Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформаційно-керуючі системи

Інтелектуальні інформаційно-керуючі системи
Лабораторний практикум

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інтегровані інформаційні системи»
спеціальності 126 Інформаційні системи та технології

Укладачі: П. І. Кравець, В. М. Шимкович, Ю. М. Бердник

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК 681.3
І

Укладачі: *Кравець Петро Іванович*, канд. техн. наук, доц.
Шимкович Володимир Миколайович, канд. техн. наук, доц.
Бердник Юрій Михайлович

Рецензент *Писаренко А.В.*, науковий ступінь, вчене звання, місце роботи

Відповідальний редактор *Тимошин Ю.О.*, канд. техн. наук, доц.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 4 від 01.02.2024 р.)
за поданням вченої ради факультету інформатики та обчислювальної техніки
(протокол № 4 від 27.11.2023 р.)

Інформаційно-керуючі системи. Інтелектуальні інформаційно-керуючі системи.
І Лабораторний практикум [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Інтегровані інформаційні системи» спец. 126 Інформаційні системи та технології / КПІ ім. Ігоря Сікорського ; уклад.: П. І. Кравець, В. М. Шимкович, Ю. М. Бердник. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 166 с.

Навчальний посібник призначено для студентів, що навчаються за спеціальністю «126 Інформаційні системи та технології» кафедри інформаційних систем та технологій. В рекомендаціях викладені основи до виконання лабораторних робіт з освітнього компоненту «Інформаційно-керуючі системи».

УДК 681.3

Реєстр. № НП 23/24-231. Обсяг 8,3 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Сучасні досягнення інформаційних технологій проектування систем керування для різних класів об'єктів і технологічних процесів пов'язані з широким впровадженням в практику проектування засобів обчислювальної техніки, інструментальних систем та спеціальних мов і засобів програмування, що регламентуються міжнародними стандартами, а також впровадженням інтелектуальних систем керування, що базуються на знаннях[1-5]. Використання та подальша розробка таких технологій і систем керування потребує фахівців з відповідними знаннями в цій галузі.

Інтелектуальне керування є міждисциплінарною предметною областю, в якій тісно переплітаються завдання і методи їх вирішення, розроблені в теорії дослідження операцій, сучасній теорії керування складними динамічними об'єктами та теорії штучного інтелекту, що обумовлює внутрішню складність вирішення проблем у даній галузі, тому що в ній не тільки зберігаються проблеми наукових областей «донорів», а й з'являються нові невирішені проблеми, викликані синергетичним ефектом їх взаємодії[6-10].

Освітня компонента обіймає проблематику вивчення сучасного стану проектування систем керування, сучасних програмних засобів та мов програмування, ідентифікації нечітких та лінгвістичних моделей складних об’єктів керування, синтезу інтелектуальних пристроїв керування та регулювання складних об’єктів, проектування лінгвістичних пристроїв та баз знань, проектування нейромережевих систем керування. На сьогоднішній день за допомогою штучних нейронних мереж успішно вирішуються завдання керування в енергетиці, авіації, транспорті та робототехніці[11-16]. У складі таких систем штучна нейронна мережа може виконувати різні функції: діагностику технологічного обладнання, керування рухомими об'єктами і технологічними процесами, прогнозування ситуацій, оцінку стану і моніторинг технологічних процесів та багато іншого[17-20].

Нейронні мережі можуть бути використані для побудови регулюючих та коректуючих пристроїв, еталонної, адаптивної, номінальної та інверсно-динамічної моделей об'єкта, на основі яких виконується спостереження та оцінка параметрів об'єкта, спостереження та оцінка величини діючих в системі збурень, пошук або обчислення оптимальної програми зміни керуючого впливу, ідентифікація об'єкта, прогнозування стану об'єкта та інше[20-25].

Для практичного засвоєння навчальних матеріалів ряд матеріалів поглиблено вивчається на лабораторних заняттях, а також на прикладах систем керування інтелектуальних роботів і ряду технологічних процесів.

Освітня компонента «Інформаційно-керуючі системи» входить у комплекс наскрізної безперервної проектно-конструкторської підготовки спеціалістів з інтегрованих інформаційних систем і є частиною навчального плану 6-7-го учбового семестрів. Освітня компонента присвячена загальним питанням теорії та практики проектування систем керування основаних на знаннях – інтелектуальних систем керування. В структурно-логічній схемі навчання освітня компонента «Інформаційно-керуючі системи» входить в цикл освітніх компонентів «Проектування інформаційних систем» і вивчається на етапі підготовки фахівців освітньо-кваліфікаційного рівня бакалавр. Вивчення освітньої компоненти базується на знаннях, навичках та уміннях, отриманих під час вивчення студентами дисциплін математичного циклу, теорії систем, моделювання процесів та систем, алгоритмічного та програмного забезпечення, комп'ютерної техніки та технологій, теорії автоматичного керування та інших.

Метою вивчення освітньої компоненти є формування у майбутніх фахівців знань та вмінь застосування сучасних методів та засобів проектування та розробки інтелектуальних систем керування та одержання навиків використання таких систем своїй практичній роботі.

ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Навчальний посібник призначений для виконання лабораторних робіт з освітньої компоненти «Інформаційно-керуючі системи». Метою освітньої компоненти є формування та закріплення у студентів наступних компетентностей відповідно освітній програмі: (КС2) - Здатність застосовувати стандарти в області інформаційних систем та технологій при розробці функціональних профілів, побудові та інтеграції систем, продуктів, сервісів і елементів інфраструктури організації; (КС3) - Здатність до проектування, розробки, налагодження та вдосконалення системного, комунікаційного та програмно-апаратного забезпечення інформаційних систем та технологій, Інтернету речей (IoT), комп'ютерно-інтегрованих систем та системної мережної структури, управління ними; (КС16) - Здатність інтегрувати програмні, технічні, інформаційні та інтелектуальні компоненти усіх рівнів ієрархії інформаційно-керуючих систем в єдину розподілену систему; (КС18) - Здатність вирішувати задачі інтеграційних процесів інформаційних систем у сфери виробництва та керування з використанням методів аналізу та синтезу засобів передачі, зберігання та обробки інформації, основ сервіс-орієнтованого підходу до обслуговування користувачів інформаційних систем, базових та прикладних інформаційних технологій та інструментальних засобів інфраструктури IT.

Навчальний посібник складаються з 15 тем лабораторних робіт де зазначена тема лабораторної роботи, мета лабораторної роботи, систематизовані теоретичні матеріали по темі лабораторної роботи, завдання та порядок виконання лабораторної роботи, приклади виконання, методики розрахунку та рекомендована література.

Базами для самостійної роботи з навчальним посібником служать освітня компонента «Теорія автоматичного керування» і лекції з освітньої компоненти «Інформаційно-керуючі системи». В результаті самостійної роботи над ними студенти отримають знання з побудови статичних та

динамічних моделей об'єктів керування, основних типів регуляторів, будуть знати математичні рівняння, що описують роботу цих регуляторів, їх передавальні функції і перехідні характеристики. Вивчать структуру регуляторів і їх основні елементи, навчаться реалізувати структуру регулятора з необхідними динамічними характеристиками, набудуть практичних навичок з розрахунку оптимальних параметрів налаштування регуляторів відповідно до заданих критеріїв якості роботи системи регулювання. Також в даному лабораторному практикумі студенти здобудуть початковий досвід в інтелектуальних системах керування, а саме в нечітких системах, нейромережевих системах та генетичних алгоритмах.

При вивченні конкретної теми необхідно ознайомитися з темою і метою лабораторної роботи, лекційним матеріалом за темою, короткими теоретичними відомостями в лабораторній роботі, поставленим завданням, порядком виконання та прикладом виконання.

Лабораторний практикум відпрацьовується кожним студентом групи індивідуально, кожний студент отримує індивідуальне завдання на лабораторну роботу.

Перед початком виконання лабораторної роботи студенти мають самостійно (до початку занять) ознайомитись з методичними рекомендаціями на поточну роботу, засвоїти теоретичні відомості, порядок виконання та свій варіант до цієї роботи.

Кожна лабораторна робота оформляється у вигляді звітного документу. Звіт по роботі формулюється на основі протоколу, який ведеться під час виконання поточної роботи та результатів теоретичної підготовки.

Звіт має містити наступні розділи:

– стандартний титульний лист (із назвами: ЗВО, факультету, кафедри; темою та номером поточної роботи; номером групи та П.І.Б. виконавця; П.І.Б. викладача що перевірятиме роботу);

– назва та мета роботи;

– завдання до роботи;

– початкові дані варіанту досліджень;

– тези теоретичних відомостей (опорні формули, визначення);

– проміжні математичні розрахунки, математичні моделі систем;

– результати моделювання у вигляді отриманих параметрів коефіцієнтів, функцій, графіків, гістограм, таблиць тощо;

– порівняння теоретичних та практичних результатів;

– обов'язково висновки про результати досліджень, в висновках охарактеризувати отриманий результат в лабораторній роботі;

ЛАБОРАТОРНА РОБОТА №1

Моделювання об'єкта керування з двома входами і одним виходом засобами нечіткої математики

Мета роботи: Засобами моделювання нечіткої логіки отримати модель функції двох змінних за варіантом. Провести дослідження форми функції приналежності на якість моделювання, а також залежність кількості правил і якості моделювання.

Короткі теоретичні відомості

Властивістю людського інтелекту є здатність приймати правильні рішення в умовах неповної та нечіткої інформації. Побудова моделей, які наближено відтворюють роздуми людини, та їх використання у комп'ютерних системах становлять одну з найважливіших проблем науки .

Першим серйозним кроком у цьому напрямку стала **теорія нечітких множин**, розроблена американським ученим Лотфі Заде (Lotfi Zadeh). Його робота «Fuzzy Sets» [26], опублікована в журналі «Information and Control» у 1965 році, заклала основи моделювання інтелектуальної діяльності людини та послужила початковим поштовхом для розвитку нової математичної теорії. Він також вперше використав термін «fuzzy logic» (fuzzy – нечіткий, розмитий, м'який) для позначення цієї нової області науки.

Нечітка множина. Нехай E – універсальна множина, x – елемент E, а R – певна властивість. Звичайна (чітка) підмножина A універсальної множини E, елементи якої задовольняють властивості R, визначається як множина впорядкованої пари $A = \{\mu A\ (x)/x\}$, де $\mu A(x)$ – характеристична функція, що приймає значення 1, якщо x задовольняє властивості R, і 0 – в іншому випадку.

Нечітка підмножина відрізняється від звичайної тим, що для елементів x з E немає однозначної відповіді приналежності відносно властивості R. У зв'язку з цим, нечітка підмножина A універсальної множини E визначається як множина впорядкованої пари $A = \{\mu_A(x)/x\}$, де $\mu_A(x)$ – характеристична

функція приналежності (або просто функція приналежності), що приймає значення в деякій впорядкованій множині M (наприклад, $M = [0,1]$).

Функція приналежності $\mu_A(x)$ вказує ступінь (або рівень) приналежності елемента x до підмножини A. Множина M називають множиною приналежностей. Якщо $M = \{0,1\}$, тоді нечітка підмножина A може розглядатися як звичайна або чітка множина.

Найбільш важливим застосуванням теорії нечітких множин є контролери нечіткої логіки для систем керування та прийняття рішень. Їх функціонування дещо відрізняється від роботи звичайних контролерів. Для опису системи замість диференційних рівнянь використовуються знання експертів. Ці знання виражені за допомогою лінгвістичних змінних, що описані нечіткими множинами та базою знань.

Загальна структура контролера, що використовує нечітку логіку, показана на рисунку 1.1. Вона містить у своєму складі наступні складові: блок фазифікації; базу знань; блок рішень; блок дефазифікації.

Блок фазифікації перетворює чіткі величини, виміряні на виході об'єкта керування, у нечіткі величини, що описані лінгвістичними змінними в базі знань.

Блок рішень використовує нечіткі умовні (*if - then*) правила, закладені в базі знань, для перетворення нечітких вхідних даних у необхідні керуючі впливи, що носять також нечіткий характер.

Блок дефазифікації перетворює нечіткі дані з виходу блоку рішень у чітку величину, що використовується для керування об'єктом.

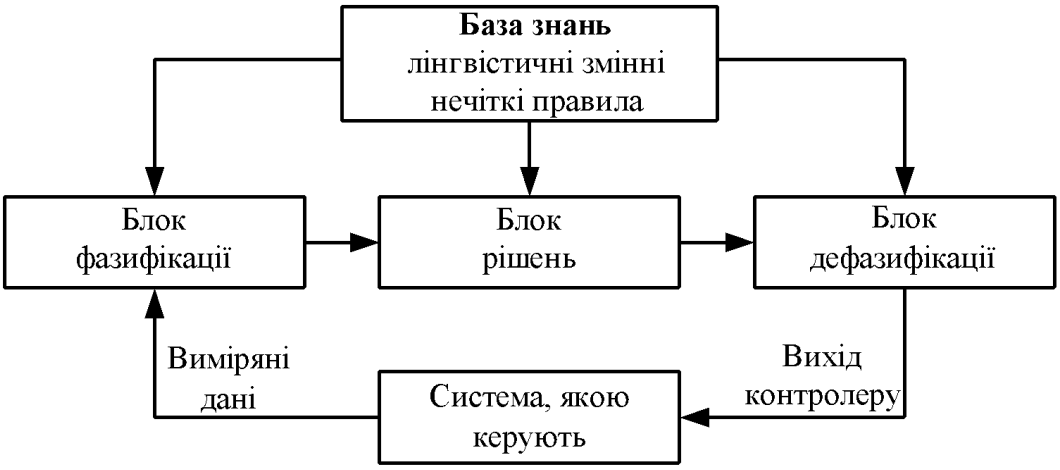

Рисунок 1.1 – Загальна структура нечіткого контролера

Як приклад відомих мікроконтролерів, що підтримують нечітку логіку можна назвати 68HC11, 68HC12 фірми Motorola, MCS-96 фірми Intel, а також деякі інші.

Всі системи нечіткого виводу функціонують за одним принципом. Показання вимірювальних приладів фазифікуються (перетворюються в нечіткий формат), обробляються, дефазифікуються й у вигляді звичайних сигналів подаються на виконавчі пристрої.

Розглянемо приклад керування мобільним роботом, задачею якого є об'їзд перешкод. Введемо дві лінгвістичні змінні: ДИСТАНЦІЯ (відстань від робота до перешкоди) і НАПРЯМОК (кут між поздовжньою віссю робота та напрямком на перешкоду).

Розглянемо лінгвістичну змінну ДИСТАНЦІЯ. Значеннями її можна визначити терми ДАЛЕКО, СЕРЕДНЄ, БЛИЗЬКО і ДУЖЕ БЛИЗЬКО. Для фізичної реалізації лінгвістичної змінної необхідно визначити точні фізичні значення термів цієї змінної. Нехай змінна ДИСТАНЦІЯ може приймати будь-які значення з діапазону від нуля до нескінченності. Відповідно до теорії нечітких множин, у такому випадку кожному значенню відстані з зазначеного діапазону може бути поставлене у відповідність деяке число від нуля до одиниці, що визначає ступінь приналежності даної фізичної відстані (припустимо 40 см) до того чи іншого терму лінгвістичної змінної

ДИСТАНЦІЯ. Ступінь приналежності визначаємо функцією приналежності M(d), де d-відстань до перешкоди. У нашому випадку відстані 40 см. Можна задати ступінь приналежності до терму ДУЖЕ БЛИЗЬКО рівним 0.7, а до терму БЛИЗЬКО – 0.3 (рисунок 1.2). Конкретне визначення ступеня приналежності може проходити тільки при роботі з експертами.

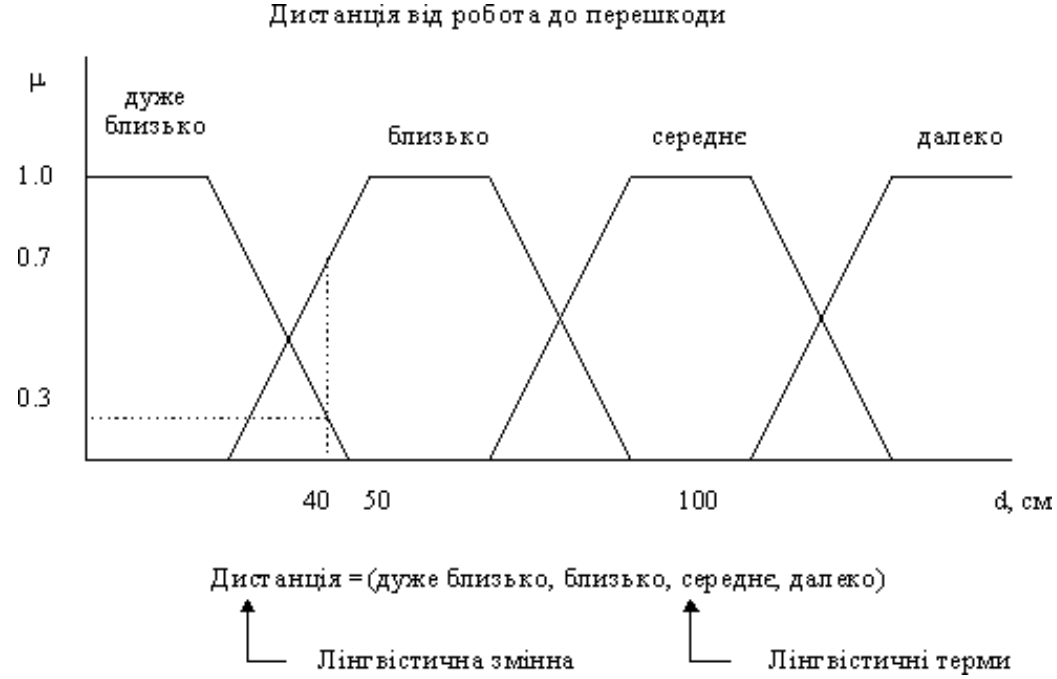

Рисунок 1.2 – Лінгвістична змінна та лінгвістичні терми

Змінній НАПРЯМОК, яка може приймати значення в діапазоні від 0 до 360 градусів, задамо терми ЛІВИЙ, ПРЯМИЙ і ПРАВИЙ.

Тепер необхідно задати вихідні змінні. У даному прикладі достатньо однієї, яка назвемо РУЛЬОВИЙ КУТ. Вона може містити терми: РІЗКО ВЛІВО, ВЛІВО, ПРЯМО, ВПРАВО, РІЗКО ВПРАВО. Зв'язок між входом та виходом запам'ятовується в таблиці нечітких правил.

Таблиця 1.1 – Нечіткі правила

напрямок \ дистанція	дуже близько	близько	середнє	далеко
правий	різко вліво	різко вліво	вліво	прямо
прямий	різко вліво	вліво	вліво	прямо
лівий	різко вправо	різко вправо	вправо	прямо

Кожний запис у даній таблиці відповідає своєму нечіткому правилу, наприклад: Якщо дистанція близько і напрямок правий , тоді рульовий кут різко вліво.

Таким чином, мобільний робот з нечіткою логікою буде працювати за наступним принципом: дані з сенсорів про відстань до перешкоди та напрямок до неї будуть фазіфіковані, оброблені згідно табличних правил, дефазіфіковані і дані, що отримані, у вигляді керуючих сигналів надходять на приводи робота.

Fuzzy logic toolbox – вбудована в Matlab сукупність функцій, що містить набір засобів, які дозволяють: створювати і редагувати нечіткі системи всередині середовища Matlab; вбудовувати нечітку підсистему в SimuLink (поставляється з Matlab) при моделюванні загальної системи; побудувати нечітку систему в Matlab у вигляді процедури, що викликається з програми, яка написана на мові Сі.

Даний набір інструментів забезпечує три категорії інструментальних засобів програмування нечітких систем: функції командного рядка (command line functions); графічний інтерактивний інтерфейс; використання вбудованих блоків SimuLink.

Перша категорія – готові функції, які можна викликати відразу з командного рядка Matlab. Практично усі вони являють собою м-файли, що

містять послідовність виразів, що виконують спеціалізований нечіткий алгоритм.

Крім того, Matlab дозволяє їх модифікувати шляхом копіювання і перейменування відповідного файлу та наступного його редагування. Таким чином, нечіткий набір інструментів є розширеним власними функціями.

Друга категорія дозволяє отримати доступ до тих самих функцій через графічний користувальницький інтерфейс, за допомогою якого набагато зручніше конструювати й аналізувати нечіткі системи.

Третя категорія – моделювання в середовищі SimuLink. Тут підсистеми представляються у виді блоків – можна з'єднати будь-яким чином і відразу отримати результати.

У Matlab є багато вбудованих функцій приналежності, зокрема: сигмоїдальна; двостороння сигмоїдальна; гаусова; дзвоноподібної форми; S-функція приналежності; Z-функція приналежності; трапецієподібна; трикутна й ін.

Усі дії над нечіткими числами задаються мінімальним набором функцій і відбуваються всередині програми. Таким чином, користувачу необов'язково вивчати усі тонкощі теорії нечітких множин, достатньо лише визначити усі вхідні і вихідні змінні і задати таблицю правил, а решту роботи робить Matlab. Дефазіфікація виконується в один з п'ятьох методів, зазначених програмістом. Крім того, можна вивести на екран відповідно до введених правил результуючі поверхні керування в залежності від комбінації входів, схему отриманої нечіткої програми, і це лише мала частина всіх можливостей даного набору інструментів.

Порядок виконання роботи:

Варіант завдання на дану лабораторну роботу вибираються з додатка Б за номером групи та номером студента в списку групи.

Для запуску тулбоксу «Fuzzy Logic» необхідно набрати в «Command Window» MatLab команду: *fuzzy*.

Як результат – з'явиться вікно, зображене на рисунку 1.3:

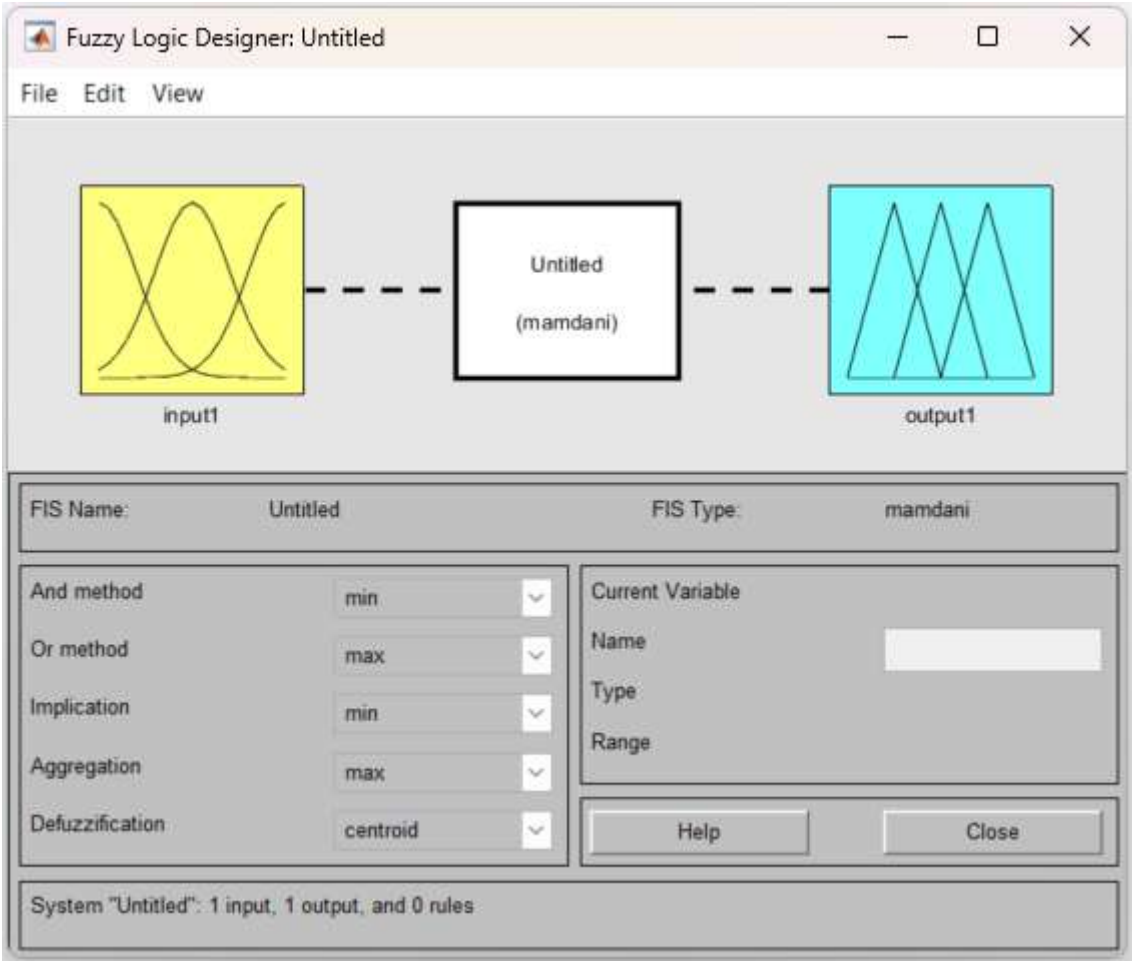

Рисунок 1.3 – Нечітка модель представлена фізифікатором (входи системи), базою знань і дефазифікатором (виходи системи)

Кількість входів і виходів обирають в залежності від задачі, що розв'язують. Щоб додати вхід – необхідно обрати: Edit->Add Variable->Input.

Виділивши змінну, можна задати її ім'я, що показано на рисунку 1.4.

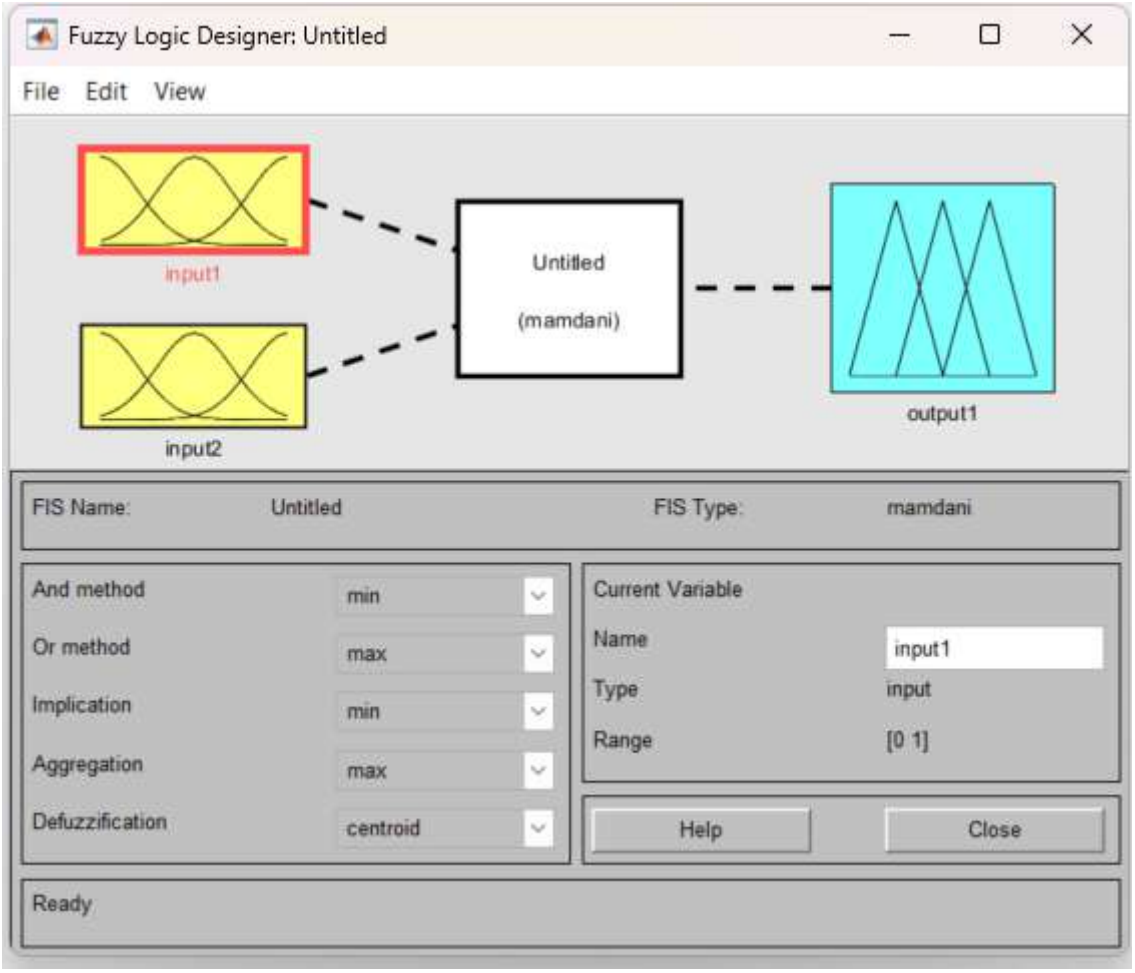

Рисунок 1.4 – Надання імені змінній

Щоб відкрити властивості змінної – необхідно виконати подвійне натискання мишкою на змінній. Це зображено на рисунку 1.5.

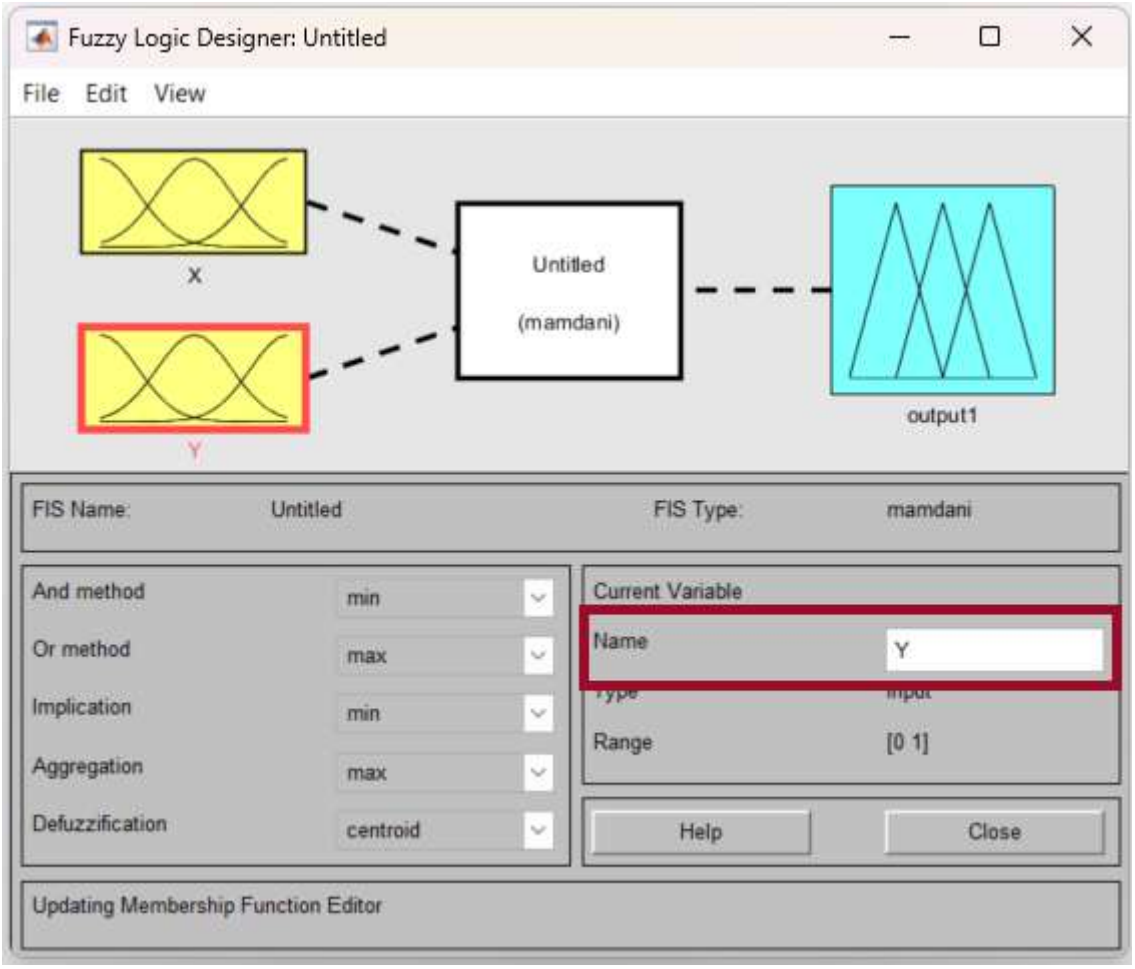

Рисунок 1.5 – Відкриття властивостей змінної шляхом подвійного клацання по її імені

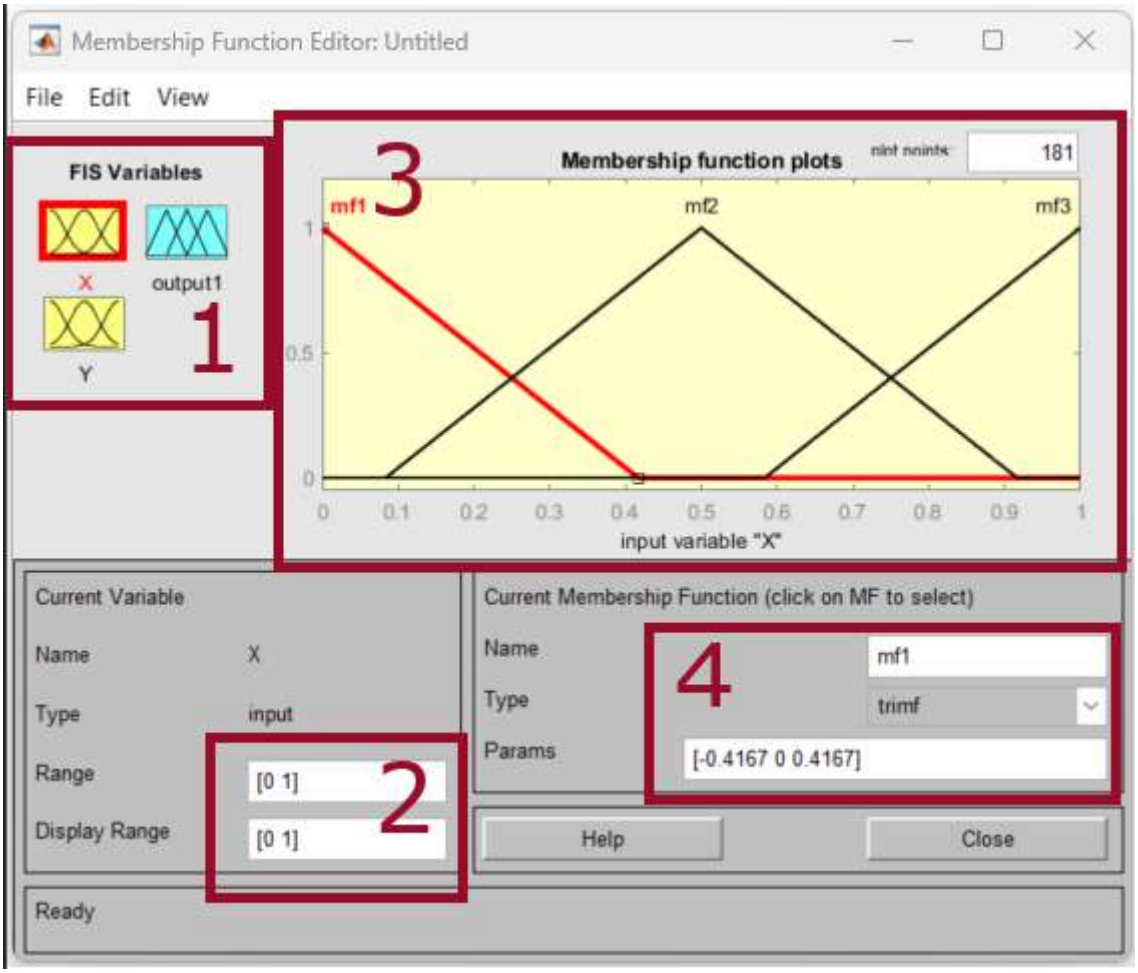

Рисунок 1.6 – Властивості змінної

В області № 1 надано вхідні і вихідні лінгвістичні змінні.

В області № 2 можна задати діапазон зміни змінної і діапазон відображення (найчастіше вони збігаються, але діапазон відображення може бути змінений для більш детального розгляду певної ділянки).

В області № 3 представлені функції приналежності. Необхідно обрати одну з функцій, щоб виконати її настройку.

В області № 4 надані настройки для вибраної функції: ім'я, тип функції і її параметри.

Алгоритм настройки вхідних і вихідних змінних:

1. Необхідно обрати змінну в полі № 1 і встановити для неї шкалу в області № 2.

2. Обрати «Edit-> Add MFs» щоб з'явилось наступне вікно, зображене на рисунку 1.6.

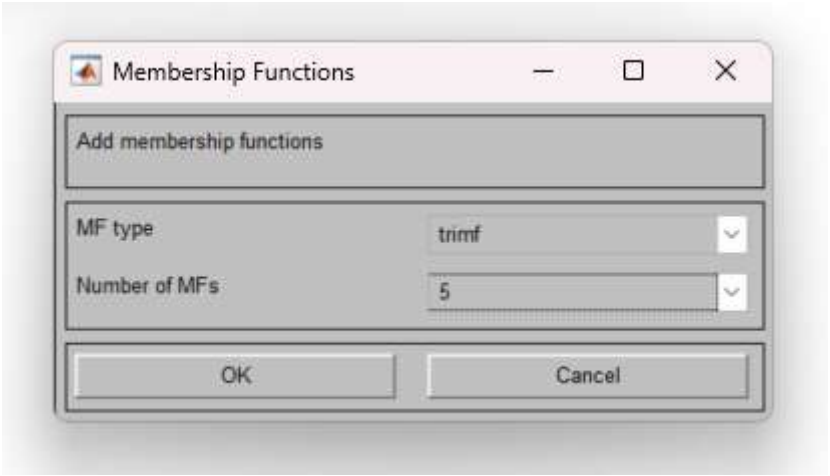

Рисунок 1.6 – Вибір типу функції і їх кількості

Необхідно обрати тип функції приналежності і їх кількість.

Після натискання «OK» в полі № 3 з'являться відповідні характеристики.

3. Вибираючи функції приналежності, задати в полі № 4 їх імена і форму.

4. Для видалення всіх функцій – обрати «Edit-> Remove All MFs» (поле № 3 очиститься). Тепер вікно з функціями приналежності можна закрити.

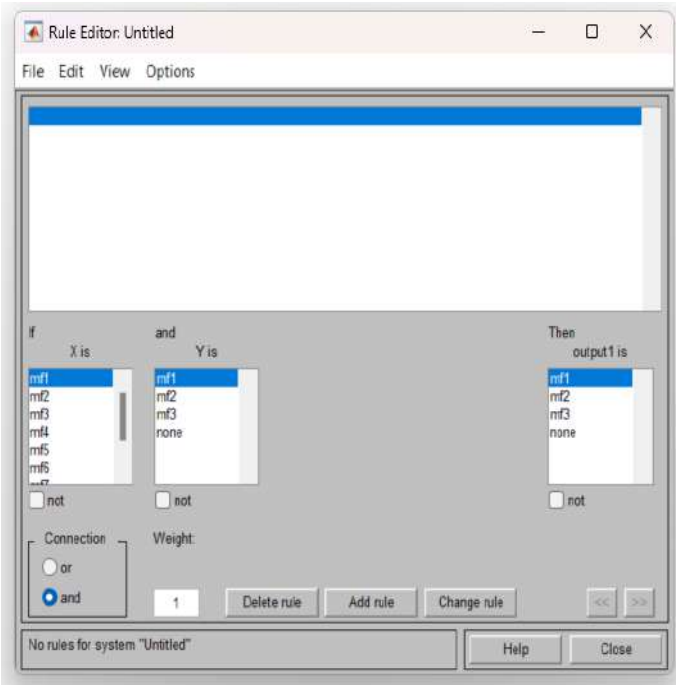

Рисунок 1.7 – Заповнення бази знань нечіткої логіки

Складаючи певну комбінацію з логічних відносин між функціями приналежності вхідних і вихідних змінних заповнюють базу знань нечіткої логіки.

Кнопка «Add Rule» додає правило з вибраними значеннями.

Кнопка «Delete Rule» видаляє вибране правило.

Кнопка «Change Rule» змінює правило на правило з вибраними значеннями.

Для того, щоб зберегти побудовану нечітку модель, потрібно зайти в меню «File-> Export-> To Disc».

Для того, щоб використовувати нечітку модель в моделі Simulink, необхідно зберегти її як змінну в робочому середовищі. Для цього зайти в «File-> Export-> To Workspace».

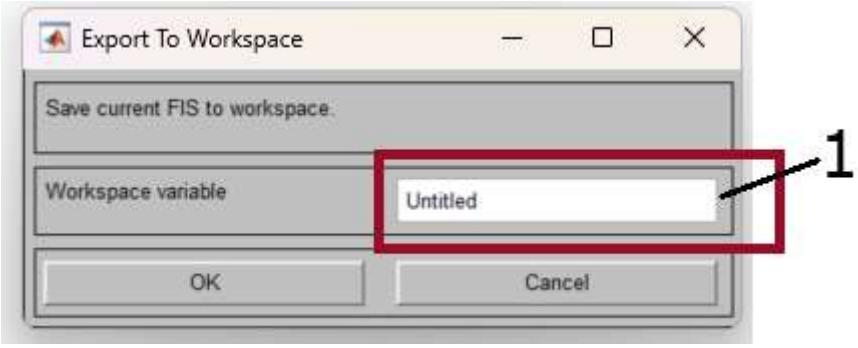

Рисунок 1.8 – Завдання імені змінної

В полі 1 необхідно задати ім'я змінної. Ім'я повинно починатися з літери і не містити пробілів (допустимі тільки латинські літери).

Поведінка нечіткої моделі визначається правилами, що є складовими бази знань. Вхід в редактор бази знань визначається подвійним клацанням миші на середньому блоці.

Fuzzy logic toolbox дозволяє виконати аналіз побудованої моделі за допомогою таких засобів, як «перегляд правил» (view rules) і «перегляд поверхні рішень» (view surface), які доступні з меню view.

При перегляді правил користувач має можливість задавати значення вхідних змінних, отримувати в ході вирішення логічних рівнянь значення вхідних змінних. Також тулбокс дозволяє відображати залежність виходу від двох (можна одного) входів в графічному вигляді. Ця функція доступна з меню view-> surface.

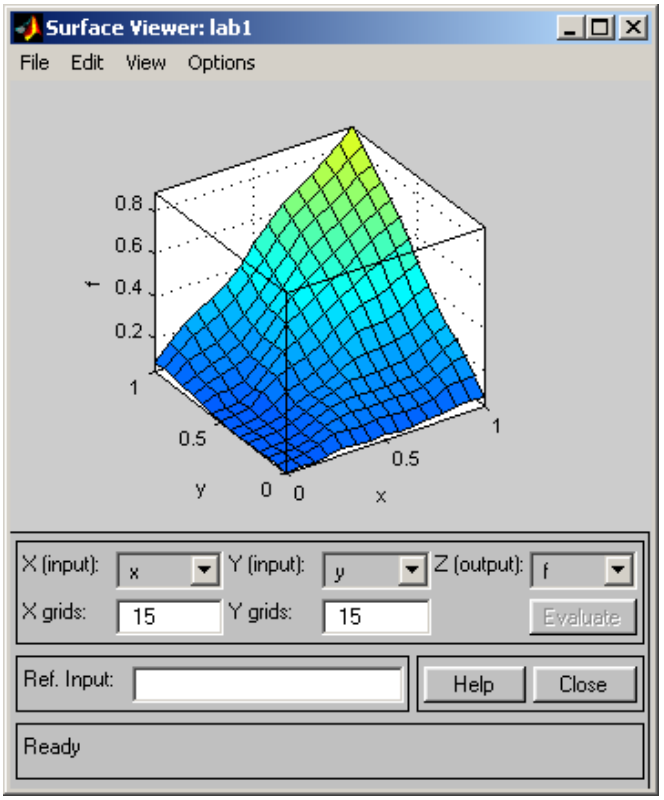

Рисунок 1.9 – Перегляд поверхні рішень

Використання побудованої нечіткої моделі в Simulink:

Створити нову, порожню модель в Matlab Simulink «File-> New-> Model». У браузері з бібліотеки Fuzzy Logic Toolbox і обрати блок Fuzzy Logic Controller (рисунок 1.10).

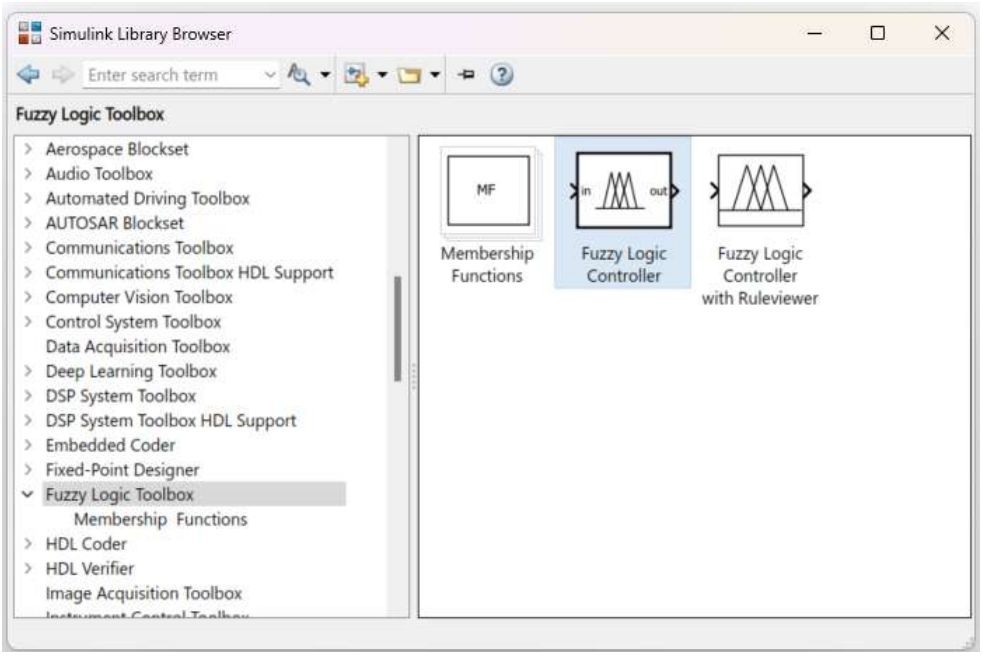

Рисунок 1.10 – Створення моделі Fuzzy Logic Controller

Далі потрібно клацнути два рази мишею на Fuzzy Logic Controller і у вікні вписати ім'я змінної під якою збережена нечітка модель в робочій області MatLab.

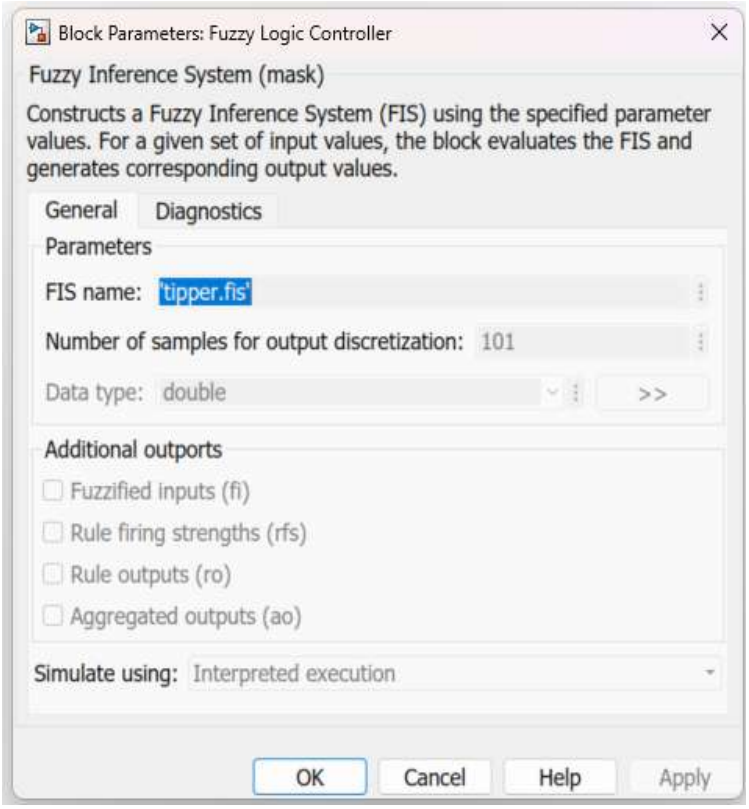

Рисунок 1.11 – Задання параметрів в блоці Fuzzy Logic Controller

Після цього необхідно додати в модель необхідні елементи для перегляду результатів моделювання.

Приклад моделювання функції 2-х змінних $z = x * y$:

1. Для моделювання функції в Simulink засобами тулбокса Fuzzy logic, в блоці Fcn (з User-Defined Functions) задати свою функцію (Наприклад: u[1] * u[2] для функції $z = x * y$).

Нехай змінні x і y змінюються однаково, тобто на обидва входи блоку Fcn подати однаковий сигнал, реалізований блоком ramp.

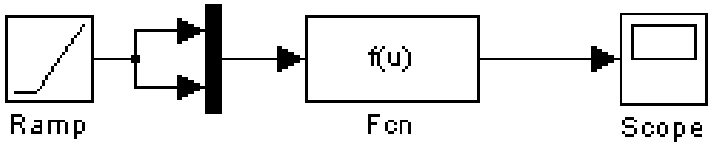

Рисунок 1.12 – Модель системи

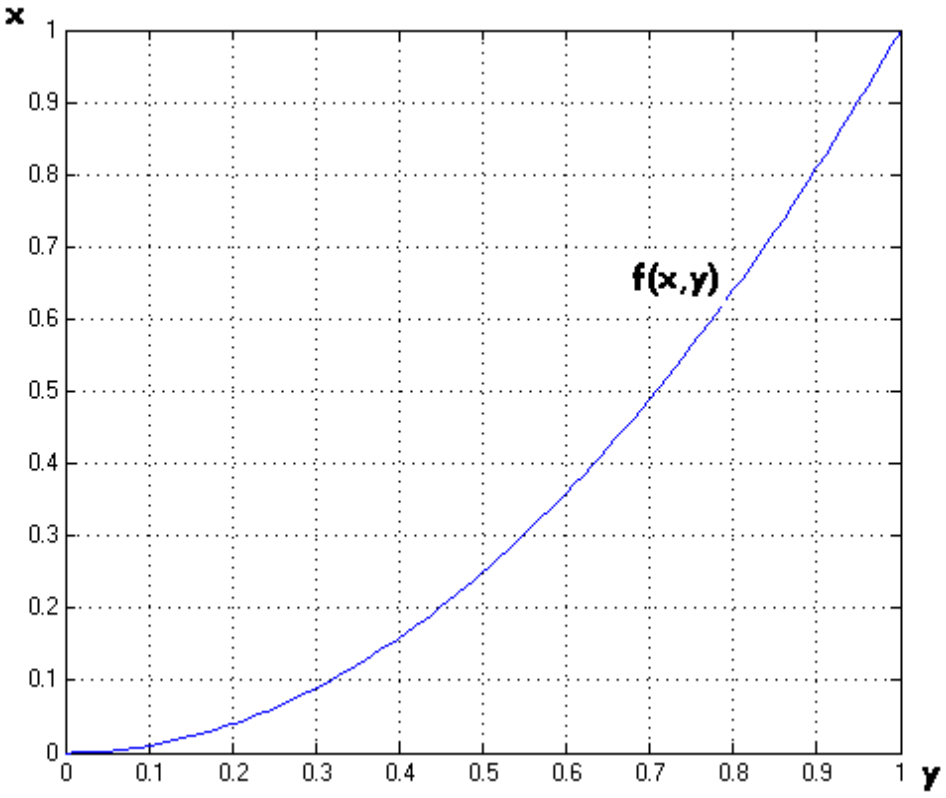

Рисунок 1.13 – Графік заданої функції

Рекомендації: для успішного моделювання з невеликим числом функцій приналежності потрібно вибирати монотонну ділянку функції.

Тепер потрібно зробити вибір кількості функцій приналежності для кожної змінної (*x*, *y*). Чим більше таких змінних тим якість моделювання вище, але тим нижче вартість при використанні нечіткої логіки. Тому в даному випадку для входів (змінні *x*, *y*) взяти 6 функцій, а для виходу 9. Розподіл функцій приналежності зручно виконувати так, як показано на рисунку 1.14.

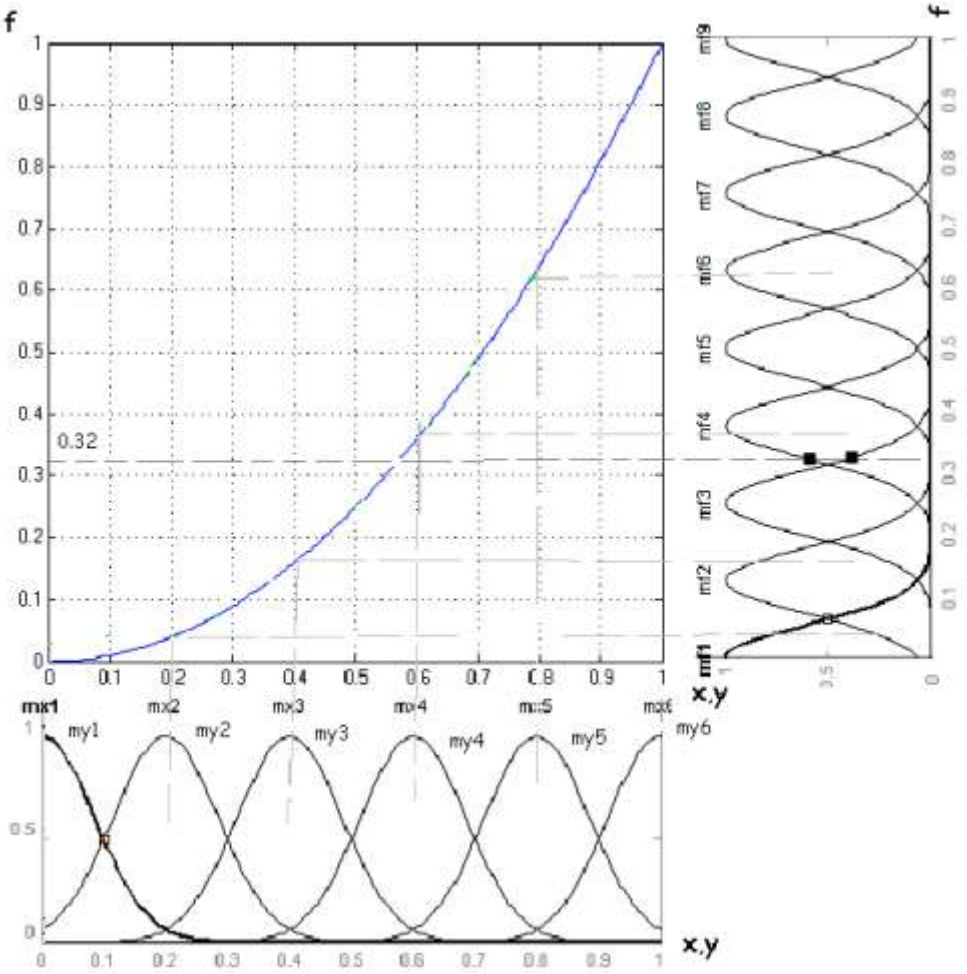

Рисунок 1.14 – Розподіл функцій приналежності

2. Створити модель функції в fuzzy (2 входи і 1 вихід)

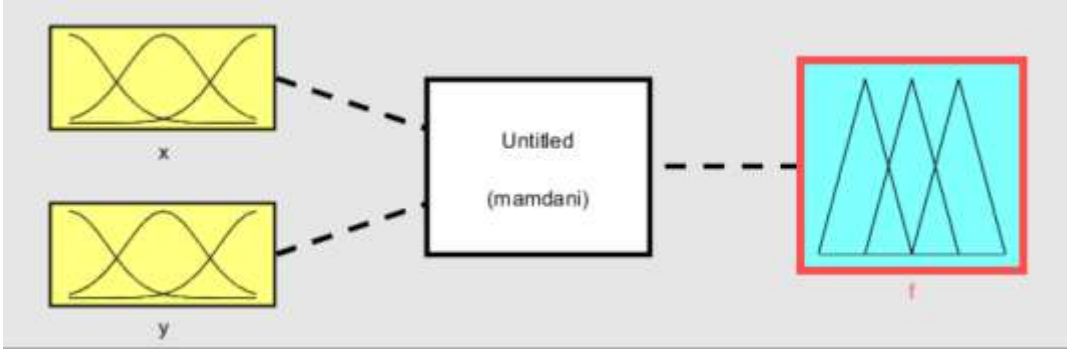

Рисунок 1.15 – Схема моделі функції в fuzzy

3. Вказати діапазони зміни вхідних і вихідних змінних, додаючи обрану кількість функцій приналежності (6 для вхідних і 9 для вихідних змінних). MatLab автоматично рівномірно розподілить їх всередині діапазону:

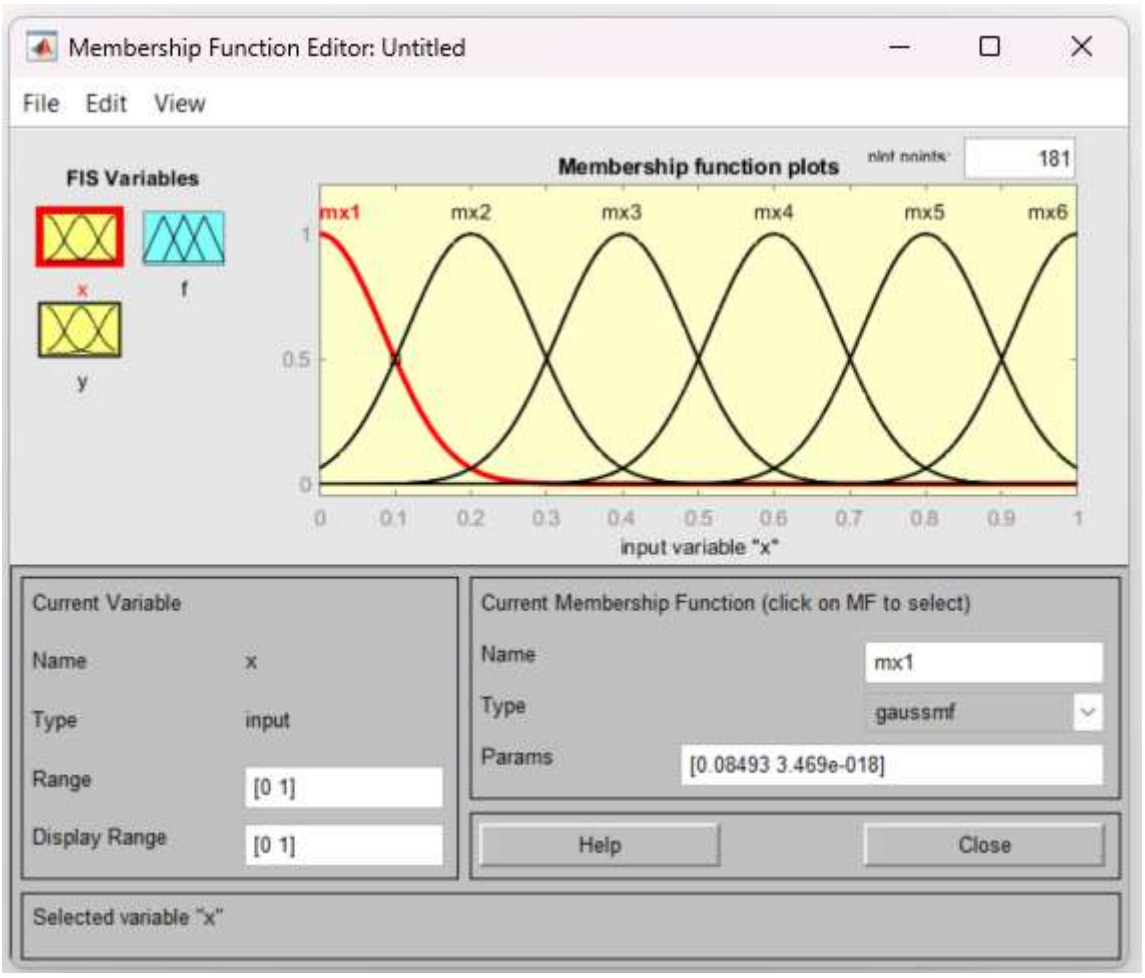

Рисунок 1.16 – Редагування властивостей змінної *x*

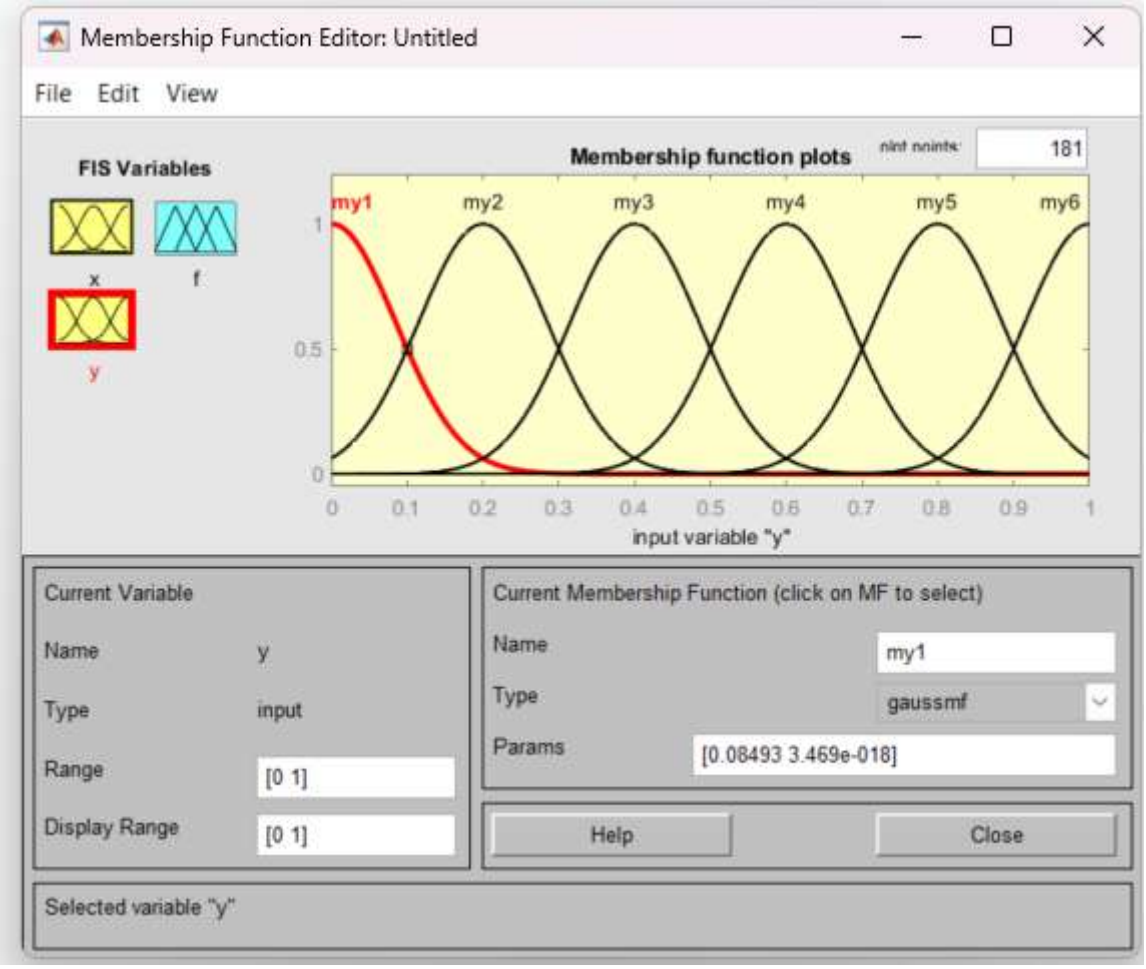

Рисунок 1.17 – Редагування властивостей змінної *y*

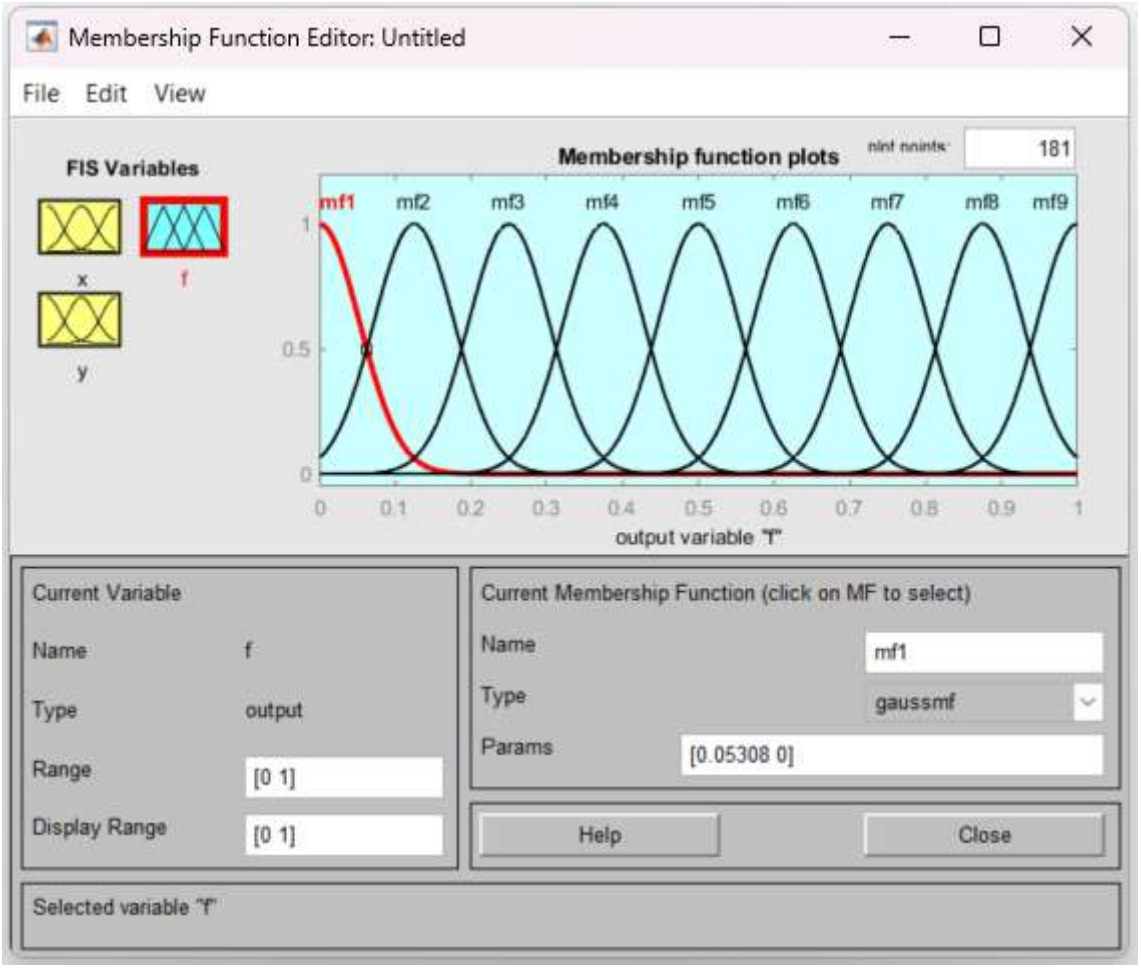

Рисунок 1.18 – Редагування властивостей змінної *z*

4. Далі необхідно заповнити базу знань моделі правилами. Правило – поставлена у відповідність виходу комбінація входів.

Так як для входів у нас по 6 функцій приналежності, то комбінацій: 6*6 = 36 правил.

Правила складають, підставляючи значення абсцис в точці максимуму функції приналежності входів в виразі *f*(*x*, *y*). Результат співставляється з тією функцією приналежності на виході, ордината якої більше.

Наприклад (див. рисунок 1.14): функції mx3 відповідає значення *x*=0.4, функції my5 відповідає значення 0.8. Тоді *f* = *f*(*x*, *y*) = 0.4 * 0.8 = 0.32, необхідно відмітити на осі абсцис і знайти функцію приналежності з максимальною ординатою (див. рисунок 1.14). Це функція mf4. Таким чином з'явилось одне правило: **якщо на вході x функція mx3 і на вході y**

функція my5, то на виході функція mf4. Далі проводиться внесення цього правила в базу знань:

If (x is mx3) and (y is mf5) then (f is mf2).

Аналогічно проводиться співставлення інших 35 правил.

Перед співставленням правил зручно скласти таблицю значень функції по максимумам вхідних функцій приналежності (Таблиця 1.2), яку потім перетворюють в таблицю імен функцій (Таблиця 1.3).

Таблиця 1.2 – Значення функцій по максимумам вхідних функцій приналежності

x y	0	0.2	0.4	0.6	0.8	1
0	0	0	0	0	0	0
0.2	0	0.04	0.08	0.12	0.16	0.2
0.4	0	0.08	0.16	0.24	0.32	0.4
0.6	0	0.12	0.24	0.36	0.48	0.6
0.8	0	0.16	0.32	0.48	0.64	0.8
1	0	0.2	0.4	0.6	0.8	1

Таблиця 1.3 – Таблиця імен функцій

	mx1	mx2	mx3	mx4	mx5	mx6
my1	mf1	mf1	mf1	mf1	mf1	mf1
my2	mf1	mf1	mf2	mf2	mf2	mf3
my3	mf1	mf1	mf2	mf3	mf3	mf4
my4	mf1	mf1	mf3	mf4	mf5	mf6
my5	mf1	mf2	mf4	mf5	mf6	mf7
my6	mf1	mf3	mf4	mf6	mf7	mf9

За такими таблицями складаються всі правила (рисунок 1.19).

1. If (x is mx1) and (y is my1) then (f is mf1) (1)
2. If (x is mx1) and (y is my2) then (f is mf1) (1)
3. If (x is mx1) and (y is my3) then (f is mf1) (1)
4. If (x is mx1) and (y is my4) then (f is mf1) (1)
5. If (x is mx1) and (y is my5) then (f is mf1) (1)
6. If (x is mx1) and (y is my6) then (f is mf1) (1)
7. If (x is mx2) and (y is my1) then (f is mf1) (1)
8. If (x is mx2) and (y is my2) then (f is mf1) (1)
9. If (x is mx2) and (y is my3) then (f is mf1) (1)
10. If (x is mx2) and (y is my4) then (f is mf1) (1)
11. If (x is mx2) and (y is my5) then (f is mf2) (1)
12. If (x is mx2) and (y is my6) then (f is mf3) (1)
13. If (x is mx3) and (y is my1) then (f is mf1) (1)
14. If (x is mx3) and (y is my2) then (f is mf2) (1)
15. If (x is mx3) and (y is my3) then (f is mf2) (1)
16. If (x is mx3) and (y is my4) then (f is mf3) (1)
17. If (x is mx3) and (y is my5) then (f is mf4) (1)
18. If (x is mx3) and (y is my6) then (f is mf4) (1)
19. If (x is mx4) and (y is my1) then (f is mf1) (1)
20. If (x is mx4) and (y is my2) then (f is mf2) (1)
21. If (x is mx4) and (y is my3) then (f is mf3) (1)
22. If (x is mx4) and (y is my4) then (f is mf4) (1)
23. If (x is mx4) and (y is my5) then (f is mf5) (1)
24. If (x is mx4) and (y is my6) then (f is mf6) (1)
25. If (x is mx5) and (y is my1) then (f is mf1) (1)
26. If (x is mx5) and (y is my2) then (f is mf2) (1)
27. If (x is mx5) and (y is my3) then (f is mf3) (1)
28. If (x is mx5) and (y is my4) then (f is mf5) (1)
29. If (x is mx5) and (y is my5) then (f is mf6) (1)
30. If (x is mx5) and (y is my6) then (f is mf7) (1)
31. If (x is mx6) and (y is my1) then (f is mf1) (1)
32. If (x is mx6) and (y is my2) then (f is mf3) (1)
33. If (x is mx6) and (y is my3) then (f is mf4) (1)
34. If (x is mx6) and (y is my4) then (f is mf6) (1)
35. If (x is mx6) and (y is my5) then (f is mf7) (1)
36. If (x is mx6) and (y is my6) then (f is mf9) (1)

Рисунок 1.19 – Всі складені 36 правил

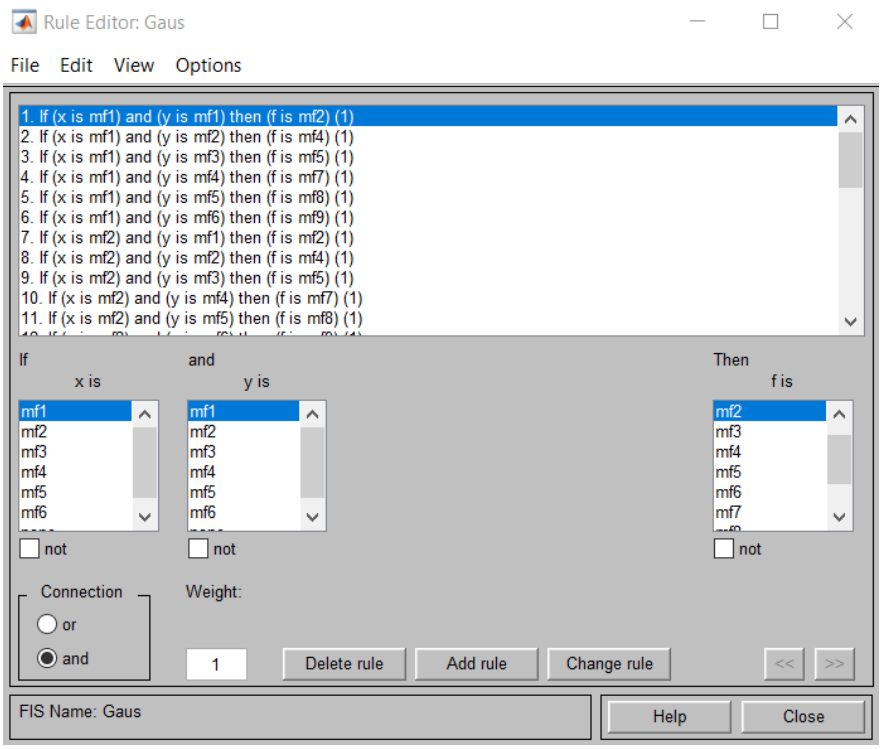

Рисунок 1.20 – Заповнення бази знань щойно створеними правилами

5. Далі необхідно зберегти модель в Workspace.

6. Додати на схему, побудовану в пункті 1, Fuzzy Logic Controller.

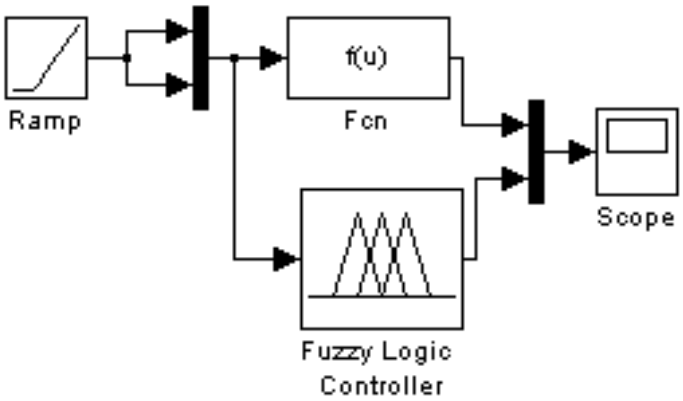

Рисунок 1.21 – Модель системи, доповнена Fuzzy Logic Controller

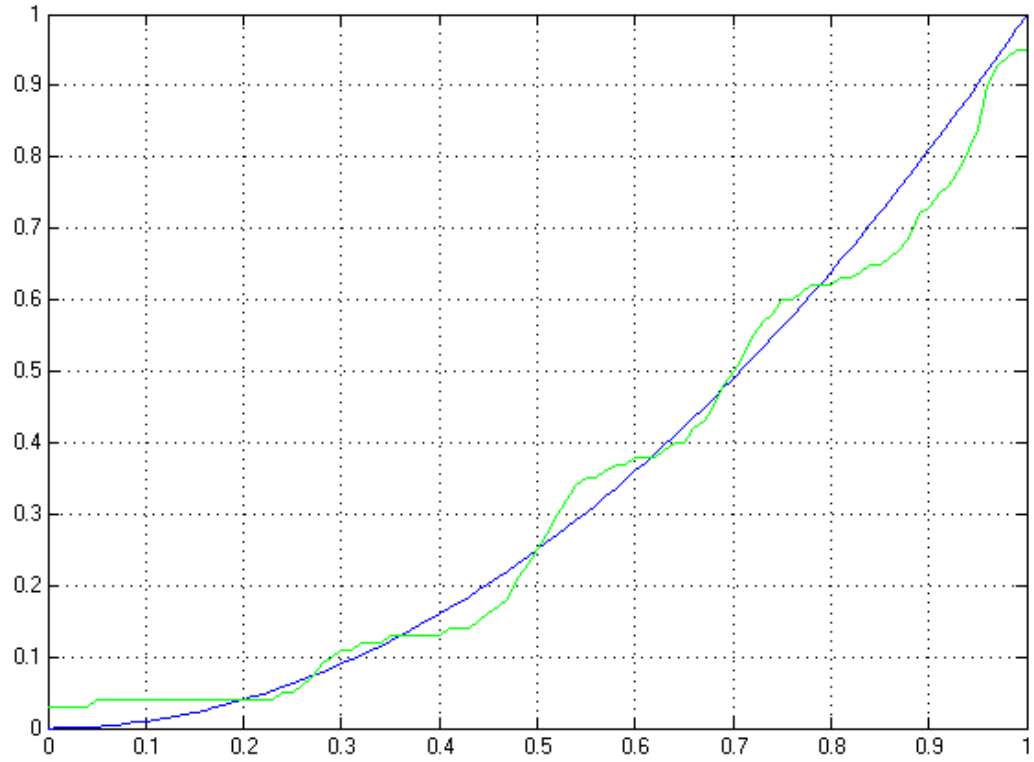

Рисунок 1.22 – Графік заданої функції з Fuzzy Logic Controller

7. Необхідно оцінити точність моделювання. Для цього необхідно порахувати середню відносну похибку моделювання.

Для розрахунку похибки застосовують формулу 1.1:

$$\varepsilon = (|Xe - Xд| / Xe) * 100\% , \qquad (1.1)$$

де Xe – еталонне значення величини; Xд – дійсне значення величини.

Використовуючи формули 1.1, блоки Fcn і integrator, потрібно побудувати частину моделі, що реалізує пошук середньої відносної похибки:

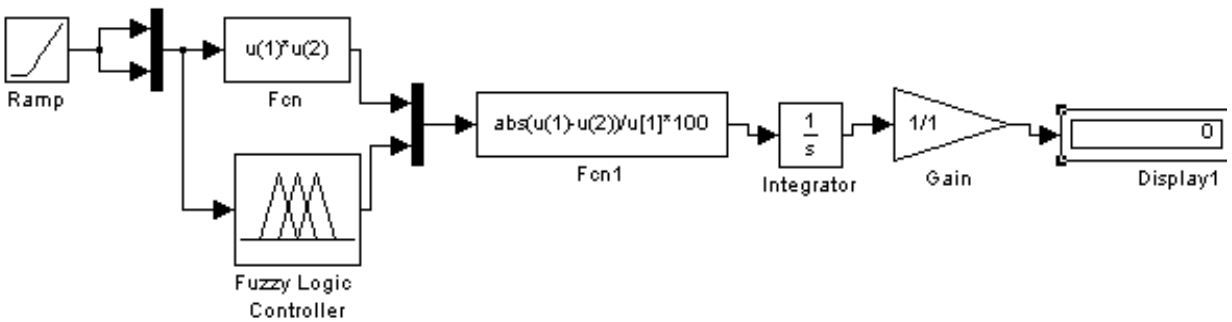

Рисунок 1.23 – Схема моделі, що реалізує пошук середньої відносної похибки

Ланкою Gain задають коефіцієнт 1/T, де T – час моделювання (меню Simulink -> Parameters).

Зауваження: З аналізу формули 1.1 видно, що якщо Xe приймає нульове значення (тобто в даному випадку функція проходить через нуль), то помилка прямує до нескінченності. Тому для уникнення цього можна «при-підняти» обидві величини на однакове значення (наприклад на 1), тобто формула 1.1 прийме вигляд :

$$\varepsilon = (|(Xe+1)- (Xд+1)| / (Xe+1)) * 100\%$$

Завдання на лабораторну роботу

1. Побудувати нечітку модель функції двох змінних згідно з варіантом, що містить 6 функцій приналежності для вхідних змінних і не менше 9 для вихідної.

2. Дослідити вплив форми функції приналежності (трикутник, трапеція, Гауса) на якість моделювання (порівняти відносні помилки моделювання).

3. Дослідити можливість зменшення числа правил за рахунок виключення деяких (перевірити достатність використання правил, що представляють тільки діагональ таблиці).

4. Зробити висновки.

ЛАБОРАТОРНА РОБОТА №2

Моделювання системи керування з двопозиційним та трипозиційним нечітким регулятором

Мета роботи: Засобами моделювання нечіткої логіки промоделювати роботу двопозиційних та трипозиційних регуляторів. На практиці провести дослідження нечітких регуляторів та порівняти з традиційними.

Короткі теоретичні відомості

Для багатьох задач, пов'язаних з керуванням технологічними процесами, необхідна побудова моделі розглянутого процесу. Знання моделі дозволяє підібрати відповідний регулятор (модуль керування). Однак часто побудова коректної моделі являє собою складну проблему, що іноді вимагає введення різних спрощень. Застосування теорії нечітких множин для керування технологічними процесами не припускає знання моделей цих процесів. Слід тільки сформулювати правила поведінки у формі нечітких умовних суджень типу IF ... THEN.

Статична характеристика двопозиційного реле із зоною невизначеності має вигляд:

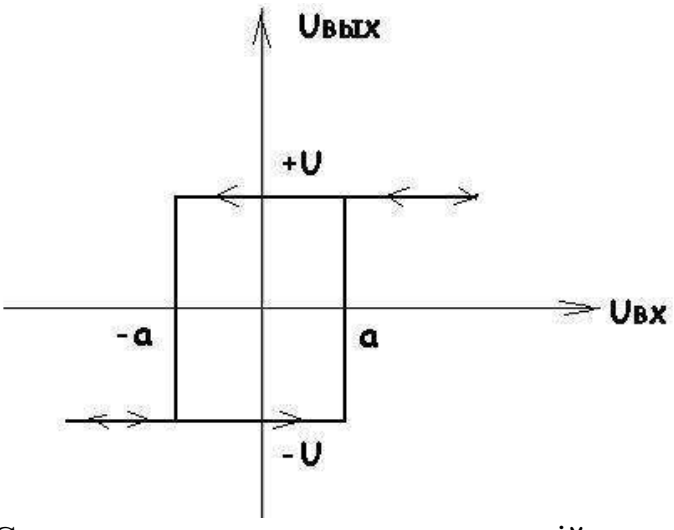

Рисунок 2.1 – Статична характеристика двопозиційного реле із зоною невизначеності

Як видно з приведеної характеристики на інтервалі [-*a*, *a*] вихід елементу може приймати як значення +*U* так і -*U*. Стан на цьому інтервалі

визначається попереднім положенням релейного елементу, тобто за допомогою похідної за часом від вхідного сигналу.

Таким чином поведінку двопозиційного реле можна записати за допомогою правил:

Якщо $U_{вх} > a$ тоді $U_{вих} = +U$

Якщо $U_{вх} < -a$ тоді $U_{вих} = -U$

Якщо $-a < U_{вх} < a$ і $dU_{вх}/dt > 0$ тоді $U_{вих} = -U$

Якщо $-a < U_{вх} < a$ і $dU_{вх}/dt < 0$ тоді $U_{вих} = +U$

Статична характеристика трипозиційного реле з двома зонами невизначеності і зоною нечутливості має вигляд:

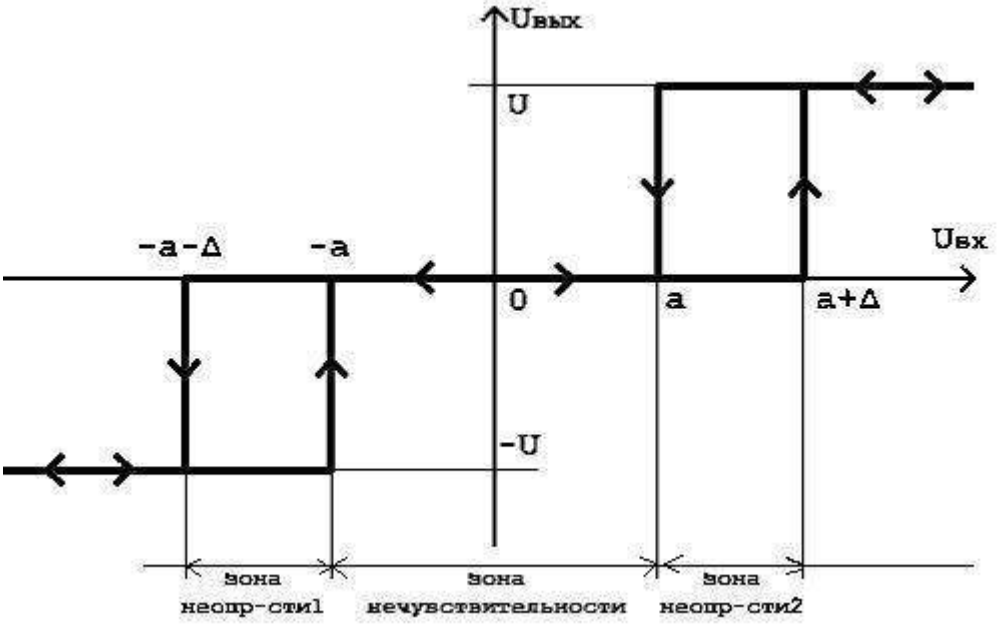

Рисунок 2.2 – Статична характеристика трипозиційного реле з двома зонами невизначеності і зоною нечутливості

Поведінка такого елементу описується виразом:

Якщо $U_{вх} < (-a - \Delta)$ тоді $U_{вих} = -U$

Якщо $U_{вх} > (a + \Delta)$ тоді $U_{вих} = U$

Якщо $a < U_{вх} < -a$ тоді $U_{вих} = 0$

Якщо $(-a - \Delta) < U_{вх} < -a$ і $(dU_{вх}/dt > 0)$ тоді $U_{вих} = -U$

Якщо $(-a - \Delta) < U_{вх} < -a$ і $dU_{вх}/dt < 0$ тоді $U_{вих} = 0$

Якщо $a < U_{вх} < (a + \Delta)$ і $dU_{вх}/dt > 0$ тоді $U_{вих} = 0$

Якщо $a < U_{вх} < (a + \Delta)$ і $dU_{вх}/dt < 0$ тоді $U_{вих} = U$

Порядок виконання роботи

Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи.

Реалізація поведінки двопозиційного реле за допомогою нечіткого регулятора виконується в наступній послідовності:

1. Визначення лінгвістичних змінних і вибір функцій приналежності. Цей етап зручно виконувати як показано на рисунку 2.3.

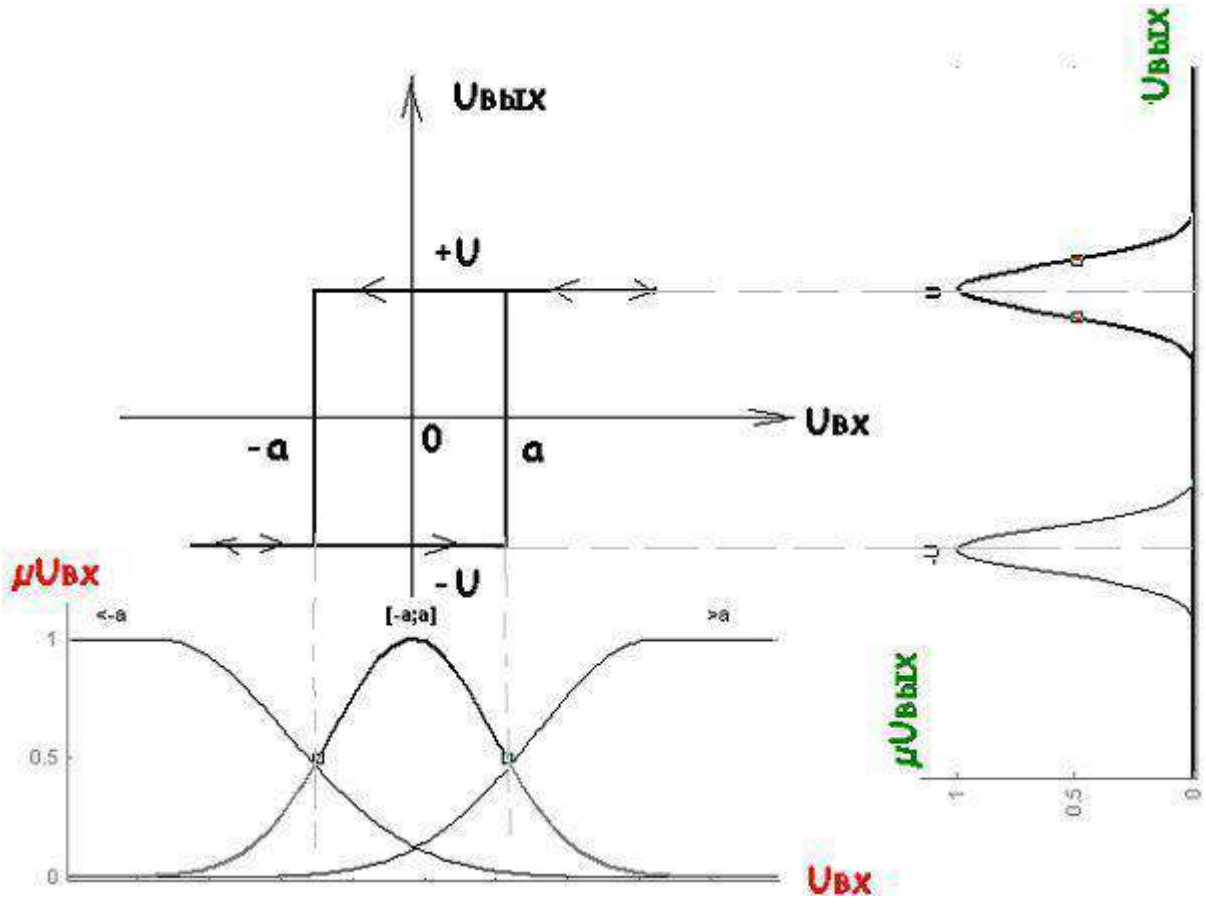

Рисунок 2.3 – Визначення лінгвістичних змінних і вибір функцій приналежності

Аналіз похідної вхідного сигналу представлено на рисунку 2.4.

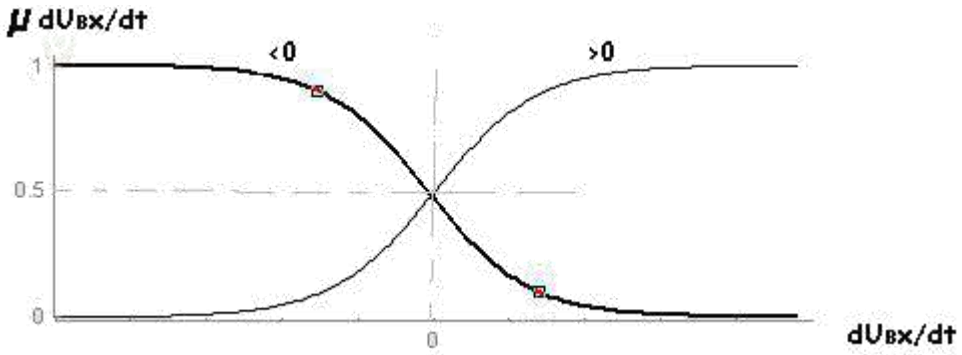

Рисунок 2.4 – Аналіз похідної вхідного сигналу

Таким чином лінгвістичними змінними є:

Вхід 1: «<-*a*», «[-*a*;*a*]», «>*a*»;

Вхід 2: «<0», «>0»;

Вихід 1: «-*U*», «*U*».

2. Заповнення бази знань.

На основі приведених виразів для поведінки реле і введених лінгвістичних змінних записуємо правила:

Якщо « > *a*» тоді «U»

Якщо «< -*a*» тоді «-U»

Якщо «[-*a*;*a*]» і «>0» тоді «-U»

Якщо «[-*a*;*a*]» і «<0» тоді «U»

3. Моделюємо релейну систему з автоколивальним перехідним процесом:

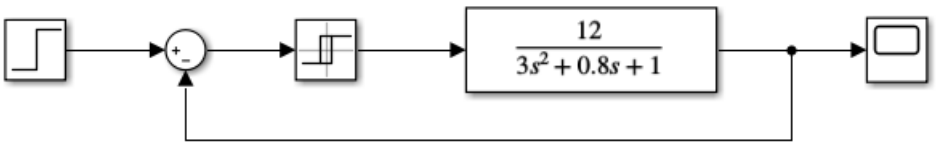

Рисунок 2.5 – Модель системи керування з автоколивальним перехідним процесом

Задаємо параметри реле як показано на наступному рисунку.

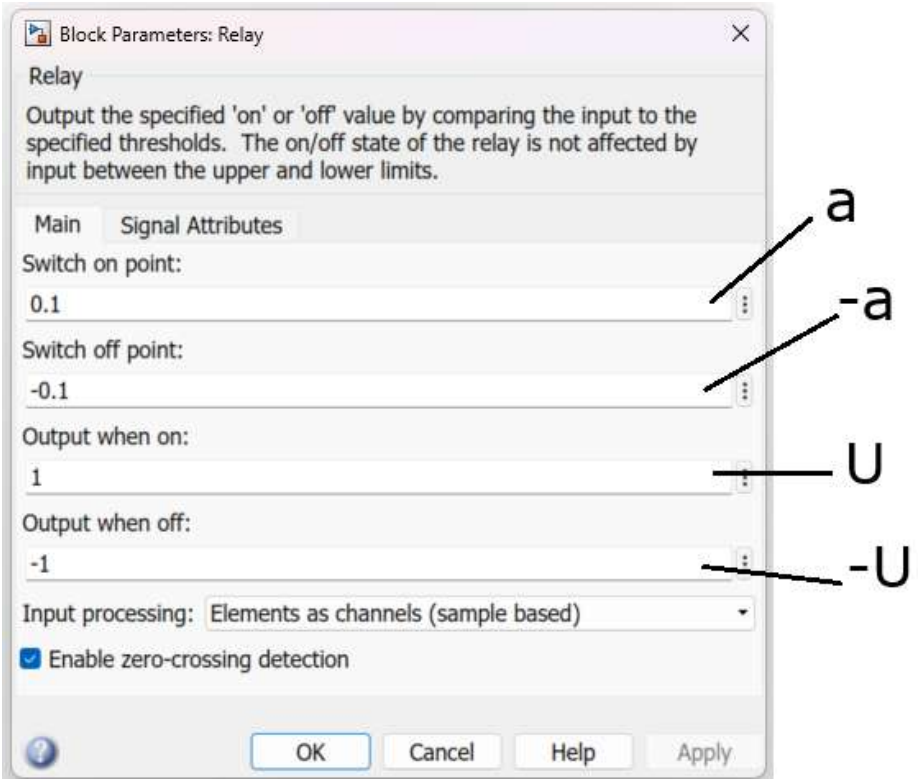

Рисунок 2.6 – Вікно параметрів реле

Перехідний процес системи керування з автоколивальним перехідним процесом представлено на рисунку 2.7.

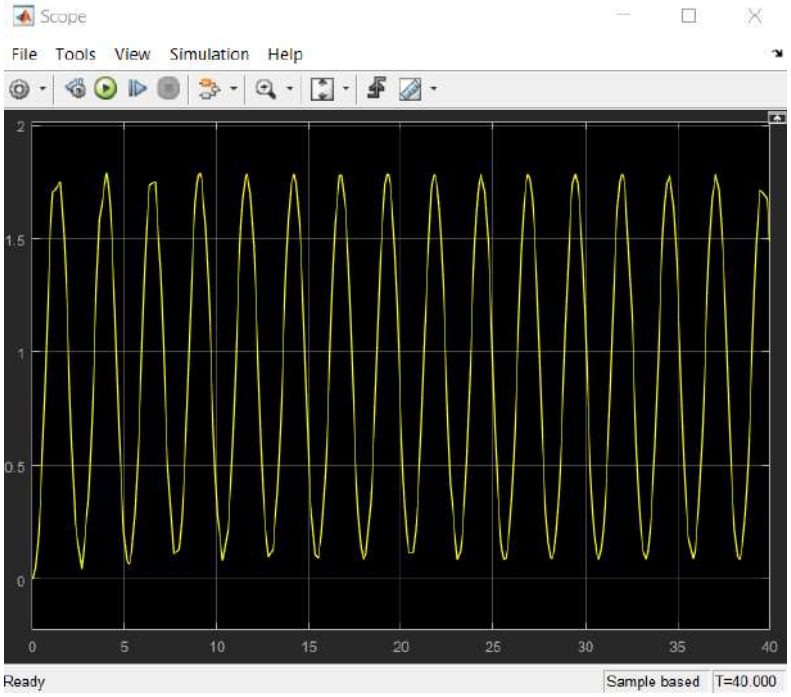

Рисунок 2.7 – Перехідний процес системи керування з автоколивальним перехідним процесом

4. Створюємо в fuzzy модель реле як показано далі.

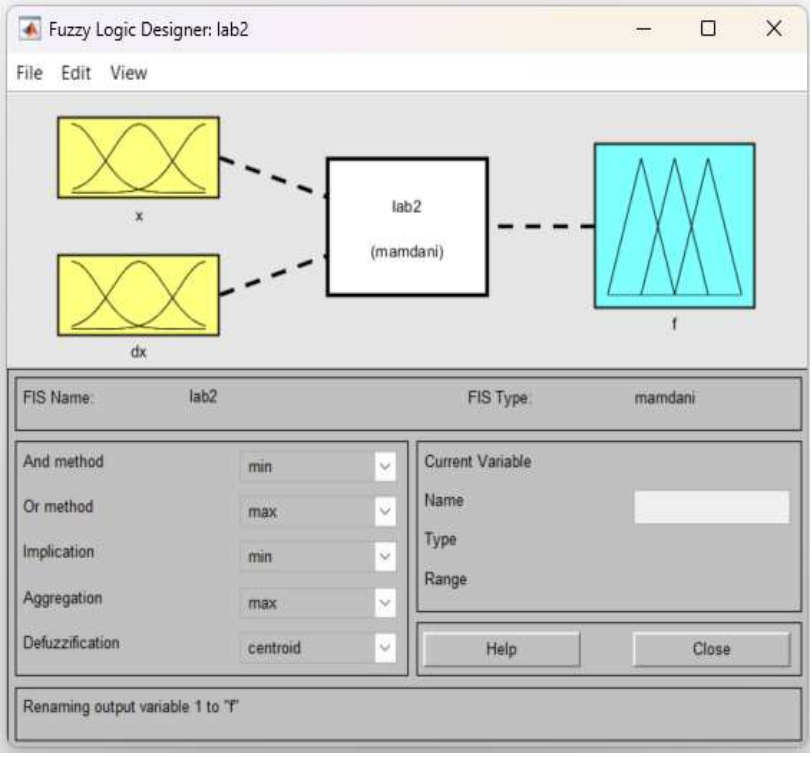

Рисунок 2.8 – Вікно задання вхідних і вихідних змінних

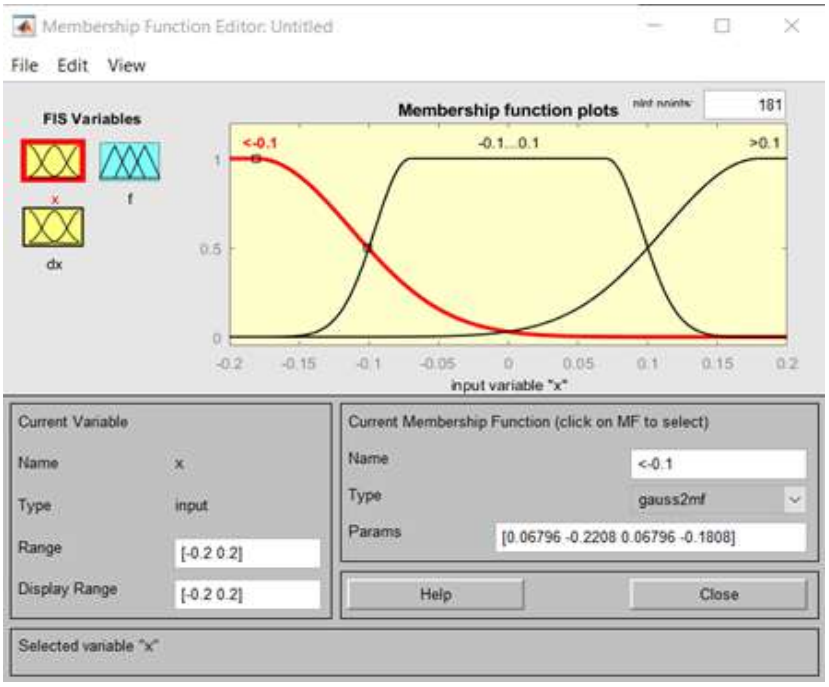

Рисунок 2.9 – Вікно задання вхідної змінної x

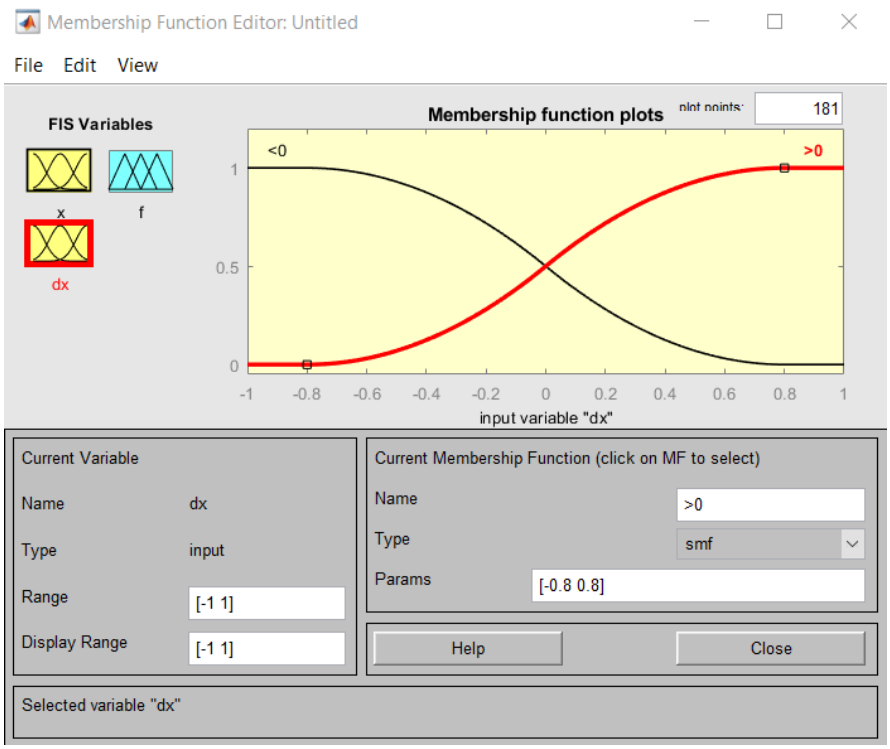

Рисунок 2.10 – Вікно задання вхідної змінної *dx*

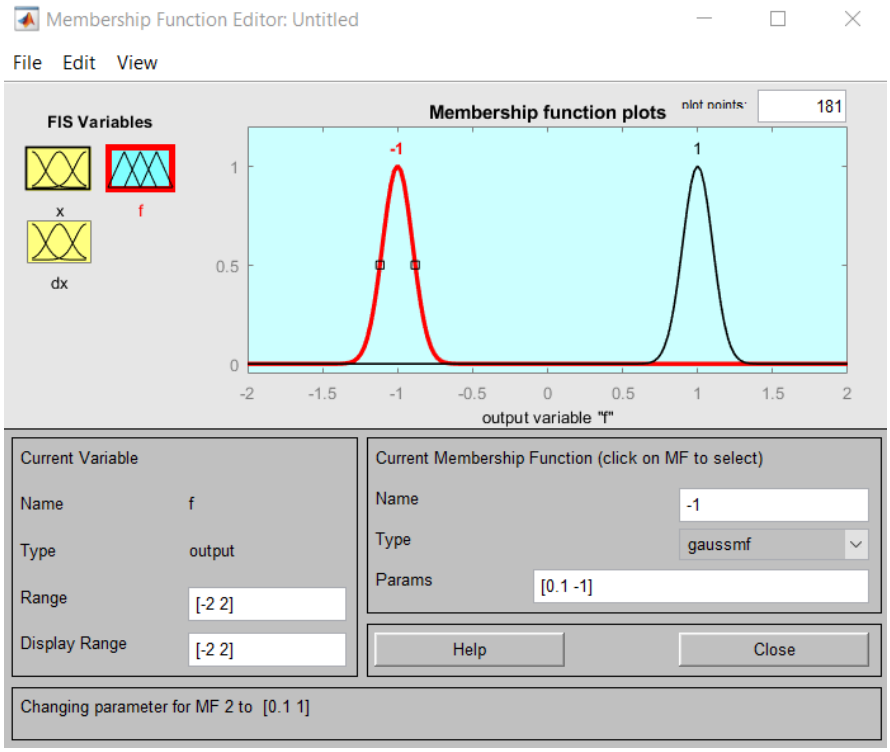

Рисунок 2.11 – Вікно задання вихідної змінної *f*

Заповнюємо базу знань:

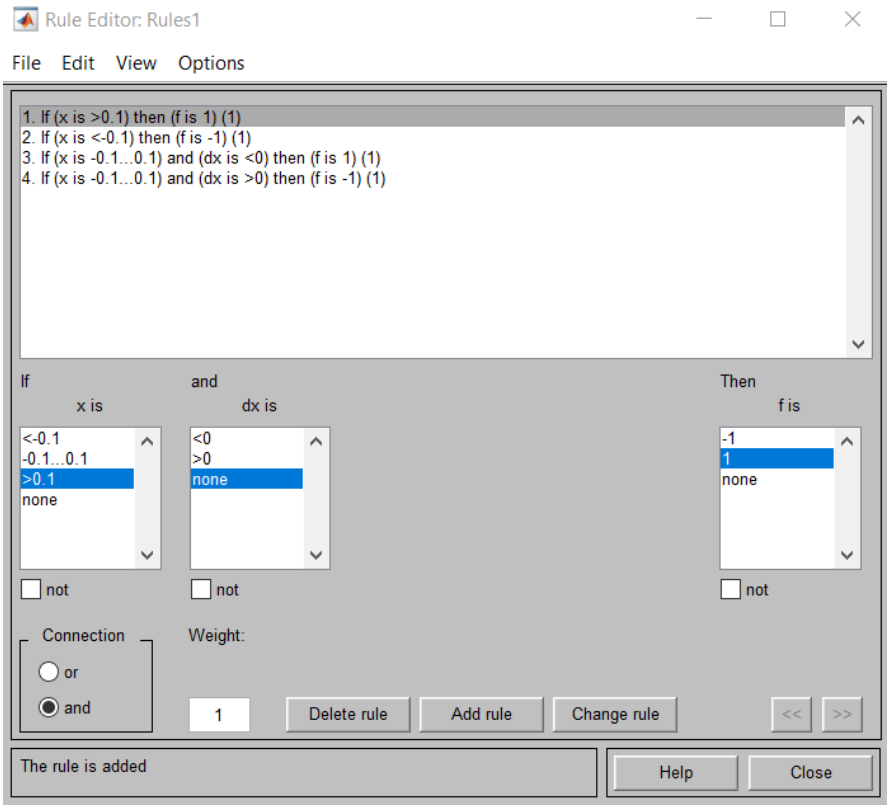

Рисунок 2.12 – Заповнення бази знань

Переносимо нечітку модель в simulink і порівнюємо результати з початковою моделлю:

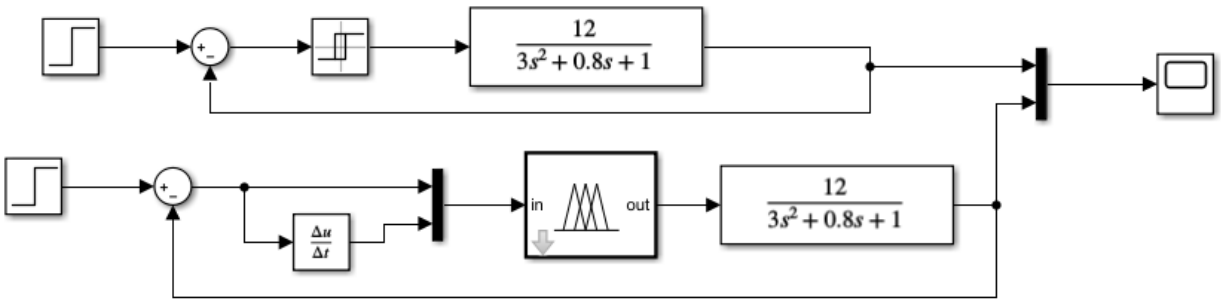

Рисунок 2.13 – Модель систем керування з класичним реле та його нечіткою моделлю

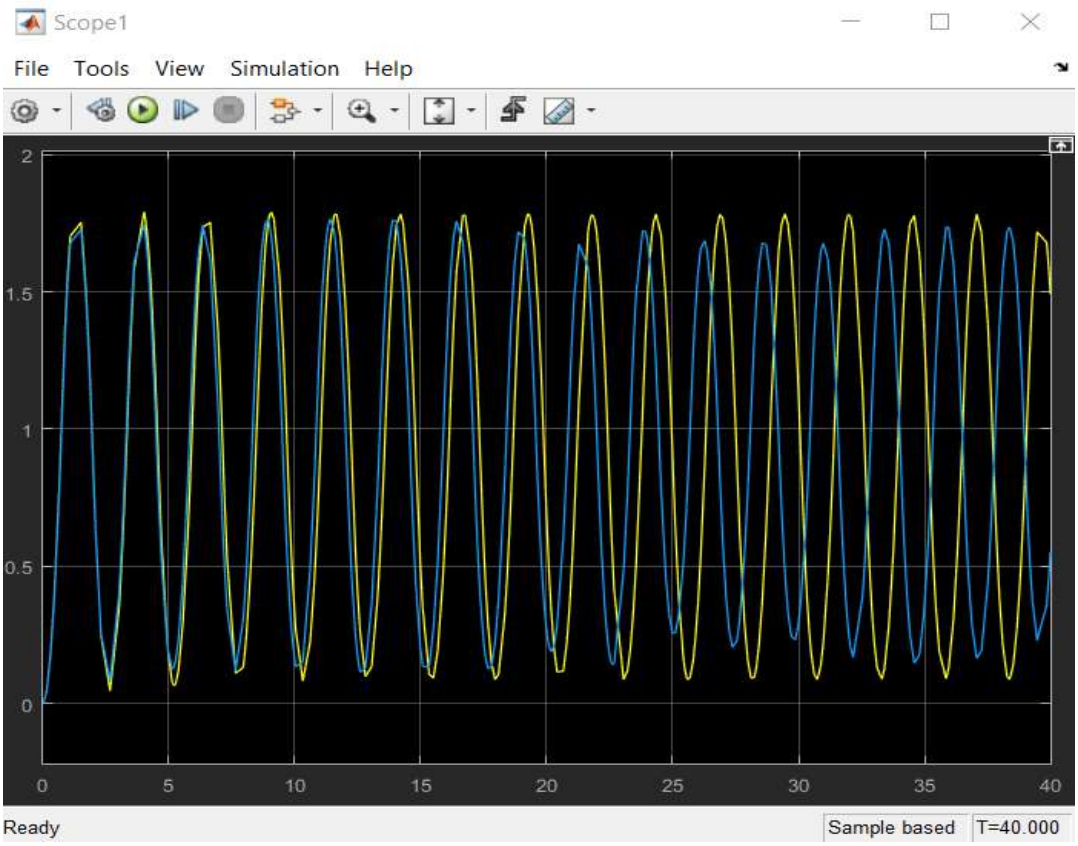

Рисунок 2.14 – Перехідний процес систем керування з класичним реле та його нечіткою моделлю

Реалізація поведінки трипозиційного реле за допомогою нечіткого регулятора виконується в наступній послідовності:

1. Визначаємо лінгвістичні змінні за статичною характеристикою:

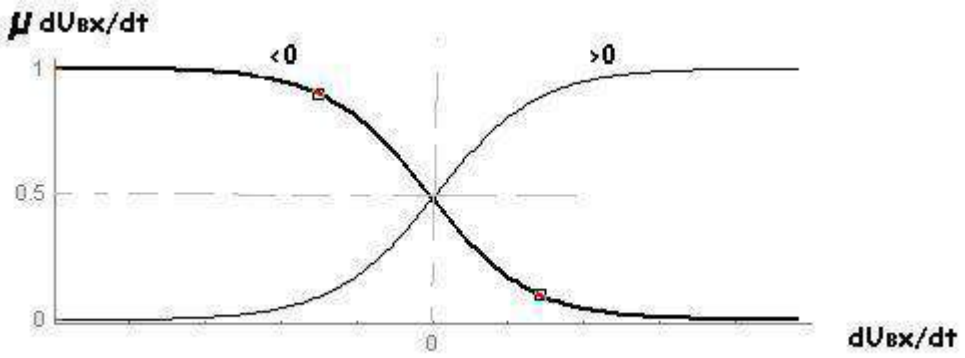

Рисунок 2.15 – Визначення лінгвістичних змінних за статичною характеристикою

Опис входу по похідній залишається тим же:

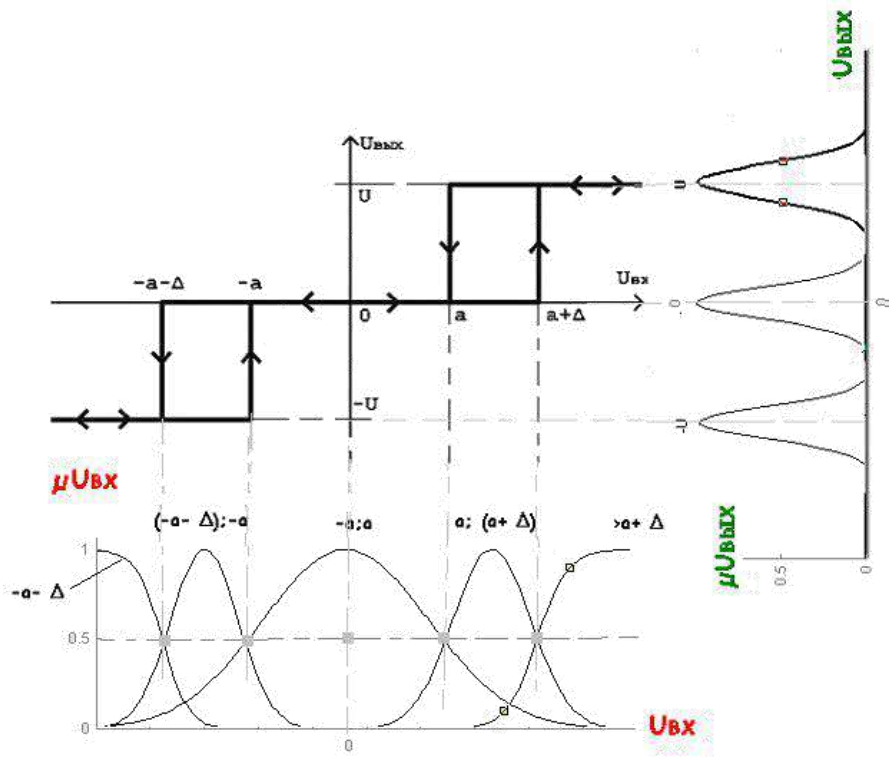

Рисунок 2.16 – Опис входу по похідній

Таким чином лінгвістичними змінні є:

Вхід 1: «<-*a* - Δ», «(-*a* - Δ); -*a*», «-*a*; *a*», «*a*; (*a* + Δ)», «>*a* + Δ»;

Вхід 2: «<0», «>0»;

Вихід 1: «-*U*» , «0», «*U*».

Тоді правила бази знань запишуться таким чином:

Якщо «<-*a* - Δ» тоді «-*U*»

Якщо «>*a* + Δ» тоді «*U*»

Якщо «-a; a» тоді «0»

Якщо «(-*a* - Δ); -*a*» і «>0» тоді «-*U*»

Якщо «(-*a* - Δ);-*a*» і «<0» тоді «0»

Якщо «*a*; (*a* + Δ)» і «>0» тоді «0»

Якщо «*a*; (*a* + Δ)» і «<0» тоді «U»

2. Моделюємо релейну систему з автоколивальним перехідним процесом:

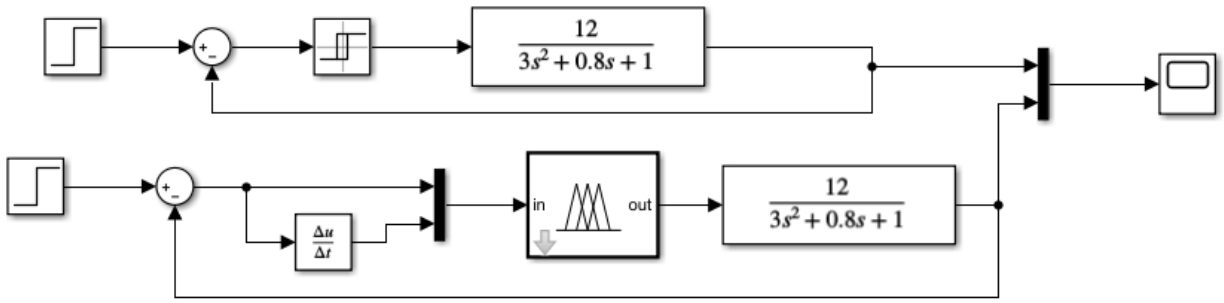

Рисунок 2.17 – Модель релейної системи керування з автоколивальним перехідним процесом

3. Задаємо параметри реле, що описує праву частину статичної характеристики:

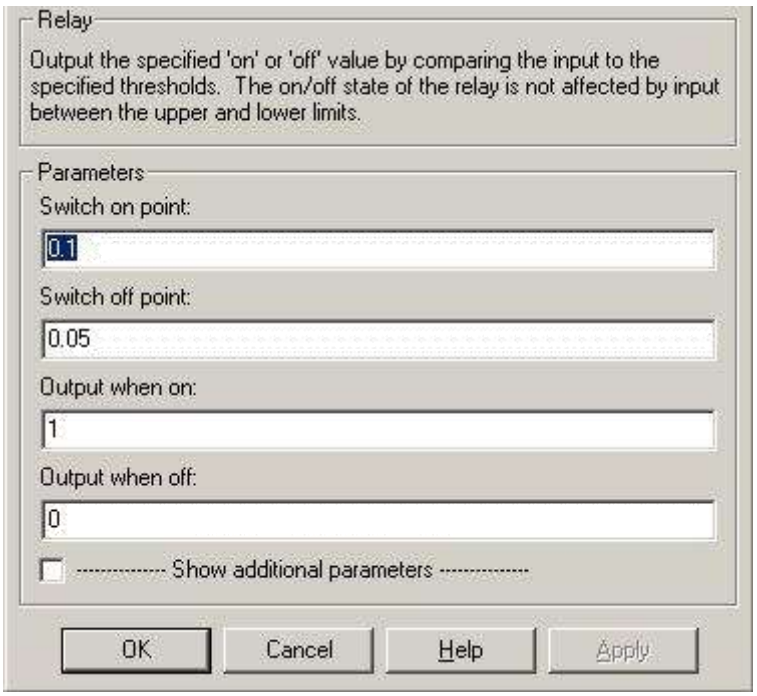

Рисунок 2.18 – Вікно для налаштування блоку реле ліву частину статичної характеристики.

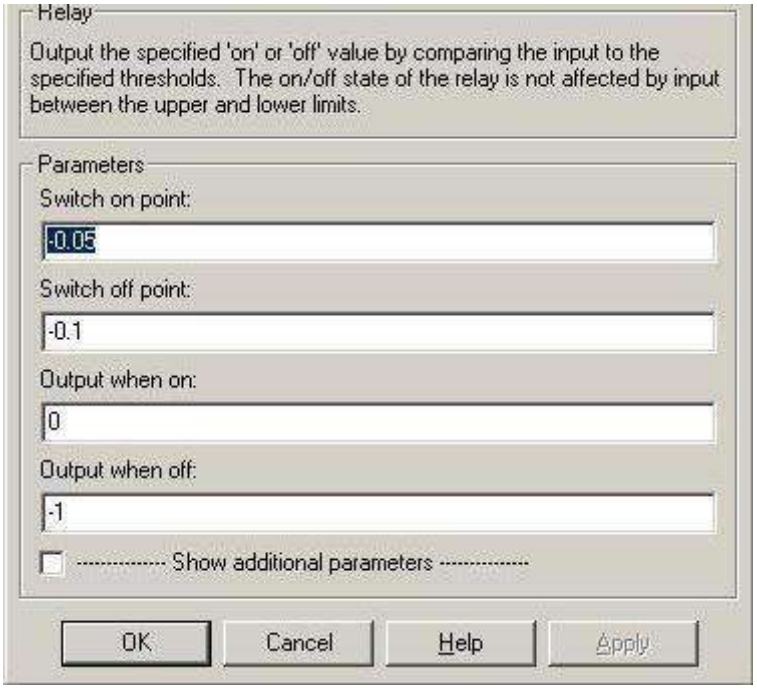

Рисунок 2.19 – Вікно для налаштування блоку реле

Отримуємо наступні графіки:

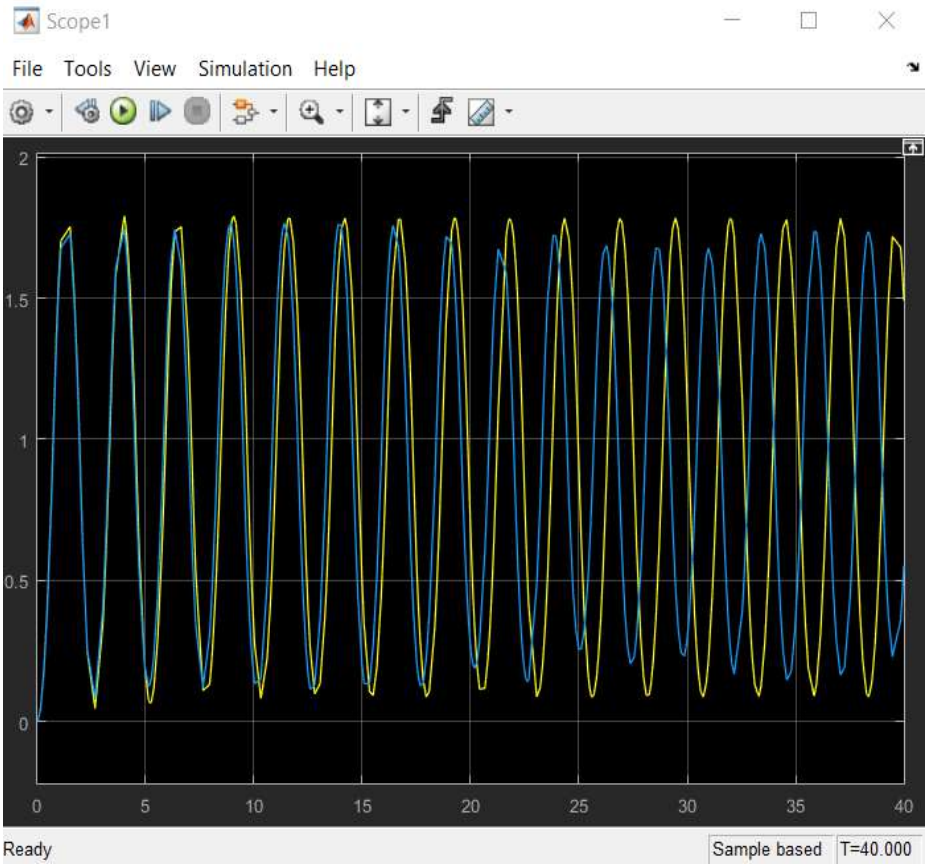

Рисунок 2.20 – Перехідний процес системи керування з автоколивальним перехідним процесом

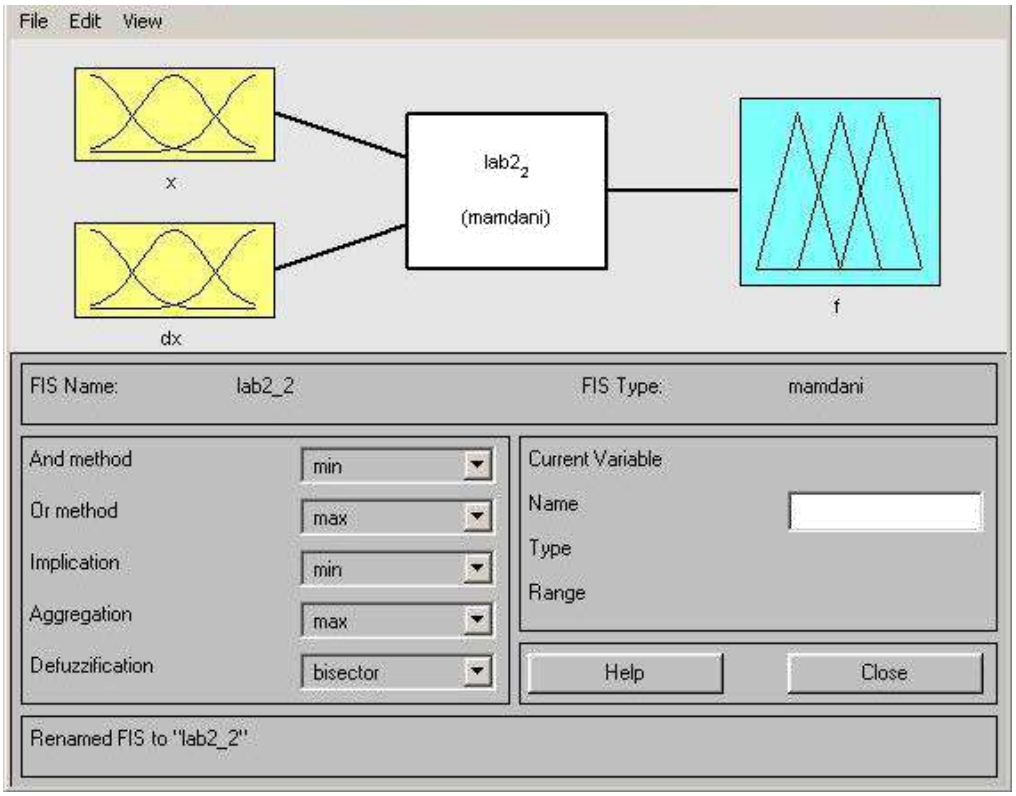

Рисунок 2.21 – Вікно задання вхідних і вихідних змінних

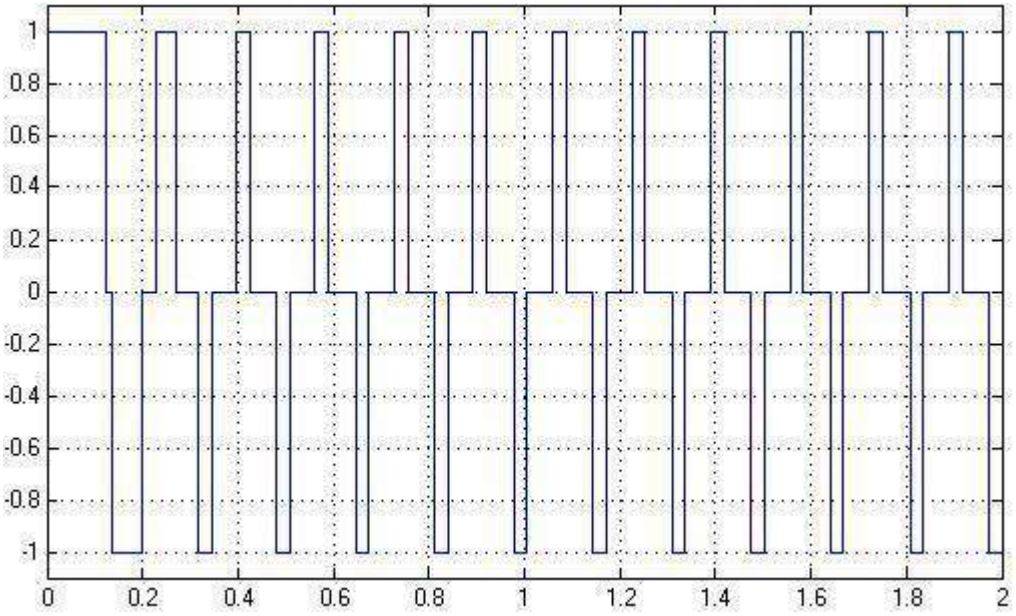

Рисунок 2.22 – Графік управляючого сигналу

Створюємо в fuzzy модель реле.

Вхід 1:

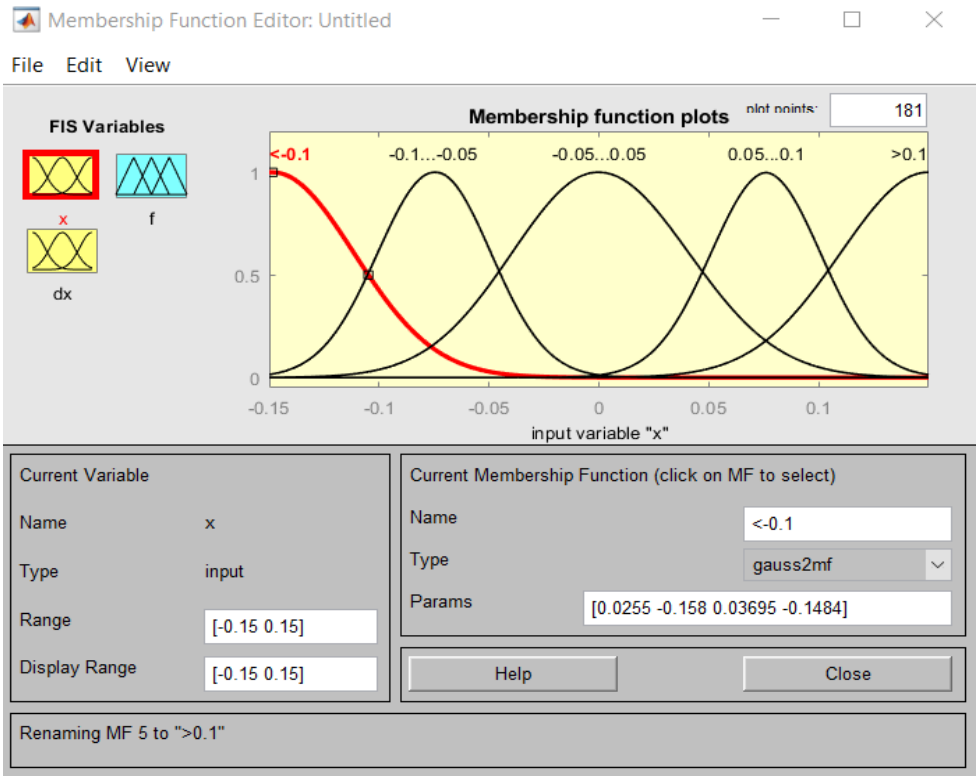

Рисунок 2.23 – Вікно задання вхідної змінної x

Вхід 2:

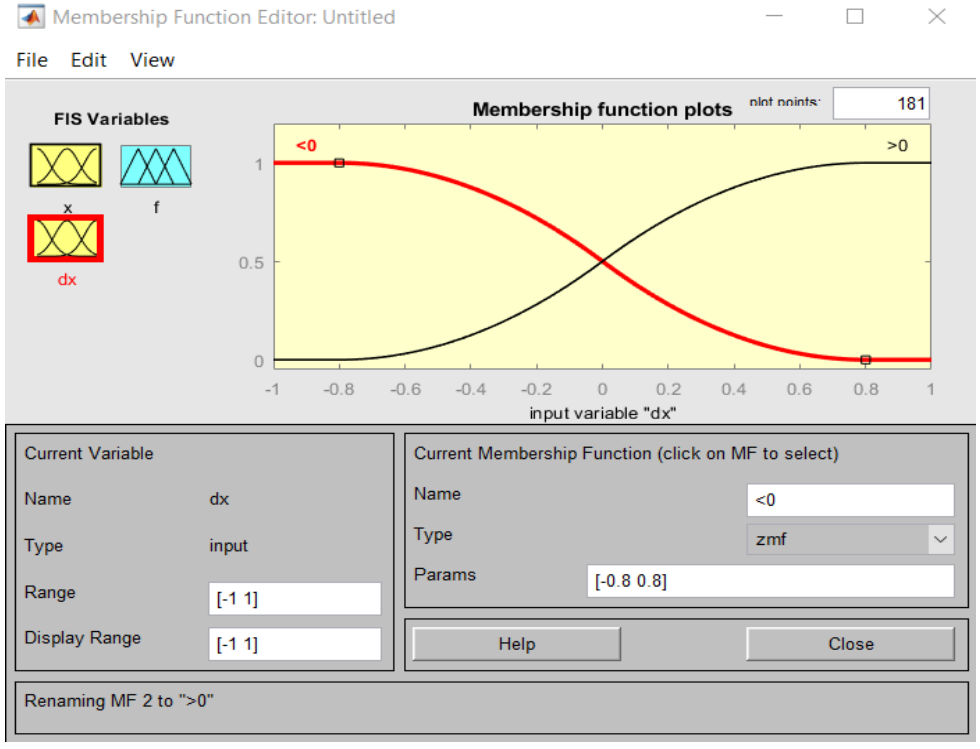

Рисунок 2.24 – Вікно задання вхідної змінної *dx*.

Вихід:

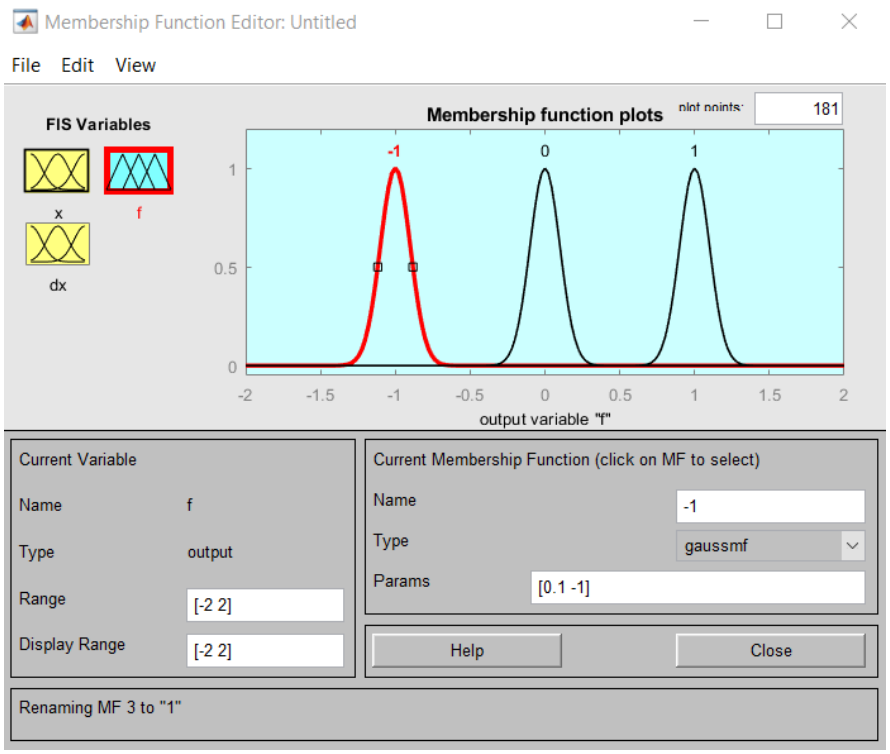

Рисунок 2.25 – Вікно задання вихідної змінної

Заповнюємо базу знань:

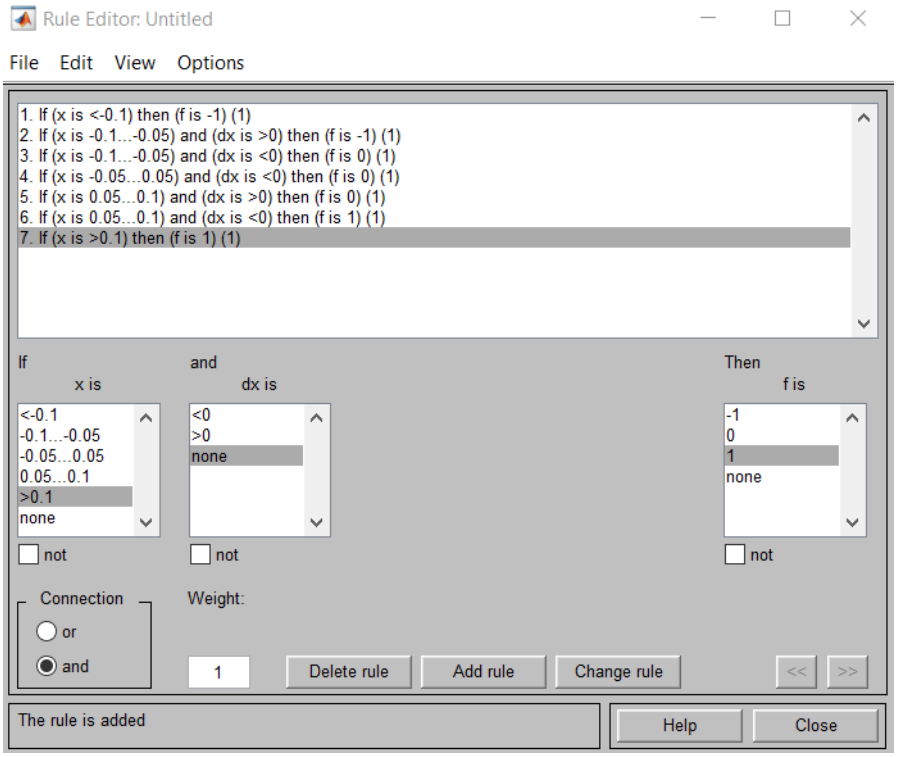

Рисунок 2.26 – Заповнення бази знань

Переносимо нечітку модель в simulink і порівнюємо результати з початковою моделлю:

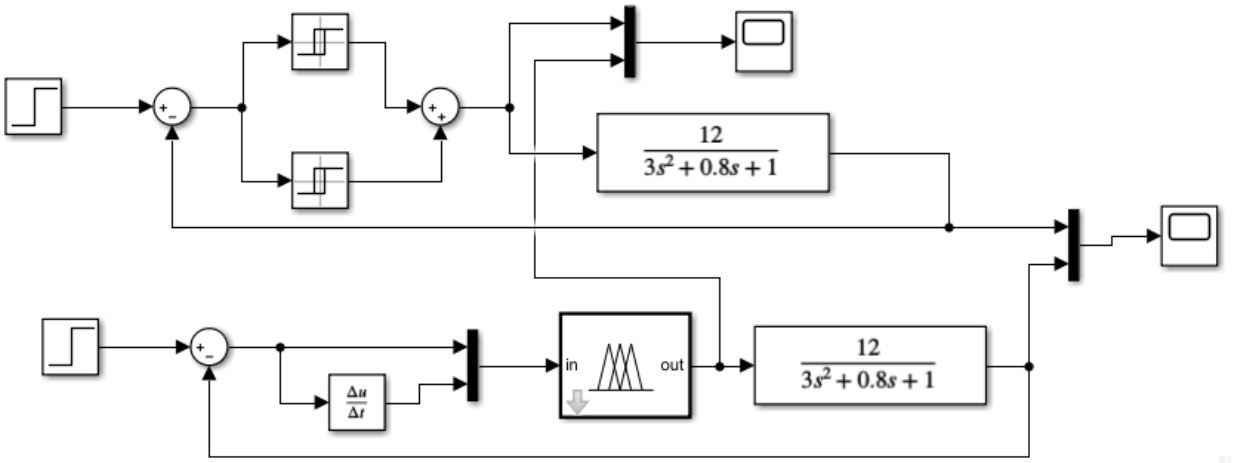

Рисунок 2.27 – Модель систем керування з класичним реле та його нечіткою моделлю

Виходи чіткої і нечіткої моделей:

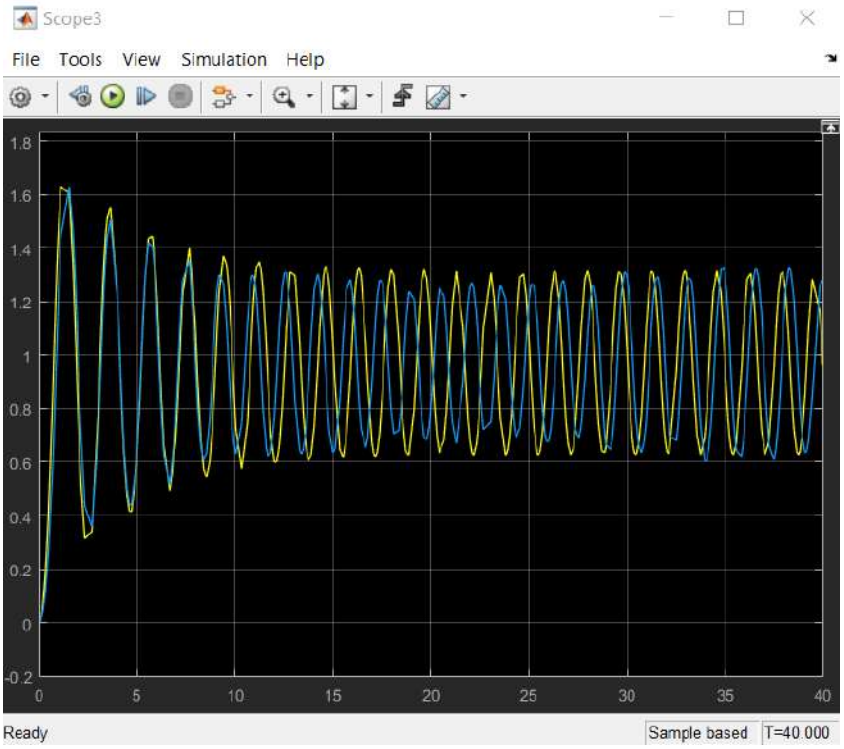

Рисунок 2.28 – Перехідний процес систем керування з класичним реле та його нечіткою моделлю

Керуюча дія (Scope 1):

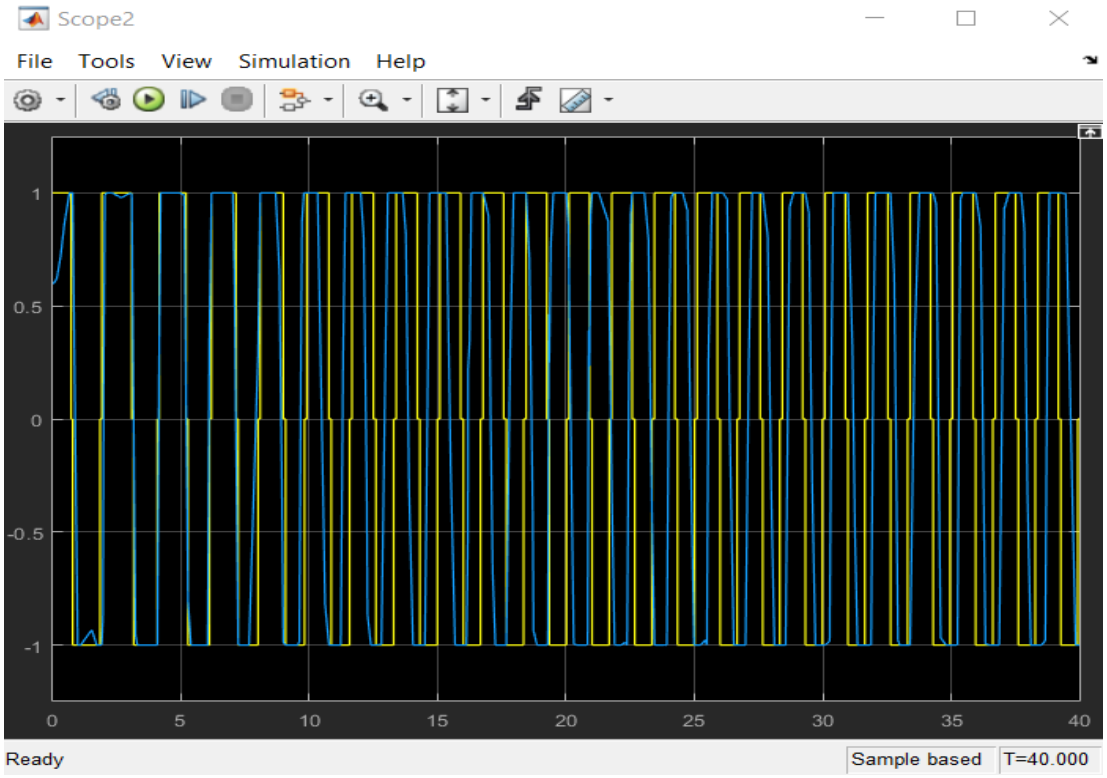

Рисунок 2.29 – Управляюча дія систем керування з класичним реле та його нечіткою моделлю

ЛАБОРАТОРНА РОБОТА №3

Моделювання системи керування з нечітким ПІД-регулятором

Мета роботи: Засобами нечіткої логіки виконати моделювання ПІД-регулятора. На практиці провести дослідження нечітких регуляторів та порівняти з традиційними.

Короткі теоретичні відомості

Пропорційно-інтегрально-диференціальний (ПІД) закон регулювання є найпростішим алгоритмом функціонування автоматичного регулятора. Він включає вплив усіх законів керування: пропорційного, інтегрального та диференційного. Розглянемо кожен з цих компонентів:

1. **Пропорційний (П)**: При пропорційному управлінні положення регулюючого органу задається пропорційно до поточної помилки між виміряною величиною і заданою величиною. Це дозволяє швидко реагувати на зміни, але може призвести до перевищення чи промаху.

2. **Інтегральний (І)**: Інтегральний компонент враховує накопичену помилку в часі. Він додає або віднімає величину, пропорційну сумі помилок, що виникали раніше. Це допомагає зменшити статичну помилку і забезпечити точність регулювання.

3. **Диференційний (Д)**: Диференційний компонент враховує швидкість зміни помилки. Він додає або віднімає величину, пропорційну похідній від помилки. Це допомагає уникнути перевищення і покращити стійкість системи.

В Японії частка ПІД-контролерів серед усіх засобів керування, що застосовуються на практиці, становить 84%. Це підтверджує, що ПІД-контролери придатні для багатьох практичних завдань завдяки простоті їх структури та принципів роботи.

Нечітка логіка в регуляторах є важливим інструментом для управління різноманітними системами. Вона використовується в комерційних системах для різних завдань, таких як:

1. Наведення телекамер під час спортивних подій. Нечіткі регулятори дозволяють точно визначити положення та кут телекамер для оптимального кадру під час трансляції спортивних подій.

2. Системи кондиціонування повітря. Нечіткі регулятори можуть ефективно керувати параметрами кондиціонерів, забезпечуючи комфортні умови в приміщеннях[1].

3. Управління автомобільними двигунами. В сучасних автомобілях нечіткі регулятори допомагають оптимізувати роботу двигунів, забезпечуючи ефективність та економію пального.

У 1974 році математик Ебрахім Мамдані запропонував **алгоритм Мамдані** [27], який використовує нечітку логіку для побудови систем керування динамічними об'єктами. Через рік вийшла публікація [28], в якій описано нечіткий ПІ-регулятор і його застосування для керування парогенератором. З того часу галузь застосування нечітких регуляторів постійно розширюється, збільшується різноманітність їх структур і виконуваних функцій. Нечітка логіка в ПІД-регуляторах використовується переважно двома шляхами: для побудови самого регулятора та для організації налаштування коефіцієнтів ПІД-регулятора. Обидва шляхи можуть використовуватися в ПІД-регуляторі одночасно. Одна з найбільш поширених структур нечіткого регулятора (нечіткого ПІ-регулятора) показана на рисунку 3.1.

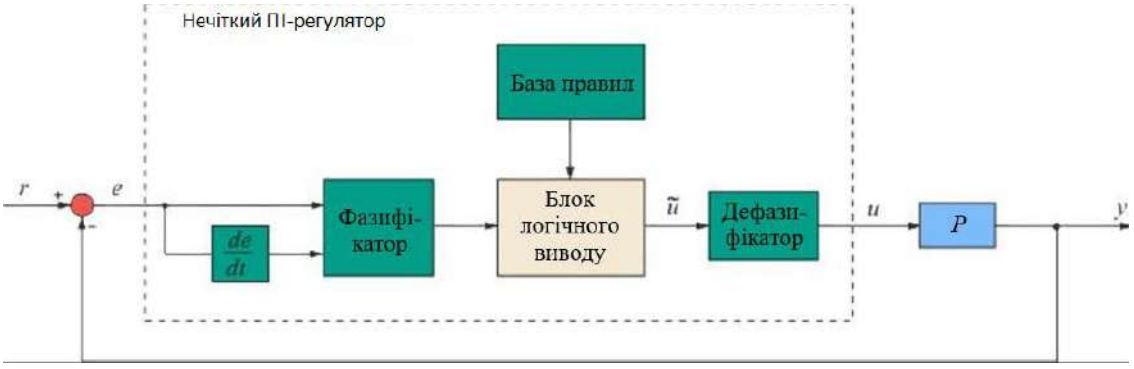

Рисунок 3.1 – Структура нечіткого ПІ-регулятора

Порядок виконання роботи

1. Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи. Далі потрібно синтезувати ПІД регулятор:

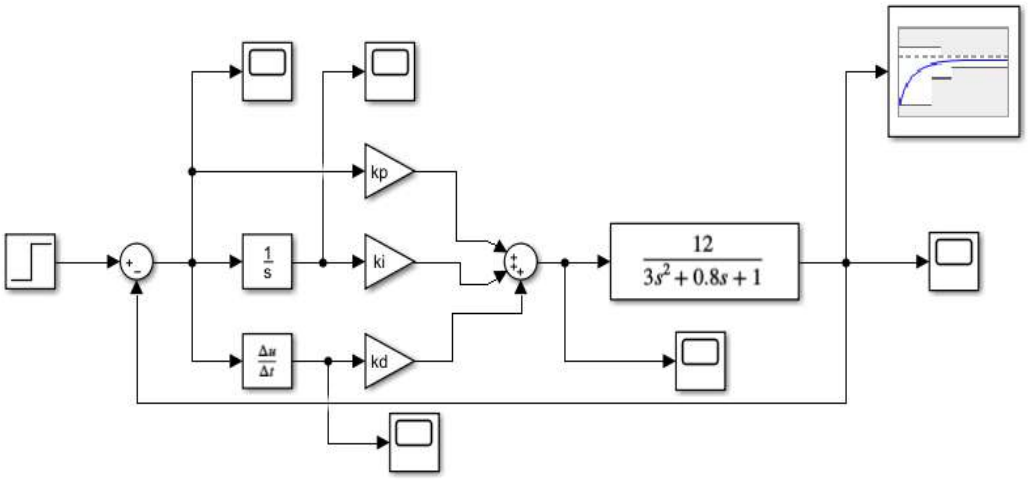

Рисунок 3.2 – Модель системи керування з ПІД-регулятором

Для роботи з утилітою NCD Outport задайте в робочій області Workspace початкові значення параметрів ПІД регулятора:

```
>>  kp=1;
>>  ki=1;
>>  kd=1;
```

Подвійний клік по NCD Outport відкриває наступне вікно:

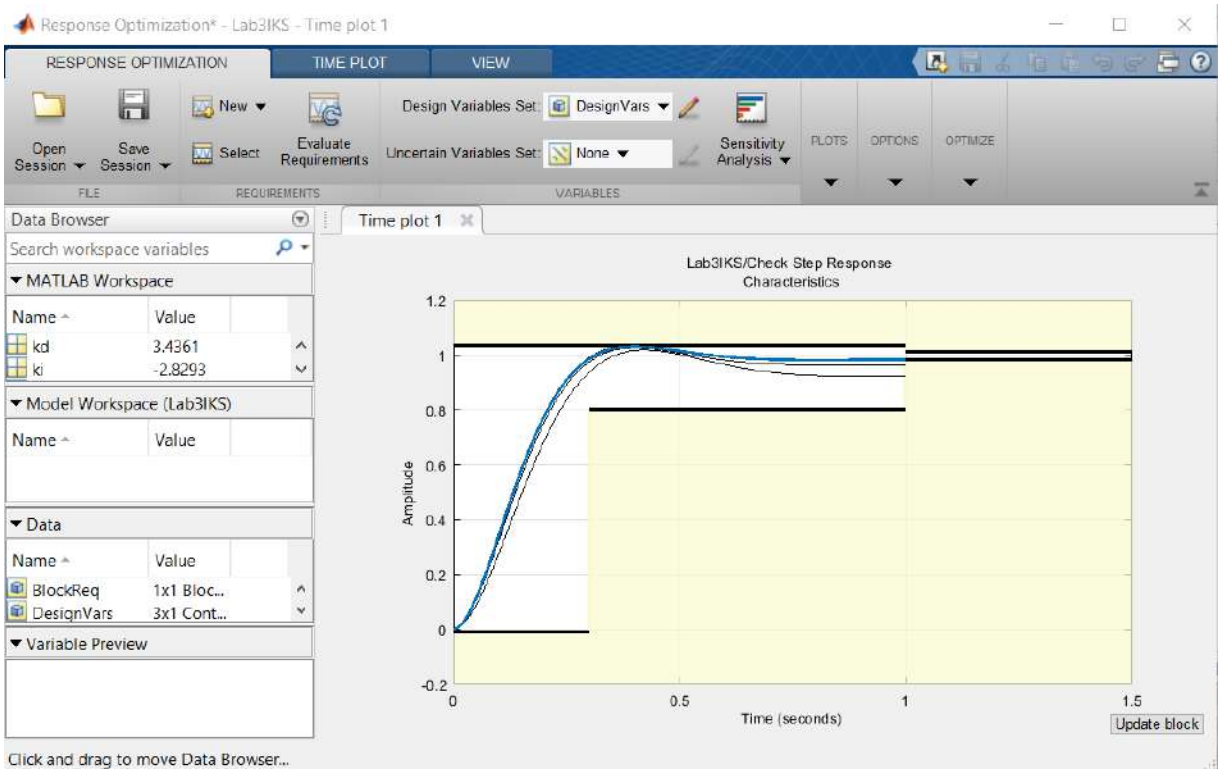

Рисунок 3.3 – Вікно налаштувань NCD Outport

Перетягуючи червоні «обмежувачі» задайте необхідні параметри перехідного процесу (час встановлення, час регулювання, перерегулювання). Після цього алгоритму NCD Outport необхідно вказати, які параметри необхідно змінювати, щоб перехідний процес системи розташувався у вибраних межах. Тому виберіть в меню Optimization ->Parametrs і в полі Tunable Variable, через пропуск вкажіть kp, ki, kd як показано на рисунку 3.4.

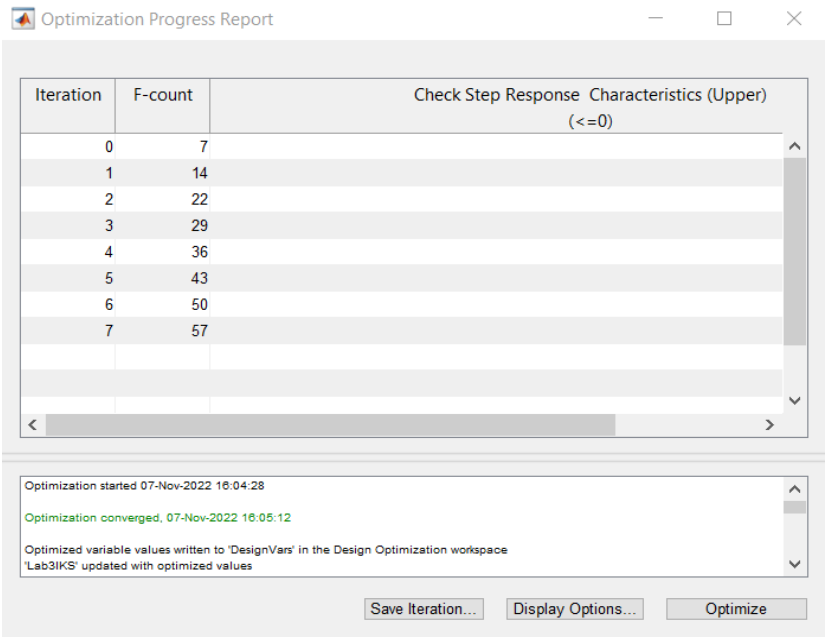

Рисунок 3.4 – Вікно для задання параметрів оптимізації Кнопкою Done застосуйте введені змінні.

Кнопка Start починає процес підбору коефіцієнтів ПІД-регулятора.

2. За отриманими графіками вхідних величин визначаємо лінгвістичні змінні:

Пропорційна гілка (Scope2):

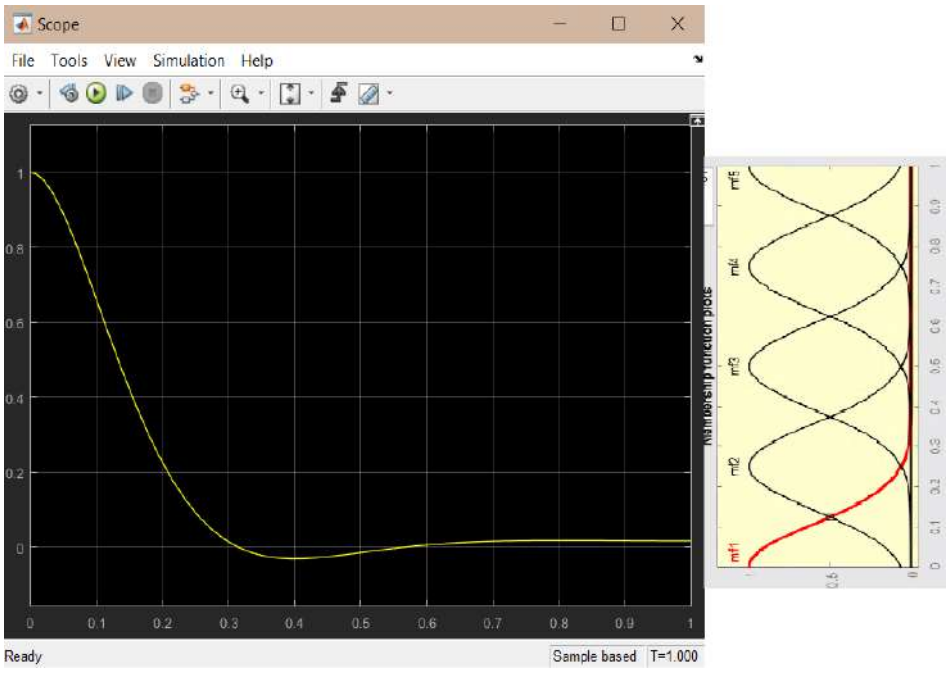

Рисунок 2.5 – Визначення лінгвістичної змінної за графіком пропорційної гілки

Інтегральна гілка (Scope4):

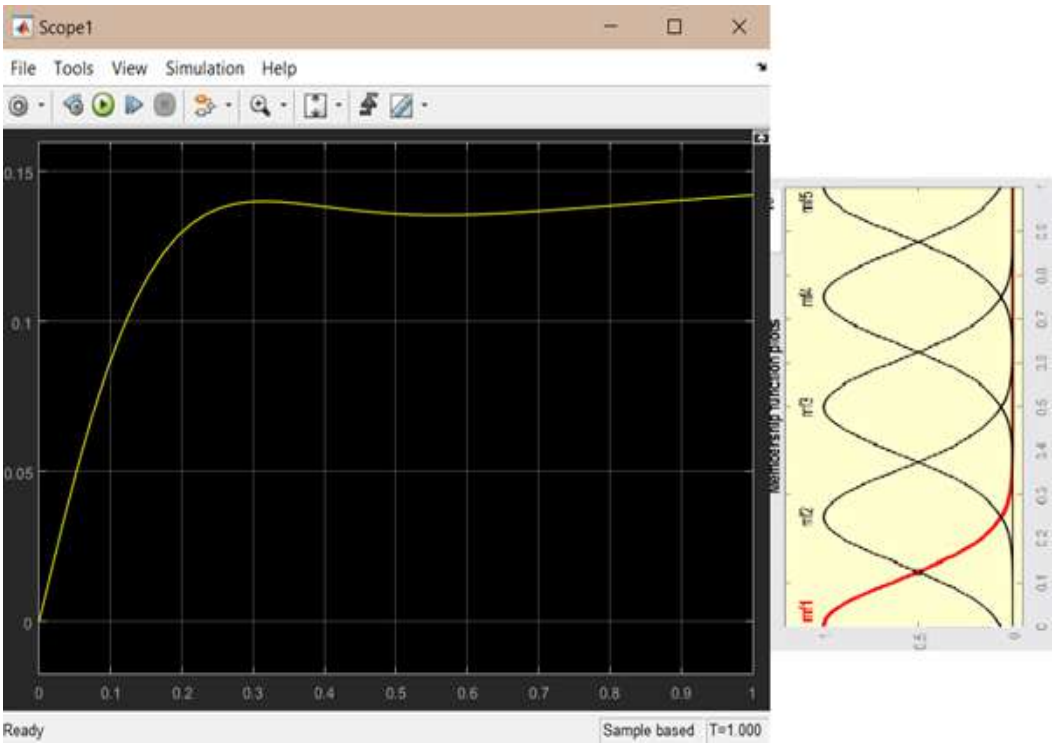

Рисунок 2.6 – Визначення лінгвістичної змінної за графіком інтегральної гілки

Диференційна гілка(Scope5):

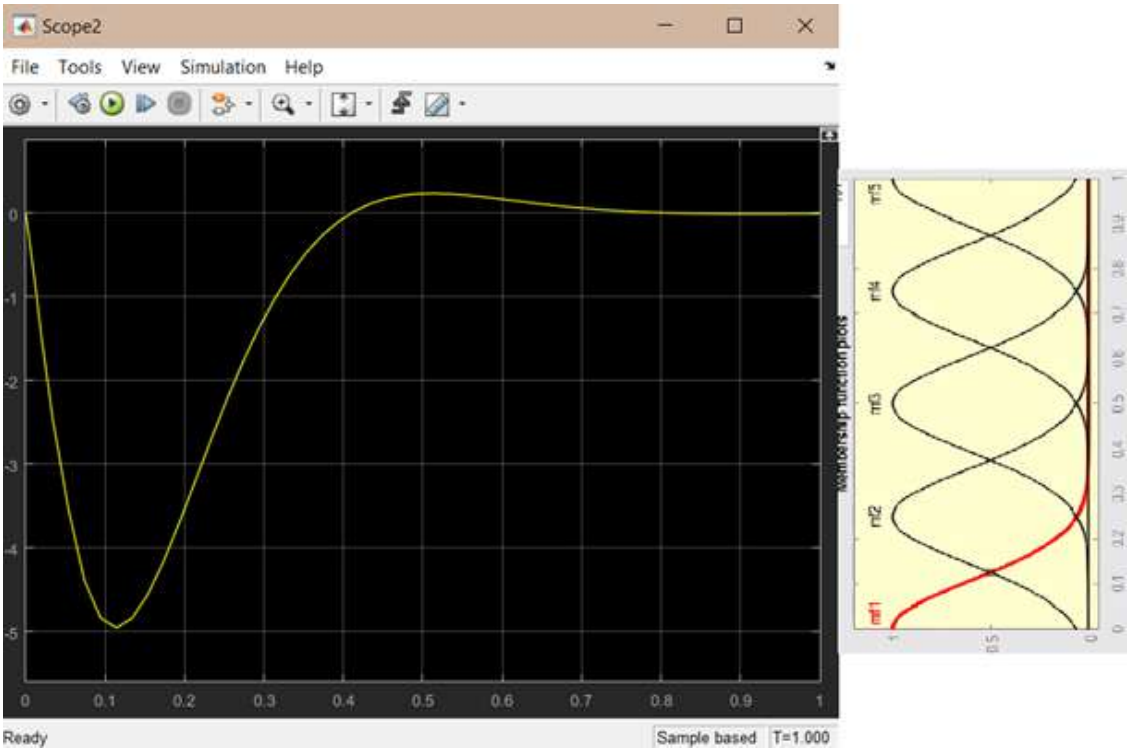

Рисунок 2.7 – Визначення лінгвістичної змінної за графіком диференційної гілки

Вихід ПІД(Scope3):

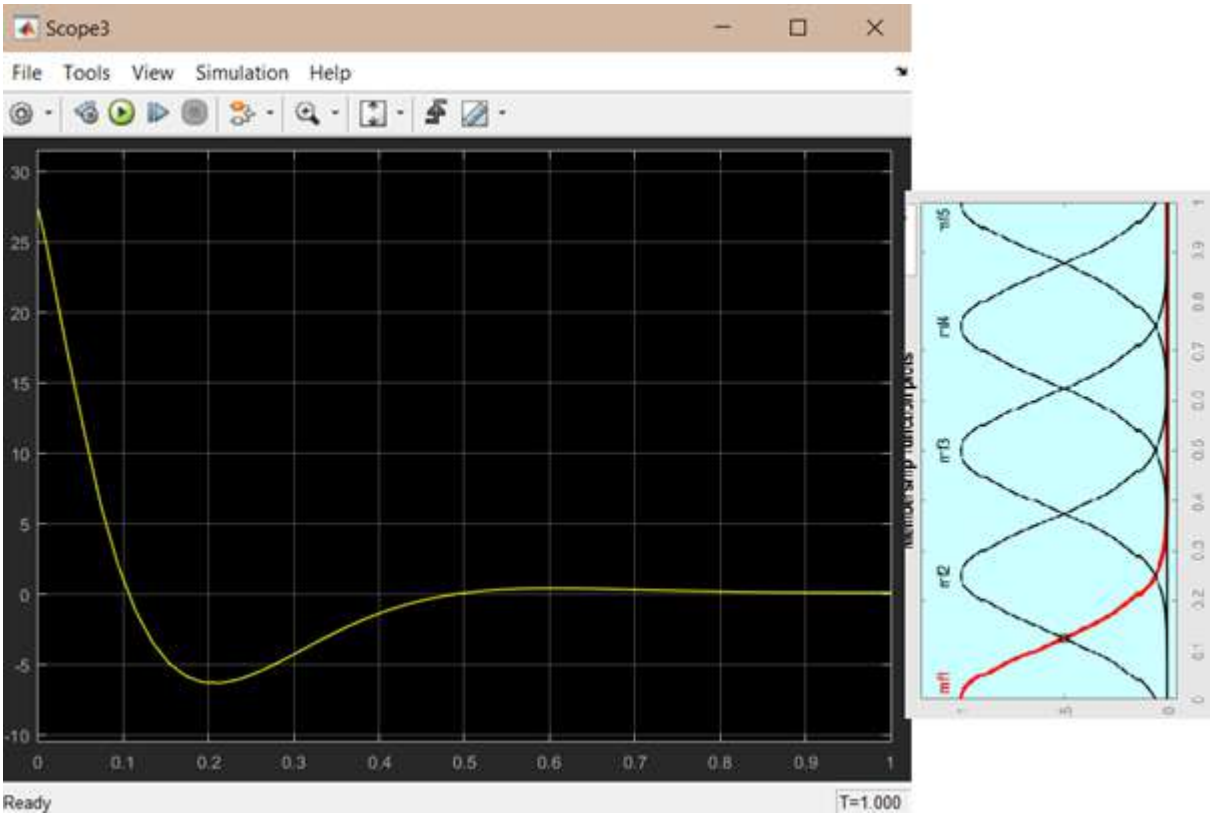

Рисунок 3.8 – Визначення лінгвістичної змінної за вихідним графіком ПІД

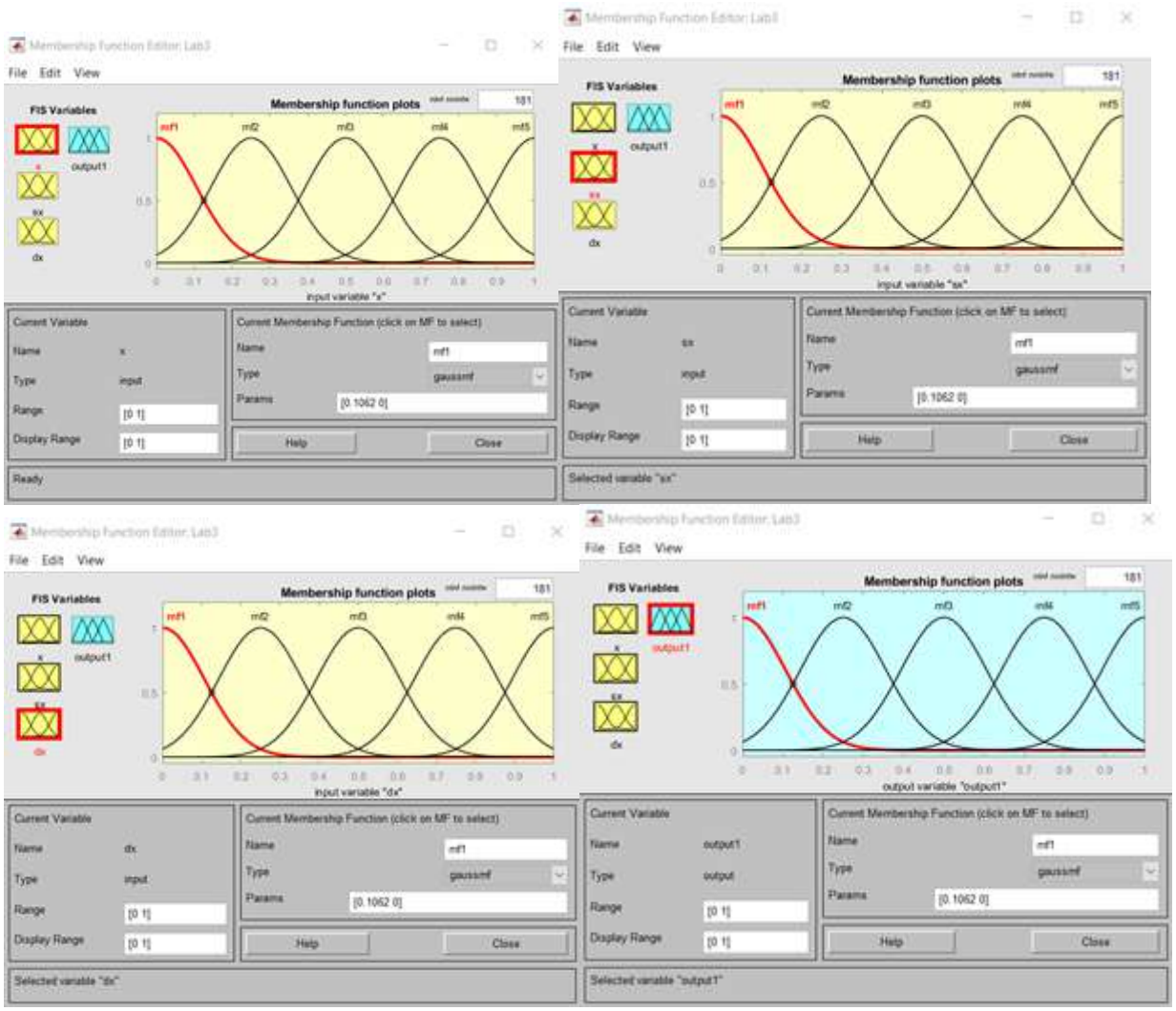

Рисунок 3.9-3.12 – Побудова лінгвістичних змінних в fuzzy logic toolbox

3. Заповнення бази знань.

Для заповнення бази знань необхідно скористатися способом «зовнішніх зрізів». Він полягає в наступному. Вибирають декілька тимчасових зрізів характеристики і знаходять яка функція приналежності відповідає вибраному моменту часу. Таким чином, комбінація вхідних і вихідних функцій приналежності дозволяє отримати правило для бази знань. Наприклад (дивися рисунки 3.4, 3.5, 3.6, 3.7). У момент часу 0.2 з маємо: пропорційну гілку – mf2; інтегральна – mf4; диференціальна – mf4; вихід ПІД-регулятора – mf1. Тоді правило матиме вигляд: if(x is mf2) and(sx is mf4) and (dx is mf4) then(output ismf1).

Так само будують від 8 до 20 правил залежно від монотонності характеристики.

1. If (x is mf5) and (sx is mf1) and (dx is mf5) then (output1 is mf5) (1)
2. If (x is mf5) and (sx is mf1) and (dx is mf4) then (output1 is mf4) (1)
3. If (x is mf5) and (sx is mf1) and (dx is mf3) then (output1 is mf3) (1)
4. If (x is mf5) and (sx is mf1) and (dx is mf2) then (output1 is mf2) (1)
5. If (x is mf4) and (sx is mf2) and (dx is mf1) then (output1 is mf1) (1)
6. If (x is mf3) and (sx is mf3) and (dx is mf2) then (output1 is mf1) (1)
7. If (x is mf2) and (sx is mf4) and (dx is mf3) then (output1 is mf1) (1)
8. If (x is mf1) and (sx is mf5) and (dx is mf4) then (output1 is mf1) (1)
9. If (x is mf1) and (sx is mf5) and (dx is mf5) then (output1 is mf1) (1)

4 Експортуємо модель в simulink і порівнюємо вихід системи керування з чіткою моделлю та нечіткою:

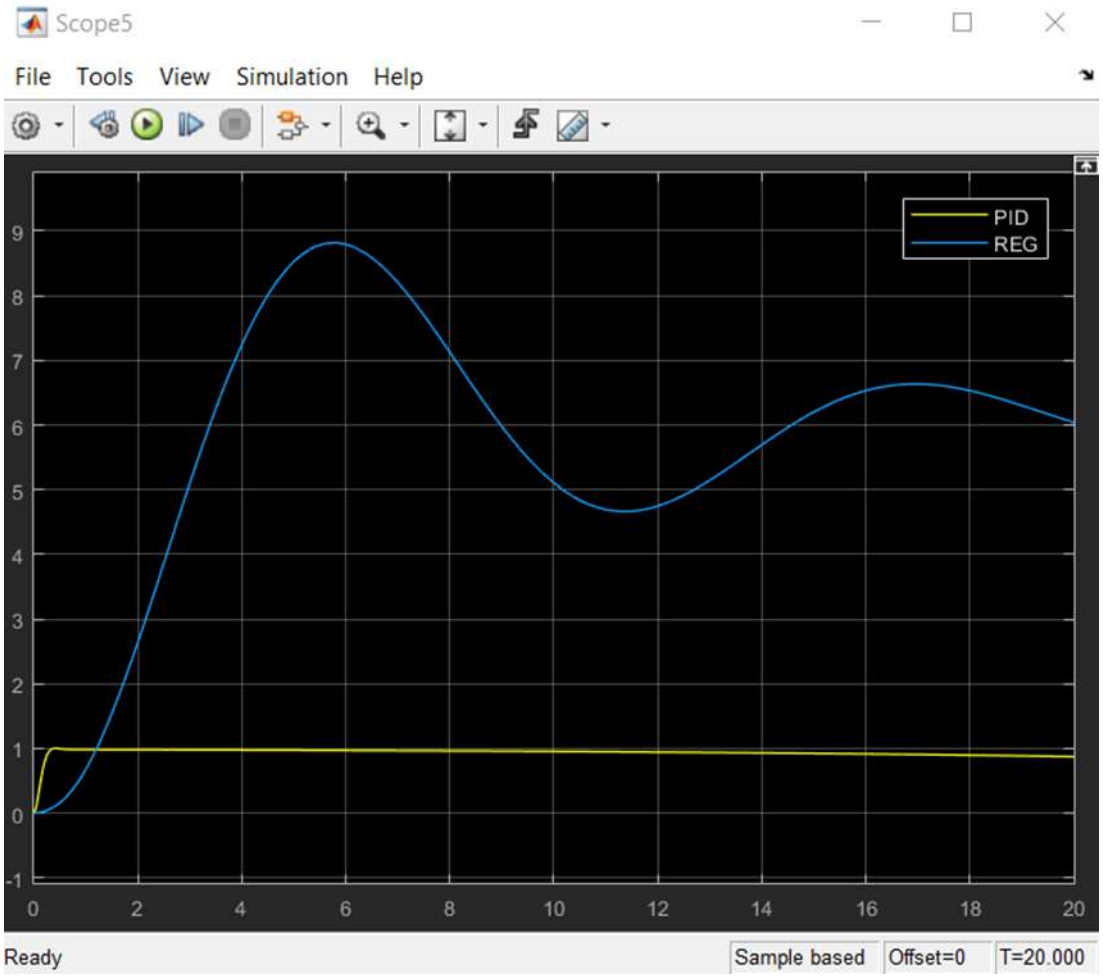

Рисунок 3.13 – Вихід системи керування з чіткою моделлю та нечіткою

ЛАБОРАТОРНА РОБОТА №4

Моделювання системи керування з нечітким регулятором змінної структури

Мета роботи: Засобами нечіткої логіки провести моделювання регуляторів змінної структури. На практиці провести дослідження нечітких регуляторів та порівняти з традиційними.

Порядок виконання роботи

Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи. Далі необхідно синтезувати систему, яка при одних значеннях помилки використовує алгоритм релейного керування, а при інших – ПІД закон.

Завдання зводиться до розмежування інтервалів роботи нечітких регуляторів при описі входів помилково в релейному і ПІД регуляторі.

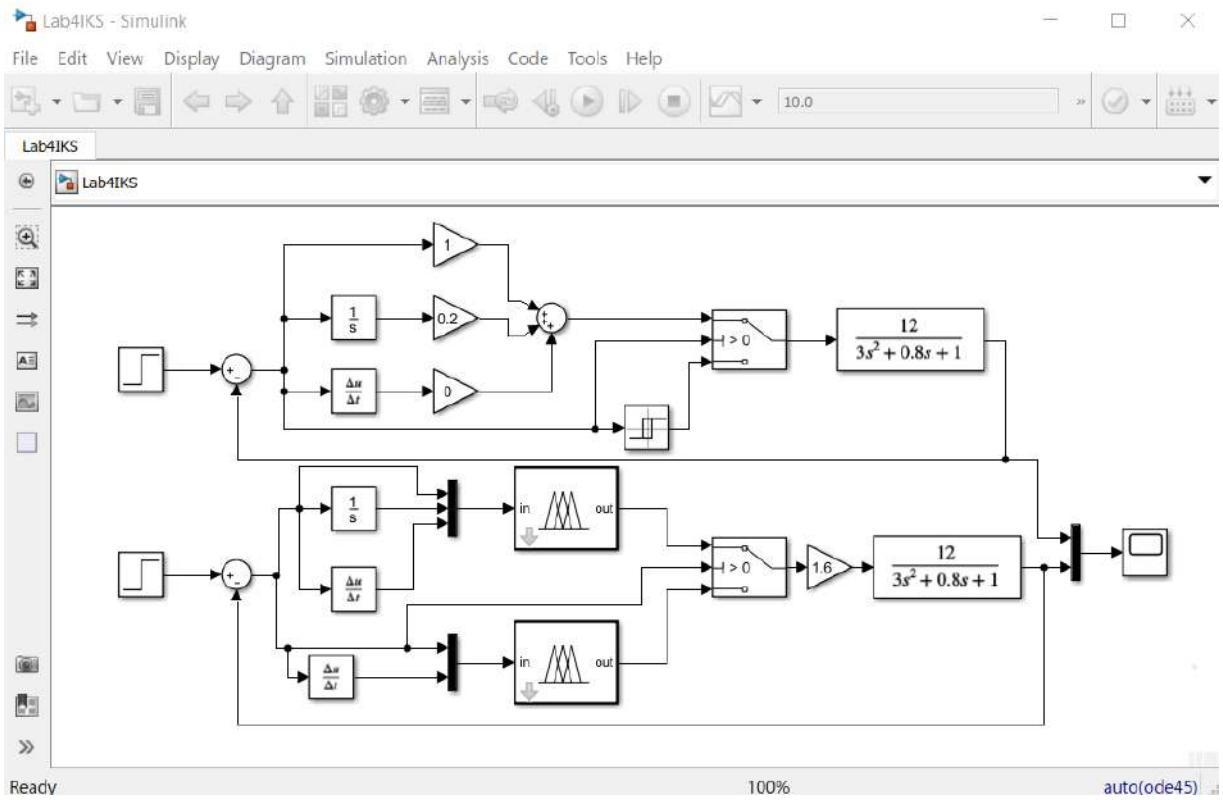

Рисунок 4.1 – структурна схема

Налаштовуємо блок реле:

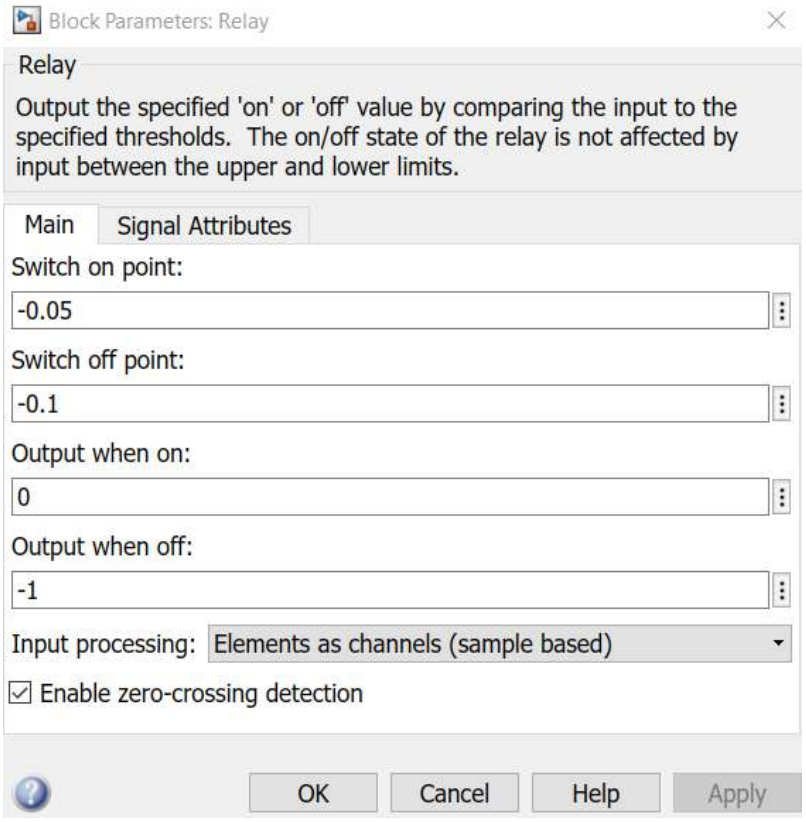

Рисунок 4.2 – налаштування блока реле

Будуємо моделі в fuzzy logic toolbox:

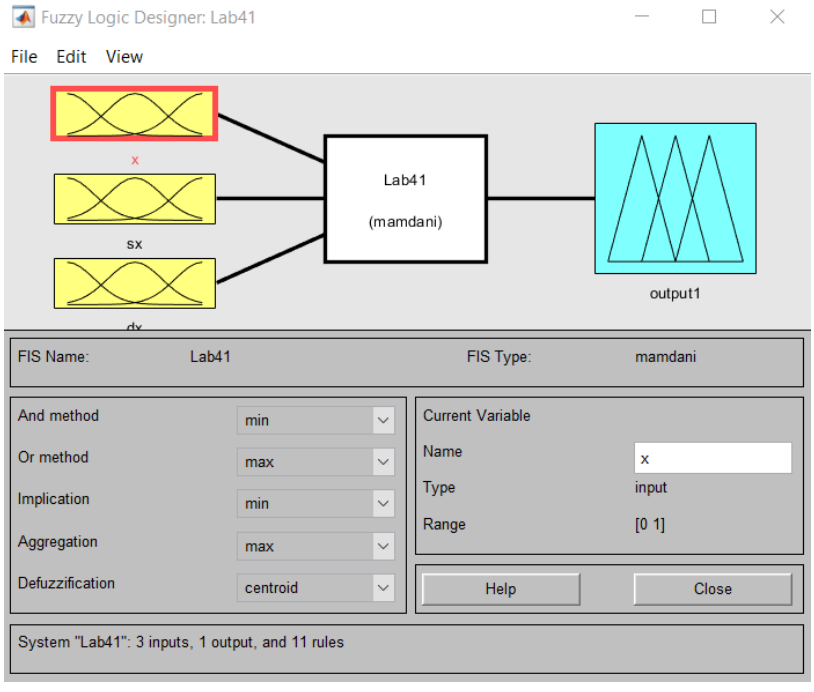

Рисунок 4.3 – моделі в fuzzy logic toolbox

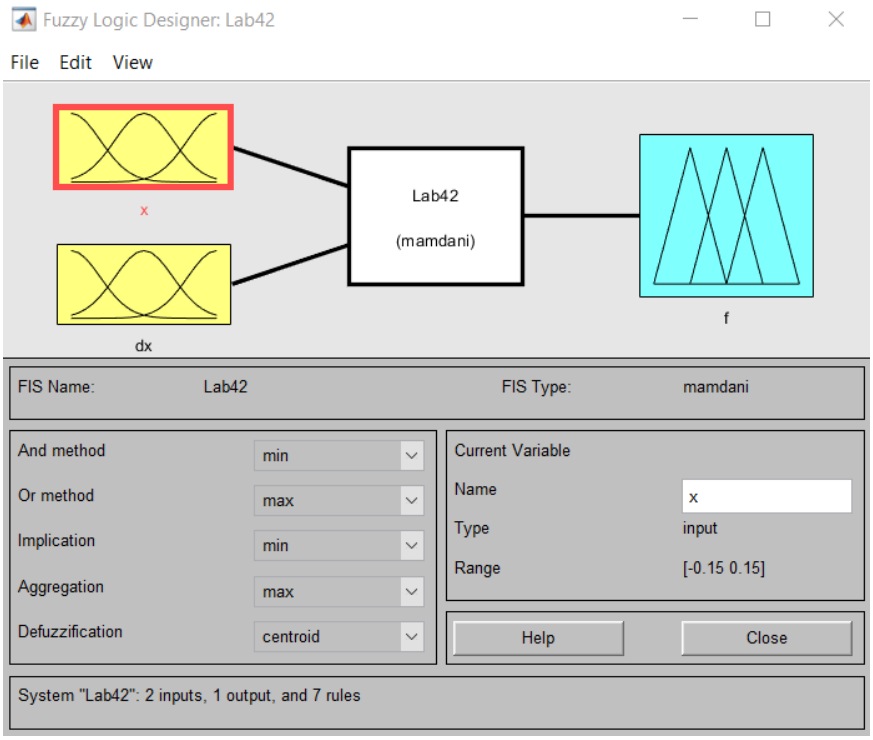

Рисунок 4.4 – моделі в fuzzy logic toolbox

Заповнюємо бази знань:

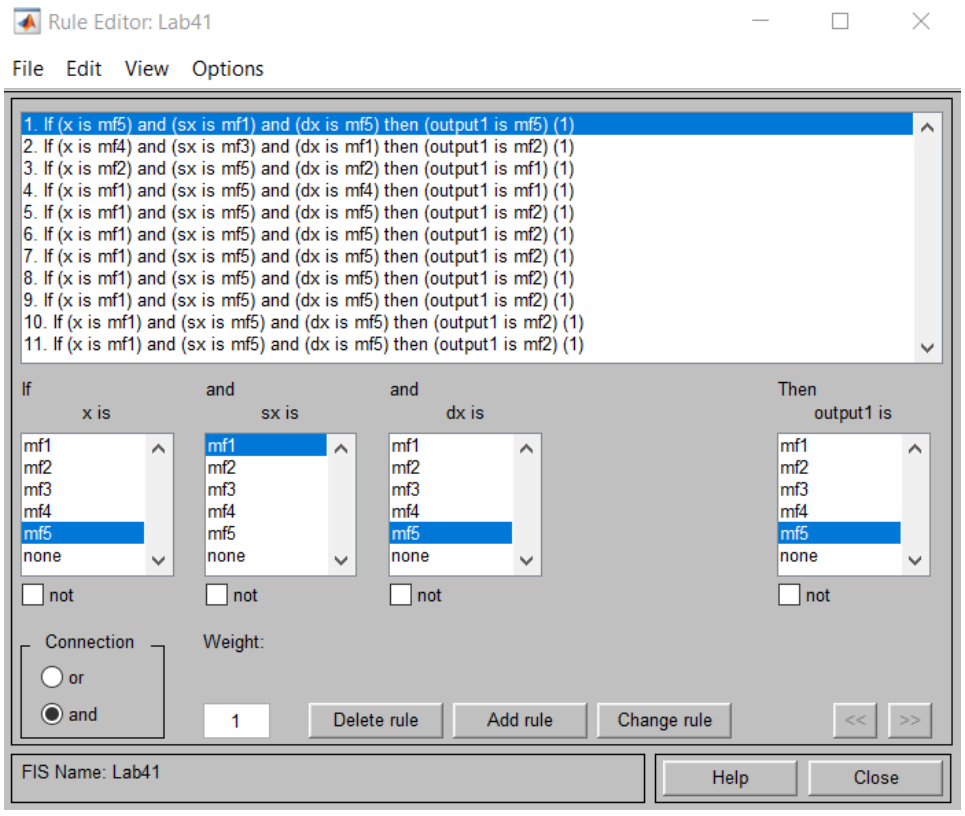

Рисунок 4.5 – Формування бази знань

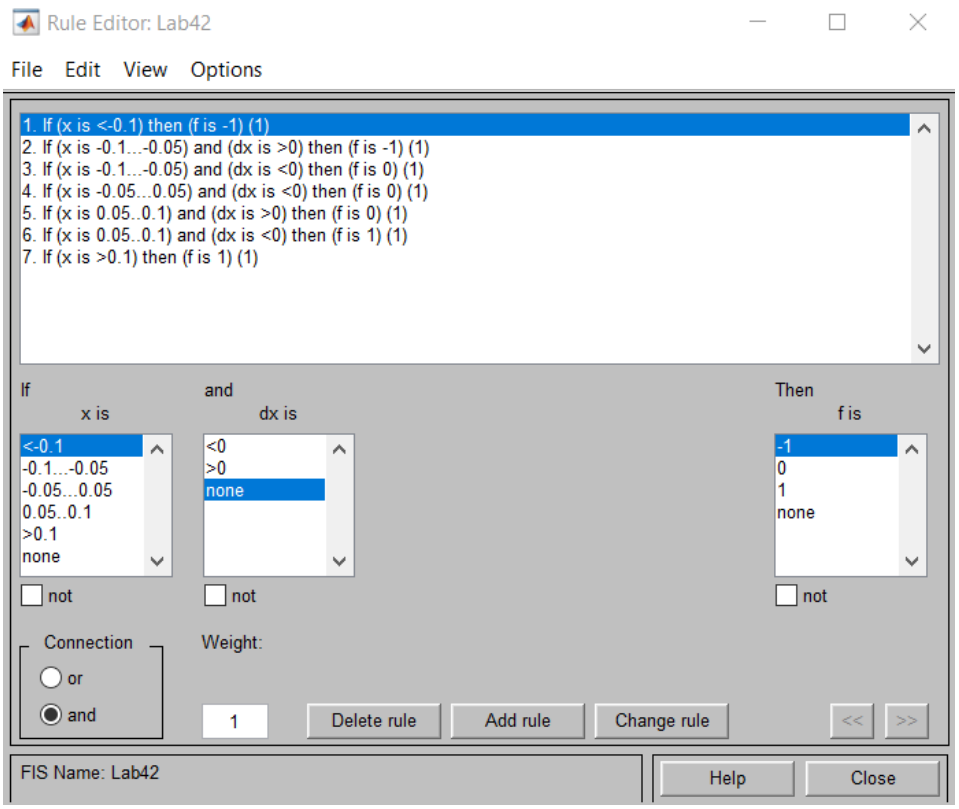

Рисунок 4.6 – Формування бази знань

Налаштовуємо блок реле:

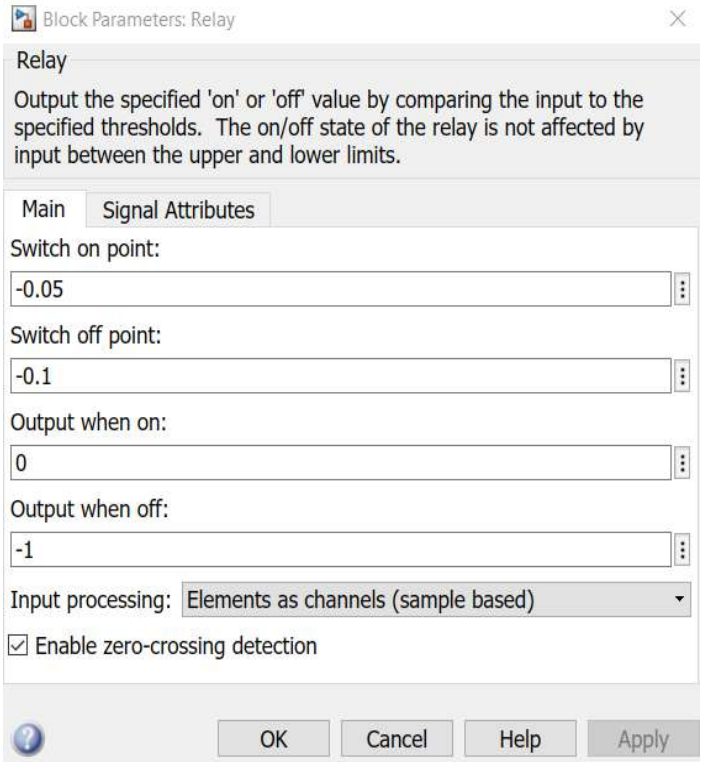

Рисунок 4.7 – Налаштування блоку реле

Будуємо моделі в fuzzy logic toolbox:

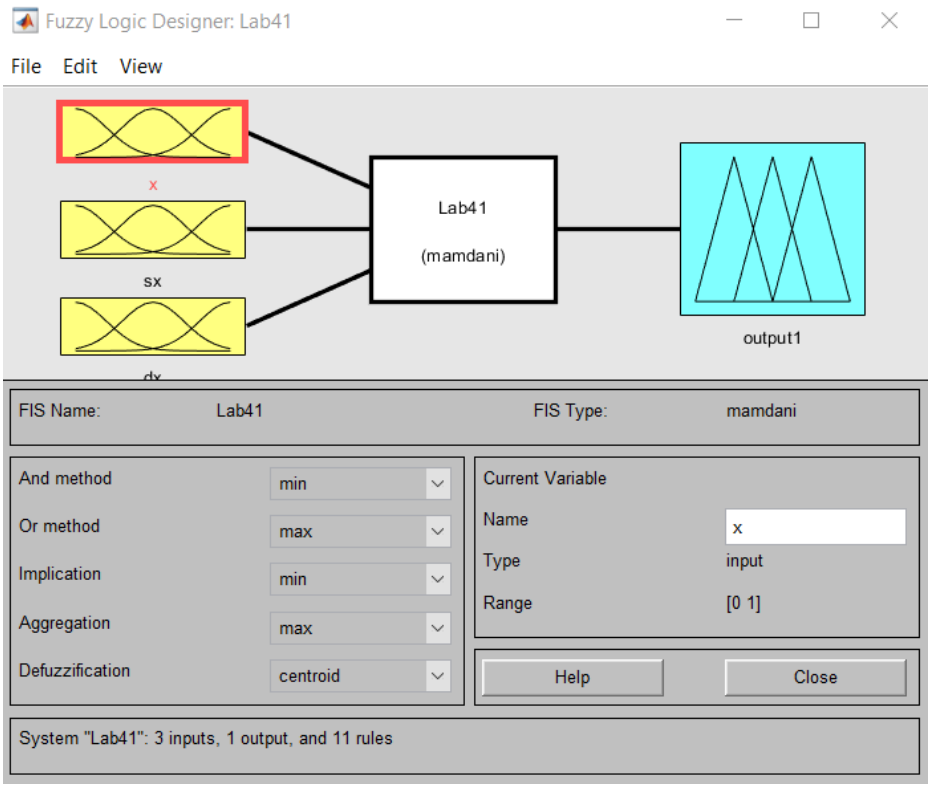

Рисунок 4.8 – Побудова моделі в fuzzy logic toolbox

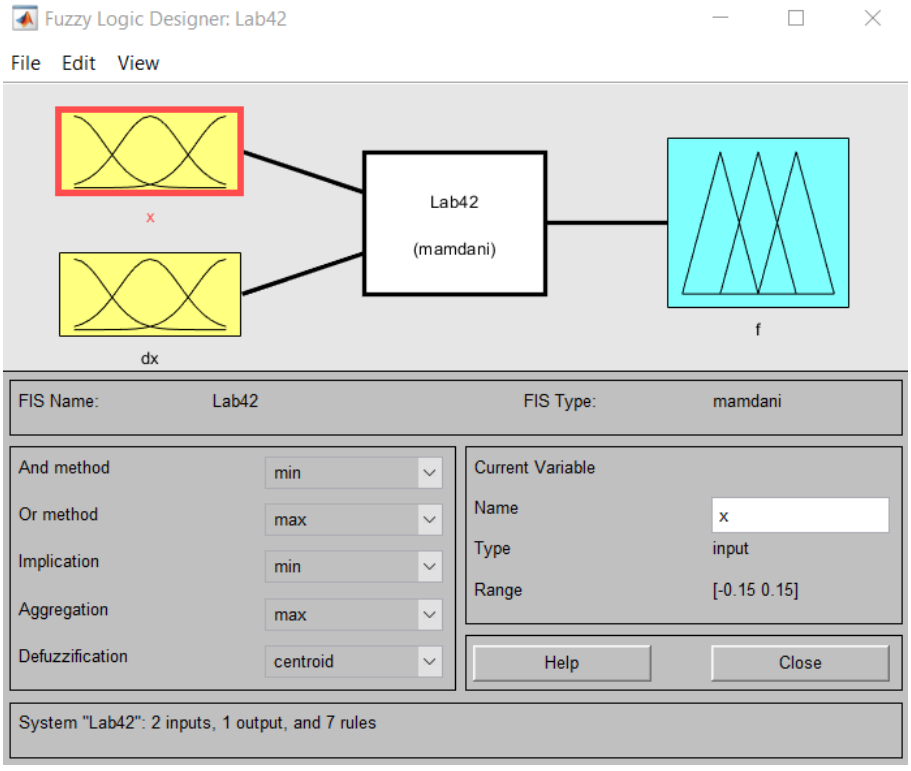

Рисунок 4.9 – Побудова моделі в fuzzy logic toolbox

Заповнюємо бази знань:

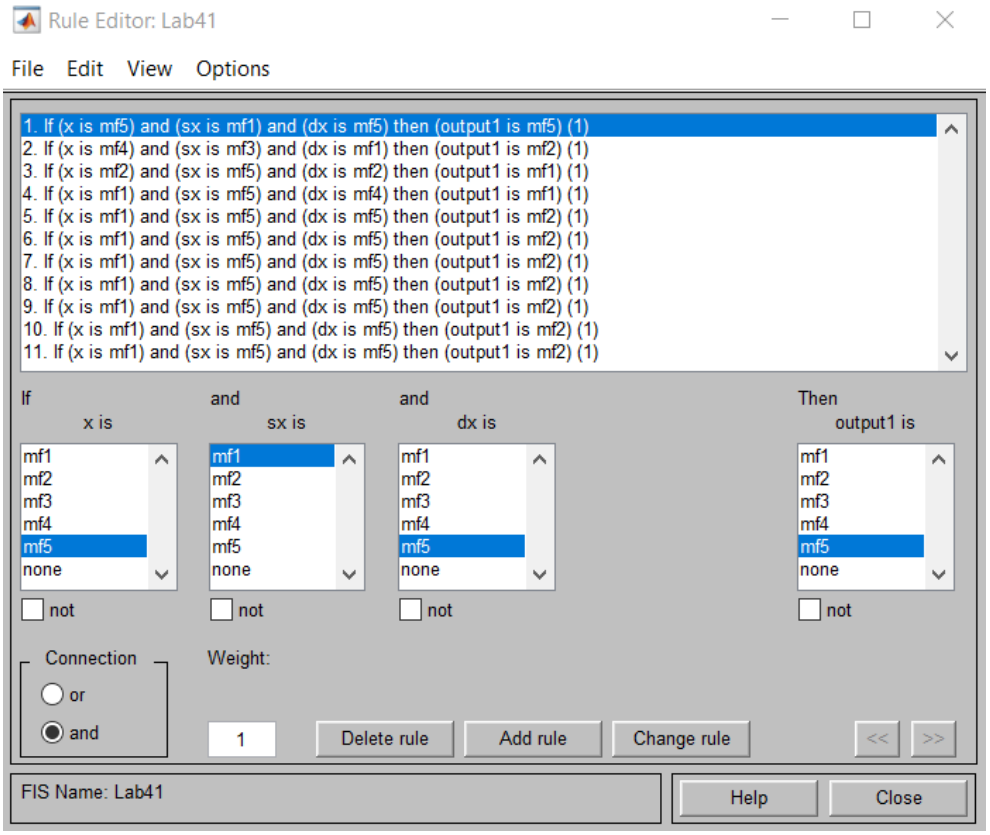

Рисунок 4.10 – Заповнення бази знань

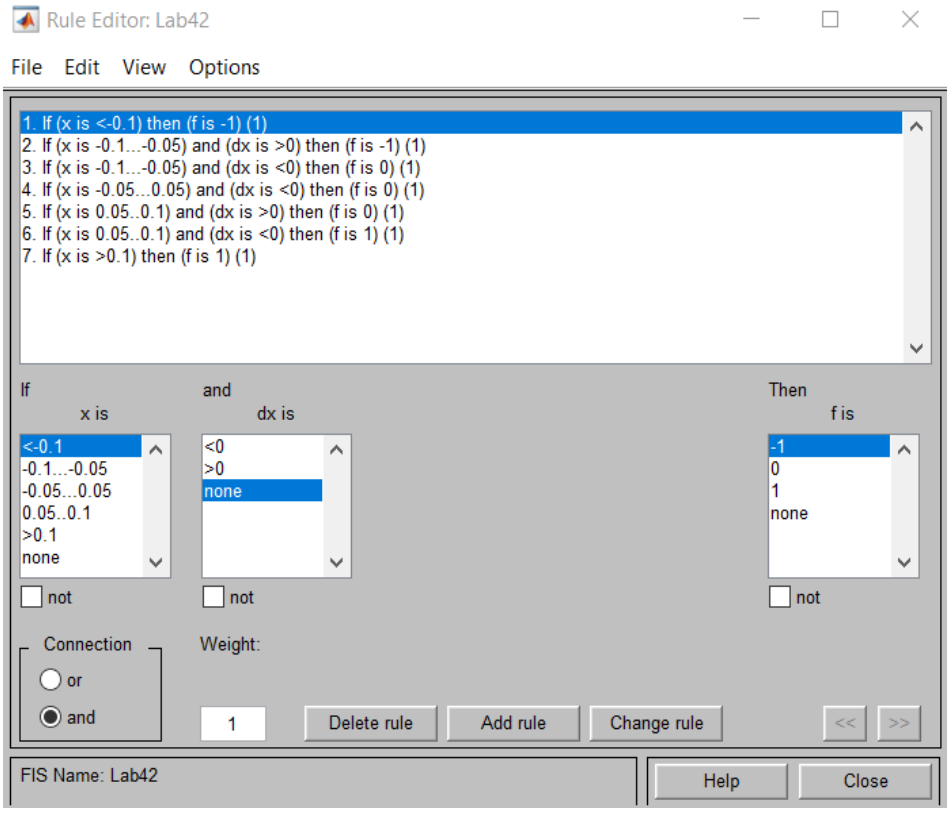

Рисунок 4.11 – Заповнення бази знань

Порівнюємо вихід системи керування з нечітким регулятором змінної структури та традиційним:

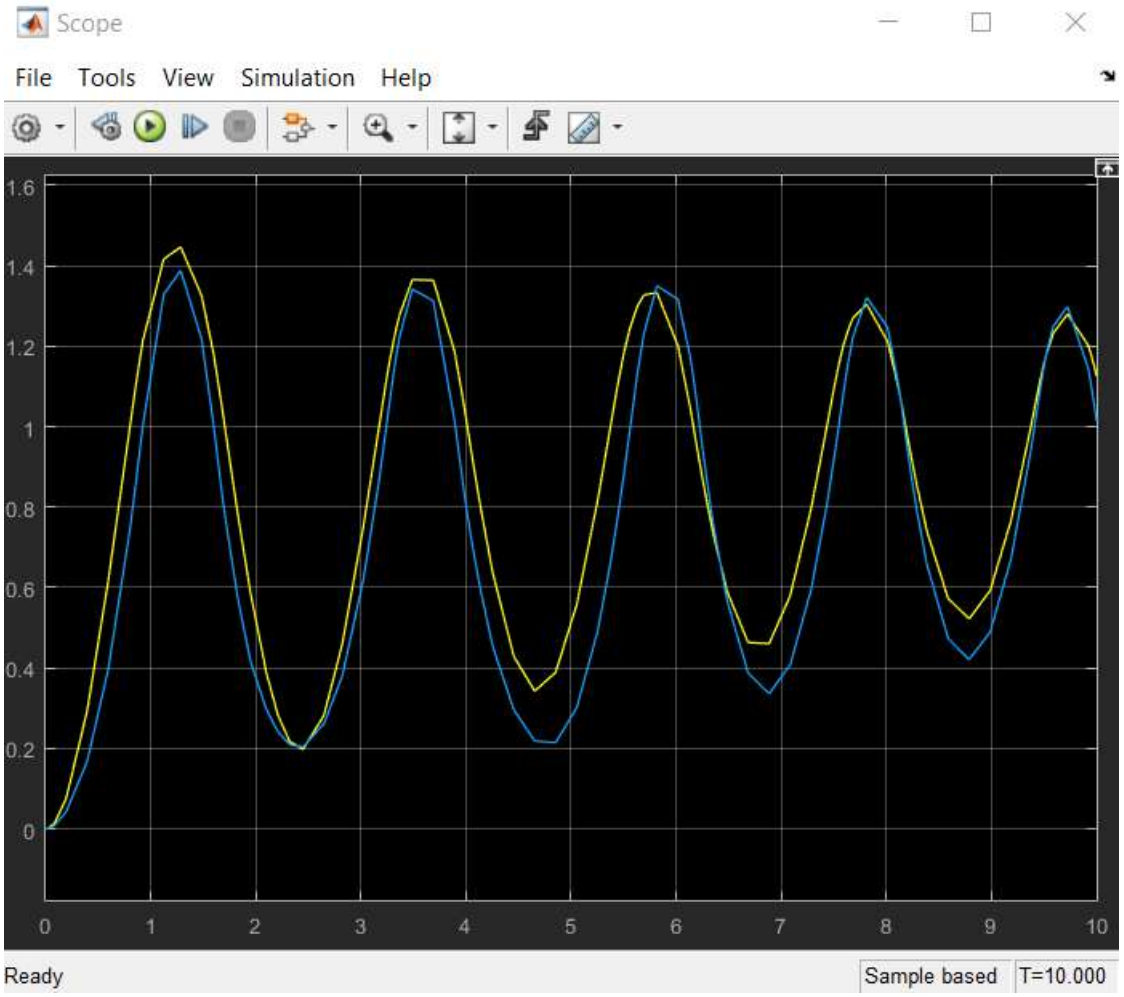

Рисунок 4.12 – Вихід системи керування з нечітким регулятором та традиційним

ЛАБОРАТОРНА РОБОТА №5

Синтез нечіткого регулятора по швидкості та прискоренню зміни похибки стану системи керування неперервним об'єктом

Мета роботи: Провести моделювання засобами нечіткої логіки регулятора по помилці швидкості і прискоренню її зміни. На практиці провести дослідження нечітких регуляторів та порівняти з традиційними.

Порядок виконання роботи

1. Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи.
2. Змоделювати в matlab систему керування неперервним об'єктом, як показано на рисунку 5.1, перехідний процес системи представлений на рисунку 5.2.

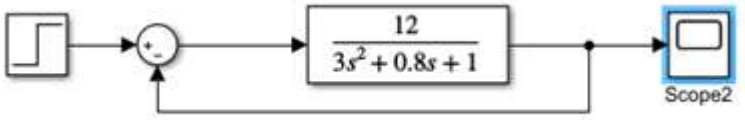

Рисунок 5.1 – Система керування неперервним об'єктом

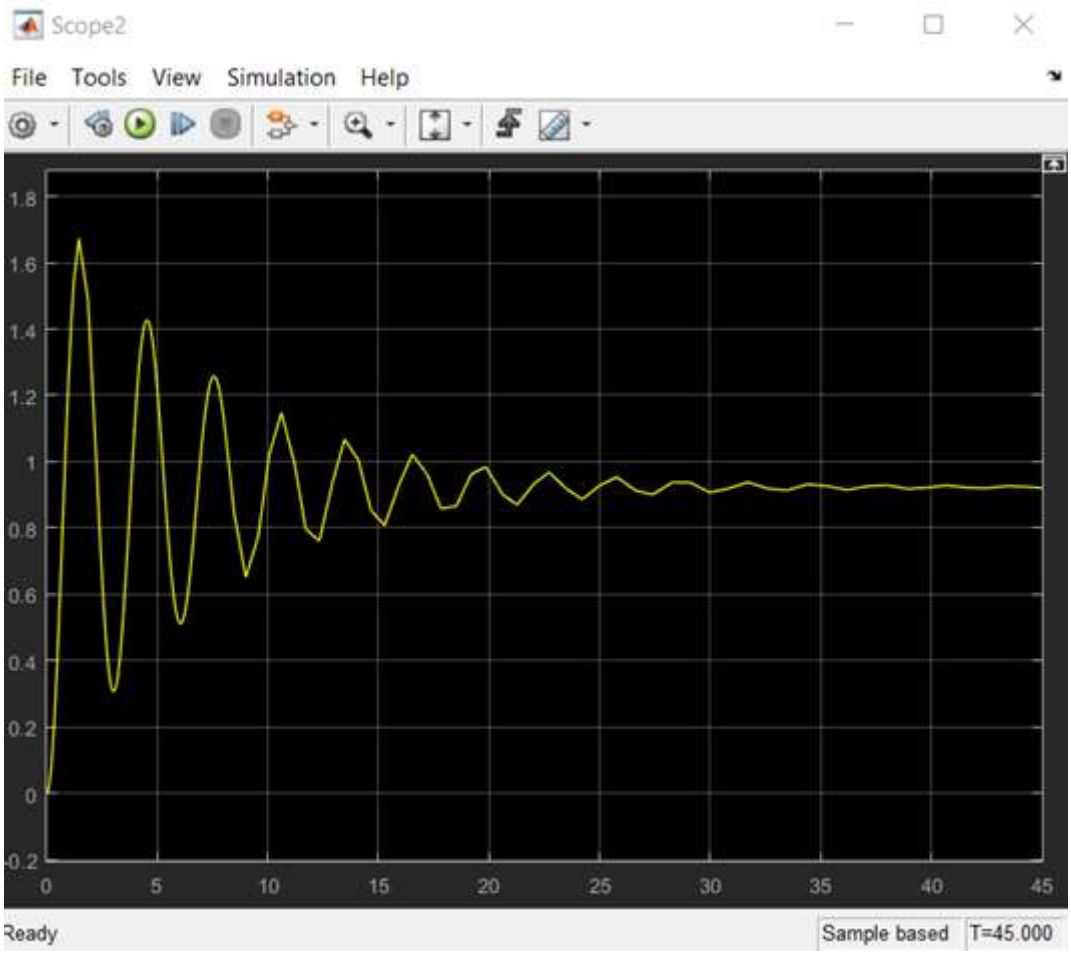

Рисунок 5.2 – Перехідний процес системи керування неперервним об'єктом

3. Побудувати структурну схему з нечітким регулятором по швидкості і прискоренню зміни помилки як показано на рисунку 5.3. Вхідними змінним нечіткого регулятора будуть помилка системи, її перша похідна ε’ (швидкість) і друга похідна ε’’ (прискорення).

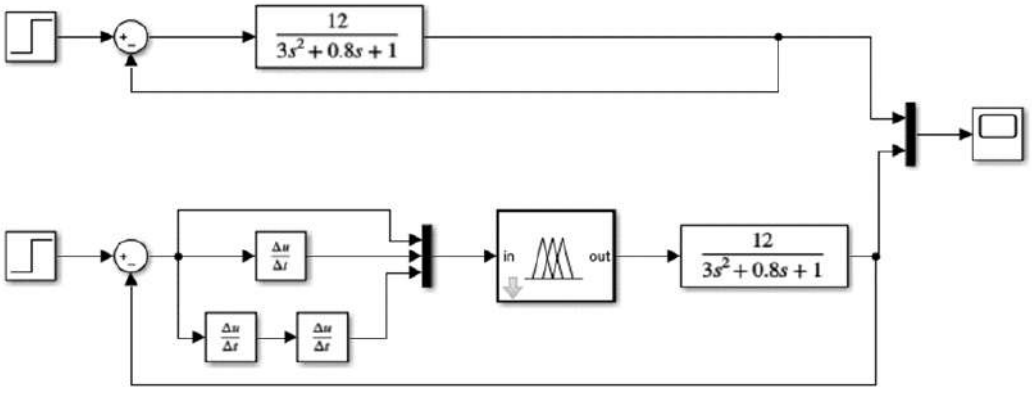

Рисунок 5.3 – Структурна схема з нечітким регулятором по швидкості і прискоренню зміни помилки

4. Синтезувати в fuzzy нечіткий регулятор. Нехай кількість функцій приналежності для усіх лінгвістичних змінних ε, *ε*’, *ε*’’, *U*) однаково – 2 функції приналежності (">0" і "<0") і вони мають вигляд:

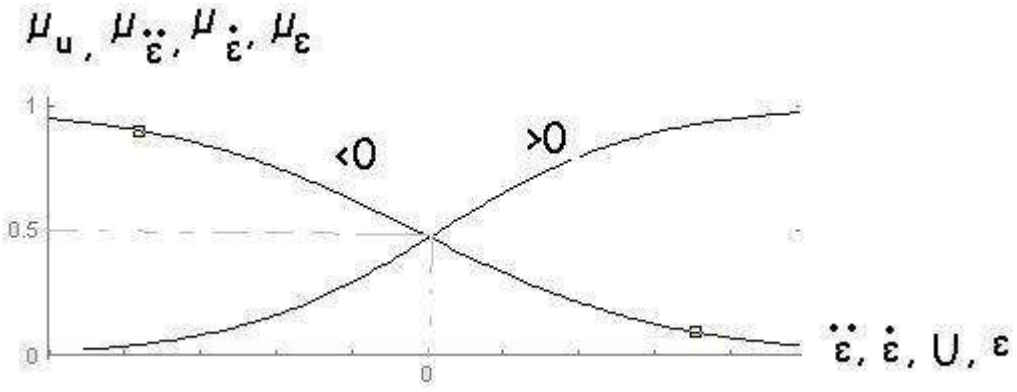

Рисунок 5.4 – Функції приналежності

Тоді нечітка модель складається як показано на рисунках 5.5-5.10.

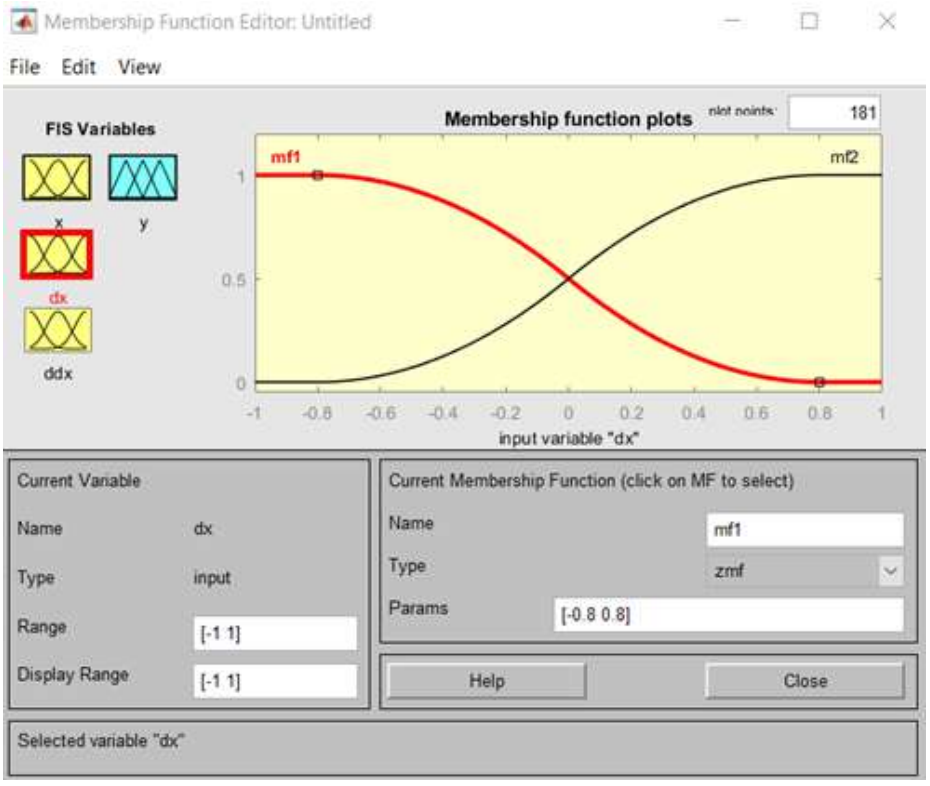

Рисунок 5.5 – Створення вхідних і вихідних змінних

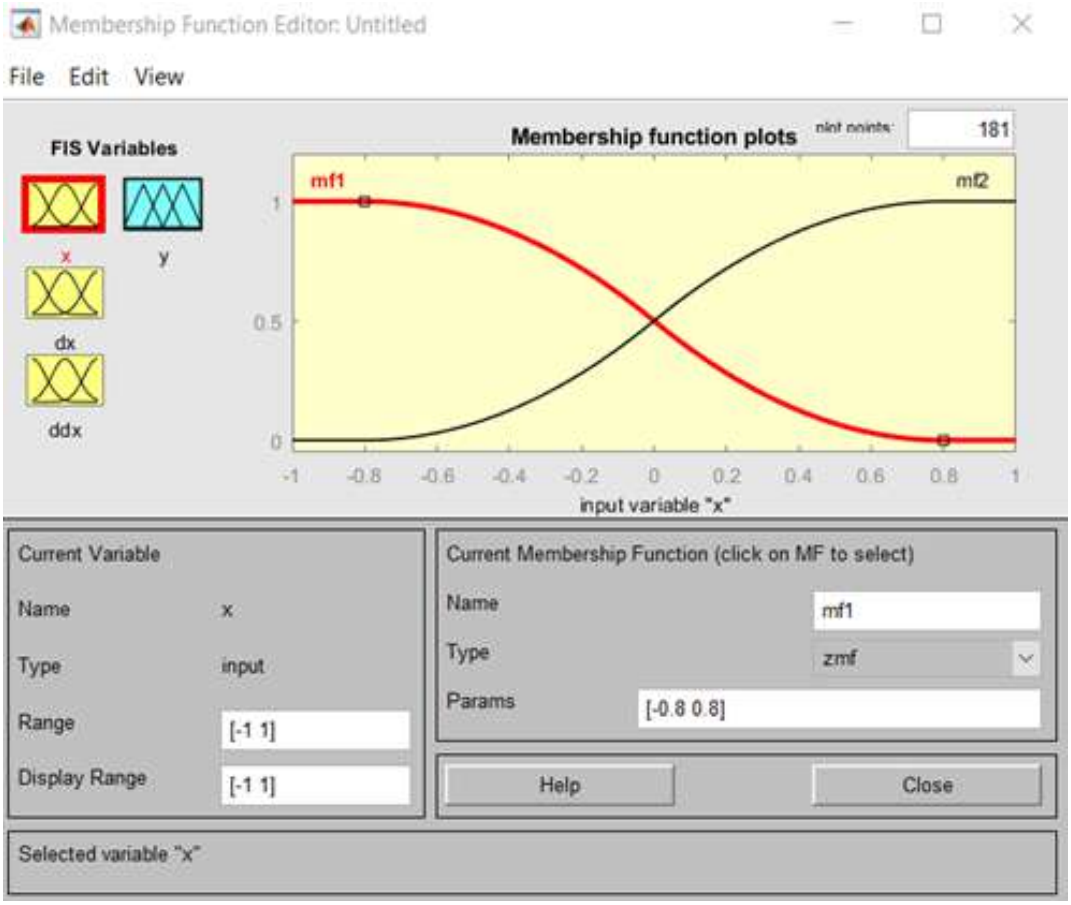

Рисунок 5.6 – Завдання вхідної змінної x

Рисунок 5.7 – Задання вхідної змінної *dd*x

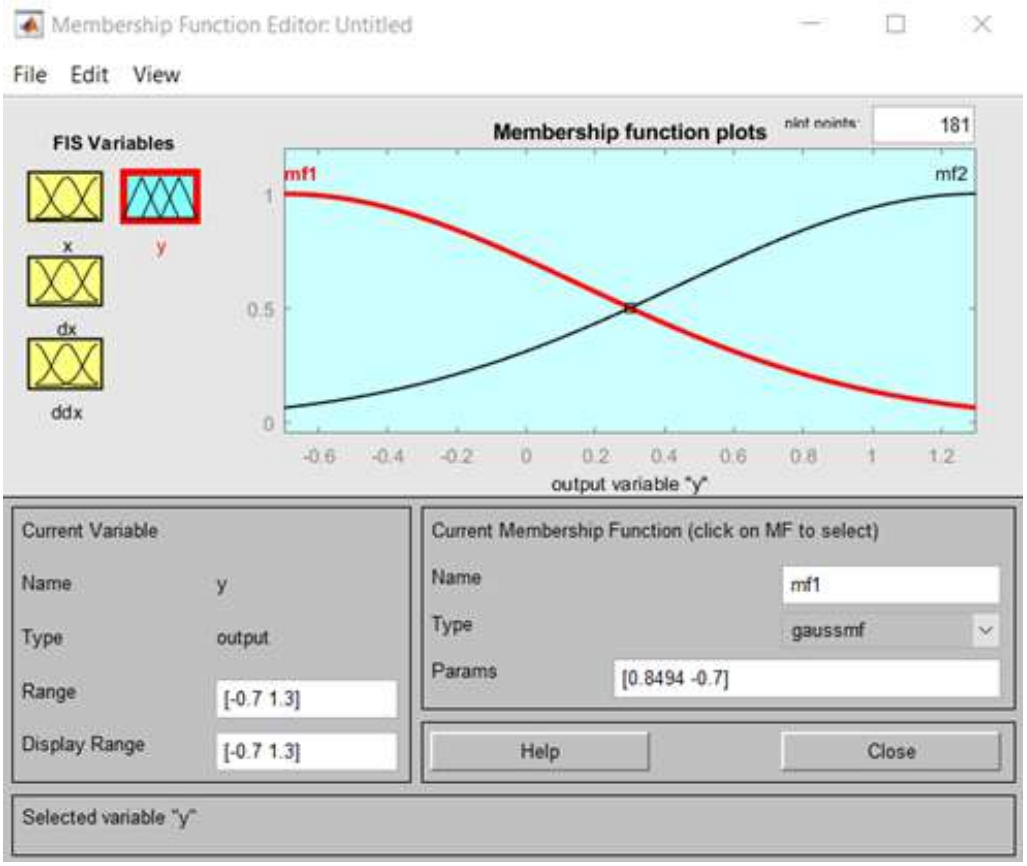

Рисунок 5.8 – Завдання вихідної змінної y

З точки зору керування для бази знань важливі тільки дві комбінації по яких складаються правила: якщо ε >0 і ε’ >0 і ε’’ >0, то U>0 якщо ε<0 і ε’ <0 і ε’ <0, то U<0.

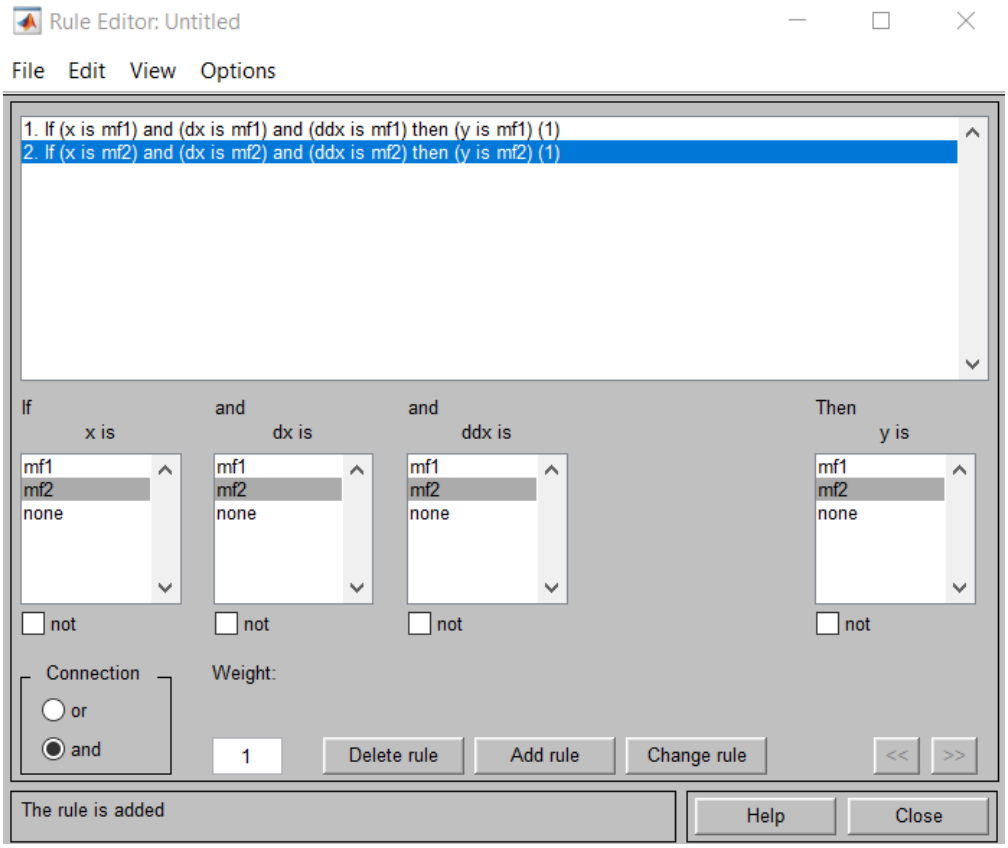

Рисунок 5.10 – Заповнення бази даних

5. Система керування неперервним об'єктом з регулятором і без нього представлена на рисунку 5.11. Перехідні процеси на вході системи без регулятора і з регулятором представлені на рисунку 5.12.

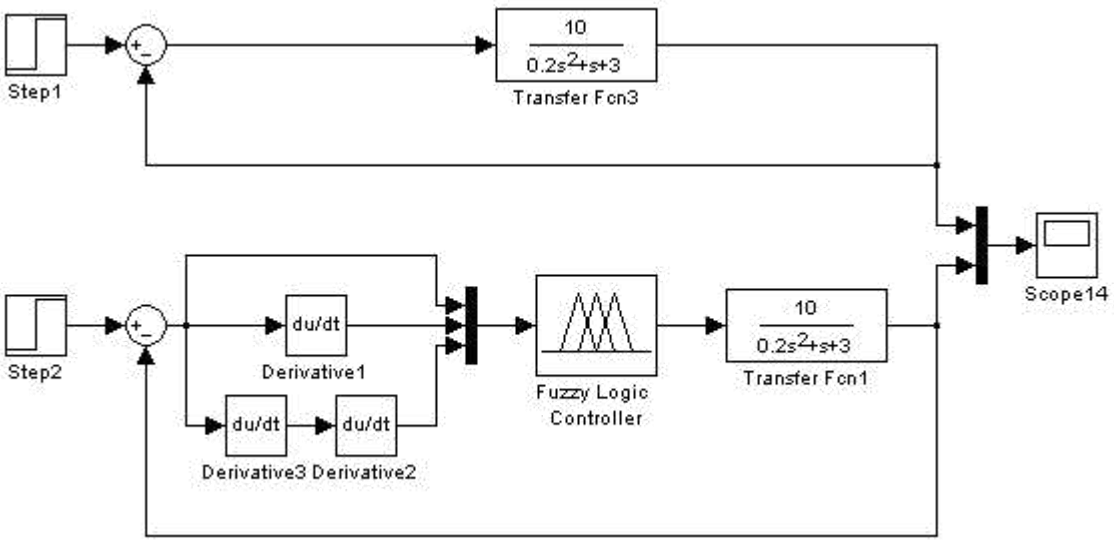

Рисунок 5.11 – Система керування неперервним об'єктом з регулятором і без нього

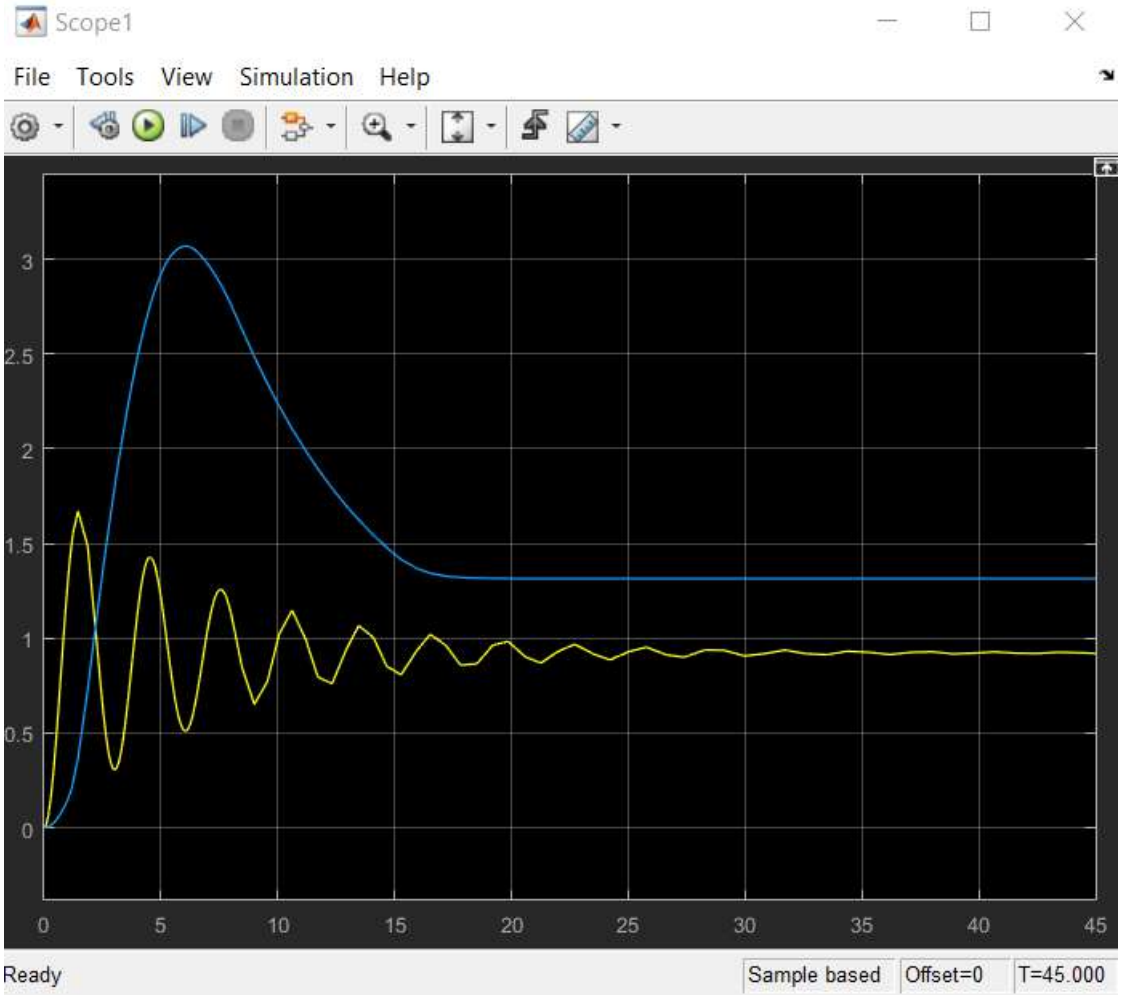

Рисунок 5.12 – Перехідні процеси на вході системи без регулятора і з регулятором

ЛАБОРАТОРНА РОБОТА №6

Дослідження та моделювання системи керування з нечіткими коефіцієнтами ПІД-регулятора

Мета роботи: Провести моделювання засобами нечіткої логіки регулятора по помилці швидкості і прискоренню її зміни. На практиці провести дослідження нечітких регуляторів та порівняти з традиційними.

Порядок виконання роботи

1. Побудувати схему представлену на рисунку 6.1.
2. Налаштувати нечіткі коефіцієнти ПІД-регулятора.

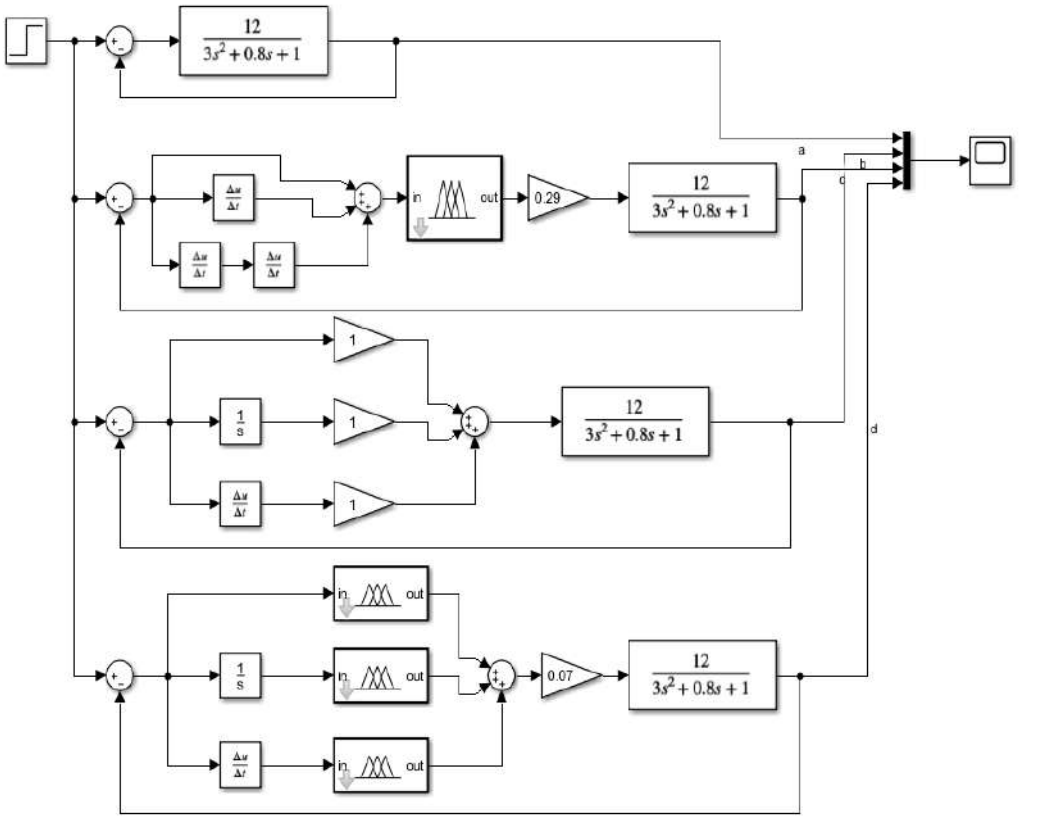

Рисунок 6.1 – Схема для дослідження нечітких регуляторів

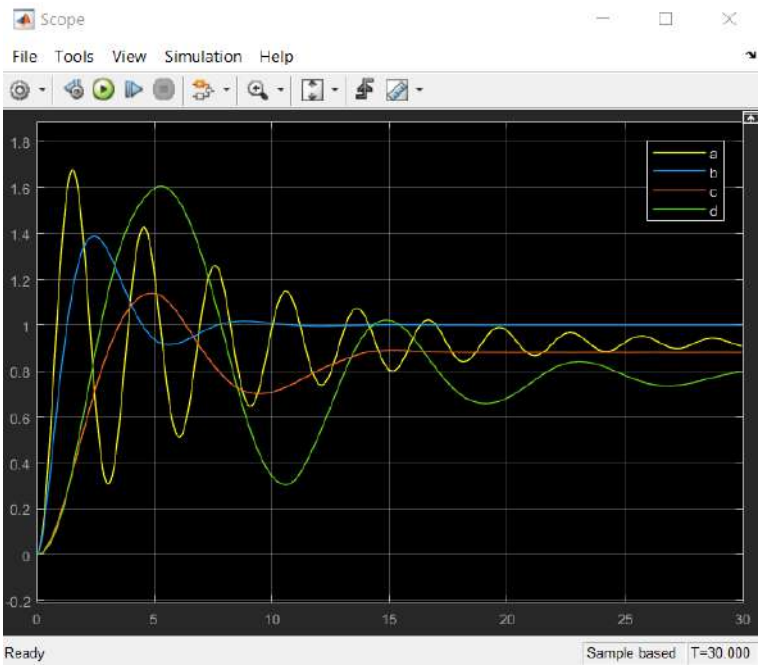

Рисунок 6.2 – Перехідні процеси схеми для дослідження нечітких регуляторів

3. Дослідити роботу регуляторів при зміні параметрів об’єкта керування. Приймемо Ку у передавальної функції = 10.

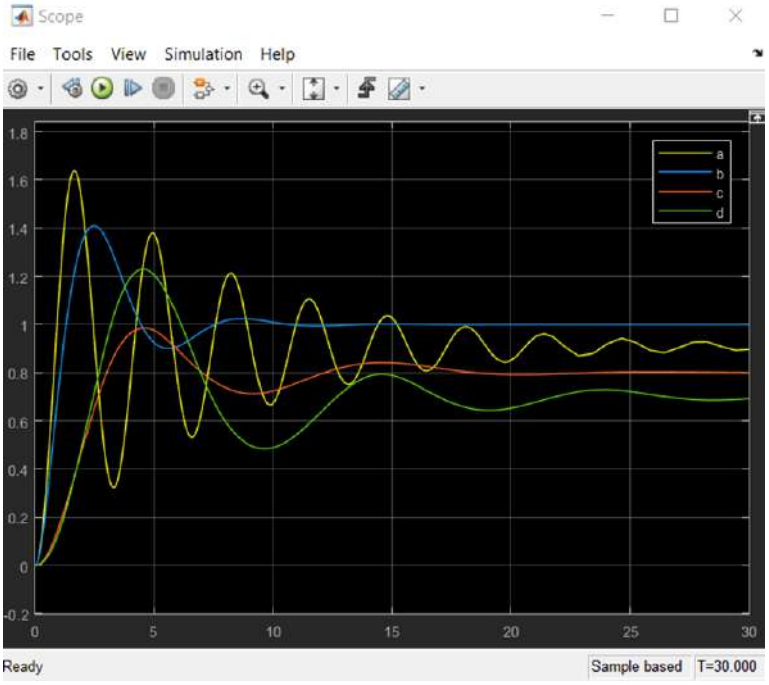

Рисунок 6.3 – Перехідні процеси схеми для дослідження нечітких регуляторів

При Ку=5.

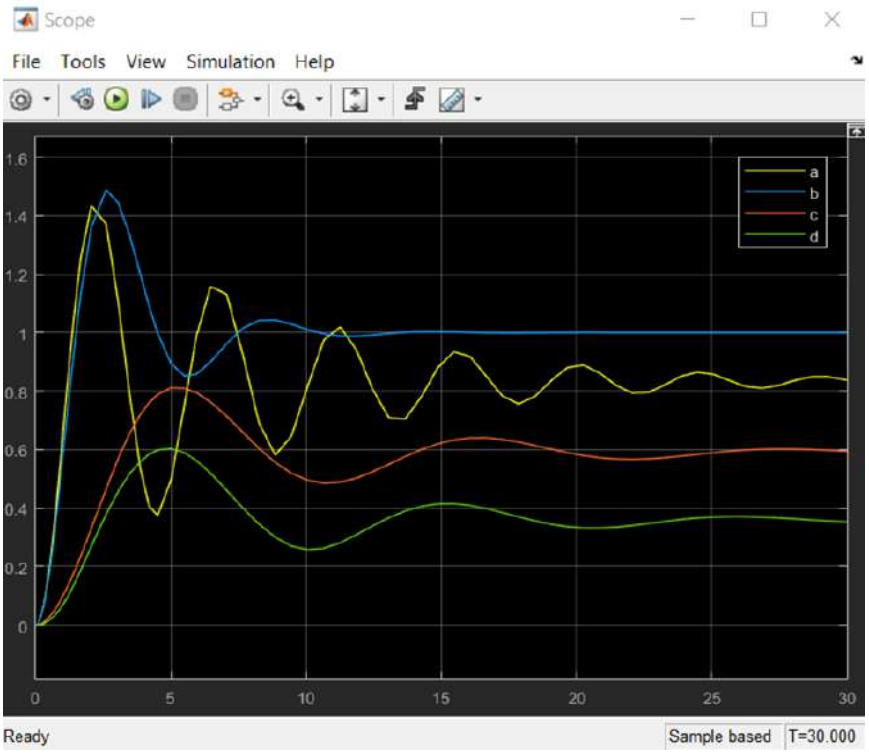

Рисунок 6.4 – Перехідні процеси схеми для дослідження нечітких регуляторів

Приведемо для прикладу конфігурації КР в ПІД-регуляторі.

Рисунок 6.5 – Конфігурації КР в ПІД-регуляторі

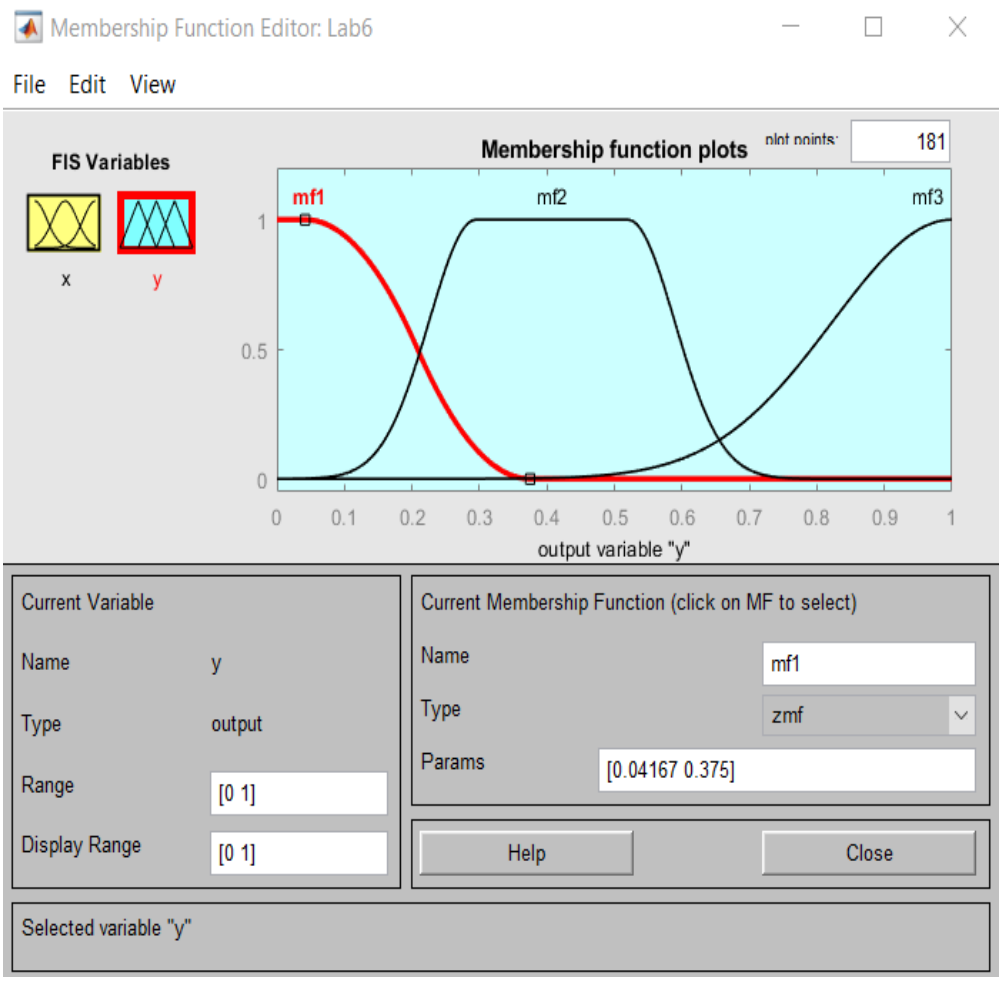

Рисунок 6.14 – Конфігурації КР в ПІД-регуляторі

ЛАБОРАТОРНА РОБОТА №7

Моделювання об'єкта керування з двома входами і одним виходом на основі нейронних мереж

Мета роботи: За допомогою нейронної мережі змоделювати функцію двох змінних. Дослідити структуру та принцип роботи нейронної мережі.

Короткі теоретичні відомості

Штучні нейронні мережі (ШНМ) – математичні моделі, а також їхня програмна та апаратна реалізація, побудовані за принципом функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Системи, архітектура і принцип дії базується на аналогії з мозком живих істот. Ключовим елементом цих систем виступає штучний нейрон як імітаційна модель нервової клітини мозку – біологічного нейрона (рисунок 7.1). Даний термін виник при вивченні процесів, які відбуваються в мозку, та при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі Маккалока і Піттса. Після розробки алгоритмів навчання, отримані моделі стали використовуватися в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах керування та інші.

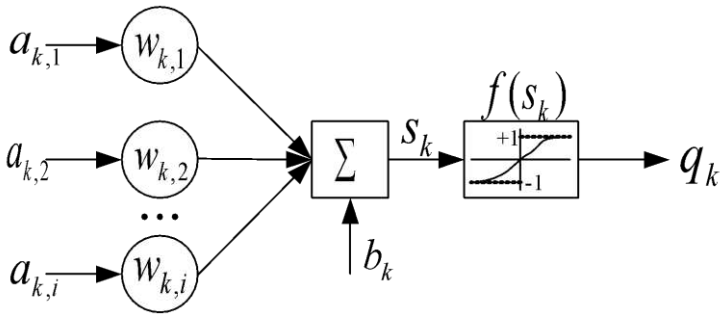

Рисунок 7.1 – Схема штучного нейрону

де q_k – вихідний сигнал *k*-го нейрона;

f – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали *k*–го нейрона;

w_{ki} – синаптична вага *k*–го нейрона;

b_k – зміщення *k*–го нейрона.

ШНМ представляють собою систему з'єднаних і взаємодіючих між собою простих процесорів(штучних нейронів). Такі процесори зазвичай достатньо прості, особливо в порівнянні з процесорами, що використовуються в персональних комп'ютерах. Кожен процесор схожої мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І тим не менш, будучи з'єднаними в досить велику мережу з керованою взаємодією, такі локально прості процесори разом здатні виконувати достатньо складні завдання. З точки зору машинного навчання, нейронна мережа являє собою окремий випадок методів розпізнавання образів, дискримінантного аналізу, методів кластеризації тощо З математичної точки зору, навчання нейронних мереж – це багатопараметрична задача нелінійної оптимізації. З точки зору кібернетики, нейронна мережа використовується в задачах адаптивного керування і як алгоритми для робототехніки. З точки зору розвитку обчислювальної техніки та програмування, нейронна мережа – спосіб вирішення проблеми ефективного паралелізму . А з точки зору штучного інтелекту, ШНМ є основою філософської течії конективізму і основним напрямком в структурному підході з вивчення можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів. Нейронні мережі не програмуються в звичайному розумінні цього слова, вони навчаються. Можливість навчання – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що у разі успішного навчання мережа зможе повернути вірний результат на

підставі даних, які були відсутні в навчальній вибірці, а також неповних та / або «зашумленних», частково перекручених даних.

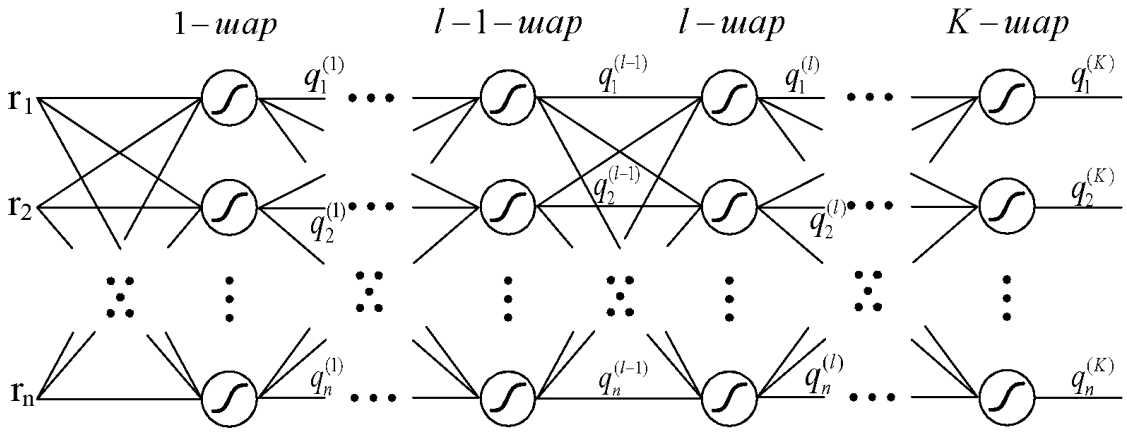

Рисунок 7.2 – Структурна схема багатошарової нейронної мережі прямого розповсюдження

Біологічна нейронна мережа складається з групи або декількох груп хімічно або функціонально пов'язаних нейронів, як показано на рисунку 7.2. Один нейрон може бути пов'язаний з багатьма іншими нейронами, а загальна кількість нейронів та зв'язків між ними може бути дуже великою. Зв'язки, які називаються синапсами, як правило формуються від аксонів до дендритів, хоча дендро-дендритичні мікросхем та інші зв'язки є можливими. Крім електричної передачі сигналів, також є інші форми передачі, які виникають з нейротрансмітерної(хімічний передавач імпульсів між нервовими клітинами) дифузії, і мають вплив на електричну передачу сигналів. Таким чином, нейронні мережі є надзвичайно складними.

Штучний інтелект і когнітивне моделювання намагаються імітувати деякі властивості біологічних нейронних мереж. Хоча аналогічні в своїх методах, перша має на меті вирішення конкретних завдань, а друга спрямована на створення математичних моделей біологічних нейронних систем.

У сфері штучного інтелекту, штучні нейронні мережі були успішно застосовані для розпізнавання мови, аналізу зображень та адаптивного

керування, для того, щоб побудувати так званих програмних агентів (в комп'ютерних і відео ігор) або автономні роботи. На даний час, більшість розроблених штучних нейронних мереж для штучного інтелекту основі на статистичних оцінках, класифікації оптимізації та теорії керування.

Сфера когнітивного моделювання включає в себе фізичне або математичне моделювання поведінки нейронних систем; від індивідуального нейронного рівня, через нейронний кластерний рівень до завершеного організму (наприклад, моделювання поведінки відповіді організму на подразники). Штучний інтелект, когнітивне моделювання і нейронні мережі є парадигмами обробки інформації натхненні системами біологічних нейронів обробки інформації.

Порядок виконання роботи

Варіант завдання на дану лабораторну роботу вибираються з додатка Б за номером групи та номером студента в списку групи.

Для побудови та навчання нейронної мережі необхідно мати дані, які представляють собою набори значень виходів з відповідними їм значеннями входів. Тому шляхом моделювання функції двох змінних ці дані будуть передані в робочу область (WorkSpace) Matlab.

Для цього використовують блоки To WorkSpace, де вихідним параметром вибираємо Array (масив). Схема отримання вихідних даних представлена на рисунку 7.3.

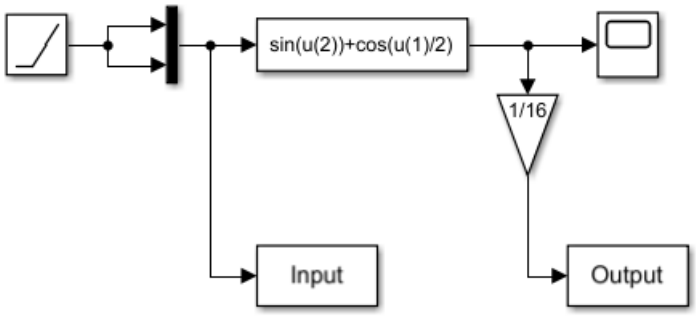

Рисунок 7.3 – Схема отримання вихідних даних

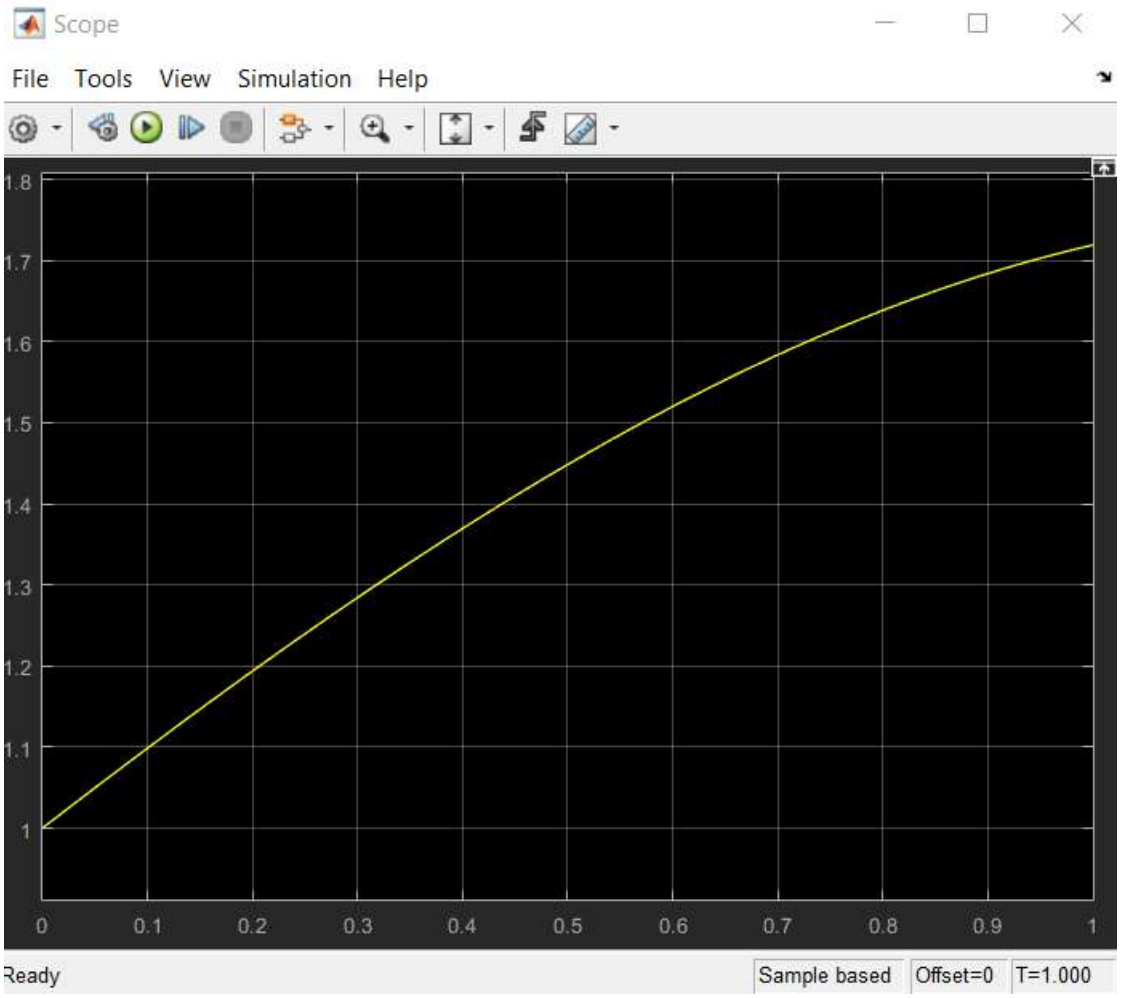

Рисунок 7.4 – Покази осцилографа

Величину коефіцієнта Gain слід вибирати так, щоб вихід функції знаходився в діапазоні (-1;1). Це пов'язано з особливостями роботи нейронної мережі.

Отже, після запуску моделювання в робочій області з'являється дві змінні: input і output. Для використання їх в якості даних для навчання нейронних мереж з ними необхідно виконати операцію транспонування:

```
>> Input = input ';
>> Output = output ';
```

Після цього приступаємо до проектування нейронних мереж. Для цього використовуємо графічну оболонку тулбокса Neural Networks, що запускається командою *nntool*.

Для нашої задачі (2 входи, 1 вихід) необхідно, щоб в першому шарі мережі було 2 нейрона і в останньому 1 (число нейронів в крайніх шарах дорівнює кількості входів / виходів). Кількість середніх шарів і нейронів в них вибирається розробником мережі.

На початку необхідно імпортувати з робочої області в тулбокс Neural Network вихідні дані (input і output). Кнопка Import, вибрати змінну input і справа as input. Натиснути done. Аналогічно імпортувати output, але вибрати обов'язково as target. Після цього натискаємо кнопку New network, та переходимо до створення мережі.

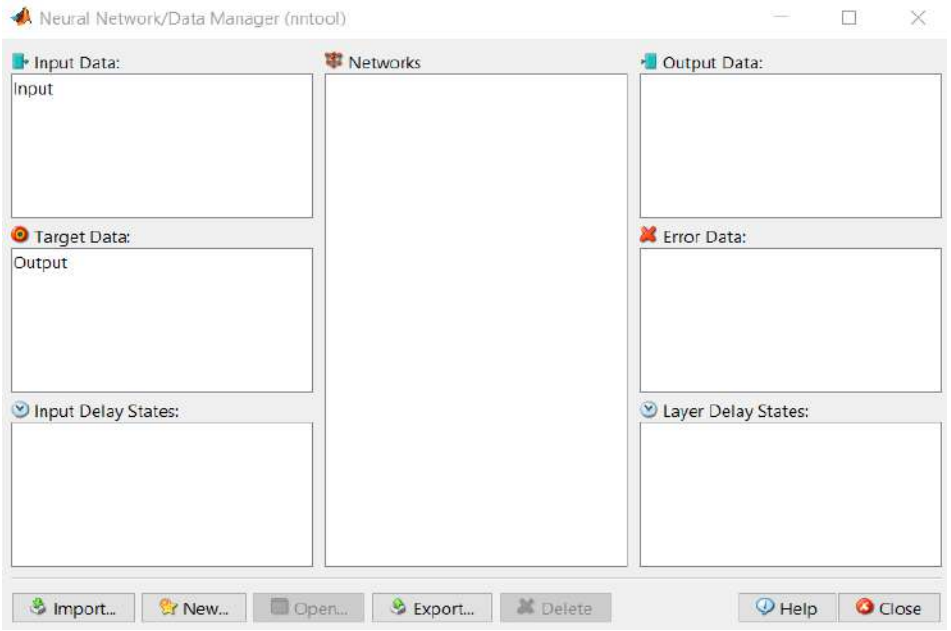

Рисунок 7.5 – Зовнішній вигляд тулбокса Neural Network

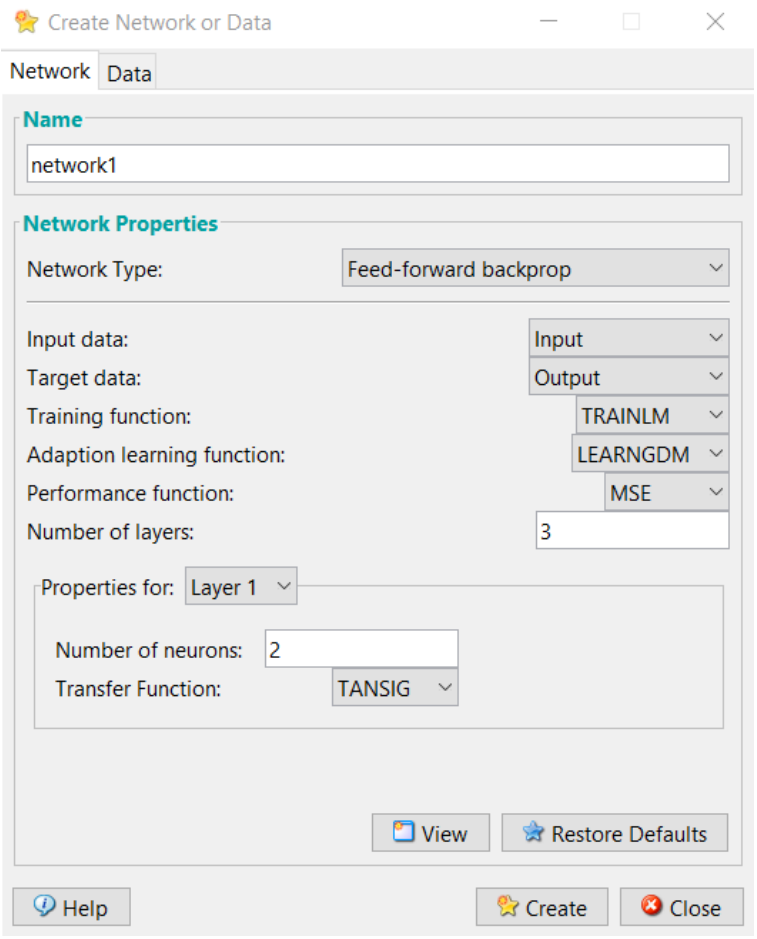

Рисунок 7.5 – Створення нової нейронної мережі

Послідовно заповнюючи поля діалогового вікна, задати параметри нової мережі, далі натиснути Create.

Далі приступаємо до навчання нейронної мережі – кнопка Train (рисунок 7.6)

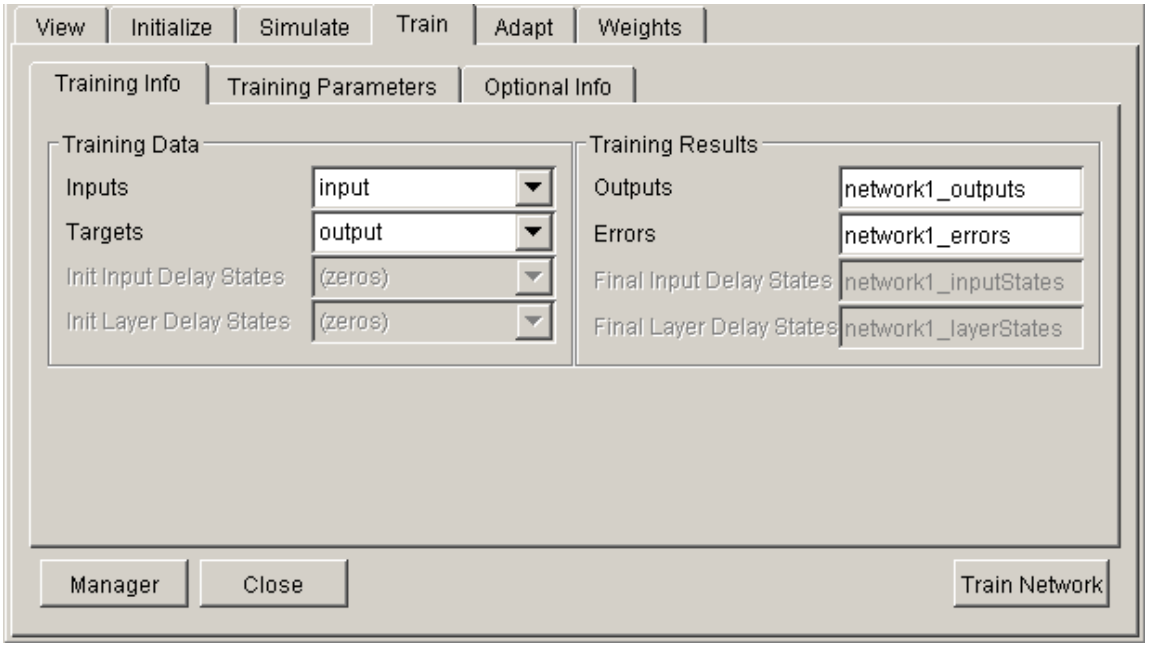

Рисунок 7.6 – Запуск навчання нейронної мережі

Вибираємо як показано на рисунку 7.6 в області Traning Data змінні input і output. На вкладці Training Parameters вибираємо необхідну кількість епох (кількість циклів навчання мережі). Натискаємо кнопку Train і можемо спостерігати графік залежності помилки в мережі від епохи:

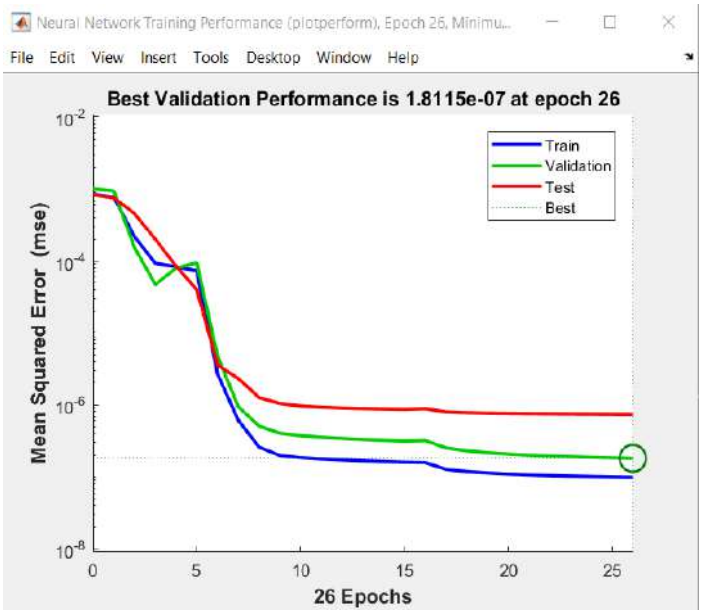

Рисунок 7.7 – Графік залежності помилки в мережі від епохи

Після закінчення процесу навчання експортуємо мережу у робочу область через головне вікно тулбокса (кнопка Export):

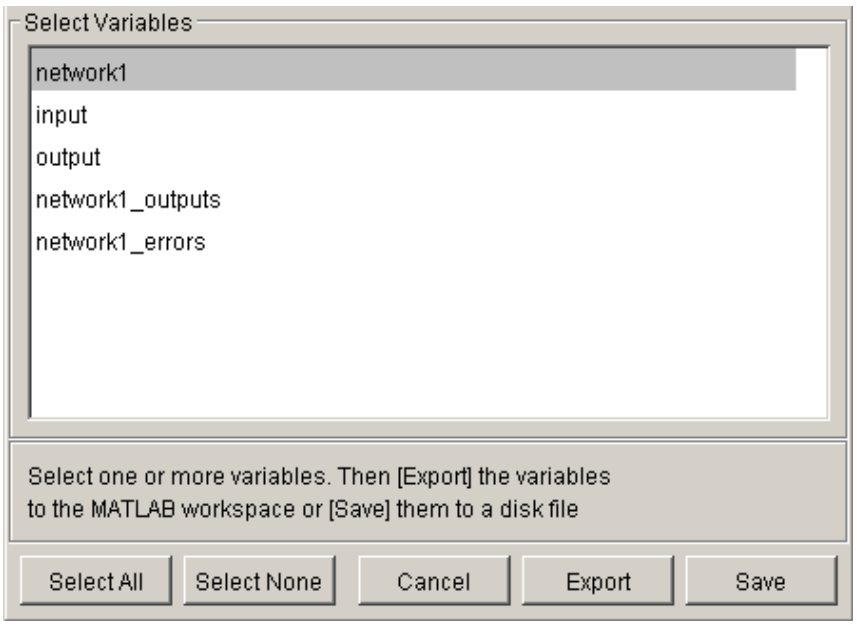

Рисунок 7.8 – Експорт мережі до робочої області

Потім в командному рядку набираємо команду *gensim* (network1, -1), яка генерує модель мережі в Simulink. Параметр -1 означає, що нейронна мережа безперервна, будь-яке значення більше нуля інтерпретується як період квантування.

Копіюємо генеровану мережу в модель, як показано на рисунку 7.9:

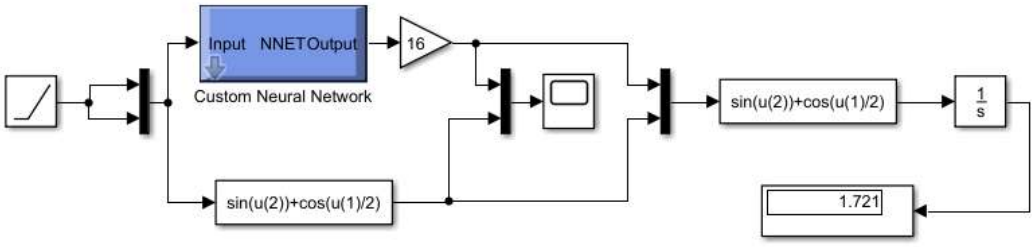

Рисунок 7.9 – Створення моделі нейронної мережі у Simulink

Вихід нейронної мережі помножимо на число, протилежне початковому коефіцієнту. Включаємо в модель ділянку підрахунку середньої відносної помилки моделювання. Графіки, побудовані блоком Fcn і нейронною мережею представлені на рисунку 13.9 (похибка складає ε = 0,0002219%):

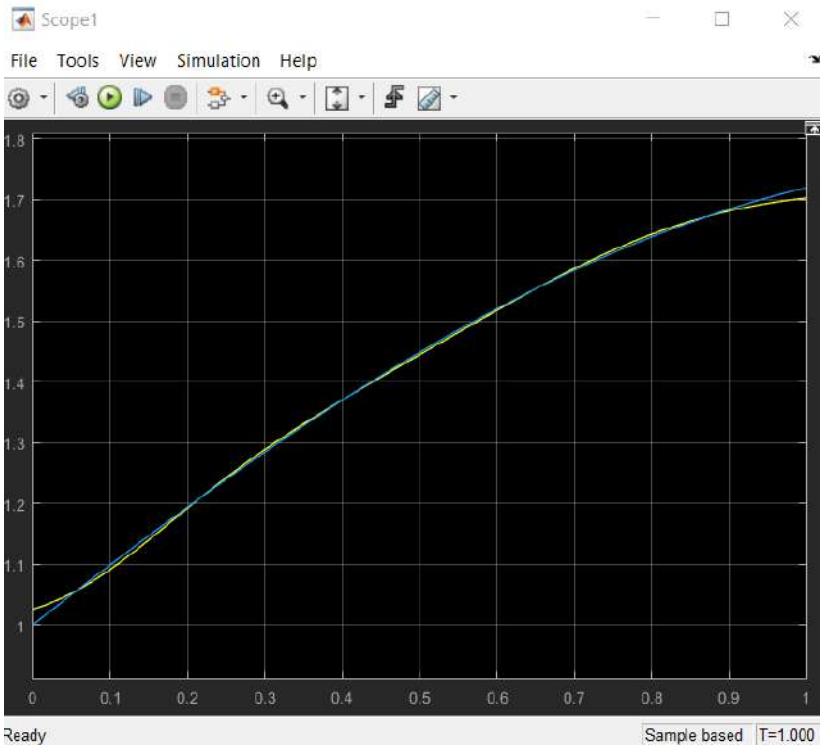

Рисунок 7.10 – Графіки, побудовані блоком Fcn і нейронною мережею

Завдання

Дослідити вплив кількості внутрішніх шарів та кількості нейронів на середню відносну помилку моделювання для різних типів мереж (feed forward backprop, cascade - forward backprop, elman backprop):

1. Тип мережі: feed forward backprop:
    a) 1 внутрішній шар з 10 нейронами;
    b) 1 внутрішній шар з 20 нейронами;
2. Тип мережі: cascade - forward backprop:
    a) 1 внутрішній шар з 20 нейронами;
    b) 2 внутрішніх шари по 10 нейронів у кожному;
3. Тип мережі: elman backprop:
    a) 1 внутрішній шар з 15 нейронами;
    b) 3 внутрішніх шари по 5 нейронів у кожному;
4. Зробити висновки на основі отриманих даних.

ЛАБОРАТОРНА РОБОТА №8

Моделювання системи керування з нейромережевими двопозиційним і трипозиційним регулятором

Мета роботи: Провести моделювання за допомогою нейронних мереж двопозиційних та трипозиційних регуляторів. На практиці провести дослідження нейромережевих регуляторів та порівняти з традиційними.

Порядок виконання роботи

1. Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи. Моделюємо систему в simulink з автоколивальним перехідним процесом, використовуючи 2-х позиційне реле:

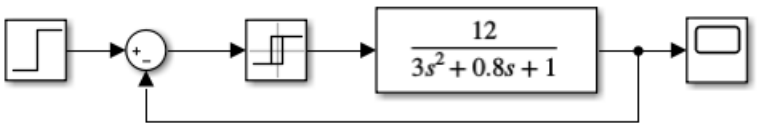

Рисунок 8.1 – Модель системи з автоколивальним перехідним процесом

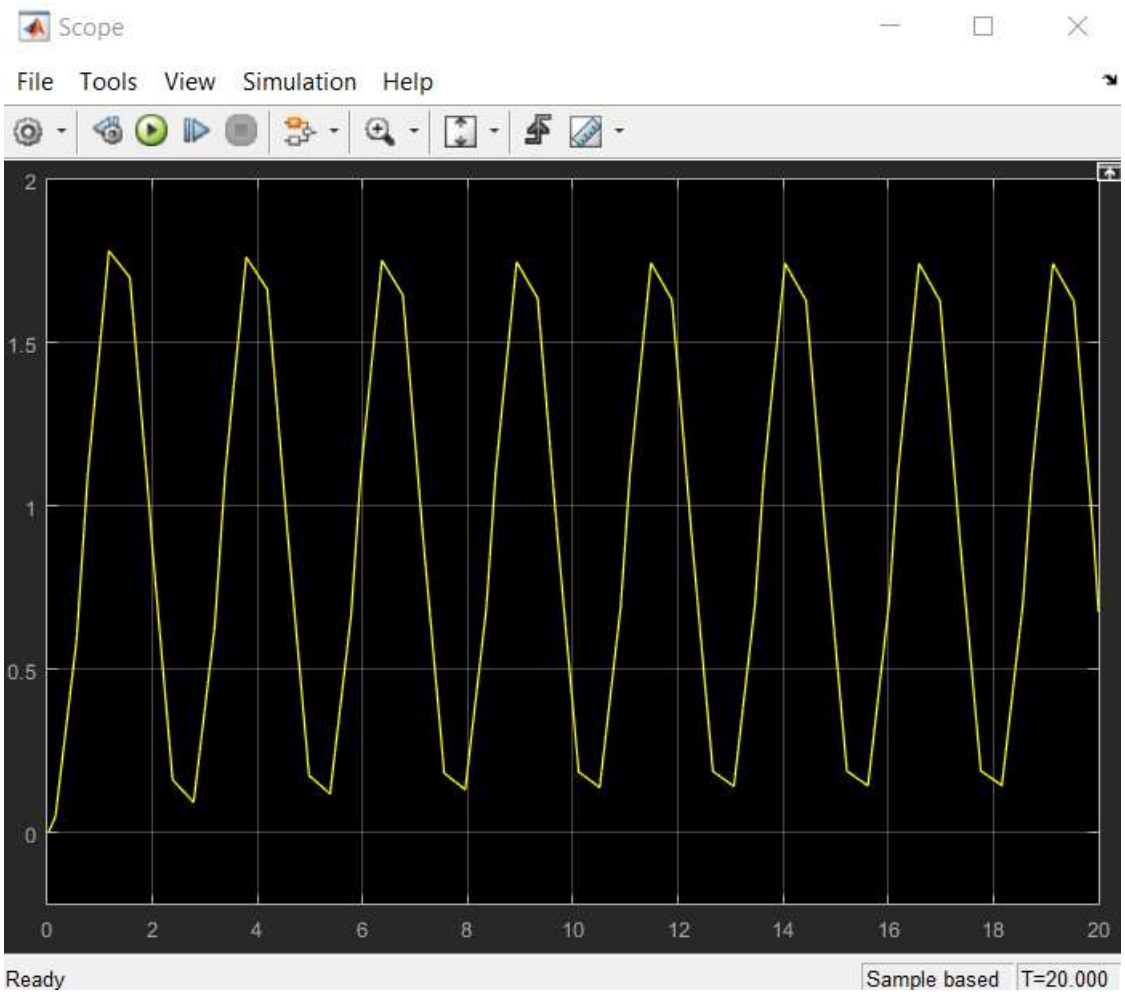

Рисунок 8.2 – Перехідний процес системи

2. Аналогічно моделюванню реле за допомогою нечіткої логіки (лабораторна робота № 2) необхідно мати інформацію не тільки про вхід релейного елемента, але і про похідної за часом від вхідної величини, тоді схема отримання даних для навчання мережі буде мати вигляд:

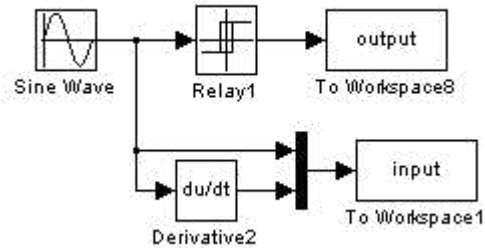

Рисунок 8.3 – Схема отримання даних для навчання нейромережі

3. Створюємо нову нейронну мережу з кількістю шарів 3, кількістю вхідних нейронів - 2, кількість вихідних нейронів - 1, кількістю нейронів у другому шарі - 12. наприклад:

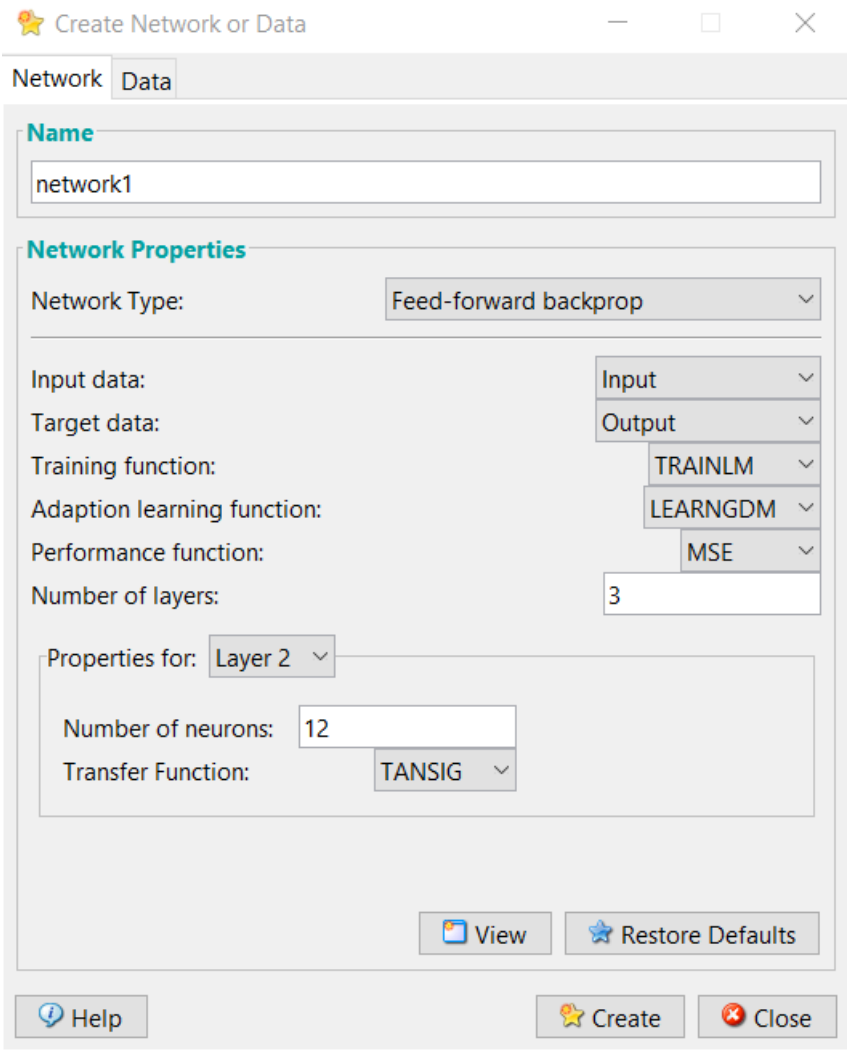

Рисунок 8.4 – Вікно створення нейронної мережі

4. Навчаємо нейронну мережу:

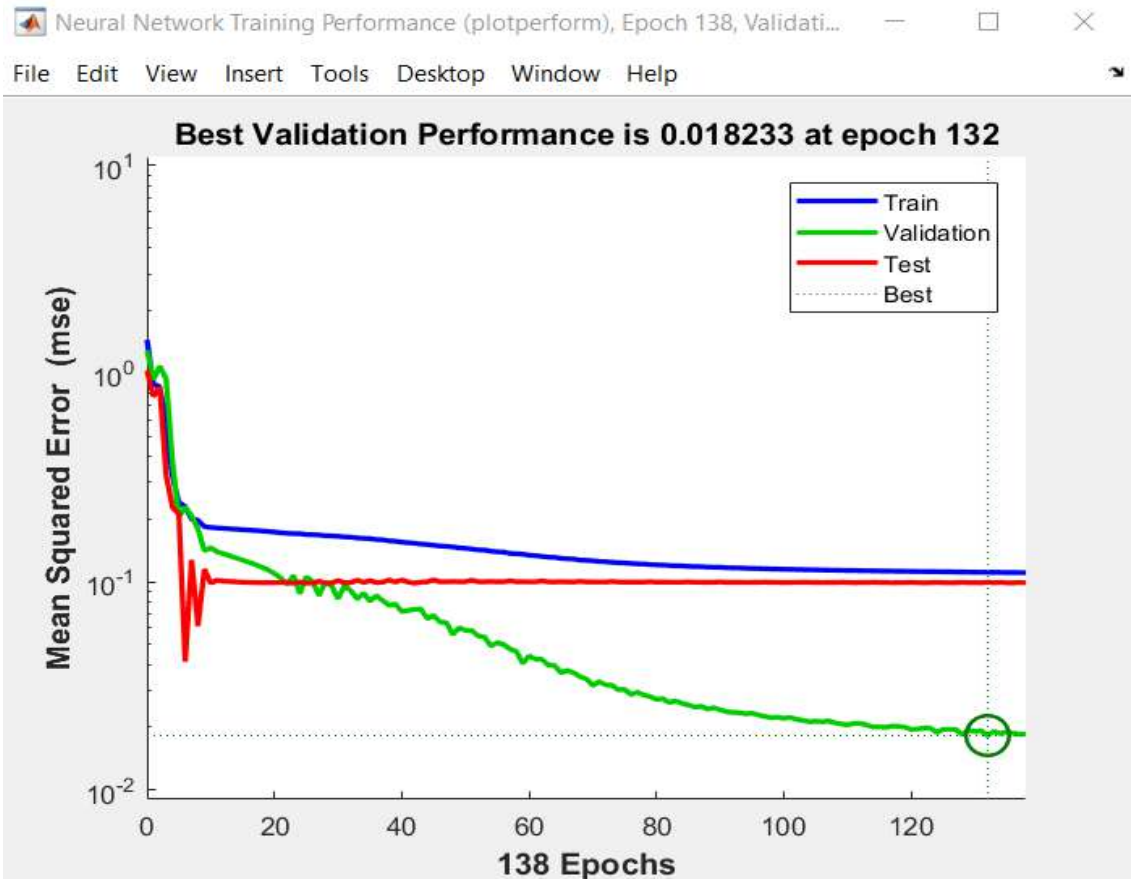

Рисунок 8.5 – Процес навчання нейронної мережі

Далі всі операції виконуються аналогічно лабораторної роботи № 2.

5. Складаємо модель, зображену на рисунку:

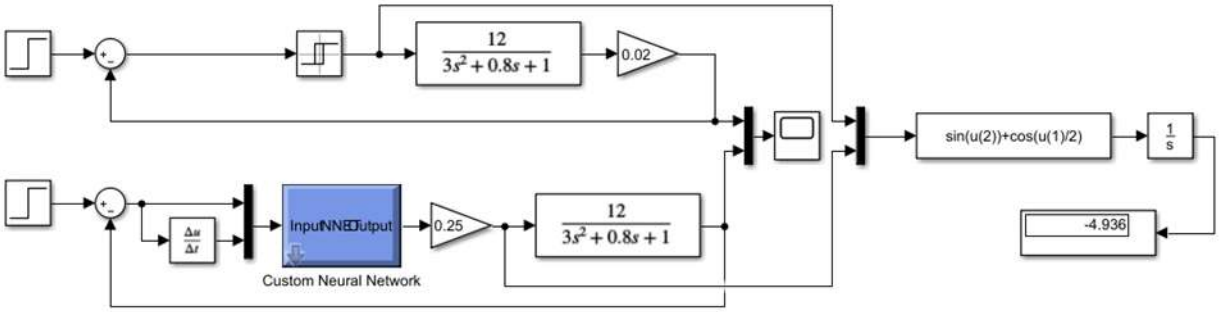

Рисунок 8.6 – Модель системи керування з класичним регулятором і нейромережевим

6. Порівнюємо графіки:

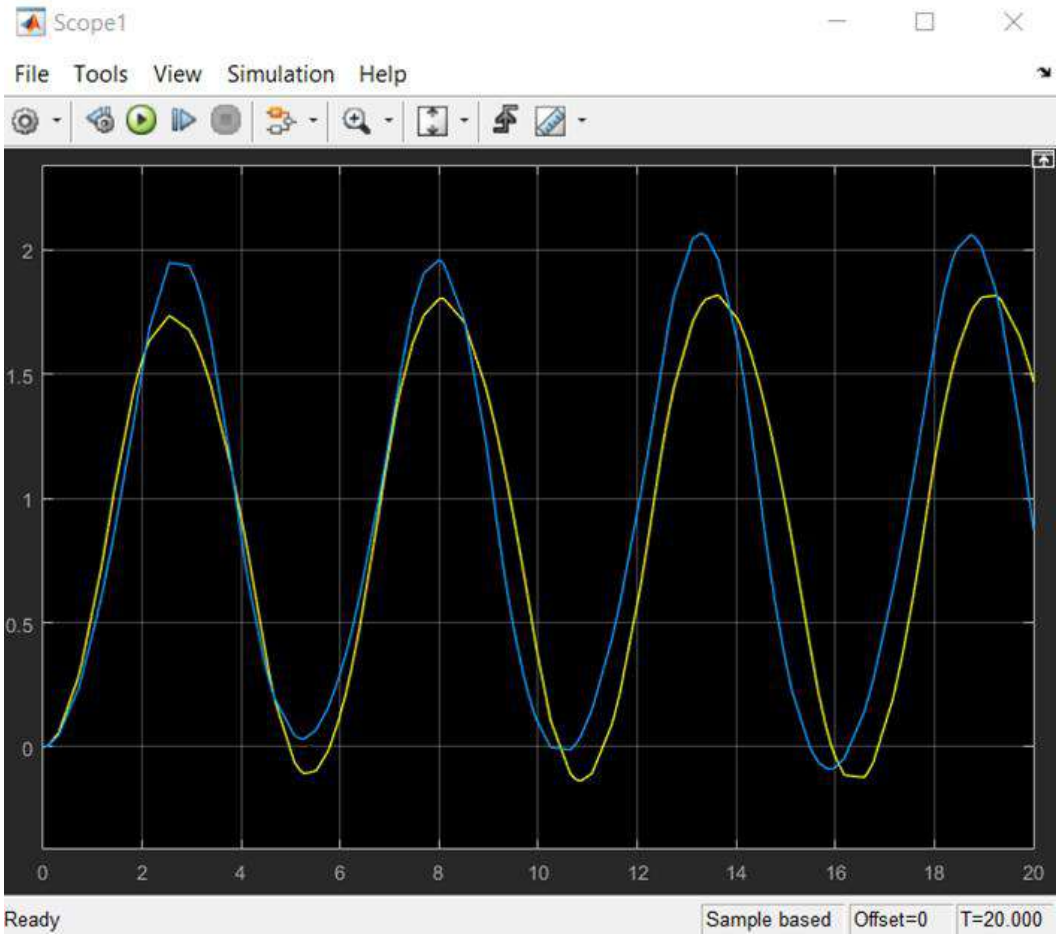

Рисунок 8.7 – Перехідний процес в системах керування з класичним регулятором і нейромережевим

Нейромережева система з автоколивальних перехідним процесом і 3-х позиційним регулятором.

Отримання даних відбувається аналогічно 2-х позиційному реле:

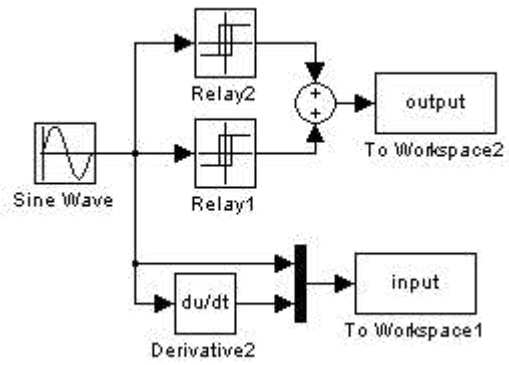

Рисунок 8.8 – Схема отримання даних з трипозиційного регулятора для навчання нейромережі

Перехідний процес системи:

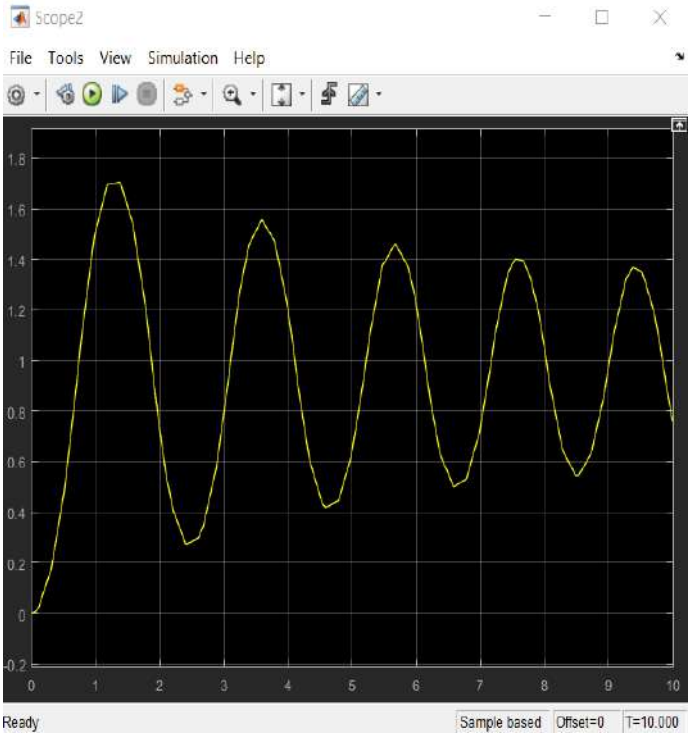

Рисунок 8.9 – Перехідний процес системи з трипозиційним регулятором

Створення і навчання мережі виконується за тим же алгоритмом.

Помилка при навчанні:

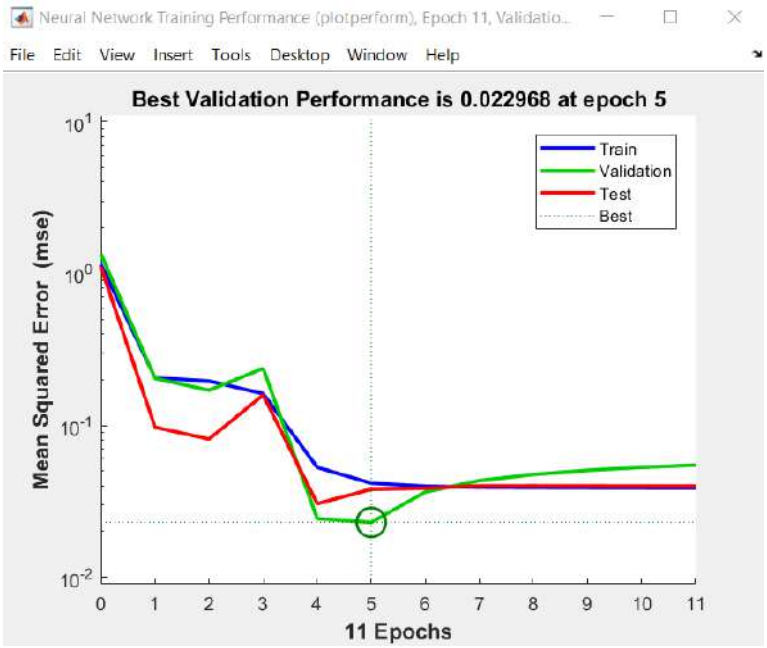

Рисунок 8.10 – Процес навчання нейронної мережі

Система з нейромережевих регулятором:

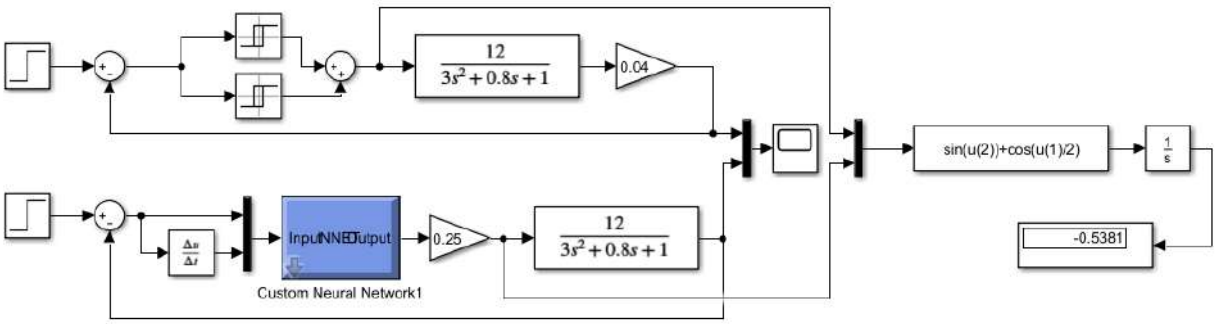

Рисунок 8.11 – Модель системи керування з класчним регулятором і нейромережевим

Перехідні процеси:

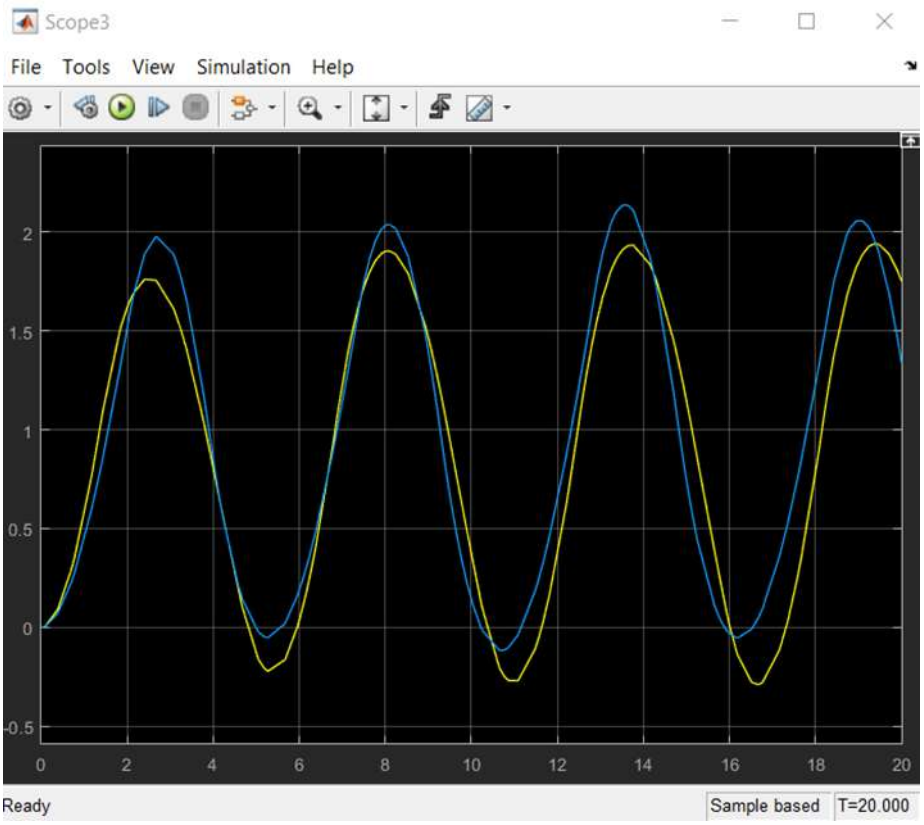

Рисунок 8.12 – Перехідний процес в системах керування з класчним регулятором і нейромережевим

ЛАБОРАТОРНА РОБОТА №9

Моделювання системи керування з нейромережевим ПІД-регулятором

Мета роботи: Провести моделювання за допомогою нейронних мереж ПІД-регулятора. На практиці провести дослідження нейромережевих регуляторів та порівняти з традиційними.

Порядок виконання роботи

1. Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи. Далі побудувати в MatLab модель системи керування з ПІД-регулятором з коефіцієнтами, знайденими аналогічно лабораторній роботі №3.

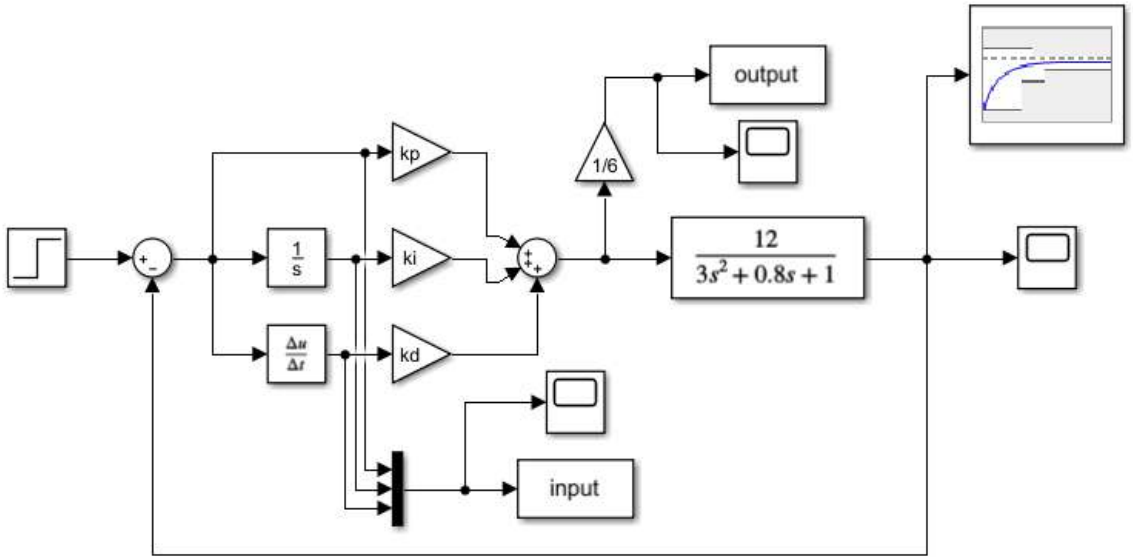

Рисунок 9.1 – Модель системи керування з ПІД-регулятором

Перехідний процес системи представлено на рисунку 9.2

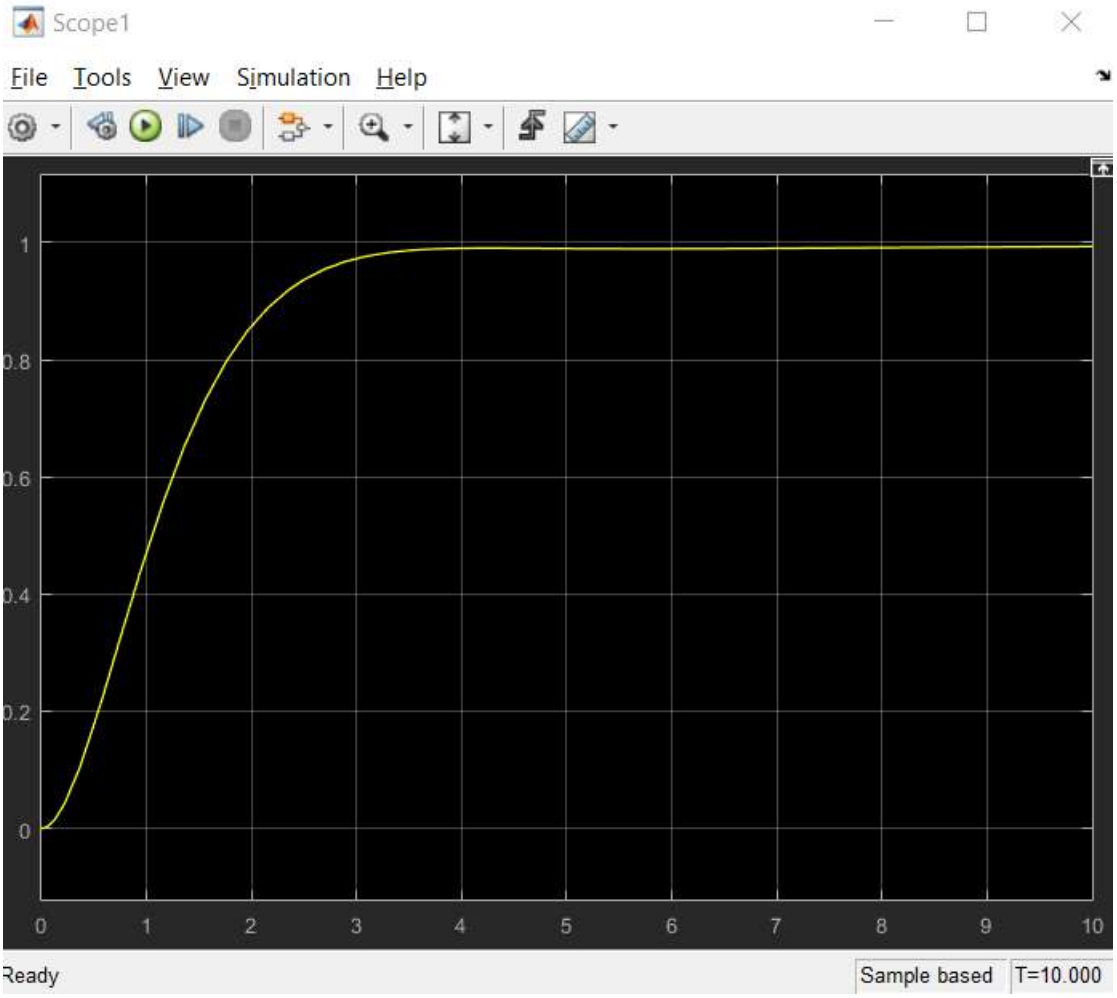

Рисунок 9.2 – Перехідний процес системи

2. За схемою представленою на рисунку 9.1 накопичуємо інформацію про вхідні і вихідні сигнали класичного ПІД-регулятора в змінні *input* і *output*, для подальшого навчання нейромережі.

3. Весь процес створення і навчання нейронної мережі аналогічний попередній лабораторній роботі. Процес навчання нейронної мережі представлений на рисунку 9.3.

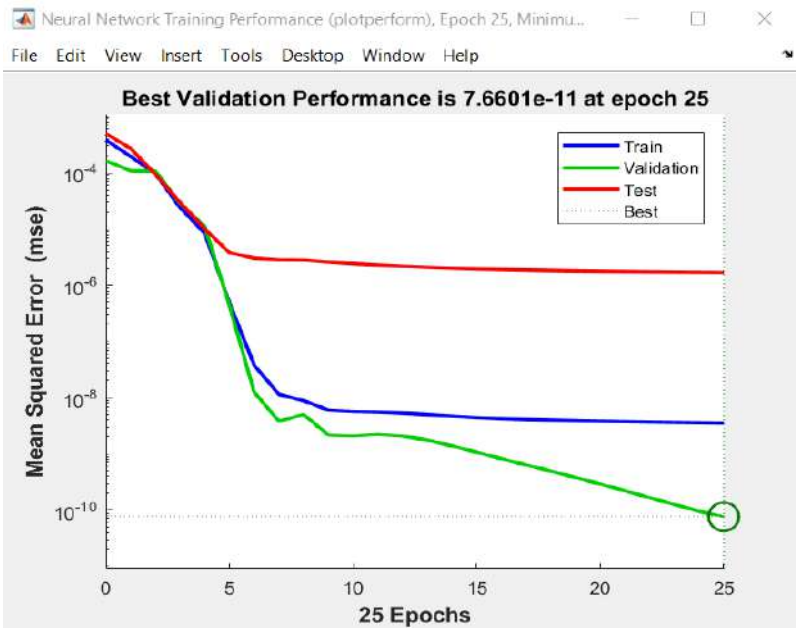

Рисунок 9.3 – Процес навчання нейронної мережі

Модель систем керування з класичним ПІД-регулятором та нейромережевим регулятором представлено на рисунку 9.4.

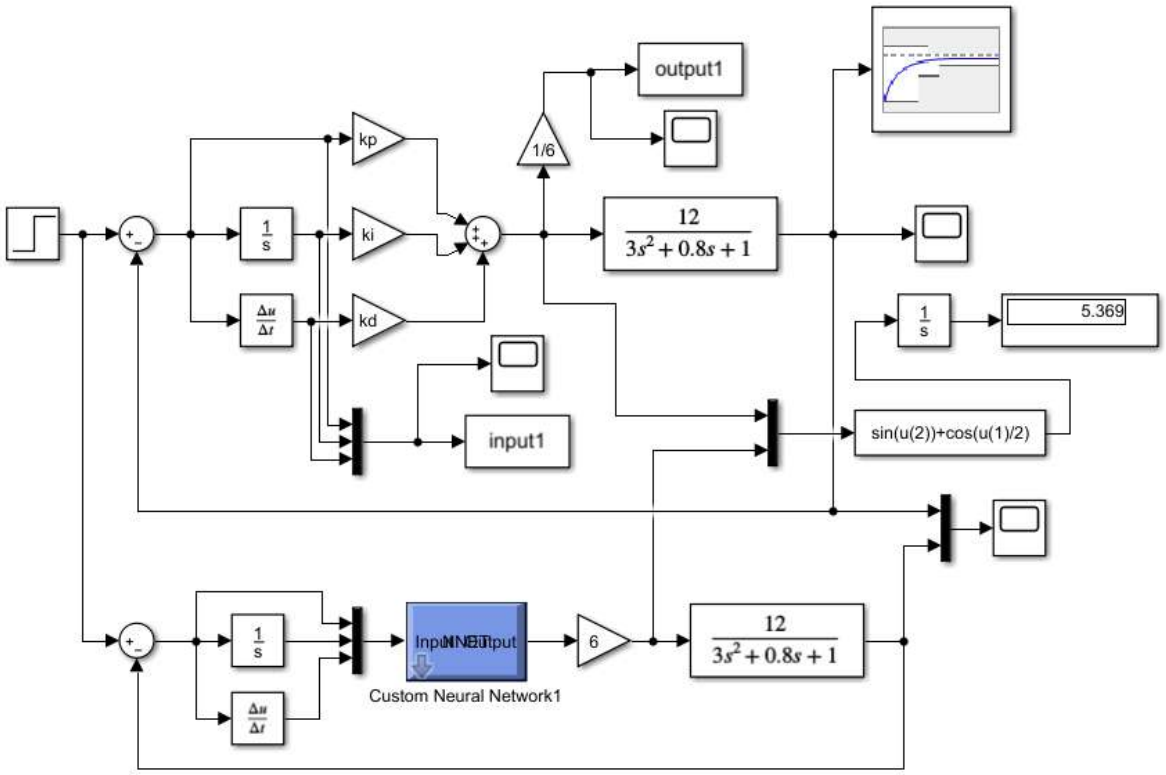

Рисунок 9.4 – Модель систем керування з класичним ПІД-регулятором та нейромережевим регулятором

Перехідні процеси двох систем представлені на наступному рисунку.

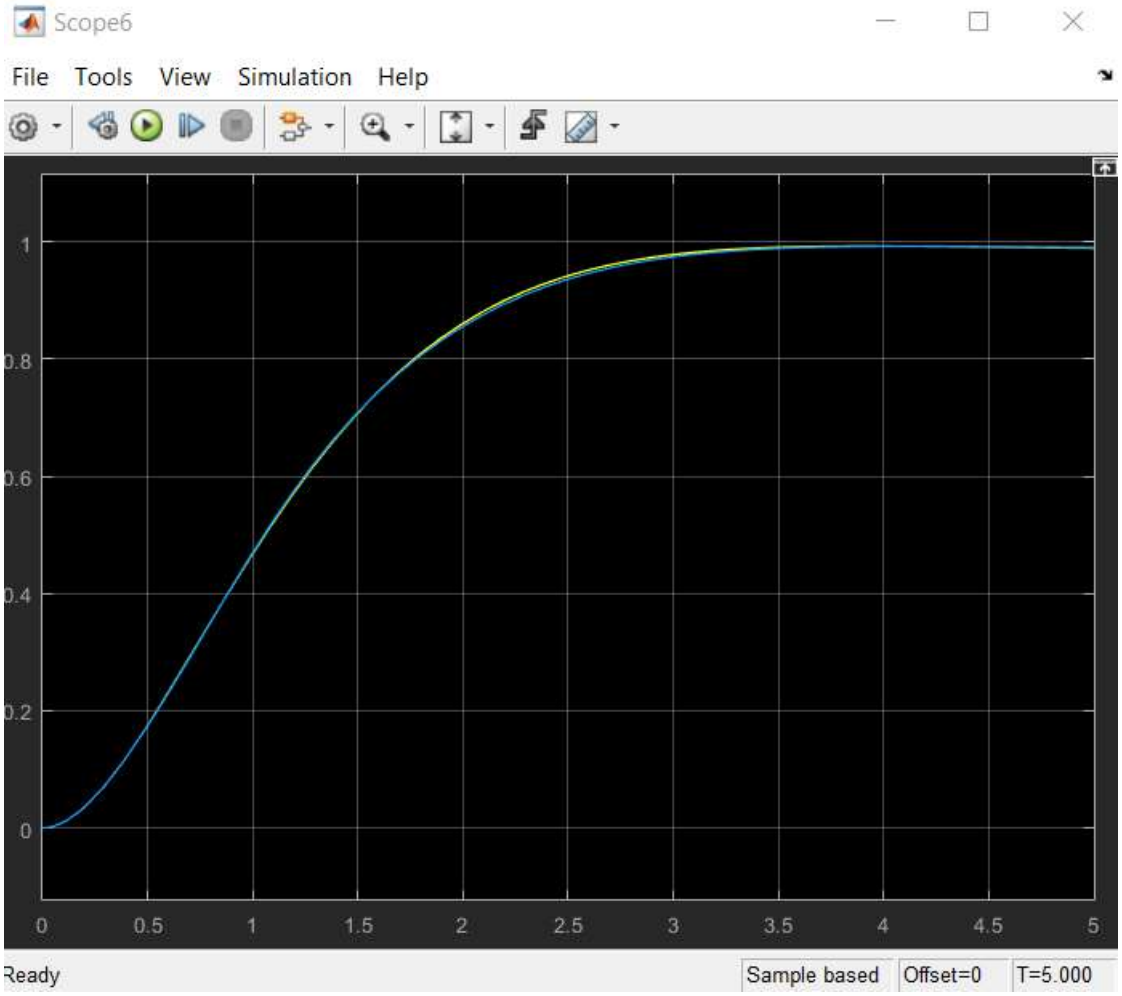

Рисунок 9.5 – Перехідні процеси систем керування з класичним ПІД-регулятором та нейромережевим регулятором

ЛАБОРАТОРНА РОБОТА №10

Моделювання системи керування з нейромережевим регулятором оптимальним по швидкодії

Мета роботи: Провести моделювання за допомогою нейронних мереж регуляторів оптимальних за швидкодією. На практиці провести дослідження нейромережевих регуляторів та порівняти з традиційними.

Порядок виконання роботи

1. Вибрати варіант завдання на дану лабораторну роботу з додатка А за номером групи та номером студента в списку групи. Побудувати модель системи керування в simulink з регулятором оптимальним за швидкодією.

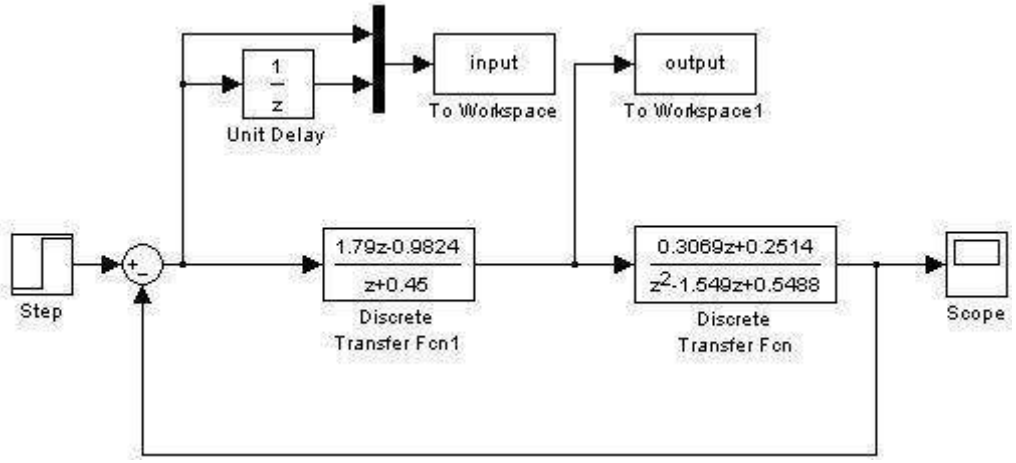

Рисунок 10.1 – Модель системи керування з регулятором оптимальним за швидкодією

Перехідний процес системи керування з регулятором оптимальним за швидкодією представлений на рисунку 10.2.

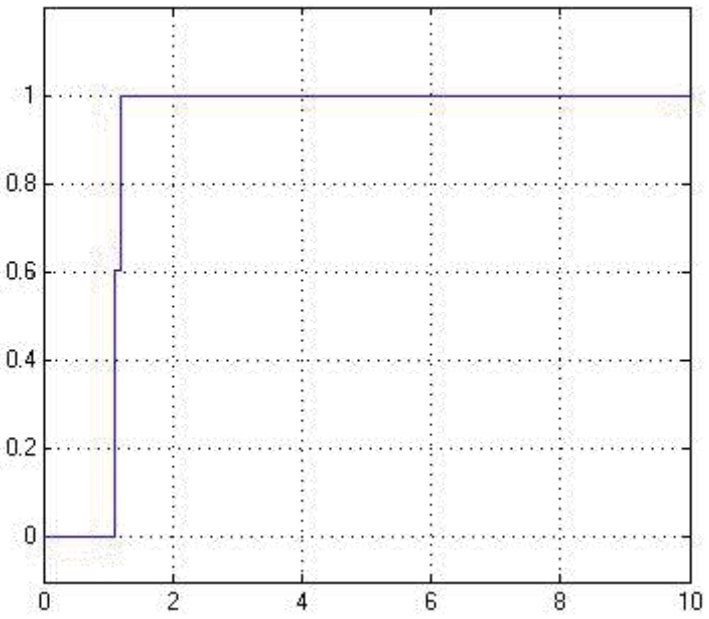

Рисунок 10.2 – Перехідний процес системи керування з регулятором оптимальним за швидкодією

2. Створюємо нову нейронну мережу з кількістю шарів 3, кількістю вхідних нейронів – 2, кількість вихідних нейронів – 1, кількістю нейронів у прихованому шарі – 15. Далі всі операції виконуються аналогічно лабораторної роботи № 1. Складаємо модель, зображену на рисунку 10.3.

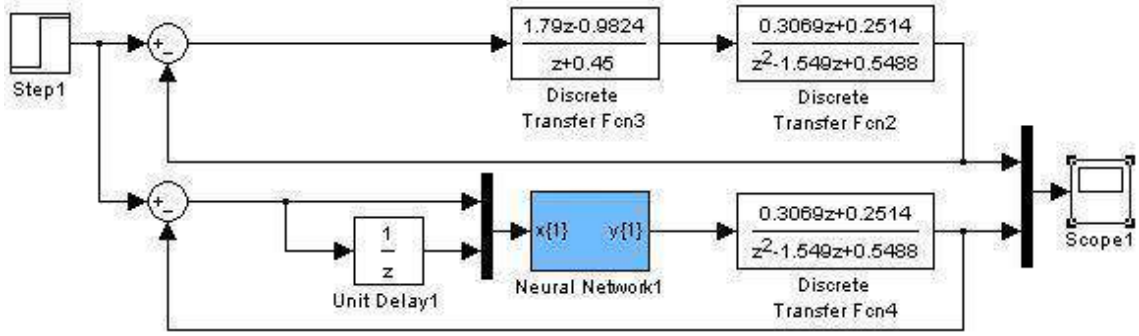

Рисунок 10.3 – Модель систем керування з класичним регулятором оптимальним за швидкодією та нейромережевим регулятором

3. Порівнюємо перехідні процеси двох систем керування.

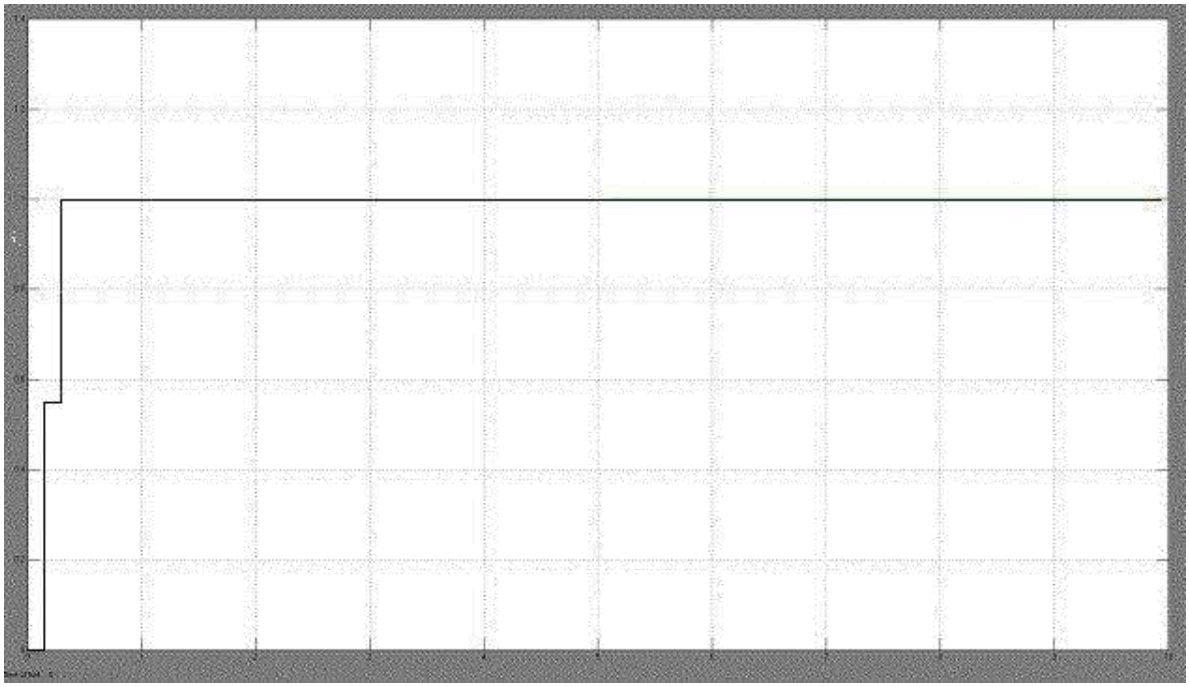

Рисунок 10.4 – Перехідні процеси двох систем керування

4. Синтезуємо нейронну мережу для наступних параметрів:

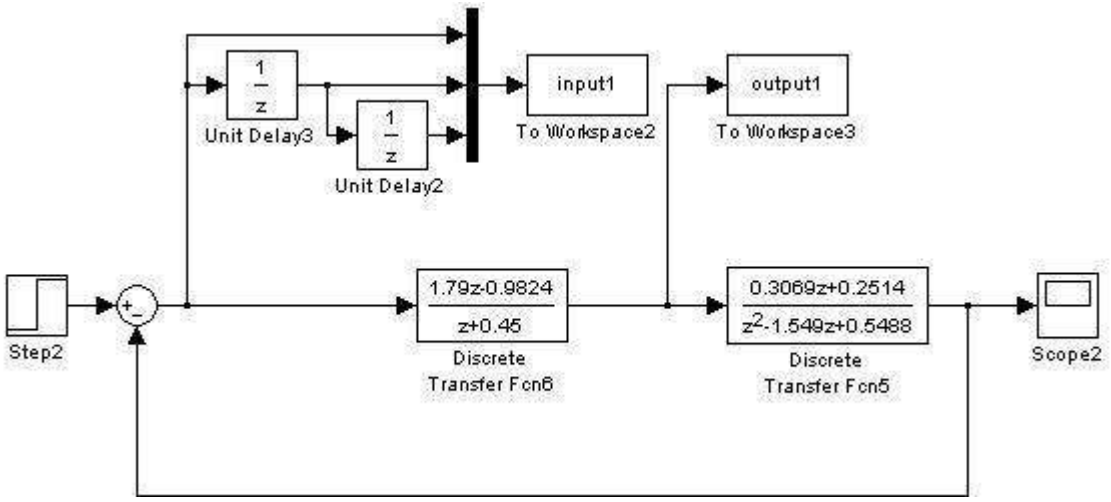

Рисунок 10.5 – Модель системи керування з регулятором оптимальним за швидкодією

5. Виконуючи аналогічні дії, представлені в пункті 2, отримуємо наступні результати:

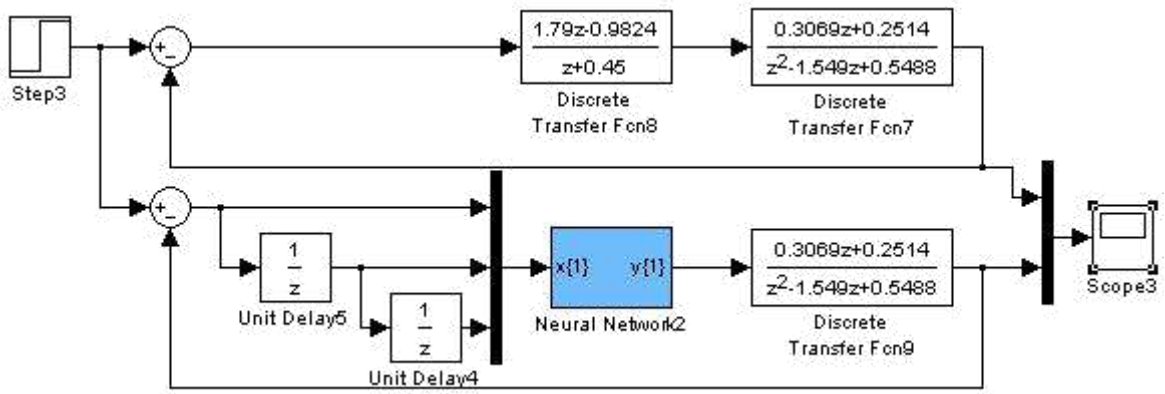

Рисунок 10.6 – Модель систем керування з класичним регулятором оптимальним за швидкодією та нейромережевим регулятором

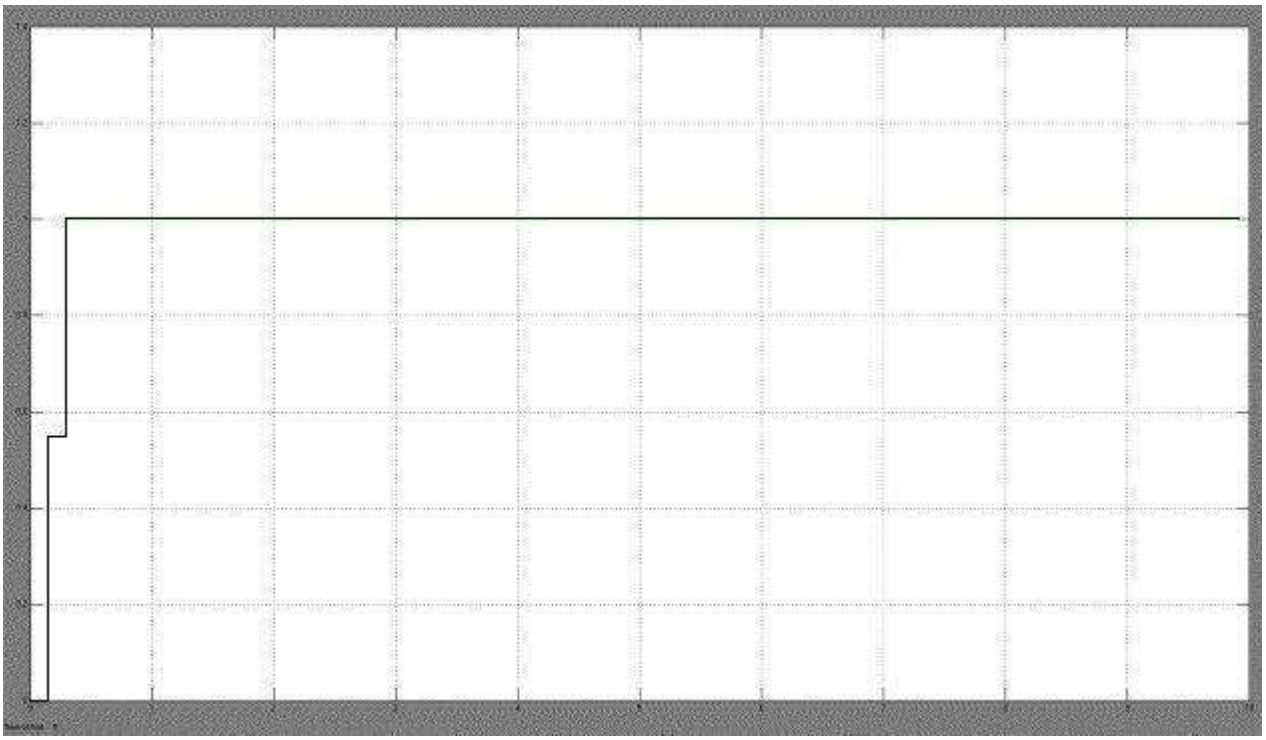

Рисунок 10.7 – Перехідні процеси двох систем керування

6. Синтезуємо нейронну мережу для наступних параметрів:

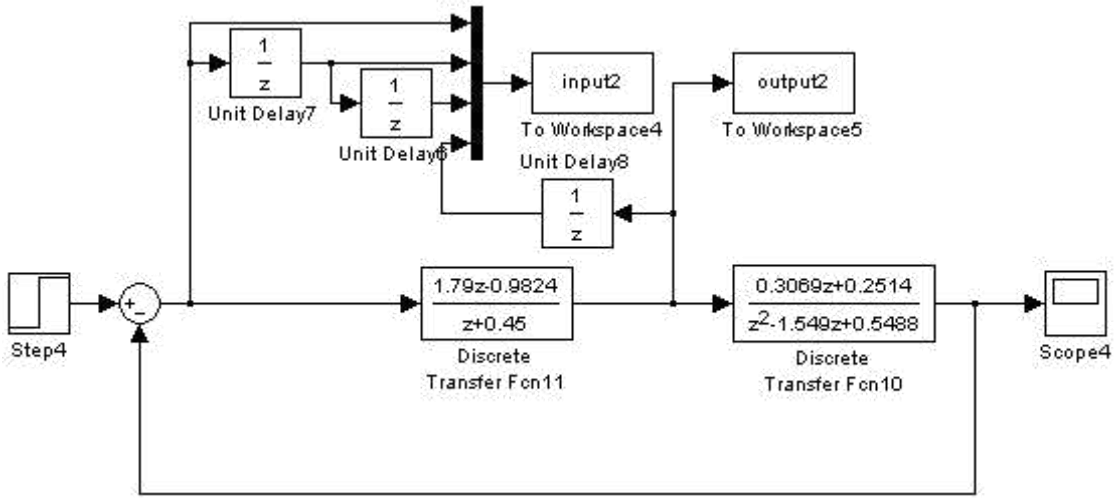

Рисунок 10.8 – Модель системи керування з регулятором оптимальним за швидкодією

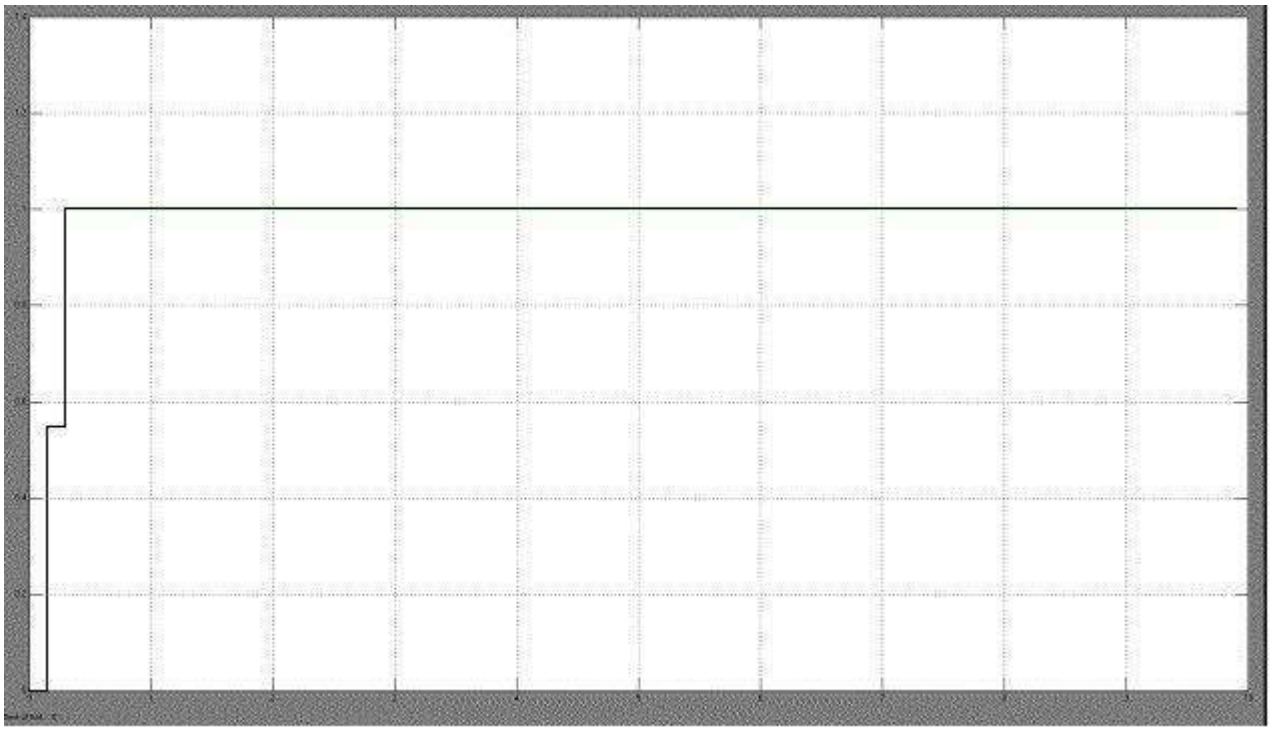

Рисунок 10.9 – Перехідні процеси двох систем керування

ЛАБОРАТОРНА РОБОТА № 11

Побудова нейромережевих моделей багатомірних об'єктів керування

Мета роботи: Використовуючи програмне забезпечення «MIMO-Plant» визначити мінімальну реалізацію нейромережевої моделі багатомірного об'єкта керування.

Короткі теоретичні відомості

Реальні промислові об'єкти керування зазвичай є багатовимірними, тобто мають кілька входів і кілька виходів MIMO (multi input, multi output) [29]. У ряді випадків такі об'єкти можна промоделювати, нехтуючи другорядними впливами, а також другорядними керованими величинами. В результаті такого обґрунтованого спрощення можна отримати просту модель з двома (керуючим і збурюючим), а в деяких випадках і з одним впливом, і з однією керованою величиною. Це істотно спрощує як аналіз об'єкта, так і розробку системи автоматичного керування ним. Однак у багатьох випадках необхідно забезпечити керування декількома керованими величинами об'єкта, його вихідними величинами, шляхом впливу на декілька його входів, тобто зміни керуючих величин, плюс до того слід врахувати ще й кілька збурень. Особливість керування та моделювання такого, досить складного в описі об'єкта, полягає в тому, що може статися, що вплив по одному входу призводить до зміни не однієї, а відразу декількох керованих величин. Наприклад, вертоліт, якщо пілот вертольота бажає прискоритися, то він, природно збільшує подачу газу. Однак тільки це не призводить до збільшення швидкості: несучий гвинт починає обертатися швидше, і вертоліт прагне піднятися. Більш того, прискорення обертання несучого гвинта змушує вертоліт ще й повертатися навколо вертикальної осі, змінюючи напрямок руху. Отже, щоб прискорити політ вертольота на тій же висоті і по тому ж напрямку, пілот повинен збільшити подачу палива і одночасно збільшити тягу заднього гвинта, змінюючи кут атаки його

лопатей, а також нахилити вертоліт вперед (точніше, перекосити площина обертання лопатей головного гвинта), щоб додана збільшенням газу потужність витрачалася на політ вперед.

Нейромережеві системи керування являють собою новий високотехнологічний напрямок в теорії керування та відносяться до класу нелінійних динамічних систем. Однією з головних рис нейронних мереж є універсальні апроксимінаційні властивості. Нейронні мережі дозволяють з будь-якою точністю обраховувати довільну неперервну функцію багатьох змінних $1-\frac{1}{1+e^{-x}}=\frac{1}{1+e^{x}}$, або $1-f(x)=f(-x)$. $f(s_1,...,s_n)$. Отже, з їх допомогою можливо як завгодно точно апроксимувати функцію, породжену будь-якою неперервною системою в тому числі і MIMO-об'єктами.

Для вирішення завдань ідентифікації і керування найбільш адекватними є багатошарові нейронні мережі прямого поширення (БНМПР). При використанні БНМПР для реалізації процедури ідентифікації динамічних систем необхідно визначити «зовнішню» і «внутрішню» структури нейронної мережі. «Зовнішня» структура нейромережевої моделі повністю визначається вектором входу і вектором виходу. При виборі «внутрішньої» структури нейромережевої моделі необхідно визначити:

- кількість прихованих шарів;
- кількість нейронів у кожному схованому шарі;
- вид активаційної функції для кожного шару.

Збільшення числа нейронів у схованому шарі і збільшення числа прихованих шарів підвищують репрезентативні можливості нейронної мережі, тобто дають можливість моделювати більш складні взаємозв'язки, але призводять до значних труднощів при практичній реалізації, збільшення часових витрат як на навчання, так і на роботу в режимі прогнозування. Це і пояснює факт прагнення використовувати мінімальну реалізацію БНМПР в більшості технічних додатків.

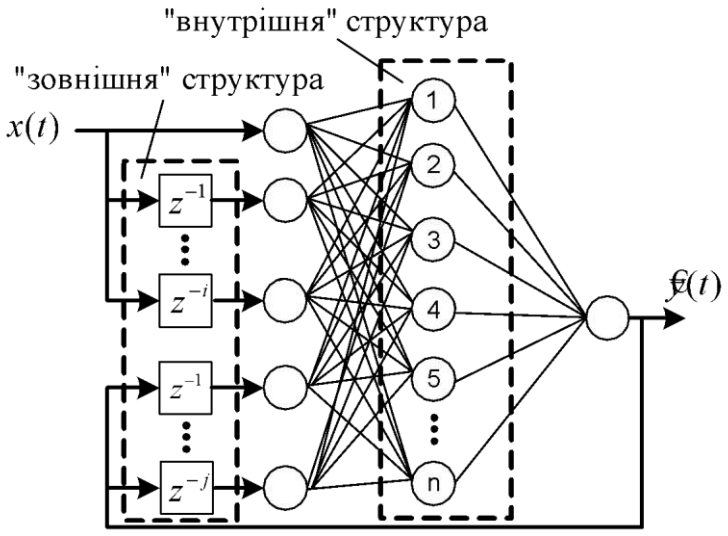

Рисунок 11.1 – Структурна схема багатошарової нейронної мережі прямого розповсюдження

Існує ряд теорем, які математично обґрунтовують апроксимуючу властивість нейронних мереж, з яких випливає що, багатошарові нейронні мережі з одним прихованим шаром, сигмоїдальними (сигмоїдальна, гіперболічний тангенс) функціями активації та лінійною функцією активації вихідного шару, може апроксимувати функції з заданою точністю в разі відсутності обмеження на число нейронів у схованому шарі. На рисунку 1 представлена структура БНМПР. Виходячи з вищевикладеного, завдання побудови нейромережевої моделі зводитися до визначення «зовнішньої» і «внутрішньої» структури, а саме визначення кількості затриманих входів *i*, затриманих сигналів виходу мережі *j*, і кількості нейронів прихованого шару *n*.

Основна ідея роботи полягає в побудові оптимальної, в силу деякого критерію, моделі за результатами спостережень над вхідними та вихідними змінними системи. На практиці реалізація процедури ідентифікації (рисунок 2) вимагає вирішення цілого ряду допоміжних завдань, основними з яких є:

- планування/проведення експерименту і попередня обробка експериментальних даних;

- вибір модельної структури;

- оцінка (оптимізація параметрів) моделі;

- прийняття рішення про адекватність моделі.

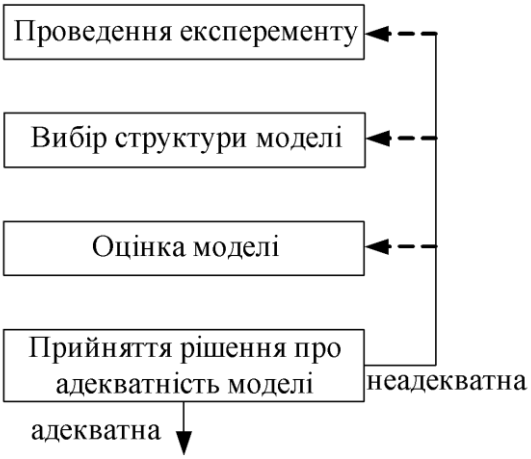

Рисунок 11.2 – Загальна схема реалізації процедури ідентифікації

Порядок виконання роботи

Для вирішення даної задачі було розроблено програмний пакет «MIMO-Plant» в середовищі MatLab [29].

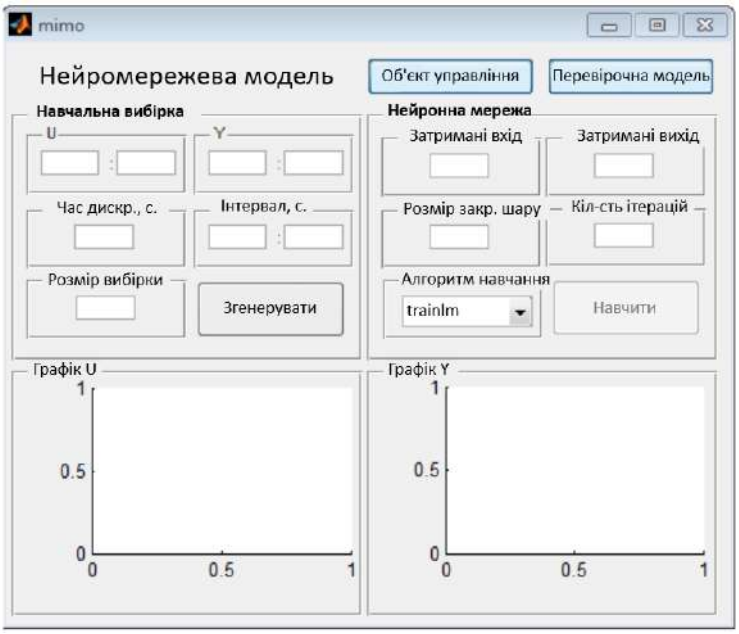

Рисунок 11.3 – Головне вікно програмного пакету «MIMO-Plant»

При натисканні на кнопку «Об'єкт керування» відкриється модель об'єкта керування.

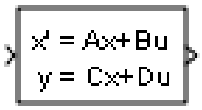

Рисунок 11.4 – Математична модель об'єкта керування

Завдання об'єкта відбувається за допомогою блоку State-Space. Цей блок реалізує систему, поведінку якої можна визначити як:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$y = \mathbf{C}x + \mathbf{D}u,$$

де x – вектор станів, u – вхідний вектор, а y є вихідним вектором.

Матриця коефіцієнтів повинна мати такі характеристики:

- матриця **A** повинна бути n на n матриця, де n-число станів;
- матриця **B** повинна бути n на m матриця, де m-число входів;
- матриця **C** повинна бути r на n матриця, де r-число виходів;
- матриця **D** повинна бути r на m матрицю.

Ширина вхідного вектора залежить від кількості стовпців в матрицях **B** і **D**. Ширина вихідного вектора залежить від кількості рядків у матрицях **C** і **D**.

Підготовчим етапом створення нейромережі є генерація навчальної вибірки, яка відбувається при натисканні кнопки «Сгенерировать».

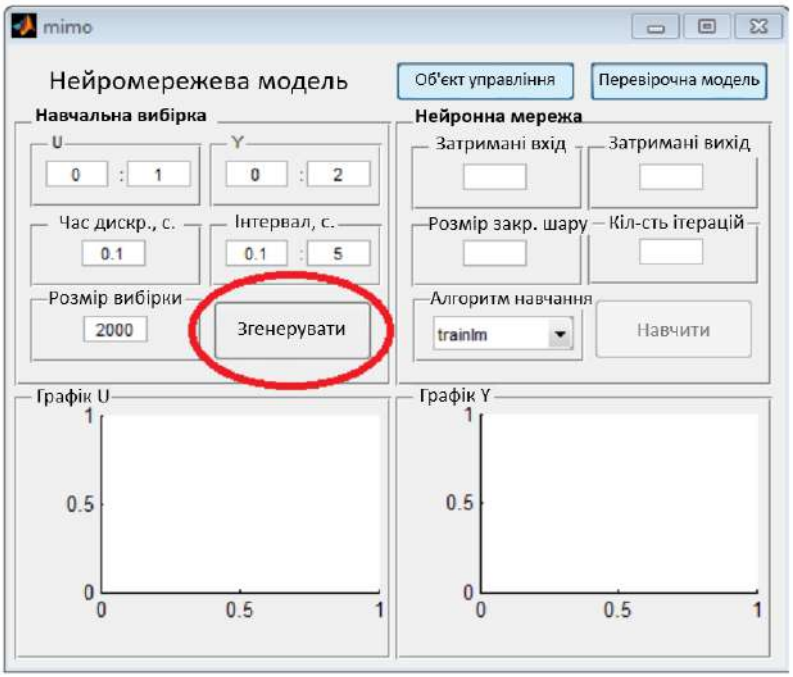

Рисунок 11.5 – Генерація навчальної вибірки

У полі навчальної вибірки необхідно задати параметри необхідні для генерації навчальної вибірки:

1. Задати діапазон зміни входу ($U_{\max}$, $U_{\min}$) і виходу ($Y_{\max}$, $Y_{\min}$) ОУ. При виборі $Y_{\max}$, не потрібно обмежувати вихід об'єкта керування, тобто $Y_{\max}$ вибирати великим усталеного значення при $U_{\max}$.

2. Розмір навчальної вибірки слід вибирати таким, щоб вона містила всі можливі комбінації амплітуд і частот з робочого діапазону системи (при часу дискретизації 0.1 рекомендуємо брати розмір навчальної вибірки порядку 2000-4000).

3. Інтервал визначає мінімальну і максимальну тривалість тестового сигналу. Причому максимальне значення повинно бути не менше часу встановлення перехідного процесу ОУ при стрибкоподібному впливі. Якщо всі поля групи «Навчальна вибірка» заповнені, то в області *PlantInput* і *PlantOutput* повинні відображатися вид вхідного впливу на ОУ та реакція ОУ на вхідні дії.

Після того як дані готові, переходимо до завдання структури МНСПР і її навчанню. Для цього необхідно заповнити поле «Нейронна мережа».

1. В полі «Розмір прихованого шару» вказуємо кількість нейронів *n*.

2. В полі «Кількість затриманих входів» вводимо значення *i*.

3. В полі «Кількість затриманих виходів» вводимо значення *j*.

4. В групі параметрів «Нейронна мережа», вибираємо алгоритм навчання (trainlm) і кількість навчальних ітерацій (200).

5. Натискаємо кнопку «Обучить».

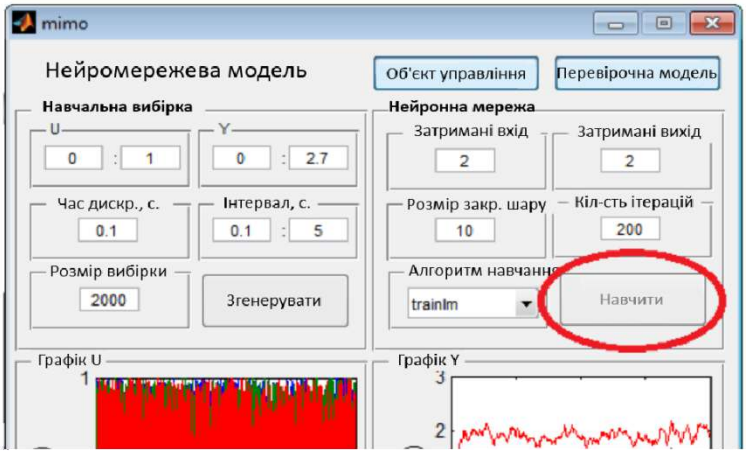

Рисунок 11.6 – Навчання нейромережевих моделей MIMO-об'єктів керування

Результатом виконання процедури навчання буде згенерована нейронна мережа, яка являє собою набір нейромережевих моделей ОУ (рисунок 11.7).

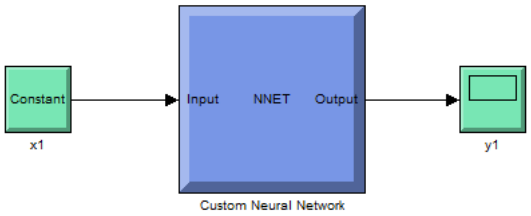

Рисунок 11.7 – Згенерована нейронна мережа

Для того, щоб подивитися перехідні процеси нейронної мережі та багатовимірного об'єкту керування, необхідно скористатися перевірочною моделлю, яка визивається при натисканні кнопки «Перевірочна модель». В результаті цих дій ми відкриємо наступну модель:

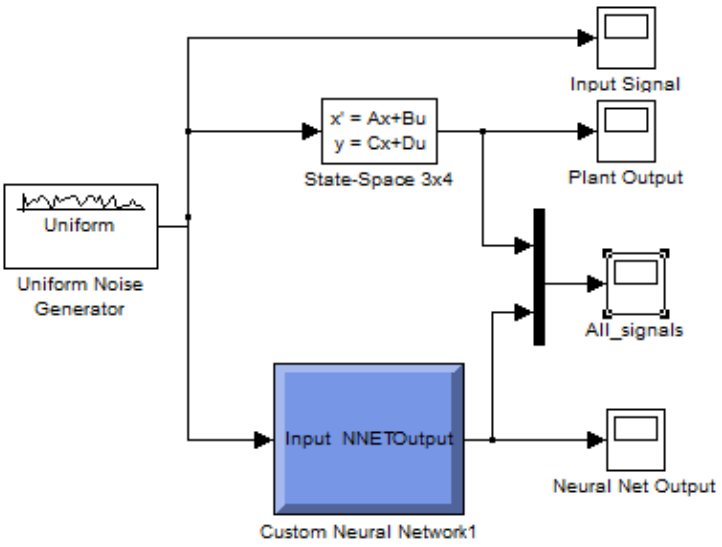

Рисунок 11.8 – Перевірочна модель

Нейронна мережа «Custom Neural Network1», яка зображена на рисунку 8, була отримана в результаті копіювання з вікна що з'являється після навчання нейронної мережі та підставлена у перевірочну модель.

При натисканні на блок «All_signals» перевірочної моделі ми можемо побачити графічне зображення виходу нейронної мережі та стандартного об'єкту керування, а також візуально порівняти отримані результати (рисунок 8).

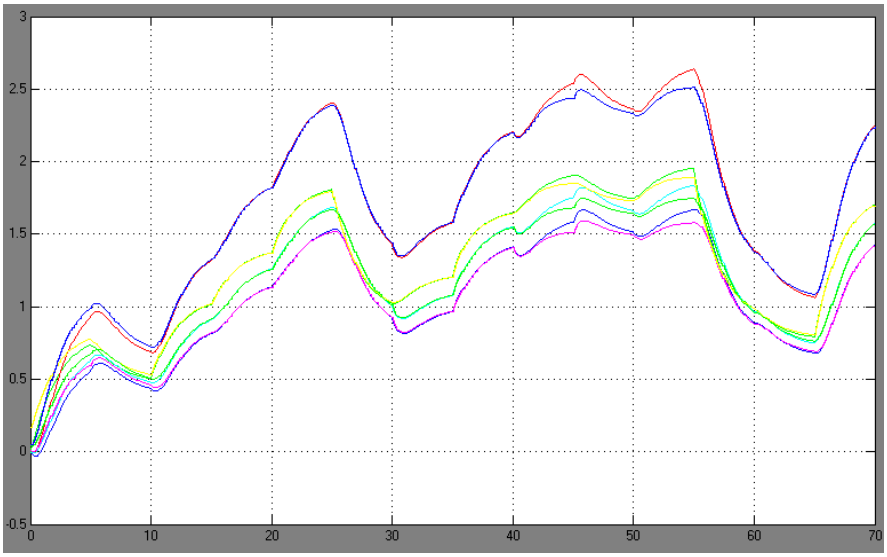

Рисунок 11.9 – Графічне зображення виходу нейронної мережі та стандартного об'єкту керування

Для обчислення сумарної середньоквадратичної помилки потрібно скористатися розробленою моделлю, де:

1) На вхід подається вхідний сигнал генератором сигналу.

2) Об'єкт керування задається блоком State-Space.

3) Під об'єктом керування знаходяться 16 варіацій структури нейромережі.

4) У блоці «Display» відображається сумарна середньоквадратична помилка, яка математично описується наступною формулою:

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{z}(\hat{y} - y)^2}{z}},$$

де z – кількість значень похибки (вибирати рівне кількості елементів вектора *tout* з workspace Matlab); y – вихід моделі об'єкта або системи; $\hat{y}$ – вихід нейромережевої моделі об'єкта або системи.

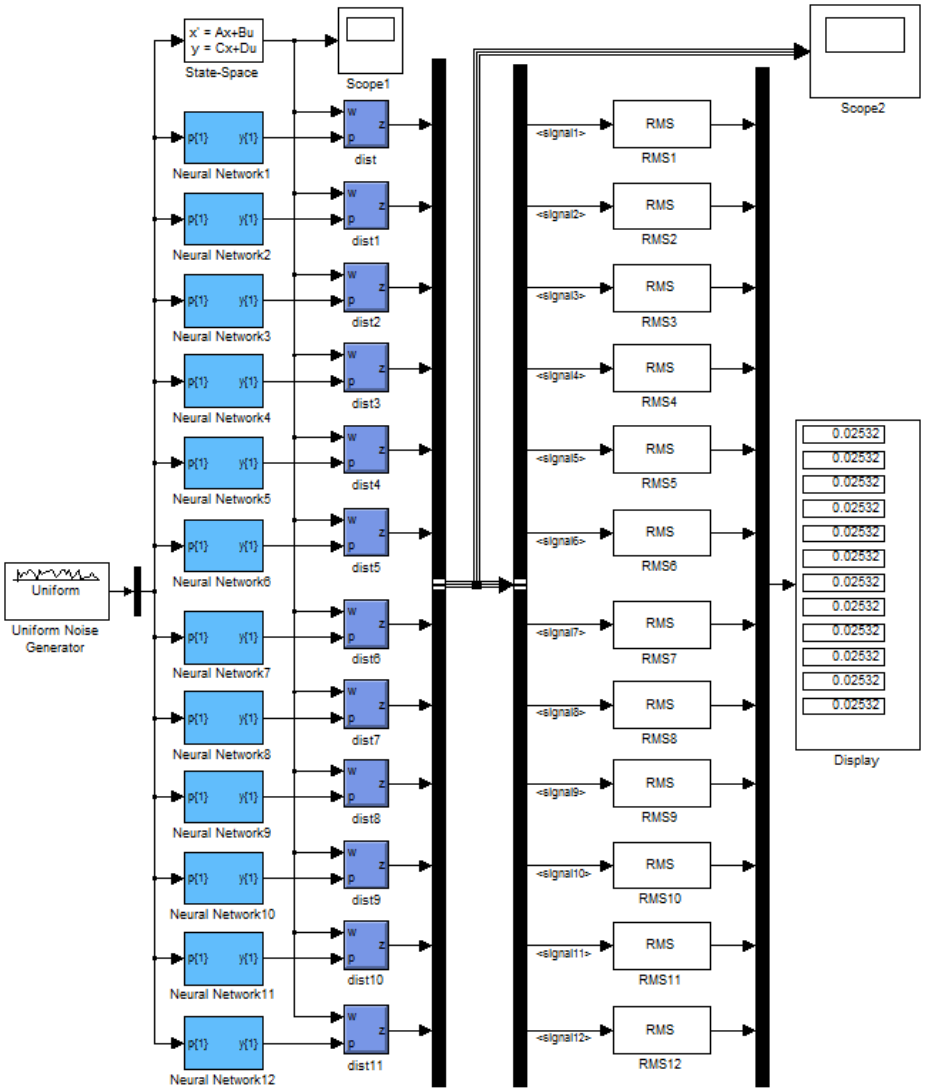

Рисунок 11.10 – Модель обчислення сумарної середньоквадратичної помилки

Далі розглянемо приклад дослідження нейромережевих моделей MIMO-об'єкту керування, який має 3 вхідних та 4 вихідних параметри, в програмному пакеті «MIMO-Plant».

Крок 1. Задаємо MIMO-об'єкт:

$$\mathbf{A} = \begin{pmatrix} -1.23 & 0 & 0 \\ 2.5 & -0.56 & 0 \\ 0 & 3 & -2.3 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0.2 & 1 & 0.3 \\ 0 & 0.4 & 0.1 \\ 0 & 0.2 & -1 \end{pmatrix},$$

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0.3 \\ 1 & 0 & 0.2 \\ 0.1 & 0.2 & 0.3 \\ 0.2 & 0 & 0.3 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Для того, щоб задати об'єкт керування, вводимо матриці A,B,C,D як параметри у вікно блоку «State-space», який визивається при натисканні кнопки «Объект управления» головного меню програми.

Крок 2. Вводимо необхідні параметри для генерації навчальної вибірки, а саме:

1) Задаємо діапазон зміни входу від 0 до 1 та діапазон зміни виходу від 0 до 2.7.

2) Обираємо час дискретизації 0.1 с.

3) Обираємо інтервал від 0.1с до 5с.

4) Обираємо розмір навчальної вибірки 2000.

5) Натискаємо кнопку «Згенерувати».

Крок 3. Створюємо та навчаємо нейронну мережу. Для цього заповнюємо поля блоку «Нейронная сеть» головного меню програми, а саме:

1. У полі «Розмір прихованого шару» вказуємо кількість нейронів n=5.

2. У полі «Кількість затриманих входів» вводимо значення i=0.

3. У полі «Кількість затриманих виходів» вводимо значення j=1.

4. У полі «Алгоритм навчання» обираємо алгоритм навчання (trainlm) і кількість навчальних ітерацій (200). Нажимаємо кнопку «Обучить».

Крок 4. Після того, як нейронна мережа створена та навчена, підставляємо її у модель обчислення сумарної середньоквадратичної помилки. У блок «State-space» цієї моделі задаємо MIMO-об'єкт. В блок «Uniform Noice Generator» задаємо матрицю нижнього значення вхідного сигналу([0 0 0]) та матрицю верхнього значення вхідного сигналу ([1 1 1]). Кількість параметрів генератора залежить від обраного об'єкту керування. В даному випадку, оскільки маємо об'єкт керування з 3 вхідними параметрами, записуємо відповідно наведені вище параметри блоку «Uniform Noice Generator».

Крок 5. Повторюємо крок 2-3 і створюємо 108 нейронних мереж з різними параметрами затриманих входів та виходів, а також різною кількістю нейронів у прихованому шарі. Для цього ми варіюємо вводимі

параметри поля «Розмір прихованого шару» від 5 до 16 нейронів, поля «Кількість затриманих входів» від 0 до 2 входів, а також поля «Кількість затриманих виходів» від 1 до 3 виходів.

Крок 6. Після того, як всі нейронні мережі створені, навчені та підставлені у модель обчислення сумарної середньоквадратичної помилки, запускаємо модель та отримуємо значення середньоквадратичної помилку у блоці «Display».

Табл. 11.1 – Результати дослідження нейромережевих моделей

i_jn	0_1	0_2	0_3	1_1	1_2	1_3	2_1	2_2	2_3
5	0.593	0.6415	0.663	0.02072	0.03368	0.01907	0.007993	0.03166	0.934
6	0.6336	0.565	0.6564	0.012	0.01389	0.05195	0.01004	0.878	0.04302
7	0.6814	0.912	0.6008	0.007073	0.02914	0.0373	0.01277	0.789	0.00856
8	0.7195	0.4416	0.8858	0.02093	0.02749	0.0159	0.01474	0.3705	0.06729
9	0.7148	0.5647	0.7339	0.01271	0.01793	0.01482	0.01554	0.02248	0.927
10	0.6226	0.899	0.3688	0.009068	0.01324	0.04971	0.01829	0.7073	0.956
11	0.675	0.5014	0.841	0.03134	0.01825	0.03142	0.07851	0.913	0.00784
12	0.5785	0.50435	0.7661	0.007977	0.003114	0.02115	0.004742	0.03676	0.03685
13	0.6581	0.936	0.856	0.003391	0.01355	0.01059	0.0186	0.3052	0.03426
14	0.783	0.3992	0.887	0.009263	0.009352	0.01895	0.02771	0.002146	0.878
15	0.981	0.731	0.734	0.01303	0.01212	0.05857	0.02063	0.02022	0.02057
16	0.837	0.6442	0.558	0.001715	0.001401	0.003874	0.005804	0.1045	0.1761

З таблиці 1 видно, що оптимальною структурою для даної моделі з 3 вхідними та 4 вихідними параметрами є структура з 16 нейронами у прихованому шарі, 1 затриманим входом та 2 затриманими виходами. При цьому отримано значення середньоквадратичної помилки 0.001401. Також у роботі було досліджено різні багатовимірні об'єкти 2x2, 4x2, 4x3, 3x3. Для системи 2x2 оптимальною структурою виявилась структура з 5 нейронами у прихованому шарі, затриманим входом та 3 затриманими виходами (5-1-3).

Середньоквадратична помилка при цьому становила 0.004196. Для моделі 4x2 – структура (5-2-3) з помилкою 0.001372, моделі 4x3 – структура (16-1-2) з помилкою 0.001452, моделі 3x3 – структура (16-1-3) з помилкою 0.005575.

Завдання на лабораторну роботу

За варіантом передавальної функції з додатка А знайти нейромережеву модель з найменшою похибкою

ЛАБОРАТОРНА РОБОТА 12

Адаптивна нейро-нечітка система

Мета роботи: отримання і закріплення знань про методи моделювання та принципи функціонування нейронечітких систем, а також формування практичних навичок з конструювання нейронечітких мереж в пакеті MATLAB.

Короткі теоретичні відомості

Як правило, в стані нестабільної економіки, яка характеризується частою зміною економічних умов, прийняття управлінських рішень здійснюється в умовах невизначеності, внаслідок чого задача планування економічної діяльності та прогнозування її результатів є однією з найбільш складних і неоднозначних.

Існує ряд класичних методів прогнозування економічних показників, які базуються на апараті математичної статистики, серед яких виділяються методи аналізу та моделювання часових рядів, методи багатовимірного регресійного аналізу. Особливістю зазначених методів є необхідність чіткої специфікації конструюються моделей, крім того, додаткові труднощі для використання даних методів створює не стаціонарність досліджуваних економічних процесів.

Перспективним напрямком в області вирішення завдань прогнозування є застосування апарату штучних нейронних мереж і нейронечітких мереж.

Принцип роботи нейронної мережі полягає в тому, що по наявного набору даних конструюється певна залежність між вхідними та вихідними змінними системи, при цьому в процесі навчання мережі настроюються параметри (ваги) одержуваних функціональних відносин. (Як правило, виявлення і визначення цієї залежності в явному вигляді не представляється можливим в силу вищевказаних причин). Модель нейронної мережі позиціонується як "чорний ящик" внаслідок того, що внутрішній алгоритм її

настройки не "прозорий", і отримані результати і взаємозв'язку складно інтерпретувати.

Нечіткі нейронні мережі або гібридні мережі покликані об'єднати в собі переваги нейронних мереж і систем нечіткого виводу. З одного боку, вони дозволяють розробляти і подавати моделі систем у формі правил нечітких продукцій, які володіють наочністю і простотою змістовної інтерпретації. З іншого боку, для побудови правил нечітких продукцій використовуються методи нейронних мереж, що є більш зручним і менш трудомістким процесом для системних аналітиків.

У даних методичних вказівках розглядаються основні поняття нейронечітких мереж, а також засоби їх розробки, що надаються системою MatLAB.

Для прогнозування використовуємо нечітку мережу TSK. Узагальнену схему виведення моделі TSK при використанні M правил і N змінних x_j можна представити у вигляді:

$$IF\ (x_1\ IS\ A_1^{(1)})\ AND\ (x_2\ IS\ A_2^{(1)})\ AND\ \dots\ AND\ (x_n\ IS\ A_n^{(1)}),$$
$$THEN\ y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$IF\ (x_1\ IS\ A_1^{(M)})\ AND\ (x_2\ IS\ A_2^{(M)})\ \ AND\ \dots\ AND\ (x_n\ IS\ A_n^{(M)}),$$
$$THEN\ y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} x_j\ .$$

Умова $IF\ (x_i\ IS\ A_i)$ реалізується функцією фазифікація, яка представляється узагальненої функцією Гаусса окремо для кожної змінної:

$$\mu_A(x_i) = \frac{1}{1 + \left(\frac{x_i - c_i}{\sigma_i} \right)^{2b_i}}, \tag{12.1}$$

де $\mu_A(x_i)$ представляє оператор A_i. В нечітких мережах доцільно задавати це умова в формі алгебраїчного добутку, з якого випливає, що для k-го

правила виведення

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1 + \left(\frac{x_i - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}}} \right]. \tag{12.2}$$

При M правилах виведення агрегування вихідного результату мережі здійснюється за формулою

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{N} w_j} \left(p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right), \tag{12.3}$$

яку можна представити у вигляді

$$y(x) = \frac{1}{\sum_{k=1}^{N} w_k} \left(\sum_{k=1}^{M} w_k y_k(x) \right), \tag{12.4}$$

де $y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} x_j$

Присутні в даному виразі ваги w_k інтерпретуються як значимість компонентів $\mu_A^{(k)}(x)$, визначених формулою (12.1). При цьому умови формулою (12.4) можна зіставити багатошарову структуру мережі (рис. 12.1). У такій мережі виділяється п'ять шарів:

- перший шар виконує роздільну фазифікація кожної змінної $x_i \ (i = 1, 2, \ldots, N)$, визначаючи для кожного k-го правила виведення

значення коефіцієнта приналежності $\mu_A^{(k)}(x_i)$ відповідно застосовуваної функції фазифікація. Це параметричний шар з параметрами навчання $c_j^{(k)}$, $\sigma_j^{(k)}$, $b_j^{(k)}$, що підлягають адаптації в процесі

- другий шар (Непараметричний) виконує агрегування окремих змінних x_i, визначаючи результуюче значення коефіцієнта приналежності $w_k = \mu_A^{(k)}(x)$ для вектора x (рівень активізації правила виводу) відповідно до формули (12.2);

- третій шар представляє собою генератор функції *TSK*, *N* розраховує значення $y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} x_j$. В цьому шарі також проходить множення сигналів $y_k(x)$ на значення w_k, сформовані на попередньому шарі. Це параметричний шар, в якому адаптації підлягають лінійні ваги p_{kj} для k = 1, 2, ..., M і j = 1, 2, ..., N,що визначають функцію слідства моделі TSK;

- четвертий шар (непараметричний) складають два нейрона-суматора, один з яких розраховує зважену суму сигналів $y_k(x)$, а другий визначає суму ваг $\sum_{k=1}^{M} w_k$;

- останній, п'ятий шар (нормалізує), складається з єдиного вихідного нейрона, в якому ваги піддаються нормалізації відповідно до формули (12.4). Вихідний сигнал y(x) визначається виразом, відповідним залежності (12.3),

$$y(x) = f(x) = \frac{f_1}{f_2}. \qquad (12.5)$$

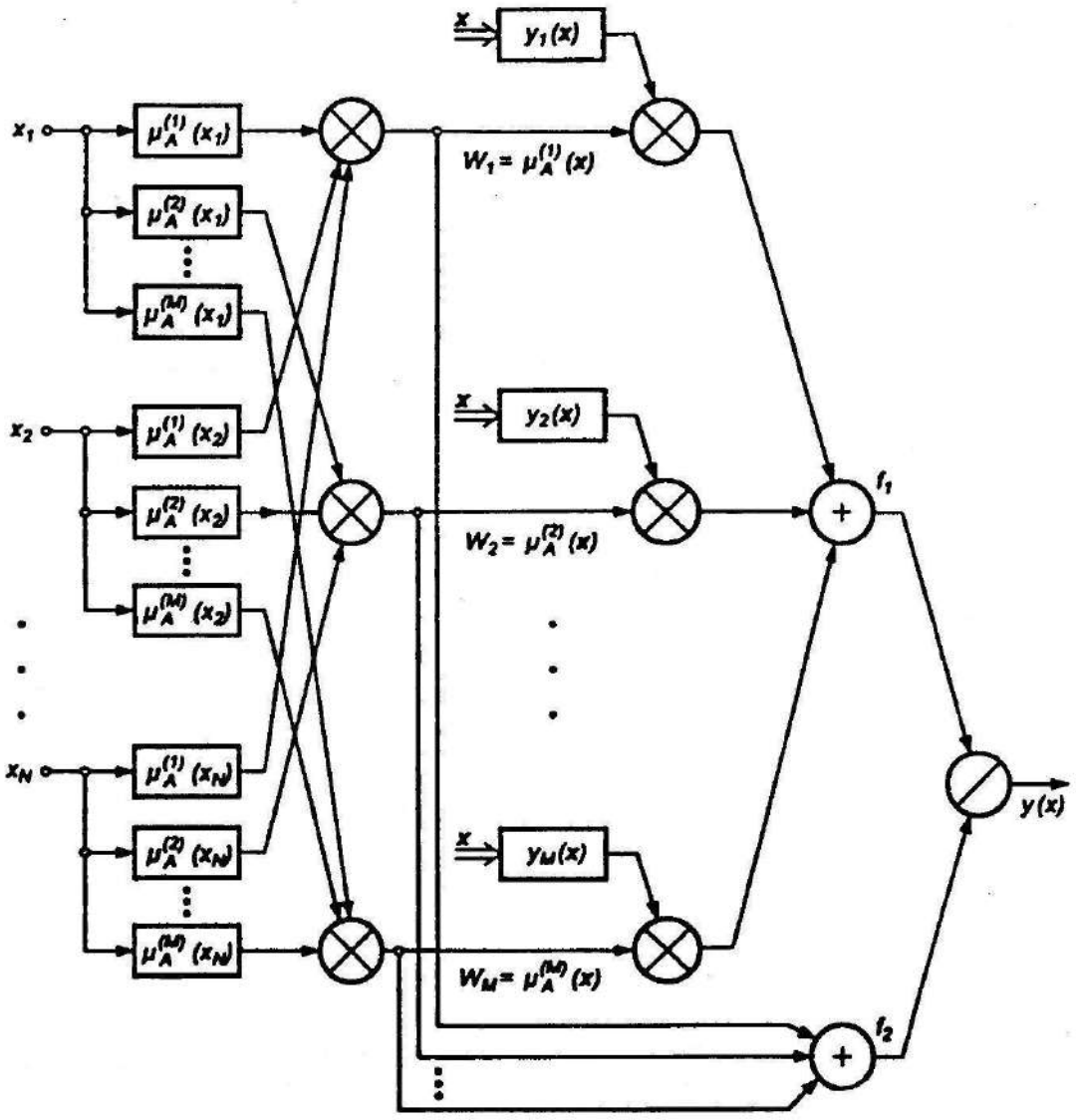

Рисунок 12.1 – Структура нечіткої нейронної мережі TSK

З наведеного опису випливає, що нечітка мережа TSK містить тільки два параметричних шару (перший і третій), параметри яких уточнюються в процесі навчання. Параметри першого шару будемо називати нелінійними параметрами, оскільки вони відносяться до нелінійної функції (12.1), а параметри третього шару - лінійними вагами, тому що вони відносяться до параметрів p_{kj} лінійної функції TSK.

При уточненні функціональної залежності (12.4) для мережі TSK отримуємо:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]} \sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]\left[p_{k0} + \sum_{j=1}^{N} p_{kj} x_j\right]. \quad (12.4)$$

Якщо прийняти, що в конкретний момент часу параметри умови зафіксовані, то функція y (x) є лінійною щодо змінних x_i (i = 1, 2, ..., N).

При наявності N вхідних змінних кожне правило формує N + 1 змінних $p_j^{(k)}$ лінійної залежності TSK. При M правилах виведення це дає M (N + 1) лінійних параметрів мережі. У свою чергу, кожна функція приналежності використовує три параметри (c, s, b), що підлягають адаптації. Якщо прийняти, що кожна змінна x_i характеризується власною функцією приналежності, то при M правилах виведення ми отримаємо 3MN нелінійних параметрів. В сумі це дає M (4N + 1) лінійних і нелінійних параметрів, значення яких повинні підбиратися в процесі навчання мережі.

На практиці для зменшення кількості параметрів що адаптуються оперують меншою кількістю незалежних функцій приналежності для окремих змінних, керуючись правилами, в яких комбінуються функції приналежності різних змінних. Якщо прийняти, що кожна змінна x_i має т різних функцій приналежності, то максимальна кількість правил, які можна створити при їх комбінуванні, складе: M = mN (при трьох функціях приналежності, що поширюються на дві змінні, це 32 = 9 правил виводу). Таким чином, сумарна кількість нелінійних параметрів мережі при M правилах виведення зменшується з 3MN в загальному випадку до 3NM1 / N. Кількість лінійних параметрів при подібній модифікації залишається без змін, тобто M (N + 1).

Гібридна мережа як адаптивна система нейронечіткого виведення. Гібридна мережа являє собою багатошарову нейронну мережу спеціальної структури без зворотних зв'язків, в якій використовуються звичайні (НЕ нечіткі) сигнали, ваги і функції активації, а виконання операції

підсумовування засноване на використанні фіксованої Т-норми, S-Кнорм або деякої іншої безперервної операції. При цьому значення входів, виходів і ваг гібридної нейронної мережі є речові числа з відрізка [0, 1].

Основна ідея, покладена в основу моделі гібридних мереж, полягає в тому, щоб використовувати існуючу вибірку даних для визначення параметрів функцій приналежності, які найкраще відповідають деякій системі нечіткого виведення. При цьому для знаходження параметрів функцій приналежності використовуються відомі процедури навчання нейронних мереж.

У пакеті Fuzzy Logic Toolbox системи MatLAB гібридні мережі реалізовані у формі так званої адаптивної системи нейронечіткого виведення ANFIS. З одного боку, гібридна мережа ANFIS - це нейронна мережа з єдиним виходом і декількома входами, які представляють собою нечіткі лінгвістичні змінні. При цьому терми вхідних лінгвістичних змінних описуються стандартними для системи MatLAB функціями належності, а терми вихідної змінної представляються лінійної або постійними функціями приналежності.

З іншого боку, гібридна мережа ANFIS являє собою систему нечіткого виводу FIS типу Сугено нульового або першого порядку, в якій кожне з правил нечітких продукцій має постійну вагу, рівний 1.

Моделювання та реалізація нейронечіткої мережі в середовищі MatLAB

У пакеті Fuzzy Logic Toolbox системи MatLAB гібридні мережі реалізовані у формі адаптивних систем нейро-нечіткого виводу ANFIS. При цьому розробка і дослідження гібридних мереж виявляється можливою:

- в інтерактивному режимі за допомогою спеціального графічного редактора адаптивних мереж, що отримав назву редактора ANFIS;
- в режимі командного рядка за допомогою введення імен відповідних функцій з необхідними аргументами безпосередньо у вікно

команд системи MatLAB.

Редактор ANFIS дозволяє створювати або завантажувати конкретну модель адаптивної системи нейронечіткого виведення, виконувати її навчання, візуалізувати її структуру, змінювати і налаштовувати її параметри, а також використовувати налаштовану мережу для отримання результатів нечіткого виведення.

ANFIS є абревіатурою Adaptive Neuro-Fuzzy Inference System - (адаптивна нейро-нечітка система). ANFIS-редактор дозволяє автоматично синтезувати з експериментальних даних нейро- нечіткі мережі. Нейронечіткого мережу можна розглядати як одну з різновидів систем нечіткого логічного висновку типу Сугено. При цьому функції приналежності синтезованих систем налаштовані (навчені) так, щоб мінімізувати відхилення між результатами нечіткого моделювання та експериментальними даними. Завантаження ANFIS-редактора здійснюється по команді anfisedit. В результаті виконання цієї команди з'явиться графічне вікно (рис. 11.2), на якому розміщені функціональні області ANFIS-редактора.

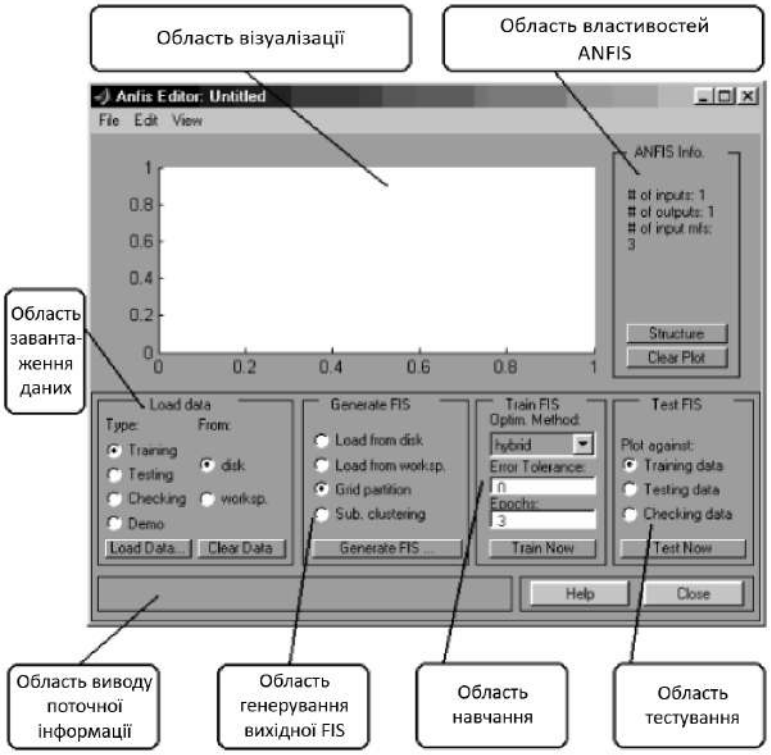

Рисунок 12.2 – Основне вікно ANFIS-редактора

ANFIS-редактор містить 3 верхніх меню - File, Edit і View, область візуалізації, область властивостей ANFIS, область завантаження даних, область генерування вихідної системи нечіткого логічного висновку, область навчання, область тестування, область виведення поточної інформації, а також кнопки Help і Close, які дозволяють викликати вікно довідки і закрити ANFIS-редактор, відповідно.

Меню File і View однакові для всіх GUI-модулів, які використовуються з системами нечіткого логічного висновку.

Меню Edit. Загальний вигляд меню приведений на рис. 7.3.

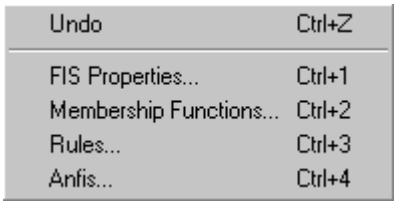

Рисунок 12.3 – Меню Edit

Команда Undo скасовує раніше виконану дію. Виконується також після натискання <Ctrl + Z>.

Команда *FIS Properties* – відкриває *FIS*-редактор. Ця команда може бути також виконана натисканням <Ctrl + 1>.

Команда *Membership Functions* – відкриває редактор функцій приладдя. Ця команда може бути також виконана натисканням <Ctrl + 2>.

Команда Rules – відкриває редактор бази знань. Ця команда може бути також виконана натисканням <Ctrl + 3>.

Команда Anfis – відкриває ANFIS-редактор. Ця команда може бути також виконана натисканням <Ctrl + 3>. Зауважимо, що дана команда, запущена з ANFIS-редактора, не приводить до виконання будь-яких дій, так цей редактор вже відкритий. Однак в меню Edit інших GUI- модулів, використовуваних з системами нечіткого логічного висновку, додається

команда Anfis, що дозволяє відкрити ANFIS-редактор з цих модулів.

Область візуалізації. У цій області виводиться два типи інформації:

- при навчанні системи – крива навчання у вигляді графіка залежності помилки навчання від порядкового номера ітерації;
- при завантаженні даних і тестуванні системи - експериментальні дані і результати моделювання.

Експериментальні дані і результати моделювання виводяться у вигляді безлічі точок в двомірному просторі. При цьому по осі абсцис відкладається порядковий номер рядка даних у вибірці (навчальної, яка тестує або контрольної), а по осі ординат - значення вихідної змінної для цього рядка вибірки. Використовуються наступні маркери:

- блакитна точка (.) - тестуюча вибірка;
- блакитна окружність (o) - навчальна вибірка;
- блакитний плюс (+) - контрольна вибірка;
- червона зірочка (*) - результати моделювання.

Область властивостей ANFIS. В області властивостей ANFIS (ANFIS info) виводиться інформація про кількість вхідних і вихідних змінних, про кількість функцій приладдя для кожної вхідної змінної, а також про кількість рядків в вибірках. У цій області розташовано дві кнопки: Structure і Clear Plot.

Натискання кнопки Structure відкриває нове графічне вікно, в якому система нечіткого логічного висновку представляється у вигляді нейронечіткої мережі. В якості ілюстрації наведена нейронечіткого мережу, яка містить чотири вхідних змінних і одну вихідну (рис. 12.4). У цій системі за три лінгвістичних терми використовуються для оцінки кожної з вхідних змінних і чотири терми - для вихідний.

Натискання кнопки Clear Plot дозволяє очистити область візуалізації.

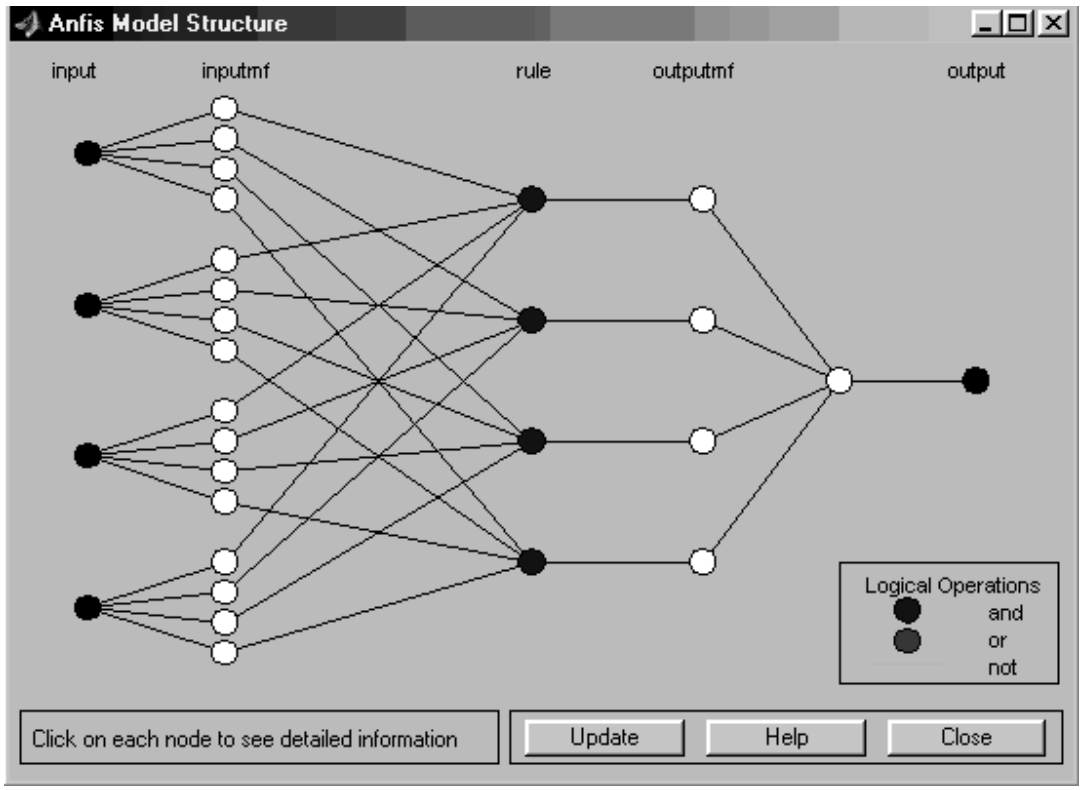

Рисунок 12.4 – Приклад структури нейронечіткої мережі

Область завантаження даних. В області завантаження даних (Load data) розташовані:

- меню вибору типу даних (Type), що містить альтернативи (Traning - навчальна вибірка; Testing - тестирующая вибірка; Checking - контрольна вибірка; Demo - демонстраційний приклад);
- меню вибору джерела даних (From), що містить альтернативи (disk - диск; worksp - робоча область MATLAB);
- кнопка завантаження даних Load Data ..., після натискання якої з'являється діалогове вікно вибору файлу, якщо завантаження даних відбувається з диска, або вікно ввести код вибірки, якщо завантаження даних відбувається з робочої області;
- кнопка очищення даних Clear Data.

Протягом одного сеансу роботи ANFIS-редактора можна завантажувати дані одного формату, тобто кількість вхідних змінних в вибірках має бути однаковим.

Область генерування вихідної системи нечіткого логічного висновку. В області генерування (Generate FIS) розташоване меню вибору способу створення вихідної системи нечіткого логічного висновку. Також можуть бути доступними наступні альтернативи:

- *Load from disk* - завантаження системи з диска;
- *Load from worksp* - завантаження системи з робочою області MATLAB;
- *Grid partition* - генерування системи за методом решітки (без кластеризації);
- *Sub*. Clustering - генерування системи за методом субкластерізації.

В області також розташована кнопка Generate, після натискання якої генерується вихідна система нечіткого логічного висновку.

При виборі Load from disk з'являється стандартне діалогове вікно відкриття файлу.

При виборі Load from worksp з'являється стандартне діалогове вікно ввести код системи нечіткого логічного висновку.

При виборі Grid partition з'являється вікно введення параметрів методу решітки (рис. 12.5), в якому потрібно вказати кількість термів для кожної вхідної змінної і тип функцій приналежності для вхідних і вихідної змінних.

При виборі Sub. clustering з'являється вікно введення наступних параметрів методу субкластерізації (рис. 12.6):

- *Range of influence* - рівні впливу вхідних змінних;
- *Squash factor* - коефіцієнт придушення;
- *Accept ratio* - коефіцієнт, який встановлює у скільки разів потенціал даної точки повинен бути вище потенціалу центру першого кластера для того, щоб центром одного з кластерів була призначена розглянута точка;
- *Reject ratio* - коефіцієнт, який встановлює у скільки разів потенціал даної точки повинен бути нижче потенціалу центру першого

кластера, щоб розглянута точка була виключена з можливих центрів кластерів.

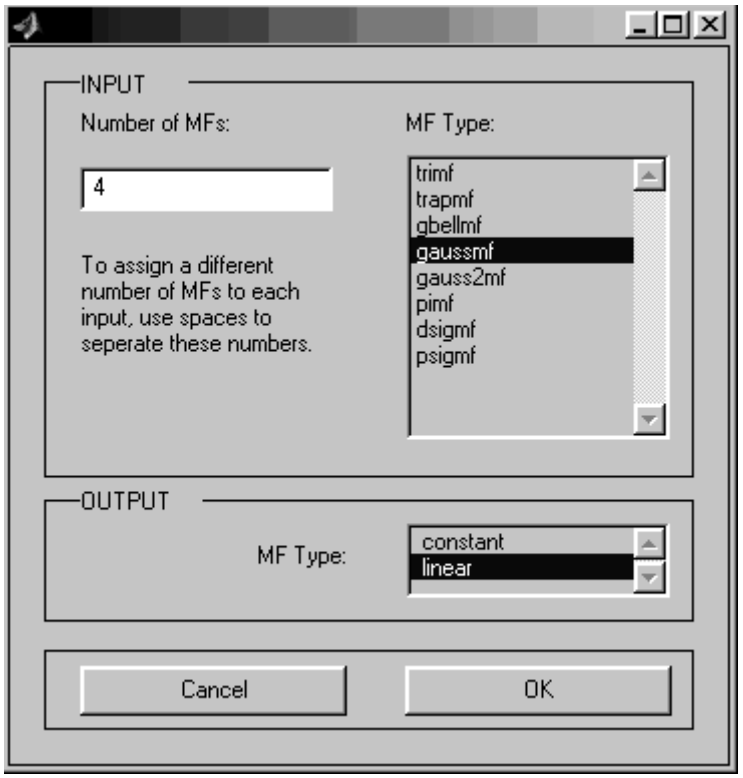

Рисунок 12.5 – Вікно введення параметрів для методу решітки

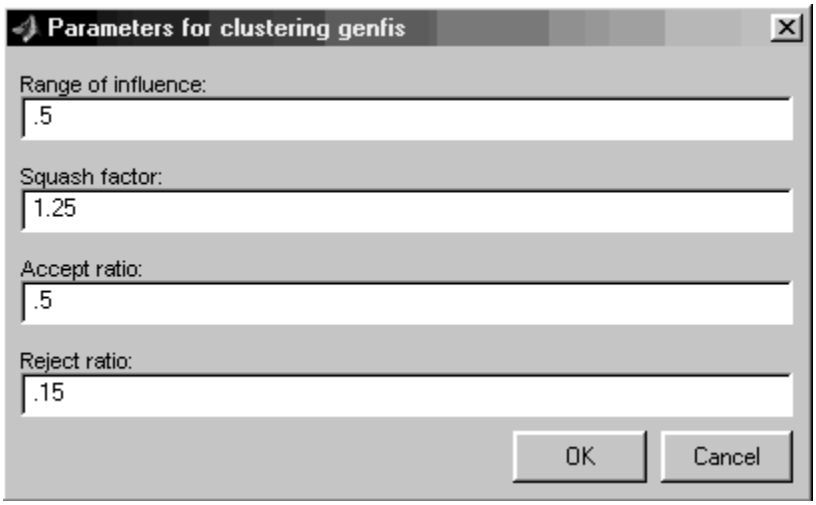

Рисунок 12.6 – Вікно введення параметрів для методу субкластерізації

Область навчання. В області навчання (Train FIS) розташовані меню вибору методу оптимізації (Optim. Method), поле завдання необхідної

точності навчання (Error tolerance), поле завдання кількості ітерацій навчання (Epochs) і кнопка Train Now, натиснувши яку запускає режим навчання. Проміжні результати навчання виводяться в область візуалізації і в робочу область MATLAB. У ANFIS-редакторі реалізовані два методу навчання:

- *backpropa* - метод зворотного поширення помилки, заснований на ідеях методу найшвидшого спуску;
- *hybrid* - гібридний метод, який об'єднує метод зворотного поширення помилки з методом найменших квадратів.

Область тестування. В області тестування (Test FIS) розташовані меню вибору вибірки і кнопка Test Now, після натискання якої відбувається тестування нечіткої системи з висновком результатів в область візуалізації.

Область виводу поточної інформації. У цій області виводиться найсуттєвіша поточна інформація, наприклад, повідомлення про закінчення виконань операцій, значення помилки навчання або тестування і т.п.

Синтез нейронечіткої мережі в середовищі MatLAB

Опис завдання: є вихідні дані зміни фінансового індексу РТС за період з 01.03.2012 по 30.04.2012. Потрібно побудувати нейронечіткого мережу і спрогнозувати значення індексу на 1.05.2012.

Загальна послідовність процесу розробки моделі гібридної мережі може бути представлена в наступному вигляді.

1. Підготовка файлу з навчальними даними (табл. 12.1). Доцільно скористатися редактором електронних таблиць MS Excel. Навчальну вибірку необхідно зберегти в зовнішньому файлі з розширенням * .dat. Алгоритм прогнозування має на увазі те, що кожне наступне значення розраховується на основі декількох попередніх.

Таблиця 12.1 – Набір даних для навчання нейронечіткої мережі

Перша вхідна змінна	Друга вхідна змінна	Третя вхідна змінна	Вихідна змінна
688,72	686,21	667,27	669,26
686,21	667,27	669,26	673,25
667,27	669,26	673,25	688,68
669,26	673,25	688,68	680,86
673,25	688,68	680,86	671,33
688,68	680,86	671,33	669,55
680,86	671,33	669,55	676,20
671,33	669,55	676,20	680,57
669,55	676,20	680,57	698,70
676,20	680,57	698,70	708,59
680,57	698,70	708,59	721,81
698,70	708,59	721,81	712,88
708,59	721,81	712,88	713,15
721,81	712,88	713,14	705,16
712,88	713,14	705,16	706,71
713,14	705,16	706,71	716,55
705,16	706,71	716,55	721,35
706,71	716,55	721,35	736,28
688,72	686,21	667,27	669,26
686,21	667,27	669,26	673,25
667,27	669,26	673,25	688,68
669,26	673,25	688,68	680,86
673,25	688,68	680,86	671,33
688,68	680,86	671,33	669,55
680,86	671,33	669,55	676,20
671,33	669,55	676,20	680,57

2. Відкрити редактор ANFIS. Завантажити файл з навчальними даними. Кнопка завантаження даних Load Data, після натискання якої з'являється діалогове вікно вибору файлу, якщо завантаження даних

відбувається з диска, або вікно ввести код вибірки, якщо завантаження даних відбувається з робочої області.

Зовнішній вид редактора *ANFIS* з завантаженими навчальними даними зображений на рис. 12.7.

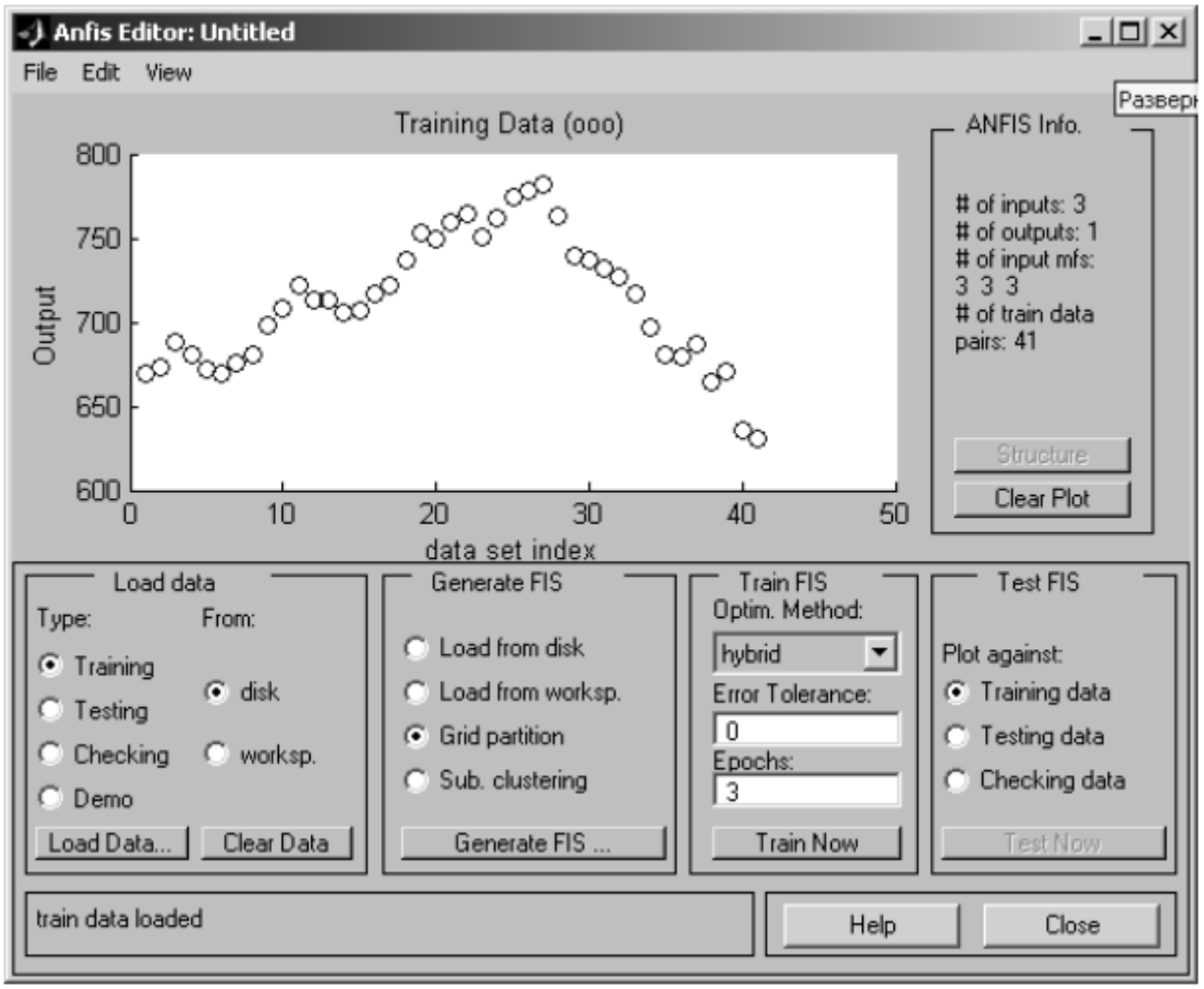

Рисунок 12.7 – Графічний інтерфейс редактора ANFIS після завантаження навчальних даних

3. Після підготовки і завантаження навчальних даних можна згенерувати структуру системи нечіткого виведення FIS типу Сугено, яка є моделлю гібридної мережі в системі MATLAB. Для цієї мети слід скористатися кнопкою Generate FIS в нижній частині робочого вікна редактора. При цьому дві перші опції відносяться до попередньо створеної структурі гібридної мережі, а дві останніх - до форми розбиття вхідних

змінних моделі.

Перед генерацією структури системи нечіткого виведення типу Сугено після виклику діалогового вікна властивостей задамо для кожної з вхідних змінних по три лінгвістичних терма, а в якості типу їх функцій приналежності виберемо трикутні функції.

Після натискання кнопки Generate FIS викликається діалогове вікно із зазначенням числа та типи функцій приналежності для окремих термів вхідних змінних і вихідної змінної (рис. 12.8).

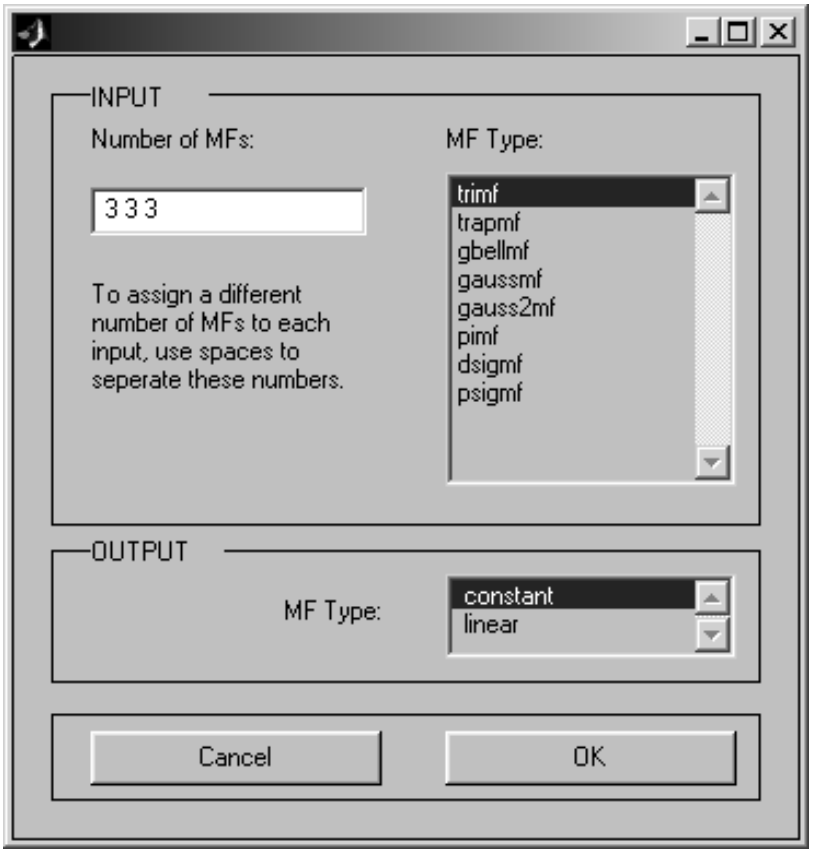

Рисунок 12.8 – Діалогове вікно для завдання кількості і типу функцій належності

4. Після генерації структури гібридної мережі можна візуалізувати її структуру, для чого слід натиснути кнопку Structure в правій частині графічного вікна (рис. 12.9).

Для розглянутого прикладу система нечіткого виведення містить три

вхідних змінних з трьома термами кожна, 27 правил нечітких продукцій, одну вихідну змінну з 27 термами.

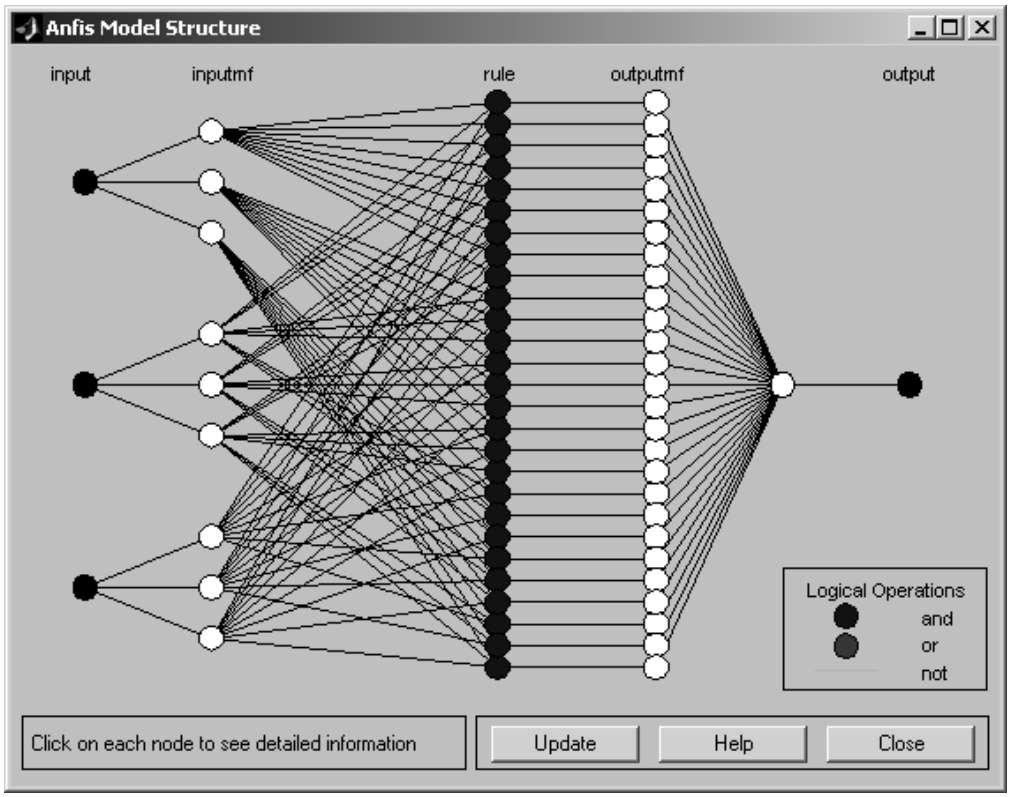

Рисунок 12.9 – Структура згенерованої системи нечіткого виведення

5. Перед навчанням гібридної мережі необхідно задати параметри навчання, для чого слід скористатися наступною групою опцій в правій нижній частині робочого вікна:

1. Вибрати метод навчання гібридної мережі - зворотного поширення (backpropo) або гібридний (hybrid), що представляє собою комбінацію методу найменших квадратів і методу зменшення зворотного градієнта.

2. Встановити рівень помилки навчання (Error Tolerance) - за замовчуванням значення 0 (змінювати не рекомендується).

3. Задати кількість циклів навчання (Epochs) - за замовчуванням значення 3 (рекомендується збільшити для розглянутого прикладу задати його значення рівним 40).

Для навчання мережі слід натиснути кнопку Train now. При цьому хід процесу навчання ілюструється в вікні візуалізації в формі графіка залежності помилки від кількості циклів навчання (рис. 12.10).

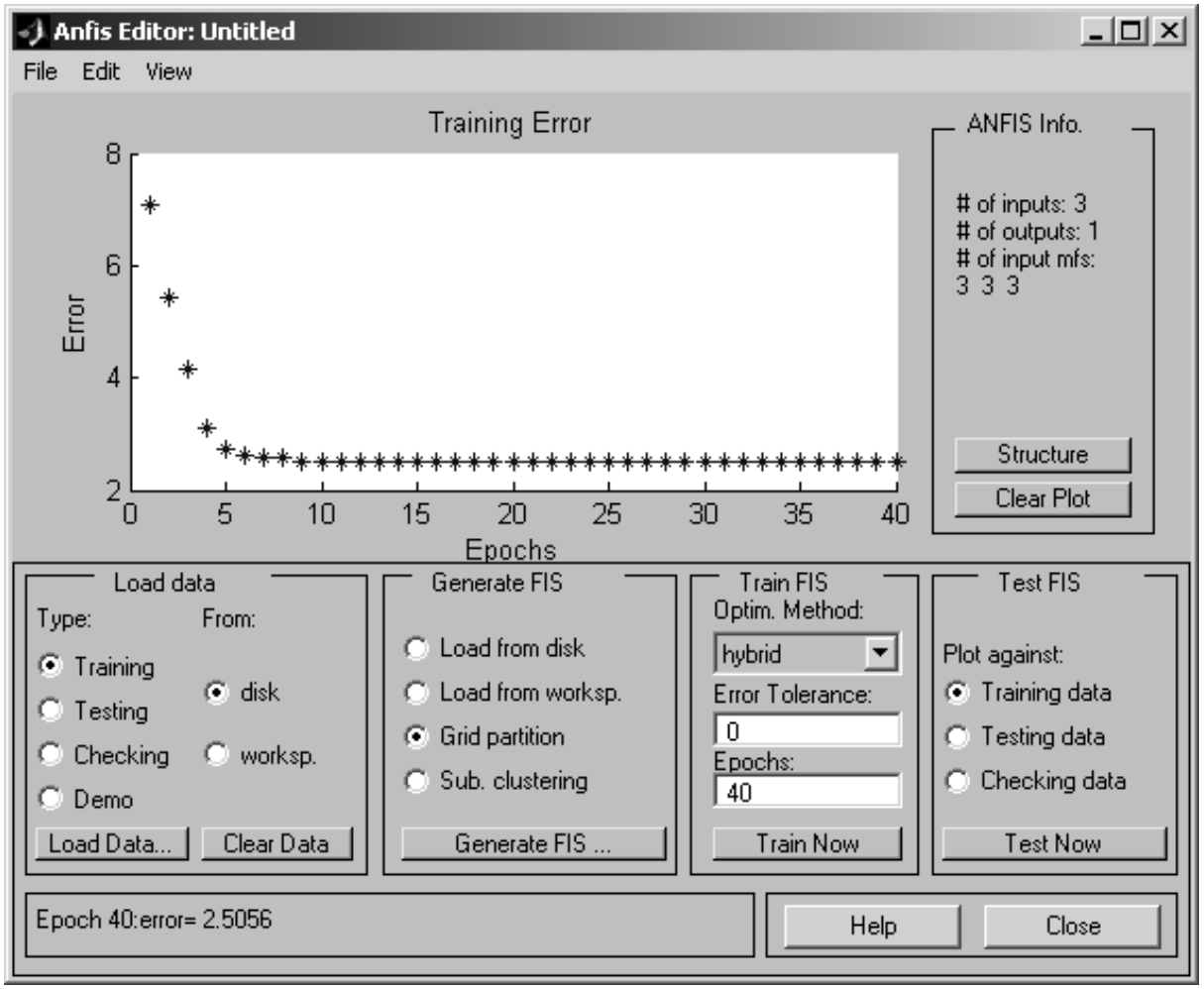

Рисунок 12.10 – Графік залежності помилок навчання від кількості циклів навчання

Подальше налаштування параметрів побудованої і навченої гібридної мережі може бути виконана за допомогою стандартних графічних засобів пакета Fuzzy Logic Toolbox (рис. 12.11 - 12.14). Для цього рекомендується зберегти створену систему нечіткого виводу в зовнішньому файлі з розширенням *.fis, після чого слід завантажити цей файл в редактор систем нечіткого виводу FIS.

6. Виконаємо перевірку адекватності побудованої нечіткої моделі гібридної

мережі. Для цього можна спрогнозувати курс долара на певний день. Для вирішення цього завдання необхідно скористатися функцією evalfis.

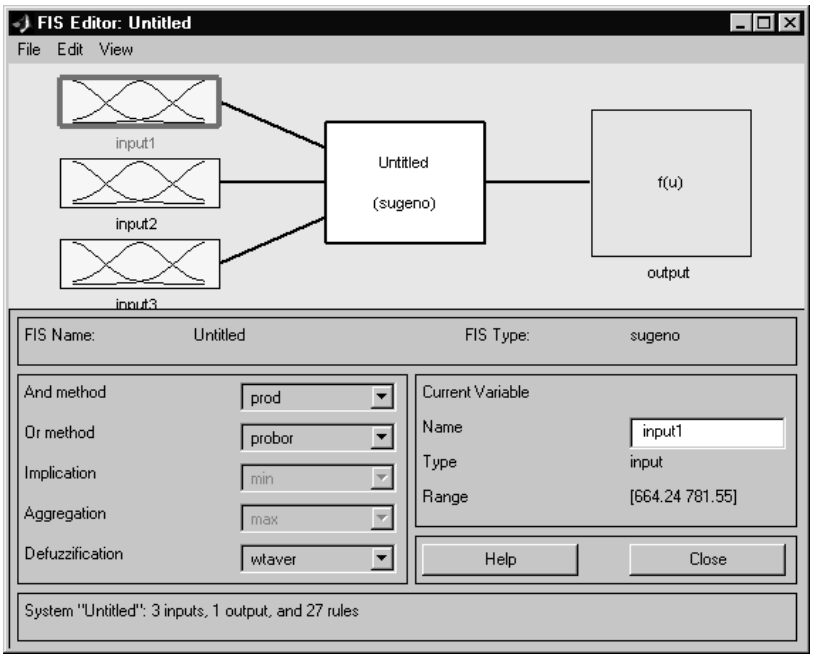

Рисунок 12.11 – Графічний інтерфейс редактора FIS для згенерованої системи нечіткого виведення

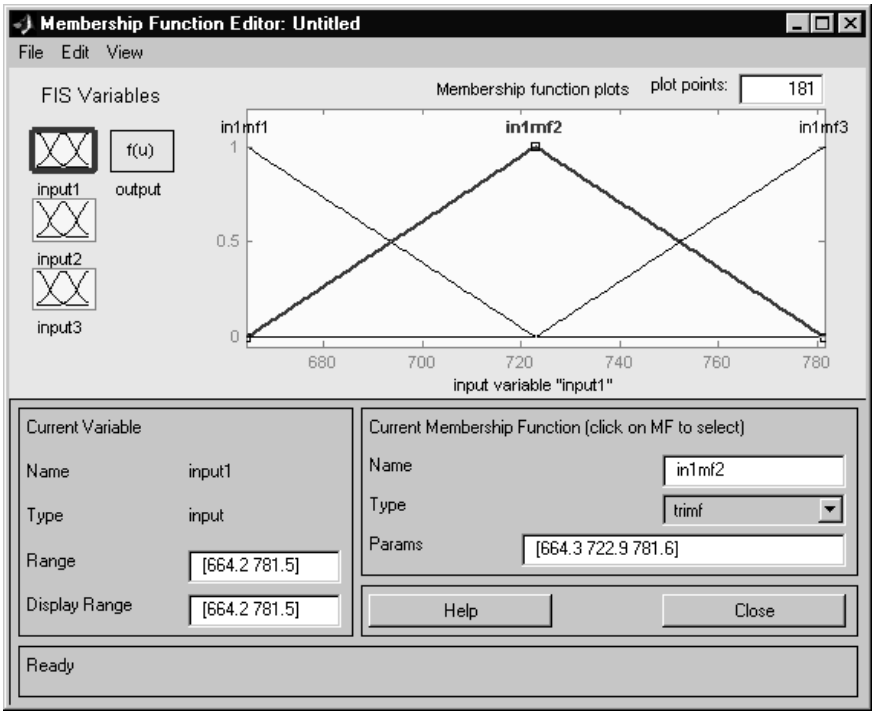

Рисунок 12.12 – Графічний інтерфейс редактора функцій належності побудованої системи нечіткого виведення

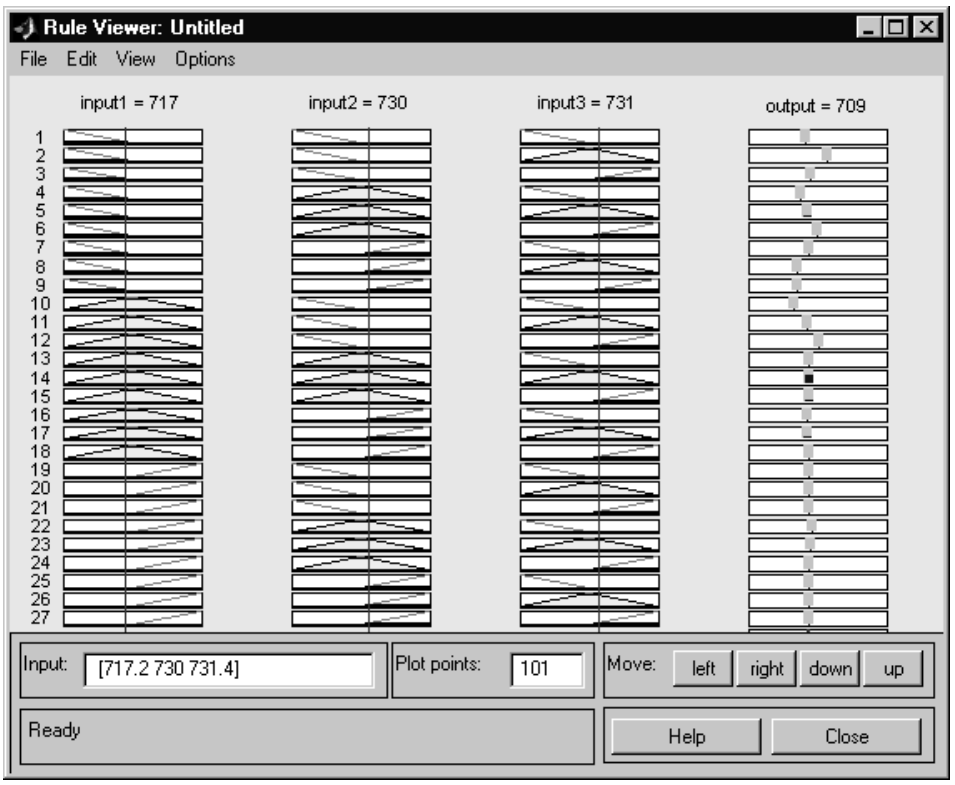

Рисунок 12.13 – Графічний інтерфейс перегляду правил згенерованої системи нечіткого виведення

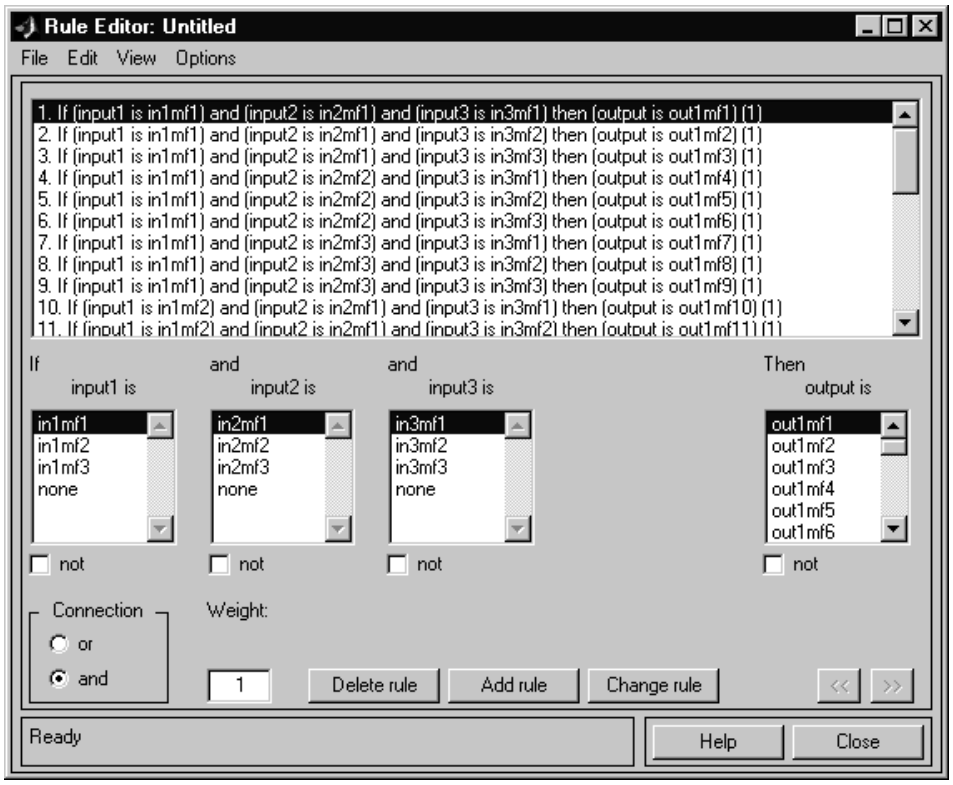

Рисунок 12.14 – Фрагмент бази нечітких правил

Індивідуальні завдання

1. Сформулювати завдання в галузі обчислювальної техніки, для вирішення якої було б обґрунтовано застосування гібридної нейронечіткої мережі.
2. Сформувати вибірку для навчання гібридної нейронної мережі.
3. Згенерувати і візуалізувати структуру гібридної нейронної мережі в системі MATLAB.
4. Навчити гібридну нейронну мережу, при цьому задати і обґрунтувати параметри її навчання.
5. Побудувати систему нечіткого виводу для отриманої гібридної нейронної мережі.
6. Виконати перевірку адекватності побудованої нечіткої моделі гібридної мережі.
7. Оформіть звіт по лабораторній роботі.

ЛАБОРАТОРНА РОБОТА №13

Знаходження мінімуму та максимуму функцій за допомогою генетичних алгоритмів

Мета роботи: Знайти мінімум (мінімізація) і максимум (максимізація) функцій одно- і двох змінних за допомогою вікна тулбоксу gatool. А також побудувати поверхні функцій.

Короткі теоретичні відомості

Генетичний алгоритм – це еволюційний метод пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі. «Батьком-засновником» генетичних алгоритмів вважається Джон Холланд, книга якого «Адаптація в природних і штучних системах» є фундаментальною в цій сфері досліджень [].

Задача кодується таким чином, щоб її вирішення могло бути представлено в вигляді масиву подібного до інформації складу хромосоми. Цей масив часто називають саме так «хромосома». Випадковим чином в масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція. Особи оцінюються з використанням функції пристосування, в результаті якої кожній особі присвоюється певне значення пристосованості, яке визначає можливість виживання особи. Після цього з використанням отриманих значень пристосованості вибираються особи допущені до схрещування (селекція). До осіб застосовується «генетичні оператори» (в більшості випадків це оператор схрещування (crossover) і оператор мутації (mutation), створюючи таким чином наступне покоління осіб. Особи наступного покоління також оцінюються застосуванням

генетичних операторів і виконується селекція і мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму. Таким критерієм може бути:

- знаходження глобального, або надоптимального вирішення;
- вичерпання числа поколінь, що відпущені на еволюцію;
- вичерпання часу, відпущеного на еволюцію.

Генетичні алгоритми можуть використовуватися для пошуку рішень в дуже великих і тяжких просторах пошуку.

Схема роботи генетичного алгоритму представлена на рисунку 13.1.

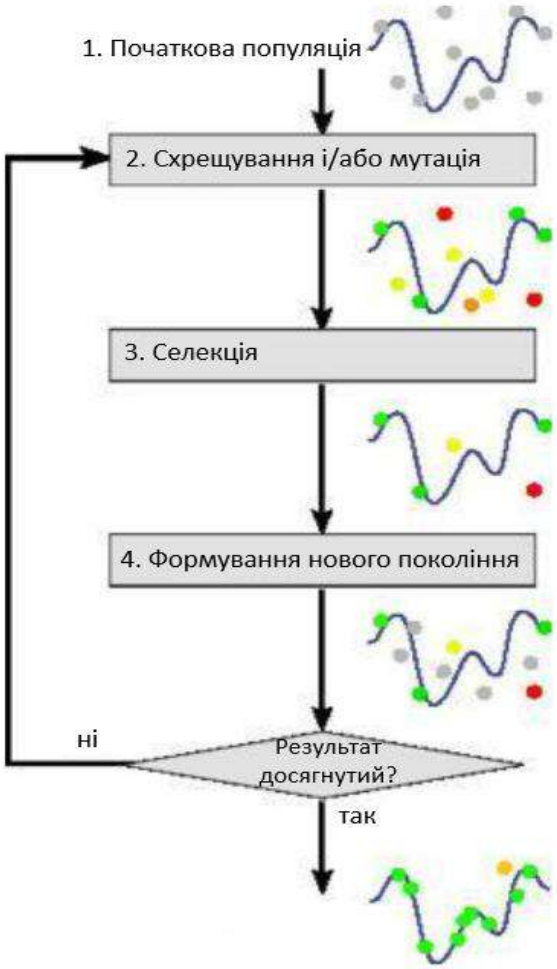

Рисунок 13.1 – Схема роботи генетичного алгоритму

Можна виділити такі етапи генетичного алгоритму:

– Створення початкової популяції;
– Обчислення функції пристосованості для осіб популяції (оцінювання);
– Повторювання до виконання критерію зупинки алгоритму;
– Вибір індивідів із поточної популяції (селекція);
– Схрещення або/та мутація;
– Обчислення функції пристосовуваності для всіх осіб;
– Формування нового покоління;

Порядок виконання роботи

Вибрати варіант завдання на дану лабораторну роботу з додатка Б за номером групи та номером студента в списку групи.

1**)** Мінімізувати функцію однієї змінної:

$$f(x) = 8x - 16 - 12\sqrt[3]{(x+4)^2}$$

2) Максимізувати функцію двох змінних:

$$z(x, y) = \exp(-x^2 - y^2) + \sin(x + y).$$

Рішення:

1) Напишемо M-file для цієї функції і збережемо його в поточній папці під ім'ям ex2.m.

```
function y = ex2(x)
y=-12*(x+4)\(2/3)+8*x-16;
```

Викличемо вікно тулбоксу за допомогою gatool.

У полі fitness function введемо ім'я цільової функції @ex2

Встановимо значення параметрів ГА: кількість особин в популяції =10, кількість поколінь = 100 (у вікні критерію зупинки алгоритму), початковий відрізок = [−4; 1] (рис. 13.1). У розділі plots встановимо прапорці для best fitness, best individual, distance. Клацнемо по кнопці start.

В результаті завершення процесу у вікні final point з'явиться значення змінної *x*, що відповідає мінімуму функції, а у вікні status and result можна побачити знайдене мінімальне значення цільової функції.

Для цієї задачі результати вийшли наступні: мінімум функції досягається точці *x* = - 89,316(рисунок 13.3).

Перший рисунок (рисунок 13.3) відображає зміну значення цільової функції. Видно, що, починаючи з 80 популяції, алгоритм зійшовся до рішення. На другому рисунку (рисунок 13.3) зображена найкраща особина. Третій рисунок відповідає зміні відстані між особинами в поколіннях. Особини стають однаковими (хемінгова відстань = 0) в останніх 18 поколіннях. ГА треба запустити кілька разів, а потім вибрати оптимальне рішення. Це пов'язано з тим, що початкова популяція формується з використанням генератора випадкових чисел.

Переконатися в правильності рішення можна, побудувавши графік функції (рисунку 13.4).

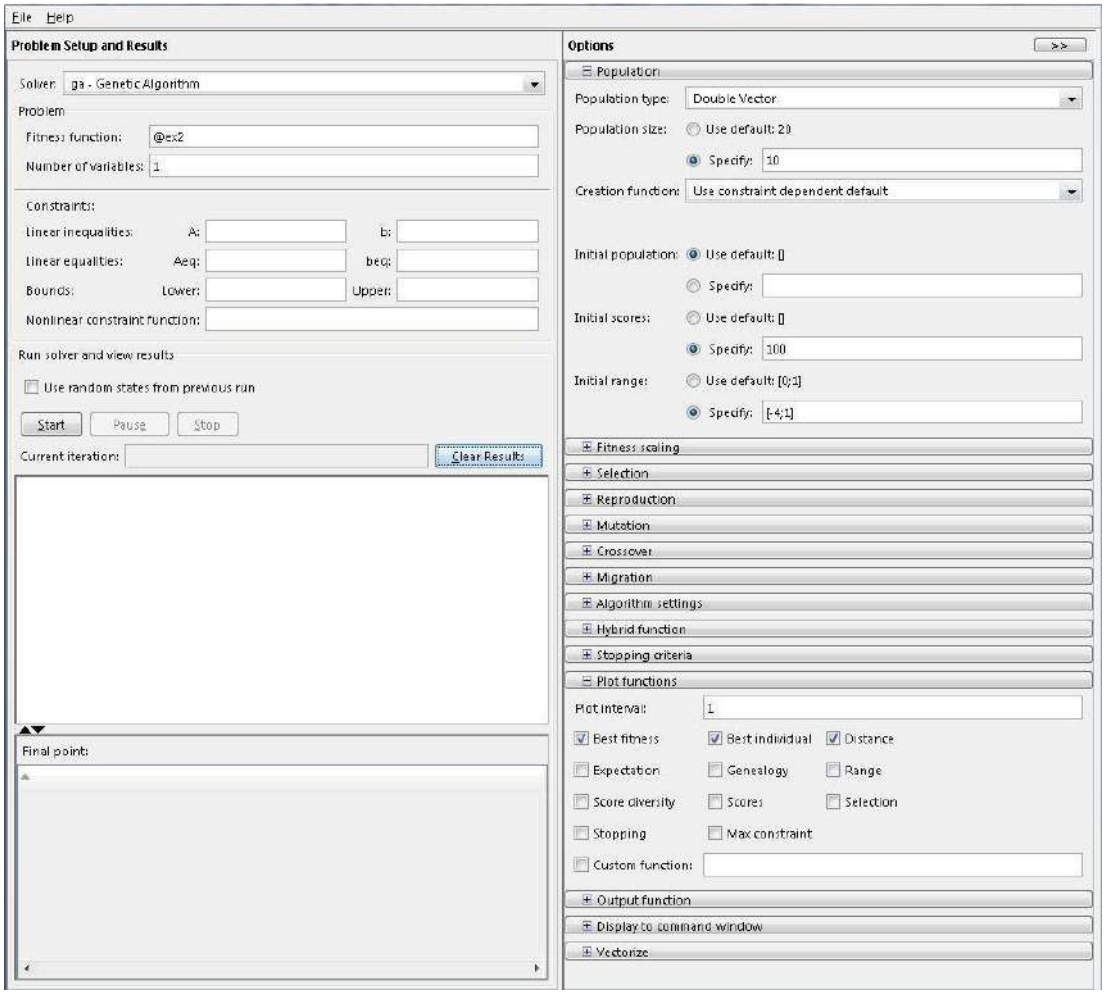

Рисунок 13.2 – Графічна оболонка gatool

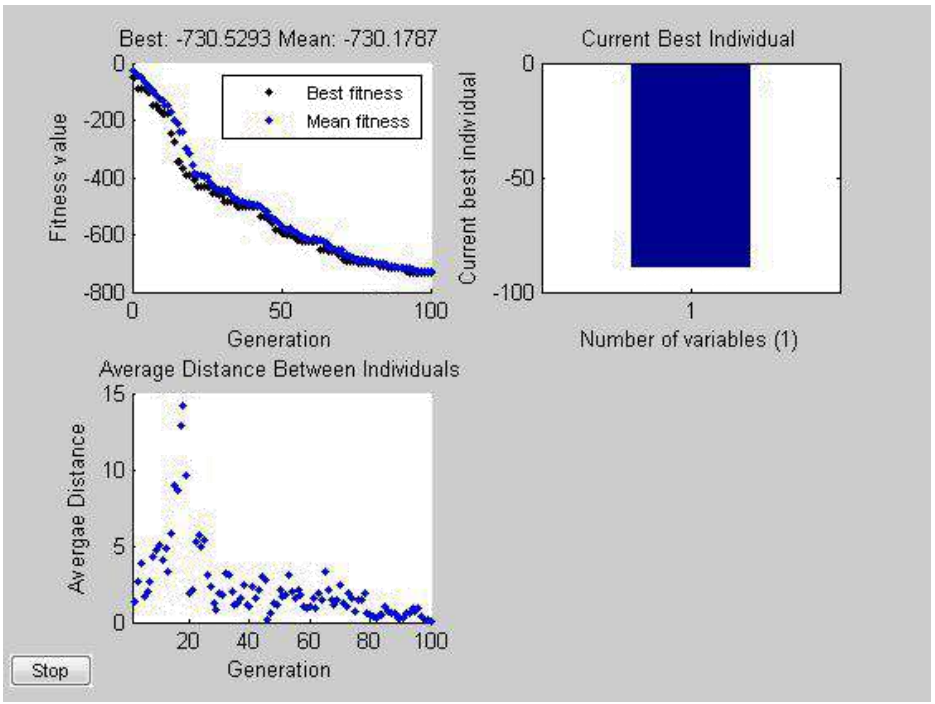

Рисунок 13.3 – Графічний аналіз рішення

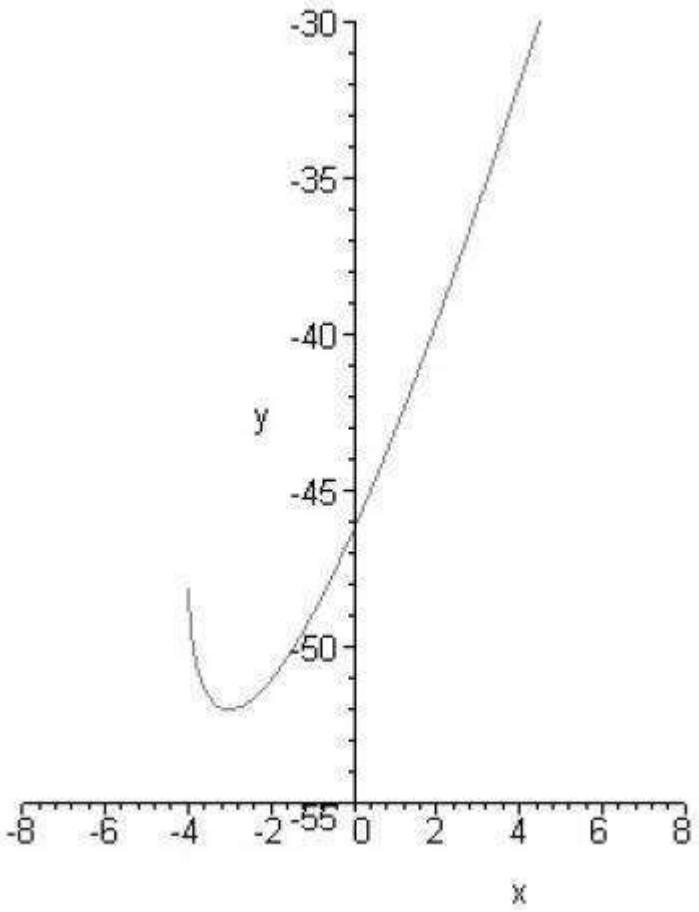

Рисунок 13.4 – Графік функції

Те ж саме можна було б отримати, використовуючи функції gaoptimset і ga. Щоб подивитися M - File виберете в меню "File" вікна "Genetic Algoritm Tool" команду "Generate M - file", збережете файл під іншим ім'ям і переглянете код. Для даної задачі отримали:

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] = ex2q
% This is an auto generated M file to do optimization
% with the Genetic Algorithm and
% Direct Search Toolbox. Use GAOPTIMSET for default
% GA options structure.
% Fitness function
fitnessFunction = @ex2;
% Number of Variables
nvars = 1 ;
```

```
% Start with default options
options = gaoptimset;
% Modify some parameters
options = gaoptimset(options,'PopInitRange',[-4 ; -1 ]);
options = gaoptimset(options,'PopulationSize',10);
options = gaoptimset(options,'MutationFcn',
{@mutationgaussian 1 1});
options = gaoptimset(options,'Display','off');
options = gaoptimset(options,'PlotFcns', {@gaplotbestf
@gaplotbestindiv @gaplotdistance });
% Run GA
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);
```

2) Тулбокс по ГА вирішує тільки завдання мінімізації, для знаходження максимуму функції $f(x)$ слід мінімізувати функцію $f(x)$. Це пояснюється тим, що точка мінімуму – $f(x)$ є деякою точкою $f(x)$, в якій досягається максимум.

Напишемо M-file для функції $z(x) = -f(x)$ і збережемо його в поточній папці під ім'ям ex13.m:

```
function z = ex13(x)
z=-(exp(-x(1)\ 2-x(2)\ 2)+sin(x(1)+x(2)));
```

Викличемо вікно тулбокса за допомогою gatool.

У полі fitness function введемо ім'я цільової функції @ex13.

Встановимо значення параметрів ГА: кількість змінних = 2, кількість особин в популяції = 10, кількість поколінь = 100 (у вікні критерію зупинки алгоритму), початковий відрізок =[-1; 3] (рисунок 10.). Для побудови графіків

в розділі plots встановимо прапорці для best fitness, best individual, distance. Клацнемо по кнопці start.

В результаті завершення процесу у вікні final point з'явиться значення змінної x, що відповідає мінімуму функції, а у вікні status and result можна побачити знайдене мінімальне значення цільової функції $z(x)$.

Для цього завдання результати вийшли наступні: максимум функції досягається в точці x = -0.038, y = -0.032 і f(-0.038; -0.032)

Поверхня функції представлена на рис. 13.6.

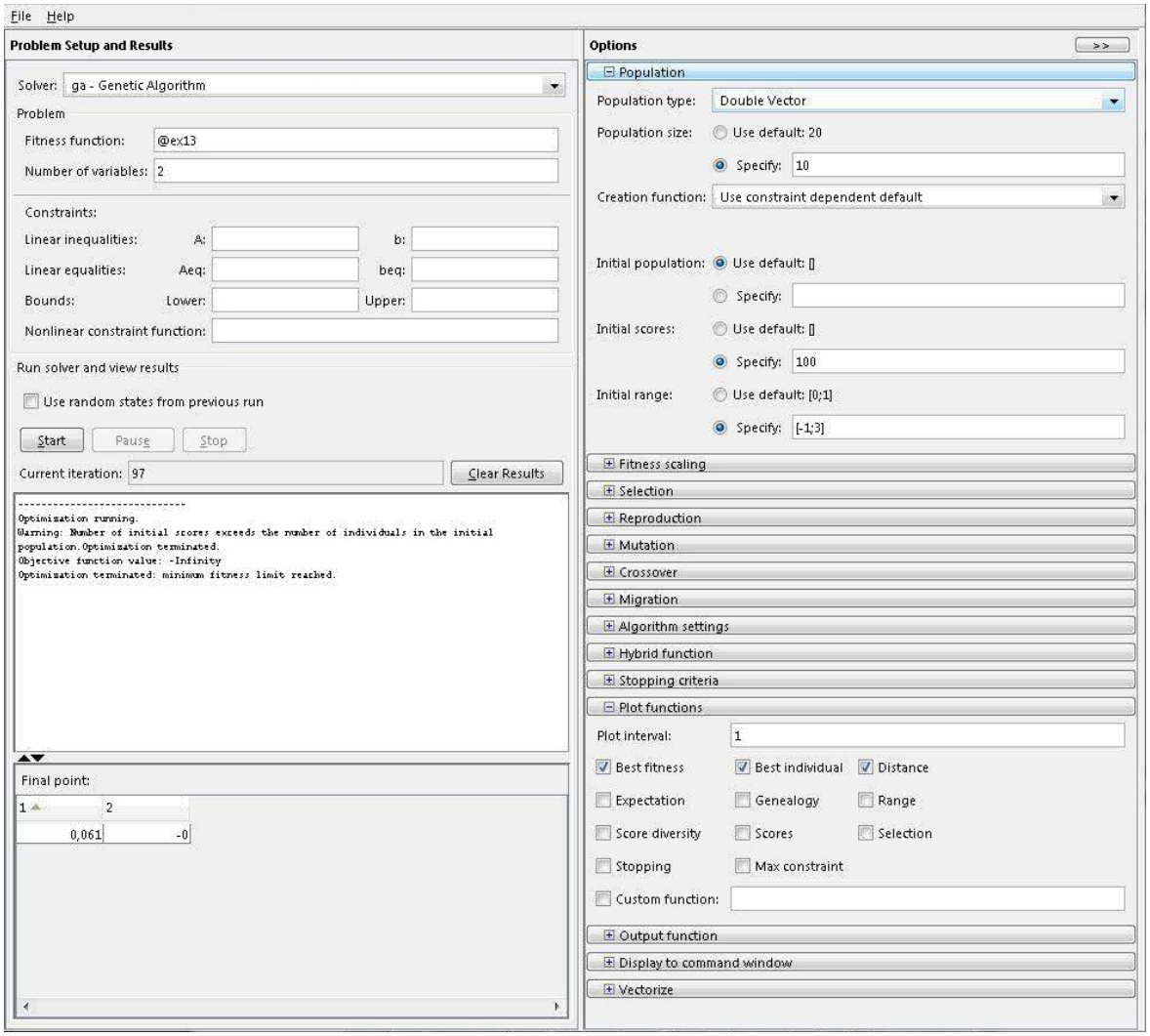

Рисунок 13.5 – Графічна оболонка gatool

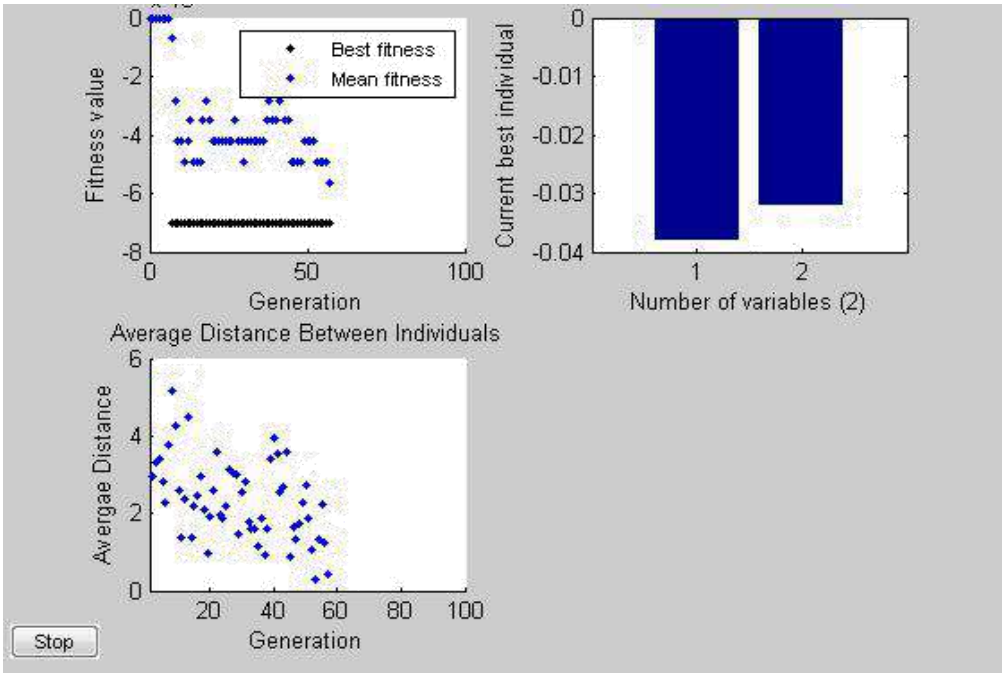

Рисунок 13.6 – Графічний аналіз рішення

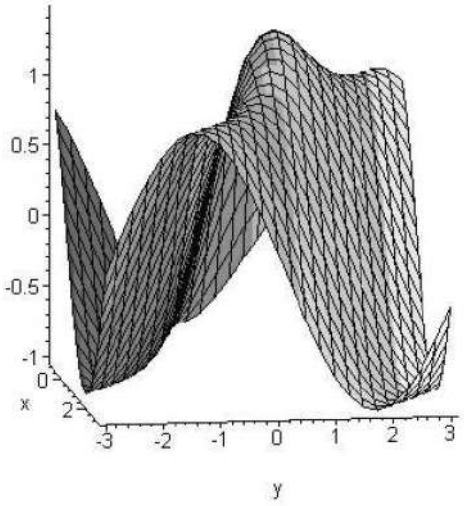

Рисунок 13.7 – Поверхня функції

ЛАБОРАТОРНА РОБОТА №14

Знаходження оптимального шляху за критерієм відстань між обласними центрами України за допомогою генетичних алгоритмів

Мета роботи: Розробити програму що за допомогою генетичного алгоритму знайде найкоротшу відстань між містами – обласними центрами України

Короткі теоретичні відомості

Задача комівояжера (комівояжер – бродячий торговець; англ. *Travelling Salesman Problem*, TSP) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста по одному разу [30]. В умовах задачі вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз. В такому випадку розв'язок знаходиться серед гамільтонових циклів.

Існує велика кількість різновидів узагальненої постановки задачі комівояжера. Такі як геометрична задача комівояжера (коли матриця відстаней відображає відстані між точками на площині), трикутна задача комівояжера (коли на матриці вартостей виконується нерівність трикутника), симетрична та асиметрична задачі комівояжера.

Прості методи розв'язання задачі комівояжера: повний лексичний перебір, жадібні алгоритми (метод найближчого сусіда), метод включення найближчого міста, метод найдешевшого включення, метод мінімального кістяка дерева. На практиці застосовують різні модифікації ефективніших методів: метод гілок і меж та метод генетичних алгоритмів, також алгоритм мурашиної колонії.

Всі ефективні (такі, що скорочують повний перебір) методи розв'язання задачі комівояжера – евристичні. У більшості евристичних методів

знаходиться не найефективніший маршрут, а наближений розв'язок. Користуються популярністю так звані any-time алгоритми, тобто алгоритми, що поступово покращують деякий поточний наближений розв'язок.

Коли задачу комівояжера було досліджено вперше невідомо. Однак, відома видана в 1832 році книга з назвою «Комівояжер – як він має поводитись і що має робити для того, аби доставляти товар та мати успіх в своїх справах – поради старого Кур'єра», в якій описано проблему, але математичний апарат для її розв'язання не застосовується. Натомість, в ній запропоновано приклади маршрутів для деяких регіонів Німеччини та Швейцарії.

Раннім варіантом задачі може розглядатись *Icosian Game* Вільяма Гамільтона 19 століття, яка полягала в тому, щоб знайти маршрути на графі з 20 вузлами. Перші згадки як математичної задачі на оптимізацію належать Карлу Менґеру, який сформулював її в математичному колоквіумі в 1930 році. Невдовзі з'явилась відома зараз назва *задача мандруючого продавця*, яку запропонував Гаслер Вітні з Принстонського Університету.

Разом із простотою постановки задачі та порівняною простотою знаходження *хороших* розв'язків задача комівояжера відрізняється тим, що пошук оптимального шляху є досить складним завданням. Зважаючи на ці властивості, починаючи з другої половини 20-го століття, дослідження задачі комівояжера, має не так практичний сенс, як теоретичний у ролі моделі для розробки нових алгоритмів оптимізації.

На прикладі задачі комівояжера було розроблено багато сучасних поширених методів дискретної оптимізації. А саме метод діленням площиною, гілок та границь та різноманітні варіанти евристичних алгоритмів.

В 1950-ті та 1960-ті роки задача комівояжера привернула увагу науковців в США та Європі. Важливий внесок в дослідження задачі належить Джорджу Данцігу, Делберту Рею Фалкерсону та Селмеру Джонсону, котрі в 1954 році в інституті RAND Corportation сформулювали задачу у вигляді

задачі дискретної оптимізації та розробили метод відсікаючої площини для її розв'язання [30]. Використовуючи новий метод вони обчислили шлях для окремого набору вузлів (екземпляру проблеми) з 49 міст та довели, що не існує коротшого шляху. В 1960-ті та 1970-ті роки численні групи дослідників вивчали задачу з точки зору математики та її застосування, наприклад, в інформатиці, економіці, хімії та біології.

Більших успіхів вдалося досягнути наприкінці 1970-х та 1980-х років, коли Мартін Ґрьотчел, Манфред Падберґ та Ґіованні Рінальді та інші, із застосуванням нових методів відсікаючої площини, гілок та границь обчислили розв'язок для окремого екземпляру задачі з 2393 містами [30].

В 1990-ті роки Девід Аплгейт, Роберт Біксбі, Вашек Шватал та Вільям Кук встановили рекорди з програмою Конкорд. Ґерхард Райнельт створив TSPLIB — набір стандартизованих екземплярів задачі комівояжера різного ступеня складності для порівняння результатів роботи різних груп дослідників. В березні 2005 року задачу з 33810 вузлами було розв'язано програмою Конкорд: було обчислено шлях завдовжки 66048945 та доведено відсутність коротших шляхів. В квітні 2006 було знайдено розв'язок для екземпляру з 85900 вузлами. Використовуючи методи декомпозиції можливо обчислити розв'язки для екземплярів задачі з мільйонами вузлів довжина яких менш ніж на 1 % більша за оптимальний.

Формальне визначення

Представлення у вигляді графу

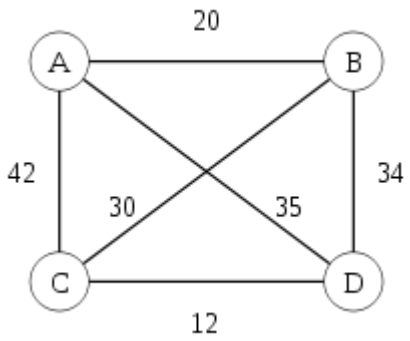

Рисунок 14.1 – Симетрична задача для чотирьох міст

Для можливості застосування математичного апарату для розв'язання проблеми, її слід представити у вигляді математичної моделі. Проблему комівояжера можна представити у вигляді моделі на графі, тобто, використовуючи вершини та ребра між ними. Таким чином, вершини графу (на рисунку 14.1: від A до D) відповідають містам, а ребра між вершинами та сполучення між цими містами. У відповідність кожному ребру можна зіставити вагу, яку можна розуміти як, наприклад, відстань між містами, час або вартість подорожі. *Маршрутом* (також гамільтоновим маршрутом) називається маршрут на цьому графі до якого входить по одному разу кожна вершина графу. Задача полягає у відшуканні найкоротшого маршруту.

З метою спрощення задачі та гарантії існування маршруту, зазвичай вважається, що модельний граф задачі є повним, тобто, що між довільною парою вершин існує ребро. Це можна досягти тим, що в тих випадках, коли між окремими містами не існує сполучення, вводити ребра з максимальною вагою (довжиною, вартістю тощо). Через велику довжину таке ребро ніколи не потрапить до оптимального маршруту, якщо він існує.

У залежності від того, що зіставляється вазі ребер, розрізняють різні варіанти задачі, найважливішими з яких є *симметрична* та *метрична* задачі.

Асиметрична та симетрична задачі

У загальному випадку, *асиметрична задача комівояжера* відрізняється тим, що ребра між вершинами можуть мати різну вагу в залежності від напряму, тобто, задача моделюється орієнтованим графом [30]. Таким чином, окрім ваги ребер графу, слід також зважати і на те, в якому напрямку знаходяться ребра.

У випадку *симетричної задачі* всі пари ребер між одними й тими самими вершинами мають однакову вагу, тобто, для ребра ваги однакові [30]. Як наслідок, всі маршрути мають однакову довжину в обидва напрямки. В симетричному випадку кількість можливих маршрутів вдвічі менша за

асиметричний випадок. Симетрична задача моделюється неорієнтованим графом (як показано на малюнку).

Насправді, задача комівояжера у випадку реальних міст може бути як симетричною, так і асиметричною в залежності від тривалості або довжини маршрутів в залежності від напряму руху.

Завдання на лабораторну роботу

Написати програмну на будь-якій зручній для Вас мові програмування, що знаходить оптимальний маршрут, за відстанню, між мінімум 10 обласними центрами України за допомогою генетичного алгоритму.

ЛАБОРАТОРНА РОБОТА №15

Параметричний синтез нейромереж

Мета роботи: Написати програму, що реалізує оптимізацію параметрів нейронної мережі, а саме її вагових коефіцієнтів, за допомогою генетичного алгоритму.

Короткі теоретичні відомості

У випадку, якщо структура нейромережної моделі визначена, тоді після відбору інформативних ознак виконується параметричний синтез нейромоделі [25].

При настроюванні значень вагових коефіцієнтів нейронних мереж заданої архітектури, як правило, застосовуються градієнтні методи. Градієнтні методи пов’язані з необхідністю обчислення значень цільової функції й не можуть бути застосовані для пошуку оптимальних значень синаптичних ваг у нейронних мережах, що містять нейрони з недиференційованими функціями активації. Також такі методи схильні до влучення в локальні оптимуми та не дозволяють знайти глобальний оптимум полімодальних функцій.

Методи еволюційної оптимізації є методами глобального пошуку й не використовують значення похідних цільової функції в процесі пошуку на відміну від градієнтних методів. При параметричному синтезі нейронних мереж доцільним є застосування еволюційних методів для пошуку оптимальних значень вагових коефіцієнтів нейронних мереж у випадках, коли градієнтні методи не можуть бути застосовані по різних причинах.

Еволюційна оптимізація може виконуватися довше в порівнянні із градієнтними методами. Еволюційна оптимізація в загальному випадку є значно менш чутливою до початкових параметрів навчання. Методи еволюційного пошуку завжди намагаються знайти глобальний оптимум. Градієнтні методи, як правило, знаходять локальний оптимум, розташований в околі початкової точки пошуку.

Для застосування еволюційного пошуку до параметричного синтезу нейронних мереж необхідно: визначити цільову функцію й вибрати спосіб подання значень ваг у хромосомі.

При виборі фітнес-функції для параметричного синтезу нейронних мереж враховують два фактори:

– помилка між реальним і модельним виходом мережі;

– складність нейронної мережі.

Як правило, як цільова функція при параметричному синтезі нейронних мереж використовується функція помилки, що обчислюється як середня різниця між поточним та заданим виходом мережі.

Хромосома при параметричному синтезі складається з K генів, що містять значення ваг і зсувів всіх нейронів мережі. При цьому для подання значень вагових коефіцієнтів у хромосомах застосовується дійсне кодування.

Розмір хромосоми визначається за формулою:

$$K = N_1\left(L+1\right) + \sum_{\mu=1}^{M} N_\mu \left(N_{\mu-1}+1\right),$$

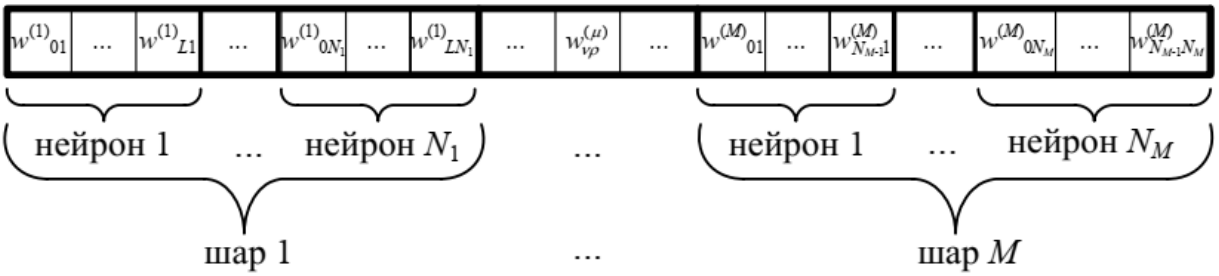

Рисунок 15.1 – Схематичне подання хромосоми при параметричному синтезі нейромоделей

де N_μ – кількість нейронів на μ-ому шарі; L – кількість ознак у навчальній вибірці; M – кількість шарів нейромережі.

Еволюційна оптимізація синаптичних ваг нейромереж може бути виконана в такій послідовності [25].

Крок 1. Виконати ініціалізацію початкової популяції хромосомами, що містять інформацію про значення вагових коефіцієнтів мережі заданої структури.

Крок 2. Оцінити хромосоми поточної популяції.

Крок 2.1. Декодувати кожну хромосому популяції в набір вагових коефіцієнтів нейронної мережі.

Крок 2.2. Побудувати нейромережі, що відповідають оцінюваним хромосомам.

Крок 2.3. Обчислити значення фітнес-функції оцінюваних хромосом, що враховує помилку й складність мережі.

Крок 3. Перевірити критерії закінчення пошуку (досягнення прийнятного значення помилки синтезованої нейромережної моделі, перевищення максимально припустимої кількості ітерацій, перевищення припустимого часу функціонування методу). У випадку, якщо критерії зупинення задовільнено, виконати перехід до кроку 7.

Крок 4. Виходячи зі значення фітнес-функції, вибрати особини для генерації нових рішень.

Крок 5. Застосувати оператори схрещування й мутації для хромосом, відібраних на попередньому кроці.

Крок 6. Сформувати нове покоління з елітних хромосом і хромосомнащадків, отриманих шляхом застосування схрещування й мутації. Перейти до виконання кроку 2.

Крок 7. Зупинення.

За даним алгоритмом навчання нейромоделей, засноване на еволюційному підході, не має потреби в обчисленні градієнта цільової функції. Дозволяє знайти значення глобальних оптимумів синаптичних ваг для багатовимірних, полімодальних і недиференційованих цільових функцій.

Ще однією перевагою еволюційного пошуку при параметричному синтезі нейромоделей є можливість застосування одного методу синтезу до

побудови різних моделей нейронних мереж (прямого поширення, рекурентних і ін.).

При використанні класичних методів еволюційної оптимізації хромосоми початкової популяції формуються випадковим чином, тобто початкова популяція хромосом являє собою набір нейронних мереж зі значеннями вагових коефіцієнтів і зсувів, згенерованими випадково. Випадкове створення хромосом початкової популяції при параметричному синтезі нейронних мереж приводить до зменшення ефективності процесу еволюційної оптимізації при пошуку оптимальних значень матриці ваг і збільшенню часу, необхідного для пошуку.

Завдання на лабораторну роботу

Написати програму на будь-якій зручній для Вас мові програмування, що навчає нейронну мережу прямого поширення моделювати функцію двох змінних за Вашим варіантом з додатку Б, за допомогою еволюційної оптимізації синаптичних ваг нейронних мереж, за алгоритмом представленим в теоретичних відомостях до даної лабораторної.

ПЕРЕЛІК ДЖЕРЕЛ

1. Simon Haykin. Neural Networks and Learning Machines 3rd Edition. – Pearson Education India, 2009.

2. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми – К.:«Корнійчук», . 2008. – 446 с.

3. Інформаційно-керуючі системи. Локальні інформаційно-керуючі системи. Лабораторний практикум [Електронний ресурс] : навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології» / П. І. Кравець, В. М. Шимкович, Ю. М. Бердник ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,9 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 142 с. – Назва з екрана. – Доступ: https://ela.kpi.ua/handle/123456789/47956

4. Y.S. Hryhorenko, P.I. Kravets, A.O. Novatskyi, V.M. Shymkovysh, L.L. Shymkovysh, A. Yu. Doroshenko. A Convolutional Neural Network Model and Software Tool for Classifying the Presence of a Medical Mask on a Human Face. PROBLEMS IN PROGRAMMING. Vol. 2 (2023). pp. 59-66

5. S.O. Bezpalko, P.I. Kravets, A.O. Novatskyi, V.M. Shymkovysh, L.L. Shymkovysh, A. Yu. Doroshenko. Software system for laser targeting dropped ammunition. PROBLEMS IN PROGRAMMING. Vol. 2 (2023). pp. 10-23

6. Peter Kravets, Anatolii Novatskyi, Volodymyr Shymkovysh, Antonina Rudakova, Yurii Lebedenko, Hanna Rudakova. Neural Network Model for Laboratory Stand Control System Controller with Parallel Mechanisms. Hu, Z., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies, vol 181. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_5

7. Y. Bezliudnyi, P. Kravets, A. Novatsky, V. Shymkovych, L. Shymkovych. Pro-russian propaganda recognition and analytics system based on

text classification model and statistical data processing methods. Адаптивні системи автоматичного управління, 1(42), с. 15-31. https://doi.org/10.20535/1560-8956.42.2023.278923

8. Дорошенко А. Ю., Шимкович В.М., Мамедов Т. А., Яценко О. А. Автоматизоване проєктування штучного нейрона для програмованих логічних інтегральних схем на основі алгебро-алгоритмічного підходу. International Scientific Technical Journal "Problems of Control and Informatics", 67(5), с. 61–72. https://doi.org/10.34229/2786-6505-2022-5-6

9. Bezpalko S. O., Shymkovysh V.M, Doroshenko A. Yu. A model and software for the inertial measurement unit. PROBLEMS IN PROGRAMMING. Vol. 2 (2022). pp. 3-12

10. Volodymyr Shymkovysh, Sergii Telenyk, Petro Kravets. Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications. Volume 33, Issue 15, pp. 9467 – 9479 https://doi.org/10.1007/s00521-021-05706-3

11. Bezliudnyi, Y. S., Shymkovysh V.M, Doroshenko, A. Y. Convolutional neural network model and software for classification of typical pests. PROBLEMS IN PROGRAMMING. Vol. 4 (2021). pp. 95-102

12. Doroshenko A., Shymkovysh V., Yatsenko O., Mamedov T. Automated Software Design for FPGAs on an Example of Developing a Genetic Algorithm. *CEUR Workshop Proceedings. 2021*. Volume 3013, pp. 74 – 8

13. Peter Kravets, Volodymyr Shymkovysh. Hardware Implementation Neural Network Controller on FPGA for Stability Ball on the Platform. Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing. 2020. Vol. 938. Springer, Cham, Switzerland. pp. 247-256 https://doi.org/10.1007/978-3-030-16621-2_23

14. Volodymyr Samotyy, Sergii Telenyk, Petro Kravets, Volodymyr Shymkovysh, Taras Posvistak. A real time control system for balancing a ball on a

platform with FPGA parallel implementation. Technical Transactions. Poland. 2018. Vol. 5. https://doi.org/10.4467/2353737XCT.18.077.8559

14. Zimmerman H. J. Fuzzy programming and linear programming with several objective function // Fuzzy sets and Systems. 1977. 1. pp. 15 – 55

15. Dubois D., Prade H. Fuzzy sets and systems: theory and applications. – N.Y.: Acad. Press, 1980.

16. Verdegay J. L. Fuzzy mathematical programming. In: Fuzzy information and decision process / Ed. by Madan M. Gupta and Elie Sanchez. North-Holland

19. Sigeru O, Khalid MB, Rubiyah Y (1996) Neuro-control and its applications. Springer-Verlag, London. https://doi.org/10.1007/978-1-4471-3058-1

20. Dreyfus G (2005) Neural networks: methodology and applications. Springer-Verlag, Berlin. https://doi.org/10.1007/3-540-28847-3

21. Edelen AL, Biedron SG, Chase BE, Edstrom D, Milton SV, Stabile P (2016) Neural networks for modeling and control of particle accelerators. IEEE Trans Nucl Sci 63:878–897. https://doi.org/10.1109/TNS.2016.2543203

22. Baptista D, Abreu S, Freitas F, Vasconcelos R, Morgado-Dias F (2013) A survey of software and hardware use in artificial neural networks. Neural Comput Appl 23:591–599. https://doi.org/10.1007/s00521-013-1406-y

23. Nelles O (2013) Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Springer, Berlin. https://doi.org/10.1007/978-3-662-04323-3

24. Rutkowska, D., M. Piliński, and L. Rutkowski. "Neural network, genetic algorithm, and fuzzy systems (Polish: Sieci neuronowe, algorytmy genetyczne i systemy rozmyte." PWN. Warszawa, 1997

25. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія / Під заг. ред. С.О. Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375 с.

26. Zadeh L. (1965) Fuzzy Sets. Information and Control. Vol.8. pp. 338 - 353.

27. Mamdani E. H. (1974) Application of fuzzy algorithms for the control of a simple dynamic plant. In Proc IEEE, pp. 121–159.

28. Kickert, W. J. M., & Nauta Lemke, van, H. R. (1976). Application of a fuzzy controller in a warm water plant. Automatica, 12(4), 301-308.

29. Шимкович В. М., Кравець П. І., Лукіна Т. Й., Ткач І. І. Розробка та дослідження технології оцінювання показників нейромережевих моделей MIMO-об'єктів керування // Вісник НТУУ «КПІ». Інформатика, керування та обчислювальна техніка: Зб. наук. пр. 2012. №57. С. 144–149.

30. Задача комівояжера. URL: https://uk.wikipedia.org/wiki?curid=226676 (дата звернення 10.10.2023)

31.

ДОДАТОК А – Таблиця варіантів передавальної функції згідно з номером студента за списком навчальної групи

Варіант №	Завдання для ІА-Х1
1.	$W(s)=\frac{16}{s^2+s+1}$
2.	$W(s)=\frac{13}{0.7s^2+s+1}$
3.	$W(s)=\frac{18}{0.6s^2+0.5s+1}$
4.	$W(s)=\frac{15}{0.8s^2+0.8s+1}$
5.	$W(s)=\frac{0.1s+15}{0.5s^2+0.3s+1}$
6.	$W(s)=\frac{0.1s+11}{0.5s^2+0.31s+1}$
7.	$W(s)=\frac{0.2s+8}{0.1s^2+0.25s+1}$
8.	$W(s)=\frac{8}{0.5s^2+0.3s+1}$
9.	$W(s)=\frac{12}{0.7s^2+0.45s+1}$
10.	$W(s)=\frac{0.17s+7}{0.4s^2+0.3s+1}$
11.	$W(s)=\frac{0.2s+5}{0.46s^2+0.6s+1}$
12.	$W(s)=\frac{0.1s+12}{0.8s^2+0.4s+1}$
13.	$W(s)=\frac{0.1s+13}{0.8s^2+0.4s+1}$
14.	$W(s)=\frac{0.2s+19}{s^2+0.2s+1}$
15.	$W(s)=\frac{0.3s+11}{0.5s^2+0.4s+1}$
16.	$W(s)=\frac{0.22s+15}{0.8s^2+0.6s+1}$
17.	$W(s)=\frac{12}{0.28s^2+0.4s+1}$
18.	$W(s)=\frac{10}{0.36s^2+0.8s+1}$
19.	$W(s)=\frac{0.45s+10}{2s^2+0.5s+1}$
20.	$W(s)=\frac{12}{4s^2+0.6s+1}$
21.	$W(s)=\frac{0.1s+12}{0.58s^2+0.4s+1}$

22.	$W(s) = \frac{0.4s + 18}{s^2 + 0.62s + 1}$
23.	$W(s) = \frac{18}{s^2 + 0.57s + 1}$
24.	$W(s) = \frac{12}{3s^2 + 0.8s + 1}$
25.	$W(s) = \frac{15}{0.3s^2 + 0.46s + 1}$
26.	$W(s) = \frac{0.43s + 15}{0.4s^2 + 0.8s + 1}$
27.	$W(s) = \frac{s + 26}{0.36s^2 + 0.6s + 1}$
28.	$W(s) = \frac{0.42s + 5}{0.6s^2 + 0.8s + 1}$
29.	$W(s) = \frac{0.96s + 11}{0.8s^2 + 0.4s + 1}$
30.	$W(s) = \frac{12}{6s^2 + 0.8s + 1}$
31.	$W(s) = \frac{25}{0.2s^2 + 0.6s + 1}$
32.	$W(s) = \frac{0.3s + 25}{0.4s^2 + 0.8s + 1}$
33.	$W(s) = \frac{s + 26}{0.6s^2 + 0.6s + 1}$
34.	$W(s) = \frac{0.2s + 4.5}{0.96s^2 + 0.8s + 1}$
35.	$W(s) = \frac{0.16s + 16}{0.8s^2 + 0.4s + 1}$

Варіант №	**Завдання для ІА-Х2**
1.	$W(s) = \frac{16}{s^2 + 2s + 1}$
2.	$W(s) = \frac{13}{0.7s^2 + s + 1}$
3.	$W(s) = \frac{18}{0.6s^2 + 0.25s + 1}$
4.	$W(s) = \frac{15}{0.8s^2 + 0.28s + 1}$
5.	$W(s) = \frac{0.1s + 15}{0.5s^2 + 0.23s + 1}$
6.	$W(s) = \frac{0.1s + 11}{0.25s^2 + 0.1s + 1}$
7.	$W(s) = \frac{0.2s + 8}{0.21s^2 + 0.5s + 1}$
8.	$W(s) = \frac{8}{0.25s^2 + 0.3s + 1}$

9.	$W(s) = \frac{12}{0.27s^2 + 0.4s + 1}$
10.	$W(s) = \frac{0.17s + 7}{0.24s^2 + 0.3s + 1}$
11.	$W(s) = \frac{0.22s + 5}{0.46s^2 + 0.6s + 1}$
12.	$W(s) = \frac{0.21s + 12}{0.8s^2 + 0.24s + 1}$
13.	$W(s) = \frac{0.1s + 13}{0.28s^2 + 0.4s + 1}$
14.	$W(s) = \frac{0.2s + 19}{2s^2 + 0.2s + 1}$
15.	$W(s) = \frac{0.23s + 11}{0.25s^2 + 0.4s + 1}$
16.	$W(s) = \frac{0.22s + 15}{0.28s^2 + 0.6s + 1}$
17.	$W(s) = \frac{72}{0.28s^2 + 0.4s + 1}$
18.	$W(s) = \frac{20}{0.36s^2 + 0.8s + 1}$
19.	$W(s) = \frac{0.45s + 12}{2s^2 + 0.5s + 1}$
20.	$W(s) = \frac{32}{4s^2 + 0.6s + 1}$
21.	$W(s) = \frac{0.1s + 12}{0.9s^2 + 0.34s + 1}$
22.	$W(s) = \frac{0.9s + 18}{s^2 + 0.62s + 1}$
23.	$W(s) = \frac{18}{7s^2 + 0.57s + 1}$
24.	$W(s) = \frac{12}{3s^2 + 0.28s + 1}$
25.	$W(s) = \frac{12}{0.3s^2 + 0.46s + 1}$
26.	$W(s) = \frac{0.43s + 15}{0.4s^2 + 0.8s + 1}$
27.	$W(s) = \frac{s + 7}{0.36s^2 + 0.6s + 1}$
28.	$W(s) = \frac{0.42s + 5}{0.6s^2 + 0.28s + 1}$
29.	$W(s) = \frac{0.96s + 11}{0.8s^2 + 0.42s + 1}$
30.	$W(s) = \frac{12}{6s^2 + 0.28s + 1}$
31.	$W(s) = \frac{25}{0.2s^2 + 0.6s + 1}$
32.	$W(s) = \frac{0.23s + 25}{0.24s^2 + 0.8s + 1}$

33.	$W(s) = \frac{s + 76}{0.6s^2 + 0.26s + 1}$
34.	$W(s) = \frac{0.2s + 64.5}{0.96s^2 + 0.8s + 1}$
35.	$W(s) = \frac{0.16s + 16}{0.28s^2 + 0.2s + 1}$

Варіант №	**Завдання для ІА-Х3**
1.	$W(s) = \frac{16}{s^2 + 3s + 1}$
2.	$W(s) = \frac{13}{0.7s^2 + 3s + 1}$
3.	$W(s) = \frac{18}{0.6s^2 + 0.35s + 1}$
4.	$W(s) = \frac{15}{0.8s^2 + 0.38s + 1}$
5.	$W(s) = \frac{0.1s + 15}{0.5s^2 + 0.33s + 1}$
6.	$W(s) = \frac{0.1s + 11}{0.35s^2 + 0.1s + 1}$
7.	$W(s) = \frac{0.3s + 8}{0.31s^2 + 0.5s + 1}$
8.	$W(s) = \frac{8}{0.8s^2 + 0.33s + 1}$
9.	$W(s) = \frac{12}{0.37s^2 + 0.4s + 1}$
10.	$W(s) = \frac{0.17s + 7}{0.34s^2 + 0.3s + 1}$
11.	$W(s) = \frac{0.32s + 5}{0.46s^2 + 0.3s + 1}$
12.	$W(s) = \frac{0.21s + 13}{0.8s^2 + 0.34s + 1}$
13.	$W(s) = \frac{0.1s + 13}{0.3s^2 + 0.34s + 1}$
14.	$W(s) = \frac{0.3s + 19}{3s^2 + 0.2s + 1}$
15.	$W(s) = \frac{0.23s + 13}{0.35s^2 + 0.4s + 1}$
16.	$W(s) = \frac{0.22s + 13}{0.38s^2 + 0.6s + 1}$
17.	$W(s) = \frac{32}{0.38s^2 + 0.4s + 1}$
18.	$W(s) = \frac{30}{0.33s^2 + 0.8s + 1}$
19.	$W(s) = \frac{0.45s + 13}{3s^2 + 0.5s + 1}$
20.	$W(s) = \frac{32}{3s^2 + 0.36s + 1}$

21.	$W(s) = \frac{0.31s + 13}{0.9s^2 + 0.34s + 1}$
22.	$W(s) = \frac{0.39s + 13}{s^2 + 0.63s + 1}$
23.	$W(s) = \frac{18}{7.3s^2 + 0.37s + 1}$
24.	$W(s) = \frac{12}{3s^2 + 0.38s + 1}$
25.	$W(s) = \frac{12}{0.3s^2 + 0.36s + 1}$
26.	$W(s) = \frac{0.33s + 13}{0.4s^2 + 0.3s + 1}$
27.	$W(s) = \frac{s + 8}{0.33s^2 + 0.6s + 1}$
28.	$W(s) = \frac{0.42s + 3}{0.6s^2 + 0.38s + 1}$
29.	$W(s) = \frac{0.96s + 31}{0.3s^2 + 0.42s + 1}$
30.	$W(s) = \frac{12}{3s^2 + 0.28s + 1}$
31.	$W(s) = \frac{35}{0.3s^2 + 0.6s + 1}$
32.	$W(s) = \frac{0.33s + 25}{0.34s^2 + 0.8s + 1}$
33.	$W(s) = \frac{s + 36}{0.6s^2 + 0.36s + 1}$
34.	$W(s) = \frac{0.2s + 34.5}{0.96s^2 + 0.8s + 1}$
35.	$W(s) = \frac{0.36s + 16}{0.38s^2 + 0.2s + 1}$

Варіант №	**Завдання для ІА-Х4**
1.	$W(s) = \frac{14}{s^2 + 4s + 1}$
2.	$W(s) = \frac{14}{0.4s^2 + 4s + 1}$
3.	$W(s) = \frac{18}{0.4s^2 + 0.25s + 1}$
4.	$W(s) = \frac{15}{0.8s^2 + 0.48s + 1}$
5.	$W(s) = \frac{0.1s + 15}{0.5s^2 + 0.43s + 1}$
6.	$W(s) = \frac{0.1s + 11}{0.45s^2 + 0.4s + 1}$
7.	$W(s) = \frac{0.4s + 8}{0.41s^2 + 0.5s + 1}$

8.	$W(s) = \frac{8}{0.45s^2 + 0.3s + 1}$
9.	$W(s) = \frac{14}{0.47s^2 + 0.4s + 1}$
10.	$W(s) = \frac{0.17s + 4}{0.44s^2 + 0.3s + 1}$
11.	$W(s) = \frac{0.42s + 4}{0.46s^2 + 0.6s + 1}$
12.	$W(s) = \frac{0.21s + 14}{0.8s^2 + 0.44s + 1}$
13.	$W(s) = \frac{0.1s + 14}{0.48s^2 + 0.4s + 1}$
14.	$W(s) = \frac{0.4s + 19}{4s^2 + 0.2s + 1}$
15.	$W(s) = \frac{0.43s + 14}{0.25s^2 + 0.4s + 1}$
16.	$W(s) = \frac{0.22s + 15}{0.48s^2 + 0.4s + 1}$
17.	$W(s) = \frac{42}{0.78s^2 + 0.4s + 1}$
18.	$W(s) = \frac{24}{0.36s^2 + 0.4s + 1}$
19.	$W(s) = \frac{0.45s + 14}{4s^2 + 0.5s + 1}$
20.	$W(s) = \frac{34}{4s^2 + 0.4s + 1}$
21.	$W(s) = \frac{0.4s + 14}{0.9s^2 + 0.34s + 1}$
22.	$W(s) = \frac{0.9s + 14}{s^2 + 0.42s + 1}$
23.	$W(s) = \frac{18}{7s^2 + 0.57s + 1}$
24.	$W(s) = \frac{14}{3s^2 + 0.48s + 1}$
25.	$W(s) = \frac{14}{0.3s^2 + 0.46s + 1}$
26.	$W(s) = \frac{0.43s + 15}{0.4s^2 + 0.48s + 1}$
27.	$W(s) = \frac{s + 4}{0.46s^2 + 0.4s + 1}$
28.	$W(s) = \frac{0.42s + 5}{0.6s^2 + 0.48s + 1}$
29.	$W(s) = \frac{0.46s + 11}{0.4s^2 + 0.42s + 1}$
30.	$W(s) = \frac{12}{4s^2 + 0.48s + 1}$
31.	$W(s) = \frac{25}{0.2s^2 + 0.4s + 1}$

32.	$W(s) = \frac{0.43s + 25}{0.24s^2 + 0.4s + 1}$
33.	$W(s) = \frac{s + 46}{0.6s^2 + 0.46s + 1}$
34.	$W(s) = \frac{0.2s + 44.5}{0.96s^2 + 0.8s + 1}$
35.	$W(s) = \frac{0.16s + 16}{0.7s^2 + 0.42s + 1}$

ДОДАТОК Б – Таблиця варіантів завдань згідно з номером студента за списком навчальної групи

№	Завдання	ІА-Х1 Номер за списком	ІА-Х2 Номер за списком	ІА-Х3 Номер за списком
1.	$y=\sin(x)+\cos(x/2)$ $z=\sin\left(2\cdot\sqrt{x^2+y^2}\right)/\left(\sqrt{x^2+y^2}+0.001\right)$	1.	30.	29.
2.	$y=\sin(x)/x$ $z=(x-y)\cdot\sin(x+y)$	2.	29.	27.
3.	$y=x\cdot\sin(x)$ $z=x\cdot\cos(y)+\sin(x)$	3.	28.	25.
4.	$y=7\cdot\sin(5/2\cdot\cos(x))$ $z=(x-2)^2\cdot(1-y^2)$	4.	27.	23.
5.	$y=\sin\lvert x\rvert\cdot\cos(3x/2)$ $z=x\cdot\sin(x+y)$	5.	26.	21.
6.	$y=\sin\lvert x\rvert+\cos\lvert x\rvert$ $z=x\cdot\sin(y)$	6.	25.	19.
7.	$y=\sin(x^2)/x$ $z=\sin\lvert x\rvert\cdot\sin\lvert y\rvert$	7.	24.	17.
8.	$y=\cos(x^2)\cdot\sin(x)$ $z=\cos(x^2)\cdot\sin(x+y)$	8.	23.	15.
9.	$y=0.2\cdot\sin(3x)\cdot x^2$ $z=\sin\lvert x\rvert\cdot\sin(x+y)$	9.	22.	13.
10.	$y=x\cdot\cos(x)+\sin(x)$ $z=\sin(x)+\cos(y/2)$	10.	21.	11.
11.	$y=\sin((x-2)/2)\cdot x\cdot\sin(x-3)$ $z=\cos(y+x/2)$	11.	20.	9.
12.	$y=\cos(x)/x-\sin(x)/x^2$ $z=\sin(x/2)+y\cdot\sin(x)$	12.	19.	7.
13.	$y=x\cdot\sin(x)\cdot\cos(x)$ $z=\cos(\sin(y))\cdot\sin(x)$	13.	18.	5.
14.	$y=\cos(x/2)+\sin(x/3)$	14.	17.	3.

	$z = 0.5\sin(x+y)\cdot\cos(y)$			
15.	$y = \sin(x)\cdot\cos(x/2+\sin(x))$ $z = x\cdot\cos(x+y)$	15.	16.	1.
16.	$y = \cos(x/2)+\sin(x/4)\cdot\sin(\cos(x+1))$ $z = x\cdot\cos\lvert y\rvert$	16.	15.	2.
17.	$y = \cos(x^2)+\sin(x)$ $z = \sin(\cos(y))+\cos(x/2)$	17.	14.	4.
18.	$y = \cos(\sin(x))-\sin(\cos(x))$ $z = \cos(x)+\sin(y/2)$	18.	13.	6.
19.	$y = \cos(x/2)\cdot\lvert x\rvert\cdot\cos(x)$ $z = x\cdot\cos(x)+\sin(y)$	19.	12.	8.
20.	$y = \cos(x/2)+\sin(x/3)/\lvert x/3\rvert$ $z = \cos(y)+\sin(x)+\cos(x+y)$	20.	11.	10.
21.	$y = 3\sin(5/2\cdot\cos(x))\cdot\lvert x/2\rvert$ $z = (x+y)\cdot\sin(x+y)$	21.	10.	12.
22.	$y = \cos(\sin(x))\cdot\sin(x/2)$ $z = x\cdot\sin\lvert y\rvert$	22.	9.	14.
23.	$y = \sin(\lvert x\rvert)\cdot\cos(x/2)$ $z = y\cdot\sin(x)$	23.	8.	16.
24.	$y = x\cdot\cos(2x)+\sin(x/2)$ $z = \sin(y)+\cos(x/2)$	24.	7.	18.
25.	$y = \sin(x^2)/x\cdot\cos(x)$ $z = \cos\lvert y\rvert+\sin(x+y)$	25.	6.	20.
26.	$y = \sin(x)/2\cdot\lvert x\rvert\cdot\sin(x^2)$ $z = 0.5\cdot\cos(x+y)\cdot\cos(x)$	26.	5.	22.
27.	$y = \sin(x)+\cos(x)\cdot\sin(\cos(x))$ $z = \sin(5\cos(x)/2)+\cos(y)$	27.	4.	24.
28.	$y = \sin(x^2)/x\cdot\cos(x-2)$ $z = \sin(\cos(x))+\cos(y/2)\cdot\sin(x/2)\cdot\cos(x)$	28.	3.	26.
29.	$y = \sin(x+\cos(x))\cdot\cos(x/2+\sin(x))$ $z = \cos(x/2)+y\cdot\cos(x)$	29.	2.	28.
30.	$y = \sin(x^2/2)\cdot\lvert x-5\rvert\cdot\cos(x)$ $z = \sin(x/2)+\cos(y/2)+\sin(y)$	30.	1.	30.