МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Басюк Наталія Василівна

Алгоритми аналізу стану комп'ютерної системи на основі нечіткої логіки / Algorithms of the computer system analyzing based on fuzzy logic

спеціальність:123 – Комп'ютерна інженерія
освітньо-професійна програма – Комп'ютерна інженерія

Випускна кваліфікаційна робота

Виконав студент групи КІм-21
Н. В. Басюк
_____

Науковий керівник:
к.т.н., Л.О. Дубчак
_____

ТЕРНОПІЛЬ - 2019

РЕЗЮМЕ

Магістерська робота на тему “Алгоритми аналізу стану комп’ютерної системи на основі нечіткої логіки” зі спеціальності 123 «Комп’ютерна інженерія» написана обсягом 95 сторінок і містить 40 ілюстрацій, 2 таблиці, 3 додатки та 50 джерел за переліком посилань.

Метою роботи є аналіз роботи комп’ютерних систем, їх основних характеристик та розробка нечіткого алгоритму аналізу їх стану, а також моделювання нечіткого контролера, що і має за основу розроблений алгоритм та нечітку систему.

Методи досліджень. В основу наукових досліджень покладено методи та алгоритми нечіткої логіки, а саме – алгоритм Мамдані.

Результати дослідження: алгоритми аналізу стану комп’ютерної системи на основі нечіткої логіки, розроблена відповідна нечітка система та контролер для практичної реалізації.

Результати роботи можуть бути використані на будь-яких підприємствах, де використовується комп’ютер, для комерційних та некомерційних проектів та в навчальному процесі.

Орієнтовні напрямки розвитку досліджень: розроблення веб-додатку чи застосунку з зрозумілим інтерфейсом, що дозволить користуватись розробкою будь-якому користувачу; розширення спектру охоплених проблем, що дозволить застосовувати розробку до більшої кількості видів комп’ютерів.

КЛЮЧОВІ СЛОВА: МАМДАНІ, НЕЧІТКА ЛОГІКА, КОМП’ЮТЕР, АНАЛІЗ, КОНТРОЛЕР.

RESUME

The master's thesis on " Algorithms of the computer system analyzing based on fuzzy logic " in specialty 123 "Computer Engineering" is written in volume of 95 pages and contains 40 illustrations, 2 tables, 3 appendices and 50 sources in the list of links.

The purpose of the work is to analyze the operation of computer systems, their main characteristics and to develop a fuzzy algorithm for analyzing their state, as well as to simulate a fuzzy controller, which is based on the developed algorithm and fuzzy system.

Research Methods. The basis of scientific research is the methods and algorithms of fuzzy logic, namely, the Mamdani algorithm.

Results of the study: algorithms for analyzing the state of the computer system based on fuzzy logic, developed a fuzzy system and controller for practical implementation.

The results of the work can be used in any enterprise using a computer, for commercial and non-commercial projects and in the educational process.

Guidelines for research development: development of a web application or application with a user-friendly interface that will allow the development of any user; expanding the range of problems covered, which will allow development to be applied to more types of computers.

KEYWORDS: MAMDANI, FUZZY LOGIC, COMPUTER, ANALYSIS, CONTROLLER.

ЗМІСТ

ВСТУП

Актуальність розробок, що пов’язані з комп'ютерними системами зростає з кожним роком у сучасному світі. Вони використовуються, практично, в усіх сферах життя суспільства, стали незамінні для сучасних професій практично усіх галузей. Завдяки комп'ютерній системі можна вирішувати прикладні завдання в предметних галузях діяльності такі як технологічна підготовка, керування, облік, автоматизація процесів. Практичне застосування комп'ютерні системи знайшли при дистанційному навчанні. Раніше, дистанційне навчання означало заочне навчання. Зараз це засіб навчання, що використовує кейс–, ТБ– і мережеві технології навчання. Користувачам комп'ютера доступне програмне забезпечення, а саме: системні та прикладні програми (наприклад, компілятори, текстові редактори, системи управління базами даних тощо). Ці програми взаємодіють з операційною системою, яка, в свою чергу, управляє роботою комп'ютера [1].

Неможливо уявити сучасну людину без комп’ютера, або ж сучасне підприємство без застосування великих комп’ютерних систем та мереж.

Тому є актуальним захист комп’ютерної системи шляхом вчасної та регулярної діагностики, яка зможе попередити 90% усіх існуючих проблем, що можуть виникнути.

Нечітка система та відповідний нечіткий контролер дозволить підтримувати задану функціональність та стійкість будь–якої комп’ютерної системи [2].

Мета проекту полягає в аналізі роботи комп’ютерних систем, їх основних характеристик та розробці нечіткого алгоритму аналізу їх стану, а також моделюванні нечіткого контролера, що і має за основу розроблений алгоритм та нечітку систему.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

– проаналізувати всі можливі стани комп’ютерних систем;

– дослідити нечітку логіку та розробити відповідний алгоритм нечіткого виводу по базі знань;

– побудувати функції належності;

– сформувати базу правил;

– дослідити роботу реалізованої нечіткої системи;

– зробити висновки щодо правильності роботи розробленої системи;

– на основі нечіткої системи побудувати контролер;

– дослідити коректність його роботи та зробити висновки.

Об’єкт дослідження – комп’ютерні системи та їх основні характеристики.

Предмет дослідження – методи та засоби підвищення стійкості комп’ютерних систем.

Методи дослідження – методи та алгоритми нечіткої логіки, а саме – алгоритм Мамдані.

Наукова новизна розробленого алгоритму та нечіткого контролера полягає в тому, що:

– удосконалено та модифіковано вже відомі алгоритми нечіткого висновку і в результаті отримано новий, ефективний та адаптований під конкретне завдання алгоритм;

– вперше одержано широкоспеціалізований контролер, котрий вимагає мінімум коштів, може бути легко модифікований та водночас забезпечує стабільну роботу та точний результат.

Дана розробка може бути корисною для будь якого підприємства, організації чи навіть пересічного користувача, адже стосується роботи комп’ютерів, без яких важко уявити виконання більшості процесів. Найбільш ефективним буде використання саме на великих підприємствах, бо похибки в роботі обладнання принесуть там значні збитки. А використання даного алгоритму, що є водночас простим в реалізації, проте ефективним в роботі, дозволить уникнути втрат та попередити можливі проблеми.

Результати даної роботи були висвітлені на декількох конференціях та відповідно опубліковано у посібниках, їх перелік подано нижче:

– «Аналіз поточного стану комп'ютерної системи на основі нечіткої логіки» на I Науково-практичній конференції молодих вчених та студентів «Інтелектуальні комп'ютерні системи та мережі» (Тернопіль, ТНЕУ – 2019) [3].

– «Застосування нечіткого контролера для системи кондиціонування» на II Науково-практичній конференції молодих вчених та студентів «Інтелектуальні комп'ютерні системи та мережі» (Тернопіль, ТНЕУ – 2019) [4].

Впровадження результатів магістерської роботи планується на IT–підприємстві, яке слугувало базою для переддипломної практики.

В першому розділі магістерської роботи висвітлено тему функціональності комп'ютерних систем, їх характеристик та найпоширеніших типів. Також розглянуто поняття стану комп'ютерної системи та основні елементи нечіткої логіки.

В другому розділі магістерської роботи розглянуто загальний алгоритм нечіткої системи, що в подальшому буде аналізувати комп'ютерні системи, а також визначено можливі методи реалізації функцій роботи програми та розроблено її загальну структуру та нечіткі функції та базу правил, за якою вона буде працювати.

В третьому розділі магістерської роботи було описано створення відповідного нечіткого контролера, його функціональні елементи та принцип роботи. Також проаналізовано результати його роботи та надані відповідні підтверджувальні рисунки.

Випускна кваліфікаційна робота оформлена на матеріалах відповідних методичок [5], [6].

1 АНАЛІЗ КЛЮЧОВИХ ПОНЯТЬ НЕЧІТКОЇ ЛОГІКИ ТА КОМП'ЮТЕРНОЇ СИСТЕМИ

1.1 Поняття комп'ютерної системи

Комп'ютерна система – це сукупність різних компонентів, що використовуються для спільної обробки даних. Мета комп'ютерної системи – зробити процес вирішення завдання на комп'ютері найбільш простим. Функціонуюча комп'ютерна система об'єднує елементи програмного і апаратного забезпечення. Апаратні елементи – це механічні пристрої комп'ютера, які виконують всі фізичні функції. Програмні елементи – це додатки, написані під систему; саме вони виконують логічні і математичні операції і надають користувачеві можливість управління комп'ютером. Документація включає в себе керівництво і списки допустимих операцій, завдяки яким можна повноцінно використовувати програмні і апаратні складові комп'ютера [7].

Разом ці компоненти утворюють комп'ютерну систему: системне апаратне забезпечення плюс системні програми плюс документація до них дорівнює комп'ютерна система. Зазвичай, до складу комп'ютерної системи входить три базові апаратних складових: сам комп'ютер, який обробляє всі дані; термінальний пристрій, що використовується як друкарська машинка для двостороннього контакту між користувачем і системою; і медіа накопичувач – для зберігання програм і даних. Три цих пристрої – комп'ютер, термінал і медіа–сховище – необхідні складові будь–якої комп'ютерної системи.

Найпоширенішою класифікацією комп'ютерних систем є класифікація (або ж таксономія) за Флінном. Це загальна класифікація архітектур ЕОМ за ознаками наявності паралелізму в потоках команд і даних. Була запропонована Майклом Флінном в 1966 році і розширена в 1972 році. Все розмаїття архітектур ЕОМ в цій таксономії Флінна зводиться до чотирьох класів [8] :

– SISD (Single Instruction stream over a Single Data stream) – обчислювальна система з одиночним потоком команд і одиночним потоком даних;

– SIMD (Single Instruction, Multiple Data) – обчислювальна система з одиночним потоком команд і множинним потоком даних;

– MISD (Multiple Instruction Single Data) – обчислювальна система з множинним потоком команд і одиночним потоком даних;

– MIMD (Multiple Instruction Multiple Data) – обчислювальна система з множинним потоком команд і множинним потоком даних.

Архітектура SISD – це традиційний комп'ютер фон–Нейманівської архітектури з одним процесором, який виконує послідовно одну інструкцію за одною, працюючи з одним потоком даних. В даному класі не використовується паралелізм ні даних, ні інструкцій, тому SISD–машина не є паралельною. До цього класу також прийнято відносити конвеєрні, суперскалярні і VLIW–процесори. На рисунку 1.1 зображено тип архітектури SISD.

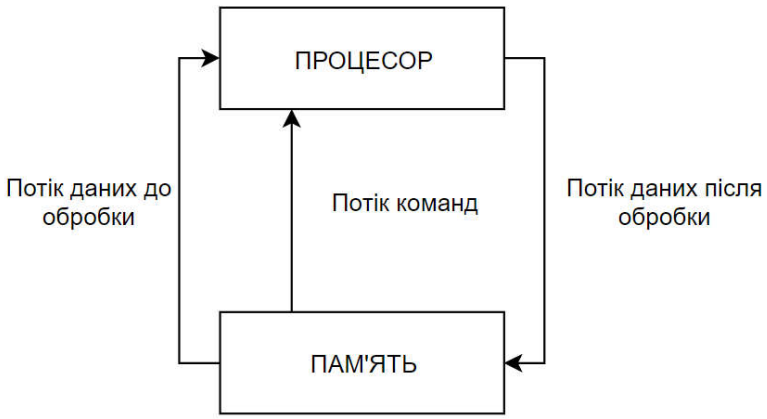

Рисунок 1.1 – Тип архітектури SISD

Типовими представниками SIMD є векторні процесори, звичайні сучасні процесори, коли працюють в режимі виконання команд векторних розширень, а також особливий підвид з великою кількістю процесорів – матричні

процесори [9]. У SIMD–машинах один процесор завантажує одну інструкцію, набір даних до них і виконує операцію, описану в цій інструкції, над усім набором даних одночасно. Тип архітектури SIMD зображений на рисунку 1.2.

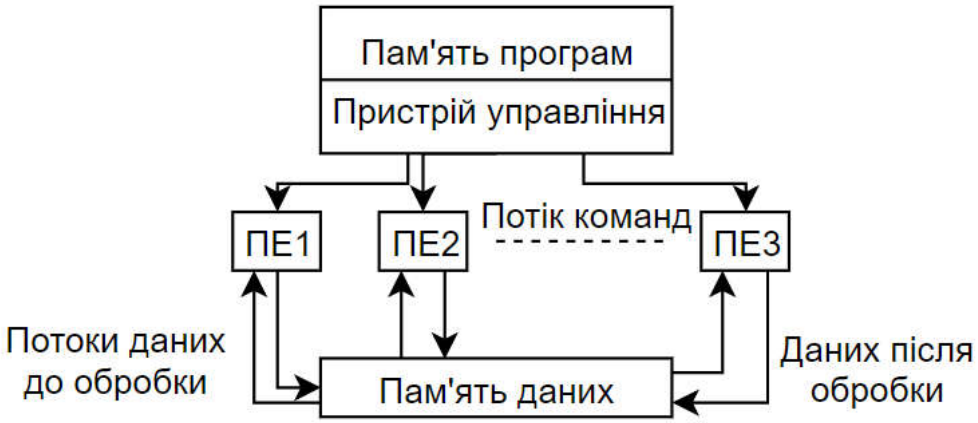

Рисунок 1.2 – Тип архітектури SIMD

До класу MISD ряд дослідників відносить конвеєрні ЕОМ, однак це не знайшло остаточного визнання. Також, можливо вважати MISD системами, системи з гарячим резервуванням. Крім цього, до архітектури MISD деякі відносять систолічні масиви процесорів. Архітектура типу MISD показана на рисунку 1.3.

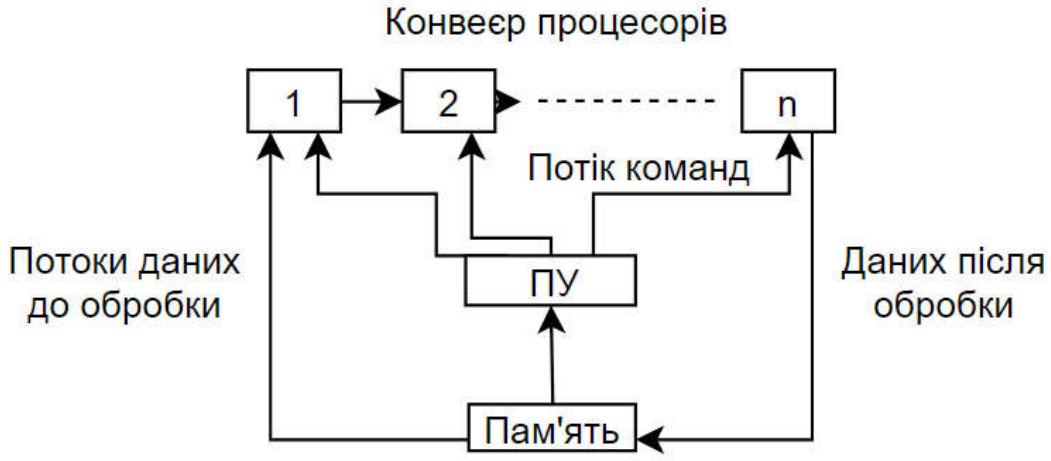

Рисунок 1.3 – Архітектура MISD

Клас MIMD включає в себе багатопроцесорні системи, де процесори обробляють множинні потоки даних. Сюди відносять традиційні

мультипроцесорні машини, багатоядерні і багатопотокові процесори, а також комп'ютерні кластери [10]. Архітектура даного класу зображена на рисунку 1.4.

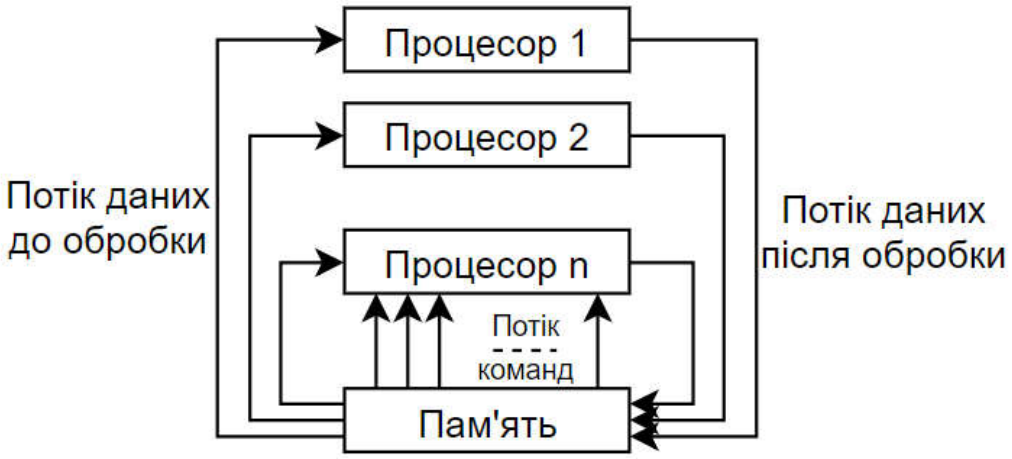

Рисунок 1.4 – Архітектура типу MIMD

У спеціалізованій літературі можна зустріти ще такі підкласи MIMD–класу:

– SPMD (Single Program Multiple Data);

– MPMD (Multiple Programs Multiple Data).

SPMD (Single Program Multiple Data) – описує систему, де на всіх процесорах MIMD–машини виконується тільки одна єдина програма і на кожному процесорі вона обробляє різні блоки даних.

MPMD (Multiple Programs Multiple Data) – описує систему в двох випадках:

– на одному процесорі MIMD–машини працює майстер–програма, а на інших підпорядкована програма, роботою якої керує майстер–програма (принцип master / slave або master / worker);

– на різних вузлах MIMD–машини працюють різні програми, які по–різному обробляють один і той же масив даних (принцип coupled analysis), здебільшого вони працюють незалежно один від одного, але час від часу надсилати та отримувати дані для переходу до наступного кроку [11].

Відношення конкретних машин до конкретного класу сильно залежить від точки зору дослідника. Так, конвеєрні машини можуть бути віднесені і до класу SISD (конвеєр – єдиний процесор), і до класу SIMD (векторний потік даних з конвеєрним процесором), і до класу MISD (безліч процесорів конвеєра обробляють один потік даних послідовно), і до класу MIMD – як виконання послідовності різних команд (операцій щаблів конвеєра) з множинним скалярним потоком даних (вектором).

1.2 Поняття стану комп’ютерної системи

Показники надійності визначаються з розрахунків, проведенням випробувань і обробкою результатів (статистичних даних) експлуатації виробів, моделюванням на ЕОМ(електронна обчислювальна машина), а також в результаті аналізу фізико–хімічних процесів, обумовлюючих надійні вироби. Розрахунки надійності засновані на тому, що при певній структурі виробу і наявному законі розподілу напрацювання повністю виробів цього типа існують сповна певні залежності між показниками надійності окремих елементів і надійні вироби в цілому. Для встановлення таких залежностей використовуються наступні прийоми: рішення рівнянні, складених на підставі структурної схеми надійності (використання послідовнопаралельних структур) або на підставі логічних зв'язків між станами виробу (використання алгебра логіки); вирішення диференціальних рівнянь що описують процес переходу виробу з одного стану в інших (використання графів станів); складання функцій, що описують стани складного виробу.

Розрахунки надійності виробляються головним чином на етапі проектування виробів з метою прогнозування для даного варіанту виробу очікуваної надійності. Це дозволяє вибрати найбільш відповідний варіант конструкції і методи забезпечення надійності, виявити «слабкі місця»,

обґрунтовано призначити робочі режими, форму і порядок обслуговування вироби. Випробування на надійності виробляються на етапах розробки дослідного зразка і серійного виробництва виробу. Існують випробування на в надійність изначальні, в результаті яких визначають показники надійності; контрольні, такі, що мають на меті контроль якості технологічного процесу, що забезпечує з деяким ризиком надійності не нижче заданою; прискорені, в ході яких використовують чинники, прискорюючи процес виникнення відмов; неруйнівні, засновані на вживанні методів дефектоскопії і інтроскопії, а також на вивченні непрямих ознак (шумів, теплових випромінювань і т.п.), супутніх виникненню відмов. Моделювання на ЕОМ(електронна обчислювальна машина) є найбільш ефективним засобом аналізу надійності складних систем. Широко поширені два алгоритму моделювання: перший, заснований на моделюванні фізичних процесів, що відбуваються в досліджуваному об'єкті (оцінка надійності при цьому визначається по числу виходів параметрів об'єкту за межі допуску); другий, заснований на вирішенні систем рівнянь, що описують стани досліджуваного об'єкту [12].

Аналіз фізико–хімічних процесів також дозволяє отримати оцінку надійності досліджуваного виробу, т.к. часто удається встановити залежність надійності від стану і характеру протікання фізико–хімічних процесів (співвідношення показників міцності і навантаження, зносостійкість, наявність домішок в матеріалах, зміна електричних і магнітних характеристик, шумові ефекти і т.д.). Найчастіше аналіз фізико–хімічних процесів застосовується при оцінці елементів радіоелектронної апаратури.

Переважна більшість користувачів регулярно працюють за комп'ютером і не замислюються про те, що в певний момент він може вимкнутися і більше не ввімкнутись зовсім. Наприклад, досить часто виникає проблема, коли щойно зібраний або оновлений комп'ютер не включається. Або ж, якщо комп'ютер раптово перестає працювати. У такому випадку головне – правильно визначити несправність. Тому, що ремонт в певних випадках може бути не обов’язковим. Спочатку треба розібратися з причинами, що можуть

викликати певну несправність [13]. Як відомо, пил і несприятливі кліматичні умови погіршують стан комплектуючих персонального комп’ютера. Тому вихід апаратного забезпечення з ладу може бути викликаний окисленням контактів, потраплянням пилу (відповідно, статичної електрики) на мікросхеми і роз'єми, їх температурний збій.

Також всі несправності можуть бути наслідком стрибків напруги, неправильної роботи блоку живлення, або ж неправильного заземлення. Найперше, що можна порекомендувати в такому випадку – використання мережевих фільтрів та заземлення комп'ютера. В першу чергу при несправності персонального комп’ютера слід провести візуальний огляд, зняти захисну кришку та добре оглянути зовнішній вигляд всіх компонентів чи нема відхилень від норми, а також варто звернути увагу чи не присутній запах диму, що часто притаманно проблемам, що пов’язані зі стрибками напруги. Якщо явних ознак несправностей на елементах комп’ютера не виявлено, то варто перевірити надійність підключення до живлення. Якщо перевірка не дала результатів, то можна включити комп’ютер і перевірити чи функціонують вентилятори на блоку живлення (БЖ) і на кулері процесора, також можна перевірити кріплення кулера. Якщо вентилятор не функціонує і жорсткий диск не відтворює характерного звуку розкручування, то можна зробити висновок, що несправність стосується блоку живлення [14].

Наявність напруги на виході блоку живлення можна перевірити тестером, поміявши величину напруги на контактах системної плати в тому місці, де вони з'єднані з блоком живлення. Для впевненості можна підключити інший блок живлення і перевірити справність інших компонентів комп’ютера. Незважаючи на те, що монітор ламається нечасто, потрібно перевірити чи подаються йому сигнали з відеоадаптера. Для цього осцилографом необхідно перевірити наявність діючих сигналів.

Система автоматичної діагностики комп’ютера являє собою комплекс програмних, мікропрограмних і апаратних засобів. Розрізняють системи тестової і функціональної діагностики. У системах тестової діагностики

результати на діагностуючий пристрій надходять від засобів діагностування. У середніх і великих електронно–обчислювальних машинах зазвичай використовуються спеціалізовані засоби діагностування [15]. У мікро–електронно–обчислювальних машинах частіше використовуються вбудовані засоби подачі тестових впливів в зовнішні засоби для зняття і обробки результатів. Процес діагностики включає в себе декілька етапів (простих перевірок), кожна з яких характеризується подаючою на пристрій або ж робочою напругою, що знімається з відповідного пристрою. Через те, що існує велика кількість проблем, які можуть виникнути, не існує єдиного підходу до діагностики комп'ютера. Існує перелік проблем з апаратною частиною комп'ютера, які зустрічаються найчастіше:

– комп'ютер не вмикається;

– комп'ютер вмикається, але на екрані немає інформації;

– комп'ютер вмикається і видає специфічні звуки ("пищить");

– різноманітні помилки, пов'язані зі стартом BIOS;

– комплектуючі не функціонують;

– комп'ютер перегрівається;

– апаратна несумісність обладнання;

– система працює дуже повільно.

Отож, очевидно, що діагностику комп'ютерної системи можна проводити за допомогою фізичних ресурсів людини, проте аналогом служать програмні засоби, які зараз є досить поширеними серед активних користувачів комп'ютера [16]. Ці програмні засоби – це так званий набір команд, які надходять до кожної складової комп'ютера і надають певний результат, що відображається в діалоговому вікні. Програмна система діагностики опирається на вже внесені дані щодо нормальної роботи складових комп'ютера, порівнює їх, аналізує та відображає отримані результати. Дізнатись найпростіші характеристики персонального комп'ютера можна і стандартними засобами Windows. Один із способів – це перегляд за допомогою диспетчера завдань (рисунок 1.5).

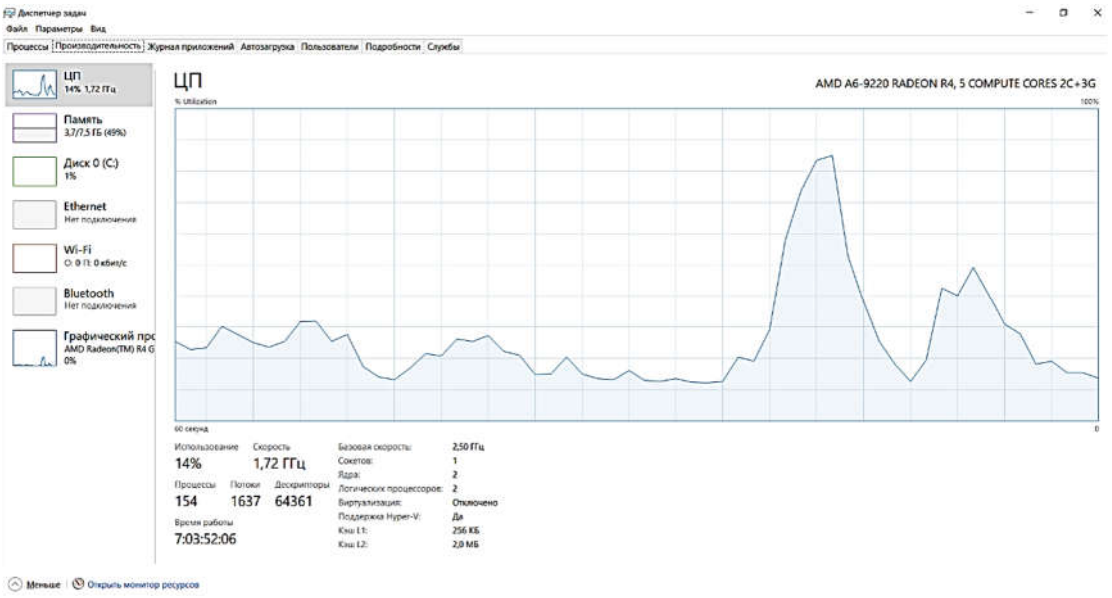

Рисунок 1.5 – Поточні характеристики комп’ютера

Для діагностики комп’ютерних систем існує безліч способів, програмних, апаратних та навіть спеціальних веб–ресурсів. Але загальні методи аналізу та діагностики є для всіх засобів однаковими. Це обов’язкова перевірка складових системного блоку як на наявність фізичних дефектів, так і перевірка за допомогою програмних засобів, що допомагають ідентифікувати неточності та неполадки, що можуть стосуватись програмного коду системних додатків або характеристик тих же апаратних засобів [17].

Аналіз комп’ютерної системи – поширена та затребувана тематика для всіх користувачів комп’ютера. Вчасна діагностика дозволяє запобігти серйозним проблемам і відповідно уникнути ремонту, який може вимагати великої кількості коштів. Тому в діагностиці важливо враховувати кожен елемент, незалежно від його значущості. Це особливо важливо при перевірці системних додатків на наявність різноманітних вірусів, адже вони можуть ховатися в найменших файлах і згодом нанести безповоротну шкоду програмному та апаратному забезпеченню. Часто такі пошкодження системи дуже важко або взагалі неможливо відлагодити [18].

1.3 Нечітка логіка та її основні поняття

Нечітка логіка, звана "fuzzy control", добре відома інженерам програмістам систем управління як зручний засіб програмування і моніторингу додатків управління технологічними процесами. Термін нечітка логіка вперше був використаний з 1965 року Лотфі Заде, професором УК Берклі в Каліфорнії. Він зауважив, що звичайна комп'ютерна логіка не здатна маніпулювати даними, що представляють суб'єктивні чи незрозумілі людські ідеї.

Нечітка логіка застосовується в різних областях, від теорії управління до медицини. Вона була розроблена для того, щоб комп'ютер міг визначити відмінності між даними, які не є ні точними, ні хибними. Щось подібне до процесу міркування людини.За аналогією з традиційними засобами управління технологічними процесами, системи на основі нечіткої логіки можуть використовуватися для опису петель регулювання і брати участь в обчисленні керуючого впливу відповідно до однієї або великої кількості точок завдання для одного або більшої кількості вимірів [19].

Правила нечіткої логіки дозволяють забезпечити:

– застосування існуючого досвіду управління;

– використовувати гнучкі правила в разі неможливості точно моделювати систему за допомогою традиційних засобів.

Покращення якості управління при цьому здійснюється за допомогою:

– саморегулювання системи управління;

– випереджаюча зміна вихідного впливу (функція попередження), базуючись на подіях, які не можуть бути враховані у разі застосування традиційних способів управління.

Кількість додатків заснованих на даних методах управління безперервно збільшується для безперервних процесів, для додатків пакетної обробки, а

також для автоматизованих систем. Нечітка логіка, завдяки використання її в цій галузі, отримала опис і формулювання в якості методу програмування. Вона дозволяє систематизувати емпіричні знання і застосувати їх для управління процесами в разі труднощів із застосуванням класичних методів управління [20]. Теорія нечіткої логіки дозволяє описати набори методів управління, які нескладно застосувати для реальної системи і дозволяє врахувати досвід операторів і технологів для динамічного управління процесом. Це дозволяє описувати на нечіткій логіці окремі частини виробничого процесу, такі як ініціалізація, завдання параметрів.

Ось кілька важливих характеристик нечіткої логіки:

– гнучка та проста у виконанні техніка машинного навчання;

– допомагає наслідувати логіку людської думки;

– логіка може мати два значення, які представляють два можливі рішення;

– ефективний метод для невизначених або приблизних міркувань;

– нечітка логіка розглядає виводи як процес поширення обмежень;

– нечітка логіка дозволяє будувати нелінійні функції довільної складності;

– нечітка логіка повинна будуватися з повним керівництвом експертів.

В інженерних задачах застосовується, як правило, механізм нечіткого виводу Мамдані, який можна реалізувати у середовищі Matlab. Метод Мамдані був одним з перших, побудованих за допомогою теорії нечітких множин. Він був запропонований в 1975 році Ібрагімом Мамдані.

Метод Мамдані є нечіткою системою виведення (НСВ). НСВ являє собою систему, що використовує теорію нечітких множин для відображення входів (функцій в разі нечіткої класифікації) до виходів (класів в разі нечіткої класифікації) [21].

Метод складається з наступних етапів:

– формування бази правил;

– фазифікація;

– агрегування підумов;

– активізація підвисновків;

– акумулювання висновків;

– дефазифікація.

Кожен етап виконується послідовно, до того ж кожен наступний етап отримує на вхід значення, що були отримані в результаті роботи попереднього. Для вхідних значень метод Мамдані використовує певну базу правил. Загальна схема роботи нечіткої логіки зображена на рисунку 1.6.

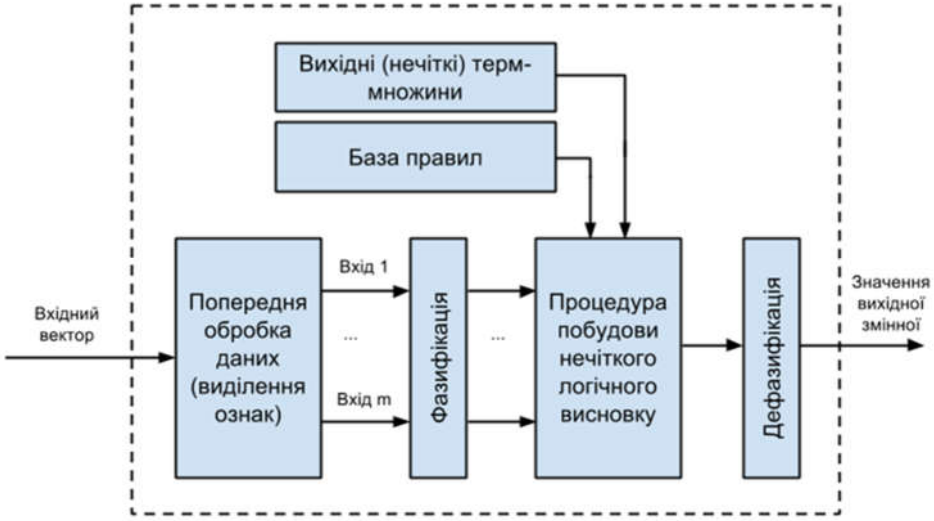

Рисунок 1.6 – Загальна схема роботи нечіткої логіки

Для прикладу, правила можуть бути представлені у такому вигляді:

– якщо $x \epsilon Y1$, тоді $z \epsilon N1$,

де x – вхідне значення,

z – вихідне значення,

Y і N – нечіткі множини.

Фазифікацію вхідних змінних ще називають приведенням їх до нечіткості. На вхід надходить вже сформована база правил і певний масив вхідних даних $A=\{a_1,...,a_n\}$. В такому масиві містяться значення всіх вхідних

змінних. Метою проведення даного етапу є отримання значень істинності для всіх умов з бази правил. Це відбувається наступним чином: для кожної з умов знаходиться значення $b_i = \mu(a_i)$. Таким чином існує безліч значень b_i $(i = 1..k)$.

На етапі агрегування умов визначається ступінь істинності умов для кожного з правил системи нечіткого виведення [20]. Тобто, для кожної умови знаходиться мінімальне значення істинності всіх її підумов. Формально, це виглядає так (формула 1.1):

$$c_j = min\{b_i\}\,, \tag{1.1}$$

де $j = 1..q$;

i – число з безліччю номерів підумов, в яких бере участь *j*–а вхідна змінна.

На етапі активізації виводів відбувається перехід від умов до підвиводів. Для кожного підвивода знаходиться ступінь істинності $d_i = c_i * F_i$, де $(i = 1..q)$. Потім, кожному *i*–му підвиводу зіставляється безліч D_i з новою функцією належності. Її значення визначається як мінімум з d_i і значення функції належності терму з підвиводу. Цей метод називається min–активізацією, який формально записується в такий спосіб (формула 1.2):

$$\mu_i{}' = min\{d_i, \mu_i(x)\}\,, \tag{1.2}$$

де $\mu_i{}'(x)$ – «активована» функція належності;

$\mu_i(x)$ – функція належності терму;

d_i – ступінь істинності *i*–го підвиводу.

Отже, мета цього етапу – отримання суми «активованих» нечітких множин D_i для кожного з підвиводів в базі правил $(i = 1..q)$.

На етапі акумуляції виводи отримують нечітку множину (або їх об'єднання) для кожної з вихідних змінних. Виконується цей етап наступним чином: *i*–тій вихідній змінній зіставляється об'єднання множин $E_i = \cup D_j$. Де

j – номер підвиводів, в яких бере участь i–та вихідна змінна ($i = 1..s$).

Об'єднанням двох нечітких множин є третя нечітка множина з наступною функцією приналежності (формула 1.3):

$$\mu_i{}'(x) = max\{\mu_1(x), \mu_2(x)\}\,, \tag{1.3}$$

де $\mu_1(x), \mu_2(x)$ – функції приналежності поєднуваних множин [22].

Метою дефазифікації є отримання кількісного значення для кожної з вихідних лінгвістичних змінних. На практиці це відбувається так: розглядається i–та вихідна змінна і її відповідна множина $E_i(1..s)$. Далі за допомогою методу дефазифікації знаходиться кінцеве кількісне значення вихідної змінної. У такій реалізації алгоритму використовується метод центру тяжіння, в якому значення i–ої вихідної змінної розраховується за формулою 1.4 :

$$y_i = \frac{\int_{\min}^{\max} x * \mu_i(x)\,dx}{\int_{\min}^{\max} \mu_i(x)\,dx}\,, \tag{1.4}$$

де $\mu_i(x)$ – функція приналежності відповідної нечіткої множини E_i;

min і max – кордони універсуму нечітких змінних;

y_i – результат дефазифікації.

Переваги нечіткої логічної системи:

– структура нечітких логічних систем проста і зрозуміла;

– нечітка логіка широко використовується в комерційних і практичних цілях;

– допомагає нам контролювати машини та споживчі товари;

– може не пропонувати точних міркувань, але єдино прийнятне міркування;

– допомагає впоратися з невизначеністю в техніці;

– переважно надійні, оскільки не потребують точних входів;

– може бути запрограмовано в ситуації, коли датчик зворотного зв'язку перестане працювати;

– можна легко змінити, щоб поліпшити або змінити продуктивність системи;

– можна використовувати недорогі датчики, що допоможе зберегти загальну вартість і складність системи на низькому рівні;

– забезпечує найбільш ефективне вирішення складних питань.

Недоліки нечітких систем:

– нечітка логіка не завжди точна, тому результати сприймаються на основі припущення, як результат вона може не бути широко прийнятою;

– нечіткі системи не мають можливості машинного навчання як добре розпізнавати шаблони нейронної мережі;

– перевірка нечіткої системи, що базується на знаннях, потребує широкого спектру тестування обладнання;

– встановлення точних, нечітких правил та функцій є складним завданням;

– нечітка логіка часто плутається з теорією ймовірностей.

1.4 Аналіз існуючих програмних засобів на основі нечіткої логіки

Модель експертної оцінки діагностичних особливостей комп'ютерної системи. У даній роботі пропонується використовувати експертну діагностичну систему (ЕЦП) для аналізу технічного стану комп'ютерної системи. Математичний апарат, який дозволяє керувати прогнозованою оцінкою стану об'єкта діагностики (апаратна, програмна чи персональна), є нечіткою логікою. На етапі підготовки діагностичного експерименту (ДЕ) пропонується описати діагностичні особливості комп'ютерної системи з

точки зору лінгвістичних змінних, що дає можливість використовувати знання та досвід експерта у знайомій формі.

Дану експертну діагностичну систему (EDS) рекомендується використовувати для аналізу технічного стану комп'ютерної системи. Математичний апарат, який дозволяє керувати прогнозованою оцінкою стану об'єкта діагностики (апаратна, програмна чи персональна), є нечіткою логікою. На стадії підготовки діагностичного експерименту (ДЕ) пропонується описати діагностичні особливості комп'ютерної системи з точки зору лінгвістичних змінних, що дає можливість використовувати знання та досвід експерта в їх знайомій формі. Принцип роботи подібної системи показано на рисунку 1.7.

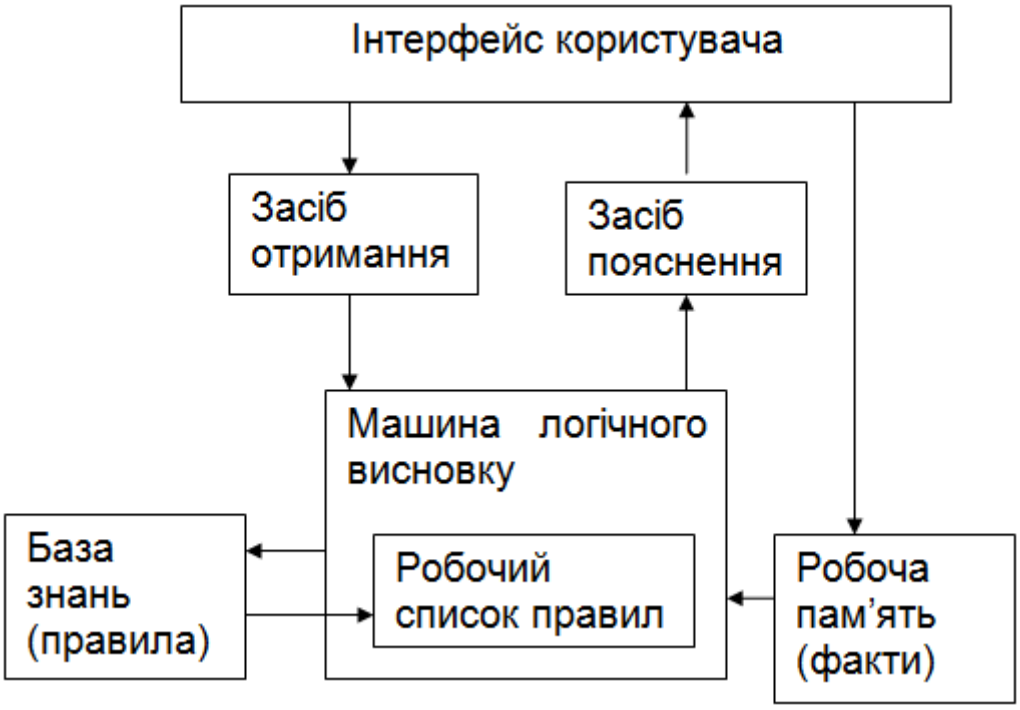

Рисунок 1.7 – Принцип роботи експертної системи

Діагностування комп'ютерного обладнання з використанням експортних систем, це поточне дослідження вводить метод, який зменшує проблеми при діагностиці апаратних збоїв. Введена експертна інтелектуальна система, що використовує технологію, основана на правилах, щоб діагностувати апаратні збої. Користувачеві або комп'ютерному техніку не

потрібно перевіряти окремі частини обладнання комп'ютера окремо. Їм просто потрібно ввести апаратне забезпечення комп'ютера та симптоми в систему, а потім діагностувати апаратне обладнання. Розробка діагностичної діагностики несправностей комп'ютерної апаратури з використанням методу, основаної на правилах, базується на методології розробки експертної системи, яка складається з восьми етапів; Дослідження та аналіз, концептуалізація, оцінка проблем, придбання та аналіз знань, проектування та впровадження, тестування, документація та управління. Система відображатиме можливі причини та запропонує рішення. Правила запропонованої експертної системи викладені у формі "if–then" тверджень. Сама система на основі правил використовує просту техніку, і вона починається з правил, що містить всі відповідні знання, закодовані в правила "Якщо–Тоді". Категорії цієї системи: аудіо, жорсткий диск, клавіатура, миша, блок живлення, процесор, запуск, серійний ATA, пристрій USB, принтер, материнська плата, процесор, оперативна пам'ять, периферія, BIOS, відеомонітор та адаптер, DVD–привід та DVD / CD запис.

Як висновок, мета побудови цієї простої експертної системи полягає в тому, щоб допомогти комп'ютерним користувачам діагностувати несправність комп'ютерного обладнання. Крім того, ця програма допомагає користувачам вирішувати деякі основні проблеми з апаратним забезпеченням або виконувати більш масштабні усунення несправностей, перш ніж звертатися за допомогою до служби підтримки або фахівців.

Синтез нейро–нечіткої діагностичної моделі з хешованим перетворенням в послідовному і паралельному режимі. Вирішене актуальне завдання підвищення швидкості побудови нейро–нечітких моделей за прецедентами.

Мета – створення нейро–нечіткого методу синтезу мережі з високою швидкістю обчислень і дозволити реалізувати синтез нейро–нечітких мереж у паралельному режимі.

Метод побудови нейро–нечітких моделей за допомогою прецедентів, що

зменшує розмір вхідних даних шляхом хешування перетворення на одномірну вісь, що зберігає локальну топологію кластерів у функціональному просторі, оцінює значення ознак і екземпляри на основі виділених кластерів, а також утворює розділ вихідного простору функцій у автоматичному режимі, синтезує структуру і автоматично налаштовує параметри нейро–нечіткої моделі, виключаючи з навчального процесу нейро–нечіткої моделі неінформативні дані, тим самим спрощуючи структуру отриманої моделі, дозволяє виконувати більшість обчислювально дорогих операцій у паралельному режимі, що дозволяє автоматизувати процес синтезу нейро–нечіткої моделі за допомогою прецедентів, а також збільшення швидкості побудови нейро–нечіткої моделі як послідовної, так і паралельної реалізації обчислень.

Програмне забезпечення, що реалізує запропонований метод, було розроблено та використано в обчислювальних експериментах, що досліджують властивості методу. Експерименти підтвердили ефективність запропонованого методу та програмного забезпечення.

Досліди також дозволяють рекомендувати їх для використання на практиці для вирішення проблем діагностики та автоматичної класифікація за ознаками.

Визначення зручності користування за допомогою багатоетапної нечіткої системи. Оцінка програмного забезпечення важлива для покращення модифікації та вдосконалення процесу розробки програмного забезпечення. Для оцінки програмного процесу існує безліч факторів. Одним з факторів є якість програмного забезпечення, яке неможливо легко обчислити; оскільки якість програмного забезпечення залежить від інших чинників. Працездатність програмного забезпечення є одним з важливих аспектів, від яких залежить якість програмного забезпечення. Ряд моделей програмного забезпечення були запропоновані рядом дослідників, кожна модель розглядає безліч факторів.

У реальному світі нам доводиться стикатися з багатьма перешкодами

для реалізації будь–якої із запропонованих моделей зручності користування, оскільки відсутнє його точне визначення та концепція загальноприйнятої зручності.

Узагальнена модель зручності використання (УМЗ) з таксономією була запропонована в даному проекті. Цей документ також показує, як визначити зручність використання програмного застосунку за допомогою нечіткої системи, яка була впроваджена за допомогою багаторівневого нечіткого логічного набору інструментів.

1.5 Аналіз дерева рішень та постановка задачі

Оскільки при діагностиці комп’ютерної системи необхідно врахувати поточні параметри системи, такі як продуктивність, допустимі затрати пам’яті та необхідний рівень стійкості до часового аналізу, то для вирішення цієї задачі варто застосувати апарат нечіткої логіки.

Математична теорія нечітких множин (fuzzy sets) і нечітка логіка (fuzzy logic) є узагальненнями класичної теорії множин і класичної формальної логіки. Дані поняття були вперше запропоновані американським ученим Лотфі Заде (Lotfi Zadeh) у 1965 р. Основною причиною появи цієї теорії стала наявність нечітких і наближених міркувань при описі людиною процесів, систем, об’єктів.

Основними перевагами нечітких систем у порівнянні з іншими є:

– можливість оперувати вхідними даними, заданими нечітко, наприклад,значеннями, що невпинно змінюються в часі (динамічні задачі);

– можливість нечіткої формалізації критеріїв оцінки і порівняння;

– можливість проведення якісних оцінок як вхідних даних, так і виведених результатів, оскільки система оперує не тільки власне значеннями даних, а й їх ступенем вірогідності та її розподілом;

– можливість проведення швидкого моделювання складних динамічних систем та їх порівняльний аналіз із заданим ступенем точності.

В інженерних задачах застосовується, як правило, механізм нечіткого висновку Мамдані. В ньому використовується мінімаксна композиція нечітких множин. Даний механізм включає наступну послідовність дій:

– процедура фазифікації: визначаються степені істинності, тобто значення функцій належності $MF_i(x)$ для лівих частин кожного *i*–го правила (передумов);

– нечіткий висновок. Спочатку визначаються мінімальний рівень "відсічення" для лівої частини кожного з правил $A_i = min(\, MF_i(x))$, а потім знаходяться "усічені" функції належності висновку $B_i = min(\, A_i, B_i)$;

– композиція або об'єднання отриманих "усічених" функцій, для чого використовується максимальна композиція нечітких множин $MF(y) = max(\, B_i(y))$;

– дефазифікація або приведення до чіткості.

Існує декілька методів дефазифікації. Наприклад, метод середнього центру або центроїдний метод. Геометричний зміст такого значення – центр ваги для кривої функції належності отриманого виходу.

Застосування апарату нечіткої логіки при створенні апаратно–програмного засобу для здійснення діагностики комп'ютерної системи шляхом вибору оптимального методу модулярного експоненціювання для кожної окремої характеристики та врахування поточних параметрів самої КС дозволить забезпечити стійкість системи до часового аналізу в режимі реального часу [4].

Для вирішення цього завдання необхідно (рисунок 1.8):

– дослідити основні параметри методів модулярного експоненціювання, що найчастіше застосовуються в сучасних комп'ютерних системах та визначити стійкість кожного з них до часового аналізу;

– розробити метод обробки нечіткої інформації під час діагностики комп'ютерної системи, враховуючи значення всіх характеристик;

– розробити структуру засобу реалізації розробленого методу та дослідити його основні характеристики.

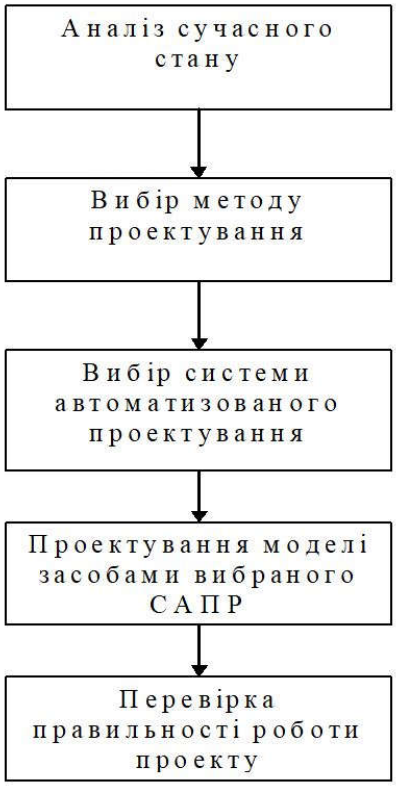

Рисунок 1.8 – Дерево рішень дипломного проекту

Спроектувавши схему, за якою буде проводитись подальша робота, можна досягнути найменшої кількості неточностей, а значить досягнути бажаного результату в найкоротші терміни. Це стане основою майбутньої розробки, за допомогою точно поставленого завдання, структурної схеми та проаналізованому матеріалу розробка нечіткої системи буде точною та з найменшою кількістю помилок, що дозволить заощадити значну кількість часу та ресурсів

1.6 Висновки до розділу 1

Перший розділ дипломної роботи носить аналітичний характер. В ньому було розглянуте поняття комп’ютерної системи, основні види та типи

архітектур комп’ютерних систем, а саме їх функціональні відмінності та спільні характеристики, основні переваги та недоліки вибору тої чи іншої архітектури.

Охарактеризовано поняття стану комп’ютерної системи, основні показники та характеристики від яких він залежить. Також описано найпоширеніші проблеми та несправності, що трапляються як з пересічними користувачами, так і у великих корпораціях.

Проаналізовано основні поняття нечіткої логіки, функціональні елементи та алгоритми побудови правил. Розглянуто основні процеси роботи нечіткої системи, а також формули, які є основою роботи функцій належності. Наведено основні переваги використання нечітої логіки в реалізації інженерних розробок.

Розглянуто існуючі програмні засоби, що працюють на основі нечіткої логіки, структурну схему роботи експертної та нечіткої систем. Зроблено порівняння принципів роботи існуючих засобів з розробленим у дипломному проекті, а також проаналізовано напрямки розвитку подібних рішень.

Під час роботи також було визначено основні задачі, які повинні бути виконані під час роботи над дипломним проектом та основні засоби та методи, за допомогою яких це може бути реалізованим.

2 НЕЧІТКІ АЛГОРИТМИ АНАЛІЗУ СТАНУ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Алгоритм Мамдані

Нечіткий вивід займає центральне місце в системах нечіткого моделювання. Процес нечіткого виведення являє собою певну процедуру або алгоритм отримання нечітких виводів на підставі нечітких передумов з використанням основних операцій нечіткої логіки. Основні етапи нечіткого виводу:

– визначення структури системи нечіткого виведення;

– формування бази правил системи нечіткого виведення;

– фазифікація вхідних змінних;

– обчислення значень ступенів належності підумови правил нечітких продукцій;

– агрегування підумови правил нечітких продукцій;

– активізація підумов правил нечітких продукцій;

– акумулювання виводів правил нечітких продукцій;

– дефазифікація вихідних змінних.

Лотфі Заде є ключовою особою в формуванні поняття нечіткої логіки. Він став не тільки батьком–засновником цілої наукової теорії, написавши в 1965 році фундаментальну працю «Fuzzy Sets», а й пропрацював різні можливості її практичного застосування. Він описав свій підхід в 1973 році в тексті «Outline of a New Approach to the Analysis of Complex Systems and Decision Processes» (опублікованому в журналі IEEE Transactions on Systems). Примітно, що відразу після його виходу одна заповзятлива датська фірма досить успішно застосувала викладені в ньому принципи для вдосконалення своєї системи управління виробничим процесом [24].

Але при всіх заслугах Л. Заде, не менш важливий внесок внесли послідовники цієї теорії. Наприклад, англійський математик Е. Мамдані (Ebrahim Mamdani). У 1975 році він розробив алгоритм, який був

запропонований як метод для управління паровим двигуном. Запропонований ним алгоритм, заснований на нечіткому логічному виводі, дозволив уникнути надмірно великого обсягу обчислень і був гідно оцінений фахівцями. Цей алгоритм в даний час отримав найбільше практичне застосування в задачах нечіткого моделювання. Перш ніж почати знайомство з алгоритмом важливо коротко ознайомитися з наступними визначеннями. Нечітка змінна – це кортеж виду <α, X , A >, де:

– α – ім'я нечіткої змінної;

– X – її область визначення;

– A – нечітка множина на універсумі X.

Прикладом є нечітка змінна <«Системний блок», $\{x \mid 0 \text{ кг} < x < 35 \text{ кг}\}$, $B = \{x, \mu(x)\}$> характеризує масу системного блока комп'ютера. Будемо вважати його важким, якщо його маса буде більше 16 кг (рисунок 2.1).

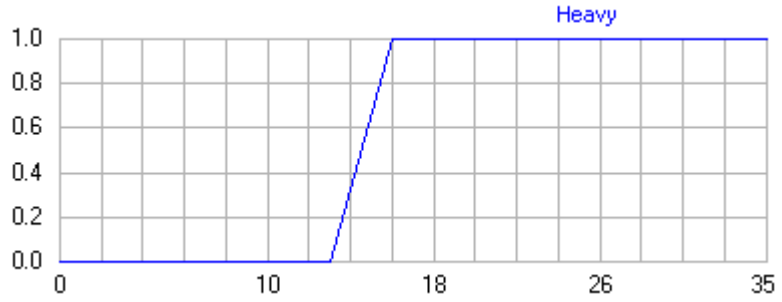

Рисунок 2.1 – Графік функції приналежності μ (x) для нечіткого безлічі B

Лінгвістична змінна це кортеж типу <β, T, X, G, M >, де:

– β – ім'я лінгвістичної змінної;

– T – безліч її значень (термів);

– X – універсум нечітких змінних;

– G – синтаксична процедура утворення нових термів;

– M – семантична процедура, яка формує нечіткі множини для кожного терма даної лінгвістичної змінної.

До прикладу, припустимо, ми маємо суб'єктивну оцінку маси системного блоку. Вона, наприклад, може бути отримана від працівників ІТ–відділу (які виступають в ролі експертів), які безпосередньо мають справу з подібною технікою. Формалізувати цю оцінку можна за допомогою наступної лінгвістичної змінної <β, T, X, G, M > (рисунок 2.2), де:

– β – Системний блок;

– T – {«Легкий системний блок (Light)», «Системний блок середньої маси (Medium)», «Важкий системний блок (Heavy)»};

– X = [0; 35];

– G – процедура утворення нових термів за допомогою логічних зв'язків і модифікаторів. Наприклад, «дуже важкий системний блок»;

– M – процедура завдання на універсумі X = [0; 35] значень лінгвістичної змінної, тобто термів з безлічі T.

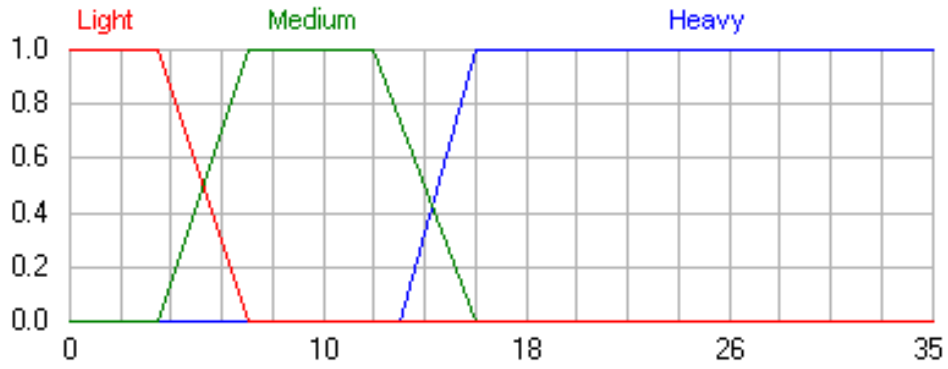

Рисунок 2.2 – Графіки функцій належності значень лінгвістичної змінної «Системний блок»

Даний алгоритм описує кілька послідовно виконуваних етапів (рисунок 2.3). При цьому кожний наступний етап отримує на вхід значення отримані на попередньому кроці [25].

Рисунок 2.3 – Діаграма діяльності процесу нечіткого виведення

Алгоритм примітний тим, що він працює за принципом «чорного ящика». На вхід надходять кількісні значення, на виході вони ж. На проміжних етапах використовується апарат нечіткої логіки та теорія нечітких множин. В цьому і полягає елегантність використання нечітких систем. Можна маніпулювати звичними числовими даними, але при цьому використовувати гнучкі можливості, які надають системи нечіткого виведення.

Правила (Rule) складаються з умов (Condition) і висновків (Conclusion), які в свою чергу є нечіткими висловлюваннями (Statement). Нечітке висловлювання включає в себе лінгвістичну змінну (Variable) і терм, який представлений нечітким безліччю (FuzzySet). На нечіткій множині визначена функція приналежності, значення якої можна отримати за допомогою методу getValue (). Це метод визначений в інтерфейсі FuzzySetIface.

Для реалізації алгоритму можна використовувати об'єктно–орієнтований підхід. Отже, етапи нечіткого виводу виконуються послідовно. І всі значення, отримані на попередньому етапі, можуть використовуватися на наступному.

Вихідний код написаний на мові програмування Java, що на діаграмі (рисунок 2.4) показує найбільш істотні зв'язки і відносини між класами, задіяними в алгоритмі.

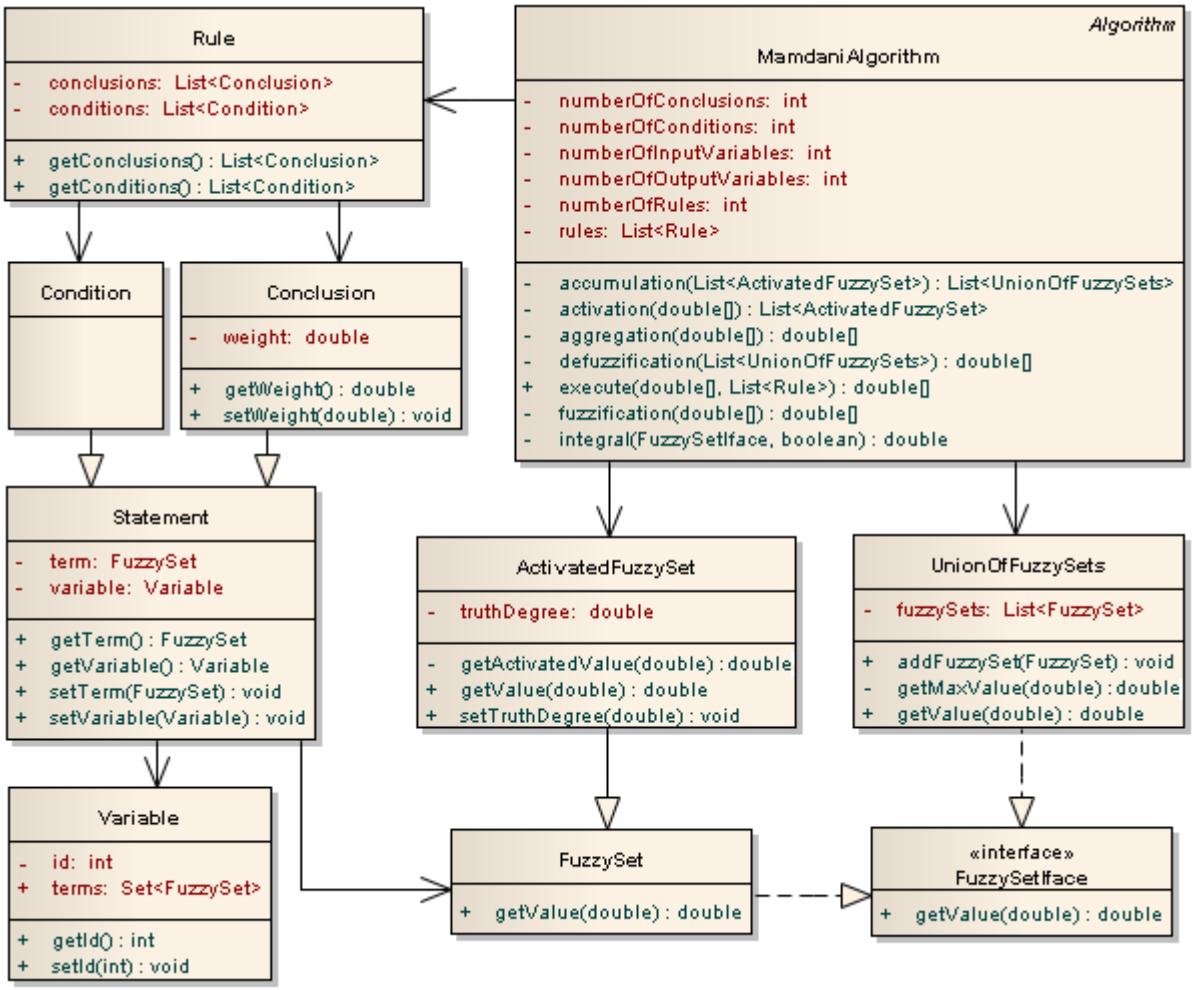

Рисунок 2.4 – Діаграма класів реалізації алгоритму Мамдані

При виконанні алгоритму необхідно буде скористатися «активізованою» нечіткою множиною (ActivatedFuzzySet), яке певним чином перевизначає функцію приналежності нечіткої множини (FuzzySet). Також в алгоритмі використовується об'єднання нечітких множин (UnionOfFuzzySets). Об'єднання також є нечіткою множиною [10].

Алгоритм Мамдані включає в себе всі етапи (рисунок 2.3) і використовує базу правил (List <Rule>) в якості вхідних даних. Також алгоритм передбачає використання «активізованих» нечітких множин (ActivatedFuzzySet) і їх об'єднань (UnionOfFuzzySets).

Перший етап – етап формування бази правил. База правил – це безліч правил, де кожному підвисновку зіставлений певний ваговий коефіцієнт.

База правил може мати такий вигляд (для прикладу використовуються

правила різних конструкцій):

– RULE_1: IF «Condition_1» THEN «Conclusion_1» (F1) AND «Conclusion_2» (F2);

– RULE_2: IF «Condition_2» AND «Condition_3» THEN «Conclusion_3» (F3);

– RULE_n: IF «Condition_k» THEN «Conclusion (q–1)» (Fq–1) AND «Conclusion_q» (Fq);

Де Fq – вагові коефіцієнти, які означають ступінь впевненості в істинності одержуваного підвисновку (i=1..q). За замовчуванням ваговий коефіцієнт приймається рівним 1. Лінгвістичні змінні, присутні в умовах називаються вхідними , а в заключних вихідними .

Використані позначення:

– n – число правил нечітких продукцій (numberOfRules);

– m – кількість вхідних змінних (numberOfInputVariables);

– s – кількість вихідних змінних (numberOfOutputVariables);

– k – загальна кількість підумов в базі правил (numberOfConditions);

– q – загальне число висновків в базі правил (numberOfConclusions).

Другий етап – фазифікація вхідних змінних. Цей етап часто називають приведенням до нечіткості. На вхід надходять сформована база правил і масив вхідних даних $A = \{a_1, ..., a_m\}$. У цьому масиві містяться значення всіх вхідних змінних. Метою цього етапу є отримання значень істинності для всіх підумови з бази правил. Це відбувається так: для кожної з підумов знаходиться значення $b_i = \mu(a_i)$. Таким чином виходить безліч значень b_i ($i = 1 .. k$).

Наступними етапами є агрегація підумов, активізація та акумуляція висновків. Метою агрегації підумов є визначення ступеня істинності умов для кожного правила системи нечіткого виведення. Спрощено кажучи, для кожної умови знаходимо мінімальне значення істинності всіх його підумов. Формально це виглядає так:

$$c_j = \min \{ b_i \},$$

де $j = 1..n$;

i – число з безлічі номерів підумови в яких бере участь j–а вхідна змінна.

На етапах активізації та акумуляції висновків відбувається перехід від умов до підумов. Для кожного підвисновку знаходиться ступінь істинності $d_i = c_i * F_i$, де $i = 1..q$. Потім, знову ж кожному i–му підвисновку, зіставляється безліч D_i з новою функцією приналежності. Її значення визначається як мінімум з d_i і значення функції належності терму з підвисновку [26].

Метою цього етапу є отримання нечіткої множини (або їх об'єднання) для кожної з вихідних змінних. Виконується він наступним чином: i–ої вихідної змінної зіставляється об'єднання множин $E_i = \cup D_j$. Де j– номери підвисновків в яких бере участь i–та вихідна змінна ($i = 1..s$). Об'єднанням двох нечітких множин є третя нечітка множина з наступною функцією приналежності:

$$\mu'_i(x) = \max\{\mu_1(x), \mu_2(x)\}, \text{ де } \mu_1(x), \mu_2(x).$$

Останнім етапом є дефазифікація вихідних змінних. Мета дефазифікація отримати кількісне значення (crisp value) для кожної з вихідних лінгвістичних змінних. Формально, це відбувається в такий спосіб.

Розглядається i–а вихідна змінна і відноситься до неї безліч E_i ($i = 1..s$). Потім за допомогою методу дефазифікації знаходиться підсумкове кількісне значення вихідної змінної [20].

У 1995 році Кастро (Castro) показав, що при симетричних трикутних функціях відповідності алгоритм Мамдані універсальний і може бути застосованим до будь якої галузі науки чи промисловості, а отже використовувати його у дослідженнях доволі раціональне і обгрунтоване рішення.

2.2 Алгоритм Сугено

У цьому підрозділі розглядається так званий Сугено, або Такагі–Сугено–Кан, метод нечіткого умовиводу. Введений у 1985 р. [27], він подібний до Методу Мамдані багато в чому. Перші дві частини нечіткий процес виводу, нечіткі входи та застосування нечіткого оператора, точно такі ж. Основна відмінність Мамдані від Сугено полягає в тому, що функції виведення членства Sugeno або лінійні, або постійні.

Типове правило в нечіткій моделі Сугено має такий вигляд:

– якщо Input 1 = x і Input 2 = y, то вихідним є $z = ax + c$.

Для моделі Sugeno нульового порядку вихідна змінна z є постійною ($a = b = 0$).

Рівень виходу z і кожного правила зважується на міцність випалу w–правила. Наприклад, для правила AND з Input 1 = x та Input 2 = y, керуючим процесом є w_i = AndMethod (F1(x), F2(y)).

Правило Сугено діє, як показано на наступній схемі (рисунок 2.5).

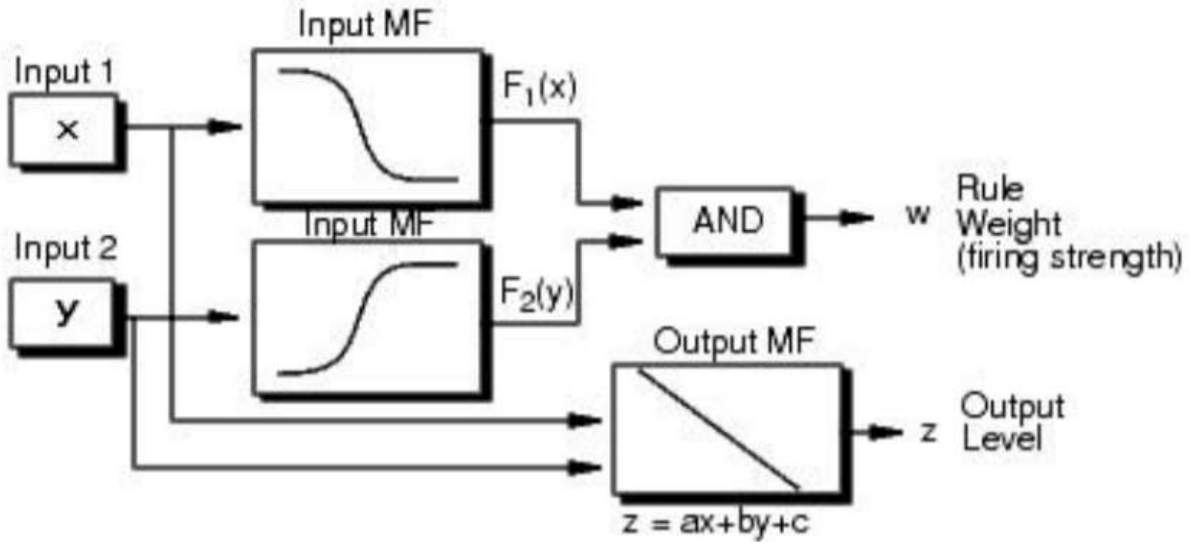

Рисунок 2.5 – Схема роботи алгоритму Сугено

Нечіткі моделі Сугено вищого порядку можливі, але вони вводять

значну складність з неочевидними перевагами. Через лінійну залежність кожного правила від вхідних змінних, метод Сугено ідеально підходить для роботи в якості інтерполяційного керівника декілька лінійних контролерів, які необхідно застосувати відповідно до різних умови роботи динамічної нелінійної системи.

Система нечітких висновків Сугено надзвичайно добре підходить до завдання плавної інтерполяції лінійних посилень, які будуть застосовані через вхідний простір. Це природний та ефективний планувальник рішень. Аналогічно, система Сугено підходить для моделювання нелінійних систем шляхом інтерполяції між декількома лінійними моделями. Акумуляція висновків нечітких правил продукцій фактично відсутня, оскільки розрахунки здійснюються із звичайними дійсними числами [28].

Дефазифікації вихідних змінних відбувається наступним чином – використовується модифікований варіант у формі методу центру тяжіння для одноточкових множин.

Вважається, що алгоритм Сугено точніше Мамдані, але він іноді важкий в реалізації. Структура системи нечіткого виводу за алгоритмом Сугено представлена на рисунку 2.6.

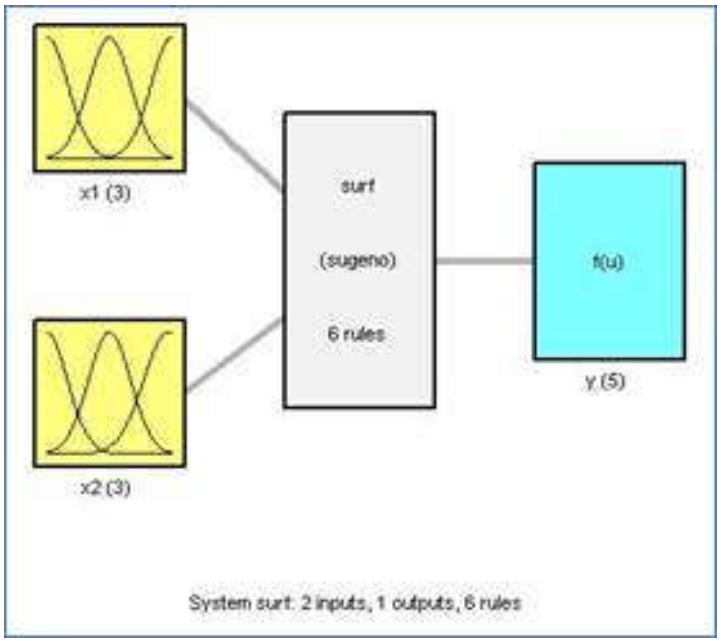

Рисунок 2.6 – Структура системи нечіткого виводу

Важливим етапом при реалізації алгоритму Сугено є фазифікація вхідних змінних. Фазифікація виконується однаково у всіх методах нечіткого виведення. Фазифікацію ще називають введенням нечіткості. Метою етапу фазифікації є встановлення відповідності між конкретним (зазвичай – чисельним) значенням окремої вхідної змінної системи нечіткого виведення і значенням функції приналежності відповідного їй терма вхідної лінгвістичної змінної.

Після завершення цього етапу для всіх вхідних змінних повинні бути визначені конкретні значення функцій приналежності по кожному з лінгвістичних термів, які використовуються в підумові бази правил системи нечіткого виведення [29].

Даний алгоритм відповідає загальній схемі системи нечіткого управління, проте значно відрізняється від алгоритму Мамдані, і в першу чергу, конфігурацією правил, що представляють базу знань. Правила мають гібридну форму: передумови правил аналогічні тим, що представлені формулою в алгоритмі Мамдані, і відповідно передбачають нечіткість вхідних змінних, в той час як вихідні змінні представлені не у вигляді нечітких множин, а деякою функціональною залежністю від вхідних змінних.

Переваги алгоритму Сугено полягають в меншій трудомісткості проведення розрахунків на його основі, а також в здатності моделювати дуже складні системи, адекватний опис яких за допомогою схеми Мамдані практично неможливо через вкрай великої кількості формуються взаємозв'язків між нечіткими параметрами. До слабких сторін алгоритму Сугено можна віднести те, що він не дозволяє представляти вихідні змінні в лінгвістичної формі, а настройка параметрів функцій приналежності є складним завданням нелінійного програмування, для вирішення якої, однак, існують ефективні методи, наприклад спираються на апарат штучних нейронних мереж.

Наступним важливим етапом в алгоритмі Сугено є агрегація підумови в нечітких правилах продукцій. Для знаходження ступеня істинності умов

всіх правил нечітких продукцій, як правило, використовується логічна операція min–кон'юнкції. Ті правила, ступінь істинності умов яких відмінна від нуля, вважаються активними і використовуються для подальших розрахунків [30].

Активізація підвисновків в нечітких правилах продукцій є наступним етапом. По-перше, використанням методу знаходяться значення ступенів істинності всіх висновків правил нечітких продукцій. По-друге, здійснюється розрахунок звичайних (не нечітких) значень вихідних змінних кожного правила.

Це виконується з використанням формули для висновку, в яку підставляються значення вхідних змінних до етапу фазифікації. Тим самим визначаються безліч значень і безліч значень вихідних змінних – загальна кількість правил в базі правил.

Акумуляція висновків нечітких правил продукцій фактично відсутня, оскільки розрахунки здійснюються із звичайними дійсними числами.

Дефазифікації вихідних змінних відбувається наступним чином – використовується модифікований варіант у формі методу центру тяжіння для одноточкових множин.

Загалом, обидва алгоритми мають багато спільного але водночас присутні ключові відмінності, які і дозволяють розподіляти їх галузі застосування. Науковці використовують дані алгоритми як основи своїх досліджень та будують на їх основі комплексні системи з великою кількістю зв’язків та відповідних правил. Переваги обох алгоритмів показано у таблиці 2.1.

Нечітка система виводу Sugeno підходить до задачі плавного інтерполяції лінійних коефіцієнтів посилення, які застосовуватимуться у вхідному просторі. Це природний та ефективний планувальник вхідних даних.

Таблиця 2.1 – Переваги розглянутих алгоритмів

Алгоритм нечіткого висновку	Переваги
Мамдані	– інтуїтивно зрозумілий – добре підходить для ручного вводу – більш інтерпретована база правил – має широке визнання
Сугено	– обчислювально ефективний – зручно працювати з лінійними методами, такими як PID–контроль – зручно працювати з оптимізацією та адаптаційними методами – гарантування безперервності вихідної поверхні – добре підходить для математичного аналізу

Отже, через лінійну залежність кожного правила від вхідних змінних, метод Сугено ідеально підходить для роботи в якості інтерполяційного супервізора декількох лінійних контролерів, які повинні застосовуватися відповідно до різних умов роботи динамічної нелінійної системи. Наприклад, продуктивність літака може різко змінитися з висотою. Лінійні контролери, хоча їх легко обчислити і підходять до будь–якого стану польоту, повинні регулярно та плавно оновлюватися, щоб бути в курсі мінливого стану польотного транспортного засобу [31].

На противагу алгоритму Сугено, в алгоритмі Мамдані вихід кожного правила є нечітким набором, отриманим від функції виведення членства та методу імплікації FIS. Ці вихідні нечіткі множини об'єднуються в єдиний нечіткий набір, використовуючи метод агрегування FIS. Потім, щоб обчислити кінцеве чітке значення вихідного сигналу, комбінований нечіткий набір виходу дефазифікується за допомогою одного з методів, описаних у методах дефазифікації [32].

2.3 Загальна схема нечіткого алгоритму аналізу стану комп’ютерної системи

Схема нечіткого алгоритму аналізу стану комп’ютерної системи буде розроблена за допомогою вже відомого методу нечіткого виводу Мамдані. Отож, для розробки нечіткої системи, що діагностує комп’ютерну систему потрібні певні вхідні дані, а також нечіткі правила, за допомогою яких і буде проводитись діагностика [33].

База правил системи нечіткого виводу призначена для формального подання емпіричних знань або знань експертів в тій чи іншій проблемній області. У системах нечіткого виведення використовуються правила нечітких продукцій, в яких умови і укладення сформульовані в термінах нечітких лінгвістичних висловлювань. База правил нечітких продукцій – це кінцева множина правил нечітких продукцій, узгоджених щодо використовуваних в них лінгвістичних змінних. Найчастіше база правил представляється в формі певного структурованого тексту:

– правило 1: якщо «умова_1, то висновок_1», або ж, до прикладу, якщо температура центрального процесора (ЦП) наближена до 100 градусів за Цельсієм, то стан ЦП критичний і потребує втручань з боку користувача.

Для того, щоб перевести вхідний параметр в нечітку область використовується фазифікація, яка в свою чергу керується функціями приналежності M_x. Вона показує ступінь приналежності параметра до одного з нечітких значень. Чим більша ступінь приналежності, тим більша ймовірність, що обчислювальна машина присвоїть змінній відповідне нечітке значення [34]. Однак не варто плутати функцію приналежності з функцією імовірнісного розподілу. Тому, зокрема, сума ступенів належності одного вхідного параметра до різних нечітких значень не обов'язково дорівнює 1.

Графік функції належності показано на рисунку 2.7.

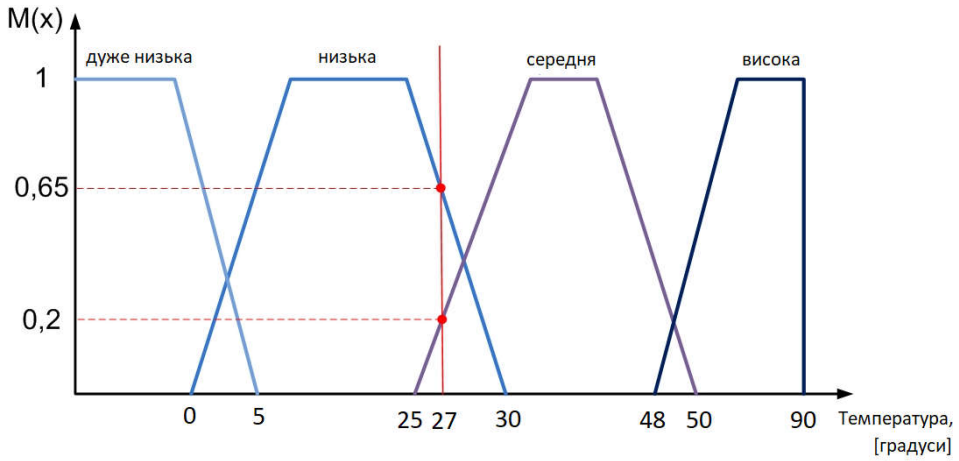

Рисунок 2.7 – Графік функції належності

Точно такі ж функції приналежності потрібно визначити і для інших вхідних і вихідних параметрів системи. Як тільки система управління фазифікує всі вхідні параметри по заданих функціях приналежності, блок прийняття рішення знайде відповідні значення вихідних параметрів, користуючись нечіткими правилами, що були складені попередньо [35].

На етапі дефазифікації система управління буде робити зворотне перетворення з нечітких значень вихідних параметрів (знайдених по таблиці) до точних цифр. Математичні алгоритми цих перетворень різноманітні і залежать від конкретного завдання. В нашому випадку алгоритм роботи вже визначений і формули, які потрібні для роботи також.

Як висновок, можна сказати, що нечіткі системи справді є дуже зручними для моделювання певних складних систем пов’язаних з ЕОМ. Вони мають велику кількість переваг перед іншими, серед яких:

– можливість оперувати вхідними даними, заданими нечітко: наприклад, такі що безупинно змінюються в часі (динамічні задачі), значення, що неможливо задати однозначно (результати статистичних опитувань, рекламні компанії);

– можливість нечіткої формалізації критеріїв оцінки і порівняння: оперування критеріями "більшість", "можливо", переважно";

– можливість проведення якісних оцінок як вхідних даних, так і виведених результатів;

– можливість проведення швидкого моделювання складних динамічних систем і їх порівняльний аналіз із заданим ступенем точності: оперуючи принципами поведінки системи, описаними fuzzy–методами, по–перше, не витрачається багато часу на з'ясування точних значень змінних і складання рівнянь, що їх описують, по–друге, можна оцінити різні варіанти вихідних значень [36].

Для кращого розуміння роботи розробленого алгоритму та відповідної системи було створено структурну схему, котра показує ключові елементи розробки та їх взаємодію між собою (рисунок 2.8).

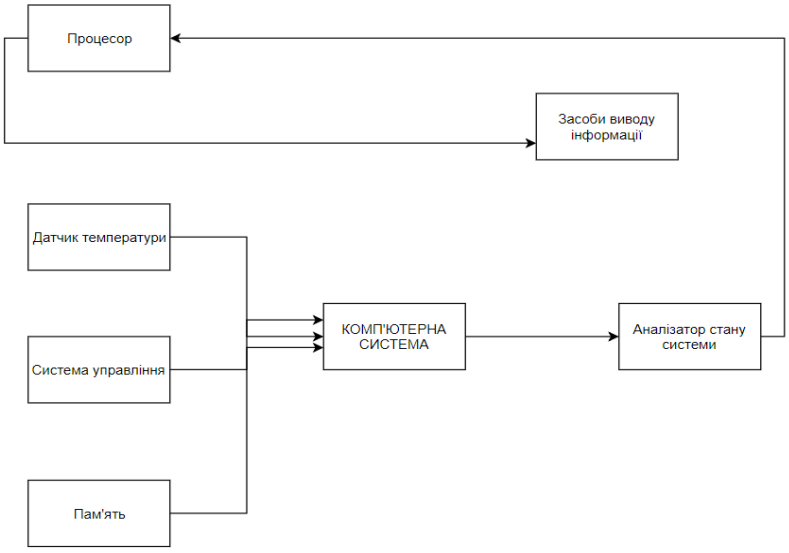

Рисунок 2.8 – Структурна схема розробки

Алгоритм – це система формальних правил, розташованих в визначеному логічному порядку, які чітко й однозначно визначають процес виконання заданої роботи незалежно від її характеру та походження.

Перед початком розробки алгоритму необхідно чітко уяснити, що програма повинна робити, яка інформація потрібна програмі (які дані є в

наявності та які існують обмеження на ці дані), які обчислення та інші дії програма повинна виконати та яку інформацію видати користувачу в якості результату роботи.

Після цього треба вирішити, як програма буде це робити. Яким буде інтерфейс користувача, як повинна бути побудована програма. Необхідно також вирішити, як будуть представлені дані в програмі та які методи будуть використовуватись для обробки даних, щоб отримати остаточний результат.

Алгоритм, як правило, будується у декілька етапів – спочатку він формулюється у загальних рисах, а потім уточнюється шляхом заміни складних дій більш простими. Цей метод називається методом покрокової деталізації або методом проектування «зверху вниз» [37].

При розробці алгоритму треба враховувати ресурсні обмеження щодо розв'язування задачі, наявність готових програм для реалізації відомих методів та алгоритмів, а також необхідно прагнути того, щоб алгоритм був більш універсальним, тобто придатним для широкого класу вхідних даних [22].

При графічному представленні алгоритм зображується у вигляді послідовності зв'язаних між собою функціональних блоків, кожний з яких відповідає виконанню однієї або декількох дій. Таке графічне представлення називається схемою алгоритму або блок–схемою.

У блок–схемі кожному типу дій (введенню–виведенню даних, обчисленню значень виразів, перевірці умов, управлінню повторенням дій, закінченню обробки и т.ін.) відповідає один або декілька блочних символів у вигляді плоских геометричних фігур, в середині яких розміщується текст або формула, що пояснює дії, які виконуються. Блочні символи з'єднуються лініями переходів, що визначають порядок виконання дій. Саме за допомогою блок–схеми представлений розроблений нечіткий алгоритм аналізу стану комп'ютерної системи (рисунок 2.9), оскільки таке рішення є простим і водночас інформативним та зрозумілим.

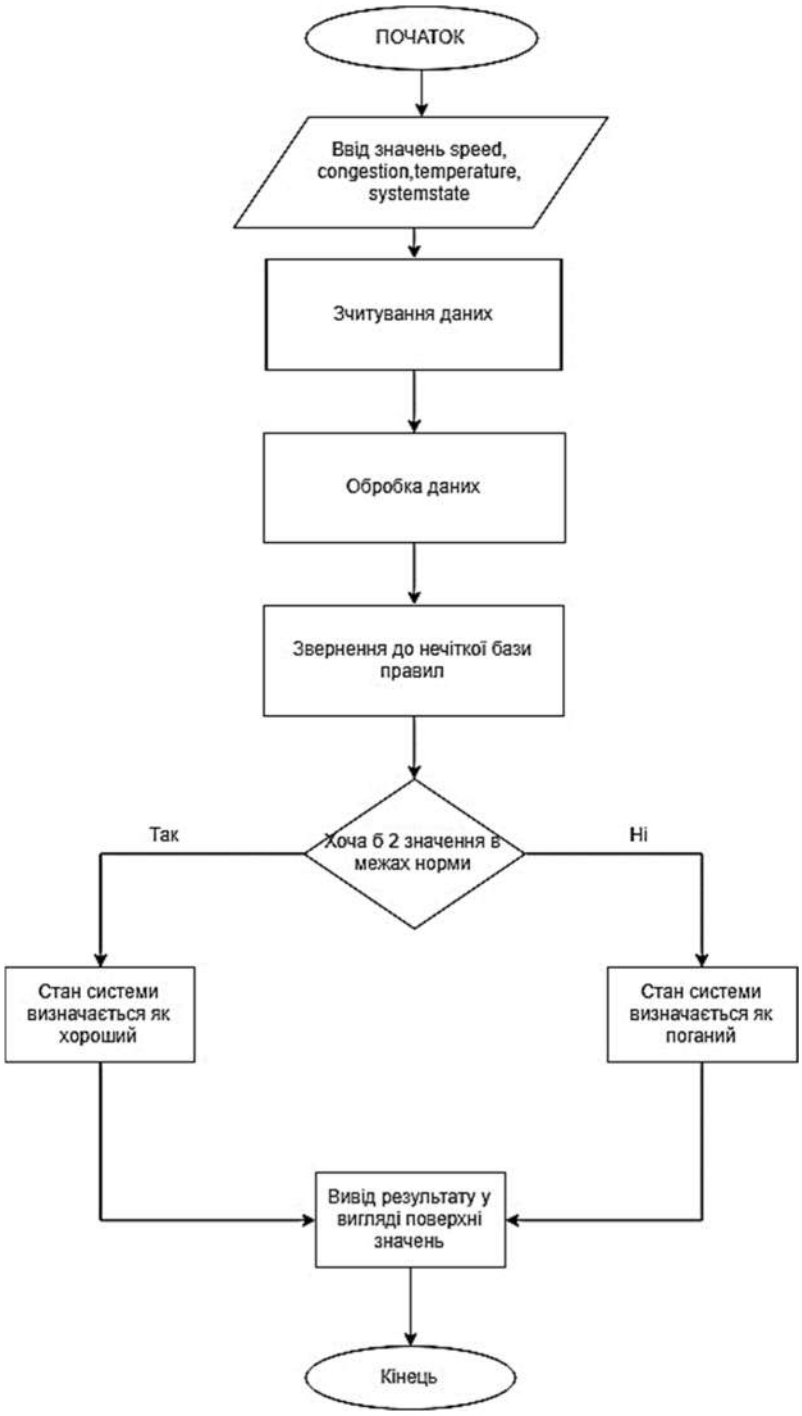

Рисунок 2.9 – Розроблений нечіткий алгоритм аналізу стану комп’ютерної системи

Найбільш поширені форми представлення алгоритмів: на звичайній мові, за допомогою псевдокоду (напівформалізований опис алгоритму), графічна форма та програмна (текст на алгоритмічній мові програмування).

Проаналізувавши ключові елементи розробленого алгоритму, можна зробити висновок, що робота алгоритму організована логічно та послідовно.

Враховані всі значимі операції, що і дозволяють реалізувати розроблену систему.

2.4 Симуляція роботи розробленого нечіткого алгоритму

Для побудови нечіткої системи на основі розробленого алгоритму потрібно для початку задати вхідні дані, а також кінцевий результат. На вході було задано наступні значення:

- температура;
- швидкодія;
- навантаженість.

На виході отримано загальний стан системи, який напряму залежить від значень вхідних даних. База правил системи нечіткого виводу призначена для формального подання емпіричних знань або знань експертів в тій чи іншій проблемній області. У системах нечіткого виведення використовуються правила нечітких продукцій, в яких умови і укладення сформульовані в термінах нечітких лінгвістичних висловлювань. База правил нечітких продукцій – це кінцева множина правил нечітких продукцій, узгоджених щодо використовуваних в них лінгвістичних змінних. Найчастіше база правил представляється в формі певного структурованого тексту. На рисунку 2.10 показано загальний вигляд розробленої нечіткої системи.

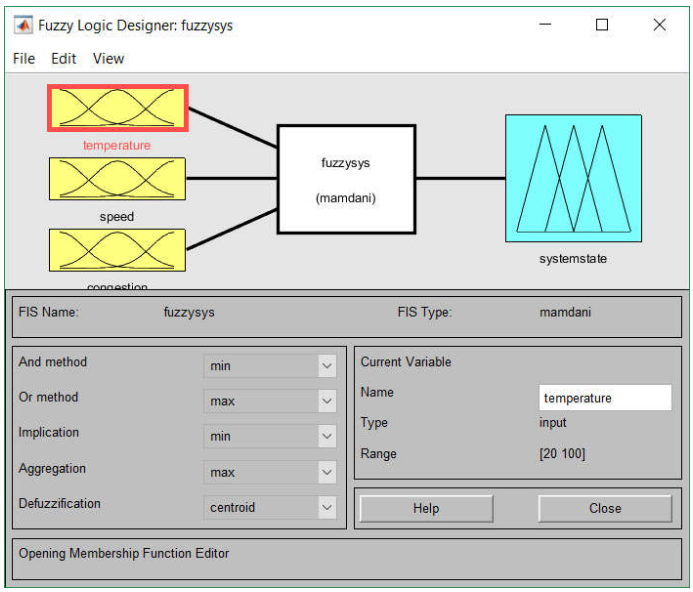

Рисунок 2.10 – Входи розробленої системи

Далі для кожного з входів, а також для виходу необхідно створити функції належності. Для входів було обрану функції типу “gbellmf”, адже вони дозволяють краще візуально показати зміну значень. Для виходу застосовуються функції типу “trimf”. Для подальшої роботи потрібно визначитись в яких межах значень будуть коливатись вхідні та вихідні дані. Проаналізувавши роботу комп’ютерних систем для входів було призначено межі, а саме для температури від 20˚С до 100˚С:

- до 35˚С – температура в нормі;
- до 70˚С – температура підвищена;
- більше 70˚С – висока температура.

Для швидкодії від 0 до 1:

- від 0 до 0,5 – висока швидкодія;
- від 0,4 до 0,7 – нормальна швидкодія;
- від 0,6 до 1 – низька швидкодія.

Для навантаженості від 0 до 200:

- від 0 до 80 – низька навантаженість;
- від 70 до 140 – середня навантаженість;
- від 130 до 200 – висока навантаженість.

Для загального стану системи було умовно встановлено шкалу від 0

до 10:

- від 0 до 3 – відмінний стан;
- від 3 до 7 – нормальний стан;
- від 7 до 10 – поганий стан.

Функції приналежності температури, швидкодії, навантаженості, а також кінцевого стану системи зображені відповідно на рисунках 2.11 – 2.14.

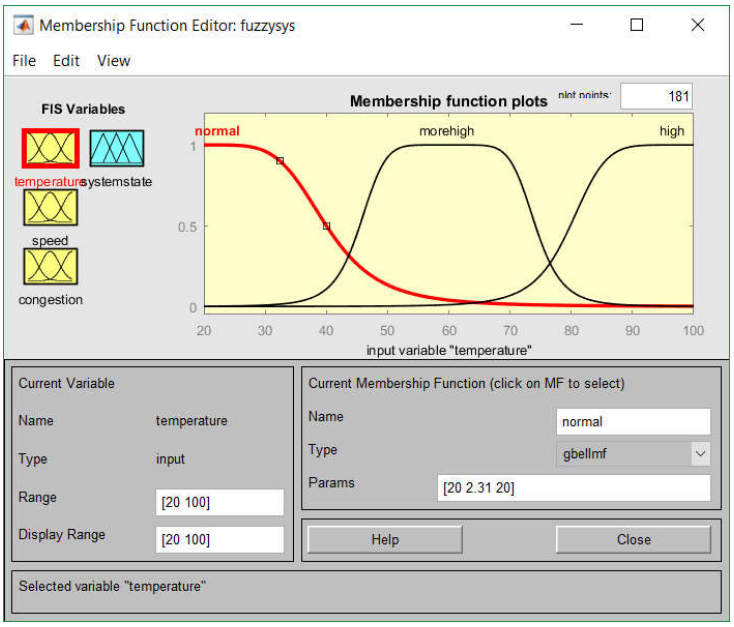

Рисунок 2.11 – Функції приналежності температури

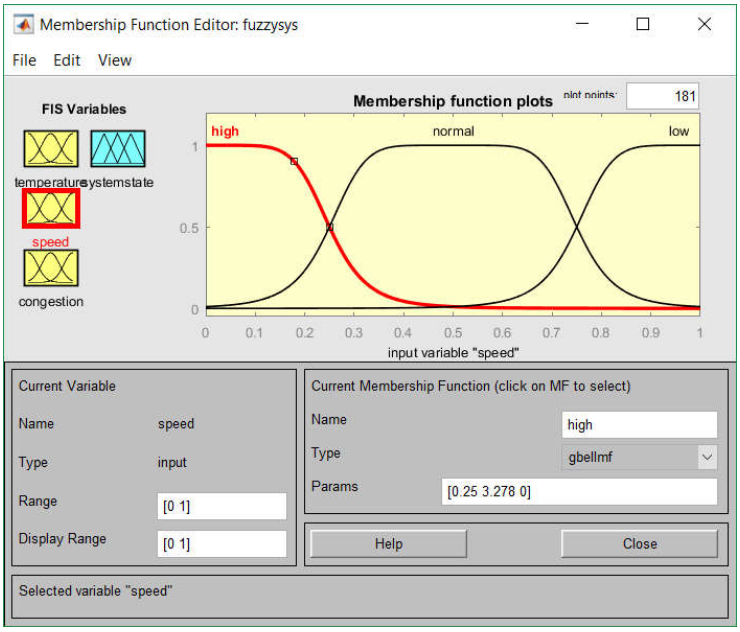

Рисунок 2.12 – Функції приналежності швидкодії

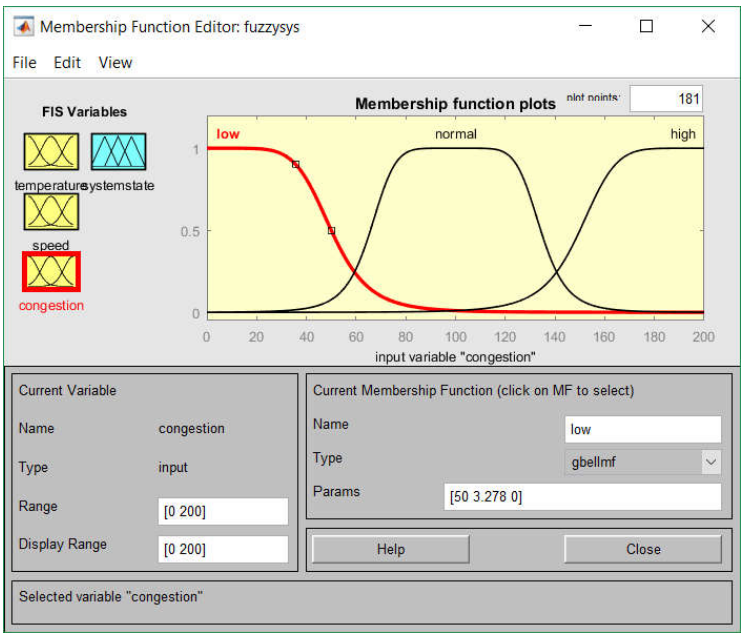

Рисунок 2.13 – Функції приналежності навантаженості

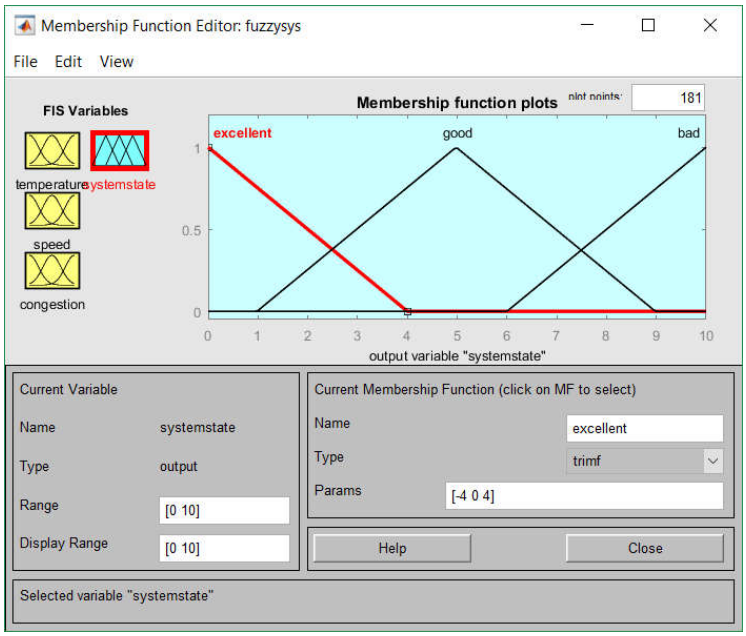

Рисунок 2.14 – Функції приналежності стану системи

Для функцій приналежності також була розроблена база нечітких правил. Нечіткою базою правил називається сукупність нечітких правил типу "якщо – то", що визначають взаємозв'язок між входами і виходами досліджуваного об'єкта.

База правил систем нечіткого виводу призначена для формального подання емпіричних знань або знань експертів в тій чи іншій проблемній

області. У системах нечіткого виведення використовуються правила нечітких продукцій, в яких умови і укладення сформульовані в термінах нечітких лінгвістичних висловлювань [38].

Усі вхідні змінні мають по три нечітких стани і ще один стан none, коли значення вхідної змінної не задане системою. Випадок, коли значення усіх вхідних змінних не задані, на практиці неможливий.

Для підтвердження працездатності нечіткої системи на рисунках 2.15 та 2.16 подані поверхні значень.

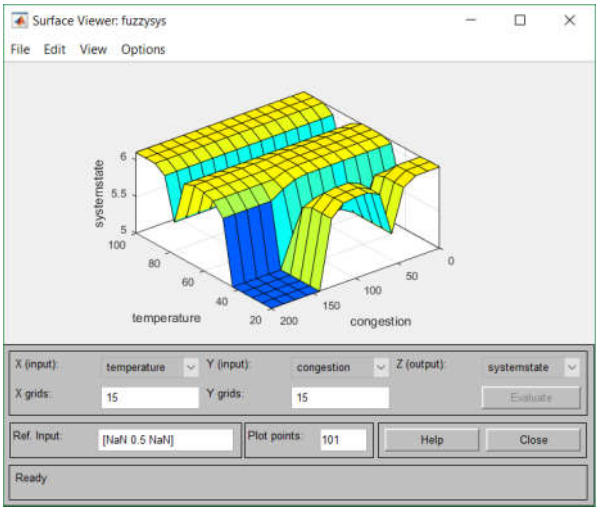

Рисунок 2.15 – Поверхня значень стану системи (systemstate) залежно від відношення температури(temperature) до навантаженості (congestion)

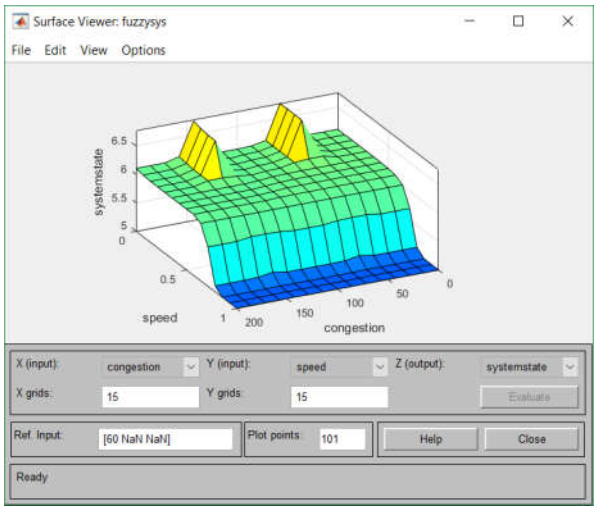

Рисунок 2.16 – Поверхня значень стану системи (systemstate) залежно від відношення швидкодії(speed) до навантаженості (congestion)

Нечіткий вивід моделі вибору стану комп'ютерної системи, побудованого на основі заданих 63 правил з поточними значеннями змінних temperature, speed, congestion та systemstate має вигляд, представлений на рисунку 2.17.

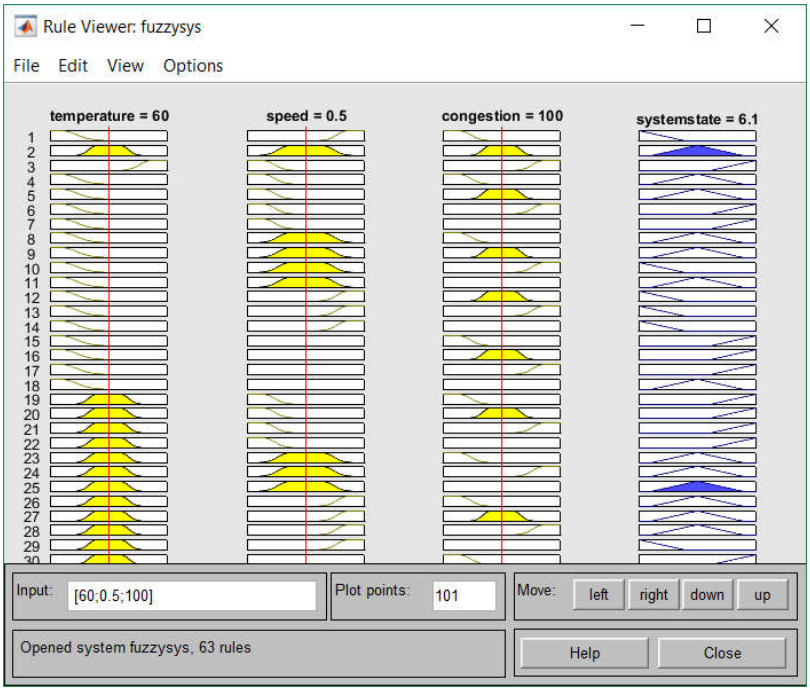

Рисунок 2.17 – Графічне зображення правил нечіткої системи

Також для підтвердження правильності роботи створеної нечіткої системи, представлено таблицю даних з результатами моделювання (таблиця 2.2).

Таблиця 2.2 – Результати моделювання нечіткої системи

Вхідні значення			Вихідне значення systemstate
temperature	Speed	congestion	
31.6	0.28	41.8	5.5
76.5	0.83	157	7
0	0.4	40	3.2
85	0	140	5.3

Продовження таблиці 2.2

Вхідні значення			Вихідне значення systemstate
temperature	Speed	congestion	
75	0.2	47	8
15	0.75	47	2.2
30.4	0.48	96	4.6
27.9	0.61	115.3	6.31
12	0.51	100.9	5.2
35	0.5	134.4	4.7

Отже, в результаті верифікації розробленої нечіткої системи, було переглянуто всі вхідні та вихідні дані і зроблено висновок, що система працює коректно, відповідно до встановлених вимог, а також до розробленої бази нечітких правил та відповідного розробленого алгоритму.

2.5 Висновки до розділу 2

У другому розділі описано два найвідоміших та найбільш використовуваних алгоритми нечіткої логіки: Мамдані та Сугено. А саме, розглянуто історію виникнення, винахідників, основні принципи та поняття, математичну основу, правила побудови відповідних нечітких систем, ключові відмінності та спільні функціональні можливості [39].

На основі відомостей та даних про два вище згаданих алгоритми, було розроблено структурну схему та блок–схему власного алгоритму, спеціалізованого під завдання для дипломної роботи. Охарактеризовано його фундаментальні поняття, принципи та методи роботи. Алгоритм є логічним та послідовним, а отже дозволяє легко реалізувати нечітку систему на його

основі.

Також було описано вже реалізовану нечітку систему на основі розробленого алгоритму. Висвітлено середовище розробки, вхідні та вихідні дані, а також розроблену на їх основі базу нечітких правил. Детально охарактеризовано використані типи функцій належності для вхідних та вихідних даних та основні функціональні процеси. Також описано результати моделювання за допомогою поверхонь значень та нечіткого виводу системи, які дають зрозуміти чи працює система коректно та відповідає початковій ідеї.

3 НЕЧІТКИЙ КОНТРОЛЕР АНАЛІЗУ СТАНУ КОМП'ЮТЕРНОЇ СИСТЕМИ НА ОСНОВІ РОЗРОБЛЕНОГО АЛГОРИТМУ

3.1 Середовище реалізації Matlab Simulink

Для створення будь–якої нечіткої системи логічними є використання програмного середовища Matlab. Matlab – це високопродуктивна мова для технічних обчислень. Вона об'єднує обчислення, візуалізацію та програмування в простому середовищі, де проблеми та рішення виражаються у знайомому математичному позначенні [40]. Типовими варіантами використання Matlab є:

- математика та обчислення;
- розробка алгоритмів;
- моделювання та прототипування;
- аналіз даних, дослідження та візуалізація;
- наукова та інженерна графіка;
- розробка додатків, включаючи створення графічного інтерфейсу користувача.

Matlab – це інтерактивна система, основним елементом якої є масив, який не вимагає розмірів. Це дозволяє вирішити багато технічних обчислювальних завдань, особливо тих, що мають матричні та векторні формулювання, на частку часу, необхідного для написання програми в скалярній неінтерактивній мові, такій як C або Fortran [41].

Назва Matlab означає матричну лабораторію. Спочатку Matlab був написаний, щоб забезпечити легкий доступ до матричного програмного забезпечення, розробленого проектами Linpack та Eispack, які разом представляють найсучасніші рішення у програмному забезпеченні для матричних обчислень.

Matlab розвивався протягом декількох років за допомогою численних користувачів. У середовищі університетів це стандартний навчальний

інструмент для просунутих курсів з математики, техніки та інших дисциплін. У промисловості Matlab є основним інструментом для розробки рішень для високопродуктивних досліджень та аналізу [42].

Matlab – це сімейство рішень для конкретних програм, які називаються наборами інструментів. Дуже важливим для більшості користувачів Matlab, є те що панелі інструментів дозволяють вивчати та застосовувати спеціалізовані технології. Панелі інструментів – це всеосяжні набори функцій Matlab (M–файли), які розширюють середовище MATLAB для вирішення конкретних класів проблем. Області, в яких доступні набори інструментів, включають обробку сигналів, системи управління, нейронні мережі, нечітку логіку, симуляцію та багато інших. На рисунку 3.1 представлений типовий інтерфейс програмного середовища Matlab.

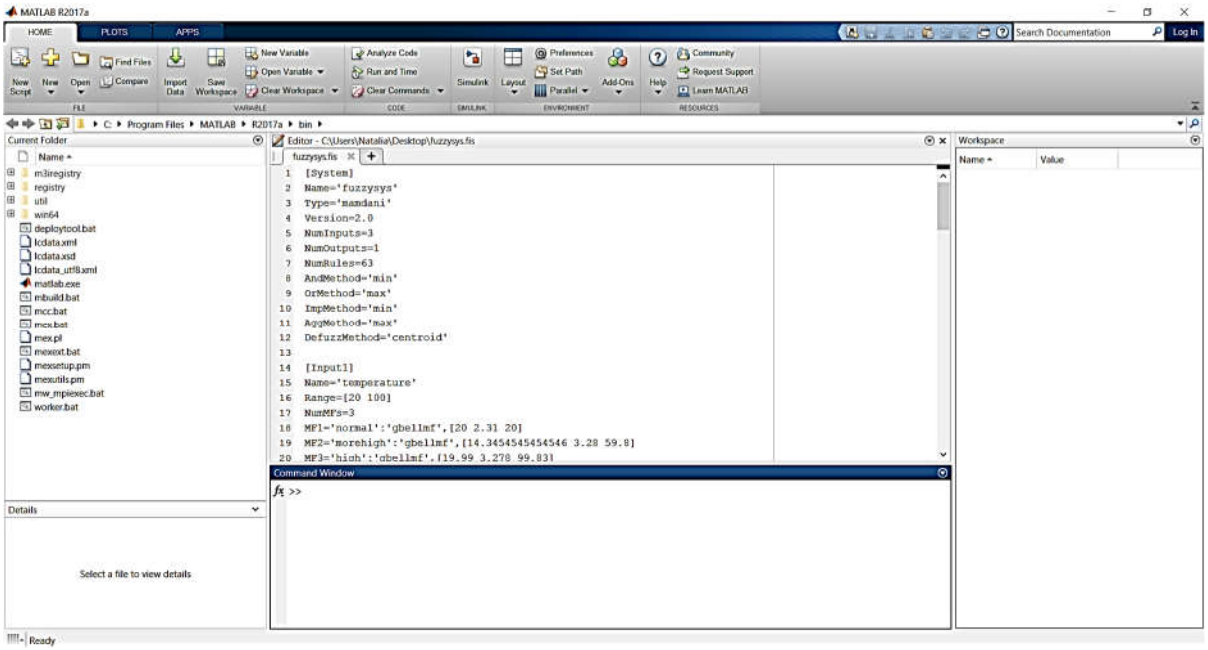

Рисунок 3.1 – Інтерфейс Matlab

Для Matlab є можливість створювати спеціальні набори інструментів (англ. Toolbox), що розширюють його функціональність. Набори інструментів – це набори функцій, написаних на мові Matlab для вирішення певного класу задач [43]. Компанія Mathworks поставляє набори інструментів, які використовуються в багатьох областях, включаючи наступні:

– цифрова обробка сигналів , зображень та даних (DSP Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communication Toolbox, Filter Design Toolbox – набори функцій, що дозволяють вирішувати широкий спектр завдань обробки сигналів, зображень, проектування цифрових фільтрів і систем зв'язку);

– системи управління (Control Systems Toolbox, μ-Analysis and Synthesis Toolbox, Robust Control Toolbox, System Identification Toolbox, LMI Control Toolbox, Model Predictive Control Toolbox, Model–Based Calibration Toolbox – набори функцій, що полегшують аналіз і синтез динамічних систем, проектування, моделювання та ідентифікацію систем управління, включаючи сучасні алгоритми управління, такі як H–управління , ЛМН–синтез , μ–синтез та інші);

– фінансовий аналіз (GARCH Toolbox, Fixed-Income Toolbox, Financial Time Series Toolbox, Financial Derivatives Toolbox, Financial Toolbox, Datafeed Toolbox – набори функцій, що дозволяють швидко і ефективно збирати, обробляти і передавати різну фінансову інформацію);

– аналіз і синтез географічних карт, включаючи тривимірні (Mapping Toolbox);

– збір і аналіз експериментальних даних (Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, Link for Code Composer Studio – набори функцій, що дозволяють зберігати та обробляти дані, отримані в ході експериментів, в тому числі в реальному часі. Підтримується широкий спектр наукового та інженерного вимірювального обладнання);

– візуалізація та уявлення даних (Virtual Reality Toolbox – дозволяє створювати інтерактивні світи і візуалізувати наукову інформацію за допомогою технологій віртуальної реальності і мови VRML);

– засоби розробки (Matlab Builder for COM, Matlab Builder for Excel, Matlab Builder for NET, Matlab Compiler, Filter Design HDL Coder – набори функцій, що дозволяють створювати незалежні програми з середовища Matlab);

– взаємодія з зовнішніми програмними продуктами (Matlab Report Generator, Excel Link, Database Toolbox, Matlab Web Server, Link for ModelSim – набори функцій, що дозволяють зберігати дані різних видів таким чином, щоб інші програми могли з ними працювати);

– бази даних (Database Toolbox – інструменти роботи з базами даних);

– наукові та математичні пакети (Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed–Point Toolbox, Fuzzy Logic Toolbox, Genetic Algorithm and Direct Search Toolbox, OPC Toolbox, Optimization Toolbox, Partial Differential Equation Toolbox, Spline Toolbox, Statistic Toolbox, RF Toolbox – набори спеціалізованих математичних функцій, що дозволяють вирішувати широкий спектр наукових та інженерних задач, включаючи розробку генетичних алгоритмів, вирішення завдань в приватних похідних, цілочисельні проблеми, оптимізацію систем та інші);

– нейронні мережі (Neural Network Toolbox – інструменти для синтезу та аналізу нейронних мереж);

– нечітка логіка (Fuzzy Logic Toolbox – інструменти для побудови та аналізу нечітких множин);

– символьні обчислення (Symbolic Math Toolbox – інструменти для символьних обчислень з можливістю взаємодії з символьним процесором програми Maple).

Можна зробити висновок, що Matlab є програмним середовищем, яке вирішує величезний спектр проблем та допомагає реалізувати найрізноманітніші наукові рішення та ідеї. Є можливість підібрати будь–який набір функції, що буде краще підходити для проектного рішення, та водночас дасть можливість реалізувати його найбільш простим та зрозумілим способом, не витрачаючи багато часу та ресурсів [13]. За допомогою програмного середовища Matlab можна вирішити всі сучасні задачі, адже з такою великою кількістю наборів функціональних пакетів практично не залишається сфери чи наукової області, де не був би потрібен Matlab.

Розробка нечіткої системи в програмному середовищі Matlab

виконується за допомогою вище описаного набору функції, що має назву Fuzzy Logic Toolbox. Fuzzy Logic Toolbox надає функції, додатки і Simulink – блоки для аналізу, проектування і моделювання систем на основі нечіткої логіки. Даний набір функцій в основному працює методами розробки виводів для нечітких систем. Функції надаються для багатьох загальних методів, включаючи нечіткі кластеризації та адаптивне нейрофузійне навчання. До ключових особливостей Fuzzy Logic Toolbox можна віднести наступні:

– програма Fuzzy Logic Design для побудови нечітких систем виведення та перегляду та аналізу результатів;

– функції належності для створення систем нечітких виводів;

– підтримка AND, OR, та NOT логіки в правилах, визначених користувачем;

– стандартні системи нечітких виводів Mamdani та Sugeno;

– формування автоматизованої функції належності за допомогою нейроадаптивних та нечітких технологій кластеризації;

– можливість вставляти нечіткі системи виводів в модель Simulink;

– можливість генерування вбудованого С–коду або автономного виконуваного двигуна нечіткого виведення.

Simulink – це графічне середовище імітаційного моделювання, що дозволяє за допомогою блок–діаграм у вигляді направлених графів, будувати динамічні моделі, включаючи дискретні, безперервні і гібридні, нелінійні і розривні системи [44].

Інтерактивне середовище Simulink, дозволяє використовувати вже готові бібліотеки блоків для моделювання електросилових, механічних і гідравлічних систем, а також застосовувати розвинений модельно–орієнтований підхід при розробці систем управління, засобів цифрового зв'язку і пристроїв реального часу.

Додаткові пакети розширення Simulink дозволяють вирішувати весь спектр завдань від розробки концепції моделі до тестування, перевірки, генерації коду і апаратної реалізації. Simulink інтегрований в середу MATLAB,

Що дозволять використовувати вбудовані математичні алгоритми, потужні засоби обробки даних і наукову графіком [45].

Simulink Library Browser (Засіб перегляду Бібліотеки Simulink) містить в собі бібліотеку блоків найбільш часто використовуваних для моделювання систем.

В цю бібліотеку входять:

– блоки безперервної і дискретної динаміки, такі як Integrator (Інтегратор) і Unit Delay (Ланка Затримки);

– алгоритмічні блоки, такі як Sum (Суматор), Product (Твір), Lookup Table (Довідкова Таблиця);

– структурні блоки, такі як Mux (Мультиплексор), Switch (Перемикач), Bus Selector (Селектор Шини).

Приклад такої бібліотеки з базовими елементами для моделювання зображенона рисунку 3.2.

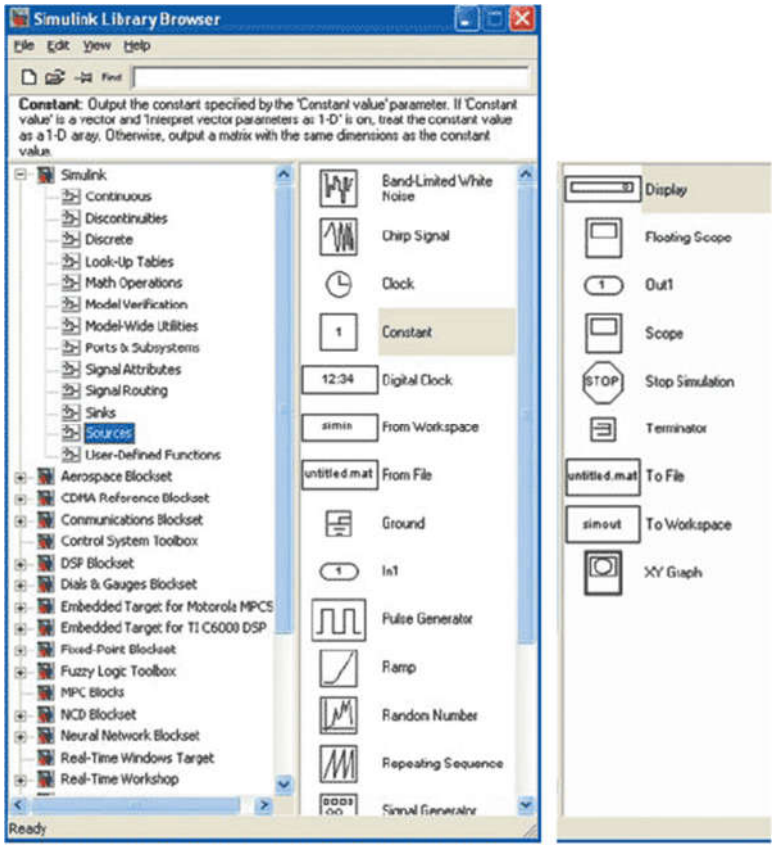

Рисунок 3.2 – Приклад бібліотеки Simulink

Можна виконувати симуляцію динамічних властивостей системи і переглядати результати, як тільки симуляція почалася.

Щоб гарантувати задану швидкість симуляції і точність, Simulink надає ODE вирішувачі з фіксованим і змінним кроком, графічний налаштовувач і підпрограму оцінки часу виконання окремих функцій моделі.

До ключових особливостей Simulink належать:

– інтерактивне графічне середовище для побудови блок–діаграм;

– розширювана бібліотека готових блоків;

– зручні засоби побудови багаторівневих ієрархічних багатокомпонентних моделей;

– засіб навігації і налаштування параметрів складних моделей – Model Explorer;

– засоби інтеграції готових C / C ++, FORTRAN, ADA і MATLAB – алгоритмів в модель, взаємодія із зовнішніми програмами для моделювання;

– сучасні засоби розв'язання диференціальних рівнянь для безперервних, дискретних, лінійних і нелінійних об'єктів (в т.ч. з гістерезисом і розривами);

– імітаційне моделювання нестаціонарних систем за допомогою перемикачів зі змінним і постійним кроком або керованим методом з пакетного моделювання MATLAB;

– зручна інтерактивна візуалізація вихідних сигналів, засоби настройки і задання вхідних впливів;

– засоби налагодження і аналізу моделей;

– повна інтеграція з MATLAB, включаючи чисельні методи, візуалізацію, аналіз даних і графічні інтерфейси.

Simulink також інтегрується з Stateflow для моделювання поведінки, викликаної подіями. Ця перевага робить Simulink найпопулярнішим інструментом для проектування систем керування і комутації, цифрової обробки і інших додатків моделювання.

3.2 Реалізація нечіткого контролера

Нечіткі контролери використовуються для управління споживчими товарами, такими як пральні машини, відеокамери, а також промисловими процесами, такими як цементні печі, поїзди метро і роботи. Нечітке управління являє собою метод управління на основі нечіткої логіки. Так само, як нечітка логіка може бути описана просто як "обчислення зі словами, а не з числами" – так і нечітке управління може бути описане просто як "управління з реченнями, а не з рівняннями".

Правила для нечіткого контролера записані в форматі if–than. Формально – if–сторона є умовою, а than–сторона є висновком. Вхідне значення "neg" є скороченням від слова Negative, вихідне значення "NB" – Negative Big, а NM – Negative Medium. Комп'ютер виконує правила і обчислення керуючого сигналу в залежності від виміряних входів помилок і змін в помилках. Важливим етапом при реалізації контролера є фазифікація вхідних змінних. Фазифікація виконується однаково у всіх методах нечіткого виведення. Фазифікацію ще називають введенням нечіткості . Метою етапу фазифікації є встановлення відповідності між конкретним (зазвичай – чисельним) значенням окремої вхідної змінної системи нечіткого виведення і значенням функції приналежності відповідного їй терма вхідної лінгвістичної змінної [46].

Після завершення цього етапу для всіх вхідних змінних повинні бути визначені конкретні значення функцій приналежності по кожному з лінгвістичних термів, які використовуються в підумові бази правил системи нечіткого виведення.

Нечіткий контролер являє собою спосіб перетворення лінгвістичної стратегії управління в автоматичну, генеруючи базу правил, яка керує поведінкою системи. Перевага використання цього лінгвістичного опису є те, що віг дозволяє легко модифікувати систему. Крім того, ніякі попередні знання про систему не використовується для розробки правил і нечітка модель

будується з даних [47].

Базовий нечіткий контролер складається з чотирьох основних компонентів:

– блок фазифікації (змінює входи, так що вони можуть інтерпретуватись і порівнюватись із правилами з бази знань);

– база знань (бази правил і бази даних, яка тримає знання, у вигляді набору правил, про те, як краще управляти системою);

– блок прийняття рішень (механізм логічного висновку, який оцінює яке правило є зараз актуальним, а потім вирішує, що має подаватись на вхід);

– блок дефазифікації (передає висновки, зроблені за допомогою механізму логічного висновку, на входи).

Блок–схема контролера зображена на рисунку 3.3. В даному випадку нечіткий контролер збирає вихідні дані y(t), порівнює їх з вхідним опорним r(t), а потім вирішує, що повинно бути поданим на вхід u(t), щоб цілі виконання були виконані.

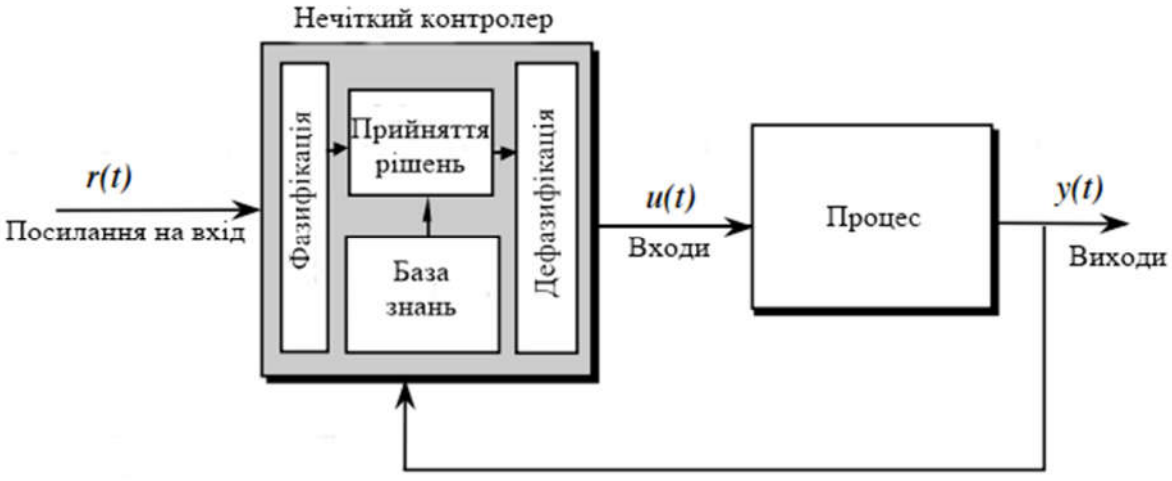

Рисунок 3.3 – Блок–схема нечіткого контролера

Нечіткі контролери використовуються в різних схемах управління. Найбільш вживаним є прямий контроль, де нечіткий контролер знаходиться в прямому каналі системи управління зворотнім зв'язком. Вихідний процес порівнюється з визначенням і якщо є відхилення, то контролер вживає заходів

відповідно до стратегії управління.

У контролерах з прогнозуванням управління вимірюване порушення компенсується. Це вимагає відповідно якісну модель, але якщо математичну модель важко або дорого отримати, то нечітка модель може реалізована із меншими затратами. На рисунку 3.4 показано контролер і нечіткий компенсатор (процес і контур зворотного зв'язку опущені для ясності). Схему, не звертаючи уваги на вхід порушення, можна розглядати як співпрацю лінійних і нелінійних управляючих дій; контролер C може бути лінійним контролером PID (пропорційний інтегрально–диференціальний), в той час як нечіткий контролер F є додатковим нелінійним контролером [48].

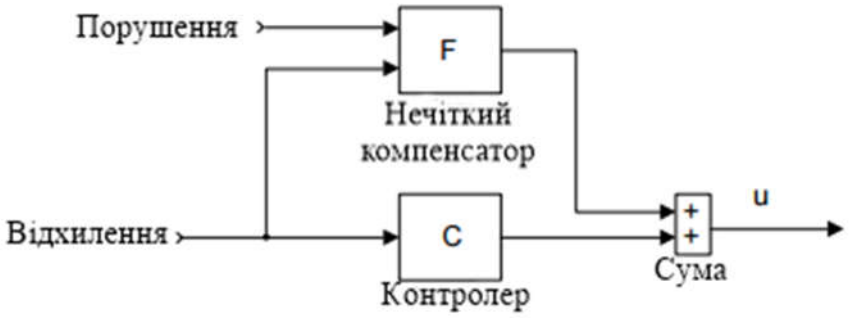

Рисунок 3.4 – Приклад контролера з нечітким компенсатором

У нечіткого контролера дані проходять через блок попередньої обробки, контролер і блок післяобробки. Попередня обробка складається з лінійного або нелінійного масштабування, а також із квантування, якщо функції приналежності дискретизуються; в протилежному випадку – набір вхідних даних може просто шукатись у відповідній функції. Післяобробка складається з масштабування вихідного сигналу, у цьому випадку контролер інкрементний. Післяобробка також включає в себе інтеграцію.

Розроблений нечіткий контролер спроектований на основі структурної схеми відповідної нечіткої системи, яке дозволяє наочно побачити зв’язки між функціональними елементами та основний принцип роботи (рисунок 3.5).

Безпосередньо контролер був розроблений за допомогою вже відомого пакету Simulink у середовищі Matlab. Для включення системи нечіткого

виведення в Simulink–модуль необхідно відкрити fuzblock командою fuzblock або через опцію Fuzzy Logic Toolbox в Simulink Library Browser. Потім вибрати блок Fuzzy Logic Controller або Fuzzy Logic Controller with Ruleviewer, зробити подвійне клацання по цьому блоку і в діалоговому вікні, що з'явилося, ввести ім'я файлу або найменування змінної з цієї області, які відповідають системі нечіткого виведення [49].

Якщо нечітка система має декілька входів, тоді в Simulink–моделі необхідно з'єднати їх разом до введення в нечіткий контролер. Аналогічно, якщо нечітка система має декілька виходів, тоді вихідні сигнали блоку будуть представлені однією мультиплексною лінією. На рисунку 3.5 зображено розроблений нечіткий контролер на основі відповідного нечіткого алгоритму та розробленої нечіткої системи:

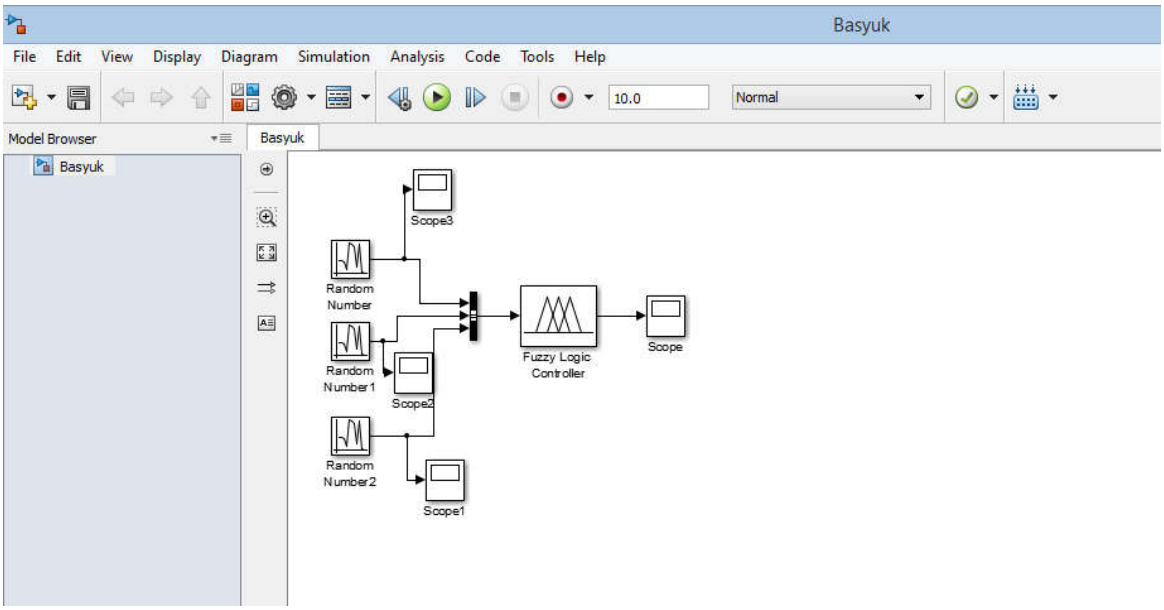

Рисунок 3.5 – Загальна схема нечіткого контролера

Як можна побачити з рисунка 3.6, контролер обробляє дані, що надходять з трьох вхідних змінних, як і було спроектовано. Їм задані відповідні числові межі, так само як і для вихідної змінної. Для більшості нечітких систем fuzblock автоматично генерує ієрархічну модель, що складається з Simulink–модулей. Автоматичний синтез моделі відбувається за допомогою FIS Wizard.

Синтезовані моделі складаються лише зі вбудованих сімулінк–модулей, тому нечітке виведення виконується дуже швидко, навіть якщо модель виходить громіздкою. FIS Wizard генерує Simulink–модулі, якщо нечітка система містить лише вбудовані функції приналежності.

Крім того, повинні використовуватися такі реалізації логічних операцій: АБО – max; ТА – min і prod; імплікація – min і агрегація – max. Якщо створити нечітку систему з Simulink–блоків не вдається, тоді використовується sffis – спеціально оптимізована під Simulink функція нечіткого виведення.

Ключовим етапом з якого починається робота нечіткого контролера є етап дефазифікації. Загальна схема роботи дефазифікатора, проведених ним перевірок та його функціональна схема зображені на рисунках 3.6, 3.7 та 3.8.

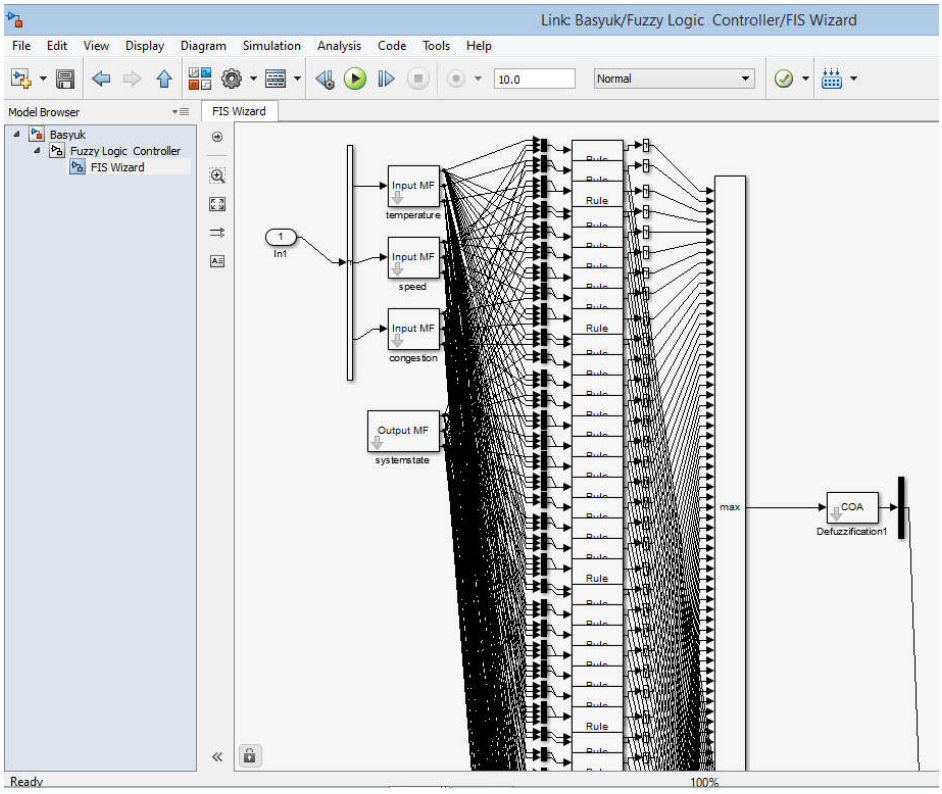

Рисунок 3.6 – Фрагмент дефазифікатора нечіткого контролера

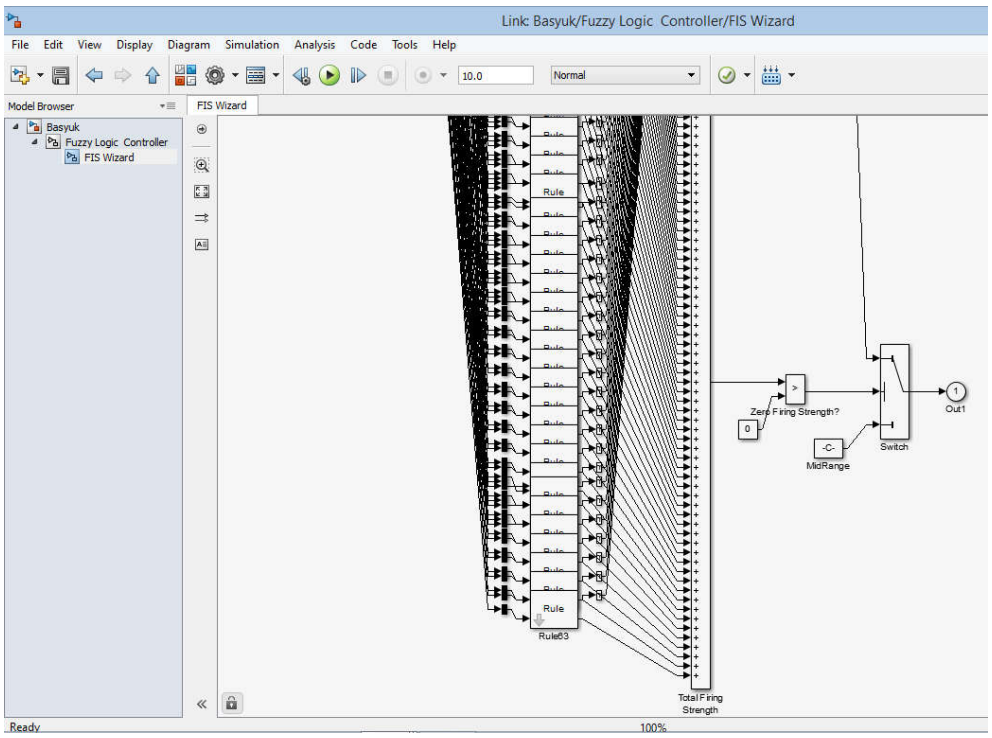

Рисунок 3.7 – Фрагмент перевірки дефазифікації нечіткого контролера

Блок фазифікації перетворює чіткі величини, виміряні на виході об'єкта керування, у нечіткі величини, що описані лінгвістичними змінними в базі знань. В свою чергу блок дефазифікації перетворює нечіткі дані з виходу блоку рішень у чітку величину, що використовується для керування об'єктом.

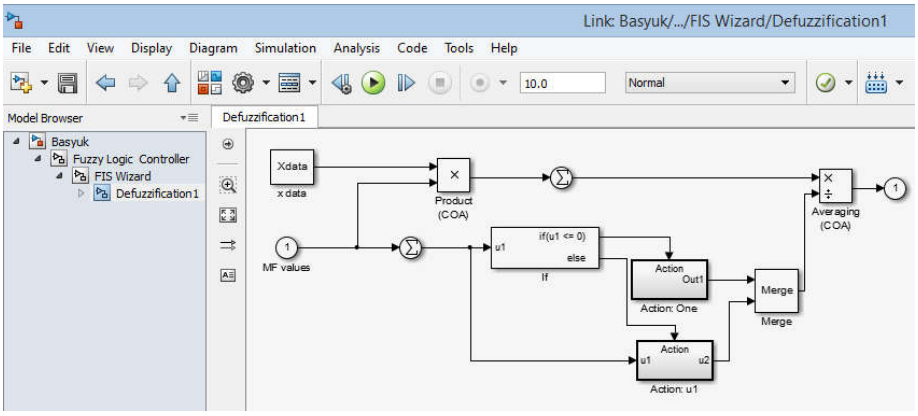

Рисунок 3.8 – Функціональна схема дефазифікатора

Дефазифікація – це процес отримання одиничного числа з виходу агрегованого нечіткого набору. Він використовується для передачі нечітких результатів висновку в чіткий результат. Іншими словами, дефазифікація

реалізується алгоритмом прийняття рішень, який вибирає найкраще чітке значення на основі нечіткого набору. Існує кілька форм дефазифікації, включаючи центр ваги (COG), середнє значення максимуму (MOM) та середньостатистичні методи. Метод COG повертає значення центру площі під кривою, а підхід MOM може розглядатися як точка, де на кривій отримано баланс.

Як відомо, основою роботи нечіткого контролера є також відповідна база продукційних правил, на основі яких і обробляються вхідні дані та проводяться відповідні підрахунки, що служать основою для вихідного результату.

Продукційні правила витягуються з нечіткої системи і тоді їх легко представити як частину роботи контролера за допомогою графічних логічних елементів. Для комплексної нечіткої системи кількість продукційних правил може бути дуже великою, в нашій системі таких правил також багато, тому для прикладу на рисунку 3.9 представлений вигляд продукційного правила, який можна побудувати у середовищі Simulink:

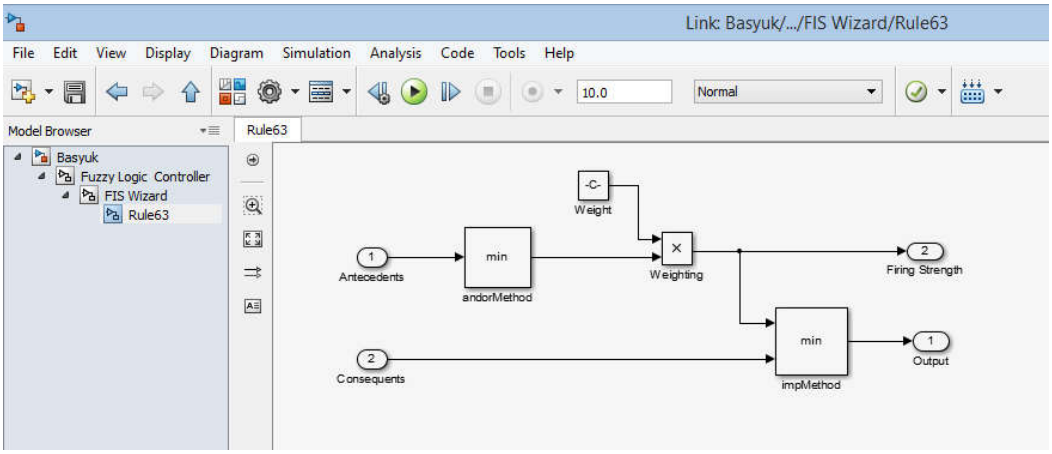

Рисунок 3.9 – Схема роботи продукційного правила

Іншою важливою складовою роботи будь якої нечіткої системи та нечіткого контролера є функції належності для вхідних та вихідних змінних (рисунок 3.10, 3.11). Вони слугують для графічного представлення даних а також для їх обробки. Як вже було описано в попередньому розділі, для

вхідних даних було обрано функції належності дзвуноподібного типу (або п–типу), а для вихідної змінної функції належності трикутного типу.

Вибір був зроблений на основі аналізі існуючих функції, їх властивостей та особливостей, враховано всі деталі. В ході комбінування та практичних спроб реалізації, було вирішено, що саме такі функції належності дозволять відобразити варіацію даних так, щоб це було максимально чітко та зрозуміло для користувача.

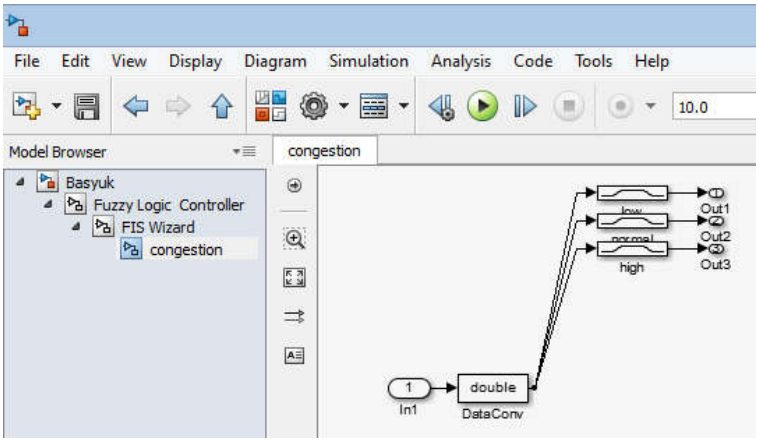

Рисунок 3.10 – Схема опрацювання функції належності вхідних змінних

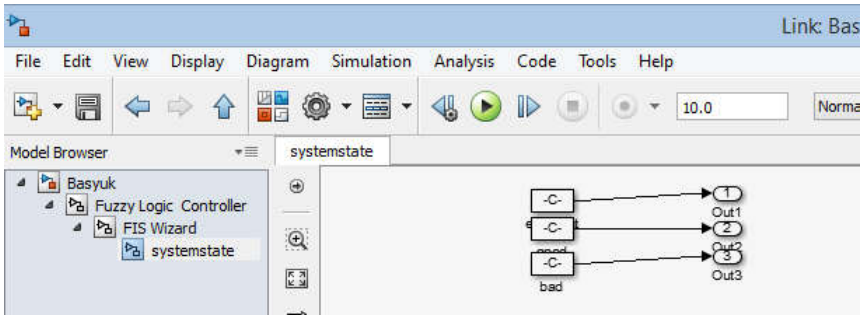

Рисунок 3.11 – Схема функції належності вихідної змінної

Нечіткий контролер виробляє сигнал адаптації в залежності від величини розбіжності між вихідними координатами об'єкта управління та еталонної моделі.

Залежно від особливостей об'єкта управління можуть використовуватися еталонна модель традиційного типу або нечітка модель. Адаптація нечіткого контролера може полягати у зміні правих частин тих

правил бази правил, які призводять до низької якості управління. Загальна ідея адаптивних систем полягає у введенні в систему управління додаткового контуру адаптації, який забезпечує корекцію закону управління. Адаптація з корекцією правил полягає в наступному. Адаптивний нечіткий регулятор оцінює інформацію про зміну параметрів об'єкта управління за значеннями помилки управління і її похідної. Контур адаптації формує коригуючий вплив, змінюючи праві частини правил нечіткого управління. Таблиця лінгвістичних правил (ТЛП) адаптації використовує інформацію про бажаний відгук системи в вигляді таблиці відповідностей. Деякі ситуації сприймаються як відповідні нормальному протіканню перехідного процесу, інші сприймаються як ті, що вимагають корекції. Корекція правил полягає в зміні не поточного правила, а попереднього правила, яке і створило незадовільну поточну ситуацію.

Отже, за допомогою застосування на практиці теорії нечіткої логіки, а також відповідних програмних засобів можна досягнути найкращого результату та побудувати ефективну та найголовніше працюючу модель контролера, яка легко може бути імплементованою до певної комп’ютерної системи.

3.3 Аналіз роботи та сфера застосування розробленого нечіткого контролера

Для того, щоб можна було легко відслідковувати результати моделювання, а отже роботу нечіткого контролера, до схеми контролера в середовищі Simulink було встановлено декілька осцилографів. Пристрій призначений, щоб досліджувати, спостерігати, вимірювати і записувати амплітудні, часові параметри електричних сигналів, які виводяться на вході, на екрані або записуються на фотострічці. Сучасний осцилограф може

вимірювати сигнал гігагерцових хвиль [50].

Осцилографи відносяться до вимірювальної техніки, яка призначена для того, щоб вивчати тимчасові, амплітудні коливання електричних сигналів. Прилад використовується в багатьох сферах для контролю якості сигналу.

На рисунках 3.12, 3.13, 3.14 та 3.15 графічно зображені значення вхідних змінних нечіткого контролера, а також вихідної змінної, яка і показує результат роботи і дозволяє визначити загальний стан досліджуваної нечіткої системи.

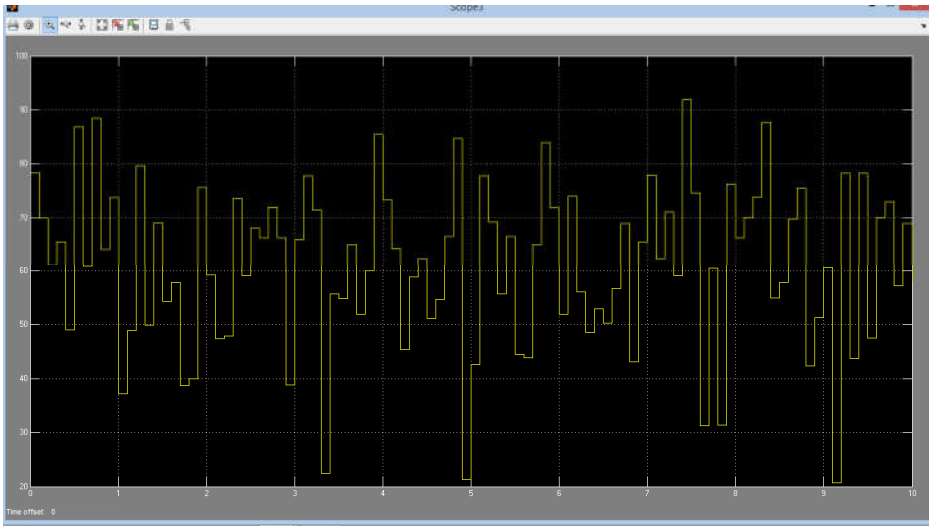

Рисунок 3.12 – Задання значень вхідної змінної температури

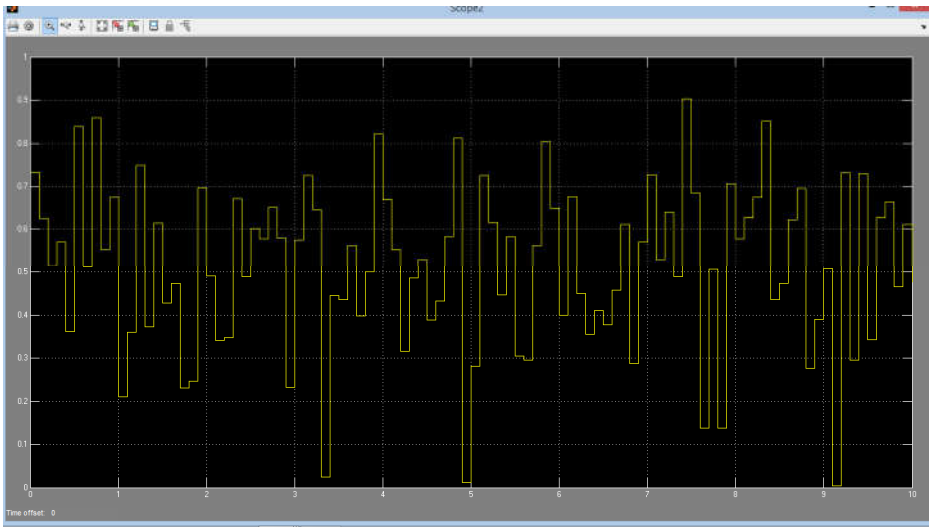

Рисунок 3.13 – Задання значень вхідної змінної швидкодії

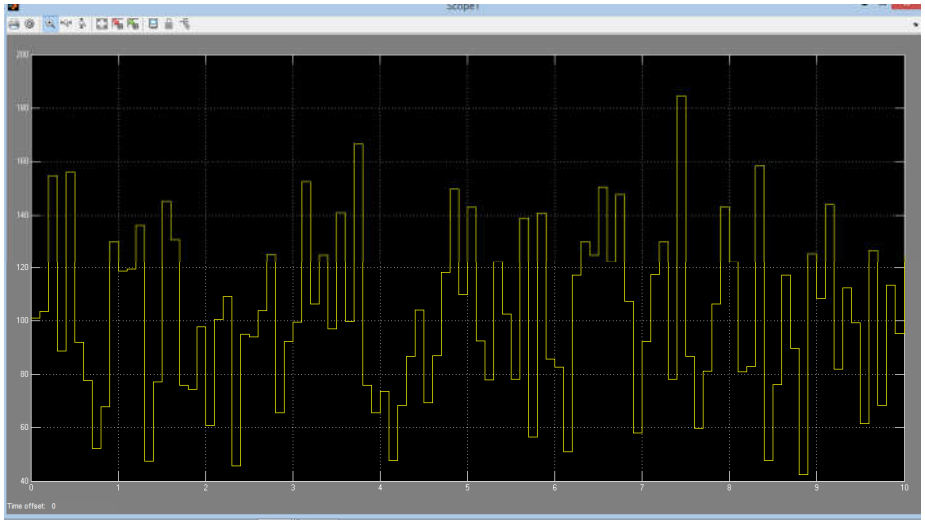

Рисунок 3.14 – Задання значень вхідної змінної навантаженості

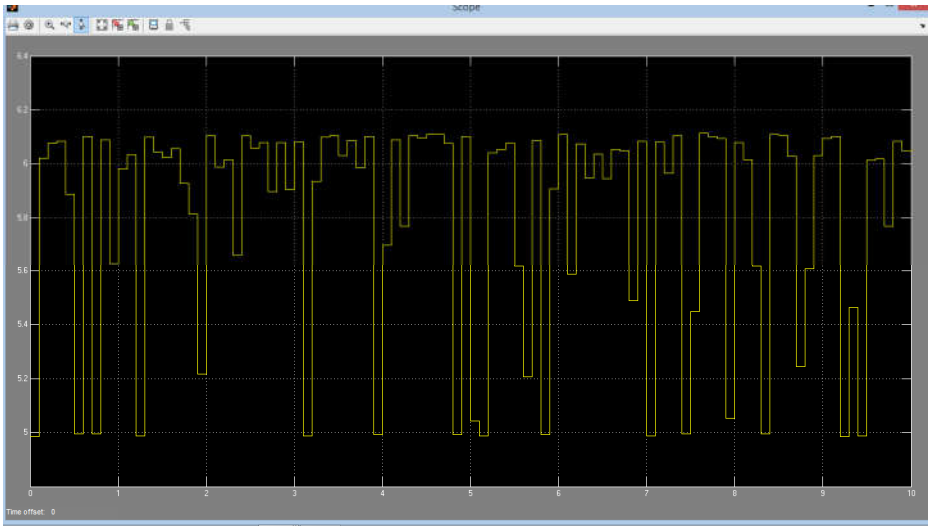

Рисунок 3.15 – Значення виходу нечіткого контролера

Отже, результати роботи контролера доволі легко отримати та відстежити. Достатньо лише щоб контролер зчитав значення з входів, опрацював їх за допомогою бази правил, методів фазифікації та дефазифікації і тоді, як результат, на виході ми отримуємо чітко структуровані значення в певних межах, які і дозволяють визначити кінцевий стан системи [50].

Контролери на основі нечіткої логіки використовуються в різних сферах людського життя, від медицини до інформаційних технологій. До

прикладу, можливе використання для реалізації системи управління навігацією мобільного робота на основі нечіткої логіки. Нечіткі правила, вбудовані в контролер мобільного робота, дозволяють йому уникати перешкод у середовищі з перешкодами, що включає інших мобільних роботів.

Щоб роботи не стикалися один з одним, кожен робот також містить набір правил запобігання зіткненням, реалізованих як модель Петрі у своєму контролері. Система управління навігацією була протестована в симуляції та на фактичних мобільних роботах. Результатом роботи можуть бути представлені результати тестів, щоб продемонструвати, що система дозволяє безлічі роботів вільно блукати для пошуку цілей у невідомому середовищі, що містить перешкоди, не потрапляючи на перешкоди чи один на одного.

Іншим прикладом застосування може слугувати розробка адаптивного нечіткого контролеру для розробки високоефективного відмовостійкого перемикаючого двигуна, що перемикає частоту (SRM). Нечіткий контролер постійно пристосовує свої властивості для регулювання крутного моменту машини, як цього бажає приводна система, навіть у несправних умовах.

Адаптація функцій нечіткої приналежності призводить до подовженого періоду провідності та збільшення максимального струму здорових фаз, щоб максимально забезпечити заданий крутний момент. Адаптивний нечіткий контролер забезпечує плавний вихід крутного моменту з мінімальною пульсацією, навіть в умовах несправності, даючи високоефективний привід SRM з відмовою від роботи.

Іншою сферою застосування є кондиціонування та регулювання температури. Забезпечення теплового комфорту та економія енергії – дві основні цілі систем опалення, вентиляції та кондиціонування (ОВК). Контролер з зворотним зв'язком температури не може найкращим чином досягти теплового комфорту. Це пояснюється тим, що на тепловий комфорт впливає безліч змінних, таких як температура, відносна вологість, швидкість повітря, радіація навколишнього середовища, рівень активності та ізоляція

тканини. У цій розробці індекс прогнозованого середнього значення (PMV) Fanger використовується як зворотний зв'язок контролера. Це спрощується без введення значної помилки. Розроблені теплові моделі кабіни та системи ОВК. Ємність охолодження випарника вибирається як критерій споживання енергії [50].

Розроблені два нечіткі контролери: один з температурою як зворотний зв'язок, а інший показник PMV як його зворотний зв'язок. Результати показують, що контролер зворотного зв'язку PMV краще контролює тепловий комфорт та енергоспоживання, ніж система із зворотним зв'язком температури. Далі параметри нечіткого контролера оптимізовані генетичним алгоритмом. Результати показують, що рівень теплового комфорту ще більше збільшується при зменшенні споживання енергії. В кінці, проводиться аналіз стійкості, який показує стійкість оптимізованого контролера до варіації змінних.

Також можливе застосування роботи нечітких контролерів для регулювання дорожнього руху. До прикладу, представлена конструкція та оцінка нечіткого логічного контролера дорожнього руху для ізольованого перехрестя.

Контролер розроблений так, щоб відповідати вимогам руху в режимі реального часу. Нечіткий контролер використовує детектори петлі транспортного засобу, розміщені вгорі від перехрестя на кожному підході, для вимірювання потоків підходу та оцінки черг. Ці дані використовуються для вирішення, через регулярні інтервали часу, продовжувати чи припиняти фазу поточного сигналу. Ці рішення приймаються за допомогою двоступеневої нечіткої логічної процедури. На першому етапі потоки руху спостережуваних підходів використовуються для оцінки відносної інтенсивності руху в конкуруючих підходах. Ці інтенсивності руху потім використовуються на другому етапі для визначення, чи слід продовжувати або припиняти поточну фазу сигналу.

Розроблений нечіткий контролер аналізу стану комп’ютерної системи

є досить універсальним, адже може використовуватись в будь якій сфері та в будь якому місці, де використовуються комп'ютери. Проблема збоїв та неполадок в комп'ютерах зустрічається досить часто, а особливо в сфері інформаційних технологій та на великих підприємствах. Це відбувається саме через інтенсивне використання техніки, в результаті чого її термін обслуговування скорочується в рази.

Тому, щоб уникнути незворотніх змін в комп'ютерних системах, які вже не будуть підлягати відновленню та ремонту, доцільно використовувати програмний чи апаратний засіб, що дозволяє попереджати схожі проблеми та вирішувати ще до їх появи.

Хорошою практикою було б поєднати розроблений контролер з якимось програмним засобом, який би працював в фоновому режимі та не вимагав складних та регулярних маніпуляцій. Достатньо було б того, що коли є якась загроза, користувач би отримував сповіщення, де була б вказана інформація про поточний стан системи та рекомендації до подальших дій.

Конкретна інформація для сповіщень була б заснованою на роботі контролера, бо саме з результатів на виході зрозуміло поточний стан та всі відхилення в момент часу. Можна сказати, що дана розробка є ефективним, швидким, а головне, бюджетним методом, що дозволяє досягнути поставленої задачі з найвищим коефіцієнтом корисної дії.

3.4 Висновки до розділу 3

У третьому розділі в основному висвітлена практична реалізація розробки. Найперше описано середовище, в якому була реалізована нечітка система та відповідний нечіткий контролер. Найкраще для цього підходить модуль Simulink та Fuzzy Logic Toolbox, що є складовими частинами програмного забезпечення Matlab. Вони легко дозволяють побудувати

бажану систему чи контролер за допомогою бібліотек нечіткої логіки та спеціалізованих інструментів.

Розроблено нечіткий контролер, що дозволяє імплементувати розробку для певної комп'ютерної системи. Описано та створено його структурну, функціональну та блок схеми. Окремо охарактеризовані фрагменти дефазифікатора та його функціональну схему. Також описані функціональні схеми продукційних правил, схема опрацювання функції належності вхідних змінних, схема функції належності вихідних даних.

Для того, щоб можна було легко відслідковувати результати моделювання, а отже роботу нечіткого контролера, до схеми контролера в середовищі Simulink було встановлено декілька осцилографів. Саме зчитані з них результати дозволяють повною мірою побачити результати роботи розробки та перевірити чи варіюються вони в відведених межах.

Описано основні галузі та напрямки для застосування нечіткої логіки, та зокрема нечітких систем та контролерів. Також зроблено припущення про те, для яких підприємств буде найкраще підходити розроблений алгоритм та контролер. Висвітлено можливі рішення з його подальшої реалізаці, пропозиції щодо покращення функціоналу та можливий додатковий функціонал у вигляді додаткового програмного забезпечення, з простим та зрозумілим інтерфейсом, що дозволить використання розробки не тільки у великих промислових масштабах, а й у масштабах звичного будинку, щоб кожен пересічний користувач зміг знайти розробку ефективною та корисною для себе.

ВИСНОВКИ

У процесі виконання магістерської роботи, було розроблено нечіткий алгоритм діагностики комп'ютерних систем, а також на його основі відповідну нечітку систему та контролер. Новизна роботи полягає у вдосконаленому алгоритмі діагностики комп'ютерних систем за використання апарату нечіткої логіки.

Протягом розробки вирішено такі основні завдання:

1) проаналізовано проблеми, які виникають при користуванні комп'ютером та в загальному в комп'ютерних системах, виділено найпоширеніші з них, що дало змогу спеціалізувати розроблений алгоритм;

2) досліджено можливі алгоритми нечіткого виводу по базі знань і на їх основі створено алгоритм для реалізації завдання дипломного проекту, який дозволяє швидко та ефективно вирішувати поставлені задачі;

3) побудовано функції належності двох типів для вхідних та вихідного значень, які є дзвуноподібними та трикутними, що дозволяє краще побачити кінцевий результат;

4) сформовано базу правил нечіткої системи, яка охоплює всі можливі випадки та складається з 60 правил типу «якщо-то»;

5) здійснено моделювання та аналіз роботи нечіткої системи за допомогою Fuzzy Logic Toolbox середовищі Matlab, що дозволяє візуалізувати отримані результати, а отже зробити їх наочними;

6) проаналізовано роботу нечіткої системи за допомогою поверхонь значень нечіткої системи, котрі дають змогу побачити результат моделювання кожного правила, а також криву змін;

7) розроблено структурну схему нечіткої системи, що дозволяє краще зрозуміти принципи роботи та правильно розподілити потоки даних між функціональними елементами;

8) на основі нечіткої системи розроблено контролер, що дозволяє реалізувати її на практиці, а отже побудувати пристрій, що буде відповідати всім вимогам та ефективно виконувати першочергову задачу аналізу;

9) проаналізовано роботу контролера за допомогою програмного забезпечення Simulink, що дало змогу побачити коректність зміни сигналів за допомогою осцилографів.

Використання розробленої нечіткої системи діагностики роботи комп'ютерних систем, дозволить зменшити шанси виникнення проблем шляхом їх попередження та вчасної діагностики, що дозволить значно зекономити час та ресурси. Надалі, планується реалізація контролера на великому підприємстві, а також створення програмного забезпечення, з простим та зрозумілим інтерфейсом, що дозволить пересічному користувачу легко застосовувати дану розробку на персональному комп'ютері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Функції обробки нечітких даних. Щоденник: веб–сайт. URL: http://company.shodennik.ua/functions(дата звернення: 11.11.2018).

2. Novak V., Perfilieva I., Mockor J. Mathematical principles of fuzzy logic: Kluwer Academic Publishers,1999. P. 15.

3. Басюк Н.В., Булило І.І. Аналіз поточного стану комп'ютерної системи на основі нечіткої логіки. Матеріали І науково–практичної конференції молодих вчених та студентів «Інтелектуальні комп'ютерні системи та мережі». Тернопіль: Видавництво ТНЕУ, 2019. С. 22.

4. Басюк Н.В., Булило І.І. Застосування нечіткого контролера для системи кондиціонування. Матеріали ІІ науково–практичної конференції молодих вчених та студентів «Інтелектуальні комп'ютерні системи та мережі». Тернопіль: Видавництво ТНЕУ, 2019. С. 13.

5. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання магістерської роботи з освітнього ступеня "Магістр". Спеціальність: 123 – Комп'ютерна інженерія. Магістерська програма – Комп'ютерна інженерія". Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2018. – 41 с.

6. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія». Під ред. О.М. Березького. – Тернопіль: ТНЕУ, 2019. – 33 с.

7. Stanford academic informations. Plato: web–site. URL: https://plato.stanford.edu/entries/logic–fuzzy(дата звернення: 22.03.2019).

8. Cintula P. Fuzzy Logics as the Logics of Chains: Fuzzy Sets and Systems. Libor, 2006. P. 606.

9. Godo L. Fuzzy Sets and Systems. Monoidal T–Norm Based Logic: Towards a Logic for Left–Continuous T–Norms. Waweland, 2001. P. 25.

10. What Is Fuzzy Logic? Mathworks: web–site. URL: https://www.mathworks.com/help/fuzzy/what–is–fuzzy–logic.html(дата звернення: 05.06.2019).

11. Zadeh L. Real–Life Applications of Fuzzy Logic. Fuzzy logic now and then. Hindawi, 2013. P. 125.

12. Yager R. Fuzzy Sets and Applications. Introduction. Wiley, 1987. P. 8.

13. Encoder the Newsletter of the Seattle Robotics Society (Fuzzy logic – an introduction). Seatlerobotic: web–site. URL: http://www.seattlerobotics.org/encoder/mar98/part2.html(дата звернення: 23.01.2019).

14. Zimmerman H. Fuzzy set theory and its applications. Fuzzy logic intoduction. Kluwer, 1991. P. 315.

15. Блюмин С., Шуйкова И., Сараев П. Нечеткая логика: алгебраические основы и приложения: монография. Киев, ЛЭГИ, 2002. 113 с.

16. Cordon O., Herrera F. A General study on genetic fuzzy systems. Genetic Algorithms in computer science. Tante, 1995. P. 33.

17. Леоненков А. Нечеткое моделирование в MATLAB и fuzzyTECH. Санкт–Петербург: БХВ–Петербург, 2005. 736 с.

18. Abadeh M., Habibi J., Lucas C. Intrusion Detection Using a Fuzzy Genetics–Based Learning Algorithm. Journal of Network and Computer Applications. 2007. No.30. P. 414–418.

19. Mamdani E. Application of fuzzy algorithms for the control of a simple dynamic plant. Proc. IEEE 121, 1974. P. 1585–1588.

20. Штовба С. Обеспечение точности и прозрачности нечеткой модели Мамдани при обучении по экспериментальным данным. Харьков, 2007. С. 102–104.

21. Дубчак Л. Метод обробки нечітких даних на основі механізму Мамдані. Системи обробки інформації. 2012. Вип. 7(105). 131с.

22. Ротштейн А., Штовба С. Идентификация нелинейной зависимости нечеткой базой знаний с нечеткой обучающей выборкой. Кибернетика и системный анализ. 2006. Вып.2. С. 17–24.

23. Мирончук Ю., Куріненко О. Побудова функцій належності нечітких множин, які відповідають кількісним експертним оцінкам фізичних величин. Системи обробки інформації. 2017. Вип.1. С. 93–97.

24. Лозинський А., Демків Л. Дослідження впливу вигляду функції належності на динамічні показники системи при багатокритеріальній оптимізації зі змінними ваговими коефіцієнтами. Електротехнічні та комп'ютерні системи. 2012. Вип.5. С. 137–144.

25. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. Москва: Горячая линия – Телеком, 2004. 452 с.

26. Лазарєв Ю. Моделювання динамічних систем у Matlab. Київ: НТУУ «КПІ», 2011. 421 с.

27. Passino K., Yurkovich S. Fuzzy Control. California: Addison–Wesley, 2001. 53 P.

28. Koo T. Analysis of a Class of Fuzzy Controllers, in Proc. 1st Asian Fuzzy Systems Sump. Singapore: Way, 1998. P. 35–38.

29. Iancu I. Extended Mamdani Fuzzy Logic Controller. California: ACTA Press, 2001. P. 143–149.

30. Simulation and Model–Based Design. Mathworks: web–site. URL: https://www.mathworks.com/products/simulink.html(дата звернення: 17.05.2019).

31. Cintula P. Fuzzy Logics as the Logics of Chains. Fuzzy Sets and Systems. Melburn: Libor, 2006. P. 606.

32. Godo L. Fuzzy Sets and Systems. Lonoidal T–Norm Based Logic: Towards a Logic for Left–Continuous T–Norms. Waweland, 2001. P. 25.

33. What Is Fuzzy Logic? Mathworks: web–site. URL: https://www.mathworks.com/help/fuzzy/what–is–fuzzy–logic.html(дата звернення: 08.07.2019).

34. A semantics–driven, fuzzy logic–based approach to knowledge representation and inference. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/35–2(дата звернення: 15.10.2019).

35. Expert diagnosis of computer systems, neuro–fuzzy knowledge base. IEEE: web–site. URL: https://ieeexplore.ieee.org/document/metrics#metrics(дата звернення: 29.11.2019).

36. The neuro–fuzzy diagnostic model synthesis with hashed transformation in the sequence and parallel mode. ZNTU: web–site. URL: https://ric.zntu.edu.ua/article/view/101022/96247(дата звернення: 02.02.2019).

37. Usability Determination Using Multistage Fuzzy System. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/article/pii/S1842(дата звернення: 18.08.2019).

38. A novel fuzzy decision–making system for CPU scheduling algorithm. Springer: web–site. URL: https://link.springer.com/article/10.1007/s00521–015–1987–8(дата звернення: 20.05.2019).

39. Network–based output tracking control for T–S fuzzy systems using an event–triggered communication scheme. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/article/pii/S0165011032(дата звернення: 21.09.2019).

40. Expert evaluation model of the computer system diagnostic features. IEEE: web–site. URL: https://ieeexplore.ieee.org/document/7027101/metrics(дата звернення: 29.12.2018).

41. Diagnosing computer hardware failures using expert system (rule–based technique). ResearchGate: web–site. URL: https://www.researchgate.net/publication/79205502_DIAGNOSING_COMPUTER_HARDWARE_FAILURES_USING_EXPERT_SYSTEM_RULEBASED_TECHNIQUE(дата звернення: 01.10.2018).

42. Computer Aided Development of Fuzzy, Neural and Neuro–Fuzzy Systems. ResearchGate: web–site. URL: https://www.researchgate.net/publication/312590719_Computer_Aided_Development_of_Fuzzy_Neural_and_Neuro–Fuzzy_Systems(дата звернення: 13.06.2019).

43. A fuzzy expert system for automatic seismic signal classification. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/article/pii/S09574114053(дата звернення: 10.10.2019).

44. Classification of Network Traffic Using Fuzzy Clustering for Network Security. Springer: web–site. URL: https://link.springer.com/chapter/10.1007/978–3–319–62701–4_22(дата звернення: 12.04.2019).

45. Abadeh M., Habibi L., Kortos N. Intrusion Detection Using a Fuzzy Genetics–Based Learning. Deli, 2007. P. 314–318.

46. Simulation and Model–Based Design. Mathworks: web–site. URL: https://www.mathworks.com/simulink.html(дата звернення: 16.03.2019).

47. Сравнение нечетких алгоритмов. Cypherpunks: веб–сайт. URL: http://www.cypherpunks.ru/fuzzy/comparison.html(дата звернення: 19.07.2019).

48. Административные и бытовые здания. Document: веб–сайт. URL: http// www.document.ua/docs/tdoc429.php(дата звернення: 30.10.2019).

49. Нечітка логіка на прикладах. iSearch: веб–сайт. URL: http://www.isearch.kiev.ua/index.php/uk/internetsecurity/837–fuzzy–message(дата звернення: 05.10.2019).

50. Mamdani algorithm – Simulink realization. Solutionmes: web–site. URL: http://solutionmes.wikidot.com/mamdani–fuzzy(дата звернення: 14.11.2019).

ДОДАТОК А

Лістинг коду розробленої системи

```
[System]
Name='fuzzysys'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=63
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='temperature'
Range=[20 100]
NumMFs=3
MF1='normal':'gbellmf',[20 2.31 20]
MF2='morehigh':'gbellmf',[14.3454545454546 3.28 59.8]
MF3='high':'gbellmf',[19.99 3.278 99.83]

[Input2]
Name='speed'
Range=[0 1]
NumMFs=3
MF1='high':'gbellmf',[0.25 3.278 0]
MF2='normal':'gbellmf',[0.25 3.278 0.5]
MF3='low':'gbellmf',[0.25 3.278 1]

[Input3]
Name='congestion'
Range=[0 200]
NumMFs=3
MF1='low':'gbellmf',[50 3.278 0]
MF2='normal':'gbellmf',[34.2494714587738 3.28 100]
MF3='high':'gbellmf',[50 3.278 200]

[Output1]
Name='systemstate'
Range=[0 10]
NumMFs=3
MF1='excellent':'trimf',[-4 0 4]
MF2='good':'trimf',[0.979 4.979 8.98]
MF3='bad':'trimf',[6 10 14]

[Rules]
```

```
1 3 1, 1 (1) : 1
2 2 2, 2 (1) : 1
3 1 3, 3 (1) : 1
1 1 1, 2 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 1 0, 3 (1) : 1
1 2 1, 2 (1) : 1
1 2 2, 2 (1) : 1
1 2 3, 1 (1) : 1
1 2 0, 2 (1) : 1
1 3 2, 1 (1) : 1
1 3 3, 1 (1) : 1
1 3 0, 1 (1) : 1
1 0 1, 3 (1) : 1
1 0 2, 3 (1) : 1
1 0 3, 3 (1) : 1
1 0 0, 2 (1) : 1
2 1 1, 3 (1) : 1
2 1 2, 3 (1) : 1
2 1 3, 3 (1) : 1
2 1 0, 3 (1) : 1
2 2 1, 2 (1) : 1
2 2 3, 2 (1) : 1
2 2 0, 2 (1) : 1
2 3 1, 2 (1) : 1
2 3 2, 2 (1) : 1
2 3 3, 2 (1) : 1
2 3 0, 1 (1) : 1
2 0 1, 3 (1) : 1
2 0 2, 2 (1) : 1
2 0 3, 3 (1) : 1
2 0 0, 3 (1) : 1
3 1 1, 3 (1) : 1
3 1 2, 2 (1) : 1
3 1 0, 3 (1) : 1
3 2 1, 3 (1) : 1
3 2 2, 2 (1) : 1
3 2 3, 3 (1) : 1
3 2 0, 3 (1) : 1
3 3 1, 2 (1) : 1
3 3 2, 2 (1) : 1
3 3 3, 2 (1) : 1
3 3 0, 2 (1) : 1
3 0 1, 3 (1) : 1
3 0 2, 3 (1) : 1
3 0 3, 2 (1) : 1
3 0 0, 3 (1) : 1
0 1 1, 2 (1) : 1
0 1 2, 2 (1) : 1
0 1 3, 3 (1) : 1
0 1 0, 3 (1) : 1
0 2 1, 2 (1) : 1
```

```
0 2 2, 2 (1) : 1
0 2 3, 2 (1) : 1
0 2 0, 2 (1) : 1
0 3 1, 1 (1) : 1
0 3 2, 2 (1) : 1
0 3 3, 2 (1) : 1
0 3 0, 1 (1) : 1
0 0 1, 2 (1) : 1
0 0 2, 2 (1) : 1
0 0 3, 2 (1) : 1
```