не цілком відповідає графіку вихідної функції $y=x^2 + x^3$, проте якість апроксимації можна покращити, якщо змінити форму термів, що описують входи та виходи нечіткої системи, або додати нові правила.

3.2 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 3.2.
За вказаними у табл. 3.2 потрібно підібрати такі параметри системи нечіткого висновку, при яких виходить апроксимація заданої кривої:

Таблиця 3.2 - Варіанти індивідуальних завдань

№ вар.	Завдана функція	Функція приналежності
1	$y = 3x^2 + x^3 + 2x$	*'gaussmf'*
2	$y = 2x^2 + 2x^3 + 3$	*'trapmf'*
3	$y = x^2 + 2x^3 + 4$	*'gauss2mf'*
4	$y = 2x^2 + x^3 + 3x$	*'sigmf'*
5	$y = x^2 + x^3 + 3x + 2$	*'smf'*
6	$y = 3x^2 + 2x^3 + x + 1$	*'zmf'*

Продовження табл. 3.2

№ вар.	Завдана функція	Функція приналежності
7	$y = 5x^2 + x^3 + 2x$	*'zmf'*
8	$y = 2x^2 + 2x^3$	*'sigmf'*
9	$y = 3x^2 + x^3 + 2x$	*'smf'*
10	$y = 2x^2 + 2x^3 + 3$	*'trapmf'*
11	$y = x^2 + 2x^3 + 4$	*'gauss2mf'*
12	$y = 2x^2 + x^3 + 3x$	*'gaussmf'*
13	$y = x^2 + x^3 + 3x + 2$	*'gaussmf'*
14	$y = 3x^2 + 2x^3 + x + 1$	*'gauss2mf'*
15	$y = 5x^2 + x^3 + 2x$	*'sigmf'*
16	$y = 2x^2 + 2x^3$	*'smf'*
17	$y = x^2 + 2x^3 + 4$	*'zmf'*
18	$y = 2x^2 + x^3 + 3x$	*'gauss2mf'*
19	$y = x^2 + x^3 + 3x + 2$	*'gaussmf'*
20	$y = 3x^2 + 2x^3 + x + 1$	*'smf'*

3.3 Контрольні питання

1. Опишіть схему нечіткого виводу Mamdani.
2. Чим відрізняється схема нечіткого виведення Larsen від схеми Mamdani?
3. Чим відрізняється схема нечіткого виведення Tsukamoto від схеми Mamdani?
4. Опишіть схему нечіткого виводу Sugeno.
5. Які варіанти існують для операції агрегування у нечіткій логічній системі?
6. Нечітка продукційна система є універсальним апроксиматором – що це означає?
7. Які умови повинні виконуватися для нечіткої системи як універсального апроксиматора?
8. Що таке дефаззифікація?
9. Опишіть основні варіанти виконання операції дефазифікації?

4 СИНТЕЗ НЕЛІНІЙНОГО НЕЧІТКОГО РЕГУЛЯТОРА П-ТИПУ

Мета роботи: засвоєння методики синтезу нелінійного нечіткого регулятору П-типу з використанням редактора системи нечіткого висновку

4.1 Теоретичні основи синтезу нечіткого регулятору П-типу

Сформулюємо умови лінійної поведінки нечіткого логічного регулятору П-типу (НЛР_П):

1) використовуються трикутні функції приналежності;

2) терми утворюють нечітке розбиття відповідних базових множин;

3) для дефазифікації використовується дискретний метод центру тяжкості.

При цих припущеннях кожне значення e має ненульове значення приналежності до двох сусідніх терм T_i та T_{i+1} з центрами e_i та e_{i+1} (див. рис. 4.1).

Значення приналежності для точки $e_i < e < e_{i+1}$ розраховуються за формулами:

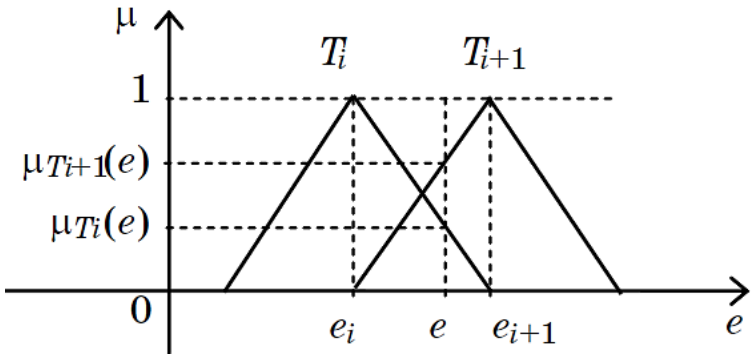


Рисунок 4.1 – Перехідні процеси при різних коефіцієнтах підсилення

$$\mu_{T_i}(e) = \frac{e_{i+1} - e}{e_{i+1} - e_i}, \mu_{T_{i+1}}(e) = \frac{e - e}{e_{i+1} - e_i}.$$

У такій ситуації спрацьовують два правила:

R_1: Якщо $e = T_i(e)$, то $u = T_i(u)$,
R_2: Якщо $e = T_{i+1}(e)$, $u = T_{i+1}(u)$.

При використанні дискретного методу центру тяжіння має значення тільки u_i − центральна точка терму $T_i(u)$ і результат дефазифікації описується формулою:

$$u = \frac{\mu_{T_i}(e)u_i + \mu_{T_{i+1}}(e)u_i}{\mu_{T_i}(e) + \mu_{T_{i+1}}(e)} = \mu_{T_i}(e)u_i + \mu_{T_{i+1}}(e)u_{i+1=}$$
$$= \frac{e_{i+1} - e}{e_{i+1} - e_i}u_i + \frac{e - e}{e_{i+1} - e_i}u_{i+1}.$$

Цей вираз можна переписати у вигляді

$$u = \frac{u_{i+1} - u_i}{e_{i+1} - e_i}e + \frac{e_{i+1}u_i - e_iu_{i+1}}{e_{i+1} - e_i} = Ke + C, \qquad (4.1)$$

де *K* – коефіцієнт підсилення; *C* - зміщення

Зауважимо, що (4.1) стосується лише постійного діапазону між e_i та e_{i+1}. Якщо відстань між термами змінюється, змінюватимуться і константи *K* і *C*.

Формула (4.1) показує, що для еквівалентності поведінки НЛР_П і звичайного П-регулятора необхідно виконання двох умов:

1) значення *K* має бути однаковим у всьому діапазоні зміни помилки *e*:

$$\frac{u_{i+1} - u_i}{e_{i+1} - e_i} = K, \qquad \forall i \in \{1, N-1\} \qquad (4.2)$$

2) значення зсуву *C* повинно бути нульовим, що виконується при:

$$\frac{e_{i+1}}{e_{i+1}} = \frac{u_{i+1}}{u_i}, \quad \forall i \in \{1, N-1\} \qquad (4.3)$$

де N – кількість термів.

Таким чином, НЛР _П, для якого справедливо (4.2) та (4.3), описується формулою

$$u = Ke. \qquad (4.4)$$

Оскільки для лінійного НЛР_П кількість термів не має значення, розглянемо регулятор із трьома правилами.

У разі використання дискретного методу центру тяжкості вихід регулятора можна описати з допомогою рис. 11.56.

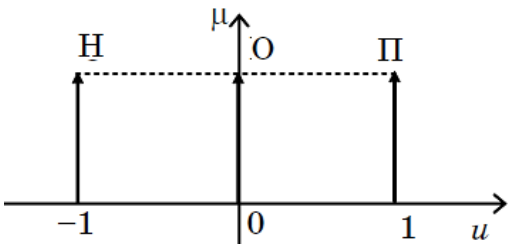


Рисунок 4.2 – Опис виходу регулятора з трьома правилами

Відповідно до рис. 4.2 виконується умова:

$$u_{i+1} - u_i = 1.$$

Нехай базову шкалу вхідної змінної описано відповідно до рис. 4.3.

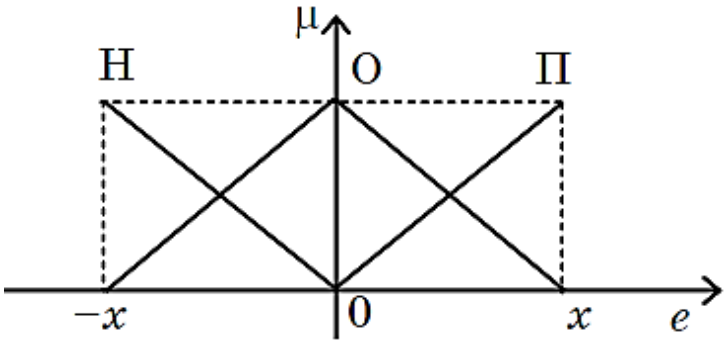


Рисунок 4.3 – Опис входу регулятора з трьома правилами

Тоді для вхідної змінної відстань між центрами термів:

$$e_{i+1} - e_i = x, \quad x \in [0,1]$$

Тоді

$$K = \frac{\Delta u}{\Delta e} = \frac{1}{x}. \qquad (4.5)$$

і що менше x, то більше вписувалося коефіцієнт посилення.
Закон управління можна описати з допомогою табл. 4.1.

Таблиця 4.1 - Закон управління НЛР_П із трьома правилами

	Н	0	П
e	-*x*	0	*x*
u	-1	0	1

Вибір параметра x визначає нахил кривої, що описує закон керування (див. рис. 4.4). Чим менше значення x, тим ближче виявляється закон керування до релейного (при x → 0 регулятор працює за знаком помилки).

Коефіцієнт посилення повинен бути великим наприкінці перехідного процесу – для зменшення статичної помилки. На початку перехідного процесу потрібно мати менший коефіцієнт посилення, щоб уникнути надмірного перерегулювання.

Формула (4.5) дозволяє легко налаштовувати значення змінного коефіцієнта посилення для різних ділянок базової шкали вхідної змінної.

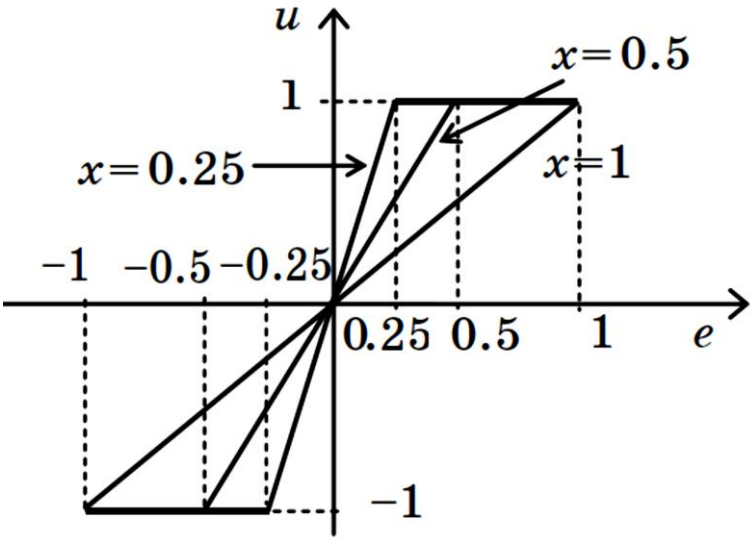


Рисунок 4.4 – Варіанти закону управління для НЛР_П з трьома правилами

Наприклад, розглянемо НЛР_П із п'ятьма термами, де дві пари термів розташовані симетрично щодо нуля (див. рис. 4.5).

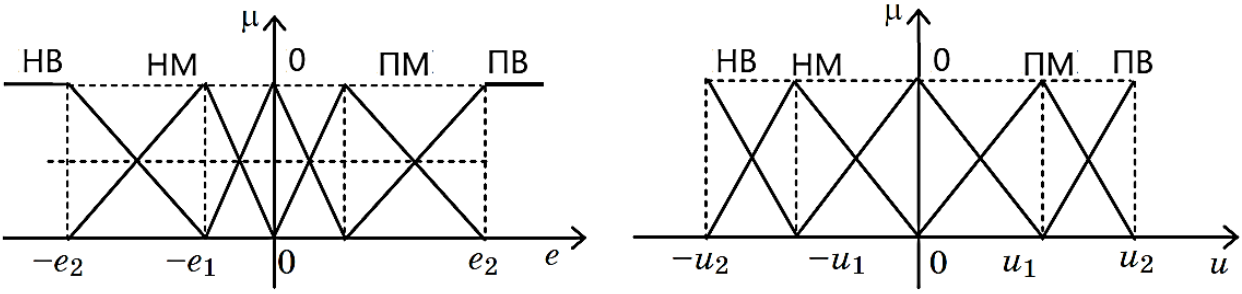


Рисунок 4.5 – Варіант опису термів НЛР_П з п’ятьма правилами

На підставі (4.5), закон управління визначає табл. 4.2.

Таблиця 4.2 – Типовий закон управління НЛР_П із п'ятьма правилами

	НВ	НМ	0	ПМ	ПВ
e	$-e_2$	$-e_1$	0	e_1	e_2
u	$-u_2$	$-u_1$	0	u_1	u_2

На ділянці між центрами термів 0 і НМ, а також 0 і ПМ коефіцієнт посилення

$$K_1 = \frac{u_1}{e_1} = -\frac{u_1}{e_1}.$$

На ділянках між центрами термів ПРО та ОМ, а також ПМ та ПБ

коефіцієнт посилення

$$K_2 = \frac{u_2 - u_1}{e_2 - e_1} = \frac{-u_2 + u_1}{-e_2 + e_1}.$$

Наприклад, нехай вибрано значення центрів термів, показані в табл. 4.3.

Тоді

$$K_1 = \frac{0{,}3}{0{,}5} = 0{,}6; \quad K_2 = \frac{1 - 0{,}3}{1 - 0{,}5} = 1{,}4.$$

Таблиця 4.3 - Варіант закону управління НЛР_П із п'ятьма правилами

	НВ	НМ	0	ПМ	ПВ
e	-1	-0.5	0	0.5	1
u	-1	-0.3	0	0.3	1

Графічне уявлення закону управління показано на рис. 4.6.

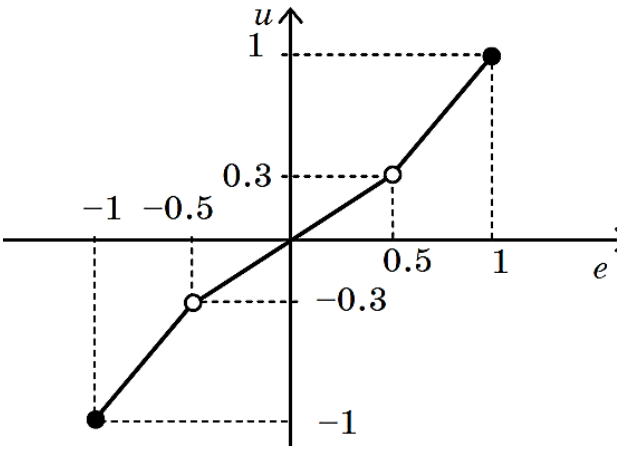


Рисунок 4.6 – Закон управління для регулятора з п’ятьома правилами

Як свідчить рис. 4.6, конфігурація кривої управління залежить від положення точок перегину (білі кружки), координати яких відповідають центрам термів НМ та ПМ вхідний та вихідний змінної НЛР_П (див. табл. 4.3). Оскільки ці центри симетричні, процес конструювання НЛР_П з п'ятьма правилами зводиться до вибору двох параметрів.

Якщо розглядати сім термів, то вибору підлягають 4 параметри, якщо дев'ять - то 6 і т.д. Таким чином, при *N* термах кількість параметрів, що настроюються *K* = *N* - 3.

У загальному випадку кількість термів (і, відповідно, правил) має бути такою, щоб задовольнити всі поставлені вимоги.

Розглянемо задачу конструювання нелінійного НЛР_П зі змінним коефіцієнтом посилення K_f як завдання корекції лінійного П-регулятора, що описується коефіцієнтом k_p. Тут можливі два варіанти корекції (рис. 4.7 та рис. 4.8).

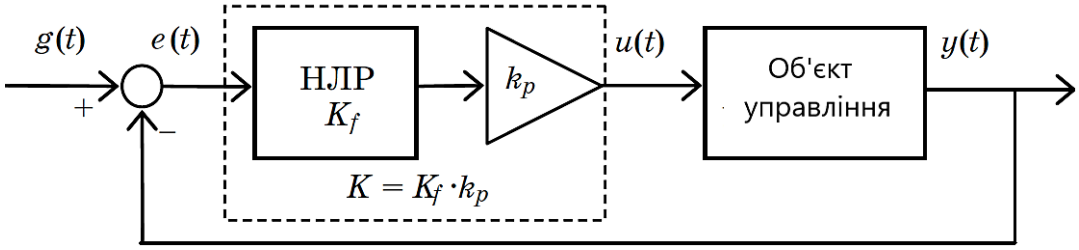


Рисунок 4.7 – Послідовна корекція П-регулятора

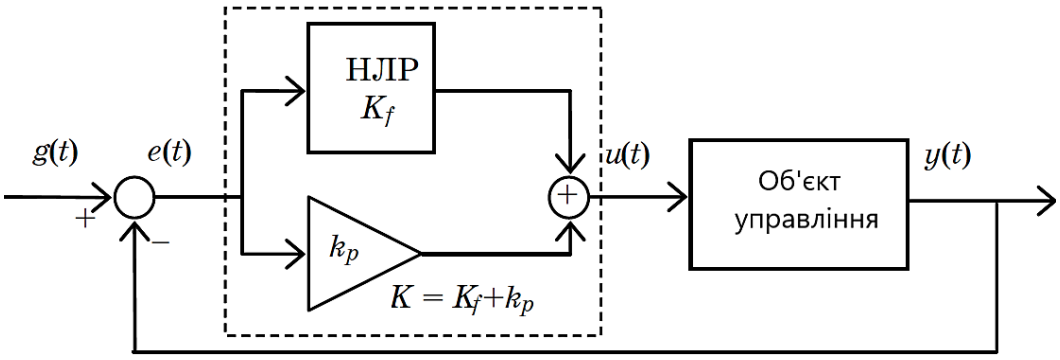


Рисунок 4.8 – Паралельна корекція П-регулятора

У цих структурах k_p визначає базовий коефіцієнт посилення. Постає питання – яким слід вибрати k_p? Оскільки НЛР_П має забезпечувати додаткове посилення, можна зажадати, щоб базовий П-регулятор забезпечував умову y_{max}=1. Тим самим гарантується, що помилка прийматиме всі значення в межах шкали [0, 1]. Інші параметри перехідного процесу можуть бути покращені за рахунок використання НЛР_П.

Таким чином, перший крок синтезу полягає у налаштуванні такого k_p, при якому забезпечується нульове перерегулювання (K_f = 1 (див. рис. 4.7) або K_f = 0 (див. рис. 4.8).

Базовий коефіцієнт посилення є дійсним числом, яке може набувати широкого діапазону значень залежно від об'єкта управління. Таким чином, можна або давати збільшення базовому коефіцієнту (див. рис. 4.8), або масштабувати його (див. рис. 4.7).

Далі розглядатимемо варіант послідовної корекції (див. рис.4.7) при використанні НЛР _П з сімома правилами.

Відповідно до (4.5) маємо

$$K_f = \frac{\Delta u}{\Delta e},$$

таким чином для отримання бажаного коефіцієнта посилення не обов'язково міняти обидва коефіцієнти – можна міняти тільки Δ*u* або Δ*e*, проте |Δ*u*| ≤ 1, тому далі оперуватимемо величиною Δ*e*.

Припустимо, що вихідні терми змінної НЛР_П рівномірно розподілені за базовою шкалою (див. рис. 4.9), тобто |Δ*u*|= 0,33.

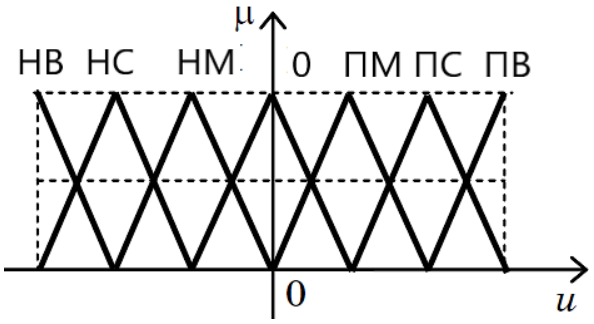


Рисунок 4.9 – Терми вихідної змінної НЛР_П із сімома правилами

Якщо терми вхідної змінної також рівномірно розташувати за базовою шкалою, то K_f = 1 будь-якого значення помилки (рис. 4.10).

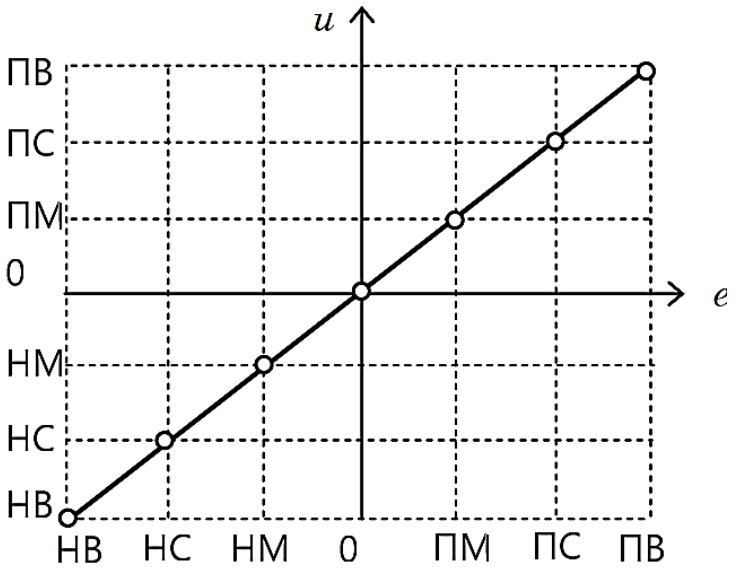


Рисунок 4.10 - Управляюча крива при рівномірному розподілі термів

На рис. 4.10 білими кружками позначені точки, у яких спрацьовує лише одне правило. Ці точки перебувають на перетині ліній, проведених із центрів однойменних термів. При переміщенні *e* між центрами термів керування лінійно змінюється. Таким чином, при семи керуючих правилах виходить п'ять лінійних ділянок закону управління.
Змінивши розташування центрів проміжних термів на базовій шкалі вхідної змінної можна отримати нелінійний закон управління (див. рис. 4.11).

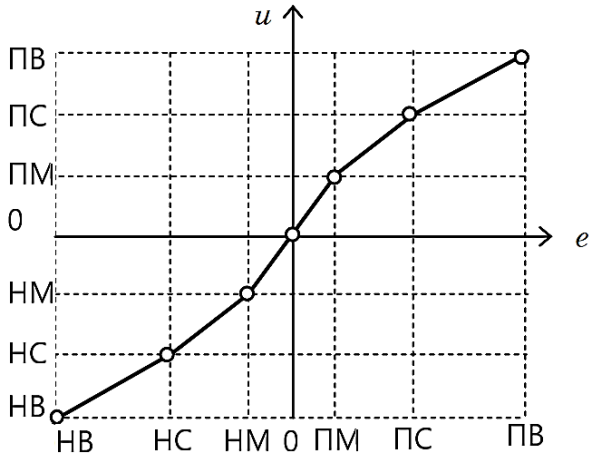


Рисунок 4.11 - Управляюча крива при нерівномірному розподілі термів

Центри термів НВ та ПВ для вхідної та вихідної змінної знаходяться у крайніх становища – у точках –1 та +1.

Розглянемо завдання вибору положення центру ПС (НС). За визначенням помилка e має в околиці цього терму середнє значення, тому вибір цього терму впливає на час наростання перехідного процесу: чим ближче ПС до ПВ, тим меншим буде час наростання, але при цьому можливе зростання перерегулювання.

При виборі положення центру ПМ (НМ) слід враховувати, що в околиці цього терма помилка має мале значення, отже, від вибору центру цього терму залежить значення помилки, що встановилася: чим менше Δe (тобто чим ближче ПМ і 0), тим більше K_f, і тим менше встановилася помилка.

Узагальнюючи всі вищенаведені міркування, можна остаточно сформулювати алгоритм синтезу нелінійного НЛР_П у межах структури, показаної на рис. 4.7.

1. Терми вхідних і вихідних змінних розподіляються рівномірно за базовими шкалами, отже K_f = 1. Вибирається значення k_p, у якому виконується умова y_{max}=1.

2. Розглядається регулятор із трьома правилами (див. рис. 4.2, 4.3). Вибирається таке значення x_1 (див. рис. 4.3), за якого забезпечується мінімальний час наростання при максимально допустимому перерегулюванні.

3. Знову розглядається регулятор із трьома правилами. Вибирається таке значення x_2, при якому забезпечується мінімальна помилка, що встановилася.

4. Розглядаються графіки трьох законів управління. Терми вихідної змінної регулятора можна рівномірно розташувати за відповідною базовою шкалою (див. рис. 4.9).

Для визначення положення центрів термів ПМ та ПС по осі e розглядаються точки перетину ліній, проведених із центрів термів ПМ та ПС по осі u та прямих, що описують закони управління x_1 та x_2 (див.

рис. 4.12). Координати точок перетину по осі *e* дають положення центрів термів ПМ та ПС для вхідної змінної регулятора.

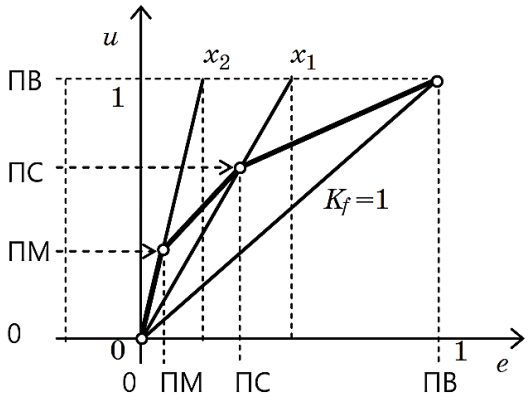


Рисунок 4.12 – Визначення положення термів НЛР_П

Очевидні геометричні розрахунки дозволяють описати закон управління як табл.4.4.

Таблиця 4.4 – Закон управління НЛР_П

	НВ (негативне велике)	НС (негативне середнє)	НМ (негативне мале)	Н (нульове)	ПМ (позитивне мале)	ПС (позитивне середнє)	ПВ (позитивне велике)
e	-1	$-0{,}66x_2$	$-0{,}33x_1$	0	$0{,}33x_1$	$0{,}66x_2$	1
u	-1	-0,66	0,33	0	0,33	0,66	1

Ширина всіх термів (включаючи терм 0) вибирається таким чином, щоб забезпечити нечітке розбиття базової шкали вхідної змінної (див. рис. 4.13).

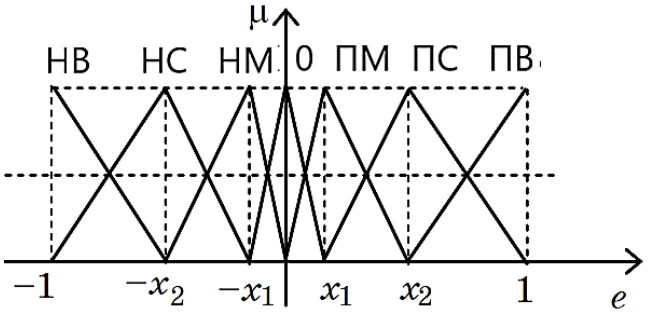


Рисунок 4.13 – Результати налаштування нечіткого регулятора П-типу

Нечіткий регулятор П-типу є основою розробки НЛР_ПД, НЛР_ПІ та НЛР_ПІД, оскільки використання у законі управління похідної помилки може зменшити перерегулювання, а використання інтеграла помилки – зменшити встановлену помилку.

Таким чином, розробка НЛР_П є першим кроком при проектуванні складніших нечітких регуляторів.

4.2 Приклад синтезу нечіткого регулятора П-типу

У Simulink MatLab контейнером для системи нечіткого логічного висновку є блок Fuzzy Logic Controller (пакет Fuzzy Logic ToolBox).

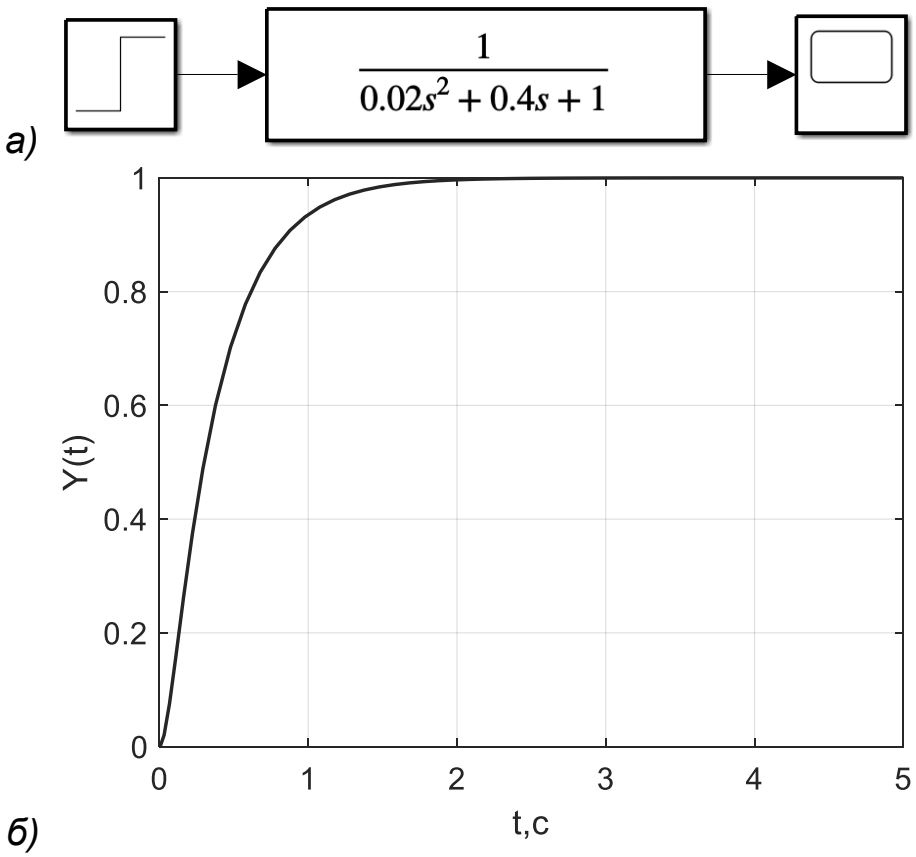
Синтезуємо нелінійний НЛР_П відповідно до методики, викладеної в п. 4.1, для об'єкта, що описується передаточною функцією:

$$W(p) = \frac{1}{0.02p^2 + 0.4p + 1}.$$

Цей об'єкт є стійким, але час перехідного процесу дуже великий (див. рис. 4.14).

Потрібно забезпечити перехідний процес з малим часом наростання, що встановилася помилкою Δ ≤ 3% та перерегулюванням d ≤ 10%.

Використовуватимемо описаний вище 3-кроковий алгоритм проектування.



а) математична модель; б) графік перехідного процесу

Рисунок 4.14 – Математичне моделювання об’єкта управління

1. Розглянемо НЛР_П з із сімома правилами, що має одиничний коефіцієнт посилення (див. рис. 4.9 та 4.13). Такий регулятор не впливає процес управління. Виберемо базовий коефіцієнт посилення k_p=6.5, при цьому забезпечується умова y_{max} = 1 (див. рис. 4.15 та 4.16).

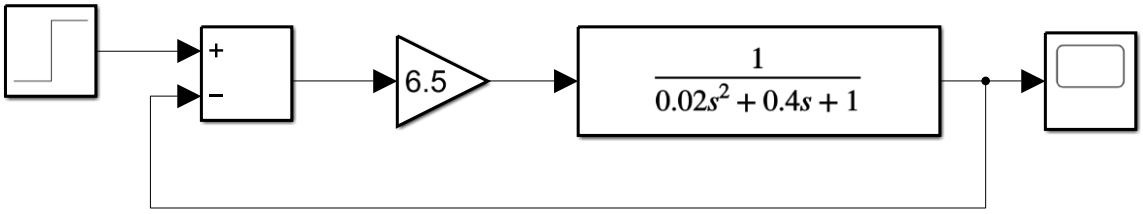


Рисунок 4.15 – Математична модель САУ з базовим П-регулятором

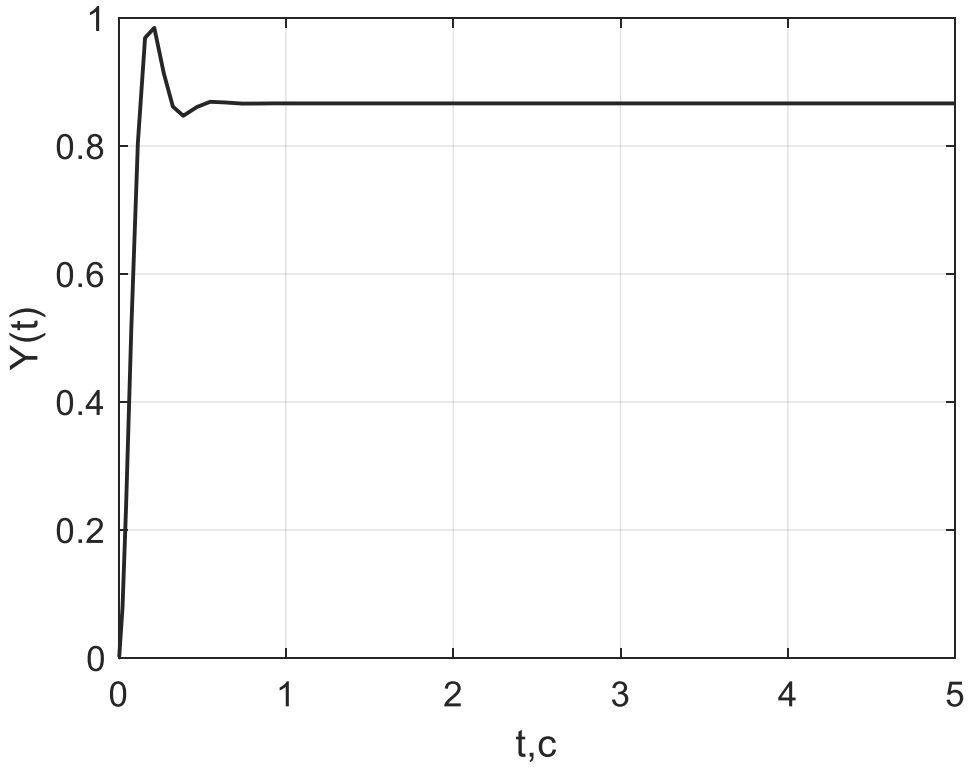


Рисунок 4.16 – Графік перехідного процесу при базовому коефіцієнте підсилення

Таким чином, центрами термів НВ та ПВ будуть точки –1 та 1.

2. Розглянемо НЛР_П із сімома правилами (рис. 4.17).

Блок Saturation використовується для приведення вхідної змінної до заданого діапазону [-1;1]. Терми вихідний змінної регулятора розташовуються за базовою шкалою відповідно до рис. 11.63, терми вхідної змінної – відповідно до рис. 4.13. Методом спроб та помилок вибирається значення x_1 = 0.5 (рис. 4.13), при цьому забезпечується мінімальний час наростання при заданому перерегулюванні.

Центрами термів ОС та ПС будуть точки –0.5 та 0.5.

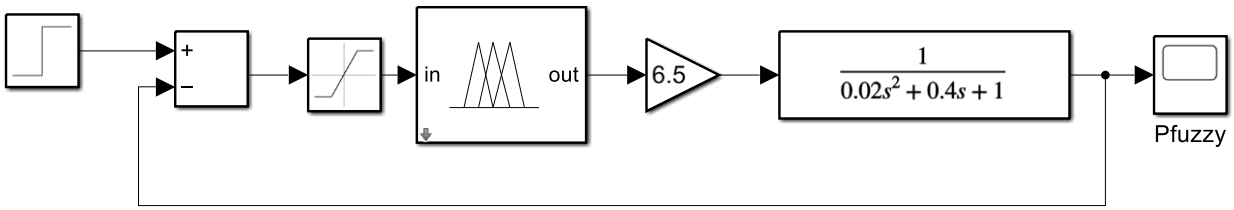


Рисунок 4.17 – Математична модель системи управління з нечітким регулятором

3. На третьому кроці знову розглядається НЛР_П із сім'ю правилами. Вибирається коефіцієнт посилення регулятора x_2 такий, щоб Δ<3%. Центрами термів НМ та ПМ будуть точки –0.1 та 0.1.

4. На останньому етапі необхідно отримати нелінійний закон управління НЛР_П. Центри термів вхідної (*e*) вихідний змінної (*u*) розраховуються відповідно до табл. 4.4. У таблиці 4.5 наведено закон управління проектованим НЛР_П.

Таблиця 4.5 – 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)
e	-1	-0,33	-0,033	0	0,033	0,33	1
u	-1	-0,66	0,33	0	0,33	0,66	1

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. 4.17). Fuzzy Logic Controller надаємо Fis name: Pfis7 (ім’я може буди будь-яке).

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

>> fuzzy

Дале проводиться налаштовування головного вікна редактора нечіткої логічної системи (див. рис. 4.18.).

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]).

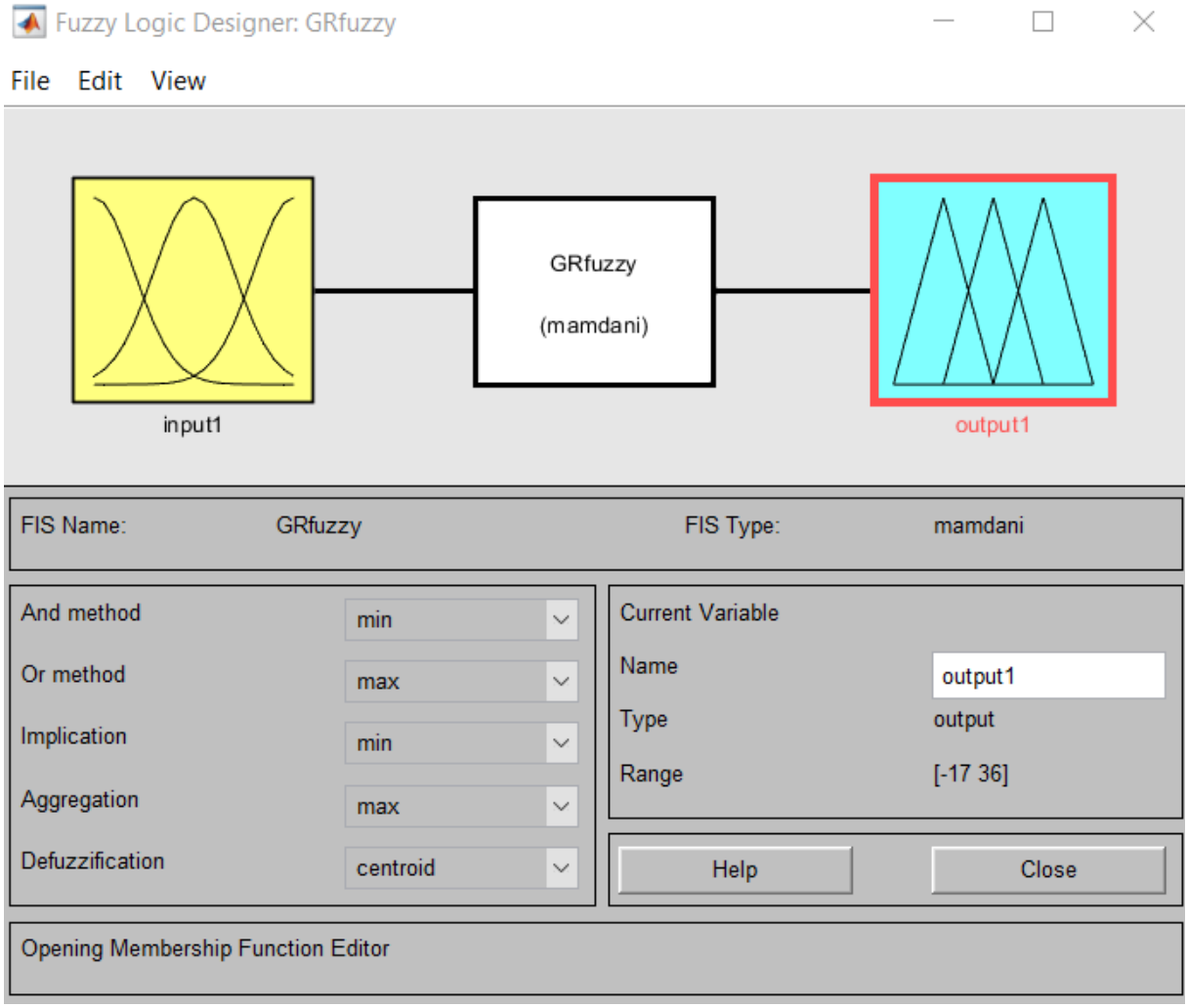


Рисунок 4.18 – Налаштування інтерфейсу FIS editor

Для опису вхідних логічних змінних у редакторі функцій приналежності (*Membership Function Editor*), задаємо для кожної змінної трикутну функцію приналежності. З параметрами:
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 наведено на рис. 4.19.

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

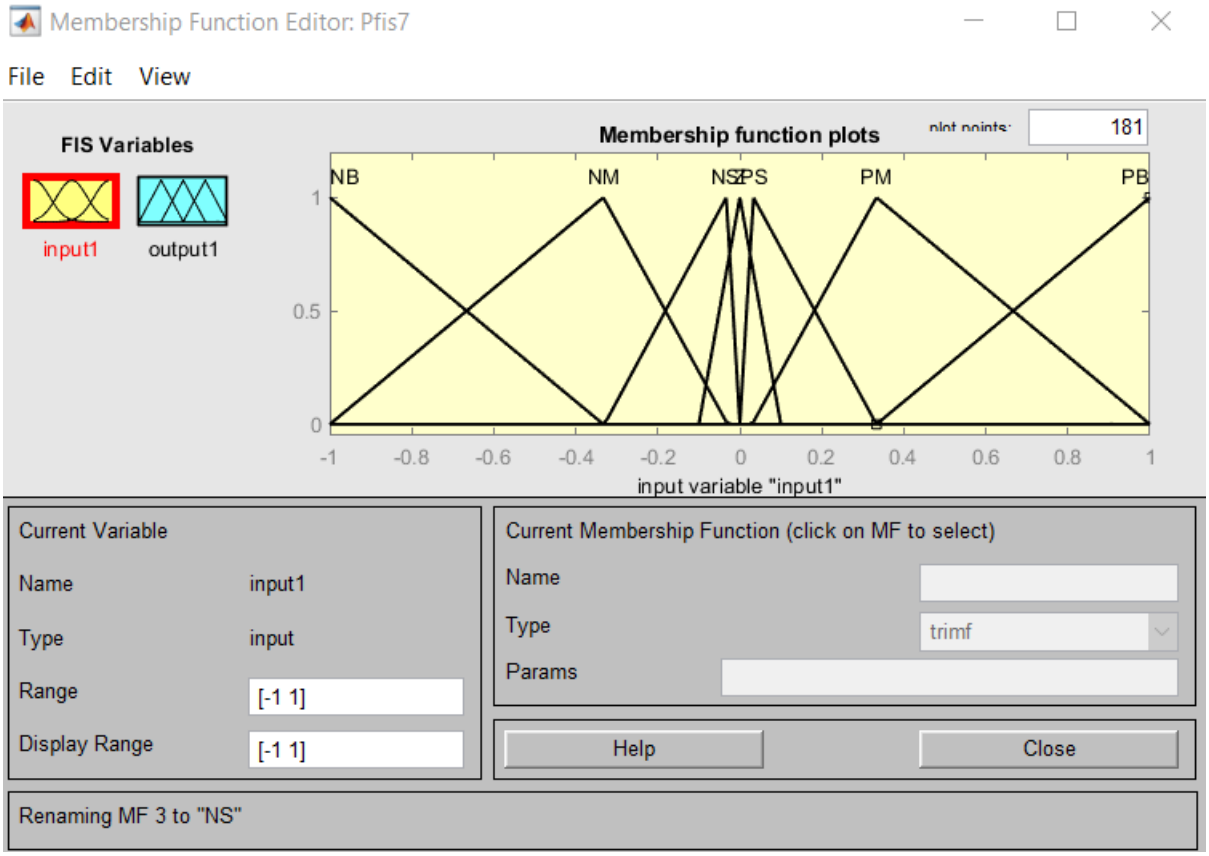


Рисунок 4.19 – Результат налаштування функцій приналежності вхідних змінних

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 вихідних змінних наведено на рис. 4.20.

Примітка. У якості центрів термів, що описуються функціями приналежності, обрані значення відповідно до табл. 4.5.

Після опису вхідні та вихідні лінгвістичні змінні, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку дорівнює 7.

1. if (input1 is NB) then (output1 is NBu) (1)
2. if (input1 is NM) then (output1 is NMu) (1)
3. if (input1 is NS) then (output1 is NSu) (1)
4. if (input1 is Z) then (output1 is Zu) (1)
5. if (input1 is PS) then (output1 is PSu) (1)
6. if (input1 is PM) then (output1 is NMu) (1)
7. if (input1 is PB) then (output1 is PBu) (1)

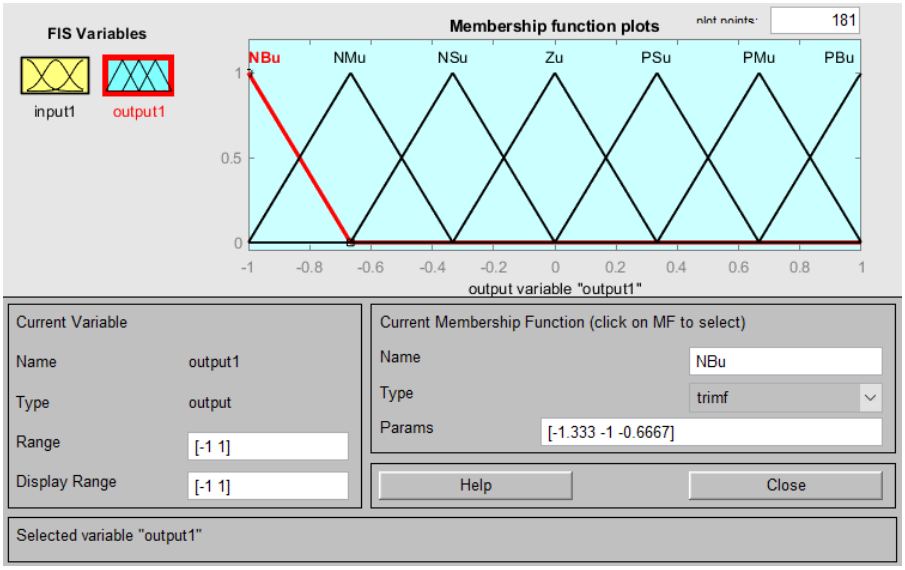


Рисунок 4.20 – Результат налаштування функцій приналежності вихідних змінних

Налаштовування правил для даного випадку наведено на рисунку (див. рис. 4.21).

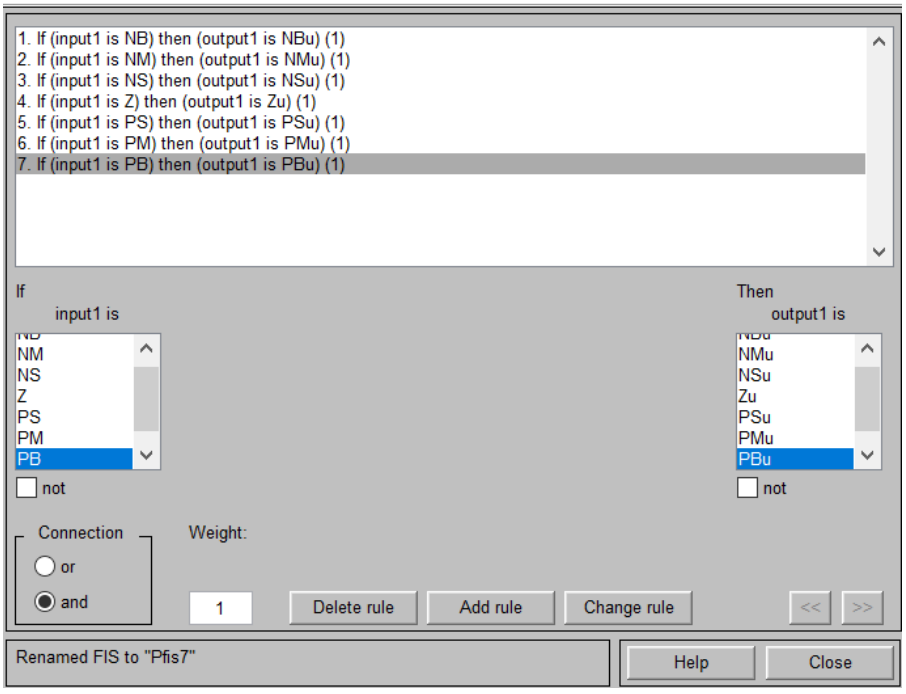


Рисунок 4.21 - Налаштовування опису правил функціонування НЛР_П

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. 4.22);

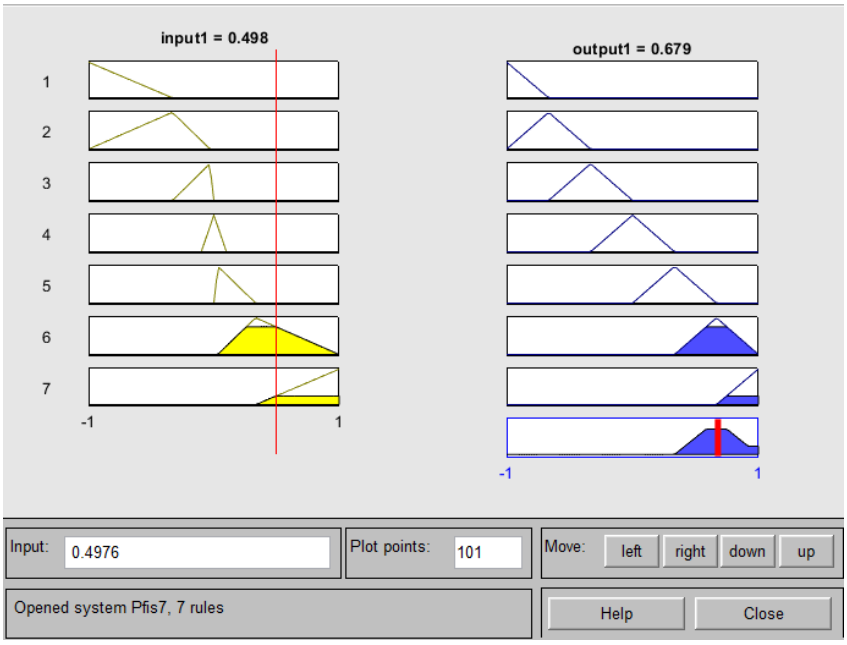


Рисунок 4.22 - Робота системи НЛР_П

– інтерфейсу Surface Viewer можливо переглянути графічне подання закону управління (рис. 4.23).

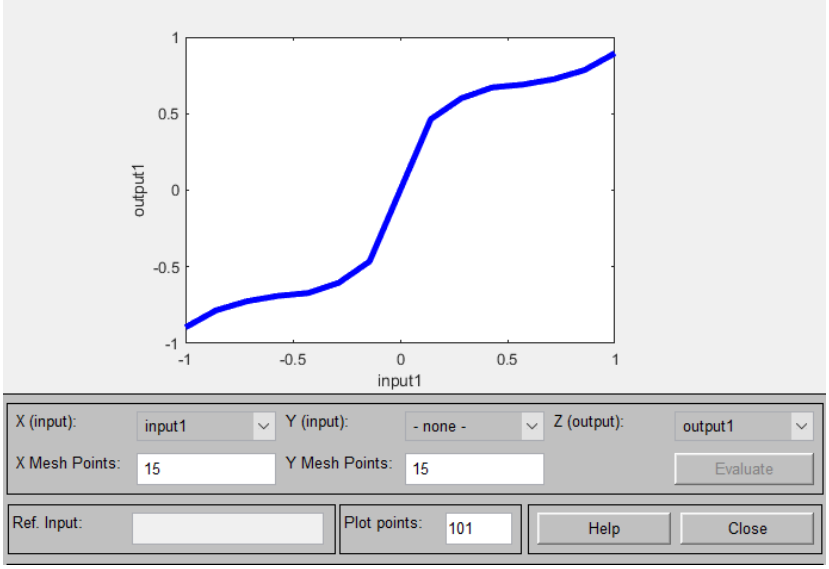


Рисунок 4.23 – Крива управління системою нечіткого висновку

Після процедур налаштування НЛР_П 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_П на математичної моделі системи управління (див. рис. 4.17) получено перехідний процес для синтезованого НЛР_П, який задовольняє поставленому завданню

проектування

На рис. 4.24 показаний перехідний процес для синтезованого НЛР_П.

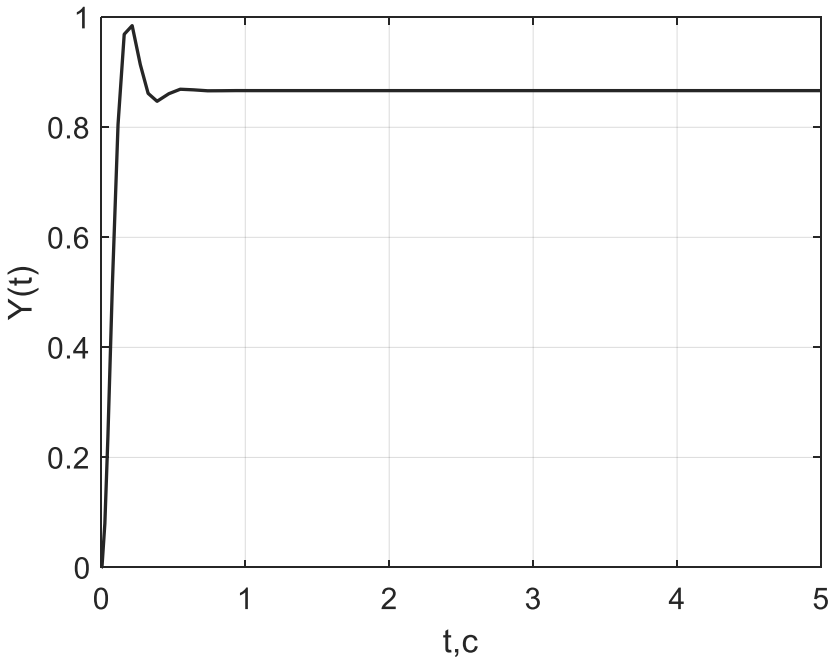


Рисунок 4.24 - Перехідний процес для НЛР_П із сімома правилами

4.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 4.6.

За вказаними у табл. 4.6 потрібно синтезувати нелінійний нечіткий регулятор П-типу з використанням системи нечіткого висновку:

Таблиця 4.6 - 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	$W(p) = \frac{3}{p^2 + 0.8p + 1}$

Продовження табл. 4.6

№ вар	*Передаточна функція об'єкту управління*
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	$W(p) = \frac{6.3}{5.2p^2 + 2.1p + 1}$
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	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

4.4 Контрольні питання

1. Опишіть принципи евристичного синтезу нечіткого регулятора П-типу.
2. За яких умов нечіткий регулятор П-типу виявляється лінійним?
3. Опишіть умови нелінійності закону керування нечіткого регулятора П-типу. У чому переваги нелінійного закону управління?
4. Скільки кроків включає синтез нелінійного НЛР _ П? Які це кроки?

5 СИНТЕЗ НЕЛІНІЙНОГО НЕЧІТКОГО РЕГУЛЯТОРА ПД-ТИПУ

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

5.1 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. 5.1.

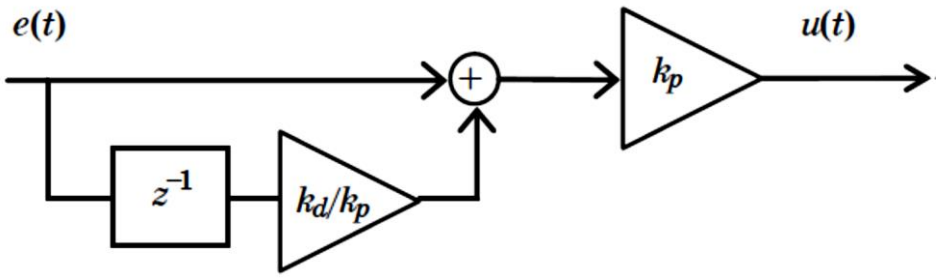


Рисунок 5.1 – Варіант опису ПД-регулятора

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.

5.2 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

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

де Fл - 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:

$$\Delta e(t) = e(t) - e(t-1) = g(t) - y(t) - g(t) + y(t-1) = y(t-1) - y(t) = -\Delta y(t)$$

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. 5.2 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.

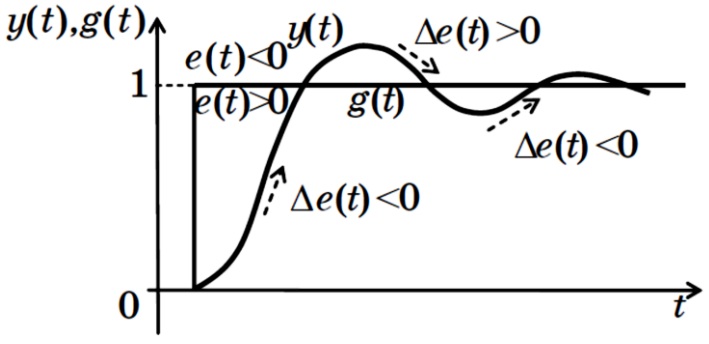


Рисунок 5.2 – Зміна помилки при нормальному перебігу перехідного процесу

Зроблені зауваження дозволяють класифікувати всі, що виникають при управлінні ситуації (див. табл. 5.1).

Таблиця 5.1 – 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. 5.1 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.5.2).

Таблиця 5.2 – Лінгвістичні правила НЛР_ПД

Таблиця правил		e*			
		Н	0	П	
Δ e*	Н	Н	Н	0	u*
	0	Н	0	П	
	П	0	П	П	

Табл. 5.2 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 (4.4), для П-регулятора знак управління збігається зі знаком помилки. Наприклад, при 5 термах справедлива табл. 5.3.

Таблиця. 5.3 – 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 Δ*e** 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+НВ=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. 5.4. Табл. 5.4 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. 5.5).

Таблиця 5.4 – Табличні лінгвістичні правила

Таблиця правил		e*						
		НВ	НС	НМ	0	ПМ	ПС	ПВ
Δ e*	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

u*

Таблиця 5.5 – Модифіковані табличні лінгвістичні правила

Таблиця правил		e*						
		НВ	НС	НМ	0	ПМ	ПС	ПВ
Δ e*	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

u*

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

Евристичний закон керування, заданий табл. 5.4 та 5.5, 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.

5.3 Приклад синтезу нечіткого регулятора ПД-типу

Нехай як базова використовується структура, показана на рис. 4.15. При заміні регулятора на НЛР_ПД, вона набуває вигляду, показаного на рис. 5.3.

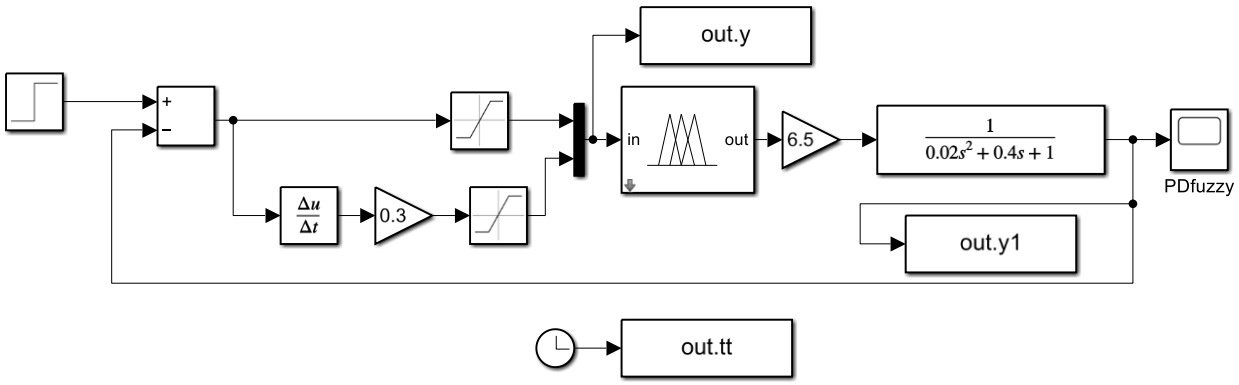


Рисунок 5.3 – Математична модель системи управління з НЛР_ПД

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 буди будь-яке).

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

>> fuzzy

Дале проводиться налаштовування головного вікна редактора нечіткої логічної системи (див. рис. 5.4.).

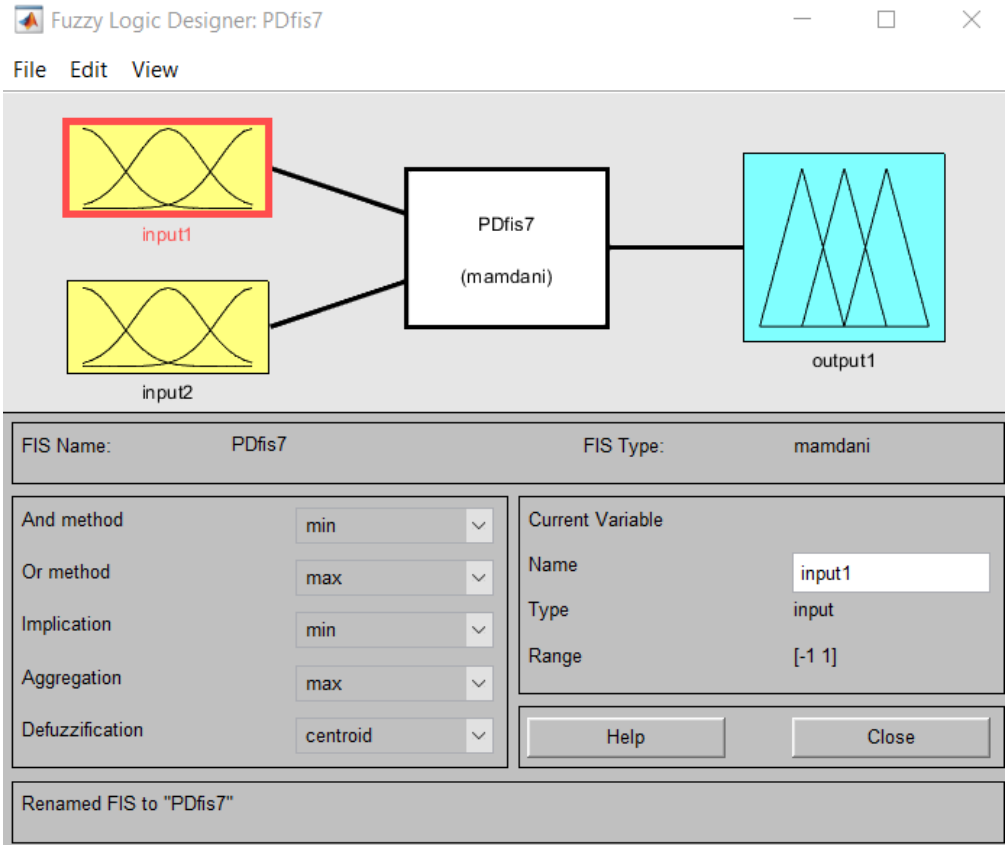


Рисунок 5.4 – Налаштування інтерфейсу FIS editor

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 рис. 4.19 згідно налаштуванням НЛР_П регулятора з параметрами:

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. 5.5.

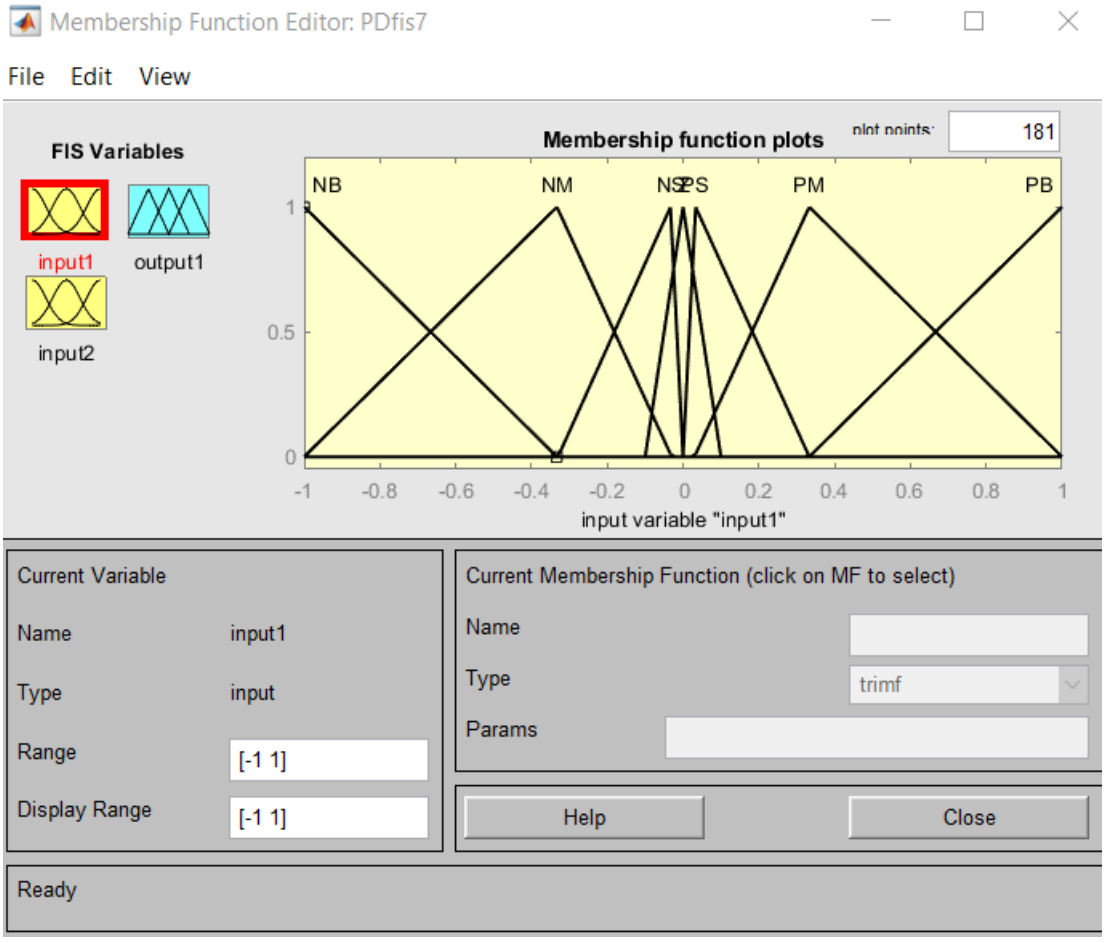


*Рисунок 5.5 – Результат налаштування функцій приналежності вхідних змінних «*Помилка управління»

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. 5.6.

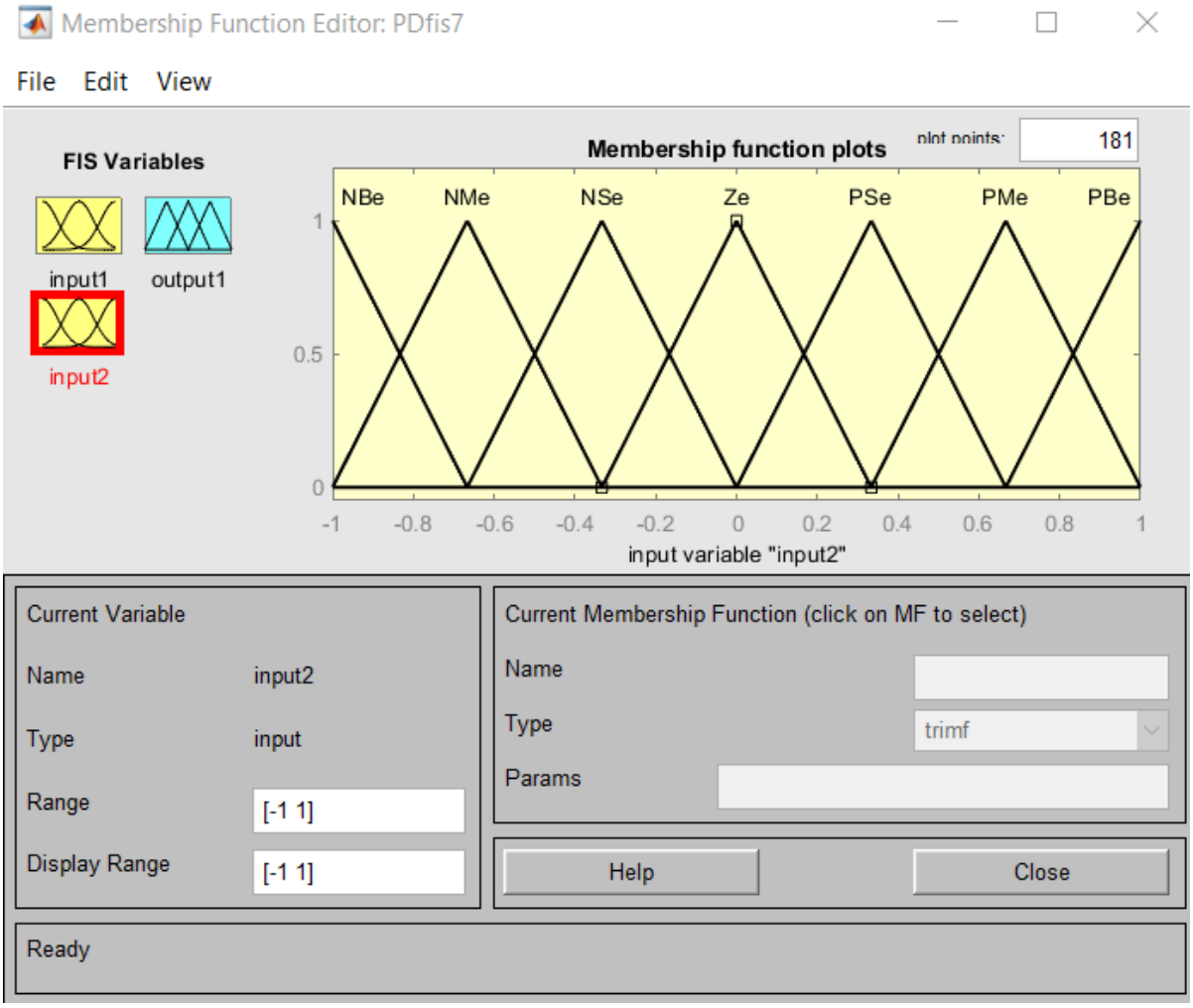


*Рисунок 5.6 – Результат налаштування функцій приналежності вхідних змінних «Похідна п*омилки управління»

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. 5.7.

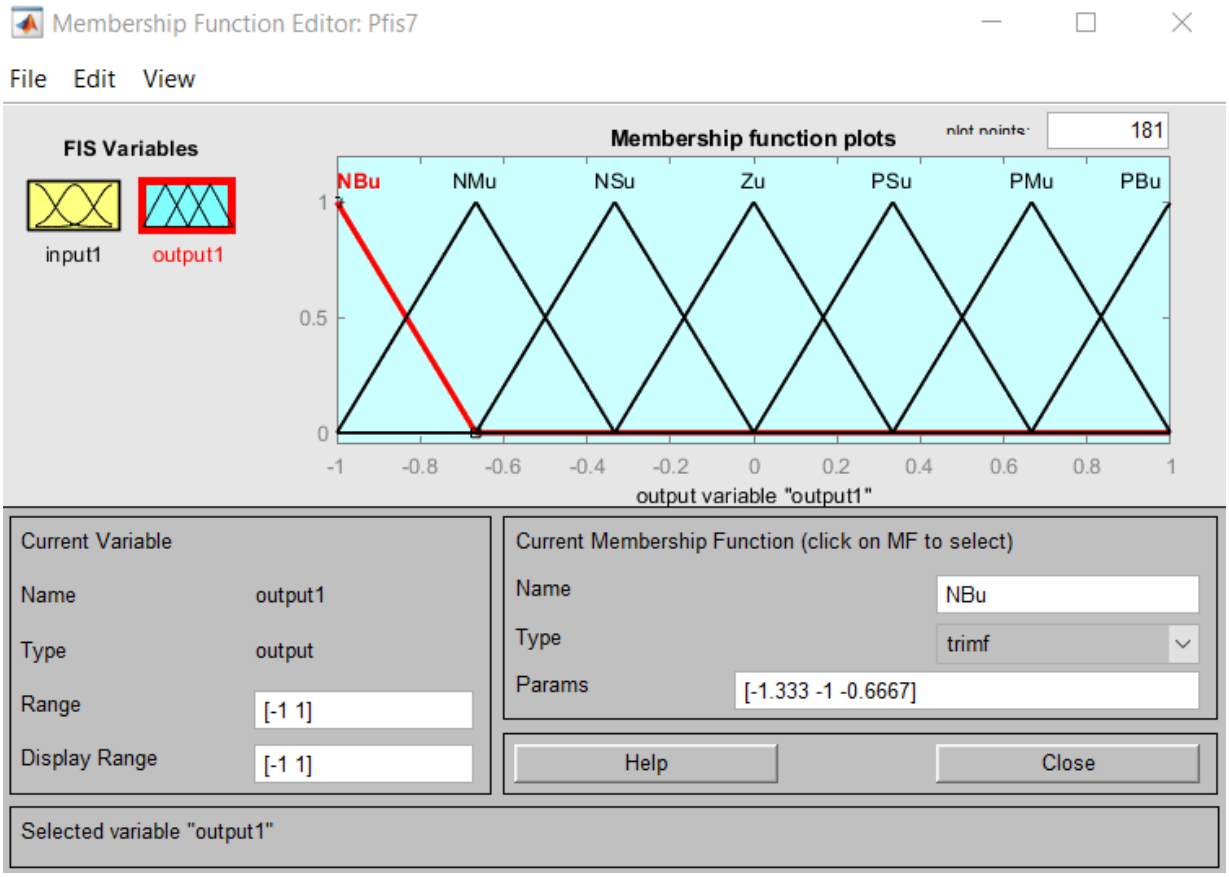


Рисунок 5.7 – Результат налаштування функцій приналежності вихідних змінних

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. 4.5:

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. 5.8).

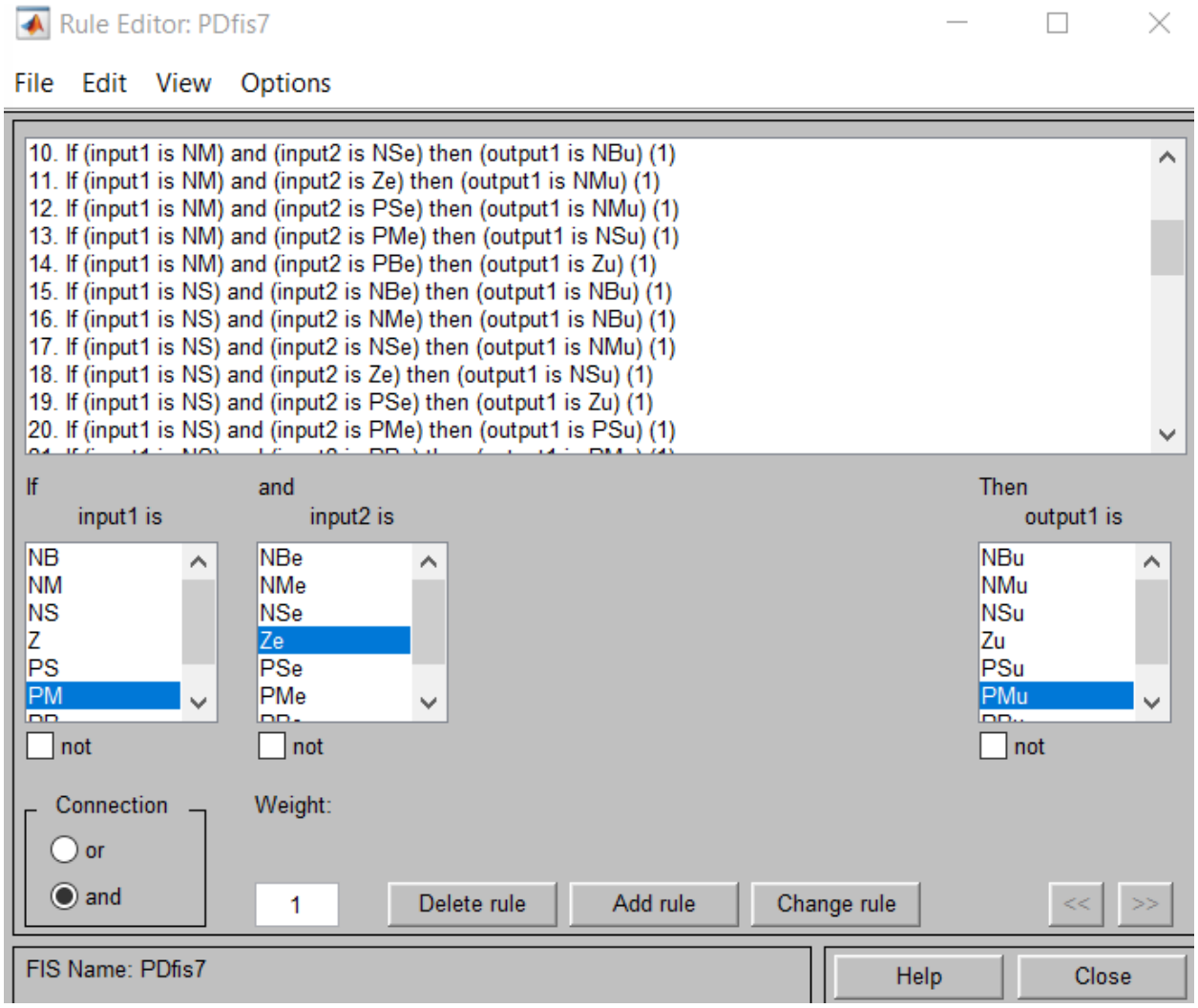


Рисунок 5.8 - Налаштовування опису правил функціонування НЛР_ПД

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. 5.9);

− інтерфейсу Surface Viewer можливо переглянути графічне подання закону управління (рис. 5.10).

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 математичної моделі системи управління (див. рис. 5.3) получено перехідний процес

для синтезованого НЛР_ПД, який задовольняє поставленому завданню проектування

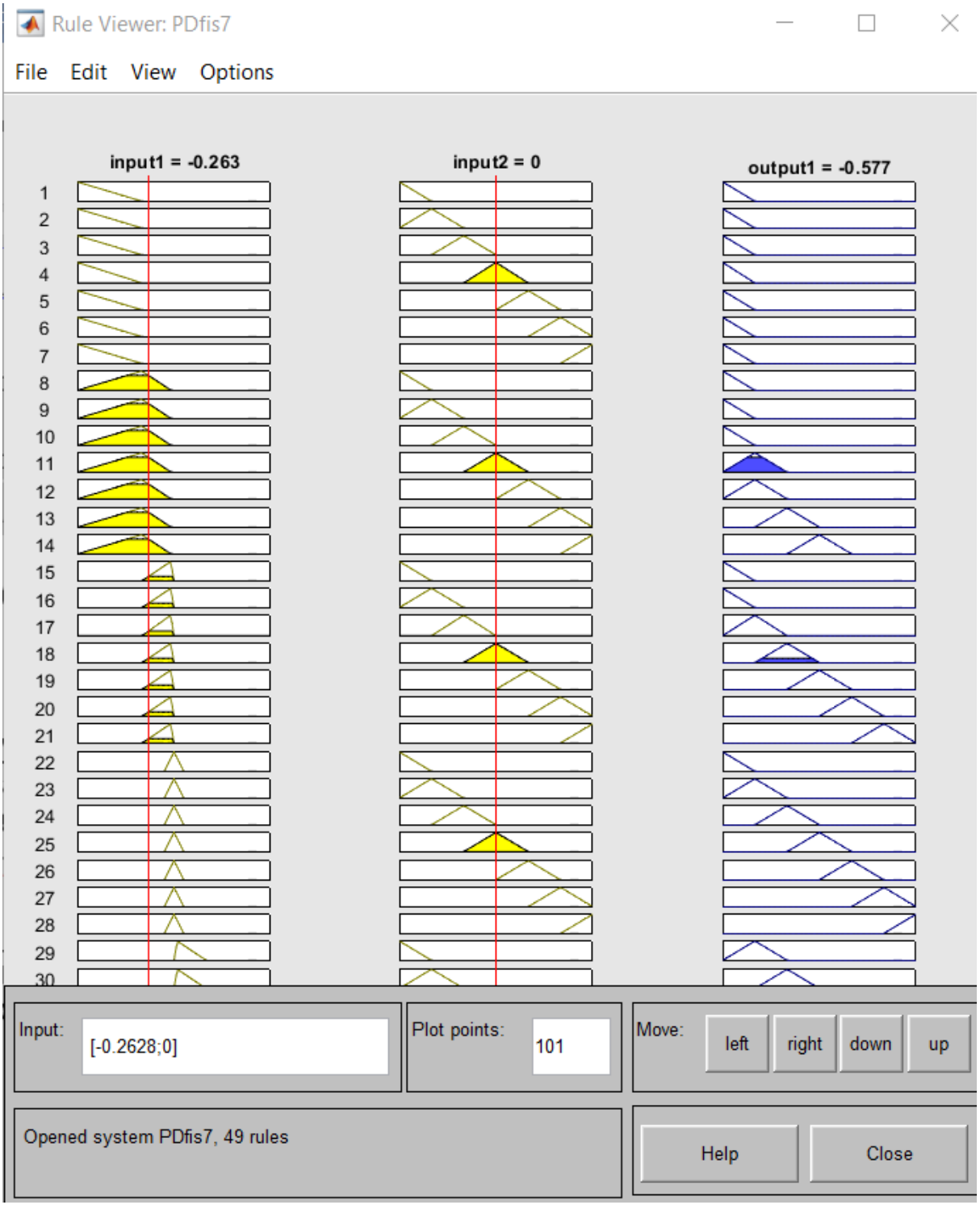


Рисунок 5.9 - Робота системи НЛР_ПД

На рис. 5.11 показаний перехідний процес для синтезованого НЛР_ПД. Як свідчить рис. 5.11, перерегулювання зведено до мінімуму.

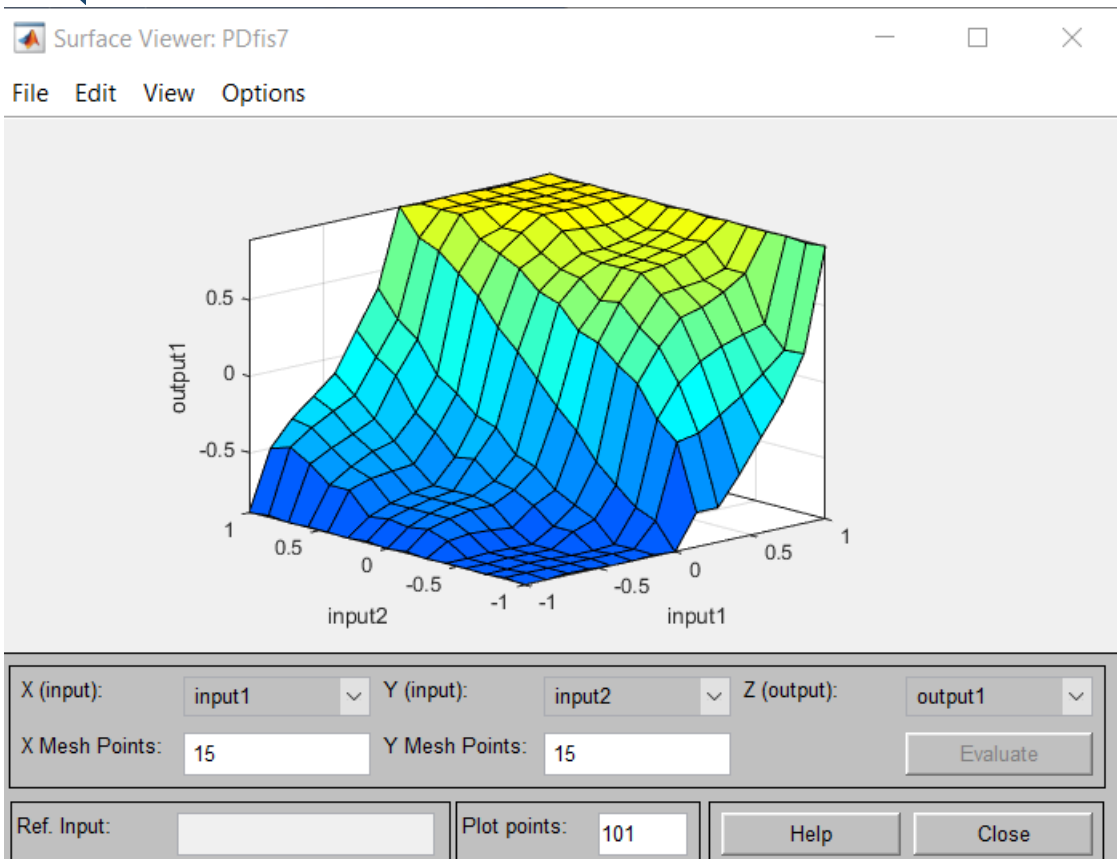


Рисунок 5.10 – Поверхня управління системою нечіткого висновку

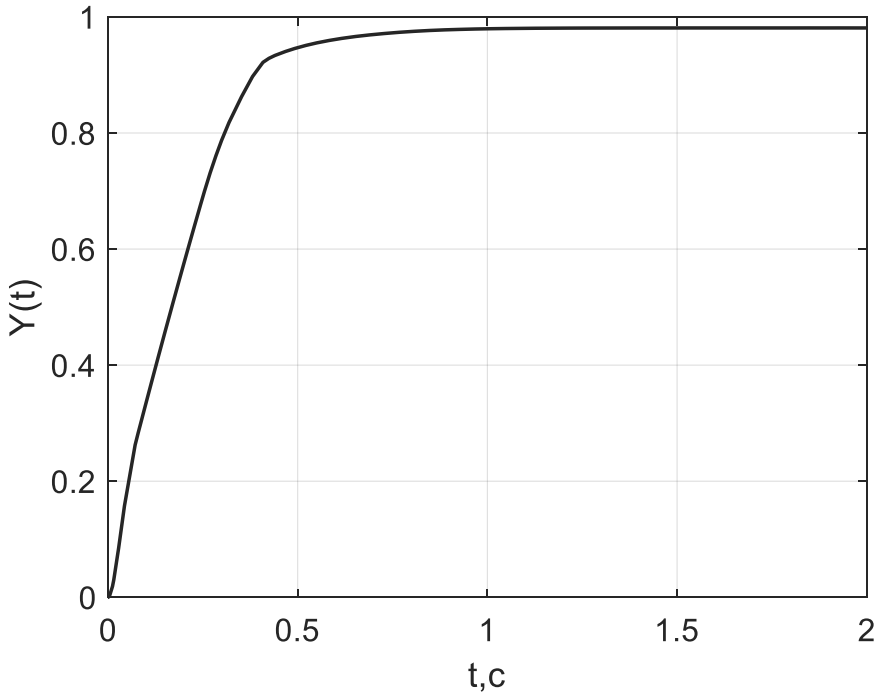


Рисунок 5.11 - Перехідний процес для НЛР_ПД із сімома правилами

Таким чином, вихідними даними для синтезу НЛР_ПД є опис об'єкта управління та вимоги до перехідного процесу: час наростання, перерегулювання, помилка, що встановилася.

5.4 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 5.6.

За вказаними у табл. 5.5 потрібно синтезувати нелінійний нечіткий регулятор ПД-типу з використанням системи нечіткого висновку:

Таблиця 5.6 - 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. 5.6

№ вар	*Передаточна 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	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

5.5 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?

6 СИНТЕЗ НЕЛІНІЙНОГО НЕЧІТКОГО РЕГУЛЯТОРА ПІ-ТИПУ

Мета роботи: засвоєння методики синтезу нелінійного нечіткого регулятору ПІ-типу з використанням редактора системи нечіткого висновку

6.1 Особливості синтезу нечіткого регулятору ПІ-типу

Можна вважати, що нечіткий регулятор ПІ-типу має такі самі входи, як і НЛР_ПД. Тому йому справедливі закони управління, показані в табл. 5.1–5.5. Основна відмінність полягає в способі формування сигналу, що управляє, використання якого дозволяє звести до мінімуму статичну помилку в системі.

Найпростіший алгоритм нечіткого ПІ управління можна сформулювати, розглядаючи рис. 5.2. Управління потрібно змінювати лише тоді, коли помилка управління та її збільшення мають однаковий знак. Таким чином, потрібні лише два правила:

− якщо $e(k)$ позитивна та $\Delta e(k)$ позитивна, то $\Delta u(k)$ позитивна;

− якщо $e(k)$ негативне та $\Delta e(k)$ негативне, то $\Delta u(k)$ негативне.

Терми лінгвістичних змінних можуть бути описані способом, показаним на рис. 6.1.

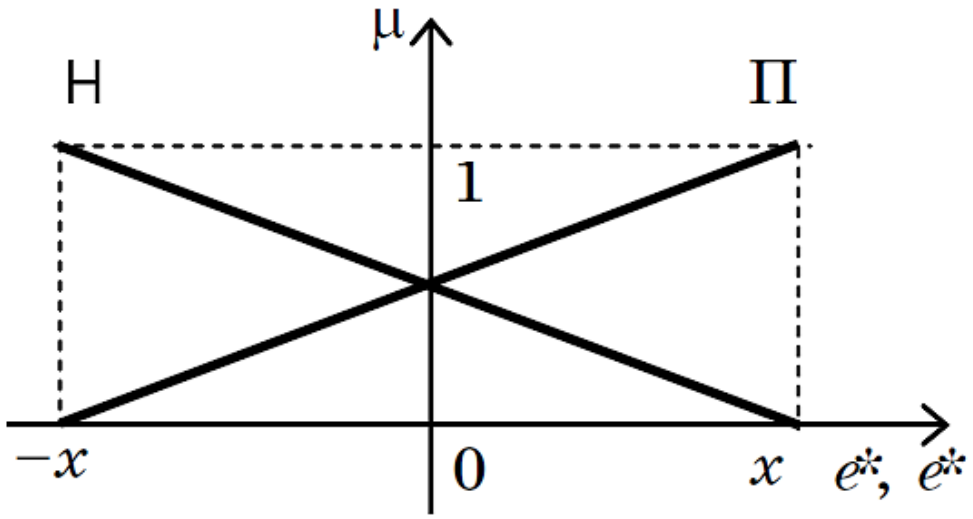


Рисунок 6.1 – Лінгвістичний опис входів та виходів НЛР_ПІ

Таблиця лінгвістичних правил НЛР_ПІ при двох термах для опису $e(k)$ та $\Delta e(k)$ і трьох термах для $u(k)$ (див. рис. 6.56) набуває наступного вигляду (табл. 6.1).

Отже, універсальний закон управління, заданий табл. 6.15 вимагає для конкретного об'єкта налаштування лише одного параметра x (див. рис. 10.1).

Таблиця 6.1 – Лінгвістичні правила НЛР_ПІ

Таблиця правил		e*(k)	
		Н	П
Δe*(k)	Н	Н	0
	П	П	П

Δu*(t)

6.3 Приклад синтезу нечіткого регулятора ПІ-типу

Будуємо в Simulink MatLab модель системи управління з НЛР ПІ при x = 6.5 (див. рис. 6.2).

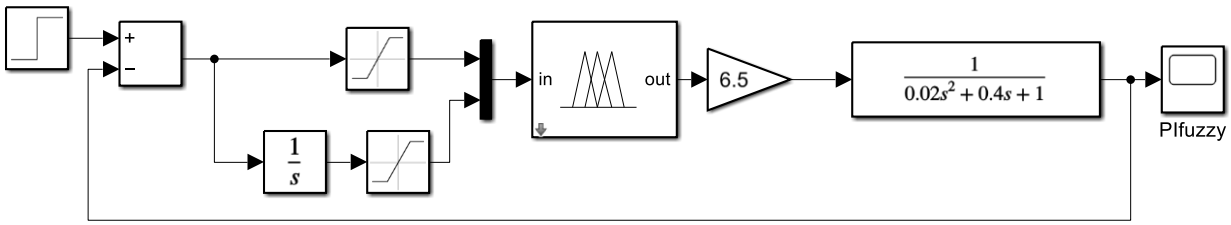


Рисунок 6.2 – Математична модель системи управління з НЛР_ПІ у в Simulink MatLab

Fuzzy Logic Controller надаємо Fis name: PIfis7 (ім'я може буди будь-яке).

Налаштування Fuzzy Logic Controller для НЛР_ПІ аналогічно налаштуванню НЛР_ПД.

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

>> fuzzy

Дале проводиться налаштовування головного вікна редактора нечіткої логічної системи (див. рис. 6.3).

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*= [-1 1]).

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

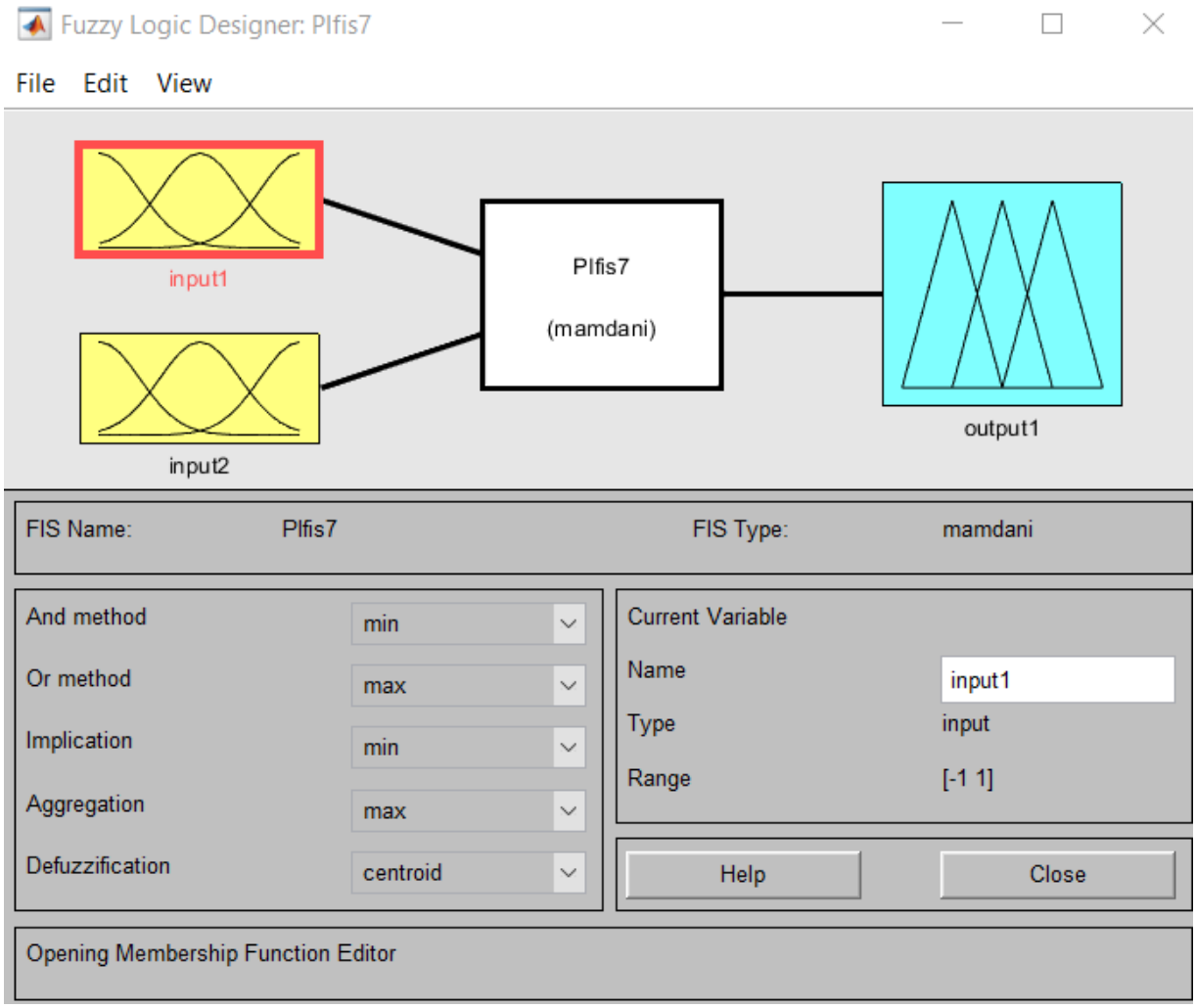


Рисунок 6.3 – Налаштування інтерфейсу FIS editor

Нехай терми лінгвістичної змінної «Помилка управління» розміщуються відповідно до рис. 4.19 згідно налаштуванням НЛР_П регулятора з параметрами:
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. 6.4.

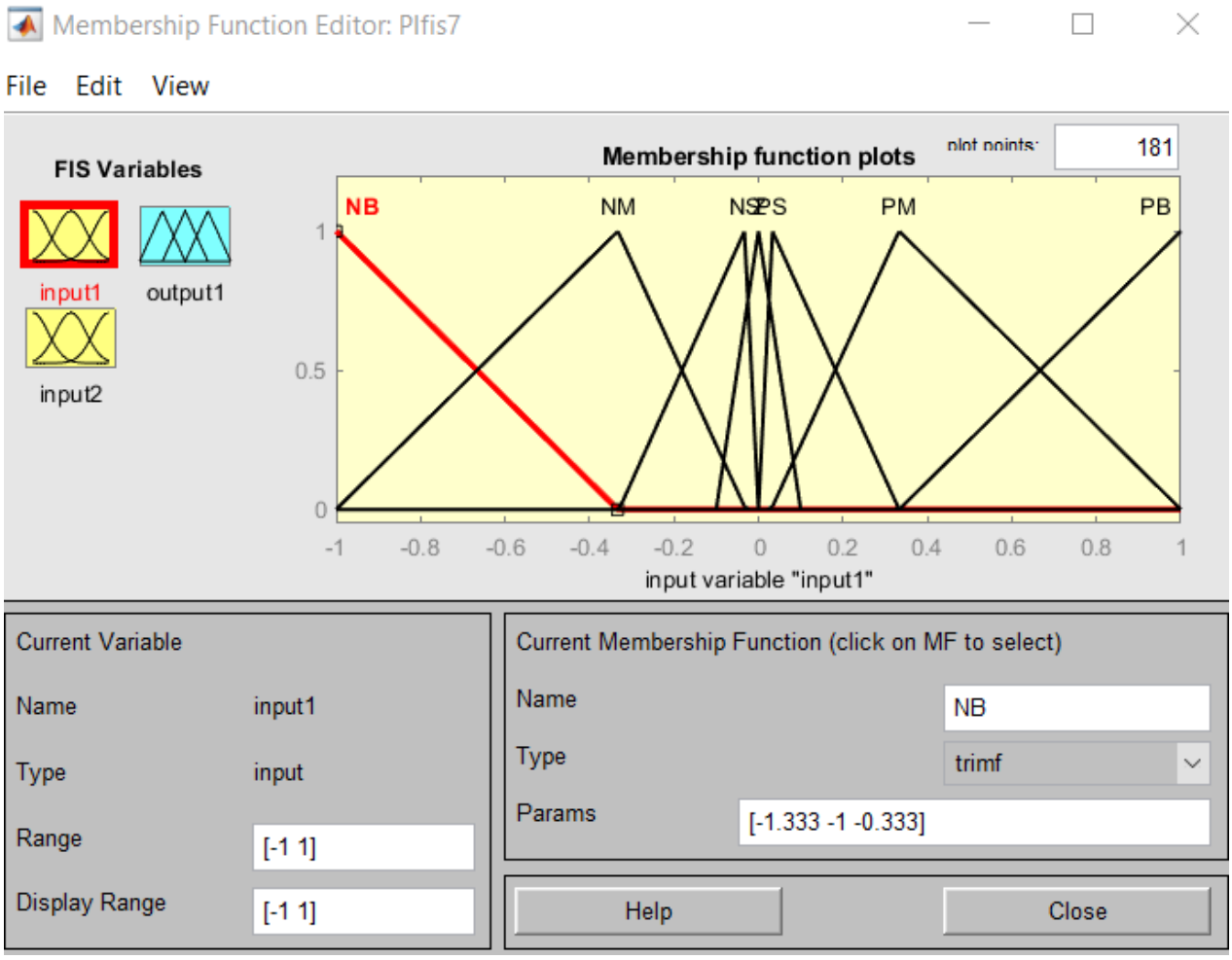


Рисунок 6.4 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

Терми вхідної лінгвістичної змінної «Інтеграл помилки управління» формуємо рівномірно. Для опису вхідних логічних змінних «Інтеграл 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 «Інтеграл помилки управління» на рис. 6.5.

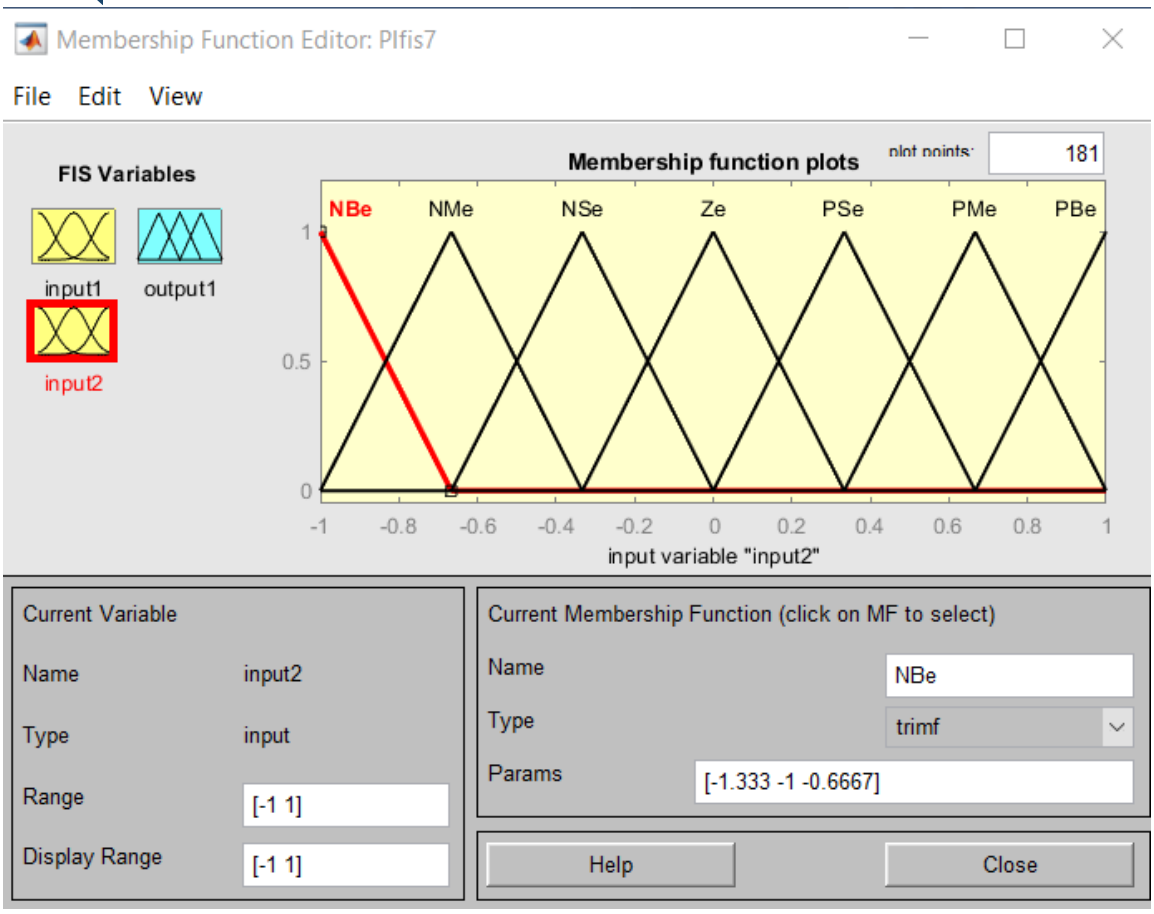


*Рисунок 6.5 – Результат налаштування функцій приналежності вхідних змінних «Інтеграл п*омилки управління»

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. 6.6.

Примітка. У якості центрів термів, що описуються функціями приналежності, обрані значення відповідно до табл. 4.4.

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. 5.5:

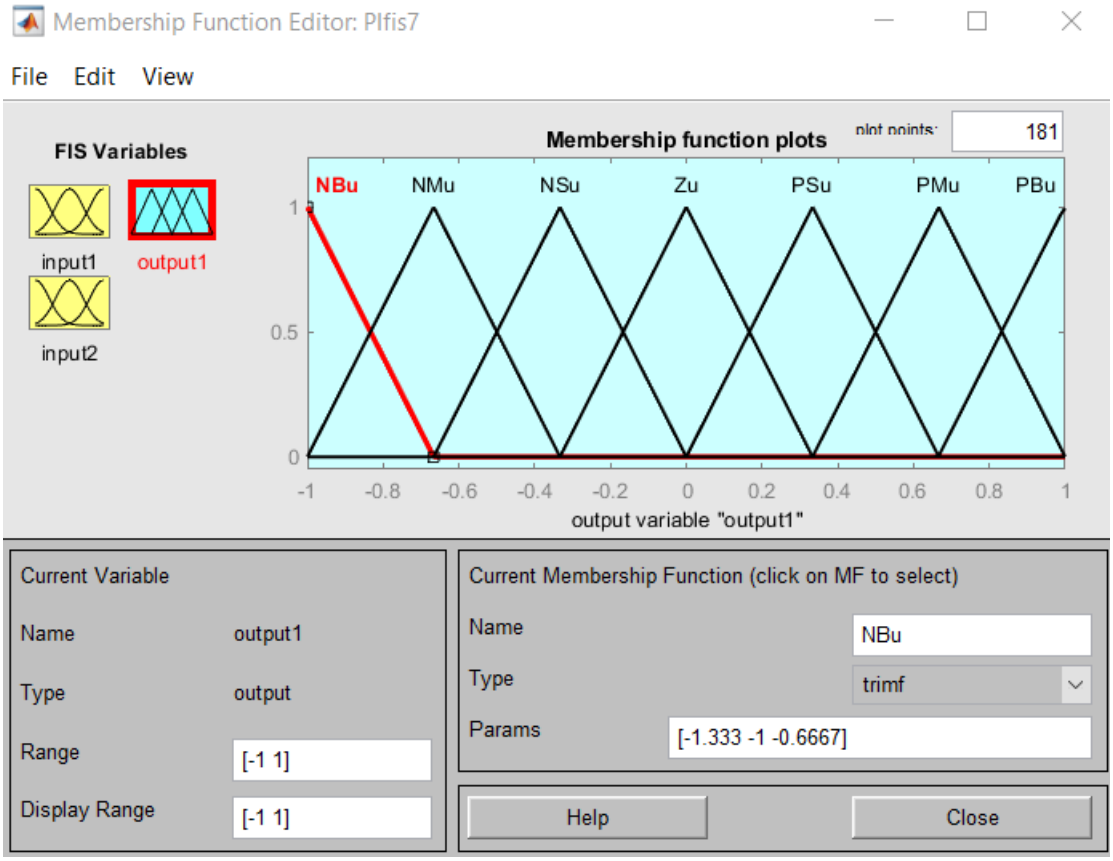


Рисунок 6.6 – Результат налаштування функцій приналежності вихідних змінних

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. 6.7).

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. 6.8);

− інтерфейсу Surface Viewer можливо переглянути графічне подання закону управління (рис. 6.9).

Після процедур налаштування НЛР_ПІ 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.

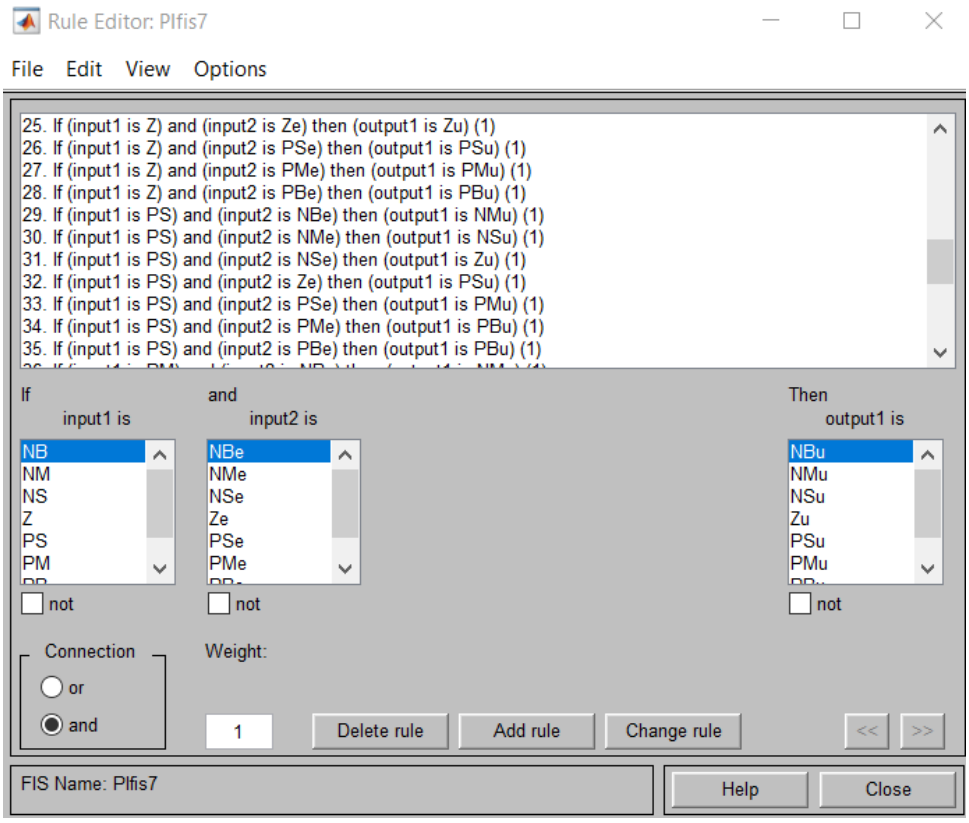


Рисунок 6.7 - Налаштовування опису правил функціонування НЛР_ПІ

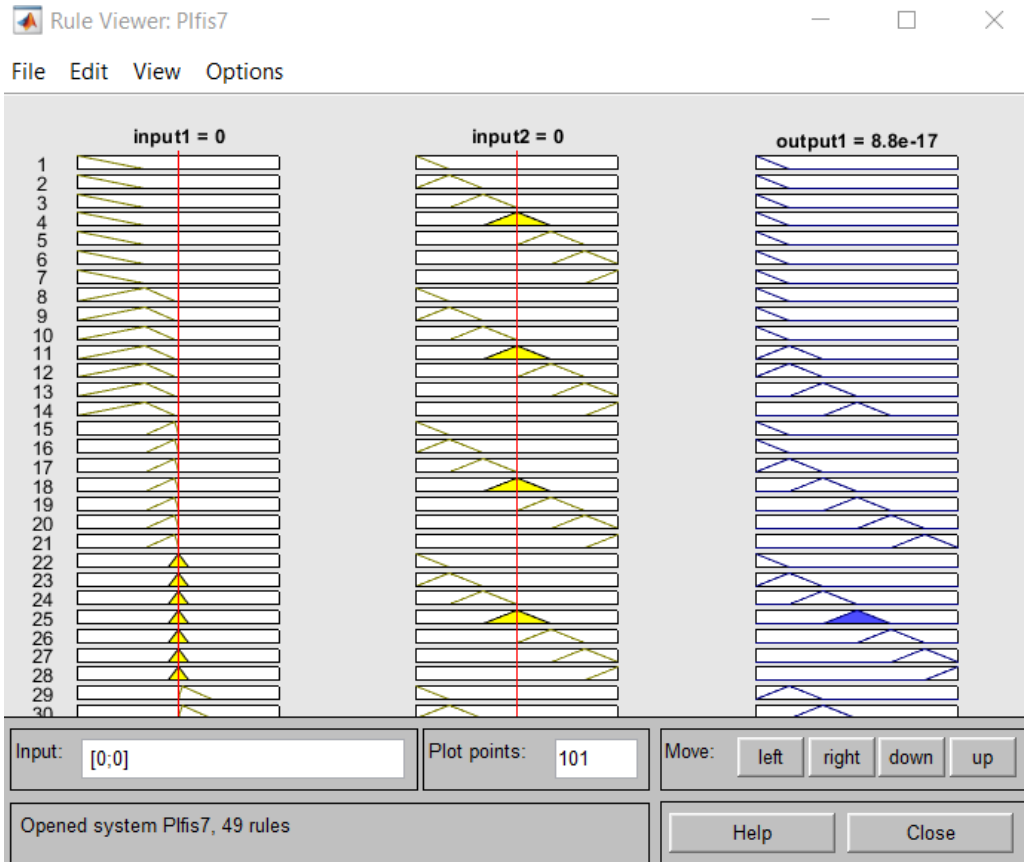


Рисунок 6.8 - Робота системи НЛР_ПІ

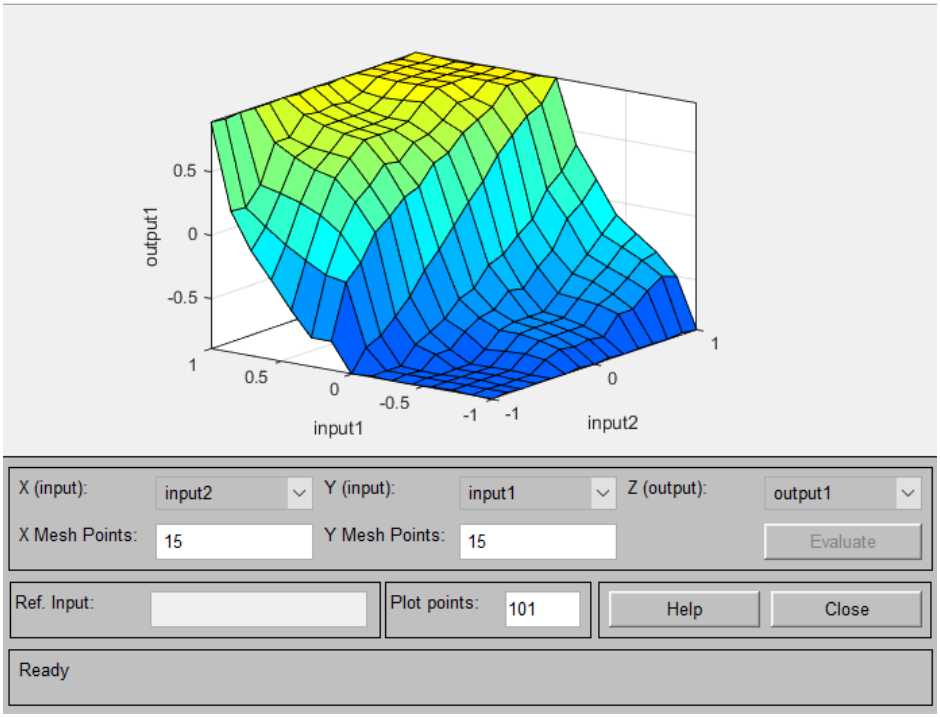


Рисунок 6.9 – Поверхня управління системою нечіткого висновку

Після експорту закону функціонування НЛР_ПІ на математичної моделі системи управління (див. рис. 6.2) получено перехідний процес для синтезованого НЛР_ПІ, який задовольняє поставленому завданню проектування

На рис. 6.10 показаний перехідний процес для синтезованого НЛР_ПІ.

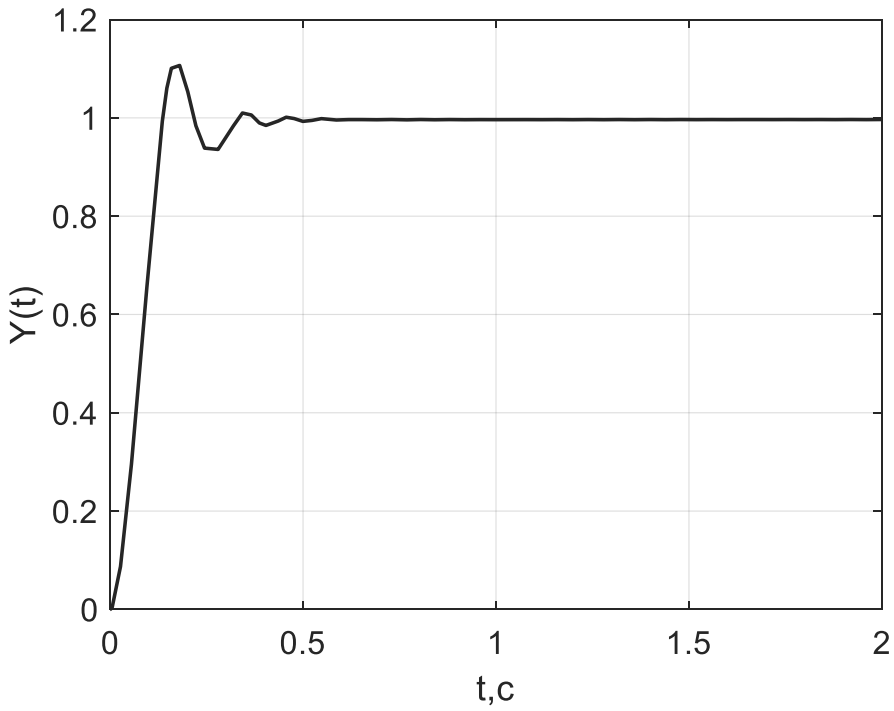


Рисунок 6.10 - Перехідний процес для НЛР_ПІ із сімома правилами

6.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 6.2.

За вказаними у табл. 6.2 потрібно синтезувати нелінійний нечіткий регулятор ПІ-типу з використанням системи нечіткого висновку:

Таблиця 6.2 - 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. 6.2

№ вар	*Передаточна 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	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

6.4 Контрольні питання

1. Викладіть евристичні міркування, що використовуються при синтезі НЛР_ПІ.
2. Який вид має таблиця лінгвістичних правил НЛР_ПІ?
3. Скільки кроків включає синтез нелінійного НЛР_ПІ-типа? Які це кроки?

7 СИНТЕЗ НЕЧІТКОГО РЕГУЛЯТОРА ПІД-ТИПУ

Мета роботи: засвоєння методики синтезу нелінійного нечіткого регулятору ПІД-типу з використанням редактора системи нечіткого висновку

7.1 Синтез нечіткого регулятора ПІД-типу на базі НЛР ПД та ПІ типу

Розглянемо реалізацію структури НЛР_ПІД з використанням синтезованих вище НЛР_ПД та НЛР_ПІ (див. рис. 7.1).

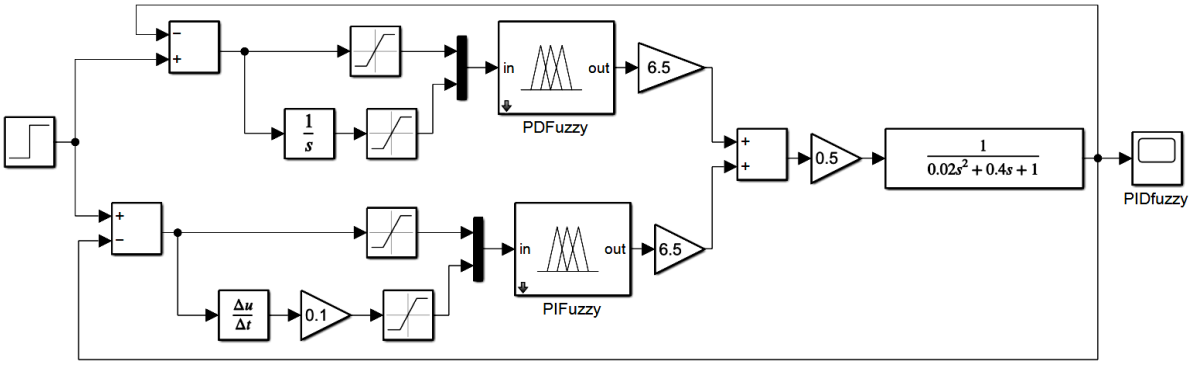


Рисунок 7.1 – Математична модель системи управління з НЛР_ПІД в Simulink MatLab

Налаштування НЛР_ПД та НЛР_ПІ здійснюється згідно методики викладеної вище

На рис. 7.2 показаний перехідний процес у системі.

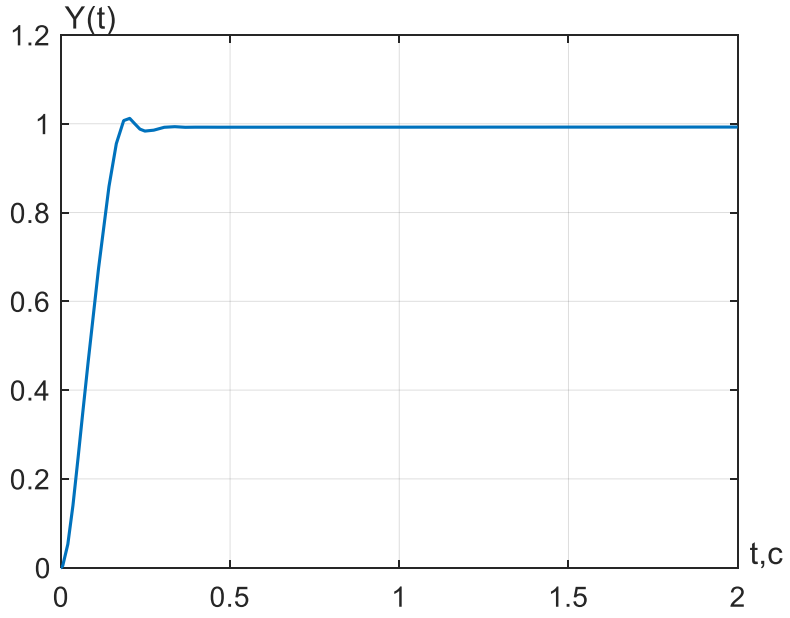


Рисунок 7.2 - Перехідний процес під керуванням НЛР_ПІД

7.2 Альтернативні варіанти синтезу НЛР_ПІД

Розглянемо далі реалізацію структури, яка наведена на рис. 7.3, тобто безпосередньо НЛР_ПІД, що отримує на вхід помилку, її похідну та інтеграл.

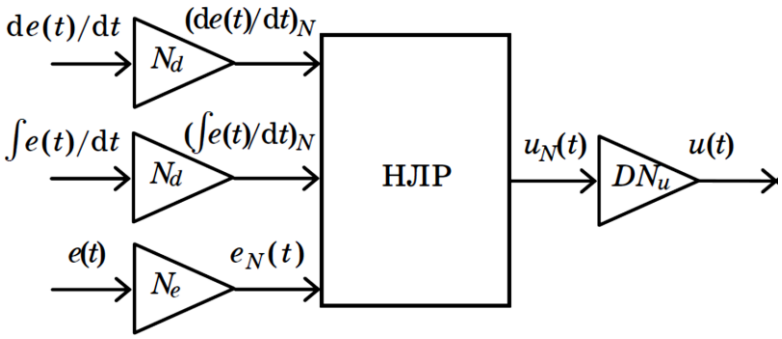


Рисунок 7.3 – НЛР ПІД-типу (НЛР_ПІД)

З метою мінімізації кількості правил будемо для вхідних змінних будемо використовувати по 3 терми (що дає 27 керуючих правил). При цьому кожної змінної відповідає власний масштаб базової шкали *x* (рис. 7.4).

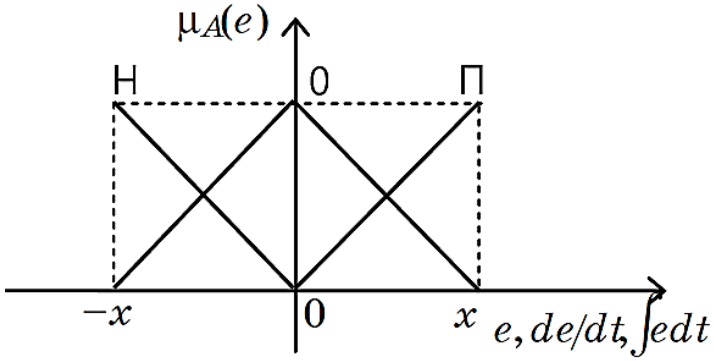


Рисунок 7.4 - Лінгвістичний опис вхідних змінний НЛР_ПІД

Для опису сигналу управління використовуватимемо 5 термів (рис. 7.5).

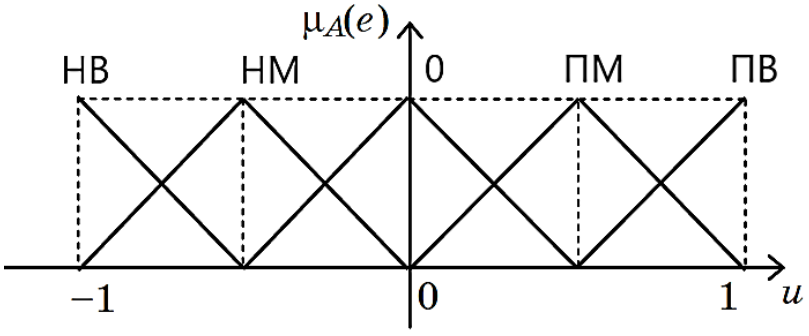


Рисунок 7.5 - Лінгвістичний опис вихідний змінної НЛР_ПІД

Розглянемо проекції поверхні управління НЛР _ПІД на площину $e \times de/dt$ за різних лінгвістичних значеннях $\int edt$.

Варіант 1: "інтеграл помилки" = "0". У цій ситуації коливання відбуваються в області нульової помилки і може бути використана симетрична таблиця лінгвістичних змінних (табл. 7.1).

Таблиця 7.1 – Правила НЛР_ПІД при нульовому інтегралі помилки

Таблиця правил		e*			
		Н	0	П	
Δ e*	Н	НВ	НМ	0	u*
	0	НМ	0	ПМ	
	П	0	ПМ	ПВ	

Варіант 2: "інтеграл помилки" = "П". У цій ситуації керована змінна може бути в області позитивної помилки, тоді потрібен сигнал корекції, величина якого пропорційна зростанню помилки. Якщо ж інтеграл помилки позитивний, а помилка негативна і зменшується, то корекція нульова (табл. 7.2).

Таблиця 7.2 - Правила НЛР_ПІД при позитивному інтегралі помилки

Таблиця правил		e*			
		Н	0	П	
Δ e*	Н	0	ПМ	ПВ	u*
	0	ПМ	ПВ	ПВ	
	П	ПВ	ПВ	ПВ	

Варіант 3: "інтеграл помилки" = "Н". Цей варіант виявляється симетричним до попереднього варіанту (табл. 7.3).

Таблиця 7.3 - Правила НЛР_ПІД при негативному інтегралі помилки

Таблиця правил		e*			
		Н	0	П	
Δ e*	Н	НВ	НВ	НВ	u*
	0	НВ	НВ	НМ	
	П	НВ	НВ	0	

Загальна таблиця правил набуває наступного вигляду (табл. 7.4, вона містить 27 правил, і кожне правило має три посилання.

Таблиця 7.4 – Правила управління НЛР_ПІД

№	$\int edt$	e	de/dt	u	№	$\int edt$	e	de/dt	u
1	0	Н	Н	НВ	15	П	0	П	ПВ
2	0	Н	0	НМ	16	П	П	0	ПВ
3	0	Н	П	0	17	П	П	Н	ПВ
4	0	0	Н	НМ	18	П	П	П	ПВ
5	0	0	0	0	19	Н	Н	0	НВ
6	0	0	П	ПМ	20	Н	Н	Н	НВ
7	0	П	Н	0	21	Н	Н	П	НВ
8	0	П	0	ПМ	22	Н	0	0	НВ
9	0	П	П	ПВ	23	Н	0	Н	НВ
10	П	Н	Н	0	24	Н	0	П	НИ
11	П	Н	0	ПМ	25	Н	П	0	НВ
12	П	Н	П	ПВ	26	Н	П	Н	НМ
13	П	0	Н	ПМ	27	Н	П	П	0
14	П	0	0	ПВ					

На рис. 7.6 показана структурна схема НЛР_ПІД Simulink MatLab, У схемі використаний інтегратор з насиченням, інакше НЛР_ПІД вироджується в НЛР_ПД.

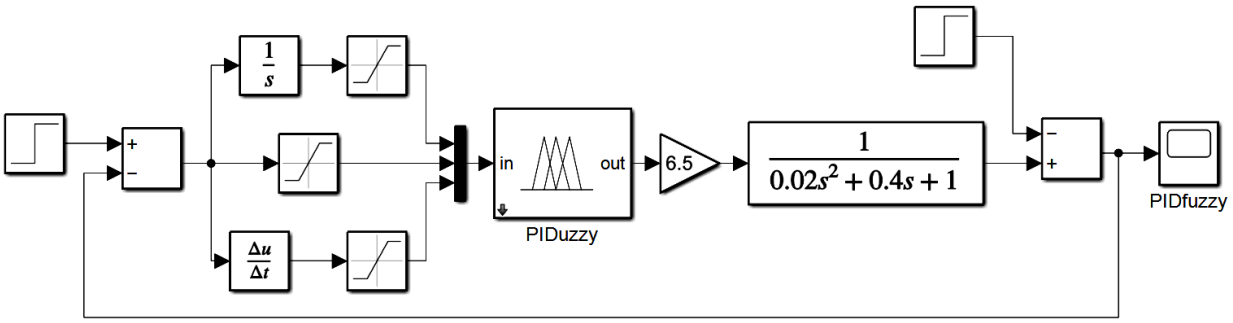


Рисунок 7.6 – Структурна схема математичної моделі об'єкта управління з НЛР_ПІД в Simulink MatLab

Відповідно до рис. 7.6 сформуємо в Simulink MatLab модель системи управління з НЛР ПІД при $x = 6.5$.

Fuzzy Logic Controller надаємо Fis name: PID2fis (ім'я може буди будь-яке).

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

>> fuzzy

Дале проводиться налаштовування головного вікна редактора нечіткої логічної системи (див. рис. 7.7.).

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*).

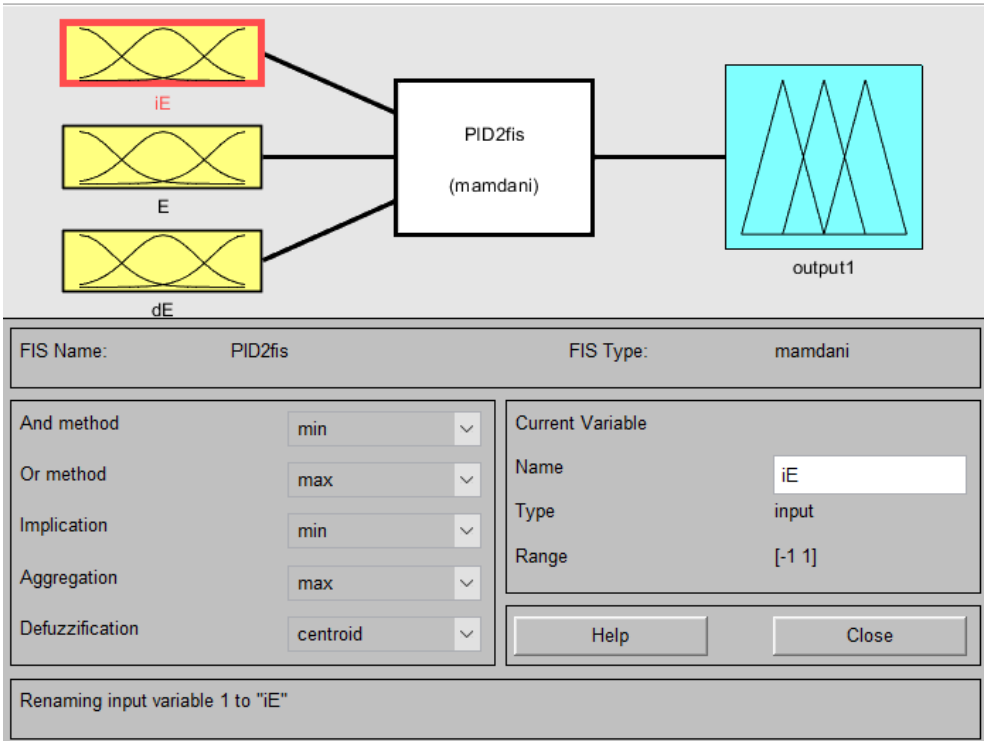


Рисунок 7.7 – Налаштування інтерфейсу FIS editor

У меню *Edit* послідовно додаємо 3 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.

Терми лінгвістичної змінної «Інтеграл помилки управління» розміщуються відповідно до рис. 7.8 з параметрами:

Name = "NiE"; Type = "trimf"; Param = [-2 -1 0];
Name = "ZiE"; Type = " trimf"; Param = [-1 0 1];
Name = "PiE"; Type = "trimf"; Param = [0 1 2].

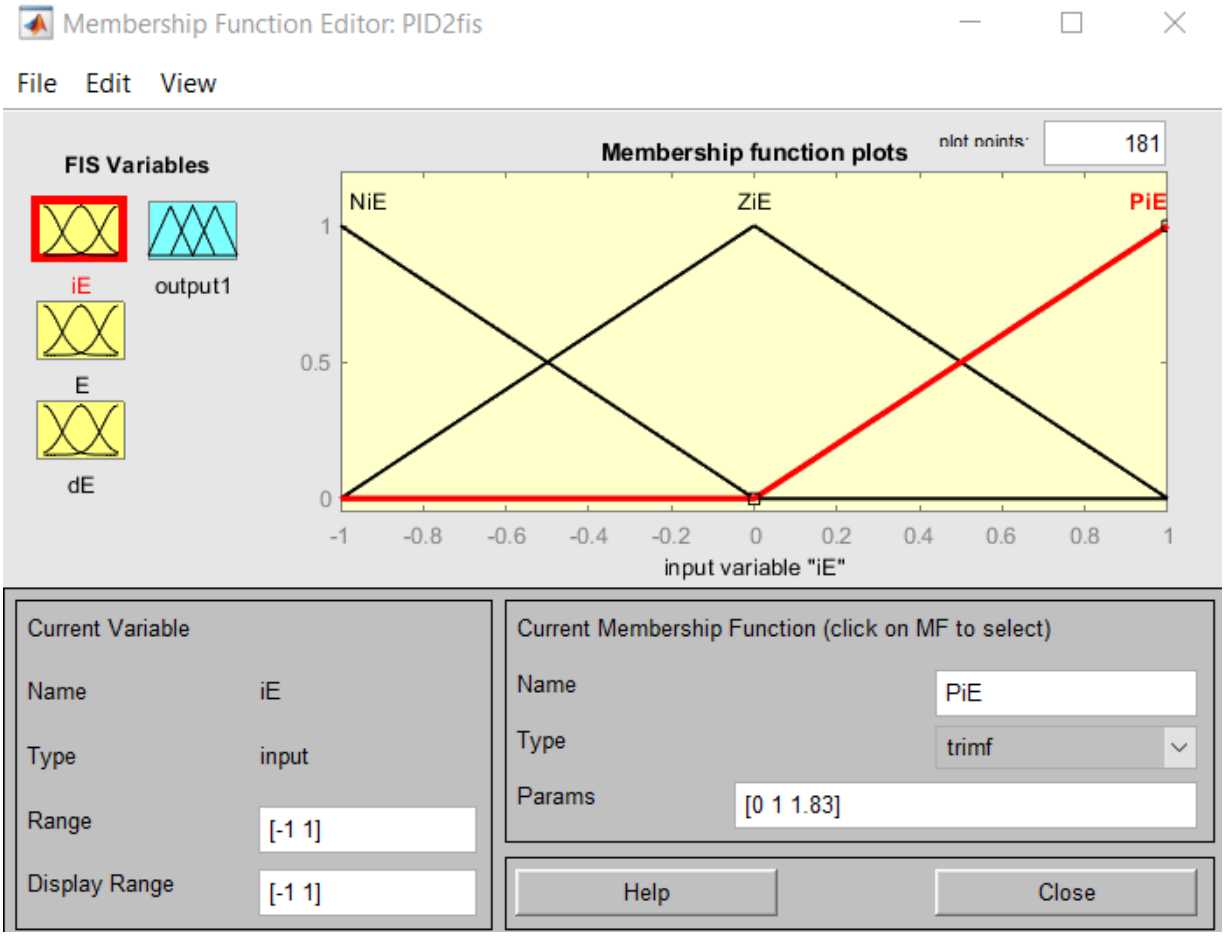


Рисунок 7.8 – Результат налаштування функцій приналежності вхідних змінних «Інтеграл помилки управління»

Терми вхідної лінгвістичної змінної «Помилка 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:

Name = "NE"; Type = "trimf"; Param = [-2 -1 0];
Name = "ZE"; Type = " trimf"; Param = [-1 0 1];
Name = "PE"; Type = "trimf"; Param = [0 1 2].

Результат налаштування функцій приналежності вхідних змінних «Помилка управління» на рис. 7.9.

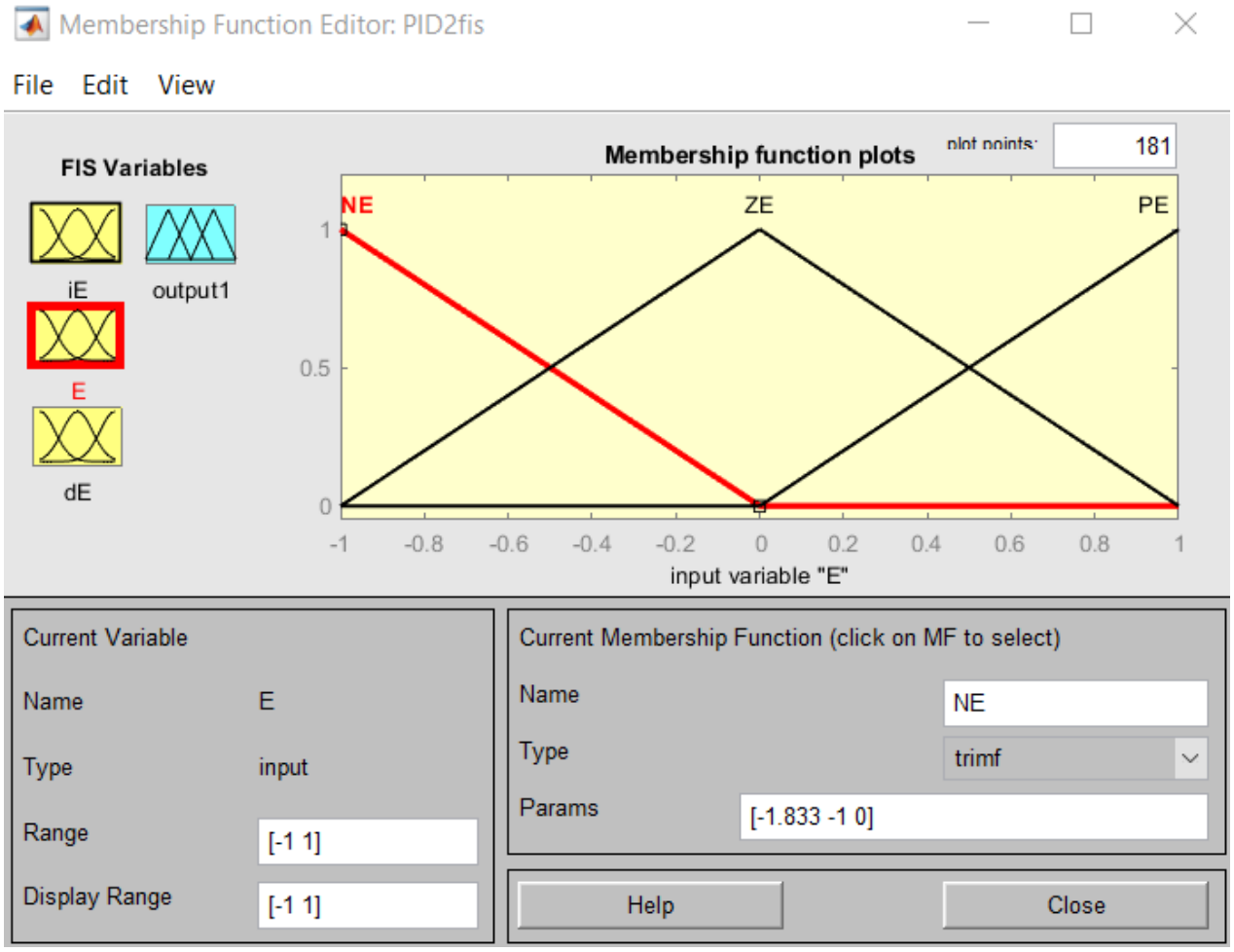


Рисунок 7.9 – Результат налаштування функцій приналежності вхідних змінних «Помилка управління»

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. З параметрами:

Name = "NdE"; Type = "trimf"; Param = [-2 -1 0];
Name = "ZdE"; Type = " trimf"; Param = [-1 0 1];
Name = "PdE"; Type = "trimf"; Param = [0 1 2].

Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління» на рис. 7.10.

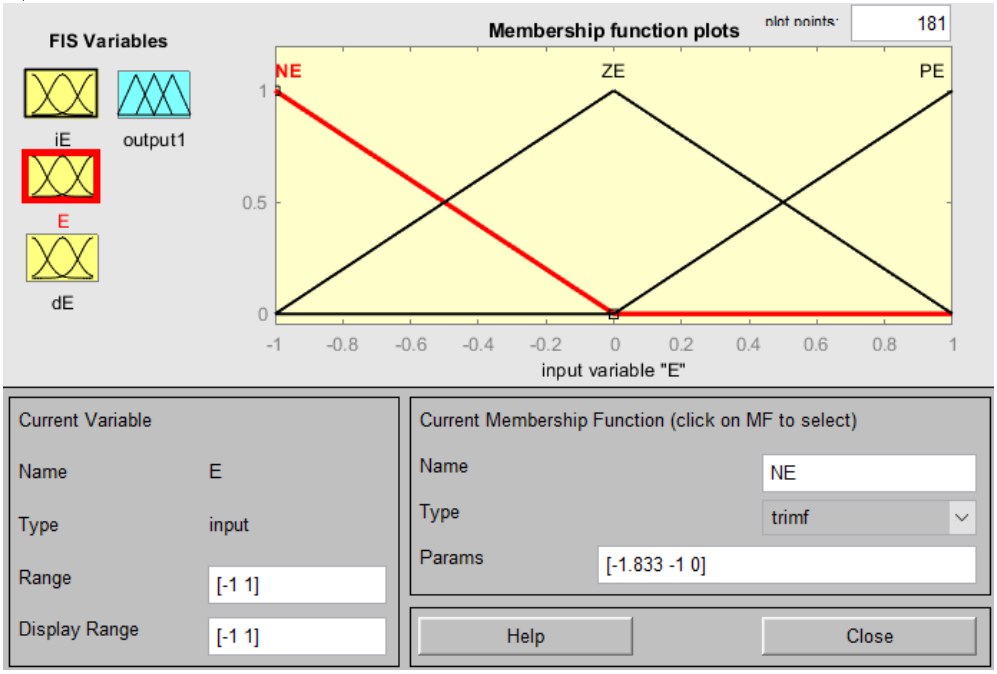


Рисунок 7.10 – Результат налаштування функцій приналежності вхідних змінних «Похідна помилки управління»

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 = [-1.5 -1 -0.5];
Name = "NSu"; Type = " trimf"; Param = [0 -0.5 0];
Name = "Zu"; Type = " trimf"; Param = [-0.5 0 0.5];
Name = "PSu"; Type = " trimf"; Param = [0 0.5 0.1];
Name = "PBu"; Type = "trimf"; Param = [0.5 1.5 1.5].

Результат налаштування функцій приналежності вхідних змінних наведено на рис. 7.11.

Після опису вхідних та вихідних лінгвістичних змінних, здійснюється опис правил у вікні редактора Rule Editor, кількість правил для даного випадку дорівнює 27. Управляючі правила сформуємо згідно з табл. 7.4.

1. if (iE is ZiE) and (E is NE) and (dE is NdE) then (output1 is NBu) (1)
2. if (iE is ZiE) and (E is NE) and (dE is ZdE) then (output1 is NSu) (1)
3. if (iE is ZiE) and (E is NE) and (dE is PdE) then (output1 is Zu) (1)
4. if (iE is ZiE) and (E is ZE) and (dE is NdE) then (output1 is NSu) (1)
5. if (iE is ZiE) and (E is ZE) and (dE is ZdE) then (output1 is Zu) (1)
6. if (iE is ZiE) and (E is ZE) and (dE is PdE) then (output1 is PSu) (1)
7. if (iE is ZiE) and (E is PE) and (dE is NdE) then (output1 is Zu) (1)
8. if (iE is ZiE) and (E is PE) and (dE is ZdE) then (output1 is PSu) (1)

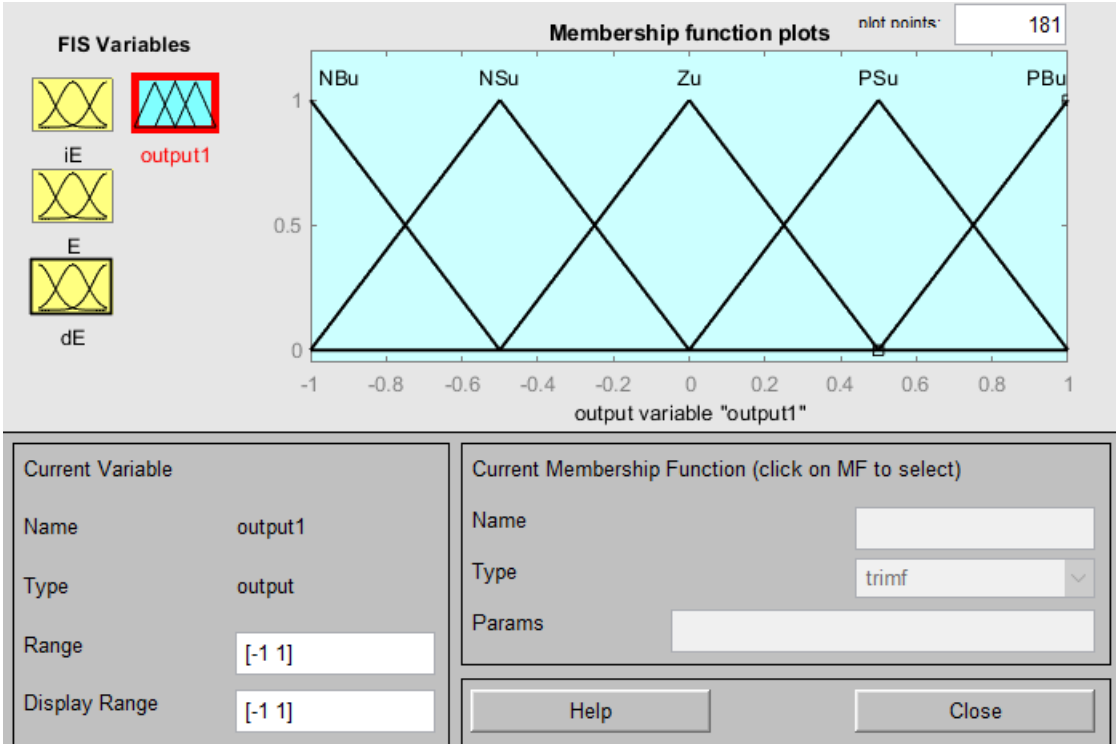


Рисунок 7.11 – Результат налаштування функцій приналежності вихідних змінних

9. if (iE is ZiE) and (E is PE) and (dE is PdE) then (output1 is PBu) (1)
10. if (iE is PiE) and (E is NE) and (dE is NdE) then (output1 is Zu) (1)
11. if (iE is PiE) and (E is NE) and (dE is ZdE) then (output1 is PSu) (1)
12. if (iE is PiE) and (E is NE) and (dE is PdE) then (output1 is PBu) (1)
13. if (iE is PiE) and (E is ZE) and (dE is NdE) then (output1 is PSu) (1)
14. if (iE is PiE) and (E is ZE) and (dE is ZdE) then (output1 is PBu) (1)
15. if (iE is PiE) and (E is ZE) and (dE is PdE) then (output1 is PBu) (1)
16. if (iE is PiE) and (E is PE) and (dE is NdE) then (output1 is PBu) (1)
17. if (iE is PiE) and (E is PE) and (dE is ZdE) then (output1 is PBu) (1)
18. if (iE is PiE) and (E is PE) and (dE is PdE) then (output1 is PBu) (1)
19. if (iE is NiE) and (E is NE) and (dE is NdE) then (output1 is NBu) (1)
20. if (iE is NiE) and (E is NE) and (dE is ZdE) then (output1 is NBu) (1)
21. if (iE is NiE) and (E is NE) and (dE is PdE) then (output1 is NBu) (1)
22. if (iE is NiE) and (E is ZE) and (dE is NdE) then (output1 is NBu) (1)
23. if (iE is NiE) and (E is ZE) and (dE is ZdE) then (output1 is NBu) (1)
24. if (iE is NiE) and (E is ZE) and (dE is PdE) then (output1 is NSu) (1)
25. if (iE is NiE) and (E is PE) and (dE is NdE) then (output1 is NBu) (1)
26. if (iE is NiE) and (E is PE) and (dE is ZdE) then (output1 is NSu) (1)
27. if (iE is NiE) and (E is PE) and (dE is PdE) then (output1 is Zu) (1)

Налаштовування управляючих правил для даного випадку наведено на рисунку (див. рис. 7.12).

Посилки у правилах пов'язані (connection) за допомогою операції

and. Введене правило забезпечене ваговим коефіцієнтом (*Weight*=1).

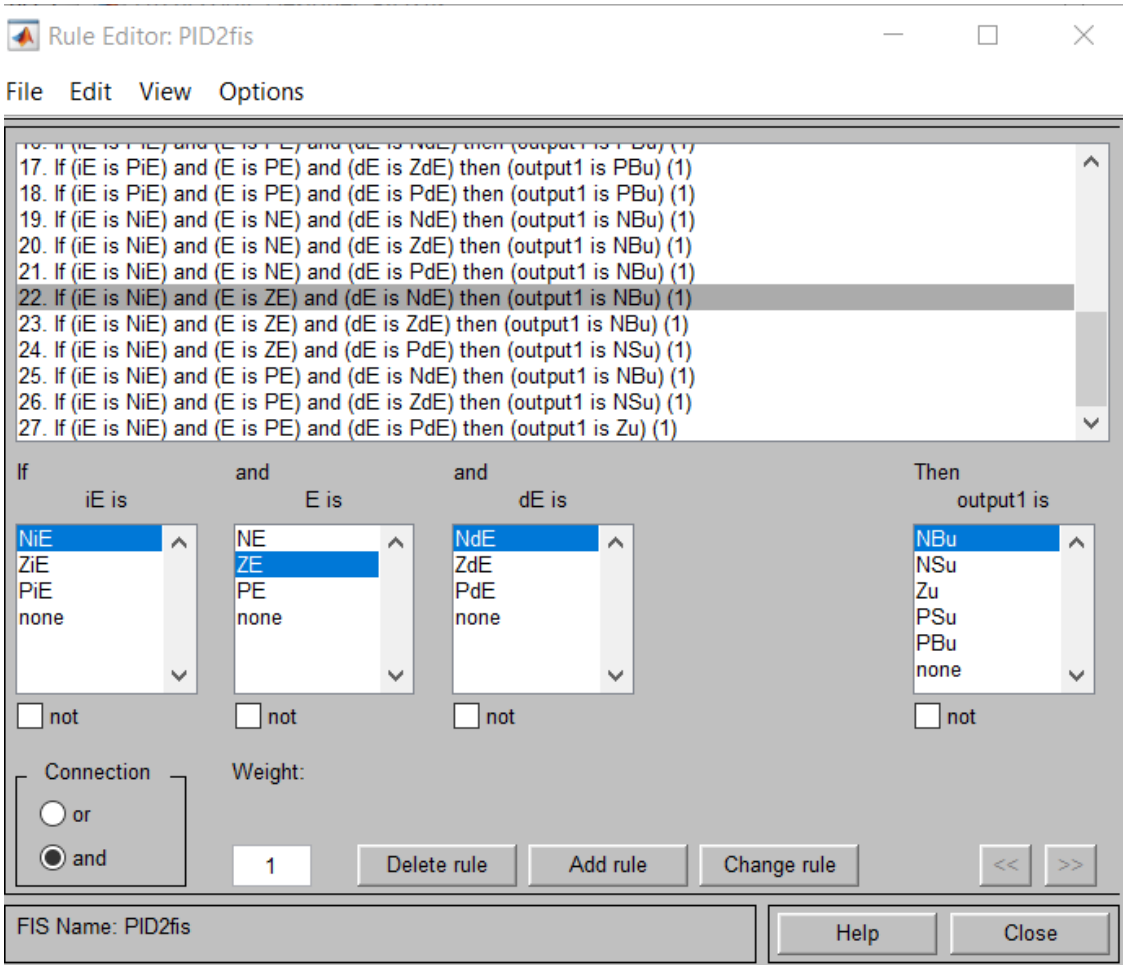


Рисунок 7.12 - Налаштовування опису правил функціонування НЛР_ПІД

Після опису правил можливо за допомогою

- інтерфейсу *Rule Viewer* можливо переглянути роботу системи нечіткого висновку за різних вхідних даних (див. рис. 7.13).

Після процедур налаштування 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.

Після експорту закону функціонування НЛР_ПІД на математичної моделі системи управління (див. рис. 7.6) получено перехідний процес для системи управління з НЛР_ПІД, який задовольняє поставленому завданню проектування

На рис. 7.14 показаний перехідний процес для синтезованого НЛР_ПІД з додавання збурення на 10с. Згідно графіку перехідного процесу можливо зробити висновок, що спроектована система управління повністю компенсує збурюючи впливи.

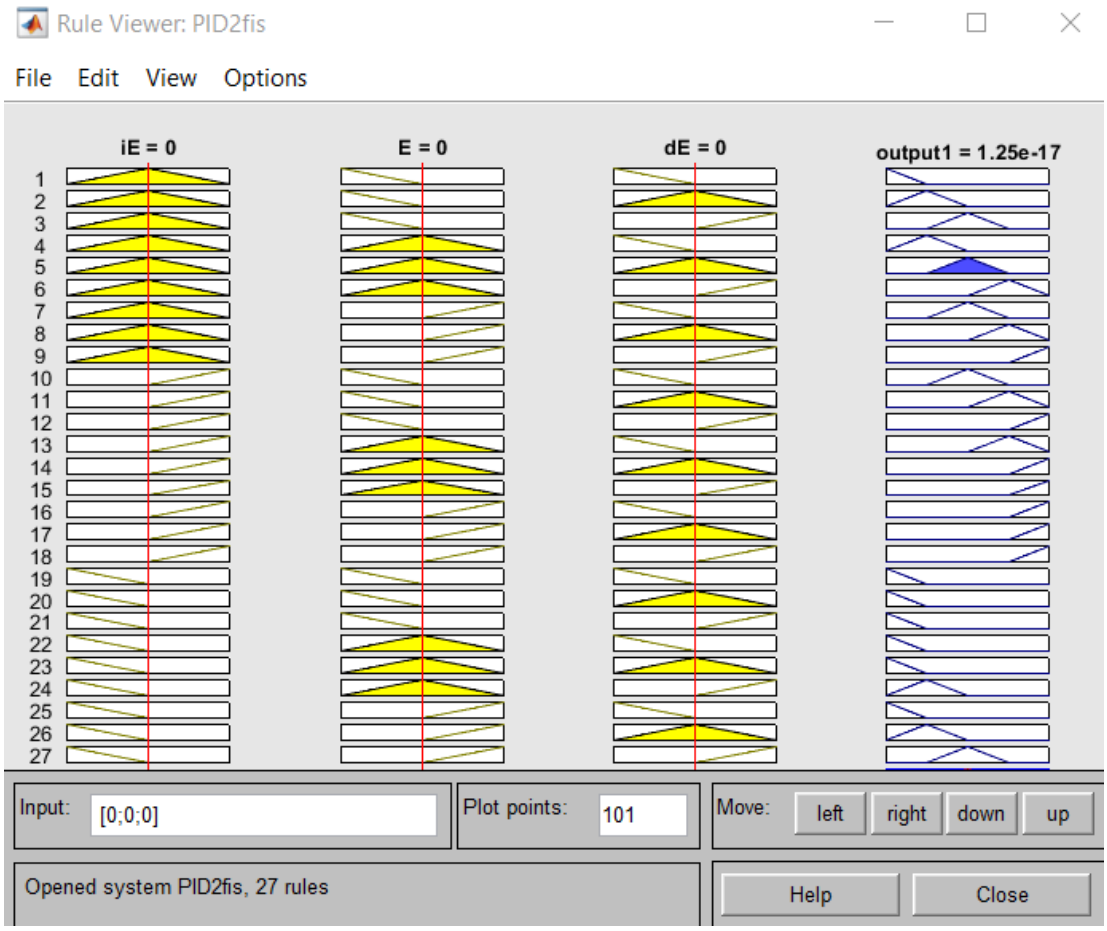


Рисунок 7.13 - Робота системи НЛР_ПІД

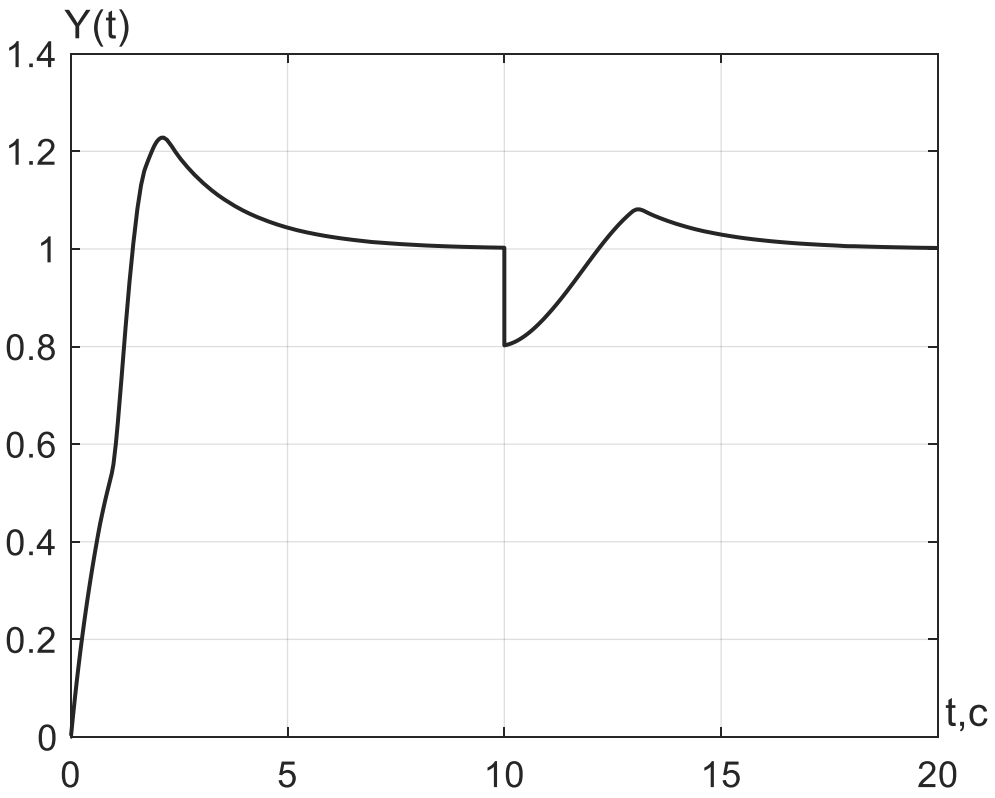


Рисунок 7.14 - Перехідний процес системи управління з НЛР_ПІД

7.3 Зміст звіту

Для виконання роботи студент отримують індивідуальні завдання, варіанти яких наведені у таблиці 7.5.

За вказаними у табл. 7.5 потрібно синтезувати нелінійний нечіткий регулятор ПІД-типу з використання методики:

1) Синтез нечіткого регулятора ПІД-типу на базі НЛР ПД та ПІ типу;

2) Синтез безпосереднього НЛР_ПІД, що отримує на вхід помилку, її похідну та інтеграл.

у режимі спостереження з використанням системи нечіткого висновку:

Таблиця 7.5 - 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	$W(p) = \frac{3}{1.5p^2 + 0.8p + 1}$

Продовження табл. 7.5

№ вар	*Передаточна функція об'єкту управління*
15	$W(p) = \frac{4.7}{5.6p^2 + 3p + 1}$
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	$W(p) = \frac{6}{7.2p^2 + 3.1p + 1}$

7.4 Контрольні питання

1. Опишіть структуру ПІД-регулятора.
2. Який вплив на перехідний процес мають різні коефіцієнти ПІД-регулятора?
3. Опишіть метод Зіглера-Ніколса для налаштування ПІД-регулятора.
4. Опишіть варіанти спрощення структури НЛР ПІД-типу

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.

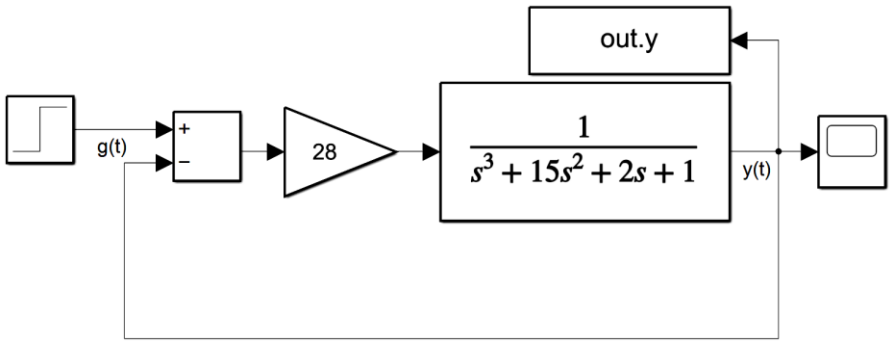
Схема моделювання у Simulink MatLab показана на рис. А.1.

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:

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

а)



б)

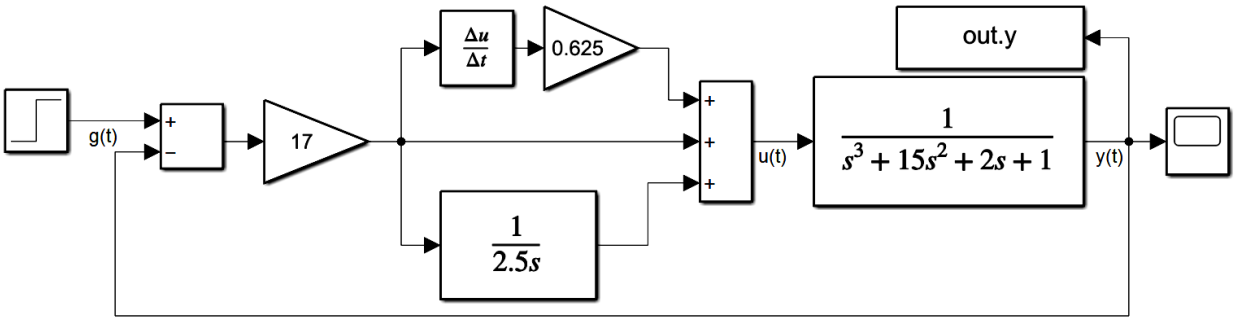


Рисунок А.1 – Математична модель роботи а) П-регулятора та б) ПІД-регулятора

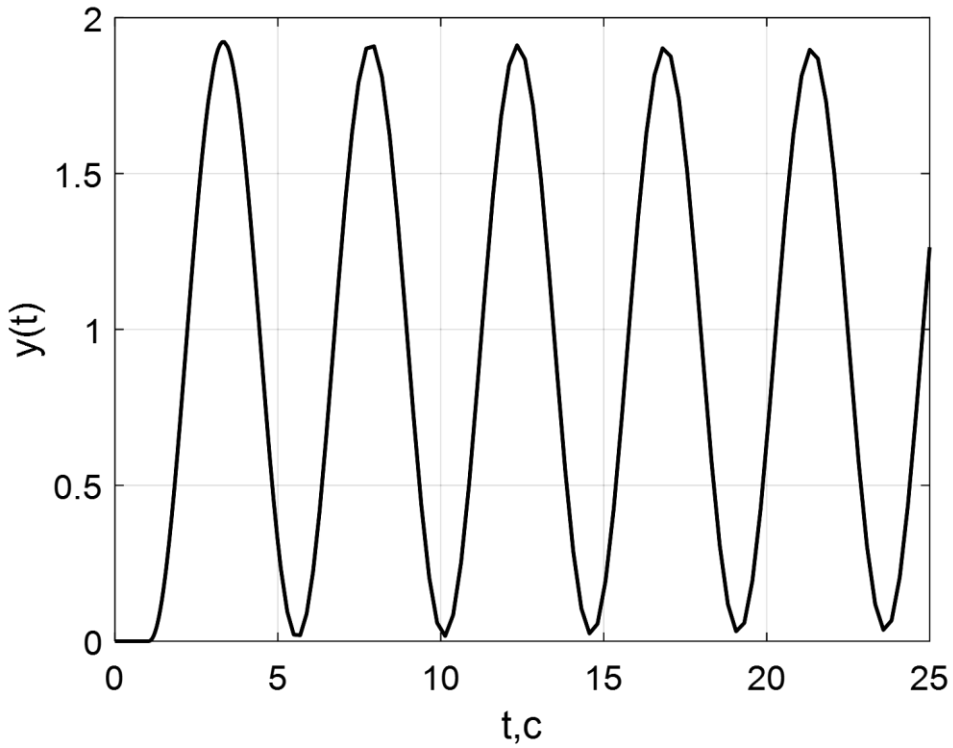


Рисунок А.2 - Незагасні коливання в системі управління

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

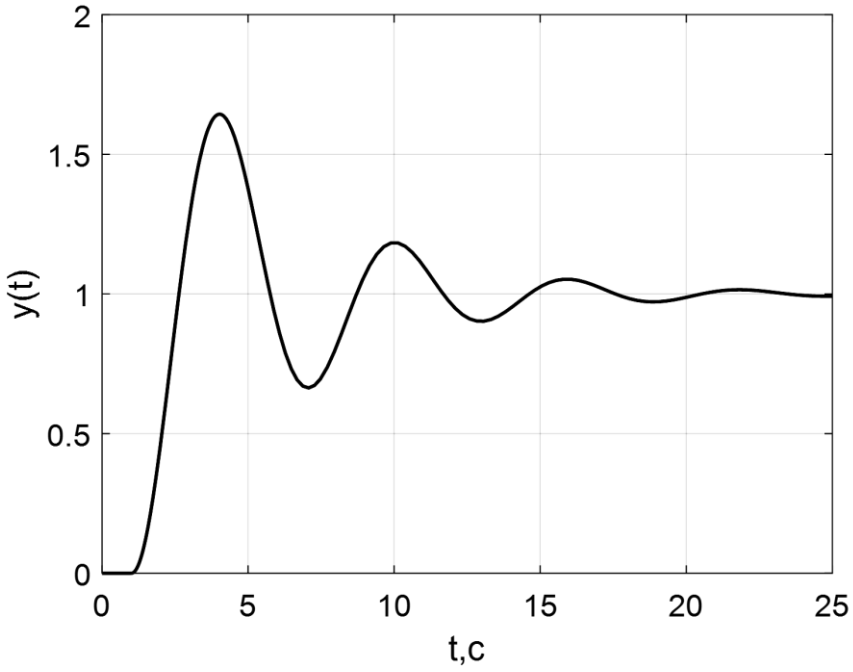


Рисунок А.3 – Перехідний процес у системі з ПІД-регулятором

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.

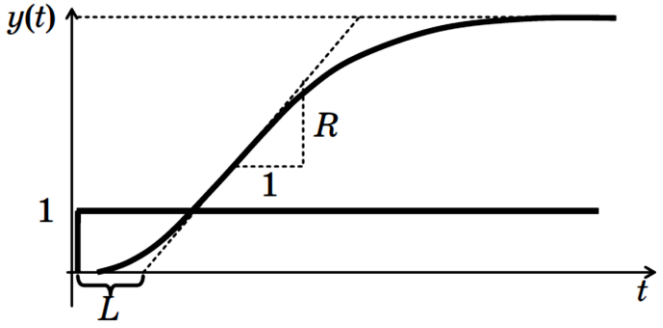


Рисунок А.4 – Визначення параметрів для стійкої системи

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.

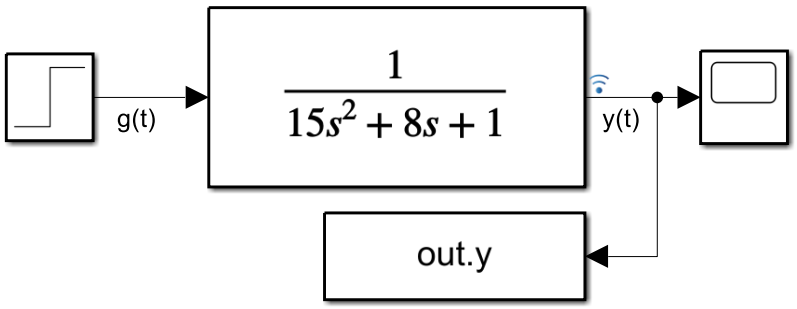


Рисунок А.5 – Математична модель об’єкта

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

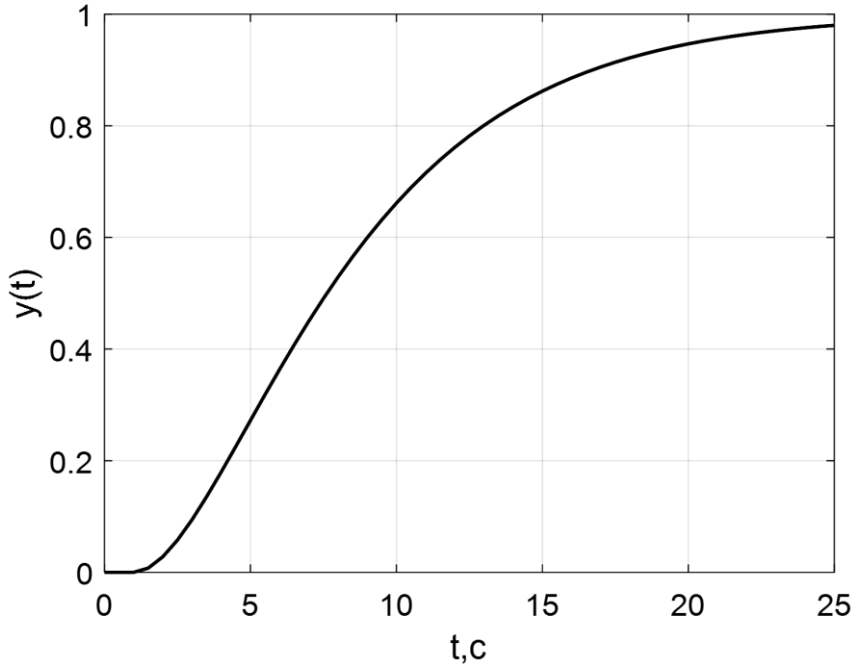


Рисунок А.6 – Графік перехідного процесу для системи без регулятора

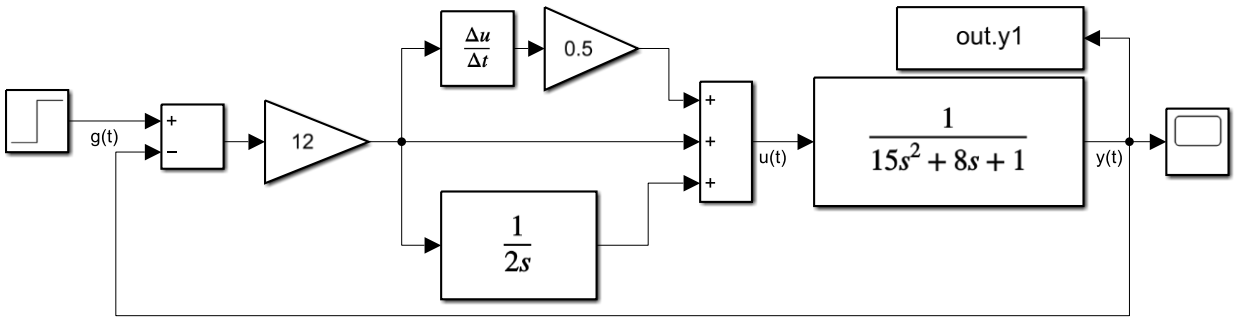
За кривою перехідного процесу, показаної на рис. А.6, визначаємо

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

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.

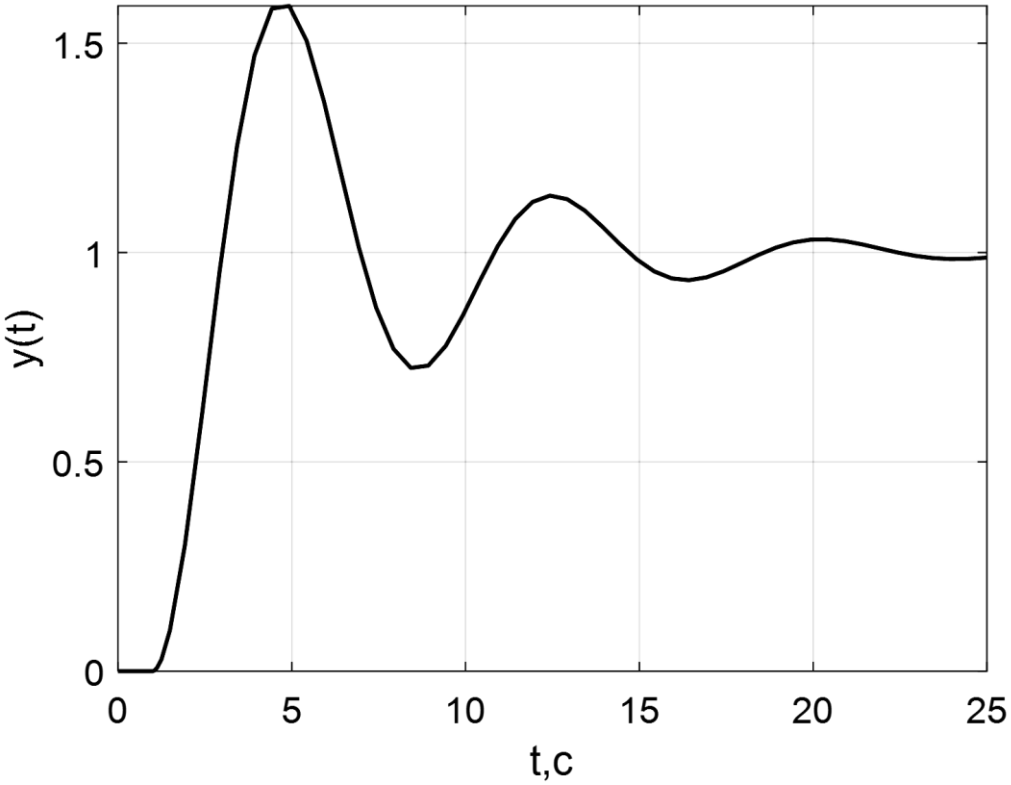


Рисунок А.7 – Графік перехідного процесу в замкненій системі з ПІД-регулятором (варіант 2)

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.