Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Діагностування комп'ютерних систем на основі нечіткої логіки

Назва теми

КвРКІ. 170348.*17.03.26 ПЗ*
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія»*
Назва

Виконав: студент IV курсу, група КІ–17–3 _____ О.А. Парніцький
Підпис Ініціали, прізвище

Керівник _____ В.Ю. Тітова
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.В. Муляр
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки
та комп'ютерних систем і
мереж _____ Ю.П. Кльоц
Підпис Ініціали, прізвище

«17» червня 2021 р.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Кібербезпеки та комп'ютерних систем і мереж

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123Комп'ютерна інженерія

Освітня програма освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри Ю. П. Кльоц

"17" 02 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Парніцькому Олегу Анатолійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи)Діагностування комп'ютерних систем на основі нечіткої логки

Керівник проекту (роботи) Тітова Віра Юріївнакандидат технічних наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університетувід 17.02.2021№ 44додаток №9

2. Строк подання студентом проекту (роботи) на кафедру 28.05.2021

3. Вихідні дані до проекту (роботи) алгоритм діагностування комп'ютерної системи на основі нечіткої логіки

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження предметної області та постановка задачі; обгрунтування базових положень шодо проєктуваннямікроконтролерної системи управління матричними світлодіодними панелями; Опис схем електричних (структурної,функційної, принципової) проєктованої системи; опис алгоритму роботи системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схеми розробленого алгоритму (Е8)

Виходи та функції приналежності розробленої системи(Е8)

Поверхні значень розробленої системи (Е8)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прий
Нормоконтроль	Муляр І.Вкандидат технічних наук, доцент	-	
Антиплагіат	Муляр І.В кандидат технічних наук, доцент	-	

7. Дата видачі завдання «_19_»_____02_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Прим
1.	Підготовка вступного розділу	Березень - 1 декада	
2.	Огляд існуючих методів, засобів	Березень - 2 декада	
3.	Обгрунтування обраних рішень	Березень - 3 декада	
4.	Підготовка опису електричних схем	Квітень - 1 декада	
5.	Виконання розрахункової частини	Квітень - 1 декада	
6.	Підготовка ескізів креслень	Квітень - 2 декада	
7.	Формулювання висновків	Квітень - 3 декада	
8.	Розробка додатків	Травень - 1 декада	
9.	Погодження розділів з консультантом з нормоконтролю	Травень - 1 декада	
10.	Оформлення графічного матеріалу	Травень - 2 декада	
11.	Оформлення пояснювальної записки	Травень - 2 декада	
12.	Попередній захист кваліфікаційної роботи	Травень - 3 декада	
13.	Доопрацювання кваліфікаційної роботи	Травень - 3 декада	
14.	Подання роботи для перевірки на плагіат	Травень - 3 декада	
15.	Захист кваліфікаційної роботи	Червень - 1 декада	

Студент О.А Парніцький

Підпис Ініціали, прізвище

Керівник проекту (роботи) В.Ю Тітова

Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «*Діагностування комп'ютерних систем на основі нечіткої логіки*».

Автор роботи: *Парніцький Олег Анатолійович*.

Керівник роботи: *Тітова Віра Юріївна*.

Пояснювальна записка: *65 с., 25 рис., 2 табл., 29 джерел*.

Графічна частина: 10 презентаційних слайдів.

Мета даної роботи полягає у в аналізі комп'ютерних систем їх характеристик, у розробці нечіткого алгоритму який дозволить проводити аналіз стану комп'ютерної системи.

Під час виконання кфаліфікаційної роботи, було створено нечіткий алгоритм який призначений для діагностування комп'ютерних систем, а також базуючись на ньому відповідну нечітку систему. Новизна даної роботи заключається у покращеному алгоритмі який діагностує комп'ютерну систему з використанням апарату нечіткої логіки.

Підпис студента

10.06.2021
Дата

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ. 170348.17.03.26 ПЗ	Пояснювальна записка	67		
			Графічні матеріали			
2		КвРКІ. 170348.17.03.26 Е8	Схеми розробленого	1		
			алгоритму			
3		КвРКІ. 170348.17.03.26 Е8	Виходи та функції	1		
			приналежності			
			розробленої системи			
4		КвРКІ. 170348.17.03.26 Е8	Поверхні значень	1		
			розробленої системи			

КвРКІ. 170348.17.03.26 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Парнійчук О.А		
Перевір.		Тітова В.Ю.		
Н. контр.		Муляр І.В		
Затв.		Кльоц Ю.П		

Літера	Аркуш	Аркушів
У	1	1

Відомість проекту
Пояснювальна записка ХНУ

ХНУ, КІ-17-3

ЗМІСТ

ВСТУП

Сучасний світ стрімко розвивається у напрямку Комп'ютерних систем, тому актуальність розробки пов'язаної з комп'ютерними системами тільки зростає з кожним роком. Комп'ютерні системи стали невід'ємною частиною нашого життя, вони використовуються у всіх сферах життя суспільства, і навіть стали незамінними для деяких професій. Комп'ютерні системи дозволяють вирішувати прикладні завдання предметних галузей діяльності, наприклад технологічна підготовка, керування, облік та автоматизація процесів. У сучасних реаліях комп'ютерні системи знайшли практичне застосування при дистанційному навчанні. Ще декілька років тому малося на увазі, що дистанційне навчання і заочне навчання це одне і теж, але зараз це звичайний метод навчання, але з використанням кейс– , ТБ– і мережевих технологій. Сьогодні користувачам доступні різне програмне забезпечення, наприклад: системні та прикладні програми(а саме: текстові редактори, системи управління базами даних, компілятори та інше).За допомогою цих програм користувач взаємодіє з операційною системою, яка вже і керує комп'ютером [1].

Оскільки комп'ютерні системи стали невід'ємною і незамінною частиною нашого життя, тому актуальність захисту комп'ютерних систем виходить на перший план. Захист комп'ютерних систем напряму залежить від своєчасного і регулярної діагностики, яка запобігає виникненню 90% наявних проблем.

Стійкість будь-якої системи може забезпечити нечітка система та відповідний нечіткий контролер.

Мета даної роботи полягає у в аналізі комп'ютерних систем їх характеристик, у розробці нечіткого алгоритму який дозволить проводити аналіз стану комп'ютерної системи.

Щоб досягти необхідної задачі потрібно вирішити наступні задачі:

1) проаналізувати можливі стани комп'ютерних систем;

2) провести дослідження нечіткої логіки та на основі отриманих знань створити алгоритм для нечіткого виводу ;

3) побудувати функції належності;

4) дослідити роботу реалізованої нечіткої системи;

5) зробити висновки щодо правильності роботи розробленої системи.

Дана робота може бути корисна для простого користувача, для будь-якого підприємства і взагалі у всіх сферах життя суспільства. Найбільш актуальним дана робота, на мою думку, буде для підприємств оскільки їх дохід напряму залежить від стабільної робота комп'ютерів. А даний алгоритм який є водночас простим в реалізації і ефективним в роботі дозволить уникнути не бажаних проблем.

1 ДОСЛІДЖЕННЯ ОСНОВНИХ ПОНЯТЬ КОМП'ЮТЕРНОЇ СИСТЕМИ ТА НЕЧІТКОЇ ЛОГІКИ

1.1 Визначення комп'ютерної системи

Сукупність різних компонентів які використовуються для спільної обробки даних називають комп'ютерною системою. Метою для комп'ютерної системи слугує процес завдяки якому вирішення складних завдання на комп'ютері стає більш простішим. Комп'ютерна система своїм функціоналом об'єднує в собі елементи програмного і апаратного забезпечення. Апаратне забезпечення це всі механічні пристрої за допомогою яких комп'ютер виконує усі свої фізичні функції. Програмне забезпечення це програми і додатки за допомогою яких і відбувається керування комп'ютером. Саме програмне забезпечення виконує усі логічні і математичні функції. Документація включає опис усіх можливих операцій за допомогою яких можна повноцінно використовувати програмне і апаратне забезпечення [2].

В поєднані усі ці елементи (програмне і апаратне забезпечення, документація) і утворюють комп'ютерну систему. Стандартний набір який входить в комп'ютерну систему складається з трьох базових апаратних складових: комп'ютер який і обробляє усі операції; термінальний пристрій який використовується для двостороннього зв'язку між користувачем і комп'ютером; і накопичувач для зберігання даних. Ці три апаратних елемента є основними і необхідними елементами для функціонування комп'ютерної системи.

Усі комп'ютерні системи класифікуються за певними ознаками. Найпоширенішою є класифікація за Флінном. Класифікація за Флінном являє собою класифікацію архітектур Електронно Обчислювальних Машин (ЕОМ) за ознаками наявності в них паралелізму в потоках команд і даних. Ця класифікація була запропонована в 1966 році Майклом Флінном і розширена в 1972 році. Завдяки цій класифікації усі ЕОМ зводяться до чотирьох класів [3]:

1. Single Instruction stream over a Single Data stream, або скорочено SISD це система яка містить в собі тільки один потік команд і один потік даних.

2. Single Instruction, Multiple Data або ж SIMD це система яка містить в собі тільки один потік команд але вже множинний потік даних.

3. Multiple Instruction Single Data (MISD) це система яка містить в собі множинний потік команд але в той самий час і одиночний потік даних.

4. Multiple Instruction Multiple Data або ж скорочено MIMD) це система яка містить в собі множинний потік команд і множинний потік даних.

SISD архітектура представляє собою класичний комп'ютер фон-Найманської архітектури з одним процесом який працюючи з одним потоком даних послідовно виконує одну інструкцію за іншою. Даний клас не використовує паралелізм тому машина на базі SISD не є паралельною. На рисунку 1.1 зображено архітектуру SISD.

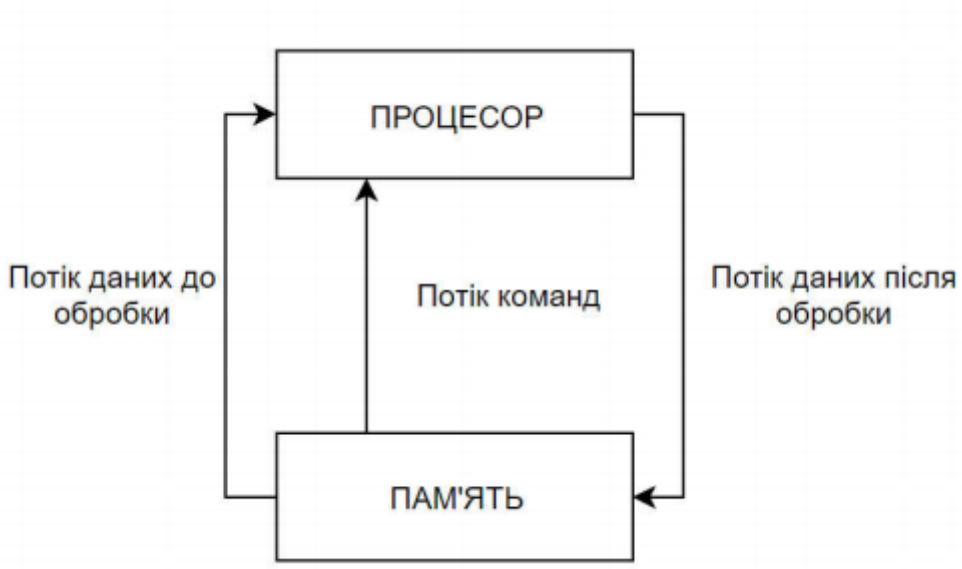


Рисунок 1.1 – Архітектура SISD

SIMD до цього класу належать векторні процесори і матричні процесори. Векторні процесори це звичайні сучасні процесори які працюють при виконанні команд в роботі з векторними розширеннями. Матричні процесори це особливий підвид процесорів з великою кількістю процесорів. У комп'ютерах з SIMD архітектурою один процес завантажує інструкцію і необхідні дані для операції

описаної в інструкції і виконує цю операцію над усім набором даних одночасно. Архітектура SIMD зображена на рисунку 1.2.

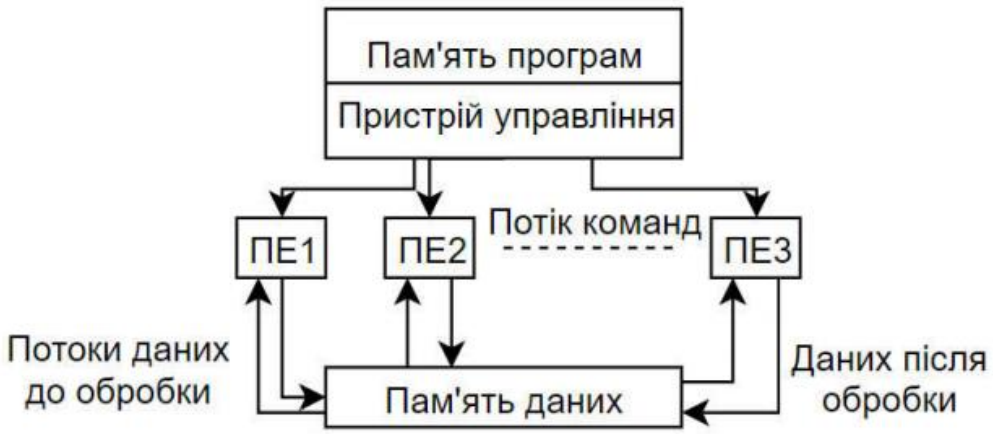


Рисунок 1.2 – Архітектура SIMD

До архітектури класу MISD деякі дослідники відносять конвеєрні ЕОМ, але це досягло остаточного визнання. Також до цієї архітектури можна віднести системи з гарячим резервуванням. Крім вище перерахованих систем до MISD ще відносять і систолічні масиви процесорів. Архітектура MISD зображена на рисунку 1.3.

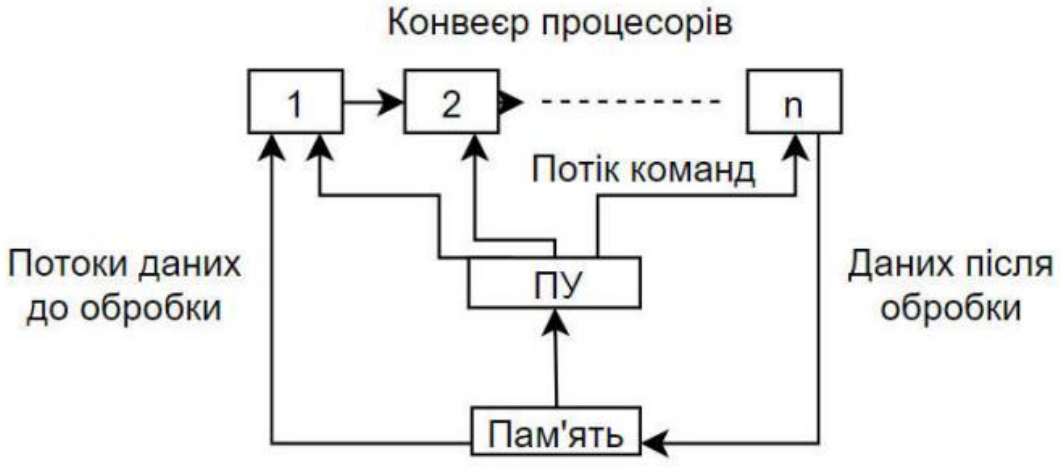


Рисунок 1.3 – Архітектура MISD

MIMD це клас включає в себе багатопроцесорні системи, в яких процесори обробляють множинні потоки даних. Також сюди можна віднести традиційні мультипроцесорні машини, комп'ютерні кластери і багатоядерні і багатопотокові процесори [4]. MIMD архітектура зображена на рисунку 1.4.

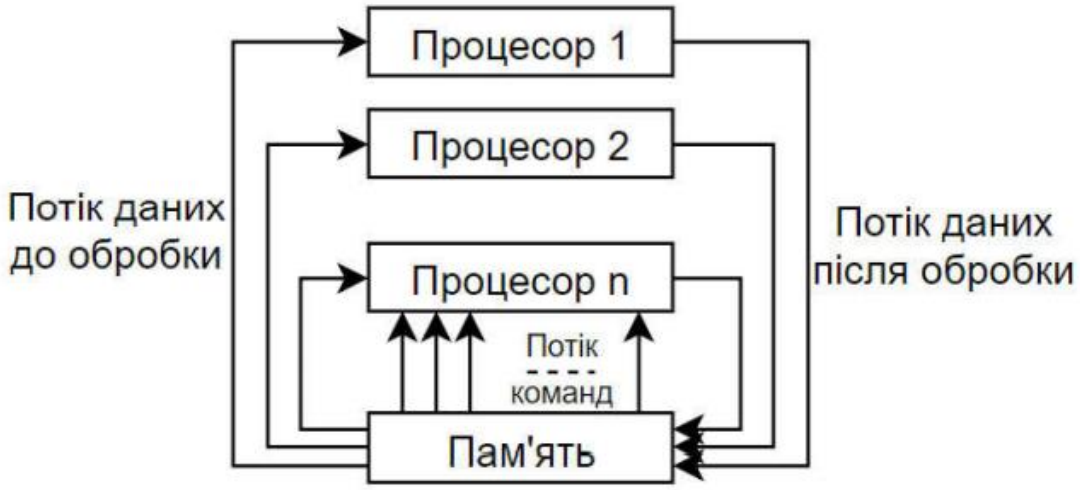


Рисунок 1.4 – Архітектура MIMD

У спеціалізованій літературі можна знайти такі підкласи MIMD:

1. Single Program Multiple Data (SPMD).
2. Multiple Programs Multiple Data (MPMD).

Single Program Multiple Data (SPMD)– це система в якій на всіх процесорах комп'ютера з MIMD архітектурою виконується тільки одна задача і на різних процесорах обробляє зовсім різні блоки даних.

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

1) на одному процесорі комп'ютера працює майстер програм а на інших підпорядкована їй програма;

2) на різних процесорах працюють різні програми які працюють незалежно один від одного і по різному обробляють дані, але в деяких випадках обмінюються даними для переходу на інший крок [5].

Відношення комп'ютера до певного класу річ суб'єктивна і сильно залежить від дослідника. Так одні і ті ж машини можуть бути віднесені до різних класів,

наприклад конвеєрні машини можуть бути віднесені до SISD(конвеєр єдиний процесор), і до SIMD(векторний потік даних з конвеєрним процесом), до MISD класу також оскільки безліч процесорів конвеєра обробляють один потік даних, навіть до класу MIMD(виконання послідовності різних команд з множинним скалярним потоком даних).

1.2 Визначення стану комп'ютерної системи

Щоб визначити показники надійності потрібно провести ряд дій за допомогою яких можна визначити даний показник. Для визначення показника надійності проводять певні розрахунки, проводять певні випробування і аналізують їхні результати, за допомогою моделювання на ЕОМ, а також після аналізу фізико-хімічних процесів які обумовлюють надійність виробу. Розрахунки надійності засновані на тому, що існують певні залежності між показниками надійності певних елементів виробу і надійності виробу в цілому. Для створення цих залежностей використовують такі методи як рішення рівнянь складених за допомогою схеми надійності, або на підставі певних логічних зв'язків між станами виробу; вирішення диференціальних рівнянь які описують процеси переходу від одного стану в інший; складання функція для опису станів складного продукту.

Процес розрахунку надійності відбувається в загальному на етапі проектування виробу для того щоб спрогнозувати для даного виробу очікуваної надійності. Розрахунок надійності на етапі проектування дозволяє вибрати оптимальний варіант конструкції і методи забезпечення надійності, виявити слабкі місці, обгрунтувати робочі режими і методи обслуговування виробу. Існують так види випробувань на надійність [6]:

1) початкові, в результаті яких визначають показники надійності;

2) понтрольні випробування направлені контроль якості технологічного процесу;

3) прискорені випробування в яких використовують чинники прискорюючі виникнення відмови;

4) неруйнівні випробування використовують методи дефектоскопії і інтроскопії, а також на досліджені ознак супутніх виникненню відмов.

Моделювання на ЕОМ є найефективнішим методом діагностування надійності складних систем. Найбільш поширені два методи моделювання: моделювання фізичних процесів і вирішення рівнянь. Перший метод моделює фізичні процеси які відбуваються в досліджуваному об'єкті. Другий метод використовує вирішення рівнянь які описують стан досліджуваного об'єкту.

Оцінку надійності також можна отримати при аналізі фізико – хімічних процесів через те що доволі часто вдається встановити залежність надійності від характеру і стану фізико – хімічних процесів(показники міцності, зносостійкості, шумові ефекти, наявність домішок в матеріалах і інше).Цей метод оцінювання здебільшого використовується при оцінені надійності елементів радіоелектронної апаратури.

Більшість користувачів при роботі за комп'ютером ніколи не задумувалися над тим, що одного разу він може вимкнутися раз і назавжди. Досить часто можуть виникати проблеми і з новим, щойно зібраним комп'ютером. Він може не вмикатися чи раптово припинити роботу. В таких випадках головне не робити поспішних висновків, а правильно діагностувати причину несправності. Так в деяких випадках комп'ютер може і не потребувати ремонту. Спочатку потрібно з'ясувати можливі причини які призводять до цієї несправності. Наприклад пил чи інші чинники можуть погіршувати, чи впливати на стан комплектуючих комп'ютера. Тому несправність апаратних елементів комп'ютера можуть бути наслідком окислення контактів, потрапляння пилу на на роз'єми чи мікросхеми і порушення температурних норм.

Причиною несправності можуть стати стрибки напруги, неправильне заземлення або ж збої у роботі блоку живлення. Один з методів запобігання цих несправностей це використання мережевих фільтрів. При виникненні несправності

в першу чергу потрібно провести візуальний огляд комп’ютера, знати кришку і перевірити чи немає в комплектуючих явних візуальних відхилень. Перевірити чи немає запаху диму, саме запах диму здебільшого є наслідком проблем з стрибками напруги. При відсутності явних візуальних відхилень потрібно перевірити правильність і надійність підключення до живлення. Якщо при перевірці не було знайдено несправностей то можна включити комп’ютер і перевірити роботу кулерів. Якщо кулери не працюють то можна зробити висновок що проблема у блоці живлення.

Щоб перевірити наявність напруги на виході блоку живлення можна за допомогою тестера поміряти напругу на контактах де системна плата зєднується з блоком живлення. Для точної перевірки можна підключити інший, задовольняючий вимоги, блок живлення і перевірити правильність роботи інших елементів комп’ютера. Також необхідно перевірити і монітор, хоч і вони ламаються рідше. Для перевірки можна використати осцилограф для вияснення чи надходять з відеоадаптера сигнали на монітор [7].

Комплекс програмних, мікропрограмних і апаратних засобів представляють собою систему автоматичної діагностики. Розрізняють дві системи діагностики: тестову і функціональну. При тестовій діагностиці на діагностуючий пристрій результати надходять від засобів діагностування. Для діагностики середніх і великих ЕОМ зазвичай використовують спеціалізовані методи діагностування. В мікро машинах використовують вбудовані засоби які надсилають результати на зовнішні засоби. Сама діагностика включає в себе декілька етапів що характерезуються робочою напругою, що знімається з відповідних пристроїв. Оскільки існує безліч різновидів несправностей немає єдиного підходу до діагностики комп’ютера. Є певний перелік несправностей які пов’язані з апаратною частиною:

1) комп’ютер не вмикається;
2) комп’ютер вмикається, але на екрані немає інформації;
3) комп’ютер вмикається і видає специфічні звуки (“пищить”);

4) різноманітні помилки, пов’язані зі стартом BIOS;

5) комплектуючі не функціонують;

6) комп’ютер перегрівається;

7) апаратна несумісність обладнання;

8) система працює дуже повільно.

Очевидно що для діагностики комп’ютера можна використовуватися фізичні властивості людини проте сьогодні існує безліч програмних засобів для діагностування які є досить популярні серед користувачів. Програмні засоби для діагностування використовують певний набір команд які надсилаються до кожної складової і повертають певний результат у відповідне діалогове вікно. Ці засоби діагностики для знаходження проблеми звіряють наявні дані складових з переліком даних для нормальної роботи складових. Дізнатися характеристики комп’ютера можна і без додаткового програмного забезпечення за допомогою стандартних засобів Windows, наприклад диспетчер задач(рисунок 1.5)

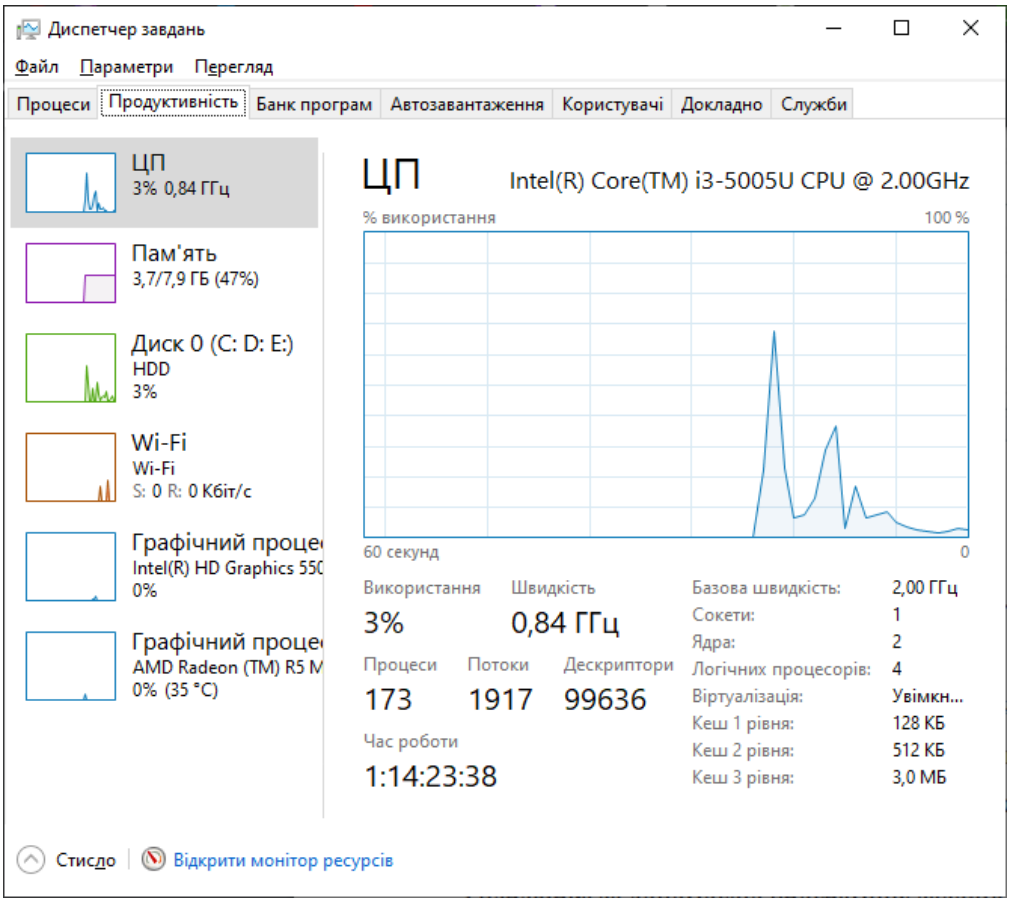


Рисунок 1.5 – Поточні характеристики комп’ютера

Сьогодні для діагностики комп’ютера існує безліч методів діагностики від програмних до спеціалізованих веб-ресурсів. Але загальні методи діагностики є спільними і обов’язковими для всіх методів. Насамперед це обов’язкова перевірка усіх складових на наявність фізичних дефектів, також програмна перевірка для знаходження проблем які можуть стосуватися програмного коду.

Аналіз комп’ютера є актуальною темою для всіх користувачів. Саме вчасна діагностика дозволяє уникнути безлічі небажаних проблем які можуть призвести до ремонту. Тому в діагностиці немає не важливих частин, а перевіряються всі навіть найменш значущі елементи. Це і стосується перевірки додатків на віруси, бо вони можуть ховатися в найменших файлах і в той же час нанести велику і безповоротну шкоду комп’ютеру. Часто такі проблеми є дуже неприємними через важкість їх виправлення.

1.3 Основні поняття нечіткої логіка

Нечітка логіка, або так звана “Fuzzy logic” є досить популярною серед програмістів які працюють з системами управління через те, що є зручним засобом для програмування і моніторингу додатків управління технологічними процесами. Вперше термін нечітка логіка був використаний в 1965 році професором університету Берклі в Каліфорнії Лютфу Заде. Його твердження заключалось в тому, що звичайна комп’ютерна логіка не може маніпулювати даними, які являють собою суб’єктивні чи незрозумілі людські ідеї.

Нечітка логіка широко розповсюджена в різних областях від медицини до теорії управління. Вона створена для того щоб комп’ютер відрізняв дані які не є хибними і не є точними. Нечітку логіку можна порівняти з міркуваннями людини. Так як і традиційні засоби керування технологічним процесом нечітка логіка використовує для опису петлі регулювання і бере участь у обчислені керуючого впливу відповідно до однієї або точок завдання одного чи більшої кількості вимірів [8].

Нечітка логіка керується такими правилами:

1. Застосування існуючого досвіду.

2. Якщо неможливо очно змоделювати систему використовуються гнучкі провила.

Для покращення якості управління використовуються саморегулювання системи і випереджаюча зміна вхідного плану, що базується на подіях які не можуть бути враховані у традиційних способах управління.

Використання нечіткої логіки з кожним роком зростає. Так на сьогоднішній день кількість додатків які використовують даний метод збільшується для безперервних процесів, для автоматизованих систем та для додатків пакетної обробки. Саме завдяки використанню нечіткої логіки в цих галузях вона отримала формулювання в якості методу програмування. Вона може систематизувати емпіричні знання і коли при використані класичних методів управління виникають проблеми нечітка логіка може їх вирішувати. Теорія нечіткої логіки надає можливість описувати набори методів управління, які з легкістю застосовуються в реальних системах і можуть враховувати досвід операторів і використовувати його для динамічного управління. Ці умови дозволяють використовувати нечітку логіку для опису окремих процесів таких як ініціалізація та задання параметрів.

Нижче перераховано деякі важливі характеристики нечіткої логіки:

1) дає змогу наслідувати логіку подібну до людських роздумів;

2) техніка машинного навчання є більш гнучкою;

3) дає можливість побудови нелінійних функцій різної складності;

4) є ефективним рішенням при вирішені приблизних або невизначених міркувань;

5) логіка може містити декілька значень, які у свою чергу представляють декілька рішень.

При виконані інженерних задач здебільшого використовується механізм нечіткого виводу Мамдані, який можна реалізувати за допомогою середовища Matlab. Саме метод Мамдані одним із перших будувався на основі теорії нечітких

множин. Метод було названо в честь англійського математика Ібрагіма Мамдані який запропонував цей метод в 1975р.

Метод Мамдані являє собою нечітку систему виведення (далі НСВ).НСВ-це система яка бере за основу використання теорії нечітких множин для відображення входів до виходів.

Метод містить в собі такі етапи як:

1) формування бази правил;
2) фазифакція;
3) агрегування підумов;
4) активізація підвисновків;
5) активізація висновків;
6) дефазифакція.

Всі етапи виконуються послідовно вхідні дані кожний етап отримує значення які були результатом попереднього. Вхідними ж значеннями для методу Мамдані є певна база правил. Нижче зображена загальна схема роботи нечіткої логіки

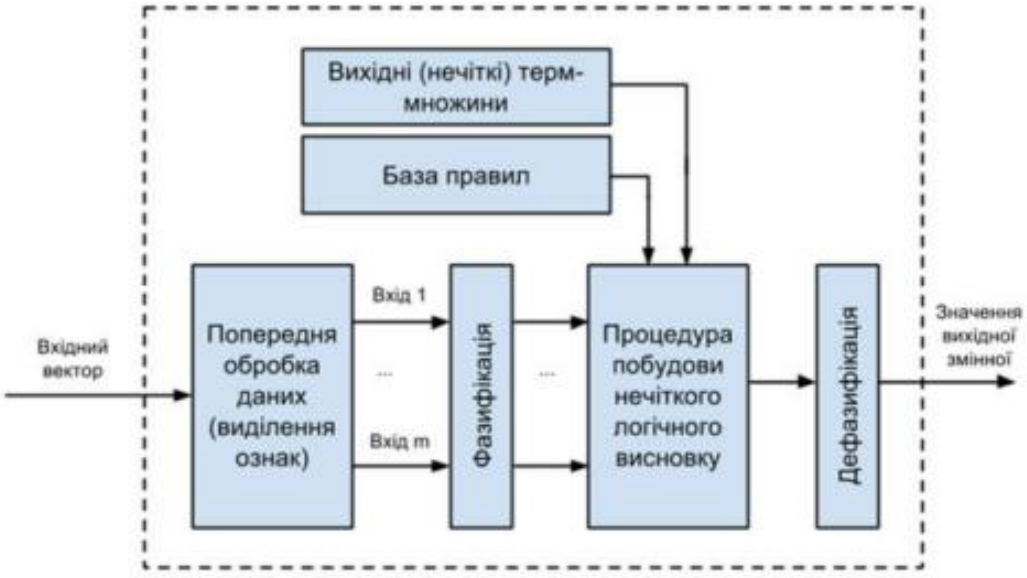


Рисунок 1.6 – Загальна схема роботи нечіткої логіки

До прикладу правилами для методу можуть слугувати такий запис

Якщо $x \epsilon Y1$, тоді $z \epsilon N1$,

Де x – вхідне значення;

z – вихідне значення;

Y і N – нечіткі множини.

Переведенням вхідних змінних до нечіткості називають фазифікацією. Вхідними даними є вже сформована база правил і деякий масив вхідних значень $A=\{a_1 \ldots a_n\}$. Такий вхідний масив містить в собі значення всіх вхідних зміних. Метою даного етапу є отримання певного значення істинності для всіх умов які містяться в базі правил. Ці значення отримуються наступним чином: для кожної умови вираховується $b_i = \mu(a_i)$. Так в результаті ми отримаємо безліч значень b_i ($i = 1 \ldots k$).

Під час етапу агрегування умов відбувається визначення ступеню інтенсивності умов для кожного правила системи нечіткого виведення. Як це відбувається: для кожної з умов знаходиться якесь певне мінімальне значення істинності всіх її підумов. Взагальному це має таки вигляд (формула 1.1).

$$C_j = \min\{ b_i \} \qquad (1.1)$$

Де $j = 1 \ldots q$, i – число з безліччю номерів підумов, в яких в яких бере участь j.

Після етапу агрегування умов проходить етап активізації виводів. Призначення даного методу є провести перехід від умов до підвиводів. Всі ці дії відбуваються наступним чином: для кожного підвивода вираховується ступінь істиності $d_i = c_i * F_i$ де ($i = 1 \ldots k$). Після цього для кожного i – го підвиводу протиставляється безліч D_i з зовсім іншою, новою, функцією належності. Значення цієї функції визначається як мінімум d_i і значення функції належності терму з підвиводу. Даний етап також називають min – активізацією і виглядає він так(формула 1.2):

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

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

$\mu(x)$ – функція належності терму;

d_i – ступінь істинності і–го підвиводу.

Взагльному метою цього етапу є отримання для кожного з підвиводів бази правил суми «активованих» нечітких множин.

Етап акумуляції виводу. На даному етапі отримуються нечіткі множини для кожної з усіх вихідних змінних. Для кожної і-тої вихідної змінної зіставляється об'єднання множин $E_i = \cup D_j$. При цьому j – це номер підвиводу де бере участь і-та вихідна зміна ($i = 1 \ldots s$) [9].

Результатом об'єднання двох нечітких множин буде третя з такою функцією приналежності (формула 1.3):

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

де $\mu_1(x)$, $\mu_2(x)$ є функціями приналежності для множин які поєднуються.

Дефазифакція. Метою даного етапу є отримання деякого кількісного значення для кожної з наявних вихідних лінгвістичних змінних. Відбувається це все наступним чином: береться якась і-та вихідна змінна і відповідна їй множина E_i (1…s). Після з використанням методом дефазифакції знаходиться кінцеве кількісне значення для вихідної змінної. При даній реалізації цього алгоритму використовується метод центру тяжіння, де значення і-тої вихідної змінною розраховується за формулою зображеною нижче (формула 1.4)

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

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

min та max – координати універсуму нечітких змінних, y_i результат дефазифакції.

Перевагами нечітких логічних систем є:

1) простота і зрозумільсть структур;

2) розповсюдженість нечіткої логіки в практичних і комерційних цілях;

3) надає змогу контролювати машини та споживчі товари;

4) надає змогу вирішити невизначені питання в техніці;

5) надійність, через можливість прийому не точних входів;

6) проста в заміні для вдосконалення чи пришвидшення продуктивності;

7) ефективна у вирішені складних питань.

Але нечіткі системи мають і ряд своїх недоліків:

1) точність. Не завжди результати є точними, а сприймаються на основі припущень;

2) відсутність можливості машинного навчання;

3) необхідність встановлення точних, нечітких правил є досить складним завданням;

4) прирівнювання нечіткої логіки до теорії ймовірності.

1.4 Постановка задачі

Судячи з цієї інформації можемо сказати, що не чіткі системи справді є досить зручними для використання у моделюванні складних систем які тісно пов’язані з ЕОМ. Ці системи мають ряд таких переваг:

1) керування системи може виконуватися і даними які не є чіткими, або ж динамічними які постійно змінюються в часі, або даними які неможливо задати однозначно;

2) змога керувати нечіткими критеріями оцінки, наприклад “можливо”,“здебільшого”, та інші;

3) проведення оцінки всіх даних як вхідних так і вихідних;

4) висока швидкість моделювання складних динамічних систем та їх аналіз із вказаною точність. Це завдяки описаним fuzzy-методам які, по-перше не

затрачають час на визначення точних значень змінних, по-друге оцінюють різні варіанти які можуть мати вхідні змінні.

1.5 Висновок до першого розділу

У даній роботі перший розділ несе аналітичний характер. Цей розділ містив в собі такі поняття як комп'ютерна система, види архітектур комп'ютерних систем, їх спільні і відміні риси, переваги та недоліки кожної з них.

Було розглянуто поняття стану комп'ютерної системи. Основні характеристики які впливають на стан комп'ютера. Основні несправності та причини їх виникнення.

Поняття нечіткої логіки теж було описано та проаналізовано у даному розділ. Було наведено основні формули за якими працює нечітка логіка, і приведено переваги використання саме такого методу. Розглянуто наявні засоби які працюють на основі нечіткої логіки.

2 АНАЛІЗ СТАНУ КОМП'ЮТЕРНОЇ СИСТЕМИ ЗА ДОПОМОГОЮ НЕЧІТКИХ АЛГОРИТМІВ

2.1 Дослідження існуючих програмних засобів на основі нечіткої логіки

Сьогодні для вирішення подібних завдань використовуються методи штучного інтелекту, такі як нечіткі системи та нечіткі-нейроні системи.

При використані нечіткої логіки стає можливим природно виражати поняття, що використовується користувачами та експертами.

У цій роботі для аналізу і діагностики технічного стану комп'ютера пропонується використовувати експертну діагностичну систему, або скорочено ЕЦП. Деякий математичний апарат, який надає змогу керувати оцінкою сану об'єкту над яким проводиться діагностика, є нечіткою логікою. Під час підготовки діагностичного експерименту рекомендовано описати діагностичні особливості комп'ютера з погляду можливих лінгвістичних змінних, які надають можливість використання досвіду та знань експерта у вже знайомій формі. На рисунку 2.1 зображено принцип роботи схожої системи [10].

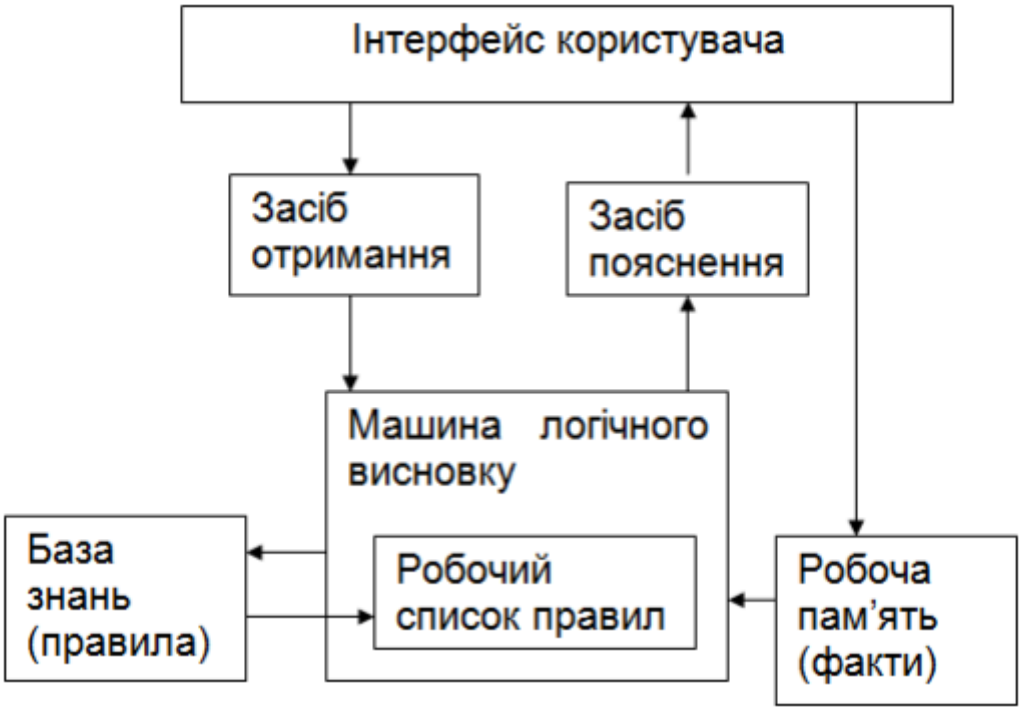


Рисунок 2.1 – Принцип роботи експертної системи

При використані діагностики на основі експортних систем зменшує проблеми пов'язані з діагностуванням апаратних збоїв. Тому і була запропонована дана система щоб покращити діагностику апаратних збоїв. З цією системою користувач не повинен окремо перевіряти різні частини обладнання, все що їм потрібно це ввести в дану систему наявні симптоми та апаратне забезпечення комп'ютерної системи і провести діагностику. Розробка даної діагностики проблем з апаратурою комп'ютера базується на експортній системі яка складається з таких етапів [11]:

1) аналіз та дослідження;
2) оцінка проблеми;
3) концептуалізація;
4) накопичення та аналіз знань;
5) впровадження та проектування;
6) тестування;
7) управління та документація.

Система побачить та відображатиме можливі причини проблеми і запропонує варіанти вирішення. Всі правила даної експертної системи розкриті у формі тверджень “if-then”. Категоріями даної системи є: клавіатура, аудіо, процесор, миша, блок живлення, жорсткий диск, принтер, серійний ATA, BIOS, пристрій USB, оперативна пам’ять, материнська плата, DVD привід, та інша периферія.

Отже мета даної експертної системи є в тому щоб полегшити користувачеві діагностику несправностей комп’ютерних систем. Ще дана система пропонує методи усунення деяких основних проблем, або навіть усунення більш масштабних несправностей перш ніж направитися за допомогою до служби підтримки чи фахівців.

Синтез діагностичних моделей основаних на нейро-нечіткій моделі з хешуванням перетворень в послідовних і паралельних режимах. Для побудови нейро-нечітких мереж на основі прецедетів було вирішено завдання з підвищення швидкості.

Метою слугує розробка і створення нейро-нечіткого методу синтезу з високою швидкістю і знайти змогу реалізації нейро-нечітких мереж у паалельних методах.

Побудова нейро-нечітких моделей на основі прецедентів, які дають змогу зменшити розміри вхідних даних за допомогою хешування що спричиняє перетворення даних в одну вісь. Через це зберігається локальна топологія кластерів у функціональному просторі, розділ вихідного простору функцій утворюється автоматично, налаштування параметрів нейро-нечіткої моделі відбувається автоматично, синтезується структура. При налаштуванні моделі автоматично видаляє з навчального процесу дані які не є інформативними, за допомогою чого структура моделі спрощується. Метод надає змогу більшість обчислювальних операцій виконувати у паралельному режимі, це дає змогу зробити синтез нейро-нечіткої моделі автоматизованим і підвищити побудову як паралельних так і послідовних обчислень.

На основі цього методу було розроблено і протестовано програмне забезпечення. Тести підтвердили ефективність даного методу. Також даний метод може використовуватися на практиці під час діагностування компютерних систем [12].

Ознака зручності користування використовуючи нечітку систему. Одним із найважливіших етапів розробки програмного забезпечення є і його оцінка. Це дозволяє покращувати і вдосконалювати його. Багато різних факторів впливає на оцінку програмного забезпечення. Одним з таких факторів є якість. Цей фактор є досить складним у визначені оскільки залежить від ряду інших факторів. Наприклад працездатність цей фактор напряму впливає на якість програмного забезпечення. Існує ряд моделей зручності програмного забезпечення і кожне з них має свої фактори.

На практиці всі ми так чи інакше стикалися з перешкодами у реалізації моделі зручності користування і основна причина цього відсутність точного визначення зручності.

У цьому проекті була запропонована узагальнена модель зручності користування. Також було запропоновано методи визначення зручності за допомогою нечіткої логіки.

2.2 Дослідження дерева рішень

Як було визначено для діагностики комп'ютерної системи необхідно враховувати параметри системи, наприклад допустимі затрати пам'яті, продуктивність та інше, тому було обрано використовувати апарати нечіткої логіки для вирішення даної задачі.

Нечітка логіка і теорія нечітких множин є узагальненням класичної теорії множин і класичної логіки. Ці поняття вперше з'явилися в 1965 році, їх запропонував американський учений Лотфі Заде. Причиною яка спонукала до

появи цих термінів стала наявність наближених і нечітких міркувань які виникають при описі людиною систем, процесів, об’єктів.

Нечіткі системи володіють певними перевагами в порівняні з іншими. Основні з них:

1) нечіткі системи мають змогу оперувати даними які не є чітко визначеними, або які невпинно змінюються;

2) наявність можливості нечітко описувати критерії і порівняння;

3) проведення якісних оцінок вхідних і вихідних даних;

4) висока швидкість моделювання складних динамічних систем та їх аналіз із вказаною точність.

При реалізації інженерних задач здебільшого використовують метод нечіткого висновку Мамдані. Він реалізований за допомогою мінімаксної композиції нечітких множин, і містить в собі наступні етапи:

1. Фазифікація. Вираховуються степені істинності, або ж, якщо точніше, значення функцій належності $M\ F_i(x)$ для кожного і-го правила.

2. Нечіткий висновок. Для початку вираховуються мінімальні в “відсічення” для лівої частини кожного з правил $A_i = \min(M\ F_i(x))$, після цього вираховують “усічені” функції належності висновку $B_i = \min(A_i, B_i)$.

3. Об’єднання “усічених” функцій, або ж композиція.

4. Приведення до чіткості, дефазифакція.

Дефазифакція існує в декількох методах, наприклад центроїдний метод і метод середнього центру. Геометричним змістом такого методу слугує центр ваги для кривої функції належності наявного виходу [13].

Якщо використовувати апарат нечіткої логіки для створення програмного засобу який буде діагностувати комп’ютер шляхом вибору оптимального методу для кожної з характеристик та враховуючи поточні властивості системи то це забезпечить високу стійкість для аналізу в реальному часі.

Для вирішення такого завдання потрібно:

1. Визначити і дослідити основні методи діагностування комп’ютерних систем і перевірити їх стійкість до часового аналізу.

2. Розробити метод який на основі нечіткої логіки буде обробляти інформацію всіх характеристик системи під час діагностики комп’ютера.

3. Розробити структуру для реалізації розробленого методу та провести дослідження його характеристик.

Рисунок 2.2 – дерево рішень дипломного проекту

Проектування схеми проведення подальшої роботи проводиться для того щоб досягнути найменшої кількості неточностей, що в свою чергу приводить до того, що результати досягаються в найкоротші терміни. Розроблена схема буде основою подальшої розробки. Завдяки цьому і точно поставленій задачі буде зекономлено вулику кількість ресурсів і часу [14].

2.3 Алгоритм Мамдані

Центральне місце у системах нечіткого моделювання займає нечіткий вивід. Суть нечіткого виводу характеризується певною процедурою або ж алгоритмом який отримує нечіткі виводи шляхом використання нечітких передумов які використовують основи нечіткої логіки. Нечіткий вивід має такі основні етапи:

1) визначення структури системи нечіткого виведення;
2) формування бази правил системи нечіткого виведення;
3) фазифікація вхідних змінних;
4) обрахунок значень ступенів належності підумови правил нечітких продукцій;
5) генерація підумов правил нечітких продукцій;
6) активізація підумов правил нечітких продукцій;
7) акумулювання виводів правил нечітких продукцій;
8) дефазифація вихідних змінних.

Одною з ключових осіб у формуванні терміну нечіткої логіки є Лотфі Заде. Написавши в 1965році “Fuzzy Sets”, фундаментальну працю нечіткої логіки, Лотфі Заде вважається засновником цілої наукової теорії. Неменшим вкладом є те, що він опрацьовував різні можливості використання цього методу. Свій новий підхід він описав в 1973 році в статі «Outline of a New Approach to the Analysis of Complex Systems and Decision Processes» яка була опублікована в журналі IEEE Transactions on Systems. Ознако. Якості цього методу було те, зо одна з датських фірм змогла успішно використати принципи описані Лотфі Заде і вдосконалити свою систему управління процесом.

Але не тільки один Л. Заде розвивав цю теорію, його послідовники відігравали не менш важливу роль у розвитку теорії. До прикладу Е. Мамдані. Цей англійський математик розробив алгоритм який був обраний як метод управління для парових двигунів. Даний метод базувався на основі нечітких виводів, це дозволило обійти великі обсяги обчислень. Метод був високо оцінений іншими

фахівцями. В сьогоденні ж цей алгоритм отримує своє практичне застосування в задачах нечіткого моделювання. Для подальшого ознайомлення з методом спочатку потрібно дізнатися про наступні визначеннями [15].

Кортеж виду <α, X, A > називають нечіткою змінною,

де α – назва нечіткої змінної;

X – область визначення змінної;

A – нечітка множина на універсумі X.

Наприклад розглянемо нечітку змінну «Системний блок». <«Системний блок», {x | 0 кг < 0 <35 кг}, B={x, μ(*x*)}> описана зміна буде характеризувати масу системного блоку. Припустимо, що якщо маса системного блоку більша 16 кілограм тоді ми вважаємо, що він важкий (рисунок 2.3).

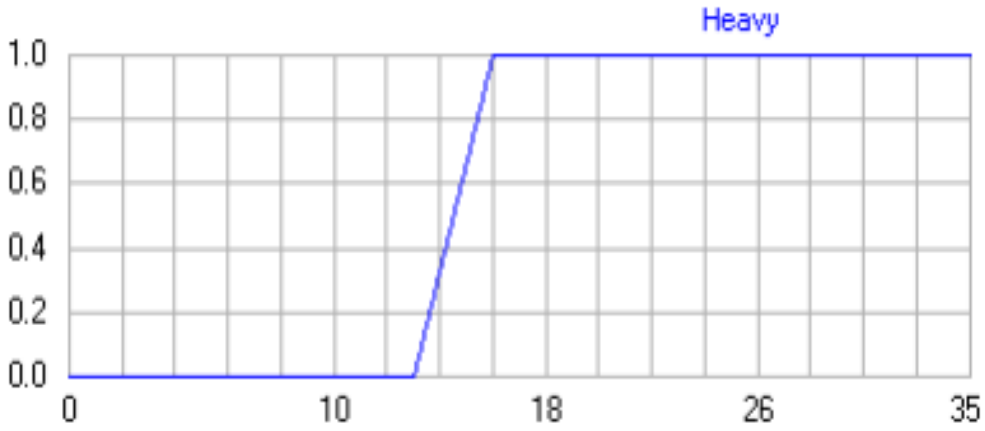


Рисунок 2.3 – Графік функції приналежності μ(*x*) для B

Кортеж типу <β, T, X, G, M> називають лінгвістичною змінною,

де β – слугує назвою лінгвістичної змінної;

T – безліч її значень;

X – універсум нечітких змінних;

G – процедура формування нових термінів;

M – процедура формування нечітких множин для кожного з термім змінної.

Щоб на практиці продемонструвати цю змінну розглянемо наступний приклад. Наприклад ми маємо деяку суб'єктивну оцінку маси системно блоку, цю

оцінку ми можемо отримати до прикладу від працівників магазину техніки, які щоденно мають справу з схожою технікою. Ми можемо за допомогою лінгвістичної змінної формалізувати цю оцінку [16].

1. β – Системний блок.
2. T – {«Легкий(Light)», «Середньої важкості(Medium)», «Важкий(Heavy)»}.
3. X – [0;35].
4. G – формування нових термінів відбувається за допомогою ддавання логічних зв’язків або мдифікаторів наприклад «надзвичайно легкий».
5. M – задання на універсумі X =[0; 35] значень лінгвістичної змінної або ж термів з T.

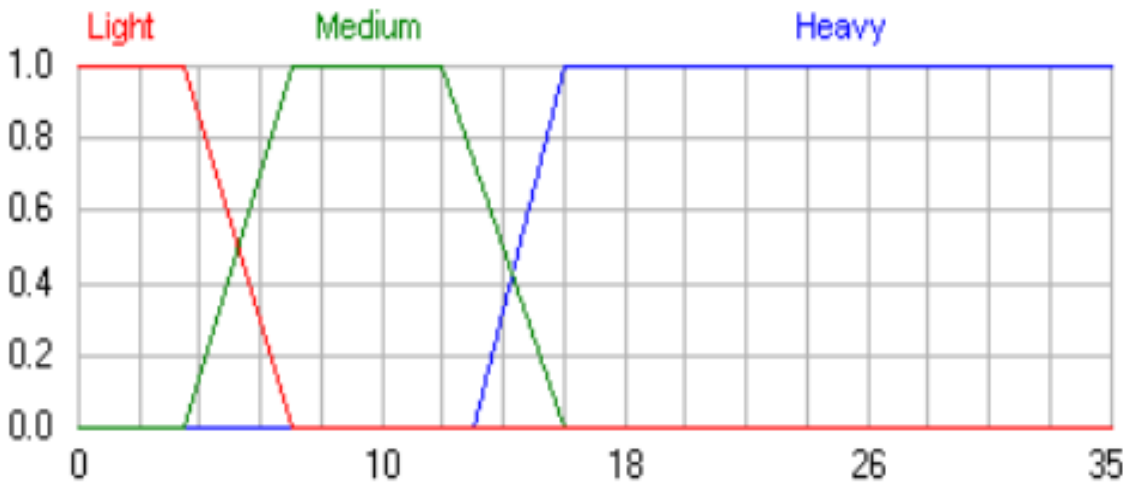


Рисунок 2.4 – Графік належності значень лінгвістичної змінної «Системний блок»

Тепер перейдем до огляду самого алгоритму Мамдані. Цей алгоритм описує декілька етапів які послідовно виконуються (рисунок 2.5). До того ж кожний наступний етам отримує на свої входи значення які були отримані в хоід виконання попереднього.

Рисунок 2.5 – Діаграма роботи нечіткого виведення

Цей алгоритм вирізняється тим що працює по принципу «чорного ящику». На входи поступають кількісні значення, і на виходах отримуємо також кількісні значення. Для обчислення проміжних результатів використовуються нечітка логіка і теорія нечітких множин. Це і є особливість нечітких систем. Можна використовувати і маніпулювати числовими даними, і водночас використовувати всі можливості які надає система нечіткого виводу.

Кожна з правил (Rule) містять в собі умови (Condition), а також висновки (Conclusion), які вже в свою чергу слугують нечіткими висловлюваннями (Statement). Лінгвістична зміна (Variable) є частиною нечіткого висловлювання, також висловлювання містить в собі терм який приставлений нечіткою безліччю (FuzzySet). Щоб отримати значення функції приналежності можна використати функцію getValue(). Даний метод знаходиться і створений інтерфейсом FuzzySetIface.

Щоб реалізувати даний алгоритм підійде використання об’єктно-орієнтованого підходу. Отже ми знаємо , що етапи виконуються послідовно і вхідні значення етапу це результат роботи попереднього.

Код створений за допомогою мови програмування Java, алгоритми дії якого зображений на діаграмі (рисунок 2.6) зображує основні зв’язки між класами задіяними в алгоритмі [17].

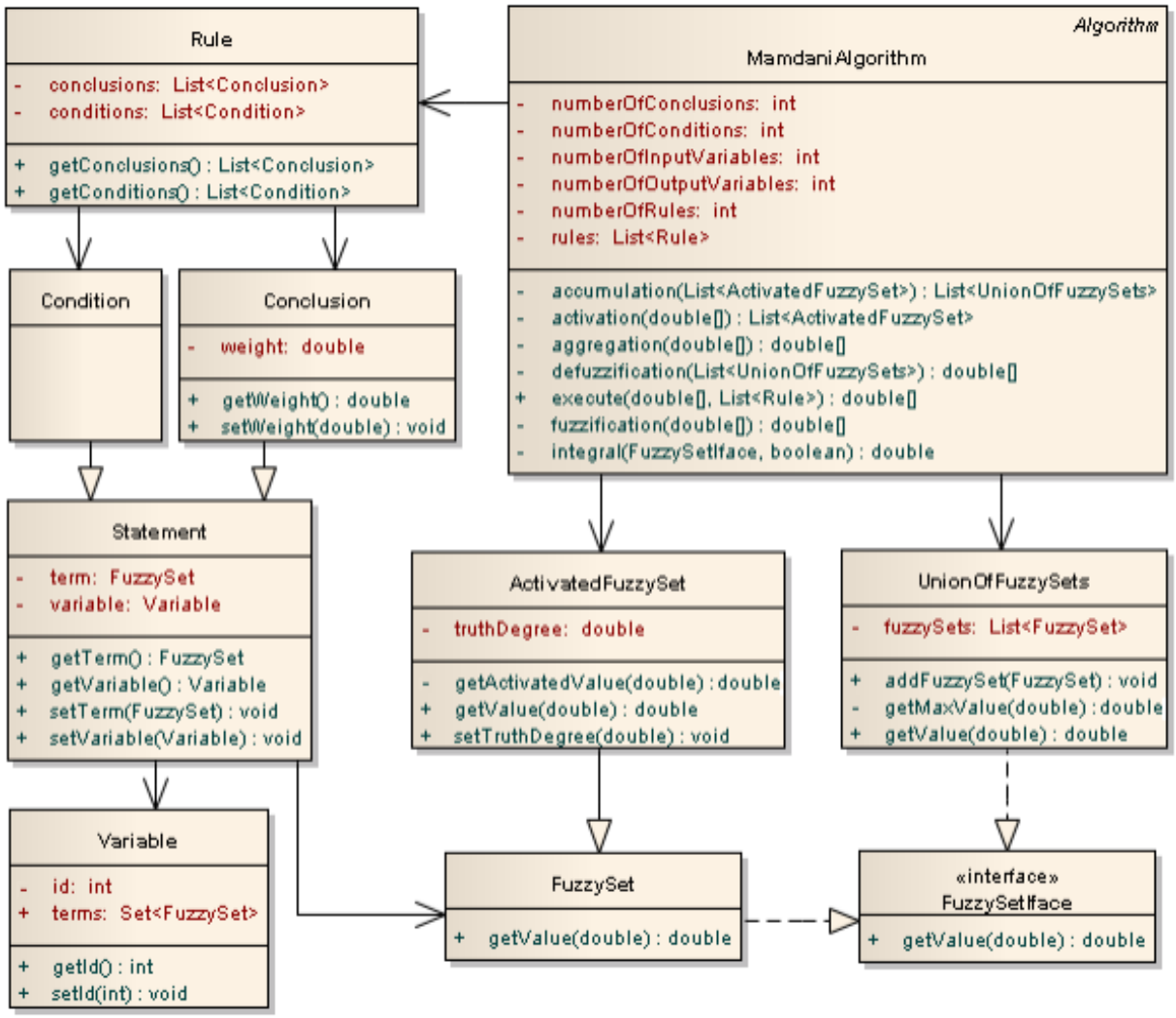


Рисунок 2.6 – Діаграма класів реалізованого алгоритму Мамдані

Коли виконується алгоритм потрібно використовувати «активізовану» нечітку множину (ActivatedFuzzySet), яка перевизначає функцію приналежності нечіткої множини (FuzzySet). Алгоритм також використовує об’єднання нечітких множин (UnionOfFuzzySets) яке в свою чергу є також нечіткою множиною [18].

Алгоритм створений Мамдані містить в собі всі етапи (рисунок 2.5) також використовує базу правил (List <Rule>) як вхідні дані. Також в алгоритмі наявна можливість використання «активізованих» множин (ActivatedFuzzySet) та їх об’єднань (UnionOfFuzzySets).

Першим етапом алгоритму є етап формування бази правил. База правил – це велика кількість правил в яких кожному підвисновку протиставлений якийсь ваговий коефіцієнт. База правил може бути приставлена в такому вигляді:

1. RULE_1: IF «Condition_4» AND «Condition_5» THEN «Conclusion_4» (F2).

2. RULE_2: IF «Condition_1» AND «Condition_2» THEN «Conclusion_5» (F4) AND «Conclusion_1» (F1).

3. RULE_3: IF «Condition_3» THEN «Conclusion_7» (F6) OR «Conclusion_1» (F1).

4. RULE_n: IF «Condition_k» OR «Condition_k» THEN «Conclusion_q-1» (Fq-1) AND «Conclusion_q-1» (Fq-1).

Де Fq – це коефіцієнт який вказує ступінь впевненості в істинності підвисновку (i=1..q).За замовчуванням даний коефіцієнт прийнято вважати рівним 1. В присутніх умовах лінгвістичні змінні називають вхідними, а в заключних умовах вихідними.

Використані позначення і їх значення:

1) n – кількісті правил нечітких продукцій (numberOfRules);
2) m – число вхідних змінних (numberOfInputVariables);
3) s – число вихідних змінних (numberOfOutputVariables);
4) – k – загальне число підумов в базі правил (numberOfConditions);
5) q – загальна кількість висновків в базі правил (numberOfConclusions).

Другим етапом алгоритму є фазифікація, або як ще його називають приведення до нечіткості. Вхідними даними є вже сформована база правил і деякий масив вхідних значень $A=\{a_1 \ldots a_n\}$. Такий вхідний масив містить в собі значення всіх вхідних змінних. Метою даного етапу є отримання певного значення істинності для всіх умов які містяться в базі правил. Ці значення отримуються наступним чином: для кожної умови вираховується $b_i = \mu(a_i)$. Так в результаті ми отримаємо безліч значень b_i ($i = 1 \ldots k$).

Під час етапу агрегування умов відбувається визначення ступеню інтенсивності умов для кожного правила системи нечіткого виведення. Як це відбувається: для кожної з умов знаходиться якесь певне мінімальне значення істинності всіх її підумов [19]. Взагальному це має таки вигляд (формула 2.1).

$$C_j = \min\{ b_i \} \quad (2.1)$$

Де $j = 1…q$, i – число з безліччю номерів підумов, в яких в яких бере участь j.

Після етапу агрегування умов проходить етап активізації виводів. Призначення даного методу є провести перехід від умов до підвиводів. Всі ці дії відбуваються наступним чином: для кожного підвивода вираховується ступінь істиності $d_i = c_i * F_i$ де ($i = 1…k$). Після цього для кожного i – го підвиводу протиставляється безліч D_i з зовсім іншою, новою, функцією належності. Значення цієї функції визначається як мінімум d_i і значення функції належності терму з підвиводу. Даний етап також називають min – активізацією і виглядає він так(формула 2.2):

$$\mu_i' = \min\{ d_i, \mu(x) \} \quad (2.2)$$

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

$\mu(x)$ – функція належності терму;

d_i – ступінь істинності і–го підвиводу.

Взагльному метою цього етапу є отримання для кожного з підвиводів бази правил суми «активованих» нечітких множин.

Етап акумуляції виводу. На даному етапі отримуються нечіткі множини для кожної з усіх вихідних змінних. Для кожної і-тої вихідної змінної зіставляється об’єднання множин $E_i = \cup D_j$. При цьому j – це номер підвиводу де бере участь і-та вихідна зміна ($i = 1…s$).

Результатом об’єднання двох нечітких множин буде третя з такою функцією приналежності (формула 2.3):

$$\mu_i'(x) = \max\{ \mu_1(x), \mu_2(x) \}, \quad (2.3)$$

де $\mu_1(x)$, $\mu_2(x)$ є функціями приналежності для множин які поєднуються.

Дефазифакція. Метою даного етапу є отримання деякого кількісного значення для кожної з наявних вихідних лінгвістичних змінних. Відбувається це все наступним чином: береться якась і-та вихідна змінна і відповідна їй множина E_i (1…s). Після з використанням методом дефазифакції знаходиться кінцеве кількісне

значення для вихідної змінної. При даній реалізації цього алгоритму використовується метод центру тяжіння, де значення і-тої вихідної змінною розраховується за формулою зображеною нижче (формула 1.4)

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

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

min та max – координати універсуму нечітких змінних, y_i результат дефазифакції.

Використання цього методу досить обґрунтоване і базується на твердженні Кастро який в 1995 році показав, що для семетричних трикутних функцій метод Мамдані є універсальним і може застосовуватися у всійх галузях промисловості чи науки.

2.4 Алгоритм Сугено

У даному підрозділі розглядається алгоритм Сугено, або як його ще називають Такагі-Сугено-Кан, метод нечіткого умововиводу. Вперше використаний в 1985 році і має багато спільного з методом Мамдані. Спільні речі між цими методами це нечіткий процес виводу, застосування нечіткого оператора та нечіткі виходи. Відмінність між цими методами полягає в тому, що на відміну від методу Мамдані в методі Сугено функції виведення або лінійні або ж постійні [20].

Одне з типових правил в методі Сугено виглядає так:

За умови коли Input 1 = x і Input 2 = y, вихідним є z = ax + c

Вхідна змінна z для нульового порядку моделі Сугено є постійа (a=b=0).

Рівень кожного правила і рівень виходу z зважується на міць випалу w-правила. Розглянемо приклад даної ситуації, наприклад правило AND Input 1 = x і Input 2 = y, для цього правила керуючим процесом є w_1 = AndMethod (F1(x), F2(y)).

Принцип дії правила Сугено продемонстровано на насупній схемі (рисунок 2.7)

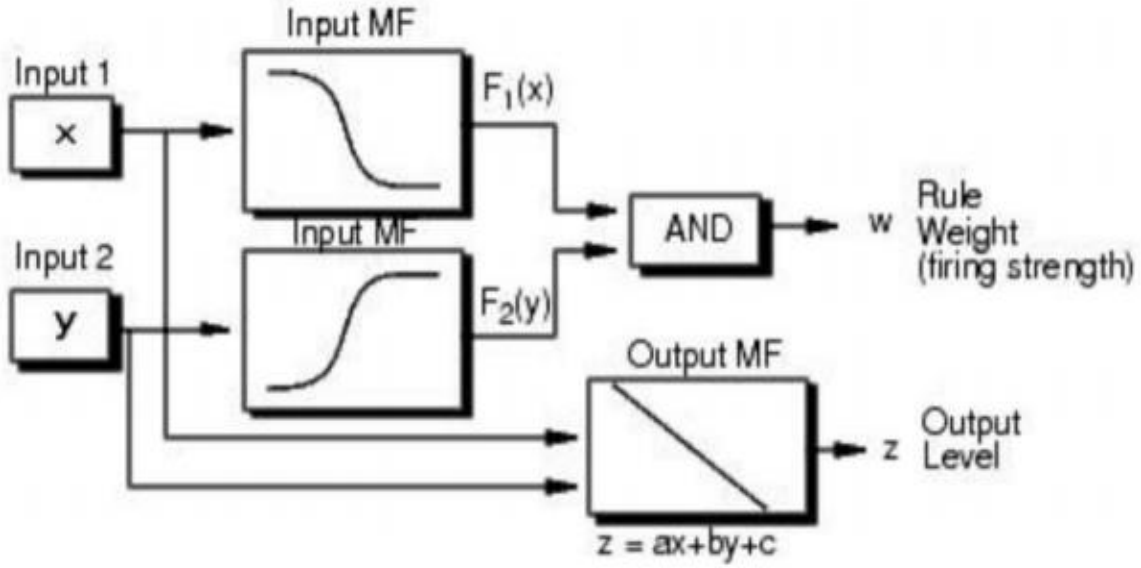


Рисунок 2.7 – Схема роботи алгоритму Сугено

А що ж до нечітких моделей Сугено вищого порядку? Вони можливі з не очевидними перевагами і певними складнощами. Оскільки метод Сугено має особливість пов'язану з лінійною залежністю правил від вхідних змінних, він ідеально вписується в роботу як інтерполяційний керівник декількох лінійних контролерів, які необхідно по різному використовувати в залежності від вхідних умов.

Надзвичайно добре використовувати систему нечітких висновків Сугено для завдання яке полягає в плавній інтерполяції лінійних посилань, які використовуються через вхідний простір. Це є ефективним і природним планувальником рішень. Також алгоритм Сугено ідеально підходить для робіт по моделюванню нелінійних систем. Генерування висновків для нечітких правил як таке відсутнє так як розрахунки проводяться з дійсними числами [21].

Як же відбувається дефазифакція? Це проходить таким чином – для роботи використовується модифікований варіант форми методу центру тяжіння для одноточкових множин.

Прийнято вважати, що метод Сугено є точнішим на відміну від алгоритму Мамдані, але він в деяких ситуаціях є важчим у реалізації. Структура системи нечіткого виводу зображена нижче (рисунок 2.8)

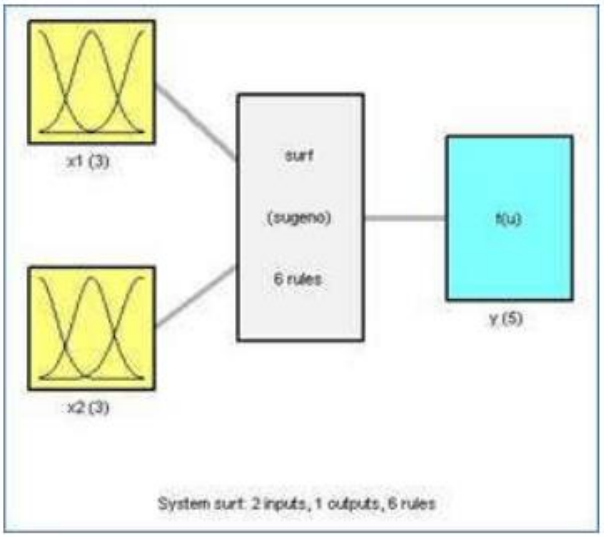


Рисунок 2.8 – Структура системи нечіткого виводу

Фазифікація є одним з найважливіших етапів при реалізації алгоритму Сугено. Саме етап фазифікації однаково виконується в усіх алгоритвах нечіткого виведення. Ще однією назвою фазифікації є приведення до нечіткості. Мета цього етапу це встановити певну відповідність між значеннями вхідних змінних та значеннями функцій приналежності відповідних термів вхідних лінгвістичних змінних.

Після фазифікацції всім вхідним змінним повинні бути протиставлені певні значення функцій приналежності термів.

Алгоритм Сугено відповідає загальній схемі яка притаманна для всіх систем нечіткого керування, але попри це суттєво відрізняється від методу Мамдані. Одна з головних відміностей це конфігурація правил, які представляють собою базу знань. Правила в алгоритмі Сугето мають гібридну форму. Нечіткість вхідних змінних як і в методі Мамдані передбачається передумовами правил, проте вихідні

змінні не є нечіткими множинами, а приставляють з себе певну функціональну залежність від вхідних змінних.

Через меншу трудомісткість проведення розрахунків алгоритм Сугенто є більш кращим і дозволяє моделювати складні системи на відміно від методу Мамдані в якому через формування великої кількості взаємозв'язків між нечіткими множинами майже неможливо моделювати складні системи. Одною з негативних сторін цього алгоритму є відсутність можливості представляти вхідні змінні в лінгвістичній формі. Також мінусом є і налаштування параметрів для функцій приналежності яке є складним завданням нелінійного програмування. Проте існують ефективні методи налаштування які базуються на основі аарату штучних нейронних систем.

Далі в алгоритмі Сугено йде агрегація підумов. Як правило для знаходження істинності умов для всіх правил нечітких методів використовують min-кон'юкцію. Правила істинність яких не дорівнює нулю вважають активними. Активні правила використовують в подальших розрахунках.

Наступним етапом після агрегації підумов йде активізація підвисновків. Суть цього етапу в наступному: знаходження значення істинності всіх висновків для правил, та розрахунок не чітких значень вихідних змінних для кожного з правил.

Все це виконується за допомогою формули для висновків. В цій формулі вхідними значеннями слугують вхідні зміні до етапу фазифікації. Цим способом визначаються і безліч інших значень.

Через те, що розрахунки здійснюються з дійсними числами, акумуляція висновків як така відсутня.

Дефазифакція проходить таким чином – для роботи використовується модифікований варіант форми методу центру тяжіння для одноточкових множин [22].

Ці два методи нечіткої логіки мають багато спільних речей але в одночас мають ключові відмінності, саме ці відмінності дозволяють розділяти їх по різних галузях застосування. Науковцями ці методи використовуються як основа на якій

будуються комплексні системи з величезною кількістю зв’язків. Всі переваги обох методів зображені в таблиці 2.1

Надзвичайно добре використовувати систему нечітких висновків Сугено для завдання яке полягає в плавній інтерполяції лінійних посилань, які використовуються через вхідний простір. Це є ефективним і природним планувальником рішень. Також алгоритм Сугено ідеально підходить для робіт по моделюванню нелінійних систем. Генерування висновків для нечітких правил як таке відсутнє так як розрахунки проводяться з дійсними числами.

Таблиця 2.1 Переваги алгоритмів Мамдані та Сугено

Алгоритм нечіткого виводу	Переваги
Мамдані	Простий і підсвідомо зрозумілий Відмінний для ручного вводу База правил більш інтерпретована Признаний усім світом
Сугено	Ефективний у обчисленнях Зручний при використані лінійних методів Простий при оптимізації Безперервна вихідна поверхня Хороший при використанні математичного аналізу

Оскільки метод Сугено має особливість пов’язану з лінійною залежністю правил від вхідних змінних, він ідеально вписується в роботу як інтерполяційний керівник декількох лінійних контролерів, які необхідно по різному

використовувати в залежності від вхідних умов. До прикладу цей метод може знайти застосування в літаку оскільки з зміною висоти літак може змінити продуктивність. Щоб бути в курсі стану літака контролери повинні регулярно і з плавністю оновлюватися.

На відміно від алгоритму розробленого Сугено, в алгоритмі нечіткого виводу Мамдані вихід для кожного правила являє собою нечіткий набір, який був отриманий від функції виведення членства та методу імплікації FIS. Використовуючи метод агрегування ці нечіткі множини об’єднуються між собою в єдиний нечіткий набір. І для того щоб обчислити вихідне кінцеве значення цей набір дефазифікується одним з методів дефазифікації

2.5 Висновок до першого розділу

В даному розділі було детально розглянуто існуючі програмні засоби які базуються на основі нечіткої логіки. Досліджено принцип і схему роботи таких систем і дана інформація була використана під час розробки власного алгоритму.

Також було розглянуто і проаналізоване дерево рішень яке відіграє не останню роль при створенні власної системи діагностування на основі нечіткої логіки.

Було детальніше розглянуто два основні методи нечіткого виводу, проаналізовано їх плюси і мінуси. На основі цих висновків було зрозуміло що в залежності від поставленої задачі кожний з цих методів має свої переваги, і в той же момент в різних ситуаціях мають і свої мінуси які і впливають на вибір.

3 РЕАЛІЗАЦІЯ АЛГОРИТМУ ДІАГНОСТИКИ СТАНУ КОМП'ЮТЕРНОЇ СИСТЕМИ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ

3.1 Загальна схема нечіткого алгоритму аналізу стану комп'ютерної системи

Для розробки схеми нечіткого алгоритму для аналізу стану комп'ютера було обрано метод нечіткого виводу Мамдані. Отож, щоб розробити нечітку схему нам потрібно мати деякі вхідні дані, до того ж потрібно ще мати нечіткі правила за якими і буде діагностуватися система.

Для того щоб описати певні емпіричні знання чи знання експертів система нечіткого виводу використовує базу правил. Системи нечіткого виведення використовують правила нечітких продукцій. Правила нечітких продукцій – це процес в якому за допомогою нечітких лінгвістичних висловлювань формулюються умови і укладення. Деяка кінцева множина правила нечітких продукцій, і згоджених лінгвістичних змінних які використовуються в правилах називається базою правил нечітких продукцій. Здебільшого базу правил описують у формі деякого, певним чином структурованого, тексту:

Правило номер 1: якщо «умова номер 1, то висновок номер 1», або ж так, якщо температура процесора (ЦП) наближається до, наприклад, 90 градусів за Цельсієм, то можна вважати, що стан ЦП критичний і вимагає безпосереднього втручання з боку користувача [23].

Щоб працював алгоритм потрібно перевести вхідні дані до нечіткості, для цього і використовується фазифікація, яка керується за допопогою функцій приналежності M_x. Фазифікація вказує на ступінь приналежності вхідних параметрів до певного з нечітких значень. Зі збільшенням ступеню приналежності збільшується і ймовірність, що вхідній змінні присвоється якесь нечітке значення. Однією з помилок є те, що дуже часто функцію приналежності плутають з функцією ймовірнісного розподілу. Тому слід пам'ятати, що сума ступенів однієї

вхідної змінної до багатьох нечітких значень може і не дорівнювати 0. Приклад графіку приналежності зображено на рисунку 3.1.

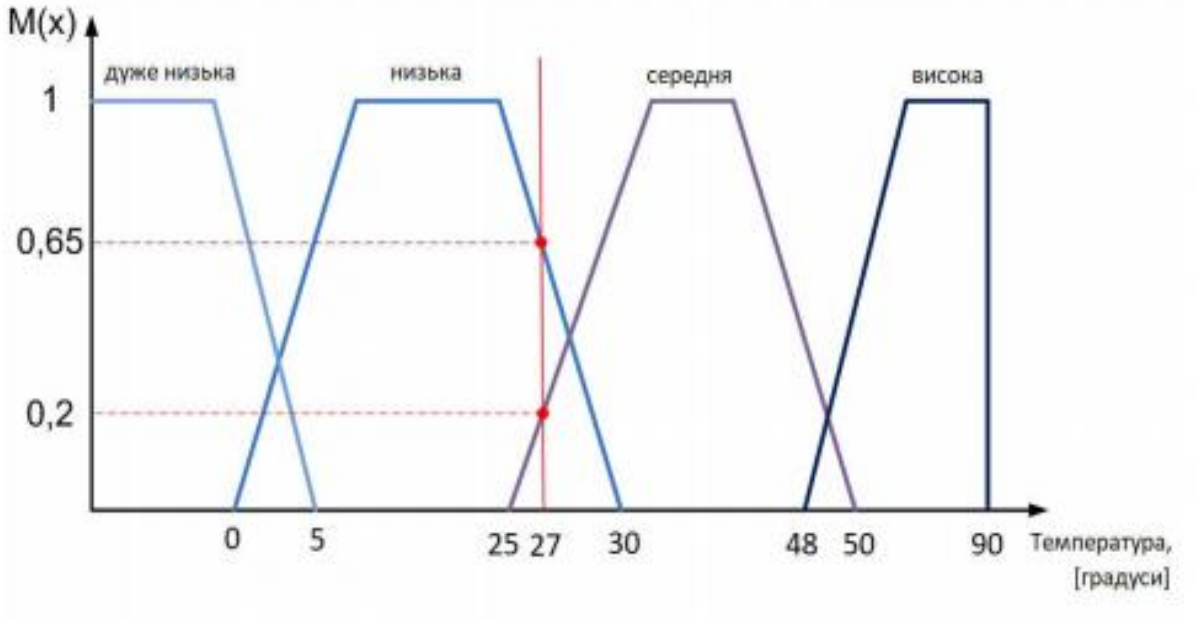


Рисунок 3.1 – Графік функції належності

Такі функції потрібно визначати абсолютно для всіх вхідних і вихідних змінних. Після того як система фазифікує усі вхідні змінні по наявних функціям приналежності, блок який відповідає за прийняття рішень зможе знайти відповідні вихідні змінні користуючись базою правил яка була складена до цього.

Під час етапу дефазифакції система робить зворотне перетворення отриманих нечітких значень до чисел. Способи і алгоритми за якими проходить цей процес різноманітні і можуть відрізнятися в залежності від задачі. Для нас же всі формули і алгоритм які потрібні для роботи вже визначені.

Судячи з вище перечисленої інформації можемо сказати, що нічіткі системи справді є досить зручними для використання у моделюванні складних систем які тісно пов’язані з ЕОМ. Ці системи мають ряд таких переваг:

1) керування системи може виконуватися і даними які не є чіткими, або ж динамічними які постійно змінюються в часі, або даними які неможливо задати однозначно;

2) змога керувати нечіткими критеріями оцінки, наприклад “можливо”,“здебільшого”, та інші;

3) проведення оцінки всіх даних як вхідних так і вихідних;

4) висока швидкість моделювання складних динамічних систем та їх аналіз із вказаною точність. Це завдяки описаним fuzzy-методам які, по-перше не затрачають час на визначення точних значень змінних, по-друге оцінюють різні варіанти які можуть мати вхідні змінні.

Щоб краще зрозуміти структуру і принцип роботи розробленого алгоритму було створено схему яка показує принцип роботи системи.

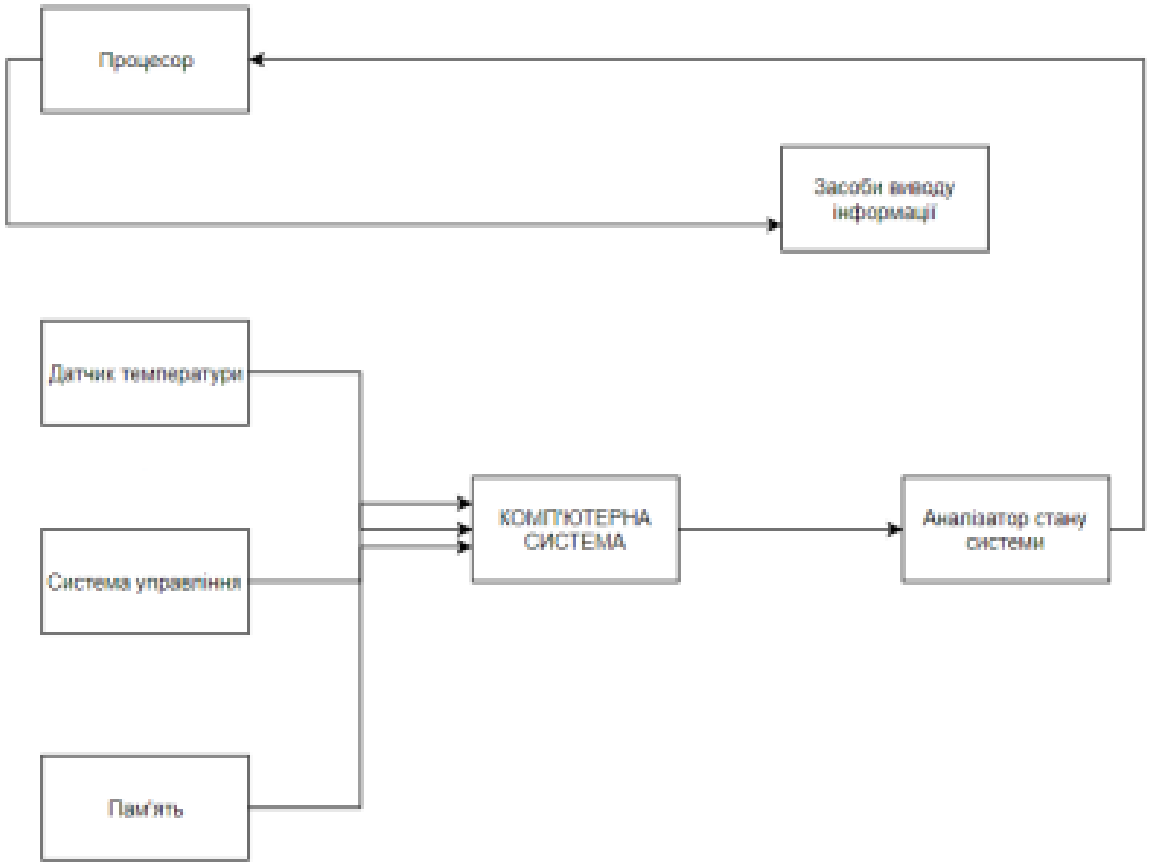


Рисунок 3.2 – Структурна схема розробленої роботи

Система з формальних правил, які розташовані в певному логічному порядку, також які чітко визначають процес для виконання необхідної роботи байдуже якого вона характеру та походження називають алгоритмом.

Перш ніж приступити до розробки алгоритму потрібно чітко розуміти, що дана програма повинна робити, які вхідні дані вона повинна отримувати, які дії повинна виконувати програма для досягнення результату і у якому вигляді вивести результат користувачеві.

Коли ми визначилися з тим що буде робити програма потрібно вирішити яким буде інтерфейс користувача і якими даними він буде маніпулювати. Також потрібно розуміти і вибрати якими методами програма буде обробляти дані для досягнення результату.

Як правило побудова алгоритму є простою у розумінні і будується в кілька етапів. Для початку алгоритм формується в загальних рисах, а вже згодом покращується і уточнюється за допомогою заміни складних якихось частин простішими. Описаний вище метод називають методом покрокової деталізації або як частіше його називають методом «зверху-вниз».

Обов’язково під час розробки необхідно враховувати наявні рішення які вже розроблені і реалізовані за допомогою відомих алгоритмів і методі, також слід урахувати і ресурсні обмеження які є під час розв’язування задачі. І звичайно під час розробки потрібно намагатися створити універсальний алгоритм який буде приймати широкий клас вхідних даних.

Щоб графічно приставити алгоритм використовують блок-схеми. Блок-схеми – це послідовно зв’язані між собою функціональні блоки, які можуть виконувати одну або декілька днів. Ще однією менш популярною назвою блок-схем є схема алгоритму.

За кожну з дій (виведення даних, перевірка умови, повторення дії (цикли), обчисленню значень, задання змінних і тому подібне) в блок-схемі відповідає певний блочний символ, який представлений у вигляді геометричних фігур, в середині яких опис який пояснює дію яку виконую даний блок. Всі блоки з’єднані між собою лінією, ця лінія визначає порядок за яким будуть виконуватися ці дії. Саме тому було обрано блок-схему для зображення розробленого алгоритму

(рисунок 3.3) тому, що це простий і зрозумілий метод представлення алгоритму [25].

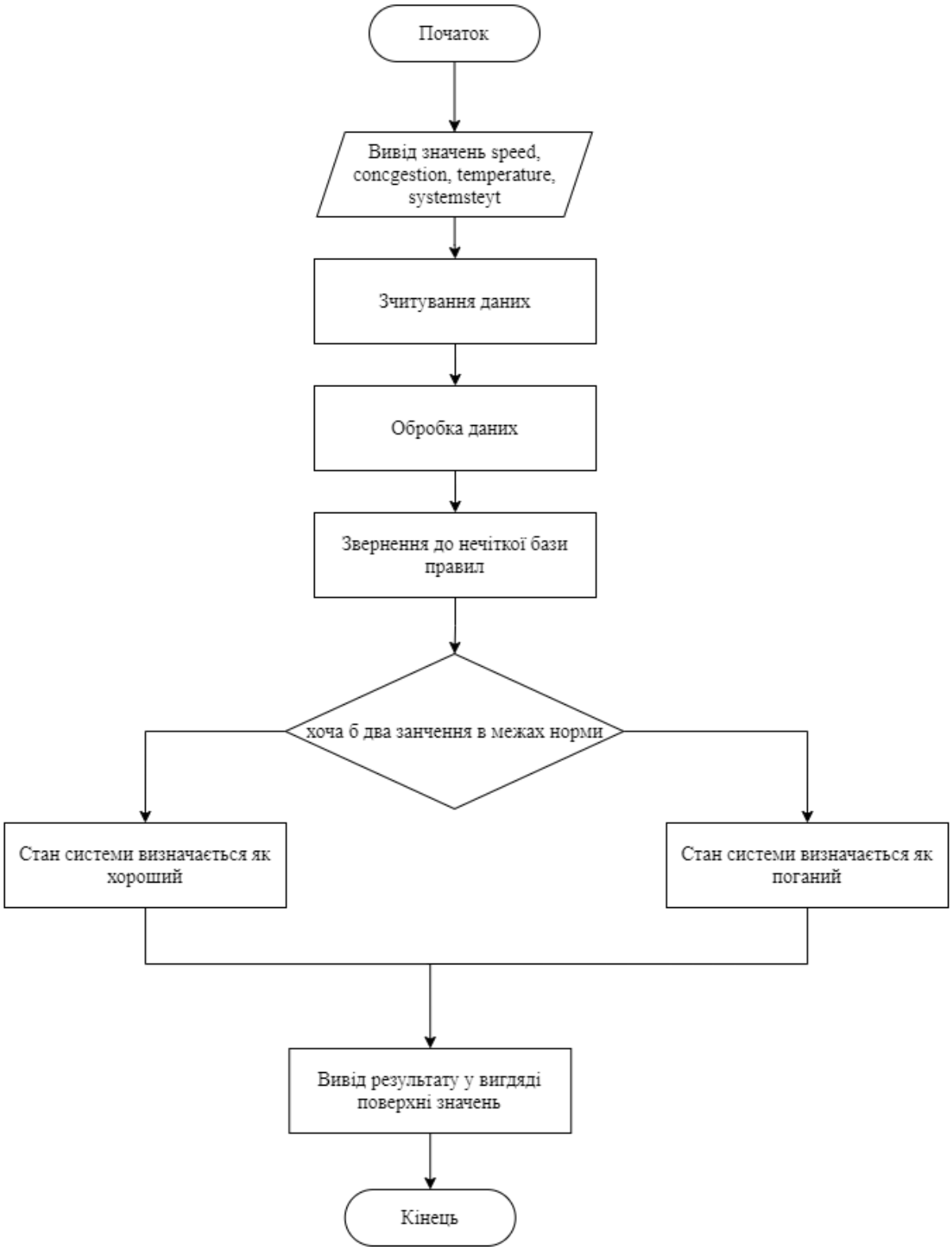


Рисунок 3.3 – Розроблений нечіткий алгоритм аналізу стану комп'ютерної системи

Найбільш поширені методи представлення алгоритму це представлення за допомогою псевдо коду, програмно або графічно.

Якщо проаналізувати алгоритм то можна дійти висновку, що його робота побудована логічно і послідовно з урахуванням усіх операцій які дають змогу реалізувати розроблену схему.

3.2 Симуляція розробленого нечіткого алгоритму.

Щоб побудувати нечітку систему яка базується на основі розробленого алгоритму необхідно задати вхідні дані, і кінцевий результат. На вхід було направлено наступні наді:

1) швидкодія;
2) температура;
3) навантаженість.

На вихід ми отримали стан системи який безпосередньо залежить від вхідних значень. Для того щоб описати певні емпіричні знання чи знання експертів система нечіткого виводу використовує базу правил. Системи нечіткого виведення використовують правила нечітких продукцій. Правила нечітких продукцій – це процес в якому за допомогою нечітких лінгвістичних висловлювань формулюються умови і укладення. Деяка кінцева множина правила нечітких продукцій, і згоджених лінгвістичних змінних які використовуються в правилах називається базою правил нечітких продукцій. Здебільшого базу правил описують у формі певного структурованого тексту. Загальний вигляд розробленої схеми зображено на рисунку 3.4

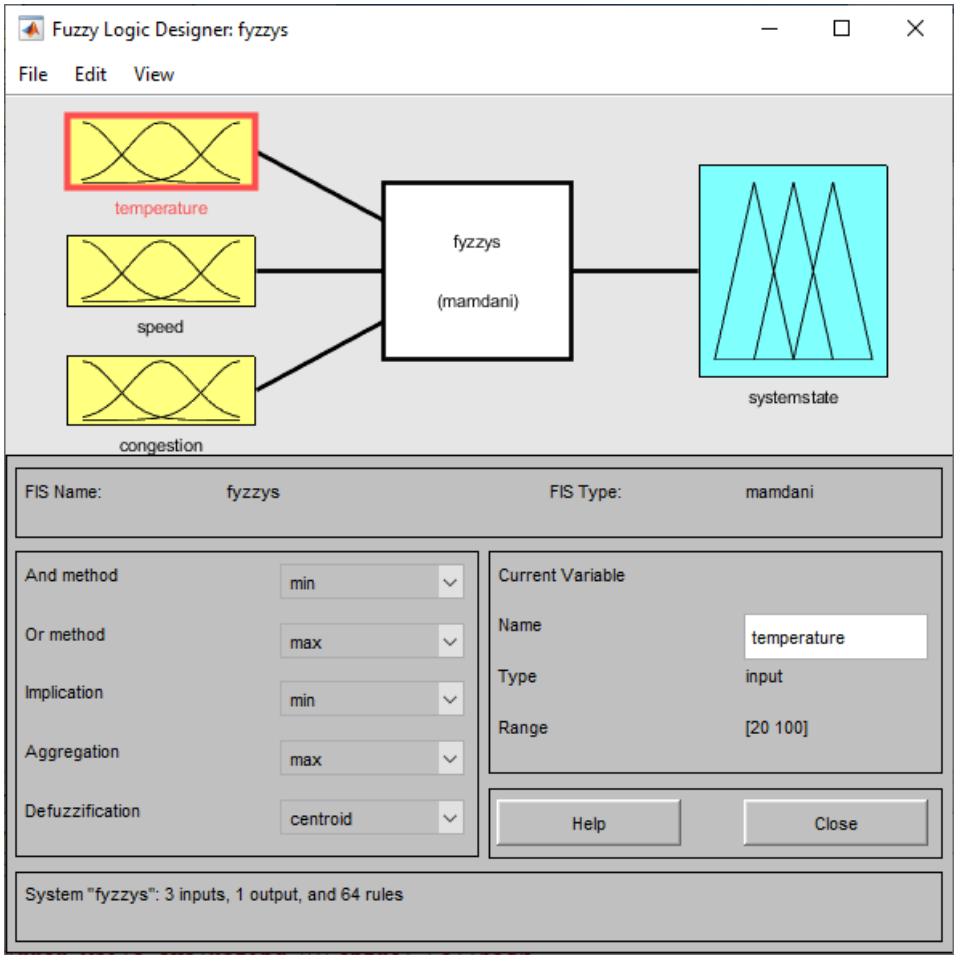


Рисунок 3.4 – Входи розробленої системи

Для роботи системи необхідно для кожного з входів і виходів розробити функції належності. Для опису входів в ході досліджень було обрано функцію “gbellmf” та як вона дає змогу візуально краще висвітлити зміну значень, а для виходів була обрана функція “trimf”. Далі нам необхідно визначити межі вхідних і вихідних функцій. Провівши декілька досліджень з різними комп’ютерами було з’ясовано і обрано наступні межі:

Для температури було обрано наступний діапазон від 35°C до 100°C:

1) до 35°C нормальний температурний режим;
2) до 70°C підвищений температурний режим;
3) більше 70°C високий температурний режим.

Для швидкодії було обрано наступний діапазон від 0 до 1:

1) від 0 до 0,5 висока швидкодія;

2) від 0,5 до 0,7 нормальна швидкодія;

3) від 0,7 до 1 низька швидкодія.

Для навантаженості було обрано наступний діапазон від 0 до 200:

1) від 0 до 80 низька навантаженість;

2) від 80 до 140 нормальна навантаженість;

3) від 130 до 200 висока навантаженість.

Для загального стану комп'ютерної системи було обрано наступний діапазон від 0 до 10:

1) від 0 до 3 ідеальний загальний стан;

2) від 3 до 7 нормальний загальний стан;

3) від 7 до 10 низький загальний стан.

На рисунках 3.5 – 3.8 зображені функції приналежності для вище перерахованих умов;

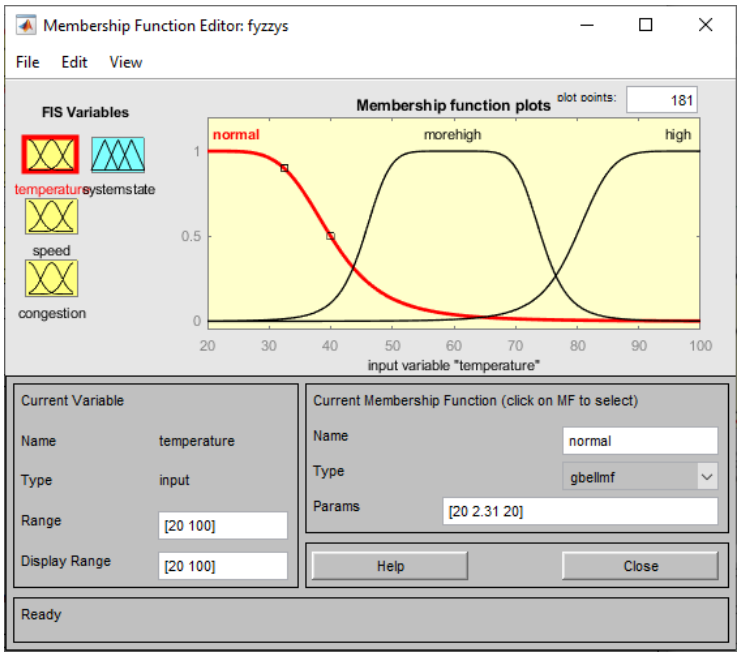


Рисунок 3.5 – Функції приналежності температури

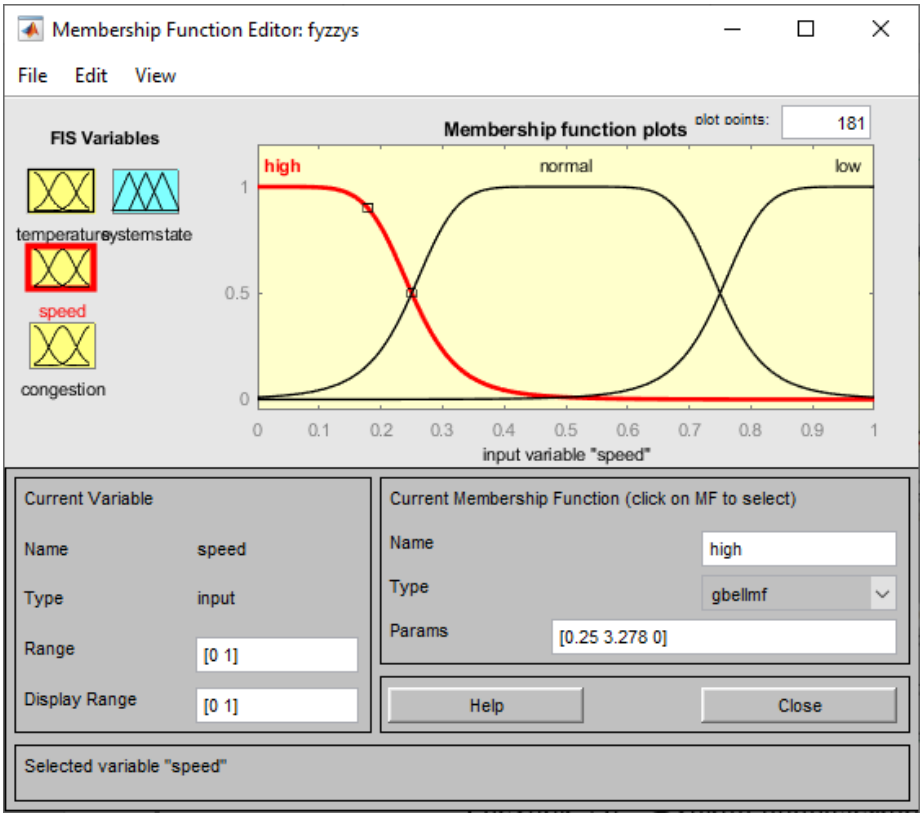


Рисунок 3.6 – Функції приналежності швидкодії

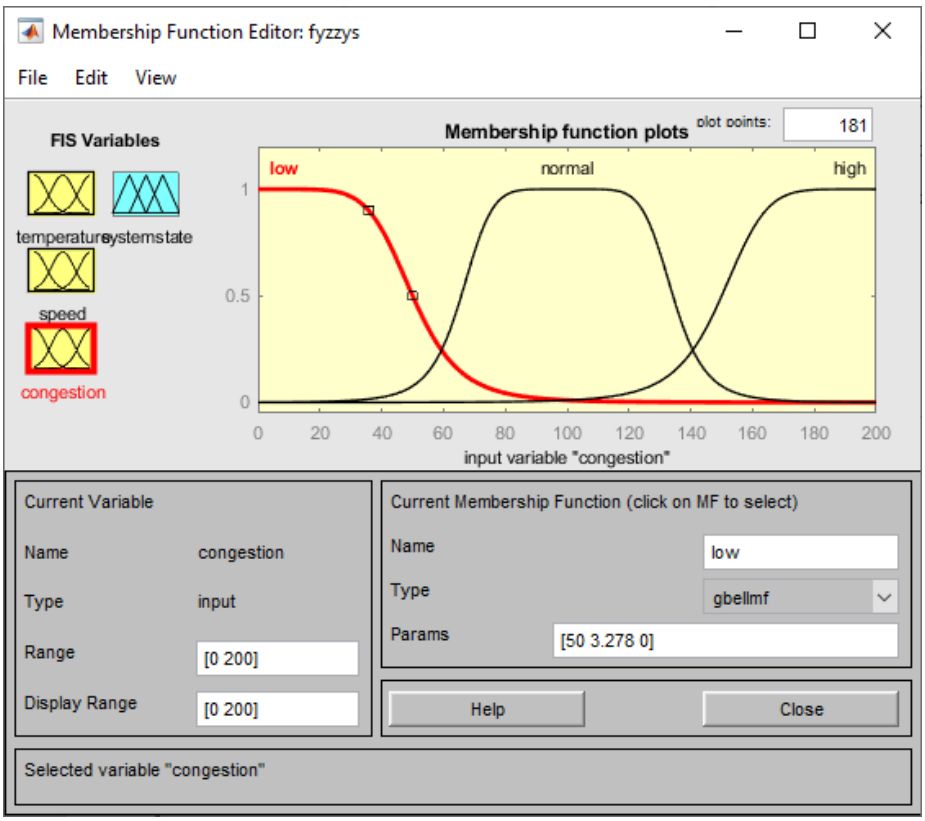


Рисунок 3.7 – Функції приналежності навантаженості

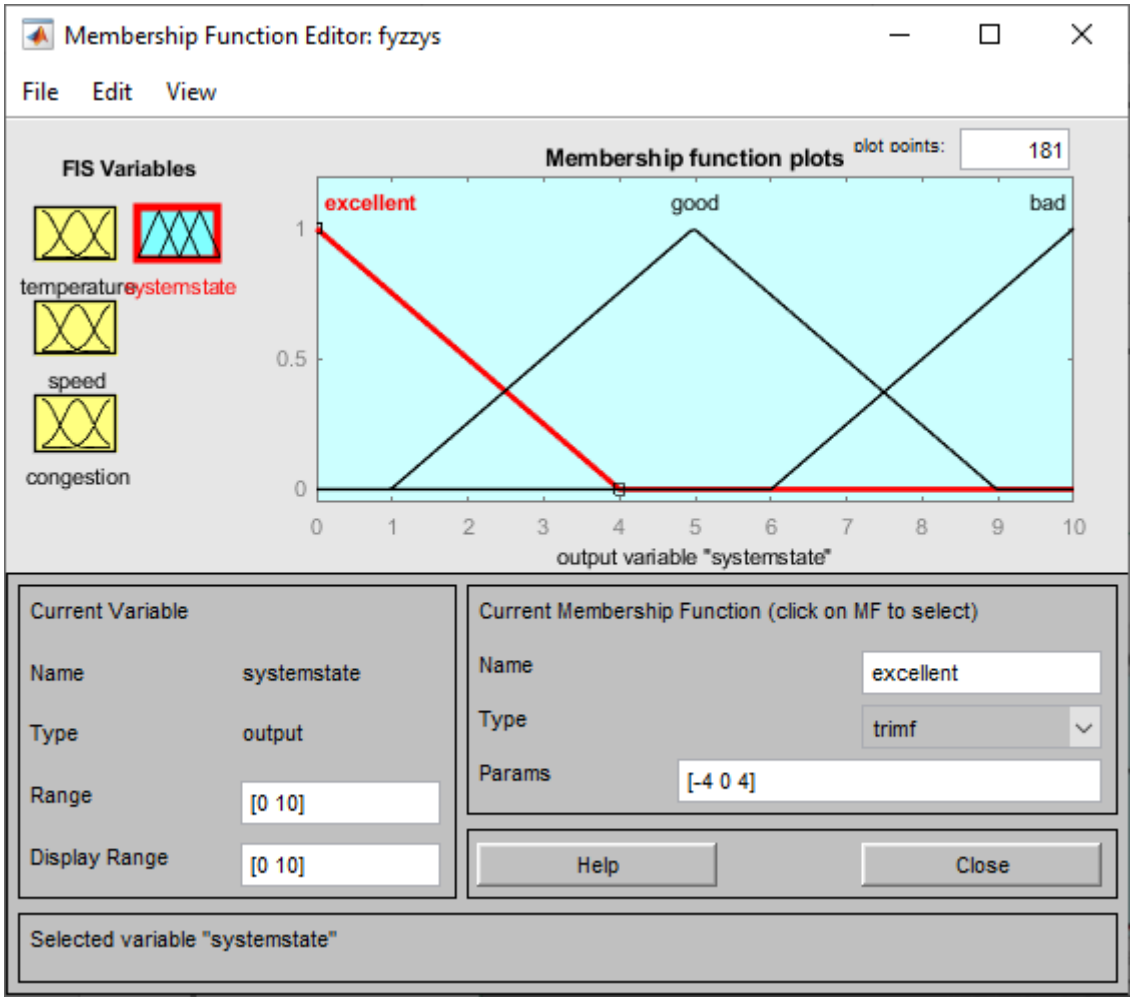


Рисунок 3.8 – Функції приналежності стану системи

Також для роботи розробленої системи була створена база правил. Сукупність нечітких правил представлених у виді «якщо – то» і визначають взаємозв’язки між входами і виходами називають базою правил.

Для того щоб описати певні емпіричні знання чи знання експертів система нечіткого виводу використовує базу правил. Системи нечіткого виведення використовують правила нечітких продукцій. Правила нечітких продукцій – це процес в якому за допомогою нечітких лінгвістичних висловлювань формулюються умови і укладення.

Вхідні змінні системи можуть мати три нечітких стани і один стан None за умови коли змінна не визначена системою. Ситуація коли всі зміні мають стан none неможливий.

Щоб підтвердити працездатність розробленої системи на рисунках 3.9 і 3.10 зображені поверхневі значення.

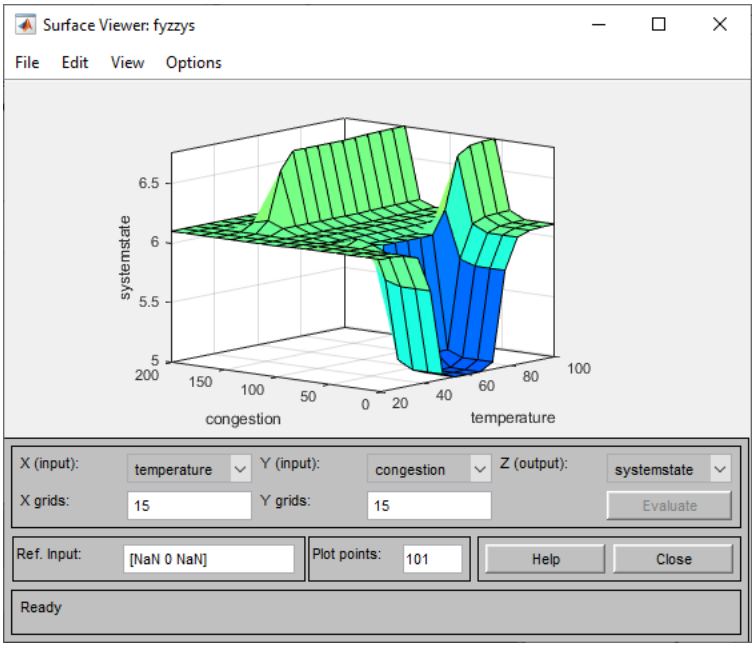


Рисунок 3.9 – Графік поверхні значень стану системи (systemstate) взалежносіті від відношення температури(temperature) до навантаженості (congestion)

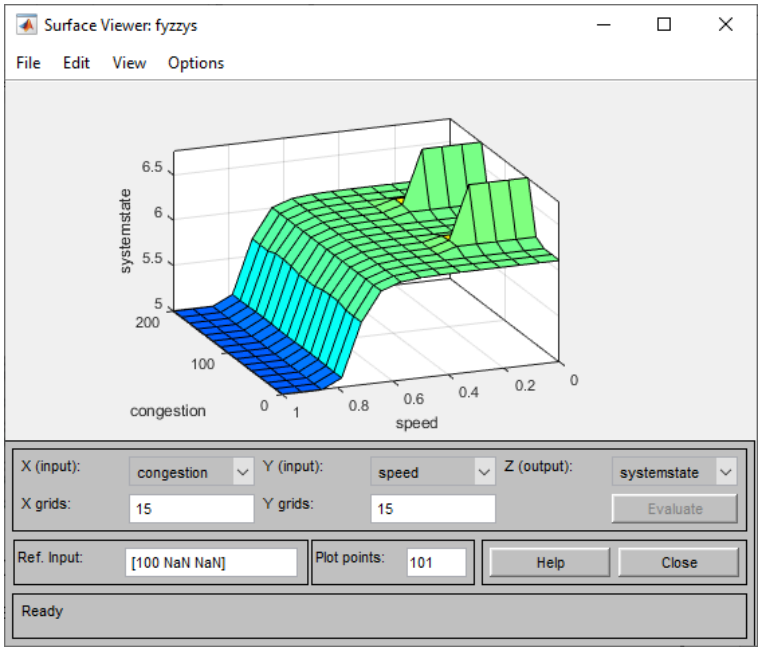


Рисунок 3.10 – Графік поверхні значень стану системи (systemstate) взалежносіті від відношення швидкодії(speed) до навантаженості (congestion)

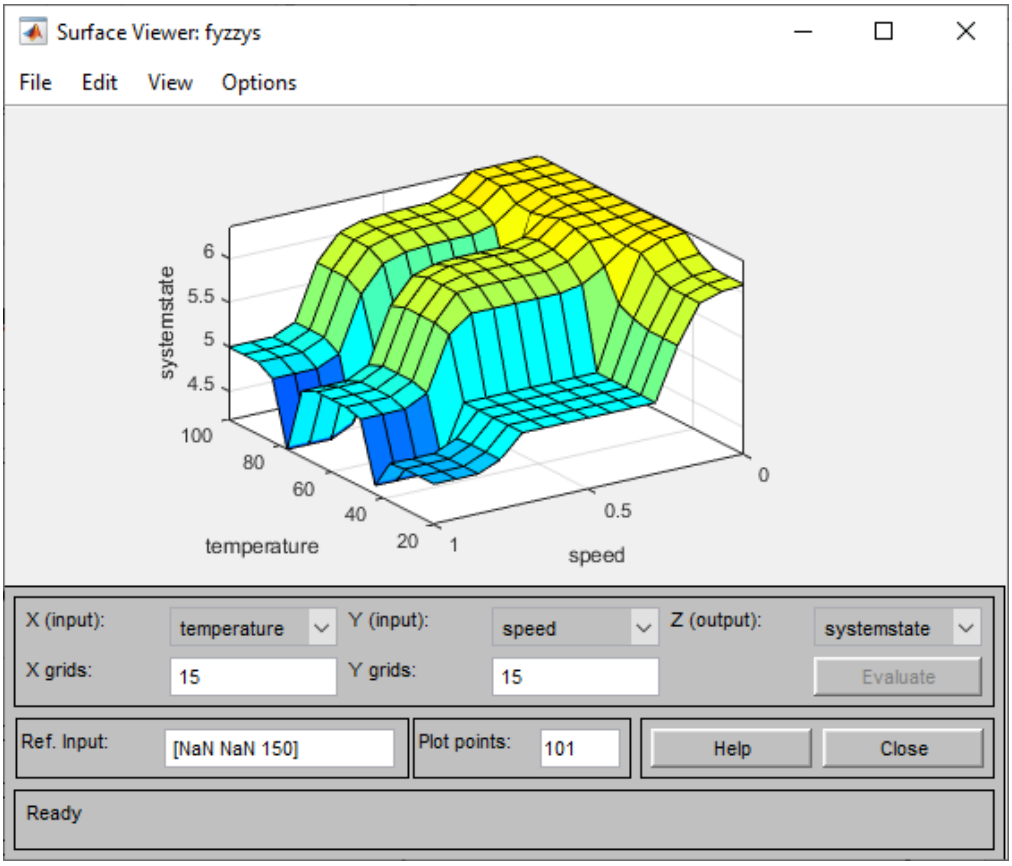


Рисунок 3.11 – Графік поверхні значень стану системи (systemstate) взалежносіті від відношення температури(temperature) до швидкодії(speed)

Нечіткий вивід розробленої моделі діагностування базується на основі 63 правил з наступними значеннями змінних temperature, speed, congestion та systemstate і має наступний вигляд (рисунок 3.11), також в додатку А наведена множина бази правил

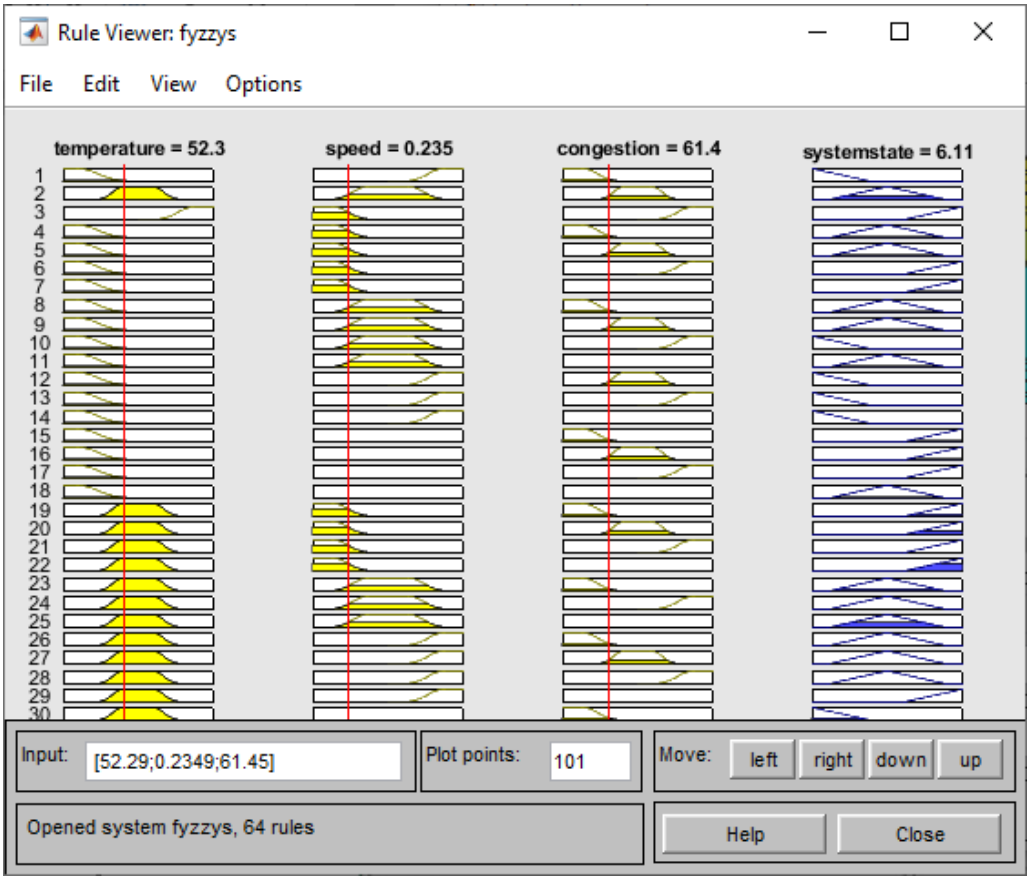


Рисунок 3.11 – Графічне зображення правил нечіткої системи

Також нижче приведена таблиця даних з результатами проведеного моделювання.

Таблиця 3.1 – Результати моделювання нечіткої системи

Вхідні значення			Стан системи (systemstate)
Температура (temperature)	Швидкодія (Speed)	Навантаженість (congestion)	
51,6	0,9	40,8	2,5
68,5	0,53	115	5
55	0,9	70	2,2
55	0,1	120	4,3

Закінчення таблиці 3.1

Вхідні значення			Стан системи (systemstate)
Температура (temperature)	Швидкодія (Speed)	Навантаженість (congestion)	
85	0,5	48	7
20	0,85	57	2,5
35,5	0,68	90	5,6
25,9	0,81	80,1	4,2
14	0,8	110	7,2
85	0,3	50,4	2,7

Під час верифікації розробленої нечіткої системи після перевірки перегляду всіх вхідних і вихідних змінних було зроблено такі висновки: по-перше це те, що система працює коректно і відповідає встановленим вимогам, розробленому алгоритму і базі нечітких правил.

3.3 Висновки до розділу 3

Проаналізувавши два методи нечіткої логіки їх особливості було розроблено власний алгоритм відповідно до завдання дипломної роботи, також структурну схему та блок-схему розробленого алгоритму. Розглянуто його принципи та методи роботи, а також висвітлено його основні поняття. Через те що алгоритм є логічно послідовним можна з легкістю реалізувати нечітку систему на його основі.

Було описано вже розроблену нечітку систему яка базується на розробленому алгоритмі. Розглянуто необхідні вхідні і вихідні дані, базу нечітких правил та середовище розробки. Продемонстровано використані функції належності для вхідних та вихідних змінних. До того ж було продемонстровано результат роботи розробленої системи з яких можна було зрозуміти чи правильно працює система.

ВИСНОВОКИ

Під час виконання дипломної роботи, було створено нечіткий алгоритм який призначений для діагностування комп'ютерних систем, а також базуючись на ньому відповідну нечітку систему. Новизна даної роботи заключається у покращеному алгоритмі який діагностує комп'ютерну систему з використанням апарату нечіткої логіки.

Було вирішено такі питання:

1. Було досліджено проблеми які можуть виникнути під час користування комп'ютером, визначено найпоширеніші проблеми.
2. Було проаналізовано наявні алгоритми нечіткого виводу і за допомогою цього створено алгоритм який вирішує завдання дипломної роботи.
3. Було створено функції приналежності для вхідних і вихідних змінних
4. Створено базу правил яка охоплює всі можливі варіанти.
5. Проведено моделювання та аналіз роботи розробленої нечіткої системи.
6. За допомогою кривих значень було проаналізовано роботу нечіткої системи.
7. Було створено блок схему для кращого розуміння роботи розробленої нечіткої системи.

При використані розробленої системи для діагностики комп'ютерів зменшується імовірність виникнення помилок завдяки вчасному діагностуванні, що зможе зберегти велику кількість ресурсів і часу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Функції обробки нечітких даних. Щоденник: веб–сайт. URL: http://company.shodennik.ua/functions(дата звернення: 23.03.2021).

2. Stanford academic informations. Plato: web–site. URL: https://plato.stanford.edu/entries/logic–fuzzy(дата звернення: 15.04.2021).

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

4. What Is Fuzzy Logic? Mathworks: web–site. URL: https://www.mathworks.com/help/fuzzy/what–is–fuzzy–logic.html(дата звернення: 25.03.2021)

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

6. 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: 23.03.2021).

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

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

9. 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.

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

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

12. 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.

13. 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

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

15. Simulation and Model–Based Design. Mathworks: web–site. URL: https://www.mathworks.com/simulink.html(дата звернення: 11.04.2021)

16. What Is Fuzzy Logic? Mathworks: web–site. URL: https://www.mathworks.com/help/fuzzy/what–is–fuzzy–logic.html(дата звернення: 10.04.2021)

17. . 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: 20.04.2021)

18 . Expert diagnosis of computer systems, neuro–fuzzy knowledge base. IEEE: web–site. URL: https://ieeexplore.ieee.org/document/metrics#metrics(дата звернення: 29.03.2021).

19. 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: 26.04.2021).

20. Usability Determination Using Multistage Fuzzy System. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/article/pii/S1842(дата звернення: 05.04.2021).

21. 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: 12.04.2021).

22. Diagnosing computer hardware failures using expert system (rule–based technique). ResearchGate: web–site. URL: https://www.researchgate.net/publication/79205502_DIAGNOSING_COMPUTE

R_HARDWARE_FAILURES_USING_EXPERT_SYSTEM_RULEBASED_TEC HNIQUE(дата звернення: 11.04.2021)

23. Computer Aided Development of Fuzzy, Neural and Neuro–Fuzzy Systems. ResearchGate: web–site. URL: https://www.researchgate.net/publication/312590719_Computer_Aided_Developm ent_of_Fuzzy_Neural_and_Neuro–Fuzzy_Systems(дата звернення: 12.04.2021).

24. A fuzzy expert system for automatic seismic signal classification. Sciencedirect: web–site. URL: https://www.sciencedirect.com/science/article/pii/S09574114053(дата звернення: 15.04.2021).

25 . 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: 18.04.2021)

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

27. Сравнение нечетких алгоритмов. Cypherpunks: веб–сайт. URL: http://www.cypherpunks.ru/fuzzy/comparison.html(дата звернення: 20.04.2021).

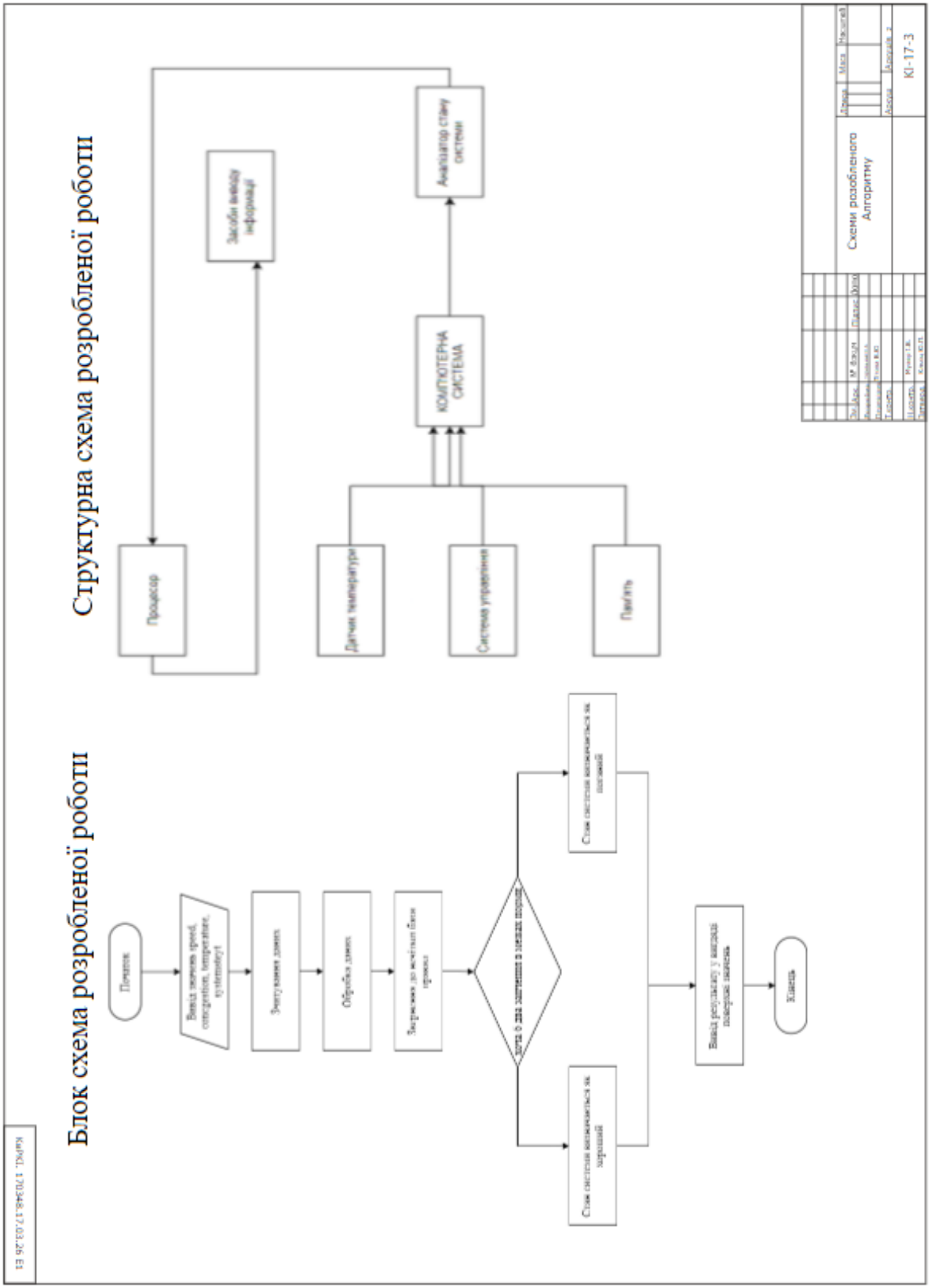
28. . Нечітка логіка на прикладах. iSearch: веб–сайт. URL: http://www.isearch.kiev.ua/index.php/uk/internetsecurity/837–fuzzy–message(дата звернення: 27.04.2021)

29. Mamdani algorithm – Simulink realization. Solutionmes: web–site. URL: http://solutionmes.wikidot.com/mamdani–fuzzy(дата звернення: 25.04.2021).

ДОДАТОК А

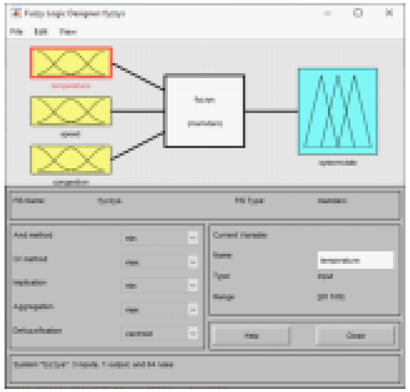
(обовязковий)

Копія графічної частини

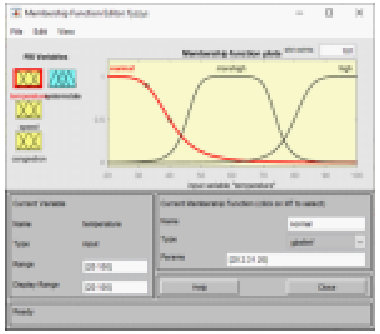


КаРКІ. 170348.17.03.26 Е8

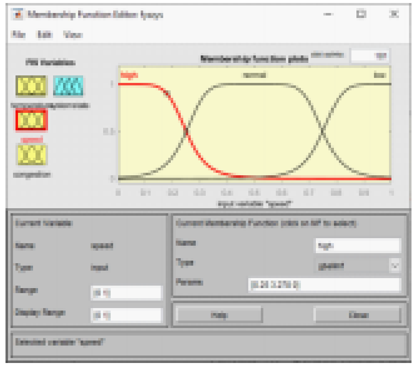
Входи розробленої системи



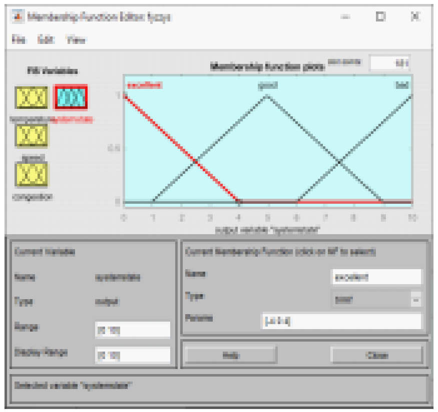
Функції приналежності температури



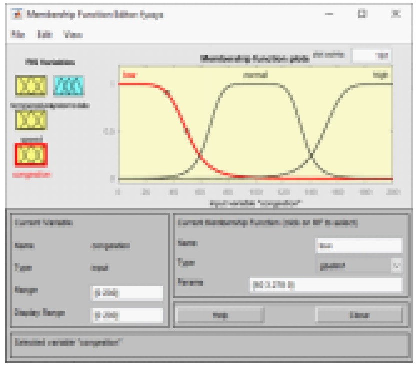
Функції приналежності швидкодії



Функції приналежності стану системи

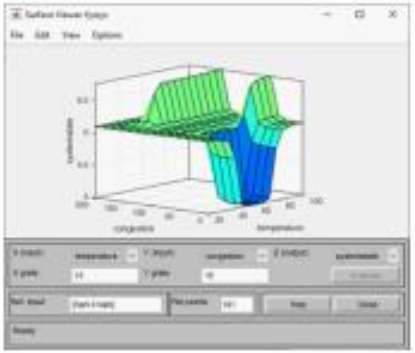


Функції приналежності навантаженості

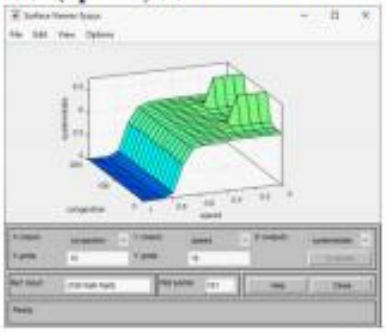


КаРКІ. 170348.17.03.26 Е8	
Входи та функції приналежності розробленої системи	КІ-17-3

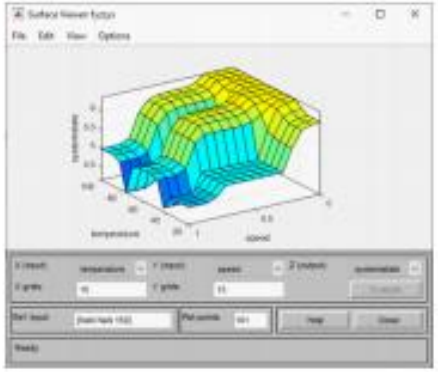
Графік поверхні значень стану системи (systemstate) взалежносіті від відношення температури(temperature) до навантаженості (congestion)



Графік поверхні значень стану системи (systemstate) взалежносіті від відношення швидкодії(speed) до навантаженості (congestion)



Графік поверхні значень стану системи (systemstate) взалежносіті від відношення температури(temperature) до швидкодії(speed)



					КвРКІ. 170348.17.03.26 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Поверхні значень розробленої системи	Літера / Маса / Масштаб
Розробив						
Перевірив						Аркуш / Аркушів 1
Т.контр.						КІ-17-3
Н.контр.						
Затвердив						

ДОДАТОК Б

(обовязковий)

Лістинг коду розробленої системи

```
[System]
Name='fyzzys'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=64
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]
```

Множина бази правил:

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

UNICHECK

User name:
Кафедра кибербезпеки

Check date:
11.06.2021 08:19:54 EEST

Report date:
11.06.2021 08:20:59 EEST

Check ID:
1008266312

Check type:
Doc vs Internet

User ID:
100005590

File name: Звіт з дипломної роботи _пл

Page count: 56 Word count: 8589 Character count: 65251 File size: 1.29 MB File ID: 1008337093

Text modifications detected (similarity score might be affected)

0.59% Matches

Highest match: 0.59% with Internet source (http://dspace.wunu.edu.ua/bitstream/316497/38254/1/%D0%91%D0%B0%D1%81%D1

0.59% Internet sources	33	..... Page 58

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters	17
Suspicious formatting	11 Pages

Fri Jun 11 07:34:35 EEST 2021, Муляра І.В, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

ID: 93222 Название: Діагностування комп’ютерних систем на основі нечіткої логіки Добавлено в БД: 2021-06-11 Авторы: О.А. Парніцький Руководители: В.Ю. Тітова Консультанты: Опоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	57093	504	135 (0%)	3 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Парніцький Олег Анатолійович

Тема: Дігностування комп'ютерних систем на основі нечіткої логіки

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 ; кількість сторінок записки 65

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка нечіткого алгоритму який дозволить проводити аналіз стану комп'ютерної системи.

2. Висновок про відповідність роботи дипломному завданню: Кваліфікаційна робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: Розділ 1 – розглянуто основні поняття комп'ютерних систем досліджено принцип діагностики комп'ютерів та коротко проаналізовано поняття нечіткої логіки. Розділ 2 – проведено аналіз наявних рішень аналізу комп'ютерних систем на основі нечіткої логіки, досліджено дерево рішень та два основні методи нечіткого виводу Сугено та Мамдані. Розділ 3 – розроблено алгоритм діагоностування комп'ютерної системи на основі нечіткої логіки. Методи нечіткої логіки надали алгоритму гнучкості і можливість задавати нечіткі вхідні дані під час діагностування.

4. Позитивні сторони роботи: Застосування розробленого алгоритму дозволить проводити діагносування комп'ютерної системи на помилки, що дозволить уникнути небажаних несправностей.

5. Негативні сторони роботи: В рамках дипломної роботи варто було приділити більше часу на аналіз можливих методів діагностування які реалізуються за допомогою нечіткої логіки.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Оформлення пояснювальної записки відповідає діючим стандартам оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні

8. Інші зауваження: _____

_____

_____

_____

_____

9. Оцінка дипломної роботи: добре. _____

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Гуркан Іван Васильович доцент кафедри інженерії програмного забезпечення

"15" червня 2021 р. _____ (підпис)

Завідувачу кафедри КБКСМ
к.т.н, доцент Кльоц Ю.П

Парніцького О.А
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ-17-3

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів(Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.03.2021
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Діагностування комп'ютерних систем на основі нечіткої логіки

Автор: Парніцький Олег Анатолійович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: ТітоваВєра Юріївна

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки;

1) запозичення розміщені в розділах які аналізують існуючі рішення, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні мають належним чином оформленні посилання;
3) в якості запозичень в окремих місцях системою зафіксовано назви книг, які є загальновідомими і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 0,59% і адресується до 1 першоджерела, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В.Ю. Тітова

Завідувач кафедри КІСП _____ Ю.П Кльоц