МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Студентська
конкурсна робота під шифром «Аналіз системи»
на тему:

«Аналіз поточного стану комп’ютерної системи на основі нечіткої логіки»

ЗМІСТ

ВСТУП

Комп'ютерні системи мають велике значення у сучасному світі. Вони використовуються практично в усіх сферах життя суспільства, стали незамінні для сучасних професій майже усіх галузей. Завдяки комп'ютерній системі можна вирішувати прикладні завдання в предметних галузях діяльності як технологічна підготовка, керування, облік, автоматизація процесів.

Неможливо уявити сучасну людину без комп’ютера, або ж сучасне підприємство без застосування великих комп’ютерних систем та мереж. Тому є актуально захистити комп’ютерну систему шляхом вчасної та регулярної діагностики, яка зможе попередити 90% усіх існуючих проблем, що можуть виникнути.

Мета даної наукової роботи полягає в аналізі поточного стану комп’ютерних систем, їх основних характеристик та розробці відповідної нечіткої системи. Нечітка система дозволить підтримувати задану функціональність та стійкість будь-якої комп’ютерної системи.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

- проаналізувати всі можливі стани комп’ютерних систем;
- дослідити нечітку логіку та обрати відповідний алгоритм нечіткого виводу по базі знань;
- побудувати функції належності;
- сформувати базу правил;
- дослідити роботу реалізованої нечіткої системи.

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

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

Методи дослідження - методи нечіткої логіки, а саме – алгоритм нечіткого висновку Мамдані.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

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 [1].

Разом ці компоненти утворюють комп'ютерну систему: системне апаратне забезпечення, системні програми та документація до них дорівнює комп'ютерна система. Зазвичай, до складу комп'ютерної системи входить три базові апаратні складові: 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.

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 [2] :

- 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 [3].

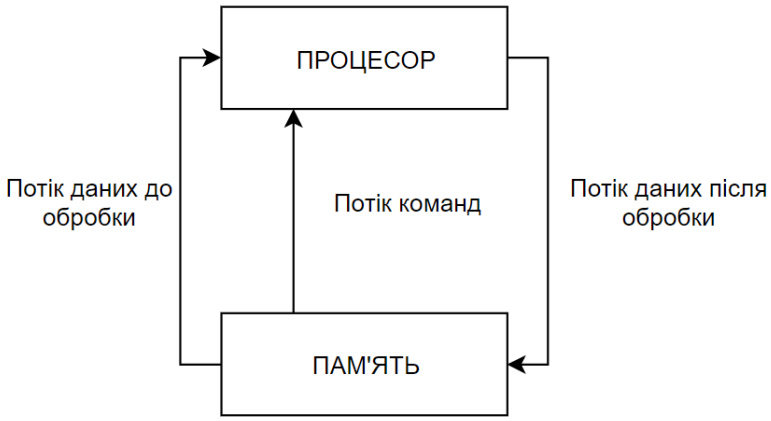

Рисунок 1.1 – Тип архітектури SISD

Типовими представниками SIMD є векторні процесори, звичайні сучасні процесори, які працюють в режимі виконання команд векторних розширень, а також особливий підвид з великою кількістю процесорів - матричні процесори [4]. 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.

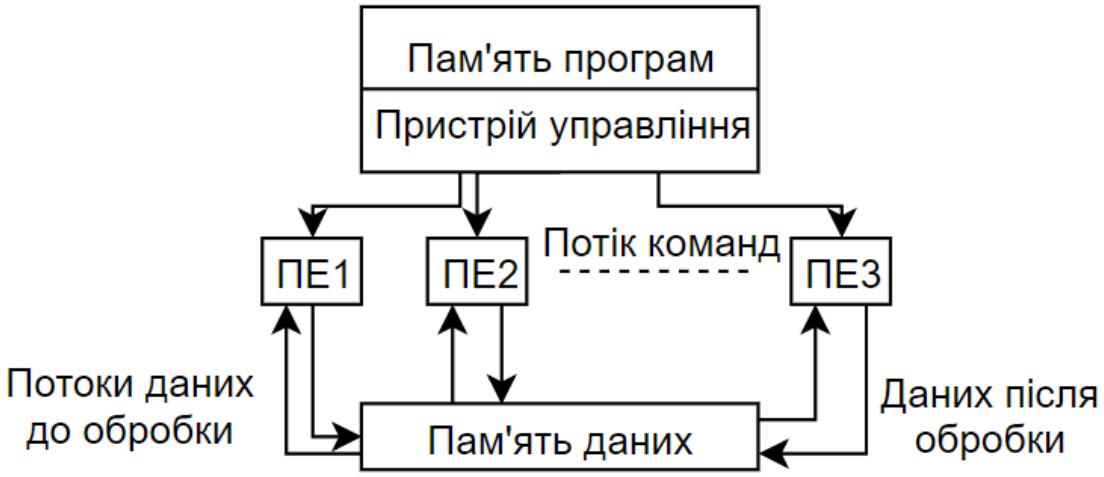

Рисунок 1.2 – Тип архітектури SIMD

До класу MISD ряд дослідників відносить конвеєрні ЕОМ, однак це не знайшло остаточного визнання. Також, можна вважати MISD системами, системи з гарячим резервуванням. Крім цього, до архітектури MISD інколи відносять систолічні масиви процесорів [5]. Архітектура типу MISD показана на рисунку 1.3.

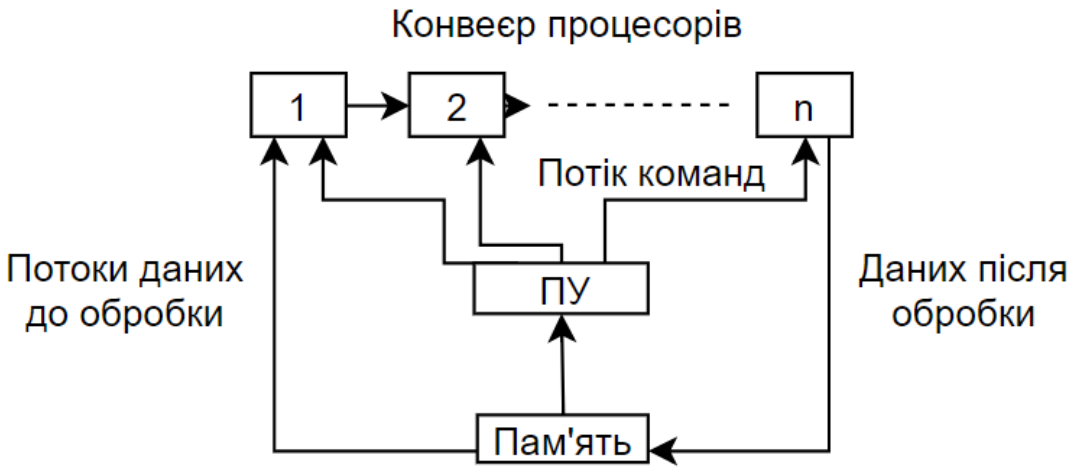

Рисунок 1.3 – Архітектура MISD

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 [6]. Архітектура даного класу зображена на рисунку 1.4.

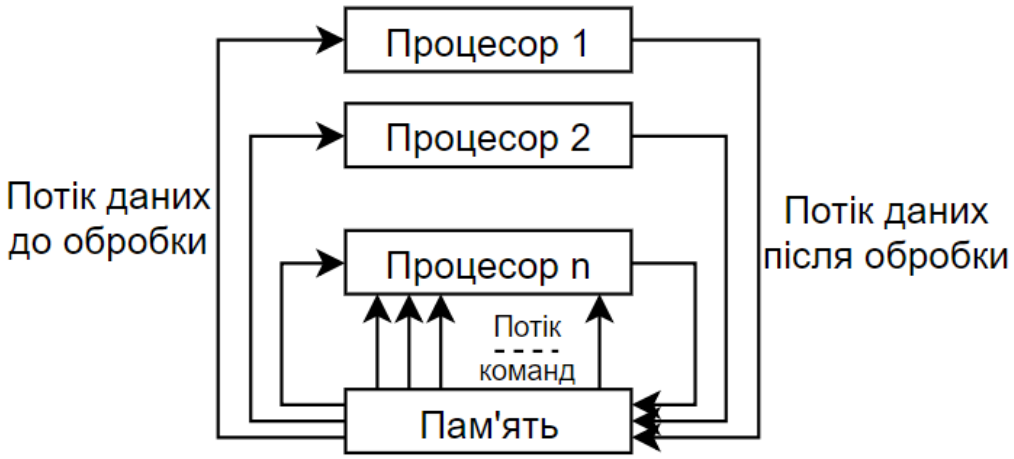

Рисунок 1.4 – Архітектура типу MIMD

У спеціалізованій літературі можна зустріти ще такі підкласи MIMD-класу [7]:

- 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 можуть надсилати та отримувати дані для переходу до наступного кроку [8].

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).

Незалежно від типу, архітектури, основне завдання будь-якої КС - безперебійна робота. Тому необхідно розробити засіб для її діагностики в режимі реального часу.

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.

1.2 Аналіз існуючих нечітких систем комп'ютерної діагностики

На даний час для розв'язання задач комп'ютерної інженерії в режими реального часу використовуються методи штучного інтелекту, а саме нечіткі та нейро-нечіткі системи.

Використання нечіткої логіки дозволяє природно виражати поняття, що використовуються експертами та користувачами [9].

В роботі [10] авторами запропонована експертна система функціональної діагностики, заснована на базі знань у вигляді нейро-нечіткої мережі. Поточні значення діагностичних параметрів вимірюються датчиками. Гібридна експертна діагностична система з базою даних нейрофізіологічної мережі підтримує рішення в ситуації, коли алгоритм діагностики невідомий і формується з вихідних даних у вигляді правил виробництва. Датчики

використовуються для автоматизації процесу накопичення знань у експертній системі.

Застосування нечіткої логіки часто здійснюється для побудови нейро-нечітких моделей, наприклад як у публікації [11]. Нечіткі системи дозволяють здійснювати аналіз як програмного [12], так і апаратного [13] забезпечення, а також для дослідження мережевого трафіку [14].

У роботі [15] пропонується використовувати експертну діагностичну систему 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 пропонується описати діагностичні особливості комп'ютерної системи з точки зору лінгвістичних змінних, що дає можливість використовувати знання та досвід експерта у знайомій формі. У цій статті, експертну діагностичну систему рекомендується використовувати для аналізу технічного стану комп'ютерної системи.

Дослідження [16] вводить метод, який зменшує проблеми при діагностиці апаратних збоїв. Запропонована 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. 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.

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. Проте в даній роботі розроблена нечітка система може витрачати великі ресурси ПК, оскільки повинна в реальному часі постійно аналізувати стан технічного забезпечення.

Перспективою розробки нейро-нечіткої системи є розвиток експертної системи, що є складною задачею, оскільки існує експертна система, яка є високорівневою системою, що займається знаннями, з якими складно впоратися. У роботі [17] описується комп'ютерне проектування інтелектуальних систем, що використовують модемну технологію штучного інтелекту.

Нечітка логіка, звана "fuzzy control", добре відома інженерам програмістам систем управління як зручний засіб програмування і моніторингу додатків управління технологічними процесами. 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 завдання для одного або більшої кількості вимірів [18].

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, постійно 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 застосовувати їх для управління процесами в разі труднощів із застосуванням класичних методів управління [19]. Теорія нечіткої логіки дозволяє описати набори методів управління, які нескладно застосувати для реальної системи і врахують досвід операторів і технологів для динамічного управління процесом. Це дозволяє описувати на нечіткій логіці окремі частини виробничого процесу, такі як ініціалізація та задання параметрів.

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 і являє собою систему, що використовує теорію нечітких множин для відображення входів до виходів [20].

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.

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 [21]. 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):

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

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

$\mu_i\left(x\right)$ – 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):

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

де $\mu_1\left(x\right), \mu_2\left(x\right)$ – 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 :

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

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

min і max – кордони універсуму нечітких змінних;

y_i – результат дефазифікації.

2 НЕЧІТКА СИСТЕМА АНАЛІЗУ СТАНУ КОМП’ЮТЕРНОЇ СИСТЕМИ

2.1 Аналіз поточного стану комп'ютерної системи

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 [23]. 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 [24].

Наявність напруги на виході блоку живлення можна перевірити тестером, помірявши 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 [25]. 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 [26]. 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 2.1).

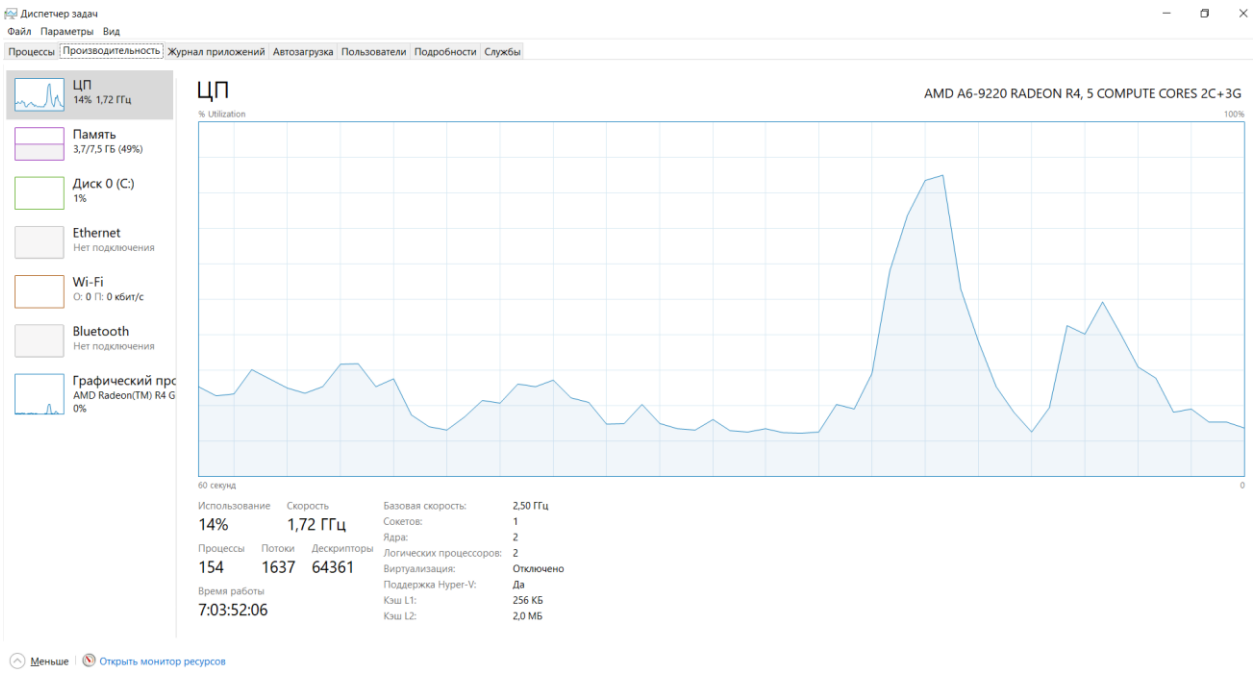

Рисунок 2.1 – Поточні характеристики комп’ютера

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 [27].

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 [28].

2.2 Загальна схема нечіткої системи аналізу стану комп'ютерної системи

Схема нечіткої системи аналізу комп'ютерної системи розроблена за допомогою вже відомого методу нечіткого виводу Мамдані. Для створення такої системи потрібен нечіткий контролер, що описаний за допомогою програмного середовища Matlab. Його загальна схема зображена на рисунку 2.2.

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 [29].

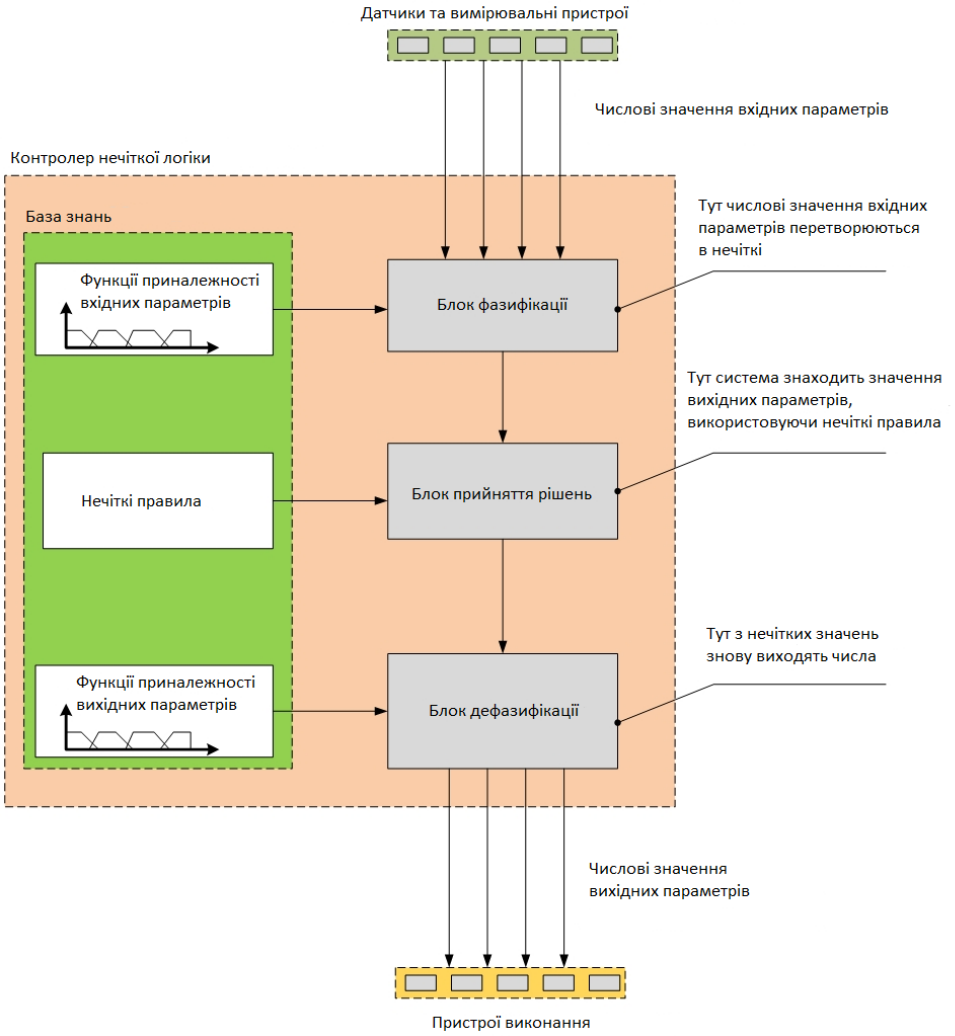

Рисунок 2.2 – Загальна схема нечіткого контролера

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.

База знань для запропонованої нечіткої системи подано в додатку А.

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 [30]. 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 2.3.

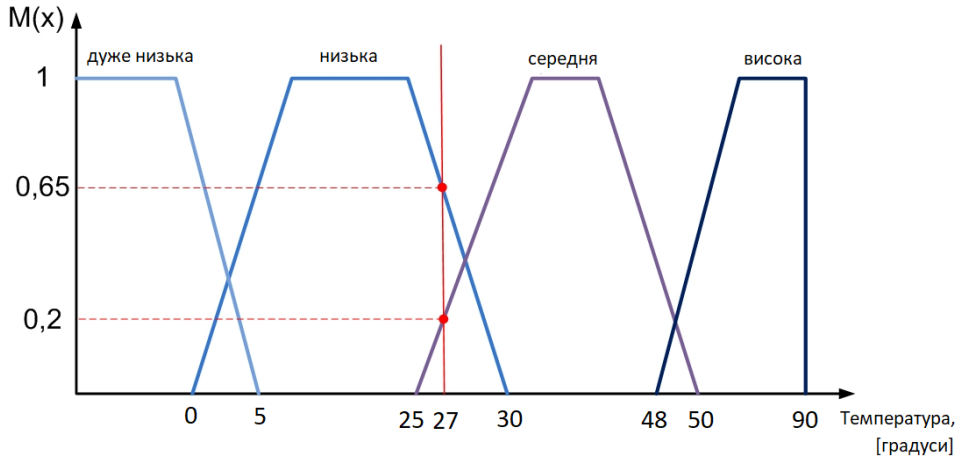

Рисунок 2.3 – Графік функції належності

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 [31].

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 [32].

3 РЕАЛІЗАЦІЯ НЕЧІТКОЇ СИСТЕМИ АНАЛІЗУ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Розробка моделі запропонованого засобу

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 [33].

Розробка нечіткої системи в програмному середовищі Matlab виконується за допомогою набору функцій, 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.

Для побудови нечіткої системи потрібно для початку задати вхідні дані, а також кінцевий результат. На вході запропонованої нечіткої системи подаються наступні значення:

- температура;
- швидкодія;
- навантаженість.

На виході отримано загальний стан системи, який напряму залежить від значень вхідних даних. Вхідні дані можна отримати різними способами, зокрема, температуру отримано за допомогою датчиків, що встановлені на центральному процесорі. Швидкодію та навантаженість можна отримати відкривши «Диспетчер задач» на заданому комп'ютері, або ж використати стороннє програмне забезпечення, що призначене для діагностики компонентів

комп’ютерної системи. На рисунку 3.1 показано загальний вигляд розробленої нечіткої системи.

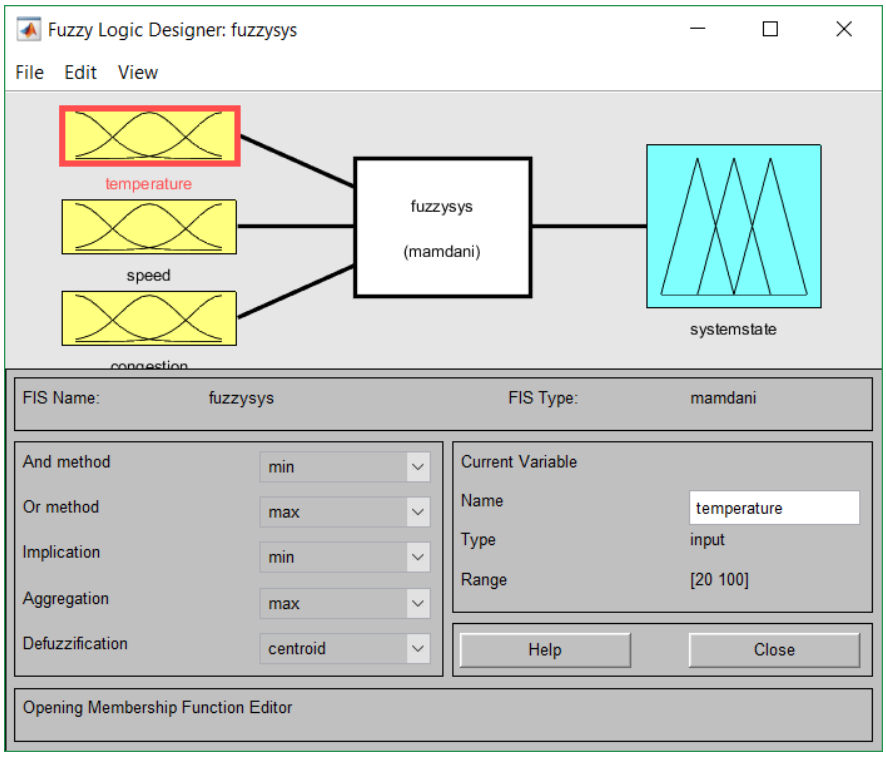

Рисунок 3.1 – Загальна схема розробленої системи

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 3.2 - 3.5.

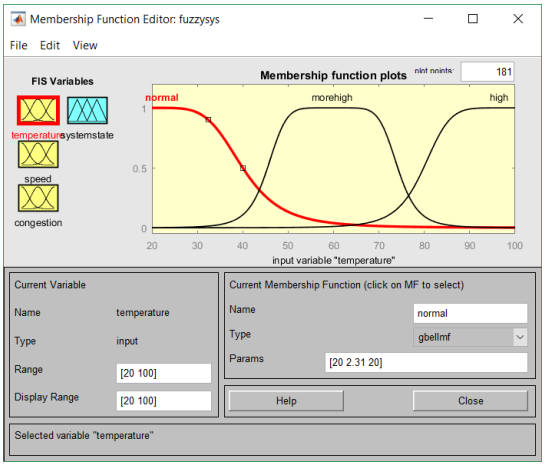

Рисунок 3.2 – Функції приналежності температури

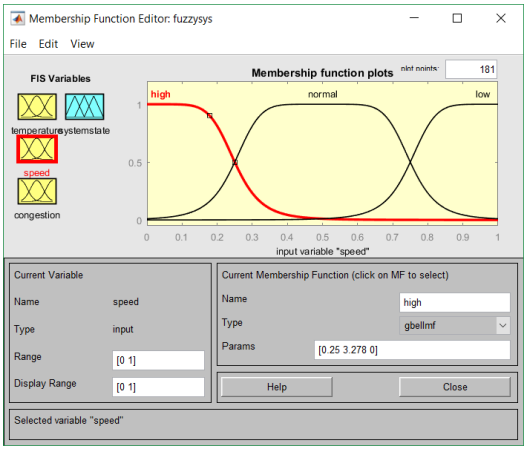

Рисунок 3.3 – Функції приналежності швидкодії

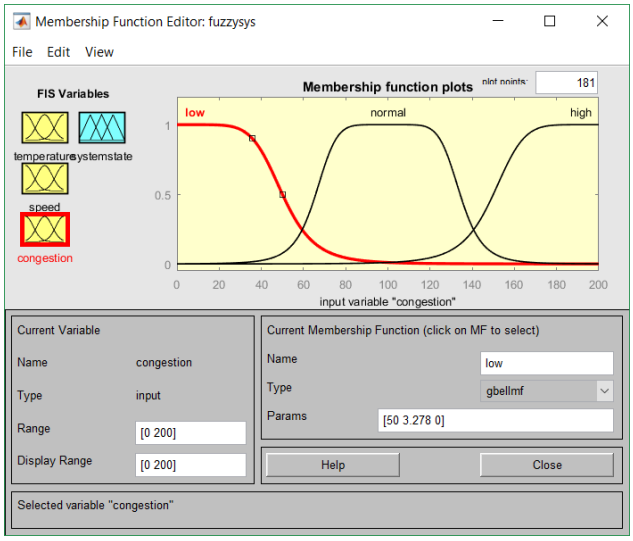

Рисунок 3.4 – Функції приналежності навантаженості

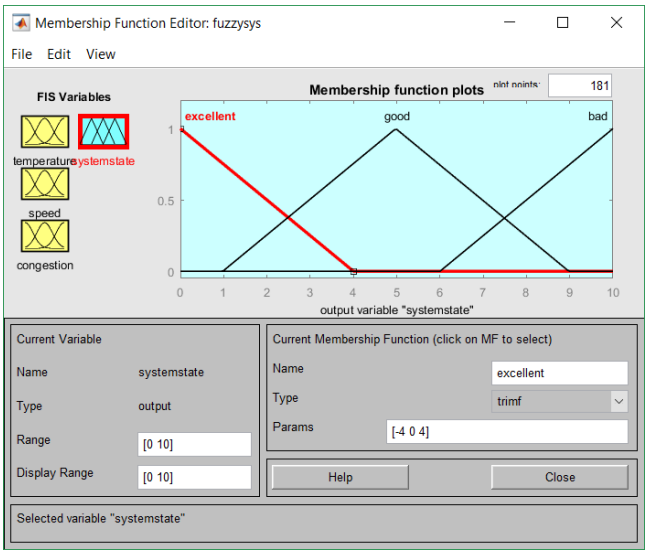

Рисунок 3.5 – Функції приналежності стану системи

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.

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.

3.2 Симуляція роботи нечіткої системи

Для підтвердження працездатності нечіткої системи на рисунках 3.6 та 3.7 подані поверхні значень.

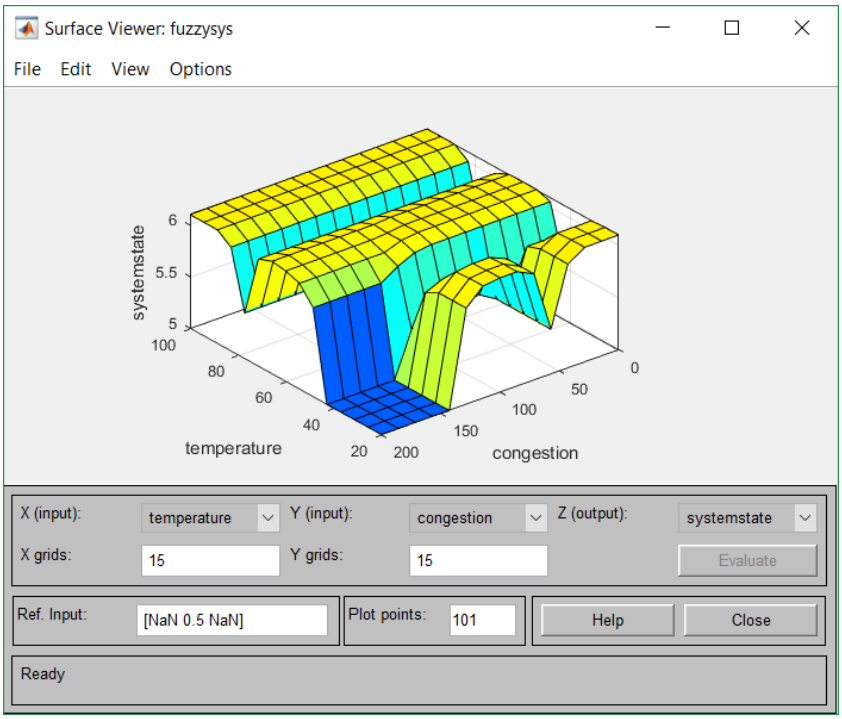

Рисунок 3.6 – Поверхня значень стану системи (systemstate) залежно від відношення температури(temperature) до навантаженості (congestion)

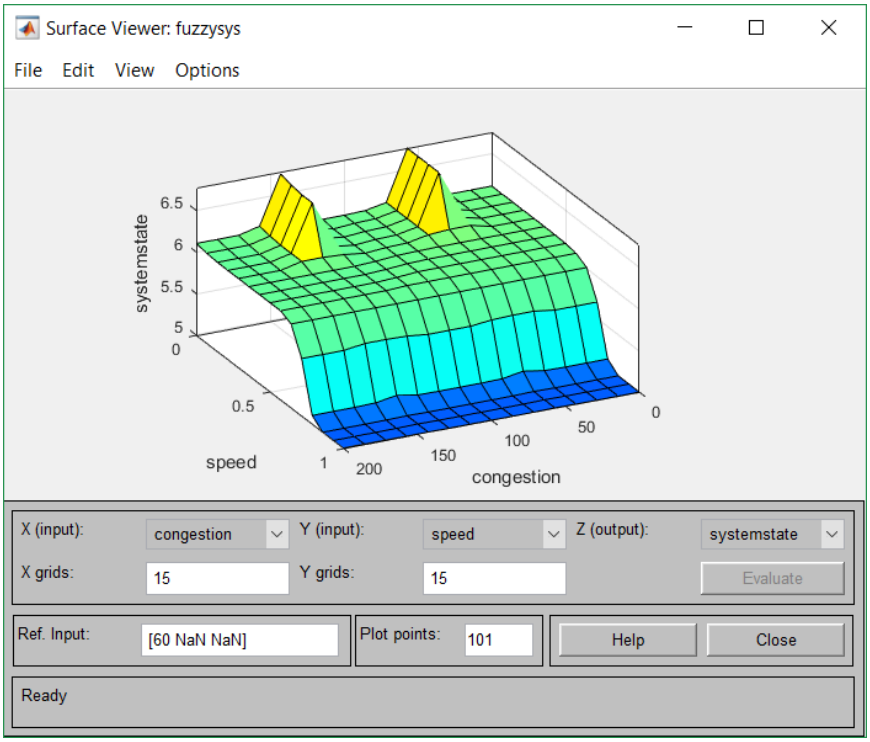

Рисунок 3.7 - Поверхня значень стану системи (systemstate) залежно від відношення швидкодії(speed) до навантаженості (congestion)

Matlab – код розробленої нечіткої моделі подано в додатку Б.

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 3.8.

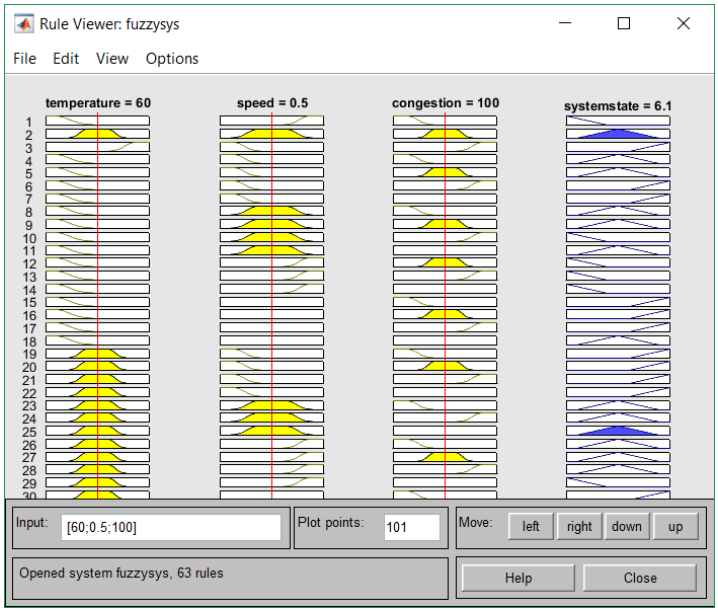

Рисунок 3.8 – Графічне зображення правил нечіткої системи

Також для підтвердження правильності роботи створеної нечіткої системи, представлено таблицю даних з результатами моделювання (таблиця 3.1).

Таблиця 3.1 – 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
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. В подальшому доцільно здійснити моделювання нечіткого контролеру, що працює на основі описаної нечіткої системи.

ВИСНОВКИ

У процесі виконання наукової роботи було розроблено нечітку систему діагностики комп’ютерних систем.

Протягом розробки вирішено такі основні завдання:

1) проаналізовано проблеми, які виникають при користуванні комп’ютером та в загальному в комп’ютерних системах;

2) досліджено переваги нечіткої логіки для побудов систем реального часу;

3) сформовано базу правил нечіткої системи аналізу стану комп’ютерної системи;

4) досліджено можливі алгоритми нечіткого виводу по базі знань і обрано для використання алгоритм Мамдані, що забезпечує високу ймовірність результатів та дає змогу легко реалізувати розроблену нечітку систему в середовищі Matlab;

5) здійснено моделювання та аналіз роботи нечіткої системи за допомогою Fuzzy Logic Toolbox середовищі Matlab.

Розроблену нечітку систему можна використати як основу для розробки нечіткого контролера чи нейрон-нечіткої системи, що може застосовуватися в сучасних комп’ютерних системах для запобігання збоїв в їх роботі та подальшої втрати даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. Щоденник [Електронний ресурс] / Режим доступу до ресурсу: http://company.shodennik.ua/functions/.

3. Zadeh L. Real-Life Applications of Fuzzy Logic // Fuzzy logic now and then – M: Hindawi, 2013. – P. 125.

4. Cintula P. Fuzzy Logics as the Logics of Chains // Fuzzy Sets and Systems – M: Libor, 2006. – P. 606.

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

6. Zimmerman H. Fuzzy set theory and its applications // Fuzzy logic intoduction – M: Kluwer, 1991. – P. 315.

7. MathWorks (What Is Fuzzy Logic?) [Електронний ресурс] / Режим доступу до ресурсу: https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html.

8. Yager R. Fuzzy Sets and Applications // Introduction – M: Wiley, 1987. – P. 8.

9. A semantics-driven, fuzzy logic-based approach to knowledge representation and inference [Електронний ресурс] / Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0957417407006276.

10. Expert diagnosis of computer systems using neuro-fuzzy knowledge base [Електронний ресурс] / Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7807669/metrics#metrics.

11. The neuro-fuzzy diagnostic model synthesis with hashed transformation in the sequence and parallel mode [Електронний ресурс] / Режим доступу до ресурсу: https://ric.zntu.edu.ua/article/view/101022/96247.

12. Usability Determination Using Multistage Fuzzy System [Електронний ресурс] / Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S1877050916000442.

13. A novel fuzzy decision-making system for CPU scheduling algorithm [Електронний ресурс] / Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s00521-015-1987-8.

14. Network-based output tracking control for T-S fuzzy systems using an event-triggered communication scheme(Article) [Електронний ресурс] / Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0165011415000032.

15. Expert evaluation model of the computer system diagnostic features [Електронний ресурс] / Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7027101/metrics#metrics.

16. Diagnosing computer hardware failures using expert system (rule-based technique) [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/279205502_DIAGNOSING_COMPUTER_HARDWARE_FAILURES_USING_EXPERT_SYSTEM_RULEBASED_TECHNIQUE.

17. Computer Aided Development of Fuzzy, Neural and Neuro-Fuzzy Systems [Електронний ресурс] / Режим доступу до ресурсу: https://www.researchgate.net/publication/312590719_Computer_Aided_Development_of_Fuzzy_Neural_and_Neuro-Fuzzy_Systems.

18. A fuzzy expert system for automatic seismic signal classification [Електронний ресурс] / Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0957417414005053.

19. Classification of Network Traffic Using Fuzzy Clustering for Network Security [Електронний ресурс] / Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-62701-4_22.

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

21. Мирончук Ю., Купріненко О. Побудова функцій належності нечітких множин, які відповідають кількісним експертним оцінкам фізичних величин // Системи обробки інформації. — 2017. — № 1. — С. 93-97.

22. Блюмин С., Шуйкова И., Сараев П. Нечеткая логика: алгебраические основы и приложения: Монография – К.:ЛЭГИ, 2002. – 113 с..

23. Cordon O., Herrera F. A General study on genetic fuzzy systems // Genetic Algorithms in computer science – M:Tante, 1995. – P. 33.

24. Леоненков А. Нечеткое моделирование в MATLAB и fuzzyTECH – СПб.: БХВ-Петербург, 2005. – 736 с.

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

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

27. Дубчак Л. Метод обробки нечітких даних на основі механізму Мамдані / Л. Дубчак // Системи обробки інформації. – 2012. – №7(105). – 131с.

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

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

30. 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. - № 5. - С. 137-144.

31. Ротштейн А., Штовба С. Идентификация нелинейной зависимости нечеткой базой знаний с нечеткой обучающей выборкой // Кибернетика и системный анализ. – 2006. – №2. – С. 17–24.

32. MathWorks (Simulation and Model-Based Design) [Електронний ресурс] / Режим доступу до ресурсу: https://www.mathworks.com/products/ simulink.html.

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

Додаток А

База правил розробленої нечіткої системи

1. If (temperature is normal) and (speed is low) and (congestion is low) then (systemstate is excellent) (1)
2. If (temperature is morehigh) and (speed is normal) and (congestion is normal) then (systemstate is good) (1)
3. If (temperature is high) and (speed is high) and (congestion is high) then (systemstate is bad) (1)
4. If (temperature is normal) and (speed is high) and (congestion is low) then (systemstate is good) (1)
5. If (temperature is normal) and (speed is high) and (congestion is normal) then (systemstate is good) (1)
6. If (temperature is normal) and (speed is high) and (congestion is high) then (systemstate is bad) (1)
7. If (temperature is normal) and (speed is high) then (systemstate is bad) (1)
8. If (temperature is normal) and (speed is normal) and (congestion is low) then (systemstate is good) (1)
9. If (temperature is normal) and (speed is normal) and (congestion is normal) then (systemstate is good) (1)
10. If (temperature is normal) and (speed is normal) and (congestion is high) then (systemstate is excellent) (1)
11. If (temperature is normal) and (speed is normal) then (systemstate is good) (1)
12. If (temperature is normal) and (speed is low) and (congestion is normal) then (systemstate is excellent) (1)
13. If (temperature is normal) and (speed is low) and (congestion is high) then (systemstate is excellent) (1)
14. If (temperature is normal) and (speed is low) then (systemstate is excellent) (1)
15. If (temperature is normal) and (congestion is low) then (systemstate is bad) (1)
16. If (temperature is normal) and (congestion is normal) then (systemstate is bad) (1)

17. If (temperature is normal) and (congestion is high) then (systemstate is bad) (1)
18. If (temperature is normal) then (systemstate is good) (1)
19. If (temperature is morehigh) and (speed is high) and (congestion is low) then (systemstate is bad) (1)
20. If (temperature is morehigh) and (speed is high) and (congestion is normal) then (systemstate is bad) (1)
21. If (temperature is morehigh) and (speed is high) and (congestion is high) then (systemstate is bad) (1)
22. If (temperature is morehigh) and (speed is high) then (systemstate is bad) (1)
23. If (temperature is morehigh) and (speed is normal) and (congestion is low) then (systemstate is good) (1)
24. If (temperature is morehigh) and (speed is normal) and (congestion is high) then (systemstate is good) (1)
25. If (temperature is morehigh) and (speed is normal) then (systemstate is good) (1)
26. If (temperature is morehigh) and (speed is low) and (congestion is low) then (systemstate is good) (1)
27. If (temperature is morehigh) and (speed is low) and (congestion is normal) then (systemstate is good) (1)
28. If (temperature is morehigh) and (speed is low) and (congestion is high) then (systemstate is good) (1)
29. If (temperature is morehigh) and (speed is low) then (systemstate is excellent) (1)
30. If (temperature is morehigh) and (congestion is low) then (systemstate is bad) (1)
31. If (temperature is morehigh) and (congestion is normal) then (systemstate is good) (1)
32. If (temperature is morehigh) and (congestion is high) then (systemstate is bad) (1)
33. If (temperature is morehigh) then (systemstate is bad) (1)
34. If (temperature is high) and (speed is high) and (congestion is low) then (systemstate is bad) (1)
35. If (temperature is high) and (speed is high) and (congestion is normal) then (systemstate is good) (1)

36. If (temperature is high) and (speed is high) then (systemstate is bad) (1)
37. If (temperature is high) and (speed is normal) and (congestion is low) then (systemstate is bad) (1)
38. If (temperature is high) and (speed is normal) and (congestion is normal) then (systemstate is good) (1)
39. If (temperature is high) and (speed is normal) and (congestion is high) then (systemstate is bad) (1)
40. If (temperature is high) and (speed is normal) then (systemstate is bad) (1)
41. If (temperature is high) and (speed is low) and (congestion is low) then (systemstate is good) (1)
42. If (temperature is high) and (speed is low) and (congestion is normal) then (systemstate is good) (1)
43. If (temperature is high) and (speed is low) and (congestion is high) then (systemstate is good) (1)
44. If (temperature is high) and (speed is low) then (systemstate is good) (1)
45. If (temperature is high) and (congestion is low) then (systemstate is bad) (1)
46. If (temperature is high) and (congestion is normal) then (systemstate is bad) (1)
47. If (temperature is high) and (congestion is high) then (systemstate is good) (1)
48. If (temperature is high) then (systemstate is bad) (1)
49. If (speed is high) and (congestion is low) then (systemstate is good) (1)
50. If (speed is high) and (congestion is normal) then (systemstate is good) (1)
51. If (speed is high) and (congestion is high) then (systemstate is bad) (1)
52. If (speed is high) then (systemstate is bad) (1)
53. If (speed is normal) and (congestion is low) then (systemstate is good) (1)
54. If (speed is normal) and (congestion is normal) then (systemstate is good) (1)
55. If (speed is normal) and (congestion is high) then (systemstate is good) (1)
56. If (speed is normal) then (systemstate is good) (1)
57. If (speed is low) and (congestion is low) then (systemstate is excellent) (1)
58. If (speed is low) and (congestion is normal) then (systemstate is good) (1)
59. If (speed is low) and (congestion is high) then (systemstate is good) (1)

60. If (speed is low) then (systemstate is excellent) (1)
61. If (congestion is low) then (systemstate is good) (1)
62. If (congestion is normal) then (systemstate is good) (1)
63. If (congestion is high) then (systemstate is good) (1)

Додаток Б

Matlab-код розробленої нечіткої 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
```