

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ СТЕНДОВИХ ВИПРОБУВАНЬ
АВІАЦІЙНИХ ДВИГУНІВ
RESEARCH AND SOFTWARE DEVELOPMENT FOR A
SYSTEM OF AIRCRAFT ENGINE BENCH TESTS

Виконав: студент 2 курсу, групи КНТ-210м
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Архипов А.С.
(прізвище та ініціали)

Керівник Пархоменко А.В.
(прізвище та ініціали)

Рецензент Зеленьова І.Я.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Архипова Арсенія Сергійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та розробка програмного забезпечення для системи стендових випробувань авіаційних двигунів. Research and software development for a system of aircraft engine bench tests

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Дослідження методів та засобів збору даних на ДП «Івченко – Прогрес». 2. Дослідження методів візуалізації даних. 3. Проєктування програмного забезпечення. 4. Конструювання програмного забезпечення. 5. Керівництво програміста. 6. Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	Пархоменко А.В., доцент		
Нормоконтроль	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання “05” листопада 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Вибір програмного забезпечення для проведення випробувань для розробки роботи	1-2 тижні	Завдання, ТЗ
2	Постановка завдання роботи	2-4 тижні	Розділ 1
3	Детальний аналіз предметної області	4-6 тижні	Розділ 2
4	Дослідження методів візуалізації даних	6-8 тижні	Розділ 3
5	Розробка архітектури програмного забезпечення	8-10 тижні	Розділ 4
6	Розробка програми. Тестування та експериментальне дослідження методів візуалізації даних	10-12 тижні	Розділ 5
7	Оформлення програмної документації	12-14 тижні	Розділ 6
	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	14 тиждень	Додатки
	Захист роботи	15-16 тижні	ПЗ

Студент

(підпис) Архипов А.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Пархоменко А.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи магістра:
214 с., 93 рис., 15 табл., 3 дод., 29 джерел.

СТЕНДОВА СИСТЕМА, ЗБІР ДАНИХ, ВІЗУАЛІЗАЦІЯ ДАНИХ, BIG DATA, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АПРОКСИМАЦІЯ.

Об'єкт дослідження – візуалізація даних автоматизованих випробувань авіаційних двигунів.

Предмет дослідження – методи та засоби візуалізації даних.

Мета роботи – дослідження та програмна реалізація методів та засобів візуалізації даних для підвищення якості представлення вимірюваних параметрів на системах стендових випробувань авіаційних двигунів.

Матеріали, методи та технічні засоби: методи апроксимації, Arduino UNO, датчик температури та вологості DHT 11, мова програмування LabVIEW, персональний комп'ютер з процесором AMD A8-6600K під управлінням операційної системи Microsoft Windows 10.

Результати. Розроблено програмне забезпечення для збору, візуалізації, збереження та пост-обробки даних випробувань авіаційних двигунів.

Висновки. Впровадження розробленого програмного забезпечення дозволить вдосконалити підсистему проведення стендових випробувань Octopus та підвищити якість візуалізації вимірюваних параметрів.

Галузь використання – промислові автоматизовані системи для випробувань авіаційних двигунів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master's: 214 pages, 93 figures, 15 tables, 3 appendixes, 29 sources.

BENCH SYSTEM, DATA COLLECTION, DATA VISUALIZATION, BIG DATA, SOFTWARE, APPROXIMATION

Object of study is visualization of data from automated tests of aircraft engines.

Subject of research are methods and means of data visualization.

The goal of the work is research and software implementation of methods and means of data visualization to improve the quality of representation of measured parameters on bench test systems of aircraft engines.

Materials, methods and technical means: approximation methods, Arduino UNO, temperature and humidity sensor DHT 11, LabVIEW programming language, personal computer with an AMD A8-6600K processor running an operating Microsoft Windows 10 systems.

Results. Software for collecting, visualizing, storing and post-processing test data of aircraft engines has been developed.

Findings. The implementation of the developed software will improve the Octopus bench testing subsystem and quality of visualization of measured parameters.

The fields of application are industrial automated systems for testing aircraft engines.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	10
Вступ.....	12
1 Дослідження методів та засобів збору даних.....	14
1.1 Промислові джерела та способи збору даних.....	14
1.2 Дослідження засобів збору даних на підприємстві ДП «Івченко - Прогрес»	16
1.2.1 Аналіз стендової системи ГАММА – 1101	19
1.2.2 Дослідження стендової системи АІВС ПБ – 2.....	24
1.2.3 Порівняння систем стендових випробувань	28
1.3 Дослідження програмних засобів для проведення випробувань на стендових системах реального часу.....	29
1.3.1 Аналіз програмного забезпечення для проведення випробувань на стендових системах АІВС ПБ – Octavo	30
1.3.2 Аналіз програмного забезпечення для проведення випробувань на стендових системах АІВС ПБ – Octopus.....	36
1.3.3 Аналіз підпрограми підготовки випробувань Octopus.....	38
1.3.4 Аналіз підпрограми управління випробуваннями Octopus	47
1.3.5 Аналіз підпрограми пост-обробки та аналізу Octopus	50
1.3.6 Аналіз підпрограми візуалізації Octopus	53
1.3.7 Висновок за дослідженням програмного забезпечення для проведення випробувань на стендових системах	56
1.4 Аналіз апаратних засобів збору даних.....	58
1.4.1 Огляд перетворювачів тиску АІР – 10Н.....	58

1.4.2	Огляд мікроелектронних датчиків тиску МІДА	61
1.4.3	Відмінність датчиків АІР – 10Н від МІДА	63
1.5	Аналіз апаратних засобів аналого – цифрового перетворення	63
1.5.1	Дослідження аналого – цифрових перетворювачів	64
1.5.2	Дослідження цифро – аналогових перетворювачів	65
1.5.3	Аналіз блок збору та перетворення інформації	66
1.5.4	Аналіз плат Advantech 1747U	70
1.6	Постановка задач дипломної кваліфікаційної роботи	74
2	Дослідження методів візуалізації даних	76
2.1	Аналіз даних випробувань авіаційних двигунів на підприємстві	76
2.2	Огляд особливостей апроксимації	78
2.2.1	Лінійна інтерполяція	81
2.2.2	Інтерполяція поліномом Лагранжа	83
2.2.3	Інтерполяція сплайнами	85
2.2.4	Ермітова інтерполяція	88
2.2.5	Апроксимація методом найменших квадратів	92
2.3	Обчислювальні експерименти	96
2.3.1	Дослідження на наборі даних з прототипу підсистеми вимірювання температури та вологості	96
2.3.2	Дослідження методів інтерполяції на стендових даних з авіаційного двигуна	108
2.4	Висновки до розділу 2	118
3	Проектування програмного забезпечення	120
3.1	Аналіз вимог до тестового програмного забезпечення для проведення випробувань на стендових системах реального часу	120

3.2 Вибір середовища розробки.....	120
3.2.1 Аналіз можливостей середовища розробки MatLab	120
3.2.2 Аналіз можливостей середовища розробки LabVIEW	127
3.3 Розробка програмного забезпечення.....	136
3.4 Архітектура тестової системи збору даних.....	139
3.5 Висновки до розділу 3	141
4 Конструювання програмного забезпечення	143
4.1 Алгоритм роботи тестового програмного забезпечення для проведення випробувань на стендових системах.....	143
4.2 Алгоритм роботи оператора з тестовою системою збору даних	146
4.2 Алгоритм роботи розробника з тестовою системою збору даних.....	149
4.3 Функції які використовувалися в розробці програмного забезпечення	152
4.4 Огляд розробленого інтерфейсу та функцій, що виконує програмне забезпечення	157
4.5 Висновки до розділу 4	166
5 Керівництво програміста.....	170
5.1 Призначення і умови використання тестової системи збору даних	170
5.2 Характеристики програми.....	170
5.3 Вхідні і вихідні дані	171
5.4 Встановлення програмно-апаратного комплексу	171
5.5 Алгоритм роботи розробника з тестовою системою збору даних.....	172
5.6 Умови використання тестової системи збору даних.....	177
6 Керівництво користувача	178
6.1 Призначення тестової системи збору даних	178

6.2 Умови використання тестової системи збору даних	178
6.3 Експлуатація тестової системи збору даних	178
6.4 Повідомлення тестової системи проведення випробувань.....	183
Висновки	185
Перелік джерел посилання	186
Додаток А Технічне завдання	190
Додаток Б Текст програми	194
Додаток В Слайди презентації.....	207

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ARINC	– Aeronautical Radio Incorporated;
CE	– Conformance Européenne;
CLK	– Clock Signal;
COM	– Communications Port;
CS	– Consent Signal;
DBL	– Double;
DLL	– Dynamic Link Library;
DMA	– Direct Memory Access;
DOS	– Disk Operating System;
EXT	– Extended;
FIFO	– First In First Out;
HART	– Highway Addressable Remote Transducer;
IBM	– International Business Machines;
ID	– Identifier;
ISA	– Industry Standard Architecture;
MS-DOS	– MicroSoft Disk Operating System;
NI	– National Instruments;
PCI	– Peripheral Component Interconnect;
QNX	– Quick UNIX;
RDY	– Ready;
SGL	– Single;
TCP	– Transmission Control Protocol;
USB	– Universal Serial Bus
VI	– Virtual Instrument;
AIBC	– автоматизована інформаційно-вимірювальна система;
APM	– автоматизоване робоче місце;

АЦП	– аналого-цифровий перетворювач;
БД	– база даних;
БЗП	– блок збору та перетворення інформації;
БП	– блок перетворення інформації;
БУС	– блок управління та синхронізації;
ГОСТ	– міждержавний стандарт;
ГРТ	– групова реєстрація тиску;
ДП	– державне підприємство;
ІВК	– інформаційно – вимірювальний комплекс;
КТСС	– кодований тактовий сигнал синхронізації;
МПП	– многоканальний перетворювач інформації;
ОС	– операційна система;
ПЕОМ	– промислові електронно обчислювальні машини;
ПЗО	– пристрій зв'язку з об'єктом;
ПК	– персональний комп'ютер;
ПКМ	– права кнопка миші;
ПРВ	– підпрограма реєстрації та відображення;
ПСО	– плата сполучення об'єктового;
СЗД	– станція збору даних;
ТО	– технічний опис;
ТТЛ	– транзисторно-транзисторна логіка;
ЦАП	– цифро-аналоговий перетворювач.

ВСТУП

Сьогодні ми можемо бачити стрімкий розвиток аерокосмічної галузі. Саме зараз втілюються такі амбіційні проекти як створення місячної бази та політ на Марс. В Україні також існують підприємства які досягли значних успіхів у розробці аерокосмічного обладнання такі як ДП «Івченко – Прогрес». Важливою складовою циклу розробки аерокосмічного обладнання є його тестування. Для забезпечення тестування розроблюваних авіаційних двигунів на підприємстві використовуються випробувальні стендові системи збору даних ГАММА – 1101 та АІВС ПБ – 2. Для проведення випробувань на стендових системах реального часу використовується програмне забезпечення VERT, Octavo та Octorus. Для забезпечення надійності двигунів під час експлуатації на літальних апаратах, та відповідності сучасним стандартам якості важливо розроблювати та вдосконалювати апаратні та програмні засоби для тестування авіаційних двигунів. Для вдосконалення апаратних засобів тестування постійно впроваджуються нові обчислювальні та вимірювальні засоби.

Програмне забезпечення для тестування необхідно постійно вдосконалювати, оскільки:

- розвиток аерокосмічної сфери, ставить нові завдання для програмних засобів тестування;
- підвищення вимог безпеки, тестування та атестації виробів веде до більш жорстких вимог до програмного забезпечення;
- нарощення темпів виробництва виробів аерокосмічної галузі призводить до необхідності збільшення швидкості підготовки до випробувань, більш швидкого освоєння нових видів випробувань;
- процедура метрологічної атестації каналів вимірювання потребує оптимізації задля збільшення кількості випробувань.

Передумови вдосконалення програмних засобів тестування на ДП «Івченко – Прогрес»:

- програмне забезпечення для проведення випробувань на стендових системах реального часу ГАММА – VERT вже застаріле, воно не підтримує сучасні рішення візуалізації та операційні системи Windows після 98 версії;

- програмне забезпечення для проведення випробувань на стендових системах реального часу АІВС ПБ – Octavo має закритий програмний код, що не дозволяє модифікувати програму силами підприємства;

- програмне забезпечення для проведення випробувань на стендових системах реального часу АІВС ПБ – Octopus ще розроблюється та впроваджується на виробництві, воно потребує тестування та вдосконалення.

Тема роботи є актуальною через наявність існуючих проблем з якістю візуалізації даних випробувань на стендових системах реального часу та потребою в удосконаленні відповідного програмного забезпечення.

Мета роботи полягає у дослідженні та програмній реалізації методів та засобів візуалізації даних для підвищення якості представлення вимірюваних параметрів на системах стендових випробувань авіаційних двигунів.

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ ЗБОРУ ДАНИХ

Збір даних – це процес збору інформації та вимірювання інформації про задані змінні, що дозволяє відповісти на актуальні питання та оцінити отримані результати [1].

Мета збору даних – відібрати якісні дані, що дозволяє при аналізі всіх даних дати переконливі та достовірні відповіді на поставлені питання.

Методи збору даних [1]:

- методи емпіричного дослідження (спостереження, порівняння, вимірювання, експеримент);
- методи, що використовуються як на емпіричному, так і на теоретичному рівні дослідження (абстрагування, аналіз і синтез, індукція та дедукція, моделювання та ін.);
- методи теоретичного дослідження (сходження від абстрактного до конкретного, ідеалізація, формалізація, аксіоматичний метод) [1].

1.1 Промислові джерела та способи збору даних

У всіх сучасних галузях промисловості використовуються системи автоматизованого збору та обробки даних. Ці системи виконують задачу збору даних швидше, точніше та з меншою кількістю помилок ніж застарілий ручний спосіб збору даних. Вони відрізняються сферою застосування, складністю архітектури, призначенням, об'ємом аналізованих даних, методами збору інформації [2].

Способи збору даних стали більш диверсифікованими і тепер включають:

- пристрої та датчики інтернету речей: з розвитком периферійних технологій з'явилася можливість збирати дані за допомогою автоматичних процесів із великої кількості джерел, включаючи датчики на промисловому

устаткуванні, каналізаційних трубах, мостах тощо, пристрої моніторингу пацієнтів та багато інших пристроїв [2];

- збір аудіовізуальних даних: по мірі того, як рішення тепер включають аналіз неструктурованих даних, у тому числі аудіо, зображення та відео, збір цих даних став важливим. Файли з цими даними займають набагато більше місця, для їх обробки потрібно більше обчислювальної потужності.

- аналітика в реальному часі: дозволяє збирати та аналізувати потоки даних на безперервній основі [2];

- анонімізований збір даних: у зв'язку з вимогами конфіденційності виникла потреба аналізувати деякі дані без їх прив'язки до фізичної особи, до якої вони належать. Збір та обробка даних тепер можуть включати угруповання за демографічними параметрами без доступу до певних персональних даних [2];

- курирування даних: професійні аналітики спеціалізуються на організації структурованих джерел даних для підтримки складного аналізу таких аспектів як кліматологія та фінансові прогнози. У зв'язку з обсягом цих наборів даних для їхнього аналізу зазвичай потрібна інфраструктура високопродуктивних обчислень[2];

- випробувальні стенди: сфери застосування стендів досить різноманітні, проте можна виділити головні – теоретична та практична наука, сфери промисловості, що працюють з складними пристроями (аерокосмічна, енергетична, суднобудівна, гірна та інші) [2].

Джерела збору даних у промисловості:

- датчики;
- вимірювальні потокові пристрої (паливо міри та інші);
- ваги;
- лічильники;
- сканери;
- дані з паперових носіїв;

- стенди реального часу.

Випробувальний стенд – установка, що використовує промислові комп'ютери для аналізу, обробки, відпрацьовування даних у жорсткому реальному часі [3], що:

- дозволяє відпрацьовувати, перевіряти та налаштовувати системи управління в реальному часі на об'єкті управління;
- дозволяє вирішувати усі помилки у системі управління реального часу до використання її на об'єкті;
- значно скорочує час налагодження та перевірки об'єкта за рахунок миттєвого аналізу великого об'єму даних [3];
- дозволяє організувати дистанційне управління за рахунок відправки керуючих сигналів на об'єкт управління;
- дозволяє автоматизувати тестування та збір інформації на об'єкті управління;
- надає можливість створення точної моделі поведінки об'єкта управління [3].

Стенд реального часу складається з апаратної та програмної системи. Крім того, данні з апаратної системи збору даних можуть бути проаналізовані за допомогою різних програмних систем [3].

1.2 Дослідження засобів збору даних на підприємстві ДП «Івченко - Прогрес»

На підприємстві для проведення випробувань авіаційних двигунів використовуються наступні системи збору даних: ГАММА – 1101 та АІВС ПБ – 2.

Використання різних систем обумовлено різним часом створення та специфікою випробувань. Далі наведені складові двох систем у загальному вигляді (рис. 1.1), на схемі сірими прямокутниками позначені комп'ютери,

жовтими – елементи системи що знаходяться поза комп'ютерами, червоним прямокутником позначено підпрограму програмного забезпечення для проведення випробувань на стендовій системі збору даних, яка була проаналізована більш докладно.

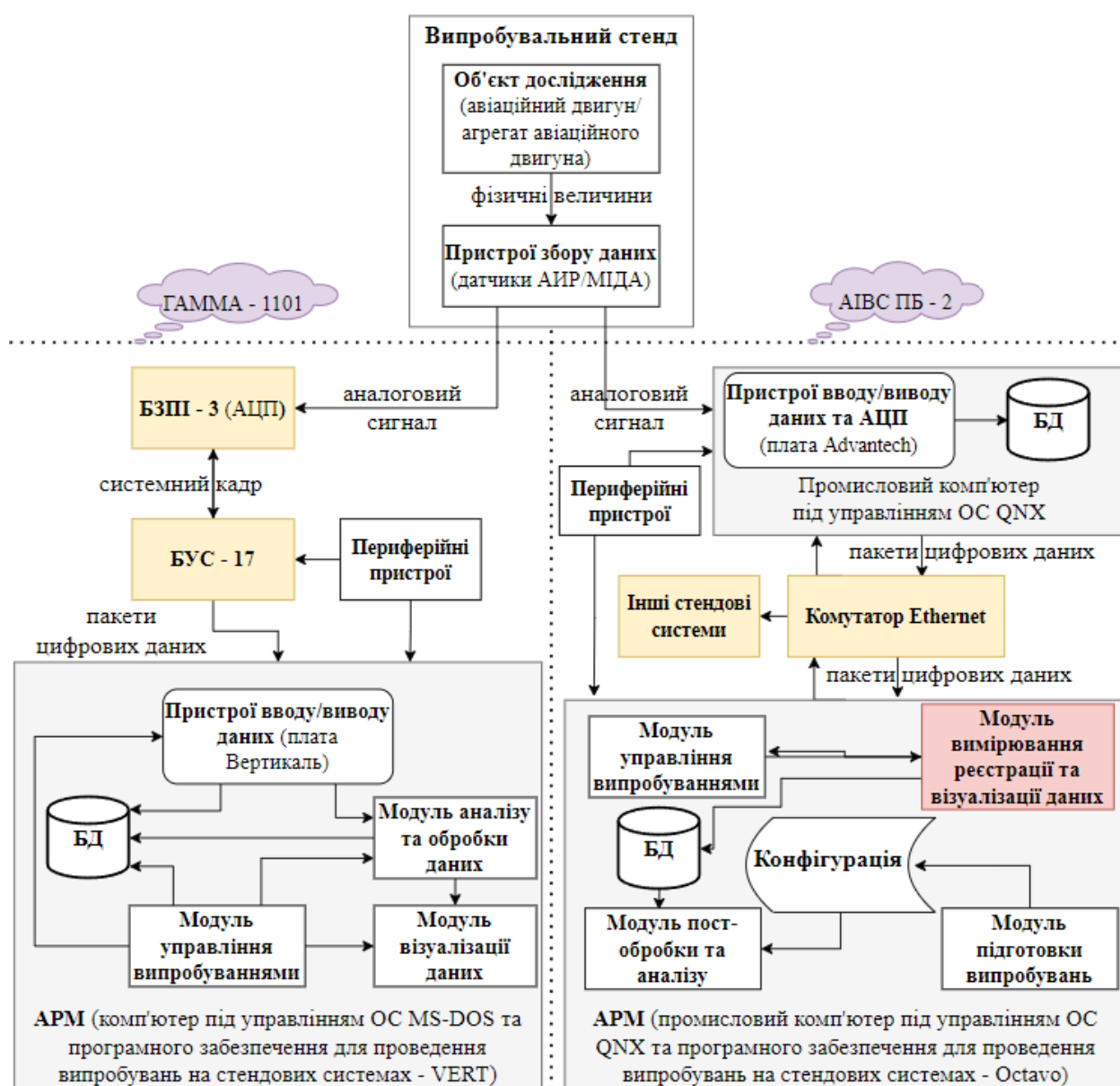


Рисунок 1.1 – Архітектура систем стендових випробувань ГАММА та AIBC

Обидві системи ГАММА – 1101 та АІВС ПБ – 2 виконують завдання збору даних з двигуна за допомогою датчиків АІР, МІДА розташованих на авіаційному двигуні. Положення двигуна фіксується за допомогою

випробувального стенда. Стрілочками позначений шлях даних від об'єкта досліджень до програмного забезпечення для проведення стендових випробувань. Основні відмінності полягають у різній архітектурі систем, часі створення та програмах для проведення випробувань на стендових системах реального часу.

На рисунку блок «Периферійні пристрої» включає в себе блок безперервного живлення, пристрої вводу виводу, перемикачі апаратного запуску систем стендових випробувань.

У системі AIBC ПБ – 2 прямокутник «комутатор Ethernet» передає дані через локальну мережу випробувального стенду. Він необхідний для управління промисловими комп'ютерами з АРМ оператора та для передачі даних на «інші стендові системи». Промислові комп'ютери являються вузлами локальної мережі, також комутатор Ethernet інтегрує інші стендові системи до яких входять:

- print server – відповідає за друк протоколів випробувань;
- пульт експериментально – дослідницького відділу, що відповідає за процес випробувань. На ньому знаходяться тумблери, важелі, індикатори для запуску систем авіаційного двигуна та монітори для візуалізації даних з AIBC ПБ – 2;
- пульт вібрографування відповідає за відображення та збереження даних вібрації на двигуні;
- комп'ютери вторинної обробки призначені для створення звітів випробувань у різних форматах. Вони необхідні через особливість операційної системи QNX, а саме, система не має інструментів для взаємодії з ОС Windows. Через те, що на виробництві всі програми протоколювання, аналізу, та конвертації даних працюють на Windows, необхідно передавати дані випробувань на інші комп'ютери. Це реалізовано за допомогою локальної мережі Ethernet.

Більш детальний аналіз систем стендових випробувань проводився за таким планом:

- введення;
- область використання;
- параметри що вимірюється;
- склад системи;
- умови експлуатації;
- особливості роботи.

1.2.1 Аналіз стендової системи ГАММА – 1101

Через специфіку випробувань авіаційних двигунів необхідна висока точність вимірювань та якість, що перевірена часом. Саме тому, в наш час, на випробувальних стендах ще можна побачити систему реєстрації стендових параметрів ГАММА – 1101.

Комплекс бортових агрегованих засобів збору, перетворення та реєстрації інформації ГАММА – 1101 використовується для вимірювання параметрів об'єктів, що досліджуються, перетворених в електричні сигнали, запису результатів вимірювань паралельним двійковим кодом на фізичні носії інформації та забезпечення можливості передачі даних телеметричним каналом при льотних випробуваннях авіаційної техніки [4].

Область використання – стендові та льотні випробування авіаційної техніки включаючи випробування літального апарату, його силових установок та різноманітного обладнання.

Відмінністю комплексу є його гнучкість, що дозволяє на основі базового складу будувати різноманітні комплекси як по функціональному призначенню так і по об'єму вимірюваної інформації [4].

Бортовим інформаційно – вимірювальним комплексом ГАММА – 1101 виконується збір та перетворення таких сигналів [4]:

- напруга постійного струму;
- напруга змінного струму;
- омичний спротив;
- послідовність імпульсів;
- разові команди;
- кодова інформація за ГОСТ 18977 – 79.

До складу бортового інформаційного комплексу входять [4]:

- засоби первинного прийому та перетворення інформації;
- засоби збору даних;
- засоби реєстрації інформації;
- засоби організації;
- засоби передачі даних по каналах зв'язку та магістралі комплексу;
- засоби живлення.

Комплекс ГАММА – 1101 має модульну структуру (рис. 1.2) з програмованим розподіленням частоти опитування, адресним опитуванням по многоканальним перетворювачам інформації та спільній, магістральній лінії зв'язку [4].

Засоби первинного прийому та перетворення інформації призначені для прийому та перетворення ненормалізованих сигналів з первинних перетворювачів (датчиків) та нормалізації їх до напруги постійного струму в межах від 0 до 6 В [4].

Засоби збору даних, що являють собою многоканальні перетворювачі інформації, призначені для прийому, комутації та перетворення в цифровий імпульсний паралельний двійковий код аналогових, аналого – дискретних та дискретних електричних сигналів, що поступають від датчиків, або організуючих та нормалізуючих пристроїв [4].

Засоби реєстрації інформації призначені для збереження даних вимірювань на фізичних носіях [4].



Рисунок 1.2 – Зовнішній вигляд ГАММА – 1101

Засоби організації призначені для синхронізації та управління роботою бортового інформаційно – вимірювального комплексу.

Засоби передачі даних по каналах зв'язку та магістралі комплексу призначені для перетворення формату даних.

Засоби живлення призначені для забезпечення напругою живлення інформаційно – вимірювальний комплекс в діапазоні 18 – 35 В [4].

Для організації роботи блоків в комплексі використовується магістральна лінія зв'язку – магістраль комплексу. До магістралі комплексу

поступає вся кодована інформація. Кожне 10 – розрядне двійкове слово видається при надходженні тактового сигналу опитування. Послідовність видачі кодових слів визначається за допомогою МПІ [4].

Тактові сигнали опитування представляють собою 4 – розрядні кодові слова далі – КТСС. Кожному МПІ надається адреса – порядковий номер за допомогою спеціального кодового перемикача. В МПІ за допомогою кодового селектора виконується вибірка тільки тих КТСС, код яких співпадає з кодом встановленим на перемикачі МПІ. З вибраних КТСС створюються в МПІ сигнали опитування, що видають на вихід інформаційний код результату перетворення. Послідовність КТСС створюється в блоці організації БУС – 17 за заданою програмою [4].

Магістраль комплексу складається з двох частин: адресно – опитувальної та інформаційно – відповідної. По адресній лінії зв'язку передаються КТСС. Інформаційна частина магістралі призначена для передачі інформаційних кодових слів результату перетворення вхідних вимірювальних сигналів [4].

Комплекс ГАММА – 1101 зберігає працездатність при наступних умовах експлуатації [4]:

- температура довкілля -60 – 60°C;
- відносна вологість повітря до 98% при 35°C;
- атмосферний тиск від 5 до 800 мм рт. ст.;
- вібраційні навантаження від 5 до 2000 Гц, при прискоренні 5 g та амплітуді зсуву до 2,5 мм;
- ударні прискорення до 12 g;
- лінійні прискорення до 10 g.

Структура даних комплексу ГАММА – 1101 визначається принципом побудови структури комплексу та характеризується розміщенням елементів інформації на ділянці кадру. Кадр комплексу складається з підкадрів, що являються кадрами МПІ. Кадр комплексу може вміщувати від 1 до 8 МПІ.

Кадр МПІ формується окремо та самостійно в кожному МПІ. Довжина кадру комплексу в залежності від програми може змінюватися від 64 до 4096. Підкадр завжди складається з 64 позицій. Кадр МПІ може складатися з кадрів БПІ. Кадр БПІ містить 8 позицій рівномірно розташованих в кадрі МПІ. Кадр МПІ містить до 8 кадрів БПІ. В комплексі ГАММА – 1101 інформаційні кодові слова видаються з частотами 512; 1024; 2048; 8192; 16384 Гц. Зміна частоти кодових слів пов'язана зі зміною частоти опитувань одночасно по всім каналам МПІ. Кодове слово, що являє собою результат перетворення інформації по конкретному каналу буде займати у кадрі комплексу конкретне положення, що визначається номером МПІ. Вся отримана кодова інформація в бортовому комплексі ГАММА – 1101 поступає до магістралі комплексу. Данні по магістралі поступають на блок перетворення паралельного інтерфейсу у послідовний потік стандарту ARINC (Aeronautical Radio Incorporated). Затримка до моменту візуалізації на промислових електронно обчислювальних машинах (ПЕОМ) складає не більше 0,2 с [4].

Блок перетворення використовується для об'єднання декількох систем ГАММА – 1101 в разі великої кількості вимірюваних параметрів на стенді. ARINC потік поступає на одноканальну плату «Вертикаль» (рис.1.3). Плата зчитує дані та передає їх до адресного простору комп'ютера по шині ISA (Industry Standard Architecture).

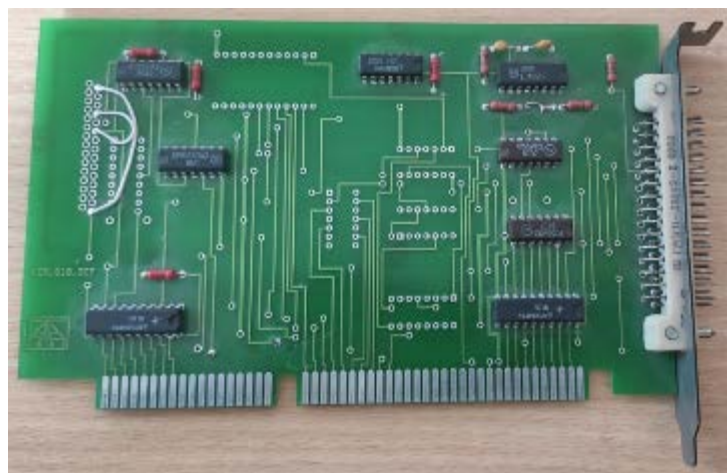


Рисунок 1.3 – Плата «Вертикаль»

1.2.2 Дослідження стендової системи АІВС ПБ – 2

Автоматизована інформаційно – вимірювальна система АІВС ПБ – 2 (рис.1.4) представляє собою апаратно – програмний комплекс призначений для проведення випробувань двигунів на випробувальному стенді ДП «ІВЧЕНКО – ПРОГРЕС».



Рисунок 1.4 – Зовнішній вигляд АІВС ПБ – 2

АІВС ПБ – 2 використовується для [5]-[6]:

- вимірювання параметрів виробів, що випробуються та стендових систем;
- автоматизованої обробки вимірюваних параметрів, обчислення фізичних та розрахункових значень параметрів;
- відображення параметрів виробу, що випробується на відеомоніторах у вигляді цифрових значень, текстових повідомлень, графіків;

- обробки параметрів згідно вимогам конструкторської документації та виводу друкованого протоколу результатів вимірювань;
- запису усіх вимірюваних параметрів випробування виробу до бази даних [5]-[6];
- відновлення з бази даних поточних випробувань параметрів виробу для повторного розрахунку;
- забезпечення обнуління та атестації вимірювальних каналів;
- розмежування доступу користувачів до програмного забезпечення та результатам випробувань.

Системою виконується збір та перетворення таких сигналів [5]-[6]:

- напруга постійного струму;
- напруга змінного струму;
- омичний спротив;
- послідовність імпульсів;
- разові команди;
- сила струму.

Програмна підсистема AIBC ПБ – 2 складається з наступних підпрограм [5]-[6]:

- підпрограма підготовки до випробувань (конфігурування, тарировка, метрологія);
- підпрограма збору інформації;
- підпрограма збереження даних;
- допоміжні сервіси.

Система функціонує під управлінням спеціалізованої операційної системи жорсткого реального часу QNX 6.4.1 та програмного забезпечення для проведення стендових випробувань – Octavo. Система візуалізації реалізована на базі графічного користувацького інтерфейсу Photon [5]-[6].

Конструктивно, інформаційно – вимірювальний комплекс (ІВК) складається з двох підсистем [5]-[6]:

- станції збору даних (СЗД) побудованої на базі трьох IBM ПК – сумісних промислових комп'ютерів з встановленими в них платами Advantech PCI, пристроями зв'язку з об'єктом (ПЗО), та комутатором консолі, комутатором Ethernet і джерелом безперервного живлення;
- підсистеми реєстрації та відображення (ПРВ). Що являє собою три панельних промислових комп'ютера встановлених на стенді (аварійного, технологічного та комп'ютеру для контролю вимірюваних приладів та автоматики) [5]-[6].

Технічні засоби ІВК експлуатуються в наступних умовах [5]-[6]:

- температура середовища від 5 до 45°C;
- відносна вологість повітря не більше 80% при 35°C без конденсації вологи;
- атмосферний тиск від 630 до 805 мм. рт. ст.;
- живлення від мережі змінного струму 220 В (-15%; +10%;), 50 Гц.

Комп'ютери ІВК являються вузлами локальної мережі Ethernet, також існує можливість інтеграції з іншими стендовими системами: print server, пульт експериментально – дослідницького відділу, пульт вібрографування, комп'ютери вторинної обробки. Апаратна частина комплексу складається з пристроїв зі стандартними інтерфейсами та допускає заміну на аналогічні за типом пристрої [5]-[6].

АІВС забезпечує паралельне опитування всієї номенклатури сигналів з синхронізацією за часом з максимальною частотою 50 Гц на канал та реєстрацією на жорсткий диск не менш як протягом 10 годин.

Сумарна апаратно – програмна затримка передачі інформації вимірювального каналу від моменту вимірювання сигналу на виході датчику до моменту прийому драйвером ПРВ складає не більше 20 мс. Затримка до моменту візуалізації на ПЕОМ складає не більше 0,25 с для аналогових та 0,1 с для дискретних сигналів [5]-[6].

Вузли СЗД та ПЕОМ об'єднані мережею Ethernet за допомогою комутатора. Джерела безперервного живлення забезпечують роботу системи протягом 10 – 15 хвилин та дозволяють безпечно завершити роботу в разі відключення живлення [5]-[6].

AIVS працює таким чином. Сигнали від первинних пристроїв перетворення поступають на СЗД звідки йдуть на вхід плати пристрою сполучення об'єктового (ПСО) на пряму або через перетворювачі сигналів.

Плати ПСО встановлені у кріплення промислових комп'ютерів, виконують перетворення сигналів до цифрового коду, що зберігається у внутрішніх регістрах, які знаходяться у адресному просторі вводу – виводу електронно обчислювальної машини та можуть бути зчитані до внутрішніх змінних програм управління та вимірювання [5]-[6].

Результати вимірювання параметрів та стан компонентів СЗД по мережі передаються на ПЕОМ, та відображаються на моніторах у вигляді чисельного значення, або в вигляді графіку.

На кожному вузлу СЗД реєструються вимірювані та вираховані параметри та зберігаються в локальних базах даних цих вузлів. По завершенню випробувань всі локальні бази даних об'єднуються в спільну, з урахуванням міток часу та алгоритмів перерахунку параметрів. Спільна база архівується та зберігається на ПЕОМ аварійній в директорії з номером стенду, номером двигуна, датою. У цій директорії також зберігаються звіти та заміри, що виконувалися протягом випробування [5]-[6].

Система реєстрації починає роботу за командою оператора, після завантаження операційної системи та системи стендових випробувань Octavo [5]-[6].

1.2.3 Порівняння систем стендових випробувань

Порівняння проводилося на основі джерел інформації зазначених в аналізі кожної з систем, зображено в таблиці 1.1.

На основі проведених досліджень, можна зробити висновок, що АІВС ПБ – 2 є порівняно сучасною системою стендових випробувань, що характеризується: підтримкою всіх наявних промислових стандартів збору даних, можливістю обміну даними в мережі, значно більший пропускну здатність передачі даних, більшою швидкістю опитування датчиків, підтримкою сучасних мов розробки конфігурування та налаштування плат, підтримкою нових сервісних додатків, легко налаштовуваним та різноманітним візуальним інтерфейсом, та кращою надійністю збереження даних випробувань.

Таблиця 1.1 – Порівняння ГАММА – 1101 та АІВС ПБ – 2

Критерій порівняння	ГАММА – 1101	АІВС ПБ – 2
1	2	3
Максимальна частота опитування всіх каналів при повному навантаженні	4 Гц	50 Гц
Максимальна швидкість передачі даних	50 – 80 мб/с	125 – 200 мб/с
Мобільність	Присутня (за рахунок модульної структури)	Відсутня
Можливі умови експлуатації	Можливо використовувати в жорстких умовах	Можливо використовувати в нормальних умовах
Дані надходять на систему у стандарті	ARINC	ARINC
Шина вводу виводу плати	ISA	PCI

Продовження таблиці 1.1

1	2	3
Затримка сигналу від вимірювання до візуалізації	Не більше 0,2 с	Не більше 0,25 с
Підтримка мережі Ethernet	Відсутня	Наявна
Підтримка операційної системи	DOS, Windows 98	Linux, Windows 10
Рік створення системи	1982	2000
Можливість оновлення програмної оболонки для проведення випробувань	Відсутня	Наявна за додаткову плату
Безпека програмної системи та надійність збереження даних	Присутня за рахунок створення копій системи та наявності рівнів допуску	Висока за рахунок дублювання баз даних, рівнів допуску, копій системи та підказок системи
Доступність системи в наш час	Не доступна для придбання	Доступна для придбання

До серйозних недоліків можна віднести майже повну відсутність мобільності, досить обмежені умови експлуатації.

Через це, усі сучасні розробки стендових систем збору даних на підприємстві проводяться на базі архітектури системи стендових випробувань АІВС ПБ – 2.

1.3 Дослідження програмних засобів для проведення випробувань на стендових системах реального часу

Програмне забезпечення для проведення випробувань на стендових системах реального часу ГАММА – VERT вже застаріле, воно не підтримує

сучасні рішення візуалізації, не підтримує сучасні додатки для створення графіків та протоколювання, не підтримує відображення курсора миші тому, аналіз програмної оболонки VERT не проводився.

Далі наведений аналіз програмного забезпечення для проведення випробувань Octavo та Octopus. Дослідження проводилося за наступним планом:

- структура програми;
- функції програмного забезпечення;
- процедура зміни версії двигуна;
- процедура додавання нових плат;
- переваги;
- недоліки.

1.3.1 Аналіз програмного забезпечення для проведення випробувань на стендових системах АІВС ПБ – Octavo

Програмне забезпечення стенду реального часу АІВС ПБ – 2 має назву Octavo використовується для проведення випробувань авіаційних двигунів [7].

Програмне забезпечення Octavo має наступну структуру [7]:

- інтегрована підпрограма підготовки виробів до випробувань;
- інтегрована інтерактивна підпрограма управління випробуваннями;
- підпрограма вимірювання та реєстрації параметрів в режимі «жорсткого» реального часу [7];
- підпрограма створення протоколів у режимі реального часу;
- підпрограма візуалізації випробувань в режимі реального часу;
- інтегрована інтерактивна підпрограма пост – обробки результатів вимірювань та аналізу [7].

Всі інтегровані підпрограми виконані у вигляді єдиної інтегрованого середовища підготовки до випробувань, проведення та обробки випробувань з єдиною базою проектних даних та випробувань (рис. 1.5) [7].



Рисунок 1.5 – Структурна схема Octavo [7]

Всі інтерактивні підпрограми об'єднанні спільною вбудованою графічною програмною оболонкою – Octavo з інтуїтивно зрозумілим графічним віконним інтерфейсом (рис.1.6) [7].

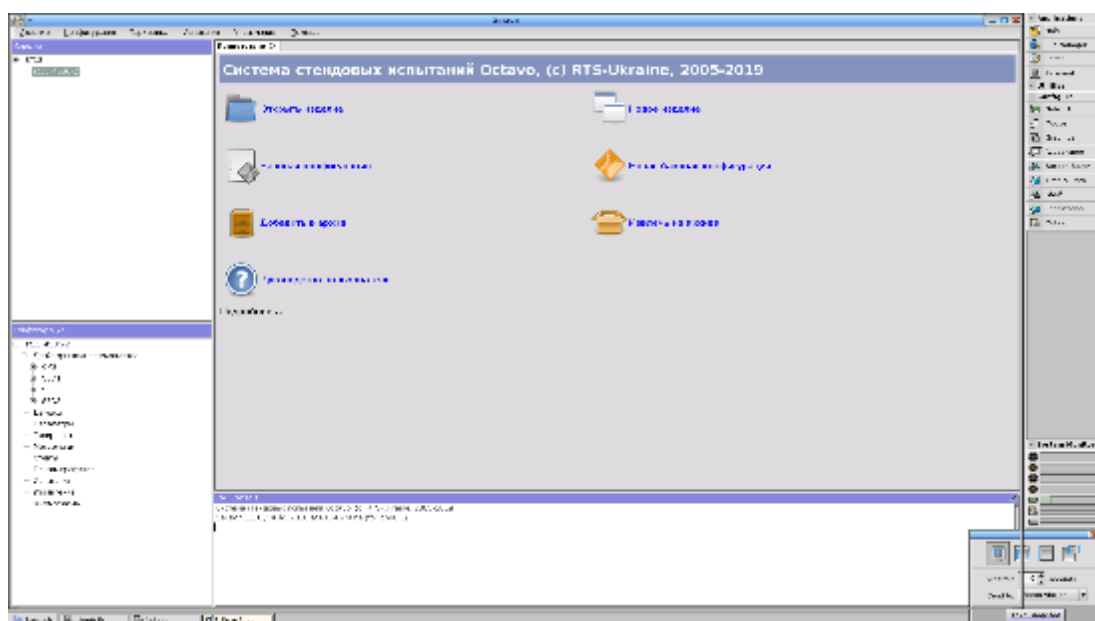


Рисунок 1.6 – Головний екран Octavo

Функції підпрограми підготовки випробувань [7]:

- формування бази датчиків виробу з таблицями для перерахунку даних за алгоритмами;
- формування списку параметрів що будуть вимірюватися, також вказуються який датчик вимірює параметр та до якої плати будуть надходити дані;
- створення екранів управління та відображення у реальному часі;
- створення шаблонів для звітів [7];
- виконання атестації ліній та налаштування датчиків;
- створення функціональних блоків для розрахунку с декількома параметрами;
- створення та налаштування базової конфігурації системи вимірювання.

Функції підпрограми управління випробуваннями [7]:

а) управління запуском та зупинкою системи реєстрації параметрів:

- 1) завантаження програм підпрограми випробувань;
- 2) зупинка випробувань;
- 3) об'єднання отриманих баз даних та перерахунок даних за

алгоритмами;

б) моніторинг роботи підпрограми збору даних[7]:

- 1) моніторинг стану мережі та процесів, що виконуються;
- 2) моніторинг часу випробування;
- 3) моніторинг дискового простору та оперативної пам'яті;

в) задання циклограми запису до бази даних;

г) ручний запуск та зупинка випробувань.

Функції підпрограми вимірювання, реєстрації та візуалізації фізичних параметрів [7]:

- збір значень параметрів по всім каналам системи та формування бази даних в контролері плати;

- опитування каналів вводу/виводу пристроїв зв'язку з об'єктом ПЗО – плат Advantech [7];

- виконання процедур обчислення;
- формування протоколу випробувань в режимі реального часу;
- запис даних на жорсткий диск;
- візуалізація випробувань в режимі реального часу.

Функції підпрограми пост – обробки та аналізу [7]:

- приведення значень, що були виміряні до фізичних величин на основі характеристик каналів вимірювання;
- формування звітів у вигляді таблиць та графіків;
- друк документів зі звітами;
- ведення архівів випробувань.

Програмне забезпечення для AIBC ПБ – 2 розроблене на двох мовах програмування («Java» та «C і C++ »). Функціонує на операційній системі QNX 6.4.1, тому що QNX демонструє високу надійність, стійкість до збоїв, збереження отриманих даних при аварійних ситуаціях [7].

Процедура зміни версії двигуна в Octavo виконується за таким алгоритмом:

- відкрити програмну директорію з файлом продукту. Архівувати поточний продукт, перемістити архів до папки з архівами продуктів двигунів такої самої моделі;

- відкрити папку з архівами продуктів необхідної версії. Розархівувати продукт, перенести продукт до програмної директорії, приклад зображено на рисунку 1.7;

- відкрити програмну директорію з файлом базової конфігурації. Архівувати поточну конфігурацію, перемістити архів до папки з архівами конфігурацій двигунів такої самої моделі;

- відкрити папку з архівами конфігурацій необхідної версії. Вибрати конфігурацію, щ обула оновлена у той самий час, що і вибраний продукт,

розархівувати конфігурацію, перенести її до програмної директорії, приклад зображено на рисунку 1.8;

- запустити Octavo, перевірити що зміна версії двигуна пройшла вдало.

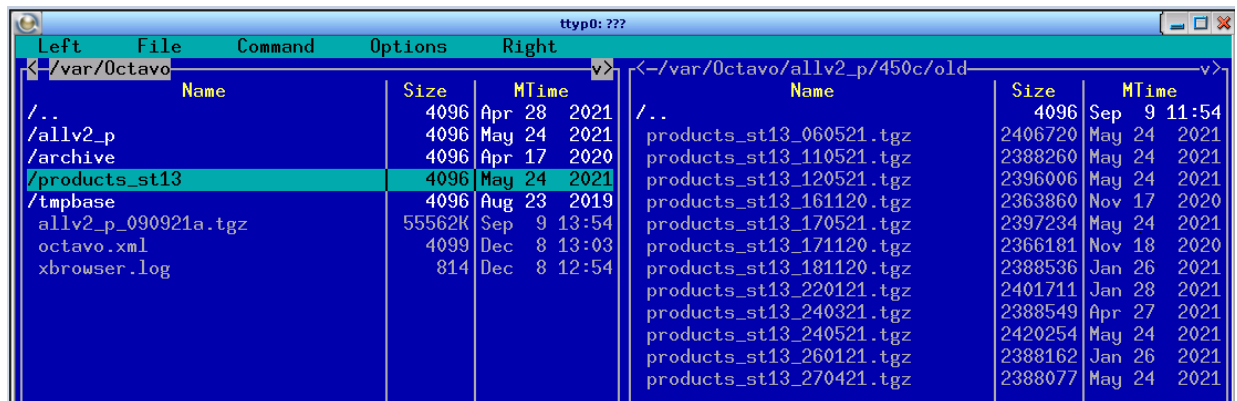


Рисунок 1.7 – Процедура зміни продукту Octavo

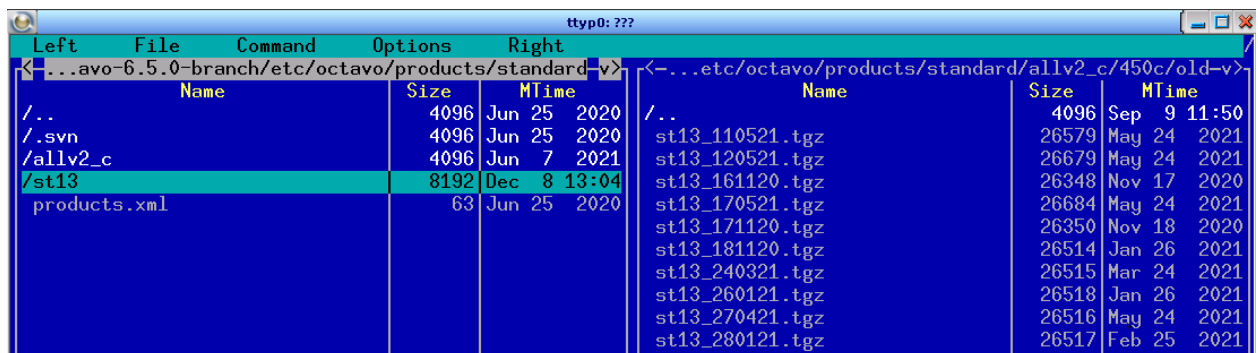


Рисунок 1.8 – Процедура зміни базової конфігурації Octavo

Процедура додавання нових плат до програмного забезпечення Octavo виконується за таким алгоритмом:

- запустити Octavo, перейти до базової конфігурації проекту;
- вибрати пункт меню «драйвери плат», з випадаючого списку вибрати «додати нову плату», вибрати бажану плату зі списку доступних;
- вибрати останню додану плату, відкрити її, налаштувати параметри плати згідно ТО для цієї плати та поставлених задач (рис. 1.9);

- у меню базової конфігурації вибрати пункт «синхронізація», після завершення зберегти проєкт, закрити базову конфігурацію;
- додати необхідні параметри до плати, та включити їх до екранів програмного забезпечення Octavo.

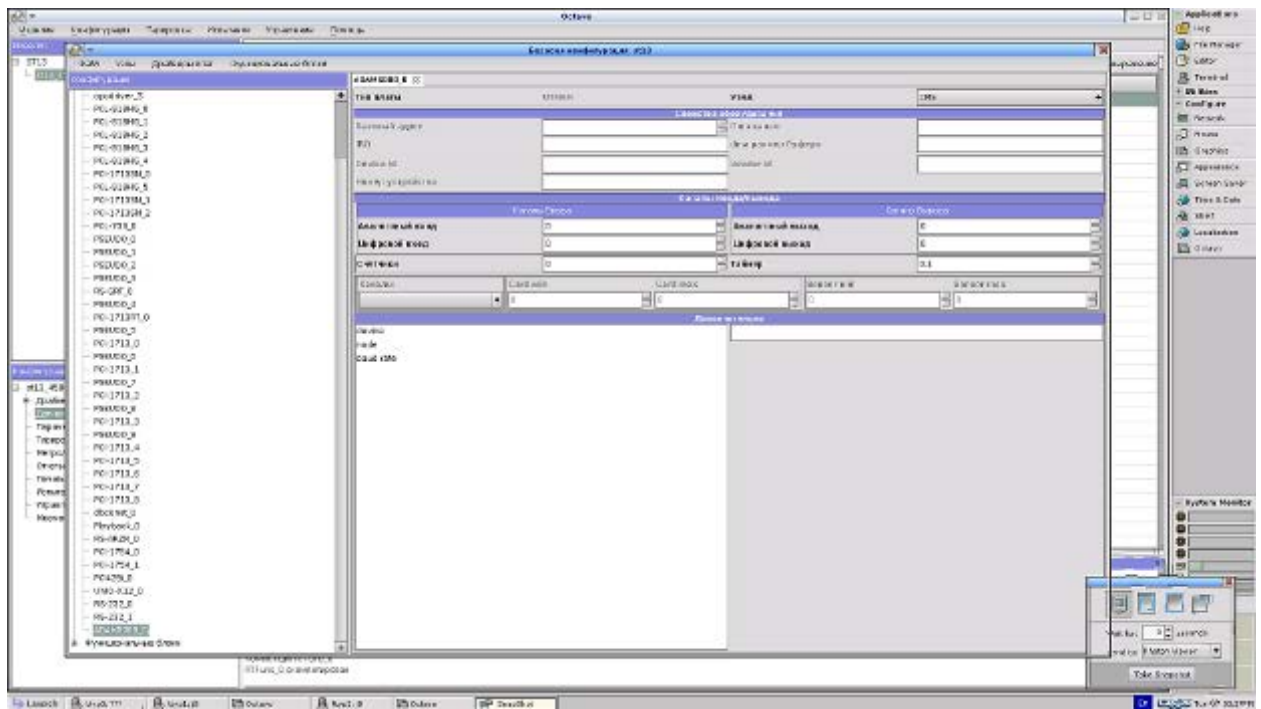


Рисунок 1.9 – Процедура налаштування нової плати Octavo

Недоліком розглянутого рішення є складна інтеграція з комп'ютерами під управлінням ОС Windows, довга процедура зміни версії двигуна, закритий програмний код проєкту, що веде до значних матеріальних втрат у випадку модернізації старих випробувальних стендів та не дозволяє проводити модернізацію програмної системи власними силами підприємства.

Задля забезпечення самостійного циклу розробки та незалежності від компанії QNX, для зменшення накладних витрат на підтримку програмної оболонки, для збільшення швидкості розробки та введення в експлуатацію програмних модулів для передових розробок, для підвищення рівня якості випробувань та не допущення розповсюдження секретної інформації, на підприємстві було прийнято рішення розробки системи стендових

випробувань на основі архітектури AIBC ПБ – 2 з програмним забезпеченням Ostopus на основі графічної мови програмування LabVIEW.

1.3.2 Аналіз програмного забезпечення для проведення випробувань на стендових системах AIBC ПБ – Ostopus

Програмне забезпечення Ostopus було розроблено для стендів реального часу на архітектурі AIBC ПБ – 2, використовується для проведення випробувань авіаційних двигунів на випробувальних стендах.

Структура програмного забезпечення Ostopus:

- а) підпрограма підготовки випробувань;
- б) підпрограма управління випробуваннями:
 - 1) модуль збору та збереження даних;
 - 2) модуль розрахунку;
 - 3) модуль передачі даних;
- в) підпрограма візуалізації;
- г) підпрограма пост-обробки та аналізу:
 - 4) модуль розрахунку;
 - 5) модуль протоколювання;
 - 6) модуль візуалізації.

Далі наведено структурну схему програмного забезпечення для проведення стендових випробувань Ostopus (рис. 1.10). Червоним прямокутником на схемі позначено підпрограму програмного забезпечення, яка була проаналізована більш докладно, цю підпрограму було обрано для вдосконалення.

Програмне забезпечення має модульну структуру. Складається з підпрограми підготовки випробувань, підпрограми управління, підпрограми візуалізації, підпрограми пост-обробки та аналізу, це забезпечує можливість гнучких налаштувань програми під різні типи двигунів та різні випробувальні

стенди, також модульна структура підвищує швидкість розробки ПЗ за рахунок створення екранів візуалізації операторами, це зменшує навантаження на розробників ПЗ.



Рисунок 1.10 – Структурна схема Ostorus

Процедура зміни версії двигуна у програмному забезпеченні Ostorus виконується за наступним алгоритмом:

- відкрити у редакторі файл Start.ini, що знаходиться у директорії проєкту;
- змінити значення поля XlsFile на назву таблиці підпрограми підготовки випробувань з необхідною версією двигуна (рис. 1.11);
- змінити значення поля ScreensFolder на назву папки з екранами необхідної версії двигуна;
- зберегти зміни, запустити Ostorus, перевірити що зміни внесені вірно.

Процедура додавання нових плат у програмному забезпеченні Ostorus виконується за наступним алгоритмом:

- дослідження нової плати, програмне налаштування роботи плати, вибір необхідних режимів роботи на основі поставленої задачі;
- розробка VI для зчитування та перетворення даних з плати;
- вбудовування розробленого VI до проєкту програмного забезпечення для проведення стендових випробувань Ostorus;

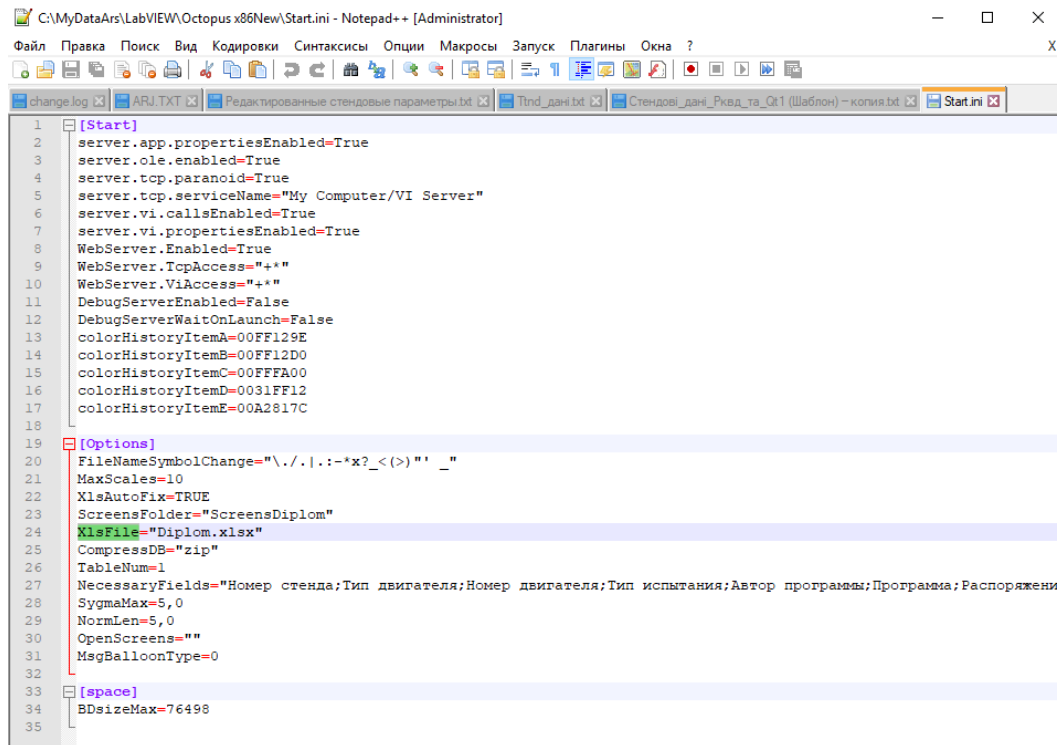


Рисунок 1.11 – Процедура зміни версії двигуна у програмі Octavo

- налаштування передачі даних від розробленого VI до main VI проєкту, компіляція проєкту.

Далі наведений докладний аналіз підпрограми Ostorus та опис їх функцій.

1.3.3 Аналіз підпрограми підготовки випробувань Ostorus

Завдання підготовки програмного забезпечення Ostorus до випробувань виконує підпрограма підготовки. Підпрограма підготовки

випробувань представляє собою книгу Microsoft Excel, створена на основі середовища розробки і платформи для виконання програм – LabVIEW.

Для коректного функціонування підпрограми підготовки випробувань, створена книга Microsoft Excel повинна мати наступні листи [8]: паспорт, екран, таблиця, пристрій вводу даних.

Програма для проведення стендових випробувань Ostorus підтримує використання таких пристроїв вводу даних [8]:

- ADAM;
- ComDevice;
- Modbus TCP;
- SLX;
- ГРТ;
- AdvantechPCI-1713;
- AdvantechPCI-1750;
- інші пристрої.

Книгу конфігурації можливо розширити наступними листами (рис. 1.12) [8]: константи, розрахункові.

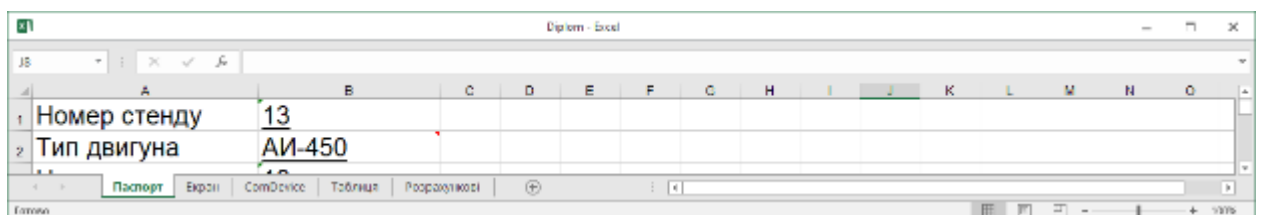


Рисунок 1.12 – Приклад Excel листів у підпрограмі підготовки випробувань

Лист «Паспорт» (рис. 1.13) використовується для створення шляху та шапки бази даних. На ньому знаходяться комірки, перший стовпчик – назва, другий – значення. Існують пари назва-значення, обов’язкові для введення, у випадку їх відсутності – вони будуть створені автоматично. Обов’язкові значення: номер стенда, тип двигуна, номер двигуна, тип випробування, автор програми, програма, розпорядження [8].

Формування шляху для створення бази даних. У каталозі DB формується папка з назвою першого поля «номер стенда», у цій папці створюється під папка, яка складається з полів «тип двигуна», «номер двигуна». У цю папку будуть складатися папки з базами даних та іменами типу 20180827T161749+03. Для паспорта з прикладу буде сформований такий шлях db\S13\AI-450-2T_10\20180827T161749+03 [8].

	A	B	C
1	Номер стенду	<u>13</u>	
2	Тип двигуна	<u>AI-450</u>	
3	Номер двигуна	<u>10</u>	
4	Тип випробування		<u>2</u>
5	Автор програми		<u>38</u>
6	Програма	Диплом	
7	Розпорядження	Магістр ордина	
8	Шифр витрат	Example	

Рисунок 1.13 – Excel лист «Паспорт»

Лист «Екран» (рис. 1.14) використовується для відображення параметрів на фронтальній панелі віртуального пристрою. На цьому листі 1 рядок – назва, 2 рядок – значення. Комірка «Частота» – частота запису в базу даних та виводу на пристрій, вимірюється в разях за секунду. Комірка «Увімкнено» – ознака, що пристрій буде оброблятися. Пристрій також не увімкнеться, якщо в нього не буде прийнятих та переданих параметрів. 3 стовпчик – назва прийнятих та переданих параметрів, при роботі замість назви будуть підставлені значення параметрів [8].

	A	B	C	D	E	F	G	H	I	J	K
1	Частота	Увімкнено									
2	2	ИСТИНА									
3	Увімкнен	Ім'я	Id	Формат	Джерело	Тип даних	Тип	Мінімум	Максимум	Індикатор1	1

Рисунок 1.14 – Excel лист «Екран»

Лист «Таблиця» (рис. 1.15) є шаблоном для створення таблиці з вимірами – звіта (рис. 1.16). Шапка – будь який текст над таблицею, буде записаний як коментар. Комірка «Час» до неї записується час вимірювання при роботі програми. У випадку відсутності комірки «Час» програма Octorus створить шаблон таблиці з всіма необхідними комірками. Комірка «Назва» – у цю комірку буде розміщено назву вимірювання. «Temp1» – назва параметру, що взята як приклад. Окрім назви параметра є можливість вказати його ID. В обох випадках параметр буде записаний до таблиці під час роботи вимірювальної системи. Звіт необхідний для швидкого доступу до даних вимірів та друку значень параметрів під час роботи вимірювальної системи [8].

	A	B	C	D	E	F	G	H	I	J
1	Час	Назва	Temp1	Hum1	Temp2	Hum2	Temp3	Hum3	TempOld	HumRand
2										

Рисунок 1.15 – Excel лист «Таблиця»

Лист «Пристрої вводу даних», в даному прикладі (рис. 1.17) використовується ComDevice для читання даних з пристроїв через Com порт [8].

Комірка «Частота» – частота запису в базу даних, вимірюється в разях за секунду. Комірка «Порт» – номер порту на ПК. Комірка «TimeOut» – час очікування відповіді пристрою. Комірка «Baud» – швидкість СОМ порту в бодах. Комірка «Parity» – парність СОМ порту [8].

Комірка «Data bits» – кількість бітів виділених під значення. Комірка «Stop bits» – стоповий біт пакету. Комірка «TermCh» – код розділового символу в пакеті даних. Комірка «TermChEn» – використовувати розділовий символ або розділяти за паузою. Комірка «Byte count» - довжина пакета в байтах. Комірка «Text» – будь який текст який можливо вставити в масив передачі, за допомогою тегу \$Text вказав його в шаблоні.

K	L	M	N	O	P
Time - Hum3	Temp2, Hum2, Temp3, Hum3 - Hum3	Time - TempOld	TempOld, HumRand - TempOld	Time - HumRand	TempOld, HumRand
10:38:26		44	10:38:26 29.00	10:38:26	54.13
10:38:26		44	10:38:26 29.00	10:38:26	55.77
10:38:26		44	10:38:26 29.00	10:38:26	55.63
10:38:26		44	10:38:26 29.00	10:38:26	54.55
10:38:27		44	10:38:27 29.00	10:38:27	55.18
10:38:27		44	10:38:27 29.00	10:38:27	55.37
10:38:27		44	10:38:27 29.00	10:38:27	54.93
10:38:28		45	10:38:28 29.00	10:38:28	55.71
10:38:28		45	10:38:28 29.00	10:38:28	55.54
10:38:28		45	10:38:28 29.00	10:38:28	56.59
10:38:29		45	10:38:29 29.00	10:38:29	56.24
10:38:29		45	10:38:29 29.00	10:38:29	56.24
10:38:29		45	10:38:29 29.00	10:38:29	55.38
10:38:30		45	10:38:30 29.00	10:38:30	55.87
10:38:30		45	10:38:30 29.00	10:38:30	55.21
10:38:30		45	10:38:30 29.00	10:38:30	56.90

Рисунок 1.16 – Звіт, створений на основі листу «Таблиця»

Рисунок 1.17 – Excel лист «ComDevice»

Комірка «ШаблВидачі» – рядок, який визначає порядок передачі даних. Комірка «ТипНомКан» – тип нумерації каналів. Комірка «Увімкнено» – ознака, що пристрій буде оброблятися. Пристрій також не увімкнеться, якщо в нього не буде прийнятих та переданих параметрів [8].

Комірка «Увімкнен» – індикатор, який визначає, чи буде параметр оброблятися та виводитися на екран. Якщо всі параметри пристрою будуть вимкнені – пристрій не підключиться. Може приймати значення «Істина» та «Брехня». Комірка «Ім'я» – назва параметру. Якщо поле Ім'я порожнє, призначається ім'я «Параметр1». Комірка «Id» – ідентифікатор параметра. Повинен бути унікальним навіть у вимкнених параметрах. Унікальність перевіряється програмою Ostorus. У разі його відсутності, призначається автоматично. Комірка «Формат» – комірка відповідає за представлення даних на індикаторах. Комірка «Канал» – ця комірка визначає номер початкового

біта даних який буде зчитаний з пакета даних, що приходить з підпрограми збору даних. Також використовується номер стопового біту, для визначення меж отриманого пакету даних. Комірка «Кількість» – комірка визначає кількість бітів даних, що будуть зчитані з пакета даних, починаючи з початкового біта, що вказаний у комірці «Канал». Комірка «ChanFormat» – комірка визначає представлення отриманих даних з підпрограми збору даних в підпрограмі візуалізації даних. Комірка «Тип даних» – Визначає тип даних параметра. У вказаному типі даних параметр запишеться в базу даних, типи даних що підтримуються вказані в таблиці. 1.2 [8].

Таблиця 1.2 – Типи даних, що використовуються в підпрограмі підготовки випробувань [8]

Тип даних	Значення, що може приймати
1	2
No type	–
I8	-128..+127
I16	-32768..+32767
I32	-2147483648..+2147483647
I64	- 9223372036854775808..+9223372036854775807
U8	0..255
U16	0..65535
U32	0..4294967295
U64	0..18446744073709551616
SGL (float)	+/- 3.4E-38..3.4E+38
DBL (double)	+/- -1.7E-308..1.7E+308
EXT (long double)	Представляє дійсне число подвійної точності з плаваючою точкою не менше 8 байт. В залежності від розміру займаної пам'яті може відрізнятися діапазон допустимих значень
Boolean	True False
String	"Будь який рядок"

Комірка «Тип» – визначає тип обробки сирих даних та необхідні значення, що він приймає. Використовують такі типи параметра (табл. 1.3). Комірка «Мінімум» та «Максимум» – очікувані межі параметра. Вони можуть не співпадати з межами налаштування датчика, проте краще коли співпадають, тому що датчик може бути зламаний в процесі тестування при надходженні на нього значення вимірюваної величини, що виходить за межі допустимих робочих діапазонів датчика. Це може статися під час метрологічної атестації каналів вимірювання та датчиків. Фахівець може помилково дати більше значення на датчик, тому повинна бути індикація цих випадків.

Межі визначають компоновку параметрів на шкалах графіка та попереджують про несправність датчика. Колір та інші атрибути шрифту в цій комірці використовуються для виділення параметра кольором при виході цього параметра за допустимі межі. Комірка «1,2,3» – ці поля містять різні дані в залежності від типу параметра [8].

Таблиця 1.3 – Типи обробки параметра та значення, що він приймає [8]

Тип	Поле 1	Поле 2	Поле 3	Поле 4	Примітки
1	2	3	4	5	6
Коди	–	–	–	–	Сирі дані
Константа	Значення константи	–	–	–	Може бути числом, символом або рядком
Поліном	Коефіцієнт 0	Коефіцієнт 1	Коефіцієнт 2	Коефіцієнт 3	Функція для розрахунку: $Y = \sum K * x^n,$ де Y – результат; x – код параметра n – ступінь полінома
Разова команда	inv	P	–	–	Інверсія – ввести («not» або «inv»)

Продовження таблиці 1.3 [8]

1	2	3	4	5	6
Ближні точки	Код 1-ої точки полінома	Значення 1-ої точки	Код 2-ої точки полінома	Значення 2-ої точки	Y – значення, що відповідає найближчому коду параметра
Лінійний	Код 1-ої точки полінома	Значення 1-ої точки	Код 2-ої точки полінома	Значення 2-ої точки	Інтерполяція лінійним методом
Формула	Нижня межа для 1 – ої формули, включно з вказаним числом	Верхня межа для 1 – ої формули, вказане число не включено в діапазон	1-а формула	–	Формула, яку вказали в комірці 3, буде працювати коли код параметра ≥ нижній межі, та < верхньої межі. Якщо параметр не входить в діапазон, то йому присвоюється значення нуль
К-ДСТУ 60584-1	Формула для розрахунку зсуву характеристики	Код параметра	Миливольти ¹	–	Розрахунок холодного спаю у мВ
KmV-ДСТУ 60584-1	Формула для розрахунку зсуву характеристики	Код параметра	°C 1	–	Перетворення °C у мВ
Розрахунковий	Нижня межа для 1 – ої формули, включно з вказаним числом. Дані беруться з комірки «Джерело»	Верхня межа для 1 – ої формули, вказане число не включено в діапазон. Дані беруться з комірки «Джерело»	1-а формула	–	Розрахунковий тип не приймає дані з датчиків. Використовується комірка «Джерело» для отримання даних параметра

Комірка «Індикатор» – назва індикатора и тип виводу параметра. Існуючі індикатори перераховані в таблиці 1.4.

Таблиця 1.4 – Індикатори параметрів [8]

Тип індикатора [Назва; атрибут1; атрибут2;...]	Значення
1	2
Екран [Назва індикатора на екрані; номер (якщо декілька параметрів виводиться на 1 екран)]	Виводить параметр на вказаний екран. При створенні екрану назва індикатора записується в поле «Label»
Font [Назва індикатора на екрані; мінімум; максимум]	При виході параметра за вказаний діапазон, атрибути шрифту індикатора змінюються на ті, що встановлені у цій комірці
Sound [Назва звуку у папці Sounds; гучність; мінімум; максимум]	Звук відтворюється при виході параметра за вказаний діапазон
SLX1 [канал; кількість каналів в пакеті даних; тип даних; ChanFormat]	SLX1 – пристрій виводу даних «SLX» Отримує дані типів: I16 та Boolean
ComDevice [позиція в блоці; тип даних; ChanFormat]	ComDevice1 – пристрій виводу даних «Com порт». Отримує дані типів: U8 или String
PCI429 [канал; адреса параметра; маска]	PCI429 – пристрій виводу даних. Отримує дані типів: U32, I16 або U16
DataSocket [шлях до змінної; тип даних]	Пристрій виводу даних «DataSocket». Отримує дані з OPC Server, DataSocket Server. Отримує дані всіх типів
ModbusTCP [канал; кількість каналів в пакеті даних; тип даних; ChanFormat]	Пристрій виводу даних «ModbusTCP». Будь який пристрій що обмінюється даними по протоколу Modbus TCP. Отримує дані типів I16 та Boolean
ГРТемул [номер блока; канал]	Пристрій виводу даних "ГРТемул". Імітує роботу ГРТ

Лист «Розрахункові» (рис. 1.18) використовується для розрахунків з декількома параметрами з різних каналів, та для більш наглядного

представлення розрахунків для розробників та для виділення усіх розрахунків в окремий лист. Комірка «Зсув» – значення, що додається до параметру після всіх розрахунків з ним. Параметри у листі використовують в якості джерела ID каналів з листа «Пристрої вводу даних», або їх назву. Тобто параметри не зчитують дані з пристроїв, а тільки виконують розрахунки з ними [8].

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Частота	Увімкнено																
2	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
3	Увімкнено	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
4	ИСТОЧНИК	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
5	ИСТОЧНИК	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
6																		
7																		
8	ЛОЖЬ	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
9	ЛОЖЬ	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
10	ЛОЖЬ	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16

Рисунок 1.18 – Excel лист «Розрахункові»

1.3.4 Аналіз підпрограми управління випробуваннями Octorus

Програма Start.exe створена на основі середовища розробки і платформи для виконання програм – LabVIEW. Це головна, керуюча програма у програмному забезпеченні для проведення стендових випробувань. Функції підпрограми управління випробуваннями Octorus (рис 1.19) [8]:

а) управління запуском та зупинкою системи реєстрації параметрів [8]:

- 1) завантаження програм підпрограми випробувань;
- 2) зупинка випробувань;
- 3) об'єднання отриманих баз даних та перерахунок даних за

алгоритмами зазначеними у підпрограмі підготовки випробувань;

б) моніторинг роботи підпрограми збору даних[8]:

- 1) моніторинг стану мережі та процесів, що виконуються;
- 2) моніторинг часу випробування;
- 3) моніторинг дискового простору та ресурсів процесору;

в) ручний запуск та зупинка випробувань.

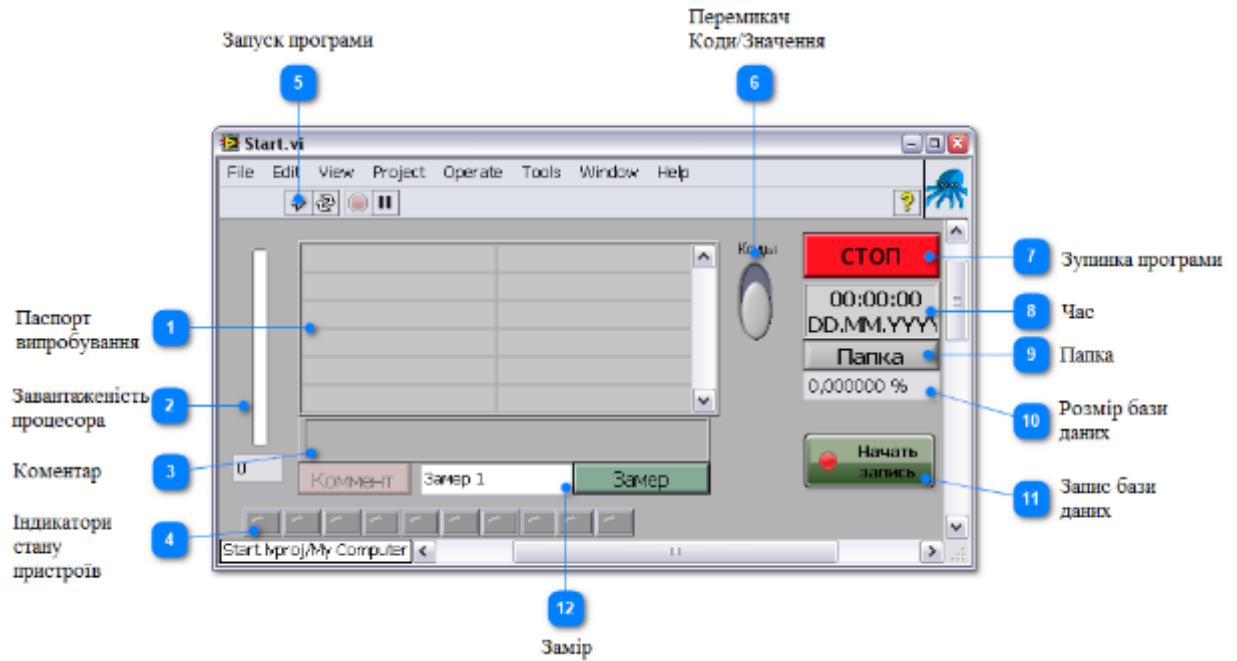


Рисунок 1.19 – Головне вікно підпрограми управління випробуваннями Octopus [8]

Паспорт випробування – заповнюється в листі «Паспорт» підпрограми підготовки випробувань. Перший стовпчик це назва, другий – значення [8].

Завантаженість процесора – демонструє завантаженість процесора у процентах.

Коментар – під час запису даних можливо додати коментар до випробування. Коментар вводиться до пусого поля та натискається кнопка [8].

Індикатори стану пристроїв – при непідключеному пристрої або помилці читання індикатор буде червоного кольору. Якщо навести курсор на індикатор можливо дізнатися подробиці несправності. Кожний пристрій опитується з певною частотою, якщо під час опитування данні не надійшли до програми, індикатор буде жовтого кольору. При правильній роботі пристрою, індикатор має зелений колір [8].

Запуск програми – можливо запустити програму комбінацією Ctrl+R.

Перемикач Коди/Значення – використовується для налаштування датчиків. Під час активного стану перемикача на екран будуть виводитися коди що надходять з датчика.

Зупинка програми – кнопка «Стоп» зупиняє запис та завершує виконання програми [8].

Час – поточна дата та час.

Папка – відкриває папку бази даних.

Розмір бази даних – відносний розмір бази даних від вільного місця на диску. При натисканні мишкою можливо побачити абсолютний розмір бази даних [8].

Запис бази даних – починає запис до бази.

Замір – назва заміру. При натисканні записується не лише час виміру, а й текст у цьому полі. Виміри записуються в каталозі бази даних у файлі типу 20151116T161115.xls. Сформувати їх можна, відкривши базу в програмі Read.exe. Зроблено це для швидкої зміни всіх вимірів в таблиці [8].

До підпрограми управління випробуваннями входять такі складові:

- модуль збору та збереження даних (дублює отримані дані та вносить їх то бази даних в режимі реального часу після запуску кнопки «запис бази даних»);
- модуль розрахунку (використовує параметри типу «Формула» та параметри у листі «Розрахункові» для розрахунку записаних до них формул, після цього оброблені дані поступають на підпрограму візуалізації);
- модуль передачі даних (програма не тільки збирає, зберігає та обробляє дані, а й може передавати їх на інші пристрої використовуючи тип індикатора в підпрограмі підготовки випробувань комірка «Індикатор» зображена в таблиці 1.4 [8].

1.3.5 Аналіз підпрограми пост-обробки та аналізу Ostopus

Програма Read.exe створена на основі середовища розробки і платформи для виконання програм – LabVIEW. Ця програма виконує функції пост-обробки та аналізу бази даних у програмному забезпеченні для проведення стендових випробувань (рис 1.20).

Функції підпрограми управління випробуваннями Ostopus [8]:

- формування звітів у вигляді таблиць та графіків за шаблонами;
- друк документів зі звітами;
- пост-обробка та аналіз баз даних випробувань;
- перерахунок сирих даних з бази за формулами та алгоритмами.

Далі зазначено докладний опис інтерфейсу підпрограми пост-обробки та аналізу програмного забезпечення Ostopus.

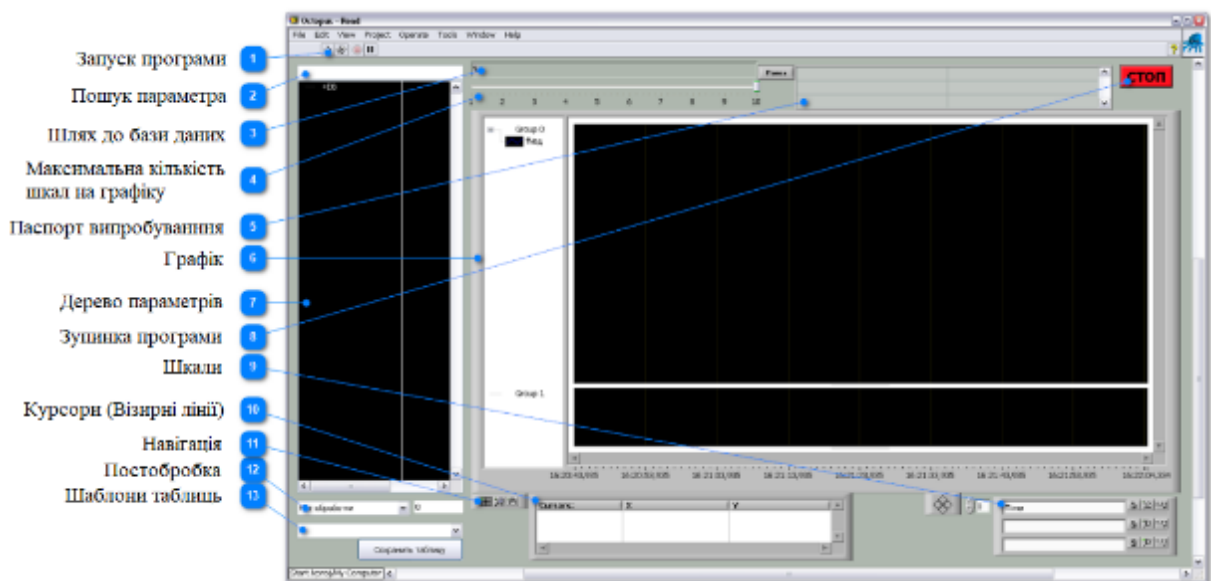


Рисунок 1.20 – Головне вікно підпрограми пост-обробки та аналізу Ostopus [8]

Запуск програми – можливо запустити програму комбінацією клавіш Ctrl+R [8].

Пошук параметра – виконує пошук параметра по введеним символам.

Шлях до бази даних – ця кнопка відкриває директорію з файлом бази даних, можливо використовувати повзунок для контролю глибини директорії.

Максимальна кількість шкал на графіку – при розміщенні шкал на графіку, вони будуть автоматично розміщатися на графіку в залежності від цього обмеження [8].

Паспорт випробування – заповнюється в листі «Паспорт» підпрограми підготовки випробувань. Перший стовпчик це назва, другий – значення.

Графік – на нього можливо мишкою перетягти параметри с дерева параметрів. На ньому відображаються коментарі та заміри. Підтримує функції зміни кольору фону.

Дерево параметрів – дерево параметрів, баз даних та шаблонів графіка. Тут можливо вибрати директорію з базою даних чи архів з базою.

Зупинка програми – зупиняє виконання програми, звільняє оперативну пам'ять [8].

Шкали – назва шкал графіка та їх атрибути. Такі як автомасштабування, формат відображення, точність, колір сітки. Атрибути кожного графіка можна знайти натиснувши ПКМ на піктограму графіка [8].

Курсори (Візорні лінії) – графік може вміщувати декілька візорних ліній, що називаються курсорами. На цьому полі їх можна додавати, змінювати атрибути чи видаляти .

Навігація – у цьому блоці знаходяться такі інструменти як «вказівник», «збільшити чи зменшити область», «наблизити чи віддалити», «рука».

Пост-обробка – перед виводом параметра на графік він може бути змінений за допомогою 11 типів пост-обробки. У полі поруч вказується інтервал в секундах чи необхідне значення для пост-обробки [8].

Шаблони таблиць – шаблони та «зберегти таблицю». Значення формуються по шаблону при читанні бази. Шаблони знаходяться у книзі Excel на листі «Таблиця». Тут можна вибрати, сформувати та зберегти таблицю – звіт.

До підпрограми пост-обробки та аналізу входять такі складові [8]:

- модуль розрахунку (використовує параметри типу «Формула» та параметри у листі «Розрахункові» для розрахунку записаних до них формул, після цього оброблені дані візуалізуються);
- модуль протоколювання зображено на рисунку 1.21 це кнопка «Зберегти таблицю». Відповідає за експорт даних та графіків у бажаний формат, та округлення даних з бази під час створення протоколів випробувань. Більш докладно його можна розглянути на;
- модуль візуалізації це область на якій виводиться графік, дерево параметрів. Відповідає за представлення даних з бази у вигляді графіку, використовуючи результати модуля розрахунку, для більш точного представлення [8].

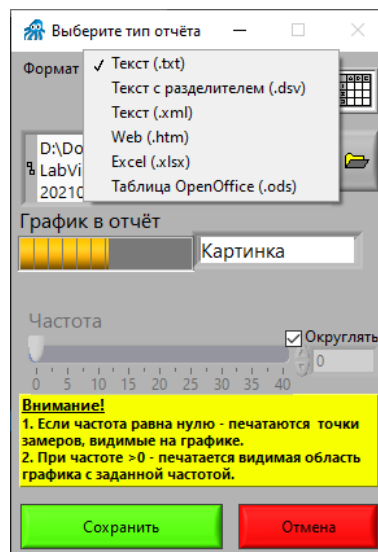


Рисунок 1.21 – Модуль протоколювання

1.3.6 Аналіз підпрограми візуалізації Octorus

Завдання візуалізації вимірюваних параметрів вирішує підпрограма візуалізації даних створена на основі середовища розробки і платформи для виконання програм – LabVIEW.

Підпрограма представляє собою файли екранів з розширенням «.vi». Екрани відкриваються цифровими клавішами з підпрограми управління випробуваннями – програма Start.exe. Під час роботи або запису усі екрани, окрім головного Start.exe, можливо вільно закривати (Alt+F4; зупинити CTRL+9) та відкривати (0-9). Файли з екранами зберігаються в директорії «Screens». Назва екрану повинна починатися з цифри як на рисунку 1.22, для того, щоб керуюча програма розуміла, які екрани потрібно відкрити під час спрацьовування цифрової клавіші [8].

Усі графіки за замовчуванням будуються за допомогою лінійної інтерполяції, проте можливо будувати їх іншим чином вказавши тип індикатора в комірці «Тип» підпрограми підготовки випробувань, що було наведено в таблиці 1.3.

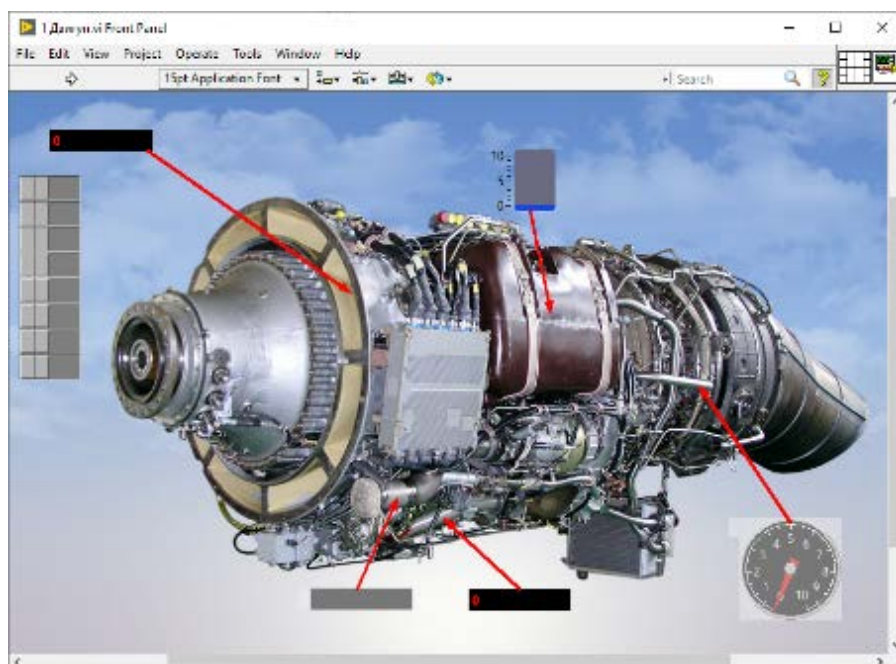


Рисунок 1.22 – Розроблений при дослідженні екран підпрограми візуалізації

Під час проведення експериментів з візуалізацією на програмі для проведення випробувань на стендових системах збору даних було помічено, що під час візуалізації параметри на графіках відображаються з гострими піками, що не відповідає реальному поведінню параметра. Зміна фізичних значень вимірюваних параметрів відбувається поступово, більш полого, експонентно. Це твердження ґрунтується на законі охолодження Ньютона [9], що каже: швидкість охолодження приблизно пропорційна різниці температур між нагрітим тілом та навколишнім середовищем. Також твердження ґрунтується на експериментально підтверджених даних датчиків тиску [10]. Для прикладу було використано базу даних записану під час досліджень, що була візуалізована за допомогою підпрограми пост-обробки та аналізу Ostopus на графіках, жовтими колами виділені місця у яких виникають похибки (рис. 1.23).

Також, під час дослідження було помічено, що деякі датчики можуть давати значну похибку вимірювань, що продемонстровано на базі даних, яка була записана при випробуваннях авіаційного двигуна АІ336 на 1 стенді (рис. 1.24). Прикладом є похибка датчику витрат палива (жовтий графік), зеленими овалами було відмічено місця похибок на графіку.

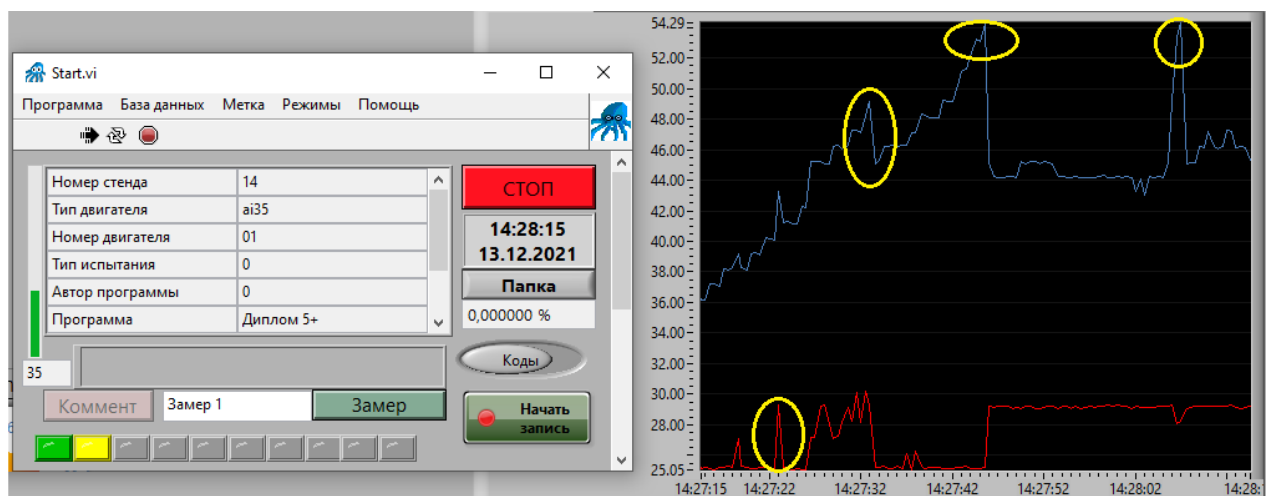


Рисунок 1.23 – Демонстрація неточності підпрограми візуалізації

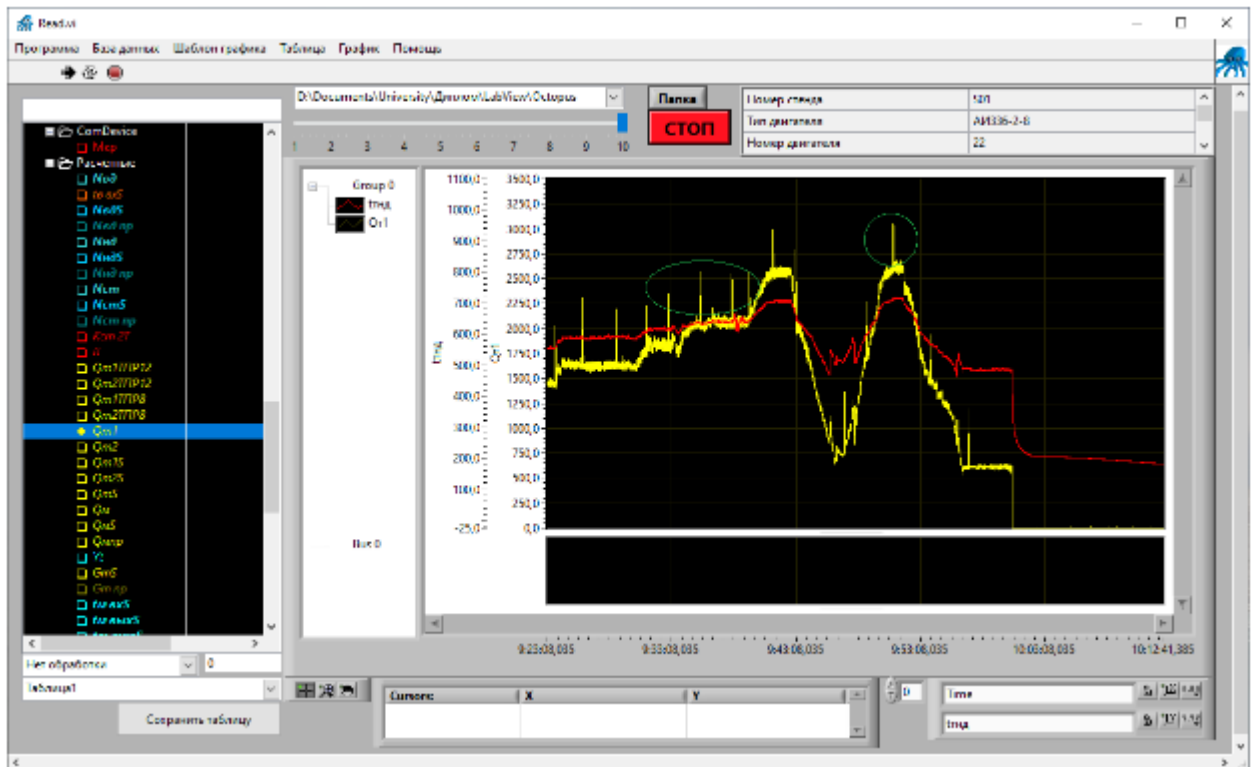


Рисунок 1.24 – Демонстрація похибок датчика витрат палива

У підпрограми пост-обробки та аналізу Ostorus є функції для усунення похибок на графіках, наприклад «усереднення», далі продемонстровано роботу цієї функції на рисунку 1.25 Було обрано усереднювати значення параметру по 10 точок.

Як можна побачити з графіку, похибки датчика витрат палива на графіку були усунені. Усунення похибок є важливою функцією, тому що збільшує швидкість аналізу графіку, підвищує точність візуалізації даних. Функції усереднення реалізовані у підпрограмі пост-обробки даних, проте в підпрограмі візуалізації такої функції не має.

До недоліків програмного забезпечення Ostorus, за результатами досліджень можна віднести наявність похибок при візуалізації даних випробування, підпрограма візуалізації не має вбудованих функцій усереднення, довга процедура додавання нових плат до проекту.

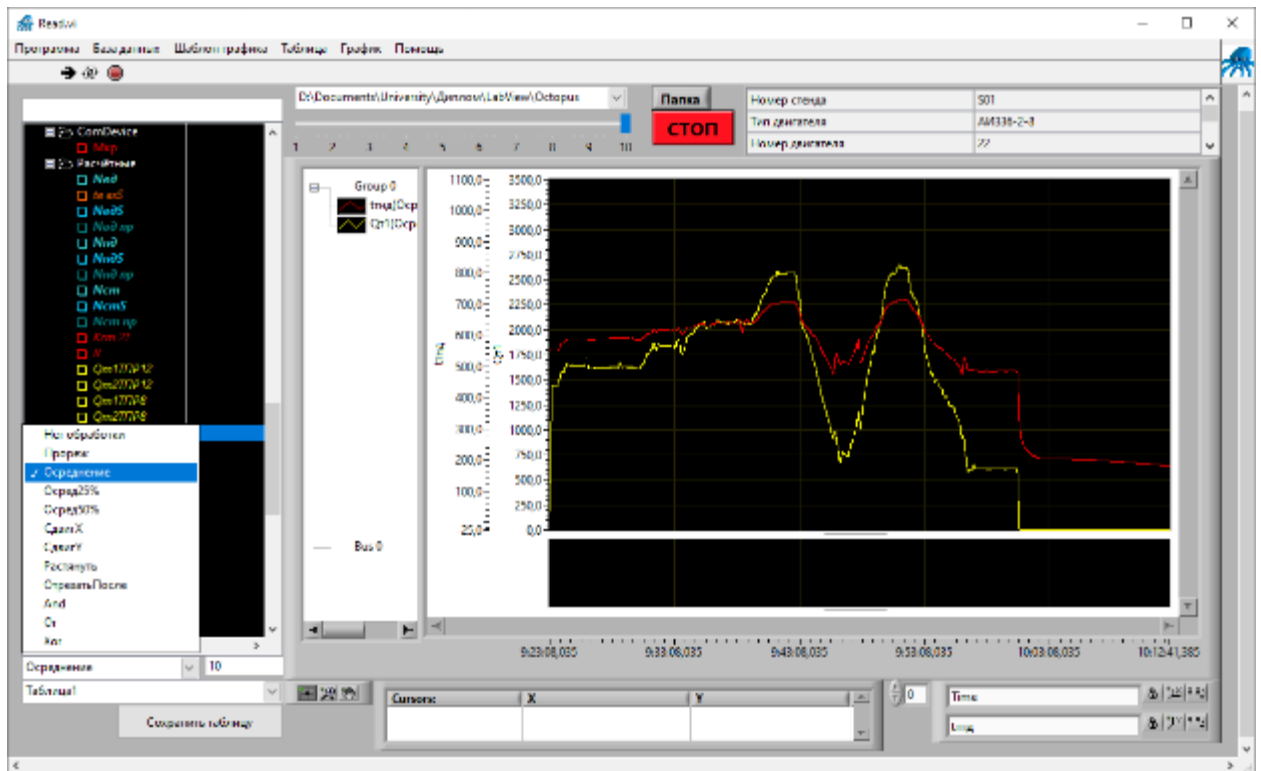


Рисунок 1.25 – Демонстрація пост-обробки бази даних

1.3.7 Висновок за дослідженням програмного забезпечення для проведення випробувань на стендових системах

На основі проведених досліджень складено порівняльну таблицю 1.5 щодо програмного забезпечення для проведення випробувань на стендових системах.

Таблиця 1.5 – Порівняння програмного забезпечення

Критерій порівняння	Octopus	Octavo	VERT
1	2	3	4
Взаємодія з Windows, Linux, Mac OS	+	Необхідна передача даних на інші стендові системи	Працює до версії Windows 98
Підтримка діалогових вікон з користувачем	+	+	+
Підтримка кнопок та перемикачів на екранах	+	+	—

Продовження таблиці 1.5

1	2	3	4
Відображення графіків, діаграм, та їх кастомізація	+	+	+
Індикація станів перемикачів	+	+	–
Протоколювання у різних форматах	Txt, xlsx, zip, rar, html, xml, ods, bmp, eps, emf	Txt, xml, tar, gz, bmp	Txt, rar, bmp, xls
Підтримка роботи мишкою	+	+	–
Можливість продовження випробувань після аварійної зупинки системи	Необхідно запустити нове випробування	+	Необхідно запустити нове випробування
Додатки для аналізу баз даних	+	+	+
Вбудований Help	+	–	–
Процедура додавання нових плат	Дослідити ТО плати, розробити VI для зчитування даних для нової плати, вбудувати VI до проєкту ПЗ	Відкрити базову конфігурацію, додати плату зі списку, конфігурувати плату за ТО, синхронізувати, зберегти	Створити нову dll бібліотеку для плати, підключити бібліотеку до файлу конфігурації, налаштувати плату за ТО
Моніторинг ресурсів системи	+	+	Тільки вільне місце в БД
Наявність багів	Зустрічаються	Є	Не виявив
Можливість вдосконалення програмної системи	+	Закритий програмний код	+
Зміна версії двигуна	Необхідно змінити назву підпрограми підготовки випробувань на необхідну у файлі конфігурації ПЗ, замінити назву підпрограми візуалізації на необхідну	Необхідно змінити активний файл продукту та файл базової конфігурації на такі самі файли для іншої версії двигуна, що змінені у однаковий час	Необхідна зміна архітектури, додаються блоки для зчитування нових параметрів, конфігуруються між собою, налаштовуються параметри та екрани

За результатами дослідження VERT є застарілою програмою, Octavo сучасна програма, проте закритий програмний код не дозволяє її

модифікувати, подальший розвиток та вдосконалення проаналізованого програмного забезпечення Octavo та VERT не є доцільним, у пріоритеті дослідження, розробка методів та засобів візуалізації вимірюваних параметрів для програми проведення стендових випробувань – Ostopus.

Передумовами для підвищення якості візуалізації вимірюваних параметрів програмою Ostopus є:

- наявність різких піків на графіку досліджуваного параметра. Це призводить до виникнення похибок при аналізі реального поведіння параметра, не дає повної та точної картини вимірюваних параметрів у будь-який момент часу;
- відсутність функції усереднення в підпрограмі візуалізації даних Ostopus. На графіках не має функції усереднення параметрів під час випробувань, тому виникають помилкові пікові значення на графіку. З цього випливає ризик неточної інтерпретації візуалізованих в реальному часі даних інженерами, що проводять випробування авіаційних двигунів, також це може призвести до помилкових рішень групи персоналу тестування.

1.4 Аналіз апаратних засобів збору даних

Для забезпечення збору даних на промисловому виробництві найчастіше використовуються датчики. В залежності від параметрів, що повинні відстежуватися використовуються відповідні датчики. Через специфіку роботи на випробуваннях авіаційних двигунів було проаналізовано наступні засоби збору даних: датчики АИР – 10Н, датчики МІДА.

1.4.1 Огляд перетворювачів тиску АИР – 10Н

Датчики АИР – 10Н використовуються в системах стендових випробувань ГАММА – 1101 та АІВС ПБ – 2, вони призначені для

неперервного перетворення значень абсолютного тиску, надлишкового тиску, надлишкового тиску – розрідження, різниці тиску та гідростатичного тиску рідких, газоподібних та агресивних середовищ до уніфікованого вихідного струмового сигналу 4 – 20 мА і у цифровий сигнал на базі Highway Addressable Remote Transducer (HART) протоколу. Зовнішній вигляд датчика зображено на рисунку 1.26.



Рисунок 1.26 – Датчик АИР – 10Н [9]

АИР – 10Н використовується в системах автоматизованого контролю, регулювання та управління технологічними процесами. У відповідності до ГОСТ 22520 датчики є [11]:

- по кількості вхідних перетворюваних сигналів – одноканальними;
- по кількості вихідних сигналів – двоканальними;
- по залежності вихідного сигналу від вхідного – з лінійною залежністю або з функцією обчислення квадратного кореня;
- по можливості перенастроювання діапазону – багатомежевими, перенастроюваними [11].

АИР – 10Н можуть підключатися до ПК за HART – протоколом для конфігурування, та отримання даних вимірювання в процесі експлуатації. Для

взаємодії датчика з ПК використовується програма HARTconfig. Процедура конфігурування включає в себе [11]:

- зміна значень верхніх та нижніх меж вимірів;
- вибір залежності вихідного сигналу від вхідного (лінійно – зростаюча, лінійно – спадна або функція обчислення квадратного кореня);
- вибір часу демпфірування;
- вибір одиниць вимірювання.

Значення похибок для датчиків зображено у таблиці 1.6.

Таблиця 1.6 – Межі допустимої похибки для АИР – 10Н [11]

Код класу	Межі допустимої основної приведеної похибки для верхніх меж діапазонів вимірювання у %							
Кгс/см ²	1	2	3	4	5	6	7	8
A01	0.1	0.1	0.15	0.2	0.25	0.3	0.4	0.5
B02	0.2	0.2	0.3	0.4	0.5	0.6	0.8	1.0
C05	0.5	0.5	0.5	0.8	1.0	1.2	1.5	2.0

АИР – 10Н складаються з первинного перетворювача та електричного пристрою. Середовище під тиском поступає до камери первинного перетворювача та деформує його мембрану, це призводить до зміни електричного опору розташованих на ній тензорезисторів, увімкнених до електричного кола дільника напруги, у результаті чого первинний перетворювач видає сингал напруги. Електронний пристрій перетворює електричний сигнал у цифровий код значення вимірюваного тиску, який перетворюється до уніфікованого токового вихідного сигналу, або сигналу на основі HART – протоколу [11].

Конструктивно АИР – 10Н складаються з металевих корпусів, в яких розміщені модулі електронних пристроїв. З однієї сторони корпусу

знаходиться штуцер з первинним перетворювачем, з іншого боку встановлено герметичну вилку, або герметичні кабельні виходи [11].

1.4.2 Огляд мікроелектронних датчиків тиску МІДА

Датчики МІДА використовуються в системах стендових випробувань ГАММА – 1101 та АІВС ПБ – 2 та призначені для промислових систем контролю та регулювання у тому числі в атомній енергетиці, рідинах та газах, що неагресивні до титанових сплавів [12].

Промислові, малогабаритні датчики тиску МІДА – 15 призначені для вимірювання абсолютного, надлишкового тиску, надлишкового тиску – розрідження, різниці тиску з аналоговими або цифровими вихідними сигналами. Сфери застосування [12]:

- системи контролю та управління технологічними процесами в гірничій сфері;
- в комунальному господарстві;
- в транспортній промисловості;
- в енергетиці.

Ступінь захищеності від впливу пилу та води відповідає стандартам IP64, IP65, IP68. За електромагнітною сумісністю датчики з вихідним сигналом 4-20 мА відповідають стандарту ГОСТ 30804.6.2 – 2013. Існують моделі що мають іскробезпечне електричне коло або вибухостійку оболонку що дозволяє використовувати їх у гірній промисловості [12].

МІДА цифровий датчик тиску – це датчик з вихідним сигналом в вигляді чисел, зручним інтерфейсом та підтримкою HART протоколу. Цифровий вихідний сигнал дозволяє підвищити точність вимірювань. Зовнішній вигляд датчика зображено на (рис. 1.27). Значення похибок для датчиків зображено у таблиці 1.7 [12].



Рисунок 1.27 – Датчик МІДА [13]

Таблиця 1.7 – Межі допустимої похибки для датчиків МІДА [12]

Назва датчику	Діапазон вимірювання (МПа)	Межі допустимої основної приведеної похибки (%)
МІДА – ДИ – 13П(М)	0 – 0,04; 0 – 0,1; 0 – 0,25; 0	0,25
	– 1; 0 – 2,5; 0 – 10;	0,5
	0 – 16; 0 – 25; 0 – 100...	1
МІДА – ДИ – 13ПК	0 – 0,04; 0 – 0,1; 0 – 0,25; 0	0,25
	– 1; 0 – 2,5; 0 – 10;	0,25
	0 – 16; 0 – 25; 0 – 100...	0,5
МІДА – ДИВ – 13П	(-0,02) – 0,02; (-0,05) – 0,05;	0,5
	(-0,1) – 0,15;	0,5
	(-0,1) – 0,3; (-0,1) – 0,5; (-	1
	0,1) – 1,5; (-0,1) – 2,4.	1
МІДА – ДИ – 12П – 05	0 – 2,5; 0 – 4; 0 – 6; 0 – 10; 0	0,5
	– 16; 0 – 25; 0 – 40;	0,5
	0 – 60; 0 – 100;	1

1.4.3 Відмінність датчиків АИР – 10Н від МІДА

АИР – 10Н – дводіапазонний датчик, МІДА – однодіапазонний. З конструкторської точки зору датчики майже однакові, існують лінійки для вимірювання тиску, температури. Датчики виконують перетворення аналогового сигналу 4-20 мА у цифровий у стандарту HART. Проте, АИР – 10Н вміщує мікропроцесор, він відрізняє дводіапазонний від однодіапазонного датчика. Завдяки мікропроцесору можливо переналаштовувати межі вимірювання датчику та виконувати інші налаштування описані в процедурі конфігурування АИР – 10Н. Недоліком дводіапазонного датчика є те, що точність вимірювання даних у ньому менша ніж у однодіапазонному, проте, значення похибки дуже мале, тому їм можна знехтувати.

1.5 Аналіз апаратних засобів аналого – цифрового перетворення

Аналого – цифрові перетворювачі (АЦП, ADC – «Analog – to – Digital Converter») та цифро – аналогові перетворювачі (ЦАП, DAC – «Digital – to – Analog Converter») використовуються для сполучення цифрових пристроїв та систем з аналоговими сигналами з реальним світом. При цьому АЦП перетворюють аналогові сигнали до дискретних (цифрових) сигналів, що поступають на цифрові прилади для подальшої обробки та збереження. ЦАП перетворюють вихідні цифрові сигнали цифрових приладів до аналогових сигналів [14].

АЦП та ЦАП використовуються у вимірювальній техніці (в цифрових осцилографах, вольтметрах, генераторах сигналів), у побутовій техніці, у комп'ютерній техніці (в пристроях вводу – виводу звуку, моніторах, принтерах), у медичній техніці, у радіоелектроніці, телефонах, та інших сферах діяльності суспільства. В якості АЦП та ЦАП звично використовують

спеціалізовані мікросхеми. На виробництві представлені у складі систем стендових випробувань ГАММА – 1101 (блок БЗПІ – 3) та АІВС ПБ – 2 (плата Advantech PCI 1747) [14].

1.5.1 Дослідження аналого – цифрових перетворювачів

Мікросхеми АЦП перетворюють вхідний аналоговий сигнал в послідовність цифрових кодів в загальному випадку мікросхему АЦП можливо представити у вигляді блоку, що має один аналоговий вхід, один або два входу для опорної (зразкової) напруги, та цифрові виходи для видачі коду, що відповідає значенню аналогового сигналу (рис. 1.28) [14].

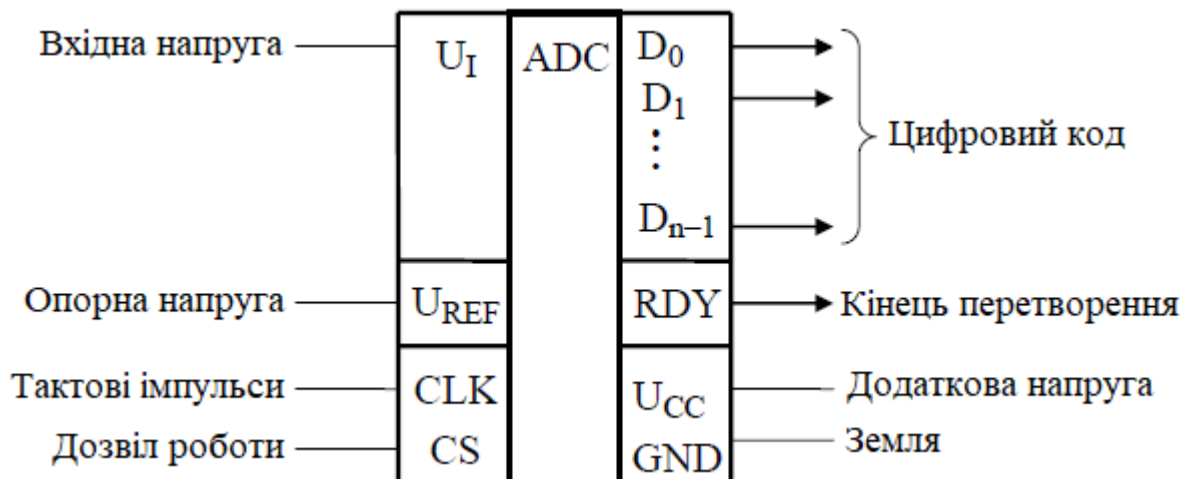


Рисунок 1.28 – Мікросхема АЦП [14]

Мікросхема АЦП також має вхід для подачі тактового сигналу (CLK – clock signal), сигналу дозволу роботи (CS – Consent Signal) та сигнал готовності вихідного цифрового коду (RDY – Ready). На мікросхему подається одна або дві живлячі напруги та спільний дріт [14].

Опорна напруга АЦП задає діапазон вхідної напруги, в якому виконується перетворення. Воно може бути постійним, або допускати зміни в деякому діапазоні [14].

У вихідному n – розрядному цифровому коді, N однозначно відповідає рівню вхідної напруги. Код може приймати 2^n значень від 0 до $(2^n - 1)$, тобто АЦП може розрізняти 2^n рівнів вхідної напруги. Кількість розрядів вихідного коду n являє собою важливу характеристику АЦП. В момент завершення перетворення аналого-цифрового сигналу видається сигнал завершення перетворення RDY, за допомогою якого зовнішній пристрій може читати код N . Керується робота АЦП тактовим сигналом CLK, який задає частоту видачі вихідних кодів. Сигнал CS дозволяє роботу мікросхеми. Найбільш розповсюджені принципи побудови АЦП: послідовний та паралельний [14].

1.5.2 Дослідження цифро – аналогових перетворювачів

В загальному вигляді мікросхему ЦАП можливо представити у вигляді блоку (рис. 1.29), що має декілька цифрових входів, один аналоговий вхід та аналоговий вихід. На цифрові входи ЦАП подається n – розрядний код N , на аналоговий вхід – опорна напруга, Вихідним сигналом являється струм I . При цьому вихідний струм пропорціонален вхідному коду та опорному струму. Для деяких мікросхем опорна напруга повинна мати фіксований рівень, для інших можливо змінювати значення в широкому діапазоні. ЦАП з великим діапазоном опорної напруги називається примножуючим ЦАП, тому що його легко використовувати для множення вхідного коду на будь яку опорну напругу [14].

Окрім інформаційних сигналів, мікросхеми ЦАП потребують підключення одного або двох джерел живлення та спільного дроту. Цифрові входи ЦАП забезпечують сумісність з стандартними виходами транзисторно-транзисторної логіки (ТТЛ) [14].

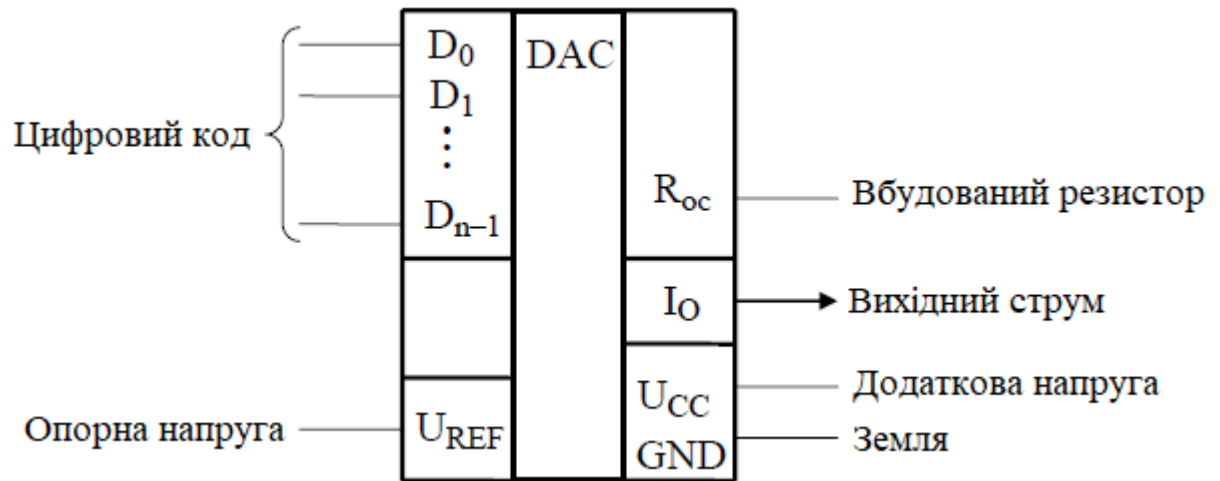


Рисунок 1.29 – Мікросхема ЦАП [14]

1.5.3 Аналіз блок збору та перетворення інформації

Блок збору та перетворення інформації БЗПІ – 3 (рис. 1.30) входить до складу бортових інформаційно – вимірювальних комплексів ГАММА – 1101 на випробувальних стендах ДП «Івченко – Прогрес», та призначений для комутації та перетворення у 10 – розрядний паралельний двійковий код напруги, що може змінюватися в межах -3 до 141 мВ, та опору в межах від -7 до 235,44 Ом. На стендах реального часу виступає пристроєм вводу-виводу даних в складі АІВС збудованих на базі інформаційно – вимірювального комплексу ГАММА – 1101, що описаний в пункті 1.5.1 дипломної кваліфікаційної роботи [15].

Можливі умови експлуатації БЗПІ – 3 [15]:

- температура довкілля -60 – 60°C;
- відносна вологість повітря до 98% при 35°C;
- атмосферний тиск від 5 до 800 мм рт. ст.

БЗПІ – 3 був створений з високим рівнем захисту від вібрацій та прискорень (до 12g), та високою точністю вимірювань. Приведена похибка вимірювання не більше за 0.5%. Блок може передавати інформацію по телеметричним каналам зв'язку, що дозволяє проводити вимірювання навіть

під час льотних випробувань літаків. Кількість інформаційних каналів 32, що розбиті на 8 груп. Кожна група каналів може перетворювати сигнали в будь-якому із зазначених в таблиці 1.8 діапазонів. Розподілення груп каналів по діапазонам забезпечується відповідними заглушками на блоці. Максимальна частота опитування кожного каналу 8 Гц. В БЗПІ – 3 проводиться послідовне опитування всіх 32 каналів [15].



Рисунок 1.30 – Блок БЗПІ – 3

Таблиця 1.8 – Розподіл каналів по діапазонам [15]

Номер діапазону	Нижня межа діапазону	Верхня межа діапазону
1	-3 мВ	141 мВ
2	-3 мВ	69 мВ
3	-3 мВ	33 мВ
4	-3 мВ	15 мВ
5	67,44 Ом	235,44 Ом
6	67,44 Ом	151,44 Ом
7	33,94 Ом	75,94 Ом
8	7,01 Ом	28,01 Ом

Читання вихідних кодів з БЗПІ – 3 здійснюється КТСС, що поступає до перетворювача чотирьох розрядним паралельним кодом. Рівень, що відповідає логічній «1» від 2,4 до 4,5 В. Рівень, що відповідає логічному «0» від 0 до 0,4 В. Найбільша частота імпульсів, що можуть опитувати БЗПІ – 3 складає 2048 Гц. Адрес блоку задається за допомогою перемикачів № МПІ (многоканальний перетворювач інформації) та № БПІ (блок перетворення інформації). Блок живиться від джерел електричної енергії, що відповідають ГОСТ 19705-81, а саме напруга постійного струму 24 – 29,4 В. Про коректну роботу сигналізує світлодіод на лицьовій панелі блоку [15].

БЗПІ – 3 складається з наступних функціональних вузлів [15]:

- комутатор вхідних сигналів;
- нормалізатор;
- аналого-цифровий перетворювач;
- пристрій керування;
- стабілізатор;
- фільтр вихідної напруги стабілізатора.

Структурна схема БЗПІ – 3 зображена на рисунку 1.31.

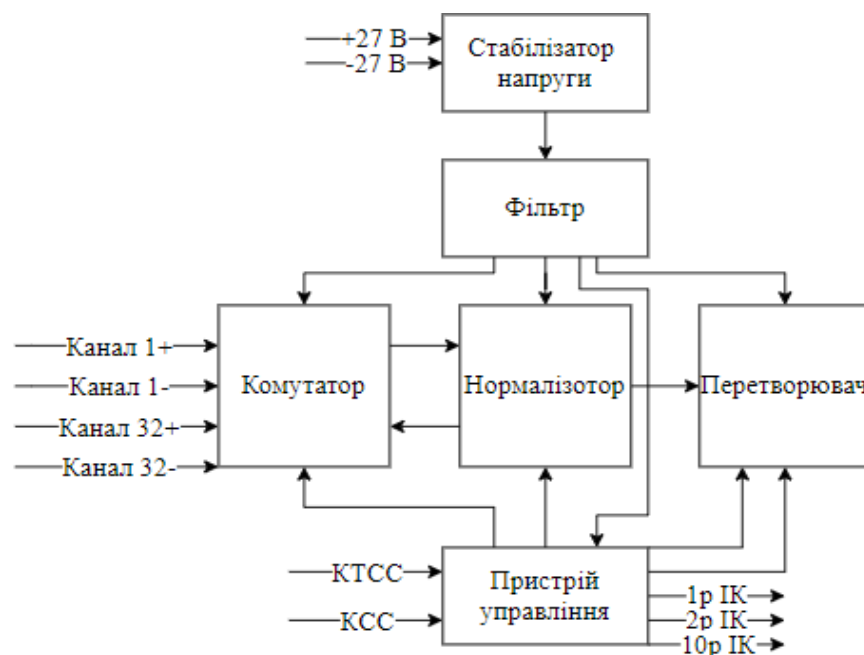


Рисунок 1.31 – Структурна схема БЗПІ – 3 [15]

Комутатор забезпечує послідовне підключення напруг до входу нормалізатора, та підключення генераторів стабільного струму до первинних перетворювачів опору [15].

Нормалізатор створює 2 стабільних струму для живлення перетворювачів опору та забезпечує підсилення вхідних сигналів з коефіцієнтом підсилення.

АЦП перетворює вихідний сигнал нормалізатора в 10 розрядний двійковий код.

Пристрій керування забезпечує [15]:

- формування імпульсу «Опитування» в залежності від положення перемикачів № МПІ та № БПІ;
- зчитування інформаційного коду з БЗПІ – 3;
- створення сигналів керування нормалізатором та АЦП в залежності від встановленої заглушки;
- створення сигналів керування комутатором [15].

Стабілізатор перетворює вхідну напругу в систему стабілізованих напруг для живлення плат блоку.

Фільтр згладжує високочастотні складові пульсації вихідних напруг імпульсного стабілізатора [15].

БЗПІ – 3 працює наступним чином. Кодовий селектор на платі керування формує з послідовності КТСС імпульс «Опитування», що зчитує код з вихідного регістру блоку. В момент, коли АЦП закінчує інтеграцію вихідного сигналу нормалізатора, в пристрої керування аналізується стан вихідного регістру. Якщо із регістра код був зчитаний у магістраль системи, то комутатор підключає до входу нормалізатора наступний робочий канал. Якщо код з регістру не зчитаний, то до входу нормалізатора підключається контрольний канал. Якщо до входу нормалізатора підключений робочий канал, то результат перетворення АЦП записується до буферного лічильника в пристрої керування. Після закінчення перетворення код з буферного

лічильника переписується до вихідного регістру пристрою керування з відки він зчитується в магістраль комплексу [15].

Таким чином в БЗПІ – 3 виконується послідовне опитування робочих каналів, а у проміжках між ними запускаються контрольні канали та виконуються операції контролю та корекції блоку [15].

1.5.4 Аналіз плат Advantech 1747U

На випробувальних стендах ДП «Івченко – Прогрес» використовується плати Advantech (рис. 1.32) для зчитування різноманітних параметрів тиску, вібрацій, витрат палива, частот та температур. Плата Advantech PCI – 1747U входить до складу інформаційно – вимірювальної системи АІВС ПБ – 2.



Рисунок 1.32 – Плата промислова Advantech PCI – 1747U [16]

Advantech PCI – 1747U плата аналогового входу з високою роздільною здатністю з великою кількістю каналів для шини Peripheral component

interconnect (PCI). Частота дискретизації до 250 кілосемплів/с, а 16-розрядна роздільна здатність забезпечує потужність необхідну для більшості програм збору даних. PCI – 1747U забезпечує 64 несиметрично сигналізованих або 32 диференціальних аналогових входних каналів або їх комбінації. Також має вбудований буфер First In First Out (FIFO) зразка 1k для аналогових входних даних [16].

Advantech PCI – 1747U надає користувачам функції вимірювання та контролю приведені нижче [16]:

- 16-розрядна висока роздільна здатність;
- частота дискретизації 250 кС/с;
- функція автоматичного калібрування;
- 64 несиметрично сигналізованих або 32 диференціальних каналів введення або їх комбінації;
- уніполярний та біполярний діапазон введення;
- FIFO зразка 1k;
- універсальна шина PCI (підтримує сигнал шини PCI 3,3 В або 5 В);
- BoardID.

PCI – 1747U забезпечує функцію автоматичного калібрування з утилітою для калібрування. Вбудована схема калібрування PCI – 1747U коригує коефіцієнт посилення і компенсує помилки в аналогових входних каналах, тим самим усуваючи необхідність додаткового обладнання та налаштувань користувача [16].

Плата має вбудований цифровий IP-перемикач, який допомагає визначити для кожної плати унікальний ідентифікатор, коли на комп'ютер встановлено декілька карт PCI – 1747U. Функція налаштування Board ID дуже корисна при створенні системи з декількома платами PCI-1747U. З правильним Board ID, ви можете легко ідентифікувати та отримувати доступ до кожної картки під час обладнання, конфігурування та програмування плати [16].

PCI – 1747U це пристрій Plug – and – Play, який повністю відповідає стандартам PCI специфікації Rev 2.2. Під час встановлення картки немає необхідності встановлювати перемички або цифрові IP – перемикачі. Натомість усі пов'язані з шиною конфігурації, такі як базова адреса вводу-виводу та переривання автоматично виконуються функцією Plug – and – Play [16].

Плата підтримує PCI – шину, що опановує direct memory access (DMA), для високошвидкісної передачі даних без розривів. PCI – 1747U виконує передачу даних по шині без втручання процесора, дозволяючи процесору вільно виконувати більш термінові завдання, такі як аналіз даних та обробка графіків. Функція дозволяє запускати всі функції вводу -виводу одночасно на повній швидкості без втрати даних [16].

Плата надає 1k семпли до FIFO буферу пам'яті для аналогових пристроїв. Це важлива функція для швидшої передачі даних та більш високої продуктивності в системі Windows [16].

PCI-1747U надає програмований лічильник таймера для генерації пускових механізмів для аналого – цифрового перетворення. Мікросхема таймера/лічильника 82C54, що містить три 16 – розрядних лічильника на частоті 10 МГц. Один лічильник використовується для підрахунку подій, що надходять із вхідного каналу. Два інші каскадно з'єднані разом, щоб створити 32-розрядний таймер [16].

Для програмування плати можливо використовувати [16]:

- Visual C++;
- Visual Basic;
- Delphi;
- C++ Builder.

Основні експлуатаційні відомості зазначені в таблиці 1.9.

Таблиця 1.9 – Експлуатація PCI – 1747U [17]

Критерій	Значення
Тип роз'єму вводу -виводу	68 пінів типу «мати»
Розміри	175 x 100 мм
Споживання енергії	Звично: +5 V 850 мА; +12 V 600 мА
	Максимально: +5 V 1 А
Температури	Експлуатація: від 0 до +60°
	Зберігання: від -20 до +85°С
Відносна вологість	5~95% без конденсату
Сертифікація	Маркування CE

Компоненти плати та їх взаємодія зазначені на рисунку 1.33.

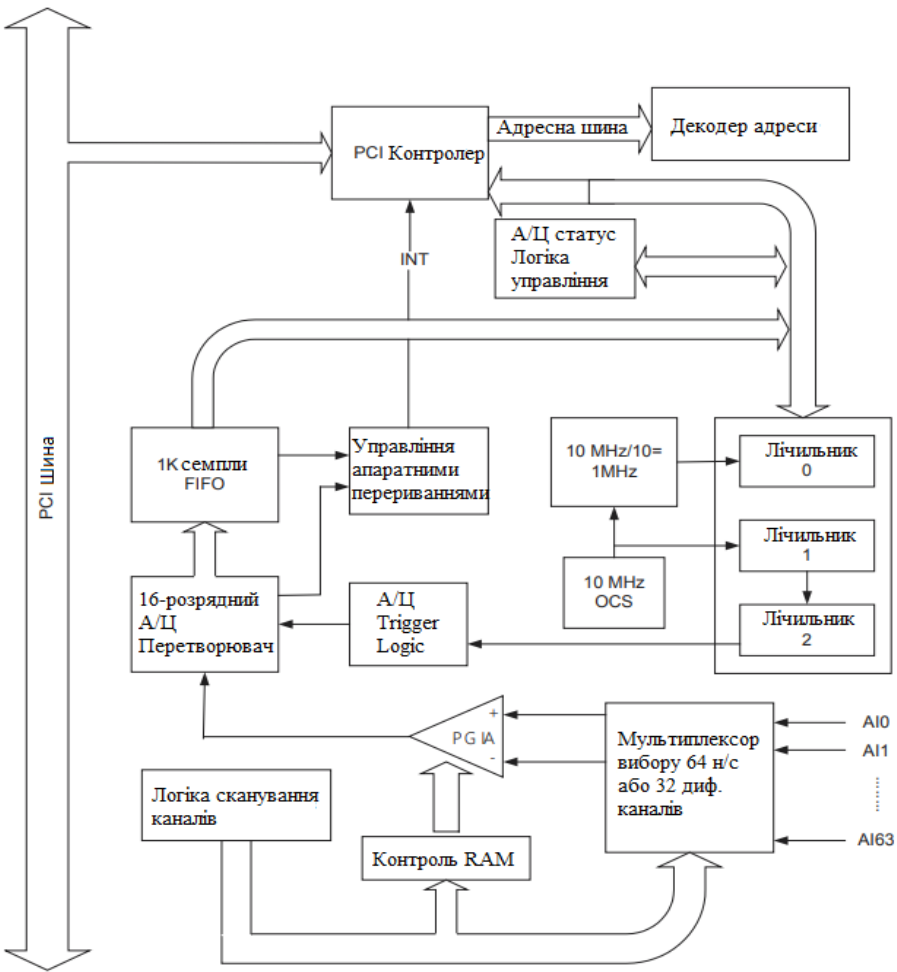


Рисунок 1.33 – Блок-діаграма PCI – 1747U [17].

Trigger Logic – являє собою метод, який дозволяє користувачеві переглядати і вибирати будь-які опції режиму роботи з метою підготовки датчика для виконання конкретного завдання. Режим Trigger Logic активується шляхом установки батареї. В активному режимі використовується певна послідовність відхилень щупа (спрацьовувань), яка дозволяє користувачеві вибирати серед серії доступних варіантів потрібні параметри режиму роботи [17].

1.6 Постановка задач дипломної кваліфікаційної роботи

Під час дослідження були проаналізовані промислові методи збору даних, архітектура стендів реального часу на ДП «ІВЧЕНКО – ПРОГРЕС», системи стендових випробувань ГАММА – 1101 та АІВС ПБ – 2, програмне забезпечення для керування стендовими випробуваннями Octavo, VERT, Octopus, апаратні засоби збору даних датчики АІР – 10Н та МІДА, засоби аналого-цифрового перетворення блок БЗПІ – 3 та плата Advantech PCI.

На основі проведених досліджень, можна зробити висновок, що АІВС ПБ – 2 є сучасною системою стендових випробувань, що може виконувати усі завдання по випробуванню авіаційних двигунів. Проте, програмна оболонка Octavo не досконала, тому на цей час ведеться розробка нового програмного забезпечення Octopus.

Мета роботи полягає у дослідженні та програмній реалізації методів та засобів візуалізації для підвищення якості представлення вимірюваних параметрів на системах стендових випробувань авіаційних двигунів.

Для досягнення мети необхідно розв'язати такі завдання:

- зібрати набори даних для дослідження, дослідити бази даних випробувань авіаційних двигунів;
- розробити інструмент для тестування методів візуалізації даних, протестувати методи, обрати відповідні для рішення поставленої задачі;

- визначити вимоги до ПЗ, розробити його архітектуру та вибрати середовище програмування;
- розробити тестове програмне забезпечення для проведення стендових випробувань, реалізувавши в ньому досліджені методи.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ВІЗУАЛІЗАЦІЇ ДАНИХ

В цьому розділі у відповідності до мети роботи та задач виконано аналіз даних на виробництві, обрано методи візуалізації даних, виконаний аналіз методів візуалізації даних, проведено розробку програмного забезпечення для тестування методів для візуалізації на декількох наборах даних.

2.1 Аналіз даних випробувань авіаційних двигунів на підприємстві

Для того, щоб розуміти які дані необхідно візуалізувати, потрібно провести аналіз даних на виробництві.

При випробуванні, інформаційно – вимірювальні системи збирають великі об'єми даних. Наприклад, під час циклічних випробувань за 3 години роботи на одному стенді збирається 30 Гб даних, це обумовлено великою кількістю вимірюваних параметрів, що інколи доходить до 1500 штук та більше на один двигун. В середньому, один двигун проходить 300 годин циклічних випробувань. За рік, 15-20 двигунів проходять циклічні випробування. За приблизними підрахунками за рік це 150 Тб інформації про роботу двигунів.

Інформація з випробувальних стендів надходить на інформаційно вимірювальний комплекс з різних джерел та у різних форматах. Наприклад, дані з стійки ГРТ передаються через COM порт використовуючи HART протокол передачі даних, дані з SLX передаються за допомогою протоколу MODBUS, дані про разові команди можуть надходити на систему реєстрації через мережу Ethernet.

Дані на інформаційно вимірювальну систему надходять по високошвидкісній потоковій передачі, швидкість передачі варіюється від 2 до

10 Гбит/с, забезпечення обчислювальних ресурсів для цих об'ємів даних та достатній об'єм місця для збереження виділяються по мірі необхідності.

Дані випробувань являються цінними для виробництва, вони постійно використовуються та аналізуються з використанням пакетної та паралельної обробки даних.

Враховуючи методику віднесення ІТ проєктів до Big Data (рис. 2.1) можливо зробити висновок, що на ДП «Івченко – Прогрес» робота ведеться саме з Big Data, тому для роботи з цими даними необхідно використовувати методи аналітики великих даних.

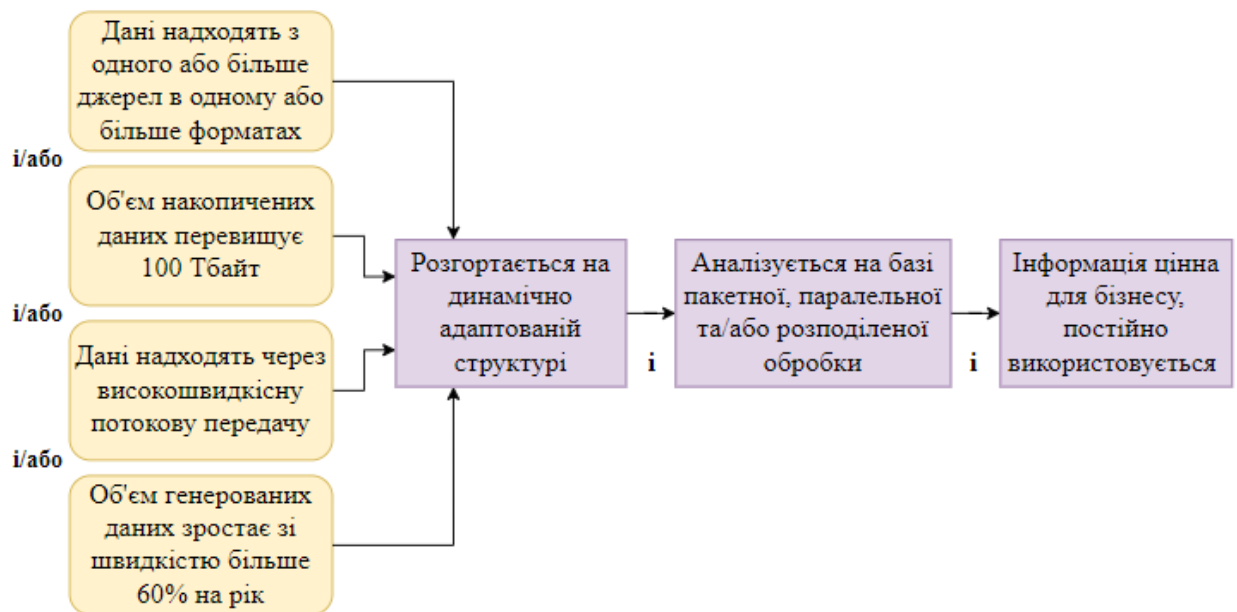


Рисунок 2.1 – Методика віднесення ІТ проєктів до Big Data [18]

Для дослідження великих даних використовують методи аналітики великих даних Big Data Analytics, що поділяються на такі групи (рис. 2.2).

Для дослідження результатів випробувань авіаційних двигунів найчастіше використовується «візуалізація даних» як метод аналізу даних. Візуалізація – методи графічного подання результатів аналізу великих даних у вигляді діаграм або анімації для спрощення інтерпретації, полегшення розуміння отриманих результатів.

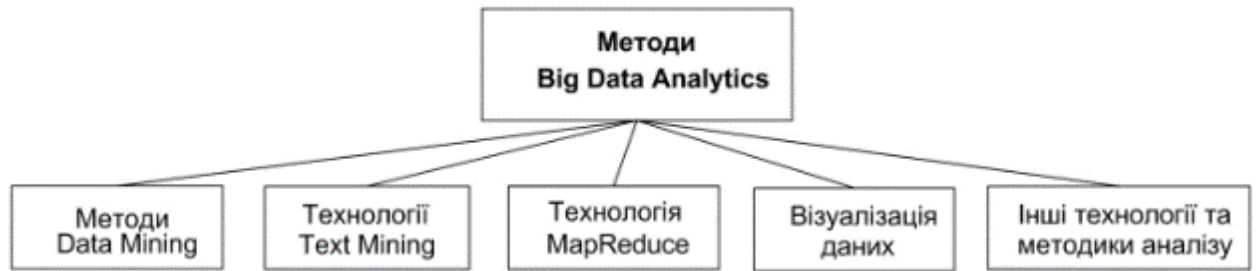


Рисунок 2.2 – Групи методів аналітики Big Data [19]

Візуалізація аналітичних даних – зображення інформації у вигляді рисунків, графіків, схем і діаграм з використанням інтерактивних можливостей та анімації для результатів, а також вихідних даних для подальшого аналізу. Наочне представлення результатів аналізу Великих даних має принципове значення для їхньої інтерпретації. Сприйняття людини обмежене, і вчені продовжують вести дослідження у галузі вдосконалення сучасних методів подання даних у вигляді зображень, діаграм або анімацій [19].

Під час досліджень двигунів, необхідно мати уявлення про поведінку функції вимірюваного параметра та її значення у кожній точці, що входить до діапазону цієї функції на основі дискретного набору точок. Ці дані необхідні для більш точної візуалізації вимірюваних параметрів на графіках та розрахунку значень вимірюваних параметрів у будь який момент часу.

Для рішення цього завдання використовують методи апроксимації.

2.2 Огляд особливостей апроксимації

Апроксимація – науковий метод, що являє собою наближене вираження будь-яких величин (геометричних чи математичних об’єктів) через інші, більш відомі або більш простіші величини, наприклад, кривих ліній – ламаними, ірраціональних чисел – раціональними, неперервних функцій – многочленами. Апроксимація дозволяє досліджувати числові характеристики

та якісні характеристики об'єкта при наявності деякого набору відомих даних [20].

Чим простіша апроксимація функції, тим менше часу необхідно для рішення задачі апроксимації. Чим більша кількість вузлів, тим менша похибка. Для кожної конкретної апроксимованої функції потрібно вибрати такий спосіб апроксимації, який забезпечить мінімальну похибку при мінімальній кількості вузлів [20].

Існують два принципово різні типи апроксимації функції [20]:

- інтерполяція – апроксимуюча функція $F(x)$ точно співпадає з табличними значеннями $y_0, y_1, \dots, y_n$ функції $f(x)$;
- метод найменших квадратів – апроксимуюча функція $F(x)$ може не співпадати з жодним табличним значенням $y_0, y_1, \dots, y_n$ функції $f(x)$, максимально наближуючись до них в середньому.

Інтерполяція – це спосіб знаходження проміжних значень параметру по відомому дискретному набору значень [20].

Наприклад, на виробництві для налаштування датчиків АИР – 10Н та МІДА використовуються дані напруги, що видає датчик та відповідні їм значення вимірюваного параметру, які в загальному вигляді відображено в таблиці. 2.1.

Таблиця 2.1 – Приклад опису функції перетворення напруги до тиску

Параметр	Значення				
Напруга x	x_0	x_1	x_2	...	x_n
Тиск $f(x)$	y_0	y_1	y_2	...	y_n

Значення аргументу x_i називають вузлами. Для рішення завдання аналізу та візуалізації даних необхідно знайти наближені значення функції $f(x)$ в будь якій довільній точці відрізка $[x_0; x_n]$ [20].

Для рішення цієї задачі необхідно замінити невідому функцію $f(x)$ іншою безперервною на відрізку $[x_0; x_n]$ функцією $F(x)$, значення якої приблизно дорівнюють значенням функції $f(x)$ у будь-якій точці відрізка $[x_0; x_n]$, тобто $F(x) \approx f(x); x \in [x_0; x_n]$. Наближення функції $f(x)$, що задана в табличному вигляді, іншою неперервною функцією $F(x)$ називається апроксимацією [20].

Значення $y_0, y_1, \dots, y_n$, що приведені в (табл. 2.1), отримані експериментально з високою точністю, тому можливо створити наближаючу функцію $F(x)$, що проходить через точки з координатами x_i, y_i ($i = 0, 1, \dots, n$), тобто значення наближаючої функції $F(x)$ повинно точно співпадати з табличними значеннями даної функції $f(x)$: $F(x) = f(x_n) = y_n$. У цьому випадку пошук наближаючої функції $F(x)$ називають інтерполяцією [20].

Завданням інтерполяції вважається пошук приближених значень табличної функції при аргументах x , що не співпадають з вузловими. Якщо x знаходиться всередині інтервалу $[x_0; x_n]$, то процес пошуку наближеного значення функції $f(x)$ називають інтерполяцією, якщо x знаходиться поза інтервалом $[x_0; x_n]$, то цей процес називається екстраполяцією [20].

Інтерполяція поділяється на такі види [20]:

- лінійна інтерполяція;
- інтерполяція поліномом Лагранжа;
- інтерполяція сплайнами;
- інтерполяція поліномом Ерміта.

Далі наведений аналіз існуючих методів інтерполяції та обчислювальні експерименти на наборі експериментальних даних реалізовані з використанням вбудованих функцій LabVIEW для дослідження їх особливостей.

2.2.1 Лінійна інтерполяція

При лінійній інтерполяції табличні значення функції у суміжних вузлових точках з'єднуються відрізками прямих, та функція $f(x)$ наближається за допомогою ламаної з вершинами в даних точках. Рівняння кожного відрізка ламаної в загальному випадку різні. Через те, що мається n інтервалів $[x_0; x_n]$, для кожного з них в якості рівняння інтерполяційного многочлена використовується рівняння прямої, що проходить через 2 точки (x_{i-1}, y_{i-1}) та (x_i, y_i) , можемо записати рівняння прямої для цих точок у вигляді формули (2.1) [20]:

$$\frac{y - y_{i-1}}{y_i - y_{i-1}} = \frac{x - x_{i-1}}{x_i - x_{i-1}}. \quad (2.1)$$

Звідки отримуємо формулу (2.2) [20]:

$$y = a_j x + b_j, x_{j-1} \leq x \leq x_j. \quad (2.2)$$

Далі знаходимо коефіцієнти прямої за формулами (2.3) та (2.4) [20]:

$$a_j = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad (2.3)$$

$$b_j = y_{i-1} - a_j x_{i-1}. \quad (2.4)$$

При використанні лінійної інтерполяції спочатку потрібно визначити інтервал в якому знаходиться значення аргументу x , потім підставити його до формули (2.5) [20]:

$$y = a_j x + b_j, x_{i-1} \leq x \leq x_i, \quad (2.5)$$

звідки знайти приближене значення функції у цій точці. Якщо значення x виходить за межі інтервалу $[x_0; x_n]$, то виконується лінійна екстраполяція по відрізкам прямих, що примикають до кінцевих точок [20].

Лінійна інтерполяція була практично реалізована на наборі даних для обчислювального експерименту зображеного в таблиці 2.2 у середовищі розробки LabVIEW результат зображено на рисунку 2.3. Дані були взяті з бази даних випробувального стенда з параметру Рш (тиск повітря на шайбі відбору), зчитування даних проводилося з датчика тиску АІР – 10Н.

Таблиця 2.2 – Набір даних для обчислювального експерименту

Параметр	Значення							
Час (с)	1	2	3	4	5	6	7	8
Значення тиску повітря (кг/см ²)	2,6	4,3	0,4	3,5	6,1	7,3	4,7	5,5

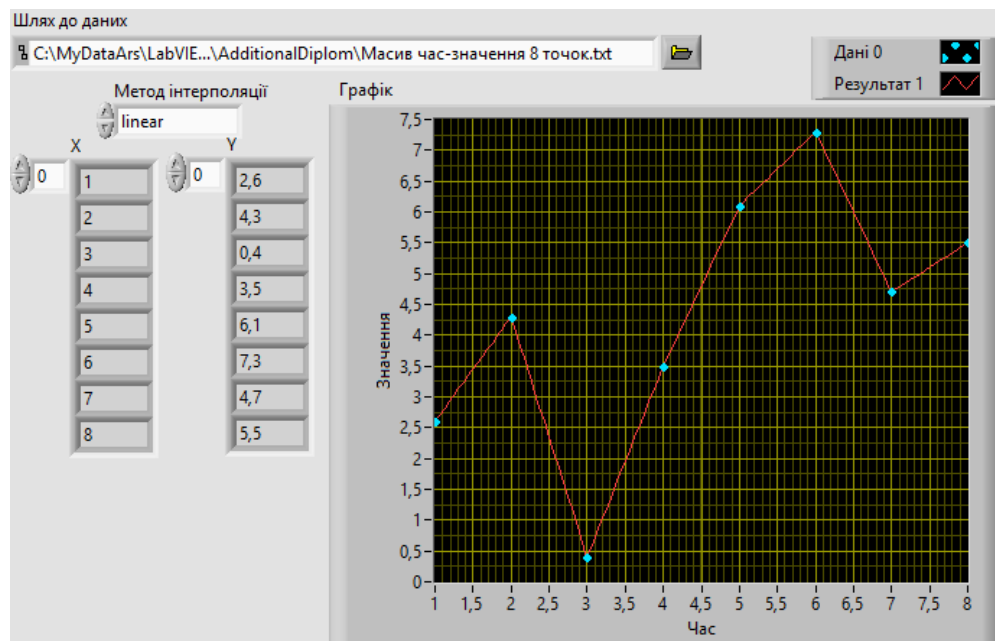


Рисунок 2.3 – Результат обчислювального експерименту з лінійної інтерполяції

Отже, це найпростіший та поширений спосіб інтерполяції. При лінійній інтерполяції інтерполуюча функція має злами у вузлах інтерполяції та розриви значень похідних. Похибка інтерполяції визначається відстанями між вузлами інтерполяції та зламами у вузлах інтерполяції [20].

2.2.2 Інтерполяція поліномом Лагранжа

Похибка лінійної інтерполяції обумовлена тим, що графік інтерполуючої функції має злами у вузлах інтерполяції. Ці злами можна прибрати, якщо в якості інтерполуючої використати таку функцію, графік якої представляє собою плавну криву (наприклад, поліном), що проходить через задані в (табл. 2.2) точки. Інтерполяція поліномом Лагранжа надає високу точність, якщо значення функцій в сумісних вузлах змінюються досить повільно. Можна побудувати інтерполяційний поліном Лагранжа $L_n(x)$, для якого виконується умова $F(x_n) = f(x_n) = y_n$ за формулою (2.6) [21]:

$$L_n(x) = l_0(x) + l_1(x) + \dots + l_n(x) = \sum_{i=0}^n l_i(x), \quad (2.6)$$

де $l_i(x)$ – поліном ступіню n ;

i – номер полінома та $i = 0, 1, \dots, n$ [21].

Також необхідна умова $x = x_j$, де $(j = 0, 1, \dots, n)$ – номер вузла, поліном $l_i(x)$ розраховуємо за формулою (2.7) [21]:

$$l_i(x_j) = \begin{cases} y_i, & \text{якщо } j = i \\ 0, & \text{якщо } j \neq i \end{cases}. \quad (2.7)$$

Після цього поліном $l_i(x)$ складаємо за формулою (2.8) [21]:

$$l_i(x) = C_i(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n). \quad (2.8)$$

У поліномі кожний з n множників у дужках є різницею між неперервним аргументом x та дискретним значенням x_j , вузла з номером j . Множник $x - x_i$ відсутній. За рахунок цього $l_i(x) = 0$ в кожному вузлі, окрім вузла, номер якого j співпадає з номером полінома i . Таким чином, виконується умова при $j \neq i$. Знаходимо коефіцієнт C_i з наступної умови $l_i(x_i) = y_i$ за формулою (2.9) [21]:

$$C_i = \frac{y_i}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}. \quad (2.9)$$

Підставивши значення C_i в поліном у формулу (2.10) [21]:

$$l_i(x) = C_i(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n), \quad (2.10)$$

отримаємо після скорочення інтерполяційний многочлен Лагранжа [21]

Інтерполяція методом Лагранжа була практично реалізована на наборі даних для обчислювального експерименту зображеного в таблиці 2.2 у середовищі розробки LabVIEW результат зображено на рисунку 2.4.

Отже, інтерполяція поліномом Лагранжа надає високу точність, якщо значення функцій в сумісних вузлах змінюються досить повільно, найчастіше використовується в теоретичних дослідженнях (при доказі теорем, аналітичному розв'язку задач). Якщо функція, що інтерполується, не має безперервної похідної порядку n , то поліном Лагранжа для неї по n вузлах може виявитися дуже поганим апроксимантом. Для отримання апроксимантів вищої якості, необхідно використовувати інші інтерполятори у складі сплайнів [21].

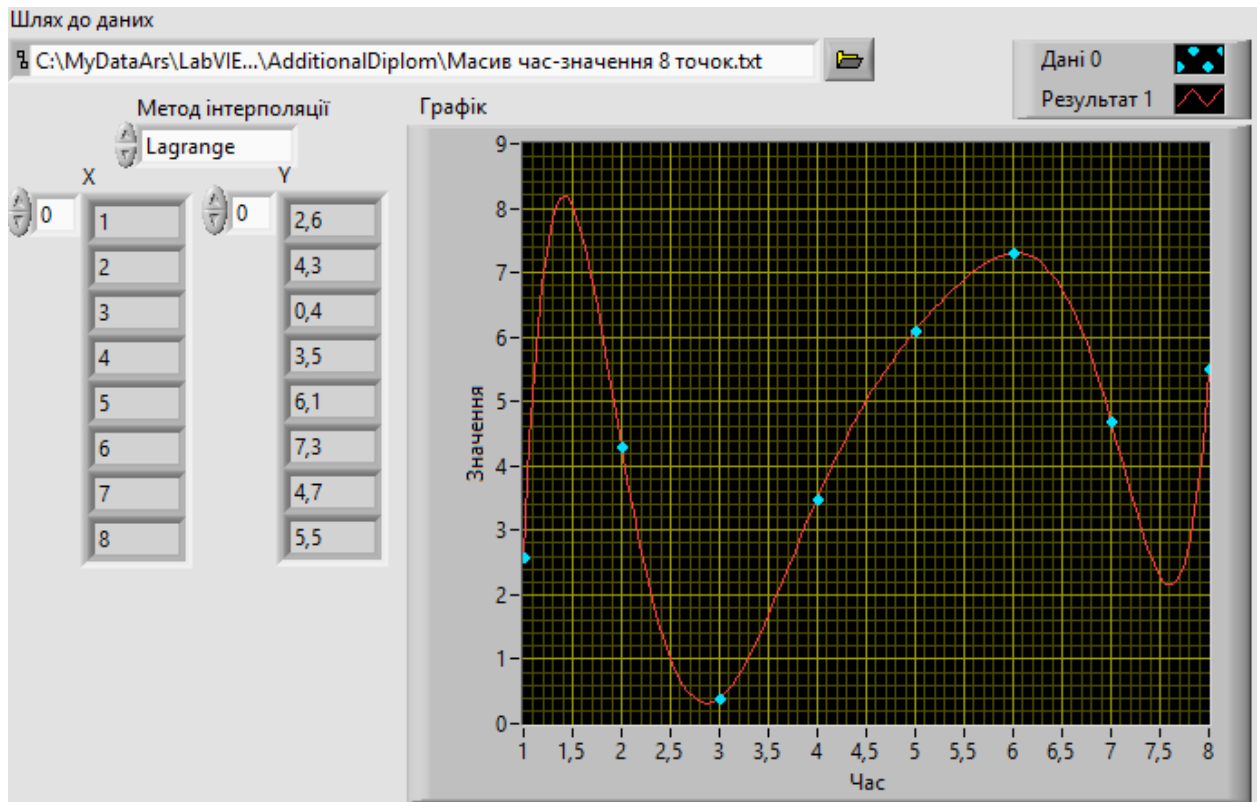


Рисунок 2.4 – Результат обчислювального експерименту з інтерполяції Лагранжа

2.2.3 Інтерполяція сплайнами

Поліноміальна інтерполяція не завжди дає необхідні результати при апроксимації функції. Незважаючи на виконання умов $F(x_n) = f(x_n) = y_n$ вузлах, інтерполуюча функція може мати значні відхилення між вузлами. Збільшення ступеню інтерполяції многочлена не завжди призводить до зменшення похибки. Виникає явлення хвилі між вузлами. При цьому, поведження полінома поблизу точки визначає його поведження у цілому. Поліноміальна інтерполяція призводить до значних похибок у разі, якщо в околиці лівої або правої межі інтервала інтерполяції знаходиться вертикальна асимптота [21].

На практиці, для побудови гладкої кривої використовують гнучку лінійку, з'єднуючи її з заданими точками. Математична теорія такої

апроксимації має назву теорія сплайн-функцій. Сплайн інтерполяцію вигідно використовувати при невеликій кількості вузлових точок (до 5-7) [21].

Якщо в якості функції $f(x)$ вибрати поліном, зі ступенем не вище 3, цей поліном називають кубічним сплайном, що на кожному інтервалі $x \in [x_{i-1}; x_i]$ має вигляд формули (2.11) [21]:

$$f(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \quad (2.11)$$

де a_i, b_i, c_i, d_i – коефіцієнти сплайна;

i – номер сплайна та $i = 1, 2, \dots, n$.

На відміну від поліноміальної інтерполяції, коли вся апроксимуюча функція описується одним поліномом, при сплайновій інтерполяції на кожному інтервалі $x \in [x_{i-1}; x_i]$ будується окремий поліном $\phi(x)$ 3-го ступеня зі своїми коефіцієнтами [21].

Апроксимуюча функція $f(x)$ являє собою послідовність сплайнів, зшитих між собою у точках, що відповідають вузловим значенням апроксимованої функції $F(x)$, що зображено на рисунку 2.5.

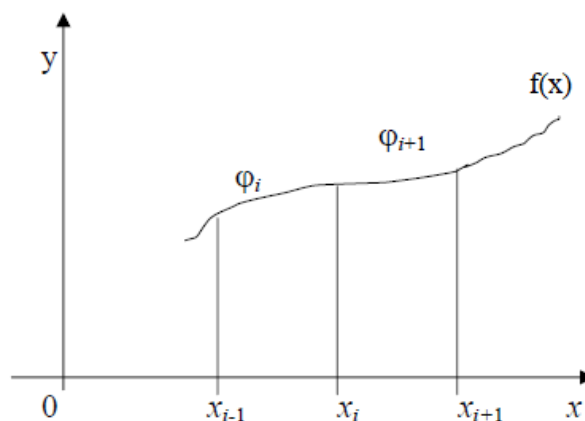


Рисунок 2.5 – Функція $f(x)$ при інтерполяції сплайном [21]

Умови зшивання сусідніх сплайнів:

– рівність значень сплайнів $\phi(x)$ та апроксимованої функції $f(x)$ вузлах за формулами (2.12) і (2.13) [21]:

$$\phi_i(x_{i-1}) = f(x_{i-1}) = y_{i-1}, \quad (2.12)$$

$$\phi_i(x_i) + f(x_i) = y_i; \quad (2.13)$$

– неперервність першої та другої похідної від сплайнів у вузлах інтерполяції – умова гладкості кривої [21].

Неперервність першої похідної означає, що сусідні сплайни в вузлових точках мають спільну дотичну. Неперервність другої похідної означає, що сусідні сплайни у вузлових точках мають однакову кривизну $\phi^{(I)}(x_i) = \phi_{i+1}^I(x_i)$, $\phi_i^{(II)}(x_i) = \phi_{i+1}^{II}(x_i)$ [21].

Інтерполяція методом сплайну була практично реалізована на наборі даних для обчислювального експерименту зображеному в таблиці 2.2 у середовищі розробки LabVIEW результат зображено на рисунку 2.6.

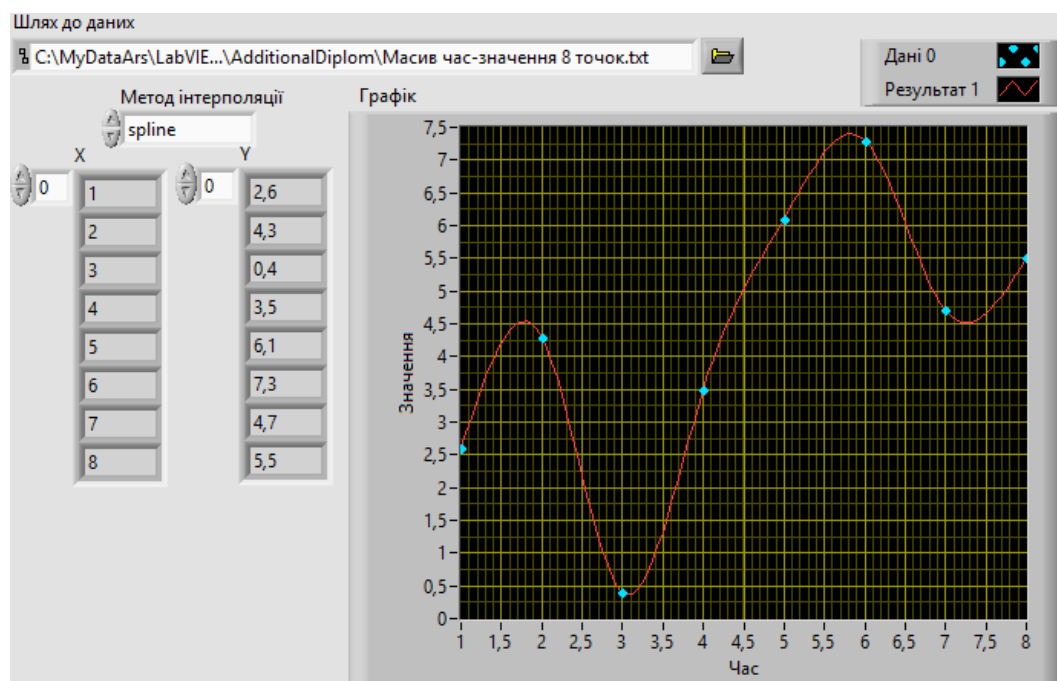


Рисунок 2.6 – Результат обчислювального експерименту з інтерполяції сплайном

Отже, сплайн інтерполяцію вигідно використовувати при невеликій кількості вузлових точок. На кожному інтервалі інтерполяції будується окремий поліном 3-го ступеня зі своїми коефіцієнтами. На відміну від інтерполяції поліномом, має меншу похибку.

2.2.4 Ермітова інтерполяція

Інтерполяція по Ерміту є спеціальним видом багато інтервальної інтерполяції, при якій інтерполюючий многочлен $H_n(x)$ забезпечує не тільки рівність $H_n(x)$ значенням $f(x)$, а також рівності деякої кількості похідних у вузлах інтерполяції. Для побудови апроксимуючої функції необхідний многочлен третього ступеня на кожному інтервалі, також сусідні многочлени повинні поєднуватися [22].

Для того, щоб побудувати інтерполюючу функцію, спочатку потрібно дослідити функцію многочлена третього ступеня за формулою (2.14) [22]:

$$f(x) = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3, \quad (2.14)$$

що визначена на відрізку $[0;1]$.

Коефіцієнти a_0, a_1, a_2, a_3 невідомі, їх можна визначити за допомогою наступних даних: $f(0) = y_0, f(1) = y_1, f'(0) = z_0, f'(1) = z_1$. Похідна функції f визначається за формулою (2.15) [22]:

$$f'(x) = a_1 + 2 \cdot a_2 x + 3a_3 x^2. \quad (2.15)$$

Напишемо рівняння: $f(0) = a_0 = y_0, f'(0) = a_1 = z_0, f(1) = a_0 + a_1 + a_2 = y_1$, де $f'(1)$ розраховується за формулою (2.16) [22]:

$$f'(1) = a_1 + 2a_2 + 3a_3 = z_1. \quad (2.16):$$

Напишемо рівняння у матричному вигляді за формулою (2.17) [22]:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}. \quad (2.17)$$

Перша частина системи лінійних рівнянь, це стовпчик з координатами

$$y_0, z_0, y_1, z_1, B = \begin{pmatrix} y_0 \\ z_0 \\ y_1 \\ z_1 \end{pmatrix}, \text{ вектор стовпчик невідомих змінних це } A = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} [22].$$

Далі потрібно вирішити систему лінійних рівнянь $P * A = B$, в загальному вигляді рішенням буде $A = P^{-1} * B$. Використовуючи алгоритм Гауса: у лівій частині таблиці записується матриця P , а праворуч приписується одинична матриця у вигляді формули (2.18) [22]:

$$P = \left(\begin{array}{ccc|ccc} 1000 & 1000 & & & & \\ 0100 & 0100 & & & & \\ 1111 & 0010 & & & & \\ 0123 & 0001 & & & & \end{array} \right). \quad (2.18)$$

Потрібно отримати обернену матрицю у вигляді формули (2.19) [22]:

$$\left(\begin{array}{c} P \\ E \end{array} \right) \longrightarrow \left(\begin{array}{c} E \\ P^{-1} \end{array} \right). \quad (2.19)$$

Отже, після перетворення отримаємо P^{-1} у вигляді формули (2.20) [22]:

$$P^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{pmatrix}. \quad (2.20)$$

За допомогою значень на кінцях відрізка $[0,1]$ та їх похідних відновлюємо всю функцію за формулами (2.21) – (2.25) [22]:

$$a_0 = y_0, \quad (2.21)$$

$$a_1 = z_0, \quad (2.22)$$

$$a_2 = -3 \cdot y_0 - 2 \cdot z_0 + 3 \cdot y_1 - z_1, \quad (2.23)$$

$$a_3 = 2 \cdot y_0 + z_0 - 2 \cdot y_1 + z_1, \quad (2.24)$$

$$f(x) = y_0 + z_0 x + (-3y_0 - 2z_0 + 3y_1 - z_1) \cdot x^2 + (2y_0 + z_0 - 2y_1 + z_1) \cdot x^3. \quad (2.25)$$

Нехай необхідно провести інтерполяцію на відрізку $[x_0; x_1]$, тоді графік функції необхідно зсунути на x_0 та розтягнути y , $x_1 - x_0$ разів, оскільки z_0 , z_1 у $x_1 - x_0$ разів менше, ніж потрібно, то їх необхідно до множити. Необхідно ввести k та h за формулами (2.26) та (2.27) [22]:

$$k = \frac{(x - x_0)}{(x_1 - x_0)}, \quad (2.26)$$

$$h = x_1 - x_0. \quad (2.27)$$

Функція (2.25) прийме вигляд формули (2.28) [22]:

$$f(k) = y_0 + z_0 * h * k + (-3 * y_0 - 2 * z_0 * h + 3 * y_1 - z_1 * h) * k^2 + (2 * y_0 + z_0 * h - 2 * y_1 + z_1 * h) * k^3. \quad (2.28)$$

Аналогічним чином можна визначити многочлен будь якого ступеню [22].

Інтерполяція поліномом Ерміта була практично реалізована на наборі даних для обчислювального експерименту зображеному в таблиці 2.2 у середовищі розробки LabVIEW результат зображено на рисунку 2.7.

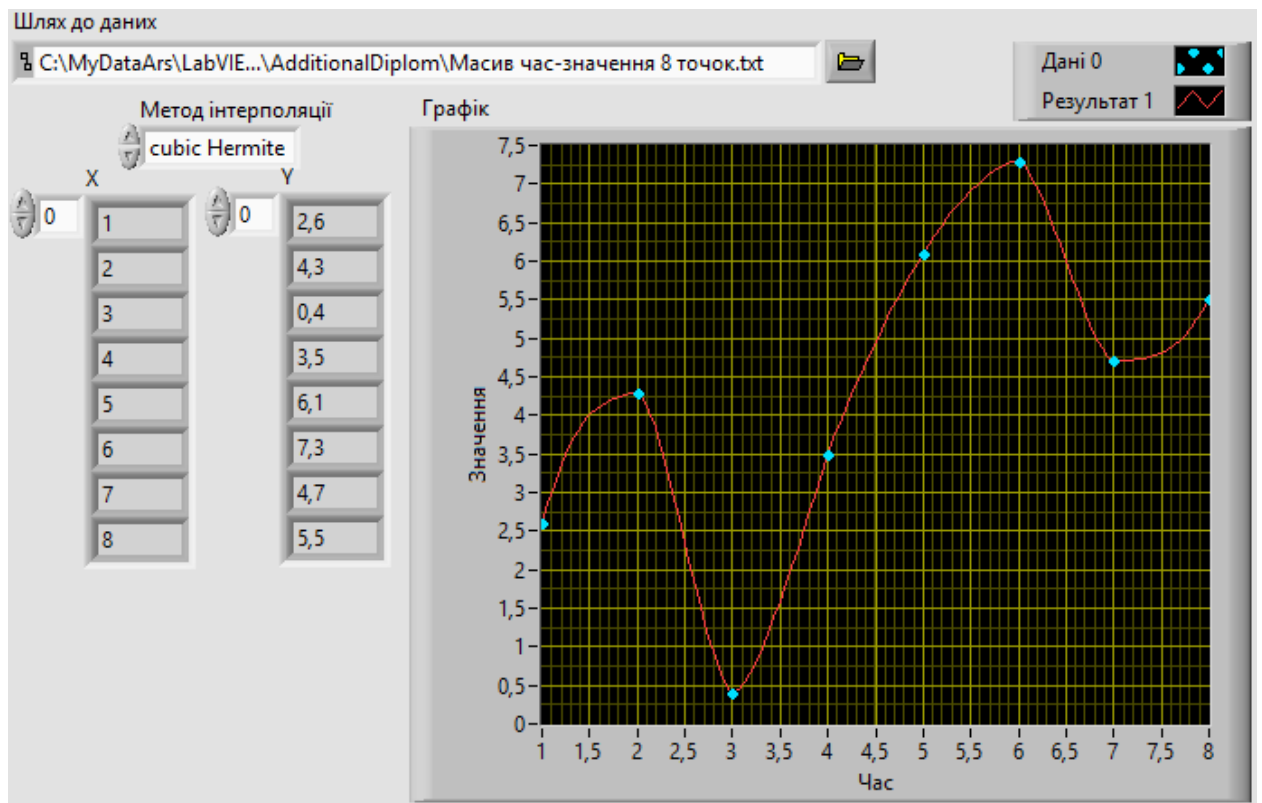


Рисунок 2.7 – Результат обчислювального експерименту з інтерполяції поліномом Ерміта

Отже, за допомогою Ермітової інтерполяції можливо досягти вищої точності в околиці вузлової точки за рахунок умови рівності деякої кількості похідних між суміжними поліномами. Ступінь апроксимуючих многочленів може бути більша за третю. Проте складність реалізації та кількість кроків необхідних для побудови поліному є більшою ніж у раніше розглянутих методах.

2.2.5 Апроксимація методом найменших квадратів

Однозначно відновити більшу частину функціональної залежності між змінними неможливо навіть у випадку відсутності похибки у експериментально отриманих величинах. Математична обробка результатів спостережень не ставить перед собою задачу визначення істинної залежності між змінними. Вона дозволяє представити результати експерименту в вигляді найбільш простої формули [20].

Нехай залежність $y = f(x)$ існує, тоді необхідно параметри цієї лінії вибрати таким чином, щоб точки y_i знаходилися по обидві сторони кривої $y = f(x)$ якомога ближче до останньої. Нехай, розкид точок y_i відносно $y = f(x)$ підкоряється закону нормального розподілу. Тоді, міра розкиду S є дисперсією σ^2 , або наближений до неї вираз – середній квадрат відхилень, яка розраховується за формулою (2.29) [20]:

$$S_n^2 = \frac{1}{n} \sum_{i=1}^n [y_i - F(x_i)]^2 = \frac{1}{n} \sum_{i=1}^n (\Delta y_i)^2. \quad (2.29)$$

Умова найнижчого розкиду буде виконана, якщо мінімізувати вираз $(\Delta y_i)^2$. Необхідною умовою мінімізації значення функції є рівність першої похідної нулю. Використання методу найменших квадратів має сенс, якщо кількість експериментальних точок n більше за число коефіцієнтів, що визначаються [20].

Далі наведена реалізація метода найменших квадратів на прикладі рівняння вигляду $y = ax + b$.

Для знаходження коефіцієнтів a та b шуканої прямої необхідно мінімізувати суму квадратів відстаней Δy_i по ординаті від точки $(x_i; y_i)$ до шуканої прямої. Відстань Δy_i визначається $\Delta y_i = y_i - ax_i - b$ Для мінімізації

$\sum_{i=1}^n \Delta y_i^2$ прирівнюємо до нуля похідні цієї суми по параметрам a, b за формулами (2.30) та (2.31) [20]:

$$\frac{\partial}{\partial a} = \left[\sum_{i=1}^n (y_i - ax_i - b)^2 \right]' = 2 \sum (y_i - ax_i - b)(-x_i) = 0, \quad (2.30)$$

$$\frac{\partial}{\partial b} = \left[\sum_{i=1}^n (y_i - ax_i - b)^2 \right]' = 2 \sum (y_i - ax_i - b)(-1) = 0. \quad (2.31)$$

Перетворюємо систему (2.32) [20]:

$$\begin{cases} a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i - \sum_{i=1}^n y_i x_i = 0 \\ a \sum_{i=1}^n x_i + bn - \sum_{i=1}^n y_i = 0 \end{cases}, \quad (2.32)$$

отримуємо систему нормальних рівнянь метода найменших квадратів. Розв'язуючи її відносно a, b отримуємо за формулами (2.33) та (2.34) [20]:

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2}, \quad (2.33)$$

$$b = \frac{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i - \sum_{i=1}^n x_i \sum_{i=1}^n x_i y_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2}. \quad (2.34)$$

Знаходячи з n експериментів необхідні суми та виконуючи вказані дії, отримуємо величину коефіцієнтів a, b . Метод найменших квадратів може використовуватися також у випадку нелінійної функції [20].

Розглянемо апроксимуючу функцію виду $F(x_i, a_0, a_1, \dots, a_m)$, де $a_0, a_1, \dots, a_m$ – невідомі параметри. Сутність методу найменших квадратів полягає у тому, щоб сума квадратів відхилень функції $F(x_i, a_0, a_1, \dots, a_m)$ відносно експериментальних значень була мінімальною (формули (2.35) та (2.36)) [20]:

$$\sum_{i=1}^n (y_i - F(x_i, a_0, a_1, \dots, a_m))^2 \longrightarrow \min, \quad (2.35)$$

$$\sum_{i=0}^n (y_i - F(x_i, a_0, a_1, \dots, a_m))^2 = Q(a_0, a_1, \dots, a_m). \quad (2.36)$$

Функція $Q(a_0, a_1, \dots, a_m)$ приймає мінімальні значення у разі, якщо похідні цієї функції по кожному параметру дорівнюють нулю формула (2.37) [20]:

$$\frac{\partial}{\partial a_m} Q(a_0, a_1, \dots, a_m) = 0. \quad (2.37)$$

Знаходимо часні похідні за кожним параметром, прирівнюємо їх до нуля та об'єднаємо в систему рівнянь за формулою (2.38) [20]:

$$\sum_{i=0}^n (y_i - F(x_i, a_0, a_1, \dots, a_m)) \frac{\partial}{\partial a_m} F(a_0, a_1, \dots, a_m) = 0. \quad (2.38)$$

Розв'язавши систему (2.38) знайдемо значення параметрів $a_0, a_1, \dots, a_m$ та шукану апроксимуючу функцію $F(x_i, a_0, a_1, \dots, a_m)$, що задовольняє сформульованим вище умовам [20].

Підставимо значення параметрів $a_0, a_1, \dots, a_m$ до рівняння (2.38) та отримуємо рівняння (2.39) [20]:

$$\sum_{i=0}^n (y_i - F(x_i, a_0, a_1, \dots, a_m))^2 = Q(a_0, a_1, \dots, a_m). \quad (2.39)$$

Знайдемо значення Q – суми квадратів відхилень табличних значень y_i відносно отриманої апроксимуючої функції $F(x)$. Величина ∂ називається середнім квадратичним відхиленням, та демонструє міру дисперсії експериментальних точок порівняно з апроксимуючою функцією та розраховується за формулою (2.40) [20]:

$$\partial = \sqrt{\frac{Q}{(n+1)}}. \quad (2.40)$$

Дисперсія випадкової величини S^2 – це міра розкиду значень випадкової величини відносно її математичного очікування, яка розраховується за формулою (2.41) [20]:

$$S^2 = \frac{\sum (x^i - \bar{x})^2}{n-1}, \quad (2.41)$$

де S^2 – дисперсія вибірки;

x^i – значення параметра;

$\bar{x}$ – середнє значення по всіх спостереженнях;

n – кількість спостережень [20].

Апроксимація методом найменших квадратів була практично реалізована на наборі даних для обчислювального експерименту зображеному в таблиці 2.2 у середовищі розробки LabVIEW результат зображено на рисунку 2.8.

Можна зробити висновок, що спосіб найменших квадратів достатньо громіздкий та при його використанні використовується обчислювальна техніка. Використання цього методу має сенс у випадку наявності пари чисел x_i , y_i , що являються експериментальними точками. Притому, кожне значення

y_i складається з двох доданків. Зі значення невідомої функції $f(x_i)$ та її похибки ε_i . Значення похибки є велике, в задачі їм знехтувати не можна.

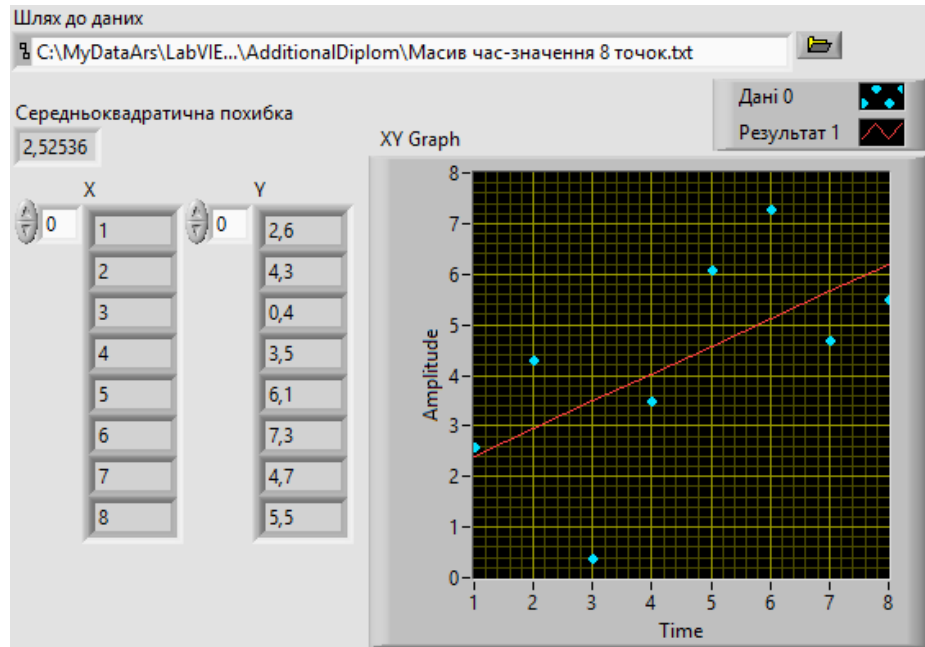


Рисунок 2.8 – Результат обчислювального експерименту з апроксимації методом найменших квадратів

2.3 Обчислювальні експерименти

Для проведення достовірного дослідження було записано набір експериментальних даних за допомогою прототипу підсистеми вимірювання температури та вологості, також отримано дані із випробувального стенду реального часу при випробуванні авіаційного двигуна.

2.3.1 Дослідження на наборі даних з прототипу підсистеми вимірювання температури та вологості

Завдання дослідження полягає у визначенні найкращих методів апроксимації для візуалізації даних отриманих з прототипу підсистеми вимірювання температури та вологості.

План дослідження:

- зібрати експериментальні дані вологості, температури та побудувати їх графіки;
- обрати кількість точок для інтерполяції;
- реалізувати методи інтерполяції та побудувати візуальний інтерфейс для порівняння отриманих результатів в середовищі розробки LabVIEW;
- визначити найкращий метод апроксимації для візуалізації експериментальних даних температури, вологості.

Для проведення експериментального дослідження були зібрані дані про температуру та вологість в приміщенні за допомогою датчика температури та вологості DHT – 11. Для передачі даних на комп'ютер була використана плата Arduino UNO (рис. 2.9). Плата передає дані через COM порт. Дані були зчитані та записані у масив.



Рисунок 2.9 – Прототип підсистеми вимірювання температури та вологості

Швидкість зчитування параметрів складає один раз за секунду, експеримент тривав 30 секунд (рис. 2.10). Дані з масиву були перенесені до текстового файлу. З файлу поводилося читання даних та їх апроксимація. Графік був побудований в реальному часі за допомогою лінійної інтерполяції. На ньому можна побачити різкі піки. Поводження температури та вологості в точках максимуму та мінімуму не відповідає дійсності, через це виникають похибки вимірювань.

Необхідно знайти метод інтерполяції, який забезпечить більшу плавність графіку, та буде точніше представляти реальні дані та поведінку параметру під час вимірювань.

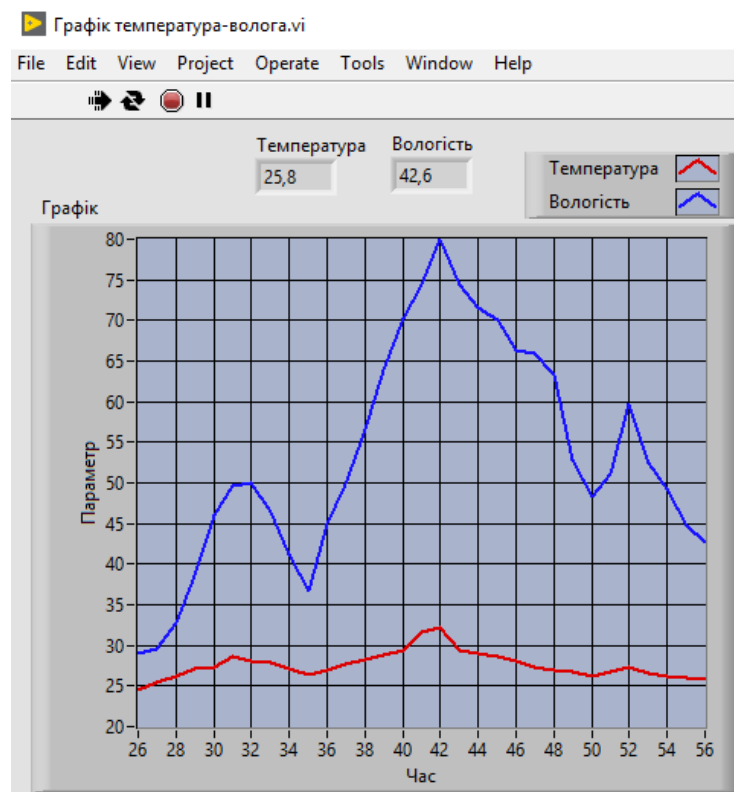


Рисунок 2.10 – Експериментальні дані вологості та температури

Для проведення дослідження був розроблений віртуальний інструмент на мові програмування LabVIEW. Серед можливостей розробленого інструмента:

- відображення графіків інтерполяції та вузлових точок;

- можливість вибору різних наборів даних для інтерполяції;
- можливість вибору метода інтерполяції зі списку;
- відображення сирих та інтерпольованих даних у масиві.

Масив збережених даних було відображено в розробленому інструменті, використовувалася лінійна інтерполяція, як на минулому графіку для візуалізації отриманих даних з Arduino Uno (рис. 2.11).

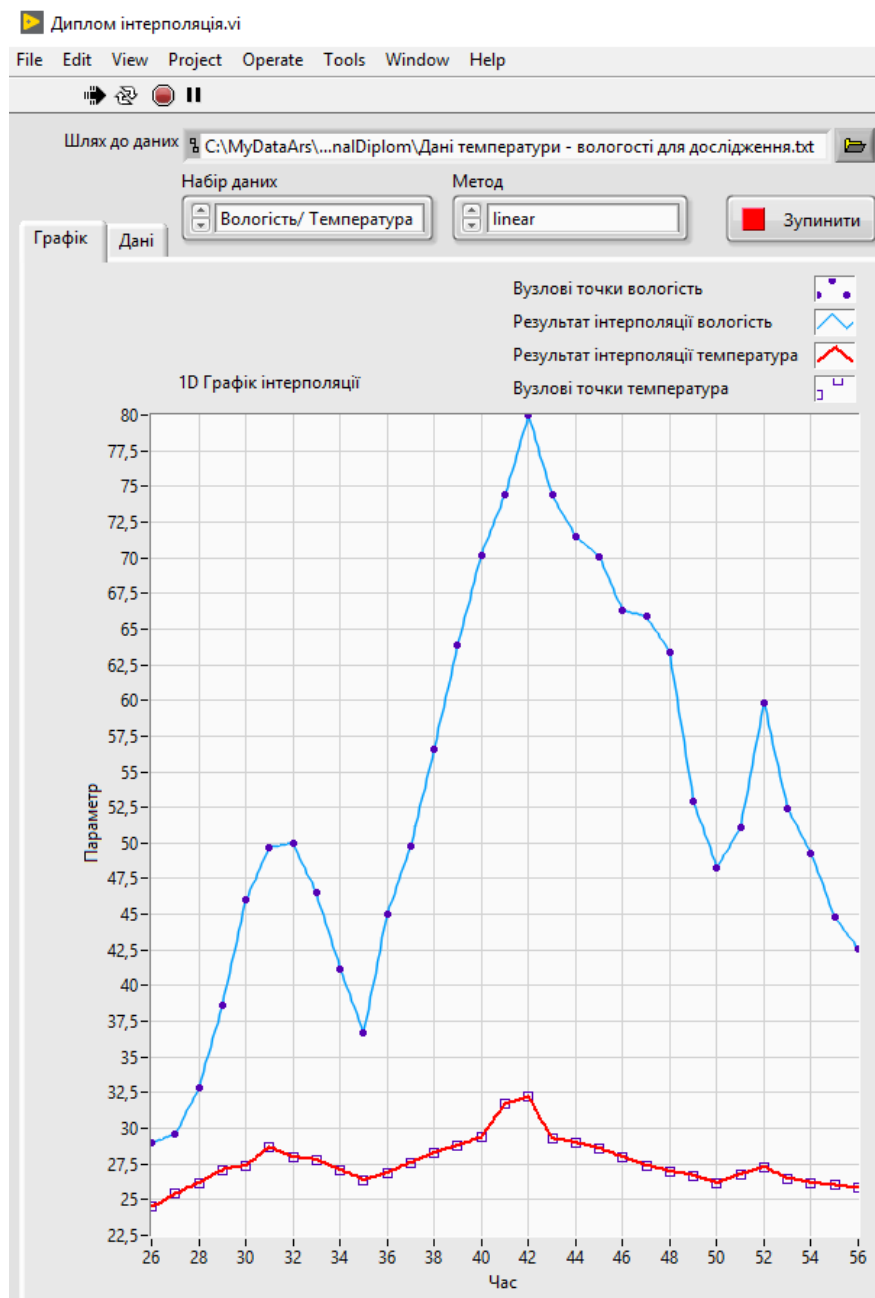


Рисунок 2.11 – Результат лінійної інтерполяції експериментальних даних

За планом дослідження, була проведена інтерполяція методом кубічного сплайну (рис. 2.12). Кожен інтервал вузлових точок був розбитий на 10 інтервалів для інтерполяції у кожній створеній точці проводилася інтерполяція.

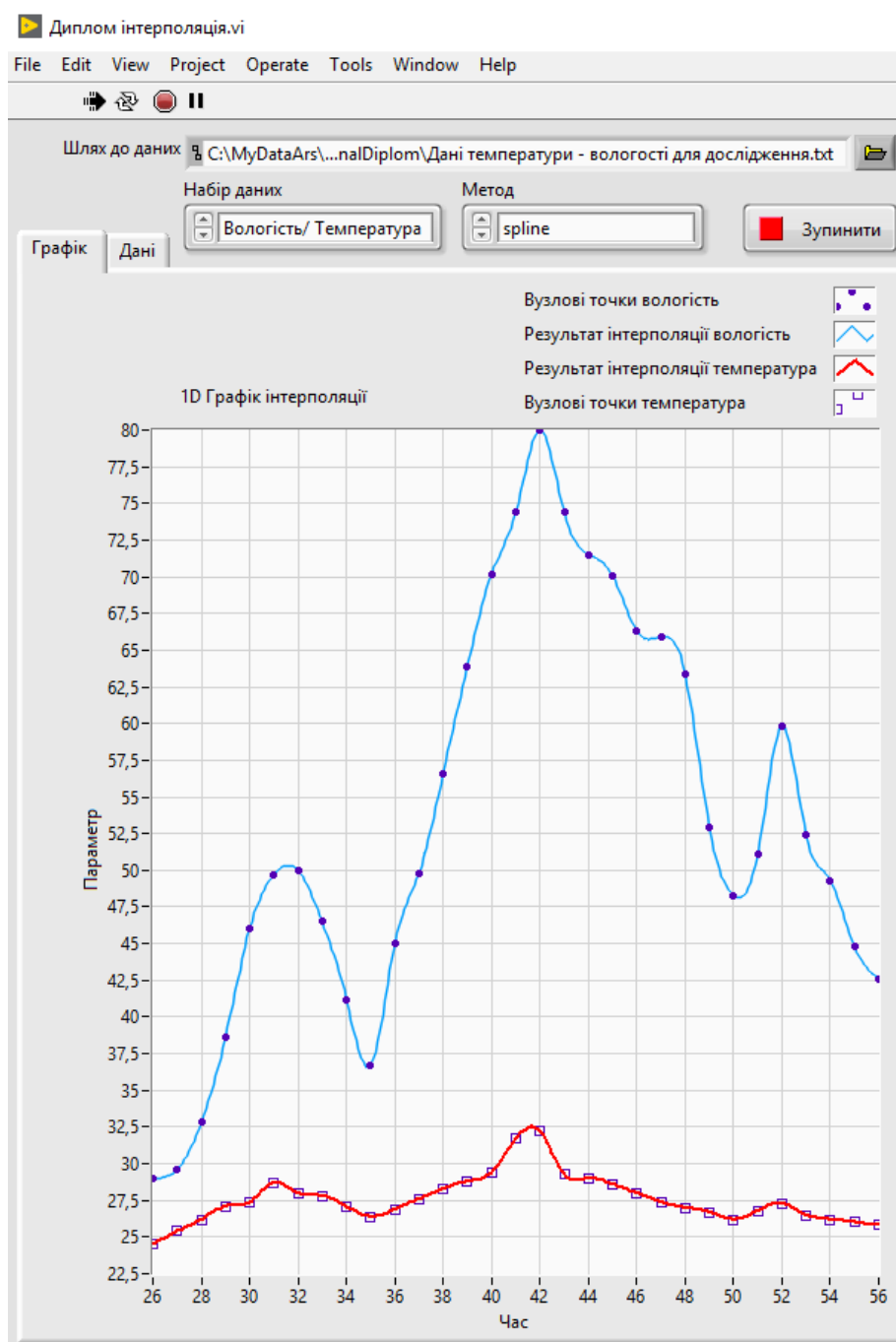


Рисунок 2.12 – Результат інтерполяції експериментальних даних кубічним сплайном

Результат був досягнений, різкі піки на графіку вологи стали більш пологими, цей графік вдалося наблизити до реального поведіння параметру волога. У точках з близьким значенням Час = [44;45] було помічено випуклість графіку, на інтервалі Час = [46;47] помічена впуклість графіку, на інтервалі Час = [31;32] графік сплайн функції перевищив зареєстрований максимум. На графіку температури є незначні зміни, графік став більш пологим. На інтервалі Час = [41;42] графік сплайн функції перевищив зареєстрований максимум. Для оцінки методу були досліджені інтерпольовані дані (рис. 2.13).

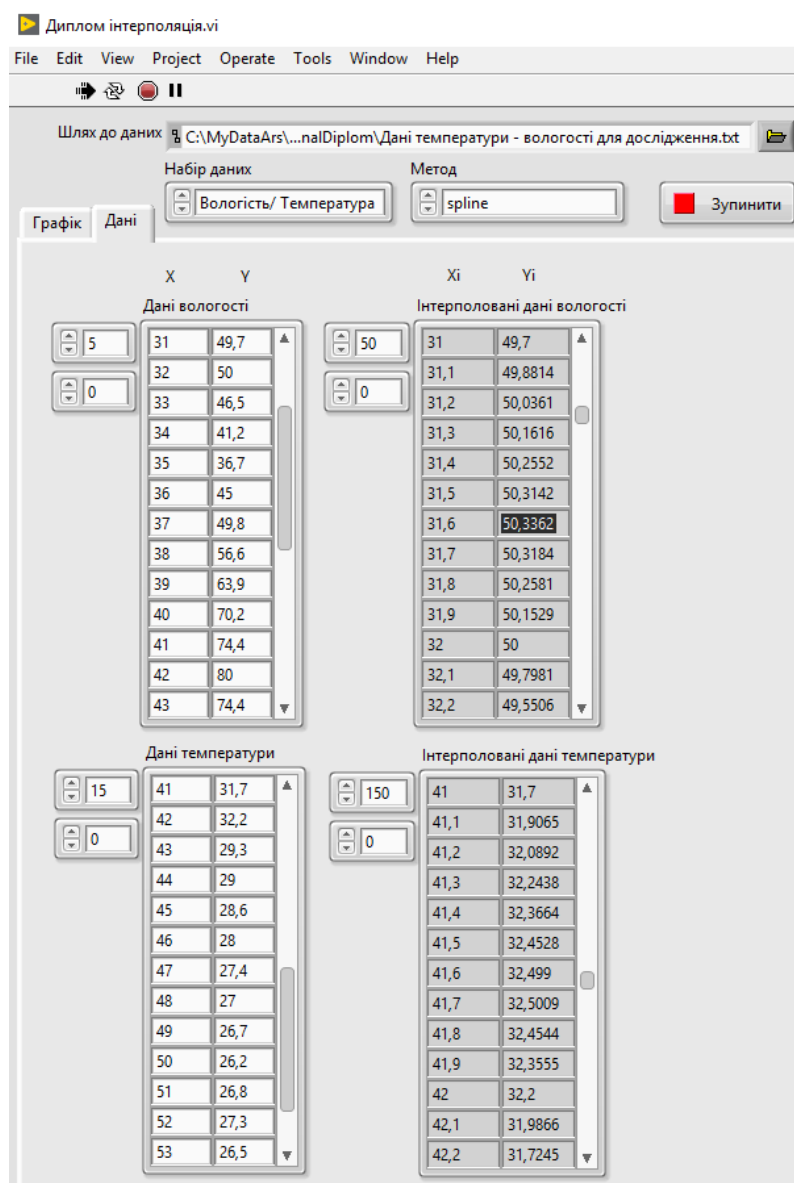


Рисунок 2.13 – Масиви сирих та інтерпольованих даних методом сплайну

Аналізуючи дані інтерполяції сплайном на всіх зазначених діапазонах було помічено вихід параметрів вологи та температури за межі діапазону вимірювань в середньому на 0,25%. Враховуючи похибку датчика DHT – 11 ($\pm 5\%$ для вологи та $\pm 2\%$ для температури) цією похибкою можна знехтувати.

Результат виявився задовільним для графіку вологи, більша частина графіку відповідає реальному поведженню параметра, пологість графіку візуально помітна та точніше передає значення параметра. На графіку температури результат ледь помітний. З цього можна зробити висновок, що параметри з більшою дисперсією краще піддаються методам інтерполяції кубічним сплайном. На обох графіках була помічена похибка інтерполяції з середнім значенням 0,25%.

На наступному етапі досліджень була проведена інтерполяція за допомогою полінома Ерміта (рис. 2.14).

Досліджуючи поведження графіку на раніше проаналізованих діапазонах графіку вологи (Час = [44;45]; [46;47]; [31;32]) та графіку температури Час = [41;42] було зроблено висновок, що графік інтерполяції по Ерміту менш пологий ніж графік інтерполяції сплайном. Було досліджено дані інтерполяції (рис. 2.15).

Проаналізувавши дані зроблено висновок, що похибка у досліджуваних діапазонах відсутня, порівняно з інтерполяцією методом кубічного сплайну. Параметри вологи та температури знаходяться в межах досліджуваних діапазонів.

Результат виявився кращим для графіку вологи та температури, більша частина графіку відповідає реальному поведженню параметра, пологість графіку візуально помітна, точніше передає значення параметру. Порівняно з інтерполяцією сплайном, інтерполуюча функція не виходить за межі діапазонів. Можна зробити висновок, що метод інтерполяції по Ерміту виявився середнім варіантом між грубою лінійною інтерполяцією та пологою інтерполяцією за допомогою кубічного сплайну.

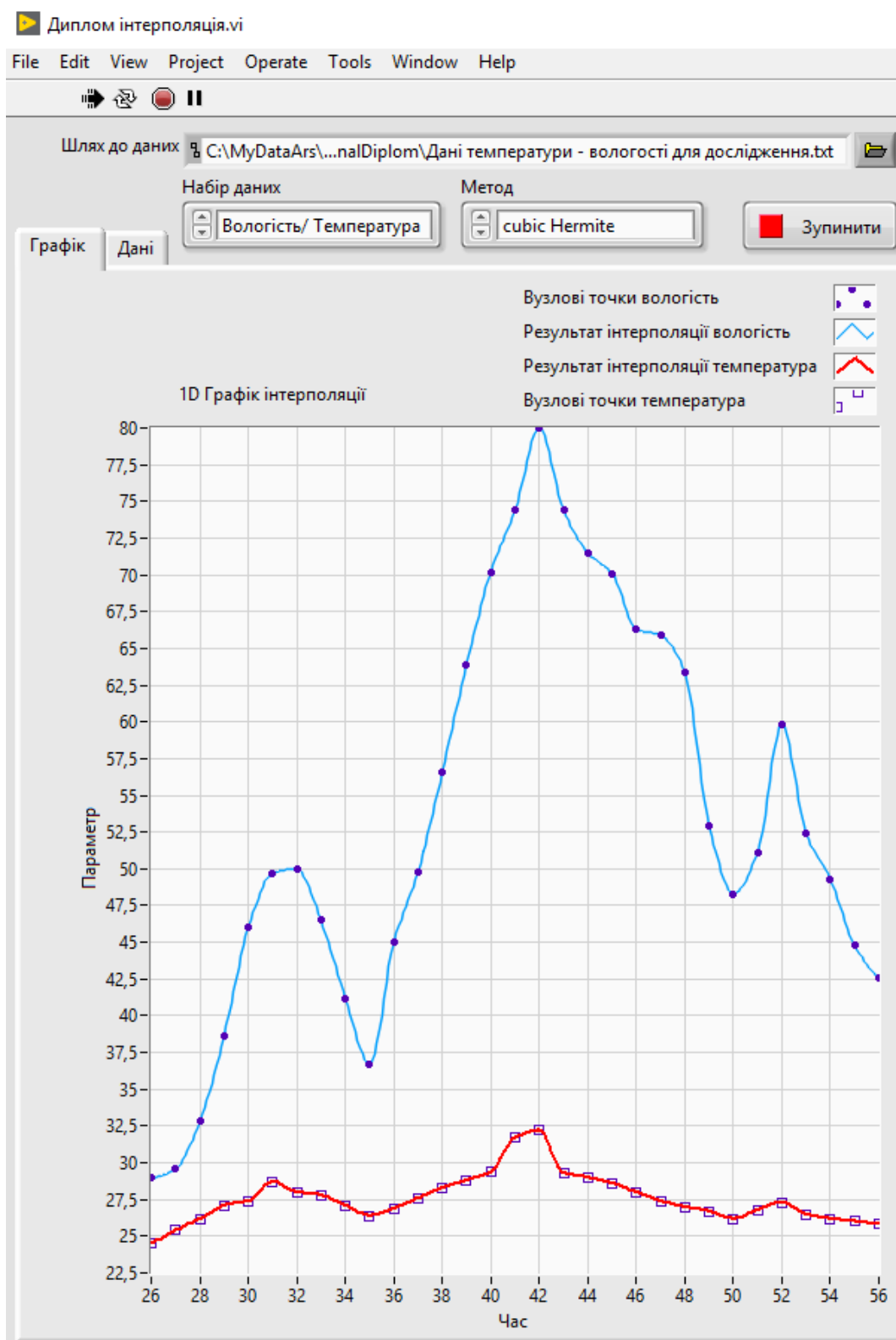


Рисунок 2.14 – Результат Ермітової інтерполяції експериментальних даних

Було досліджено дані інтерполяції (рис. 2.15).

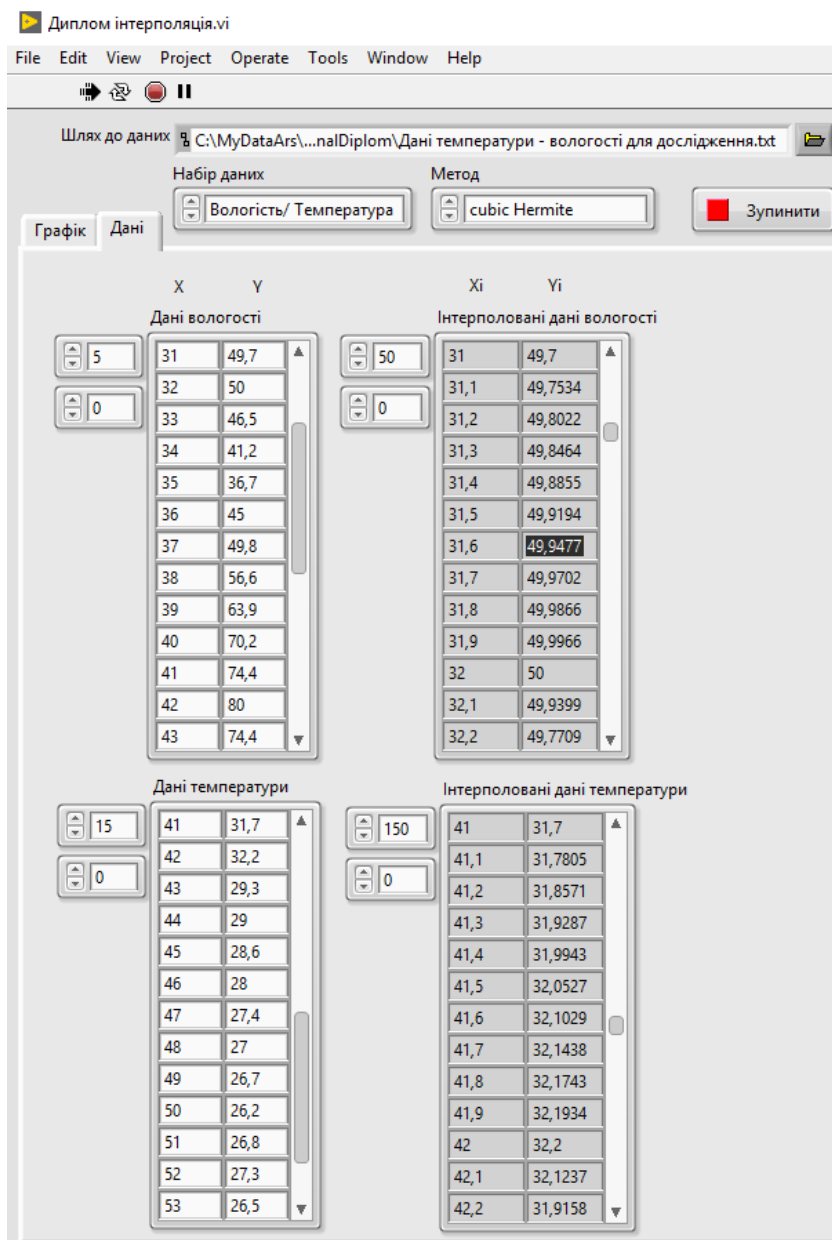


Рисунок 2.15 – Масиви сирих та інтерпольованих даних методом Ерміта

На наступному етапі досліджень була проведена інтерполяція за допомогою полінома Лагранжа (рис. 2.16).

Проаналізувавши графік була помічена екстремальна похибка біля крайніх діапазонів вузлових точок (Час = [26;28]; [54;56]). Дослідження даних на інтервалах не має сенсу. Зважаючи на похибку можна зробити висновок, що метод інтерполяції Лагранжа не дає достовірні результати апроксимації, не підходить для візуалізації експериментального набору даних.

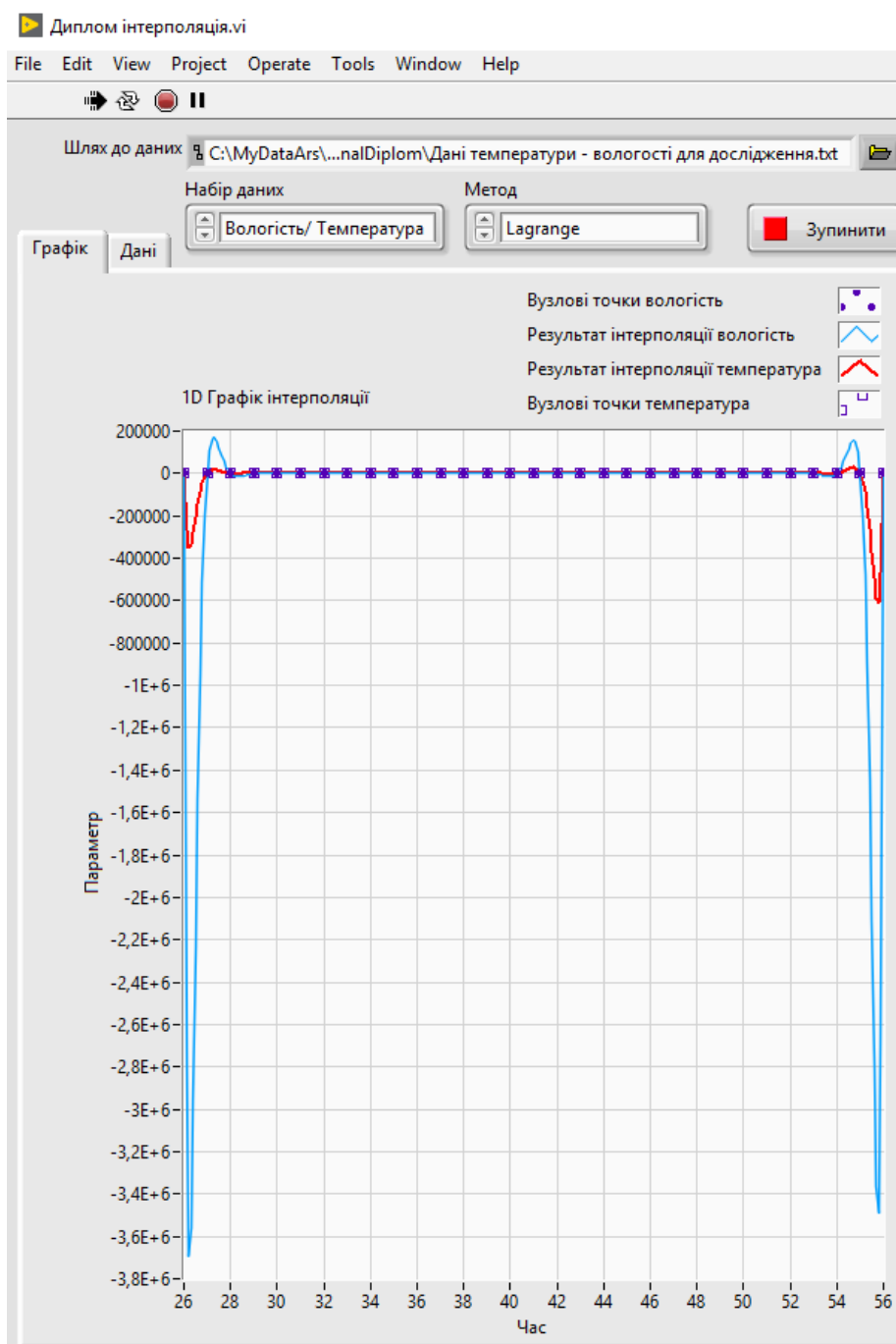


Рисунок 2.16 – Результат інтерполяції експериментальних даних методом Лагранжа

Для дослідження експериментальних даних методом апроксимації найменших квадратів було розроблено відповідний інструмент на мові LabVIEW (рис. 2.17).

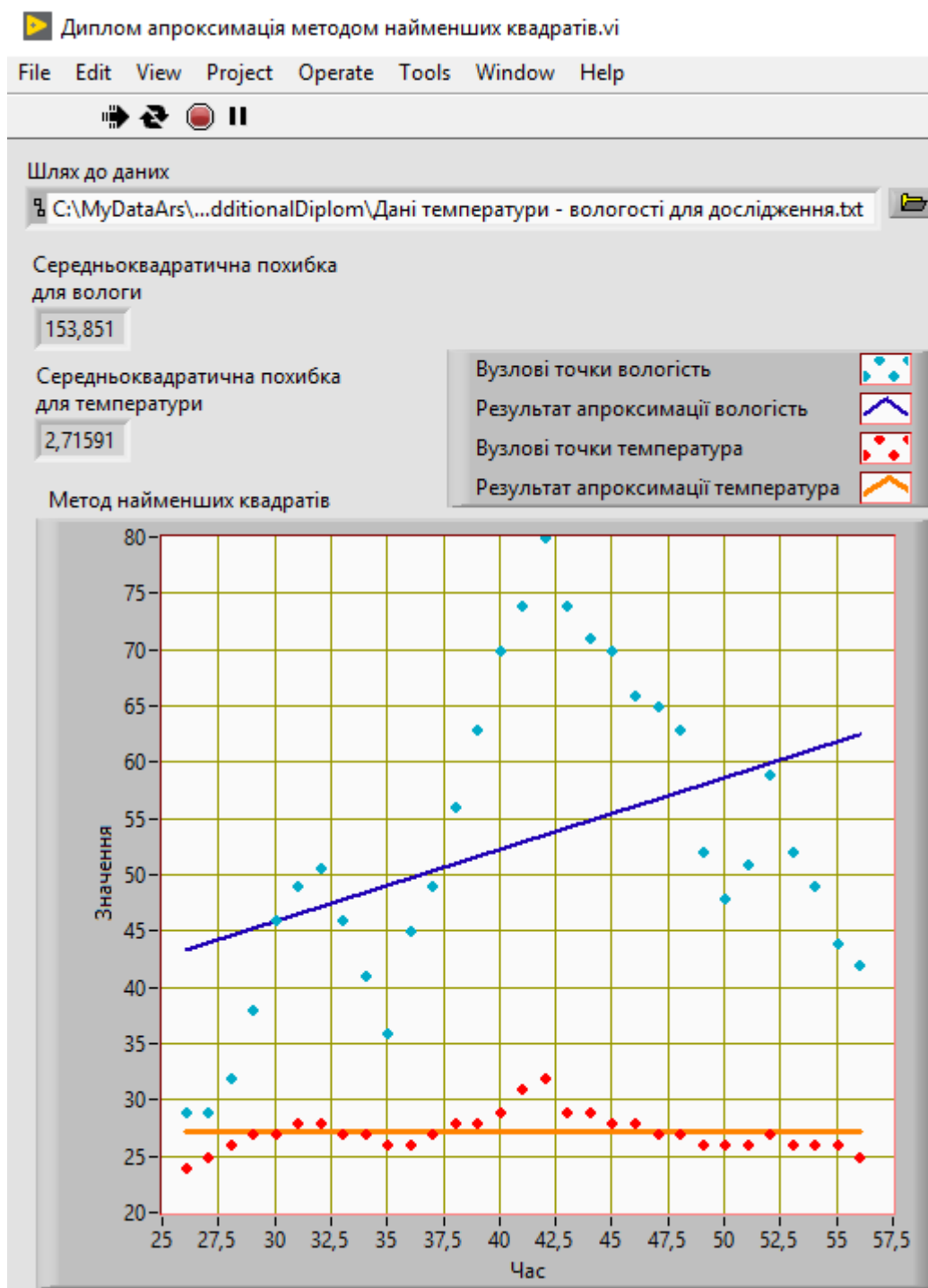


Рисунок 2.17 – Результат апроксимації експериментальних даних методом найменших квадратів

Досліджуючи результат апроксимації методом найменших квадратів було зроблено висновок, що даний метод не підходить для вирішення задачі візуалізації вимірюваних параметрів. Метод надає візуалізацію загального поведінки параметру та його тренд, не надає можливості визначити реальні

дані в будь-який момент часу. Апроксимацію методом найменших квадратів доцільно використовувати для даних отриманих зі значною похибкою, у випадку набору експериментальних даних похибка складає $\pm 5\%$ для вологи та $\pm 2\%$ для температури, що не є значною похибкою, тому результат апроксимації не задовільний для задачі візуалізації вимірюваних параметрів. Висновок з експерименту полягає у використанні методів інтерполяції для побудови графіків по вузловим точкам для подальших досліджень.

За результатами дослідження та експериментів була побудована порівняльна таблиця 2.3 проаналізованих методів апроксимації на масиві даних отриманих з прототипу підсистеми вимірювання температури та вологості. Для оцінки методів використовувалася шкала балів {1;2;3}, де «1» найгірша оцінка, «3» – найкраща.

Таблиця 2.3 – Порівняння методів апроксимації для набору експериментальних даних

Критерій оцінки	Методи апроксимації				
	лінійний	лагранж	сплайн	ерміт	найменших квадратів
Складність реалізації	3	2	2	1	2
Використовується для аналізу даних без похибки	3	3	3	3	1
Забезпечує плавність графіку	1	2	3	2	1
Демонструє найнижчу похибку	2	1	2	3	1
Відповідає реальному поведженню параметра	2	1	3	3	1
Сума балів	11	9	13	12	6

В результаті проведеного дослідження на експериментальному наборі даних було встановлено, що метод інтерполяції сплайном виявився кращим за інші, він демонструє найкращу пологість при низькій похибці. Метод найменших квадратів не відповідає меті досліджень та не буде використовуватися в подальших дослідженнях. Метод Лагранжа дає екстримальну похибку.

2.3.2 Дослідження методів інтерполяції на стендових даних з авіаційного двигуна

Завдання дослідження полягає у визначенні найкращих методів інтерполяції для візуалізації даних отриманих з випробувального стенду.

Для другого етапу досліджень використовувалася база даних отримана при випробуванні двигуна моделі АІЗ36, тип випробувань «пред'явницькі» (сдавальні), дані отримані за допомогою програми для проведення випробувань авіаційних двигунів Octopus. Досліджувана база даних зображена на рисунку 2.18, вертикальною жовтою лінією на графіку зображено момент часу до якого база даних була досліджена. Для проведення дослідження було обрано параметри Q_{t1} (витрати палива, вимірюється у літрах за годину) та t_{tnt} (температура турбіни низького тиску, вимірюється у градусах за Цельсієм).

Для проведення дослідження на стендовому наборі даних, досліджувані параметри у вигляді звіту записано до текстового файлу за допомогою вбудованих функцій протоколювання програми Octopus. З отриманого текстового файлу дані були зчитані за допомогою програми для проведення інтерполяції, розробленої на минулому етапі дослідження методів апроксимації. Далі була проведена лінійна інтерполяція стендових даних (рис. 2.19). Кожен інтервал вузлових точок був розбитий на 4 інтервали для проведення інтерполяції на кожному з них.

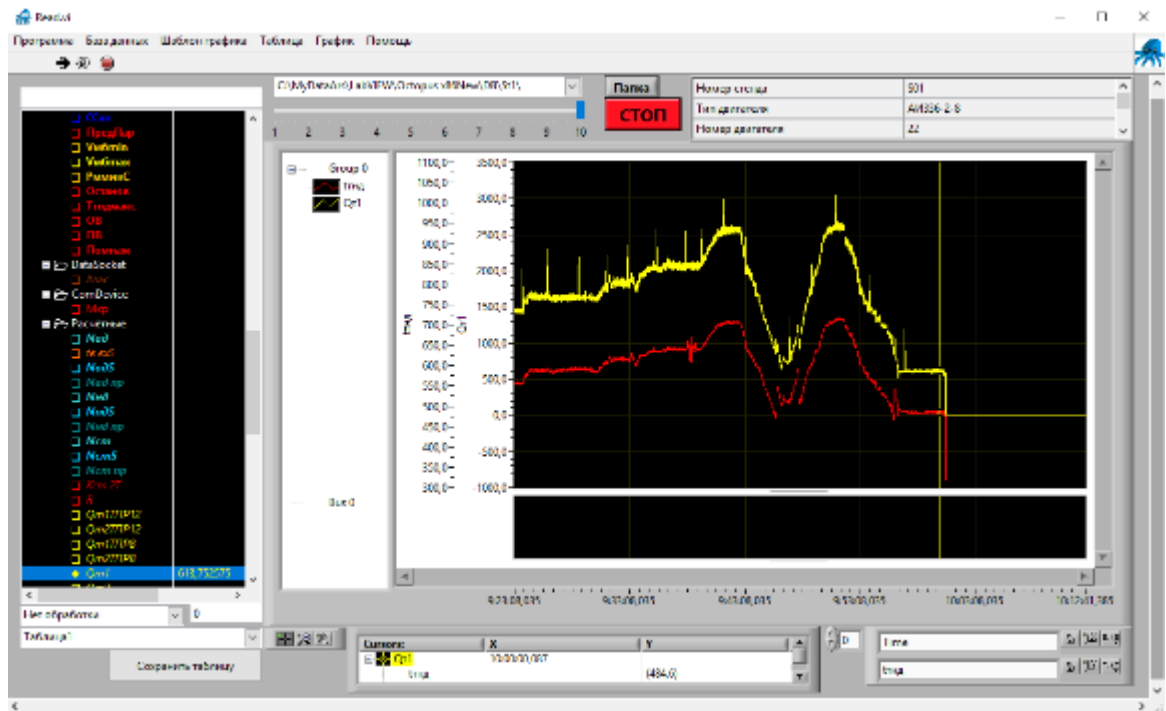


Рисунок 2.18 – Досліджувана база даних візуалізована за допомогою Ostorus

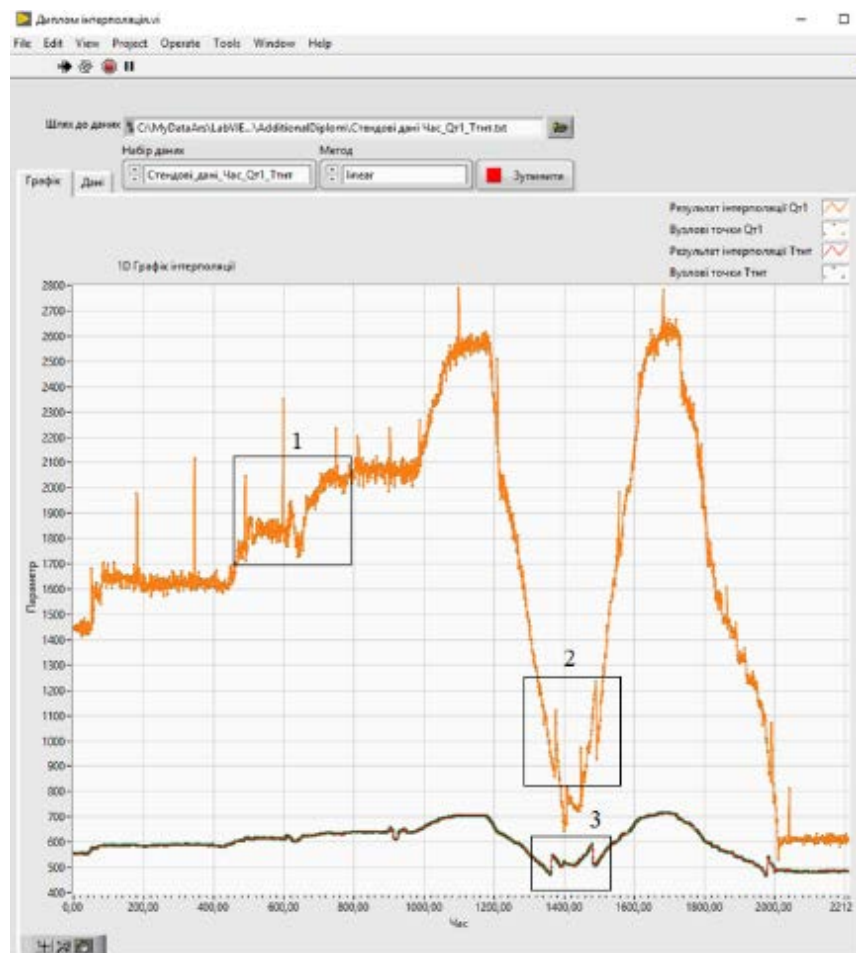


Рисунок 2.19 – Результат лінійної інтерполяції стендового набору даних

Через обмеженість простору, графік вийшов досить щільним, тому було прийняте рішення досліджувати параметри на 3 сегментах графіку. Вони виділені прямокутниками та відповідно пронумеровані.

За планом дослідження, була проведена інтерполяція методом кубічного сплайну (рис. 2.20).



Рисунок 2.20 – Результат інтерполяції сплайном стендового набору даних

Порівнявши візуально 2 графіка можна зробити висновок, що різниця відсутня. Необхідно з'ясувати, чи є різниця при відображенні графіків у більшому масштабі. Графіки можуть відображатися на 24 дюймових моніторах, на яких відмінності можуть бути помітні. Для цього зроблено порівняння 2 методів інтерполяції (лінійного та сплайну) на графіках у сегменті 1, результат зображено на рисунку 2.21.

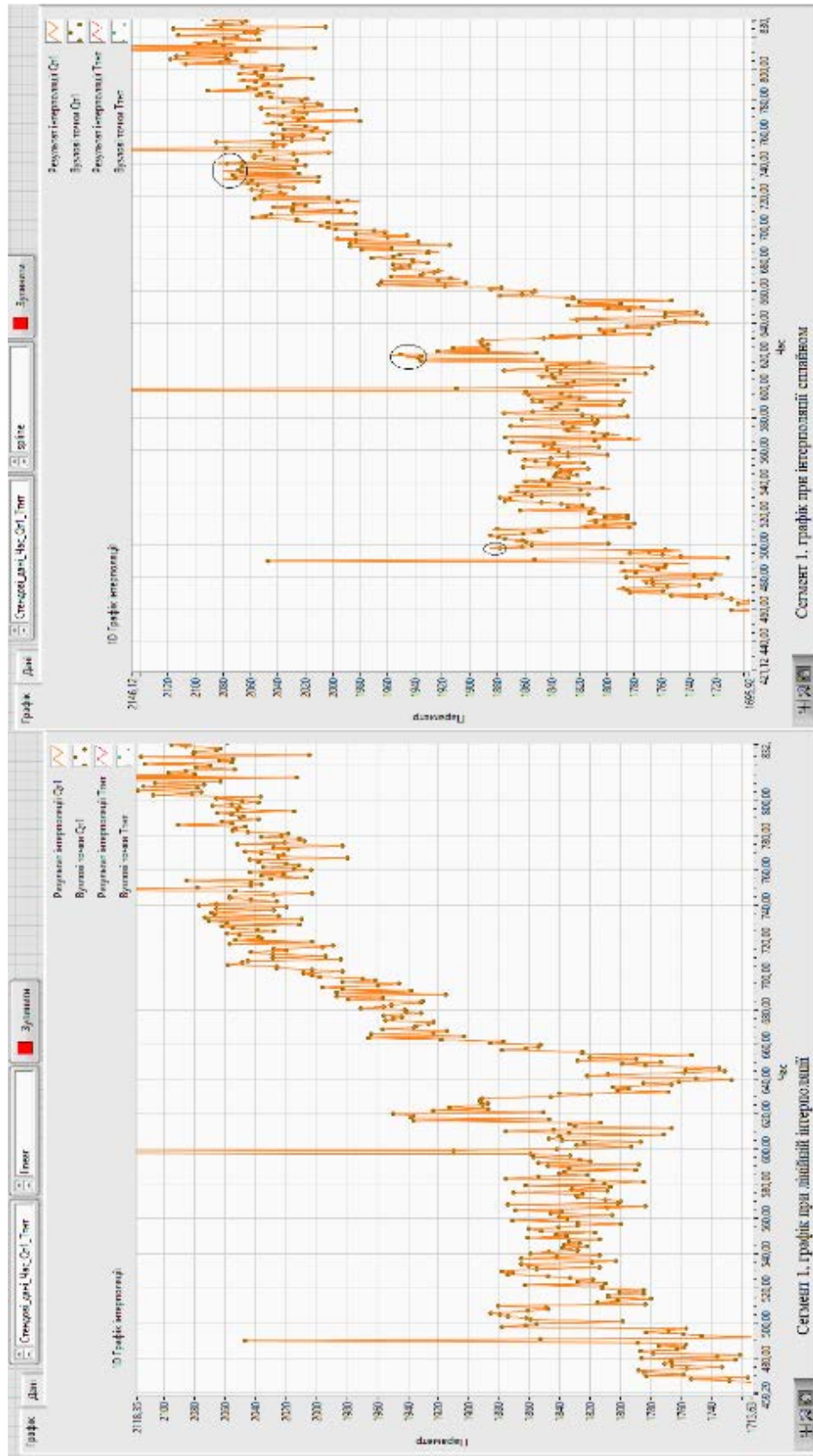


Рисунок 2.21 – Порівняння 2 методів інтерполяції на сегменті 1

Під час порівняння виявлено, що пологість на графіку інтерполяції сплайном – відсутня, проте з'явилися похибки значень у точках відмічених овалами, де $x = \{500, 620, 740\}$. Для підтвердження візуальних спостережень необхідно знайти ці точки у масиві інтерпольованих даних (рис. 2.22).

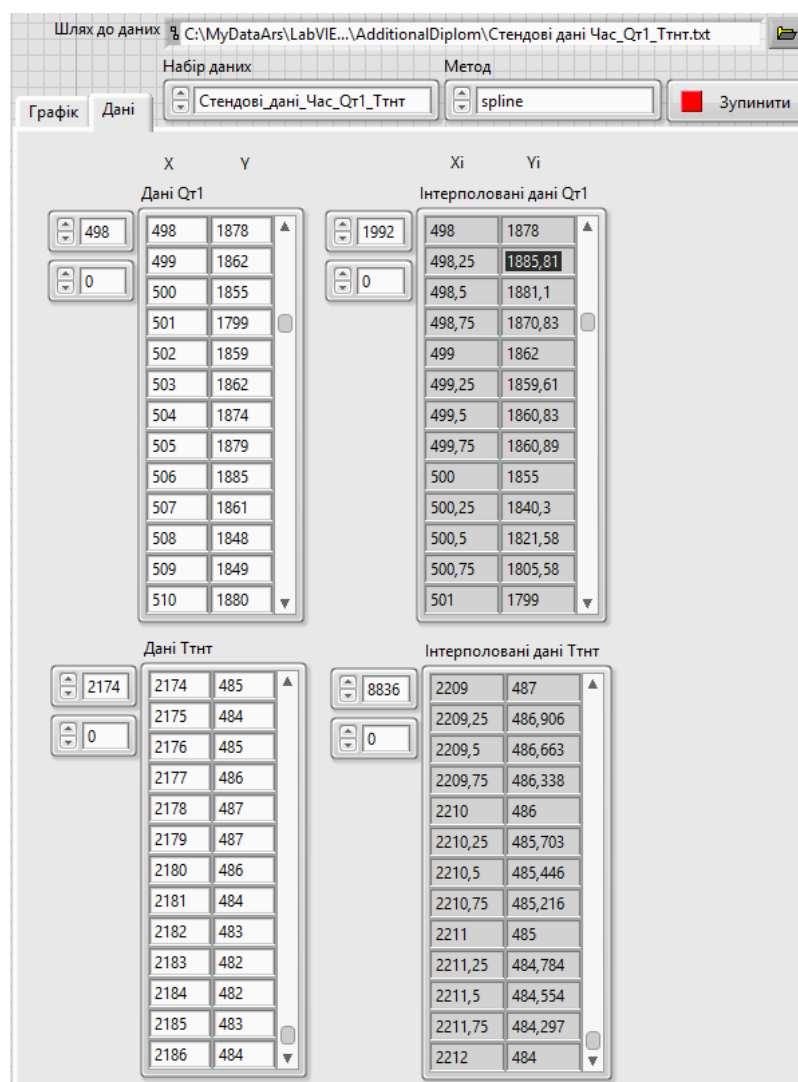


Рисунок 2.22 – Аналіз інтерпольованих даних на сегменті 1

Проаналізувавши дані можна зробити висновок, що параметр Qt1 на зазначених діапазонах виходить за межі вимірювань в середньому на 0,08%. Враховуючи похибку датчика витрат у 0,5% похибкою інтерполяції можна знехтувати. Було проведено порівняння графіків параметра Ttnt на сегменті 3 (рис. 2.23).

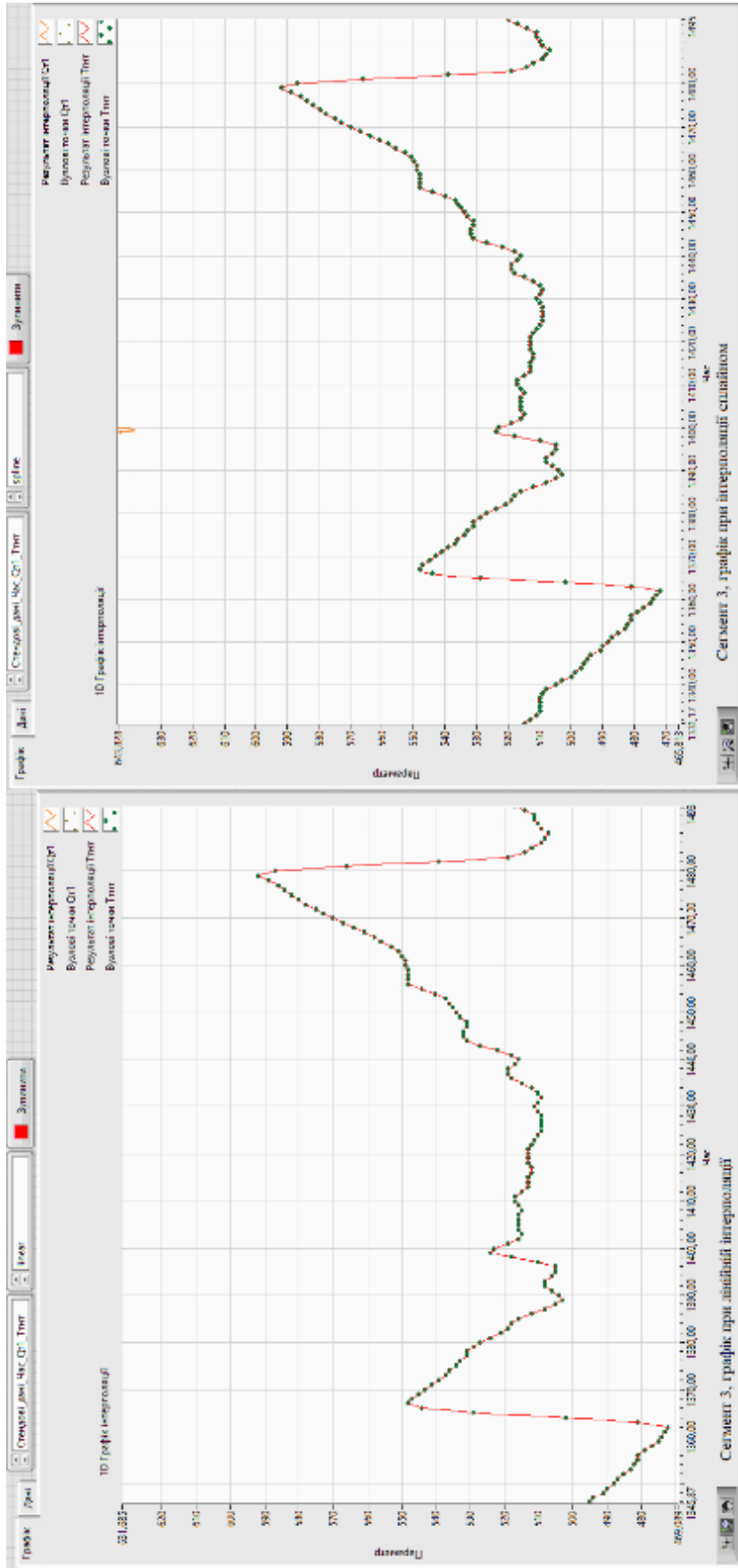


Рисунок 2.23 – Сравнение 2 методов интерполяции на сегменте 3

Проаналізувавши графік параметра Ттнт було зроблено висновок, що пологість графіку після інтерполяції сплайном візуально не помітна. Похибка інтерполяції відсутня.

Далі була проведена інтерполяція методом Ерміта (рис. 2.24).

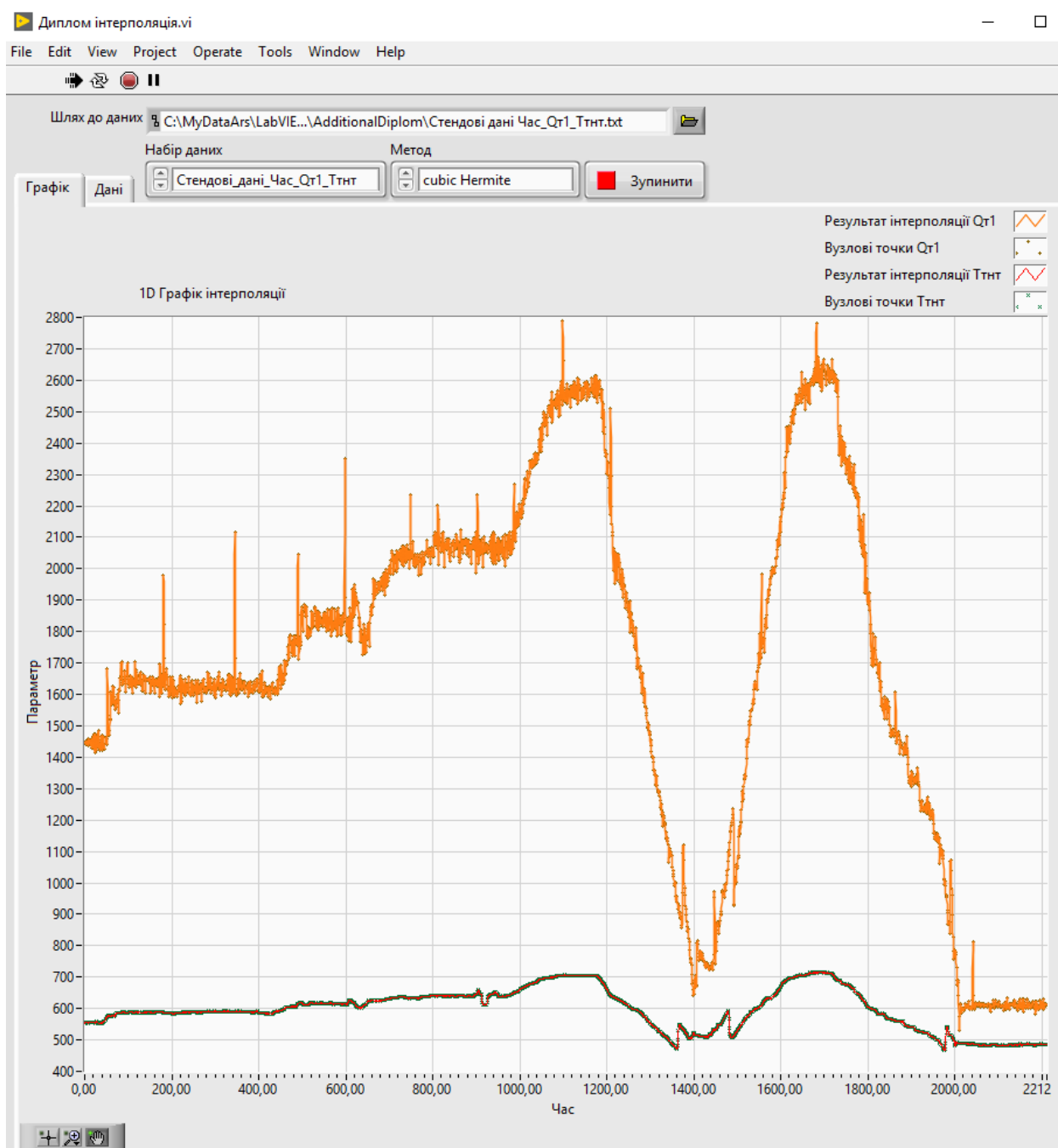


Рисунок 2.24 – Результат Ермітової інтерполяції стендового набору даних

Порівнявши візуально графік лінійної та Ермітової інтерполяції можна зробити висновок, що різниця відсутня. Необхідно з'ясувати, чи є різниця при відображенні графіків у більшому масштабі. Для цього зроблено порівняння 2 методів інтерполяції (лінійного та Ерміту) на графіках у сегменті 2, досліджуваний параметр Q_{t1} (рис. 2.25).

Під час порівняння виявлено, що пологість на графіку Ермітової інтерполяції – відсутня. Проаналізувавши інтерпольовані дані та графіки було зроблено висновок про відсутність похибки інтерполяції. Аналіз параметра T_{int} при Ермітовій інтерполяції порівняно з лінійною, не дав видимих результатів.

Далі була проведена інтерполяція методом Лагранжа (рис. 2.26).

Досліджуючи отриманий графік, було зроблено висновок, що інтерполяція методом Лагранжу не підходить для візуалізації стендового набору даних. Розроблена програма для дослідження не змогла побудувати графік за заданими вузловими точками. Зважаючи на отриманий результат, подальший аналіз цього методу не доцільний. З дослідження методу інтерполяції Лагранжа було зроблено висновок, що цей метод доцільно використовувати у тому випадку, коли відстань між вузловими точками не рівнозначна, в інших випадках, інтерполяція методами сплайну, та Ерміту надають значно кращий результат за менший проміжок часу.

Поліном Лагранжа не завжди доцільно будувати за всіма вузлами, тому що при цьому можуть виникати необґрунтовані «викиди», що призводять до похибки, яка унеможливорює подальшу роботу з даними. В практичних розрахунках часто заздалегідь вибирається ступінь інтерполіруючого полінома. Потім, як нульовий x_0 береться вузол, найближчий зліва до значення аргументу x , для якого потрібно обчислити значення невідомої функції $f(x)$. Інтерполянт n -го ступеня будується за допомогою значень функції y_i у вузлі x_0 та n вузлах, розташованих праворуч.

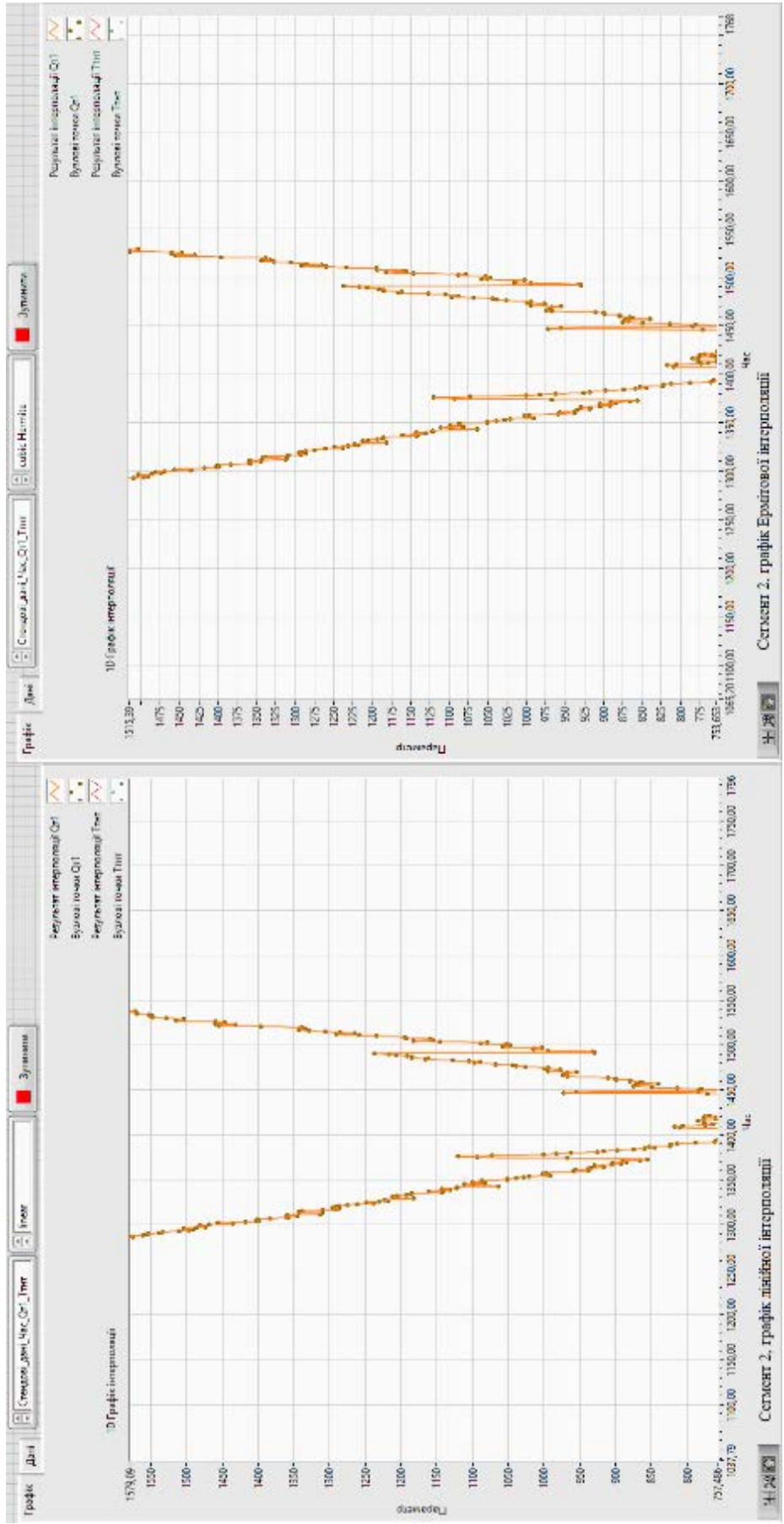


Рисунок 2.25 – Порівняння 2 методів інтерполяції на сегменті 2

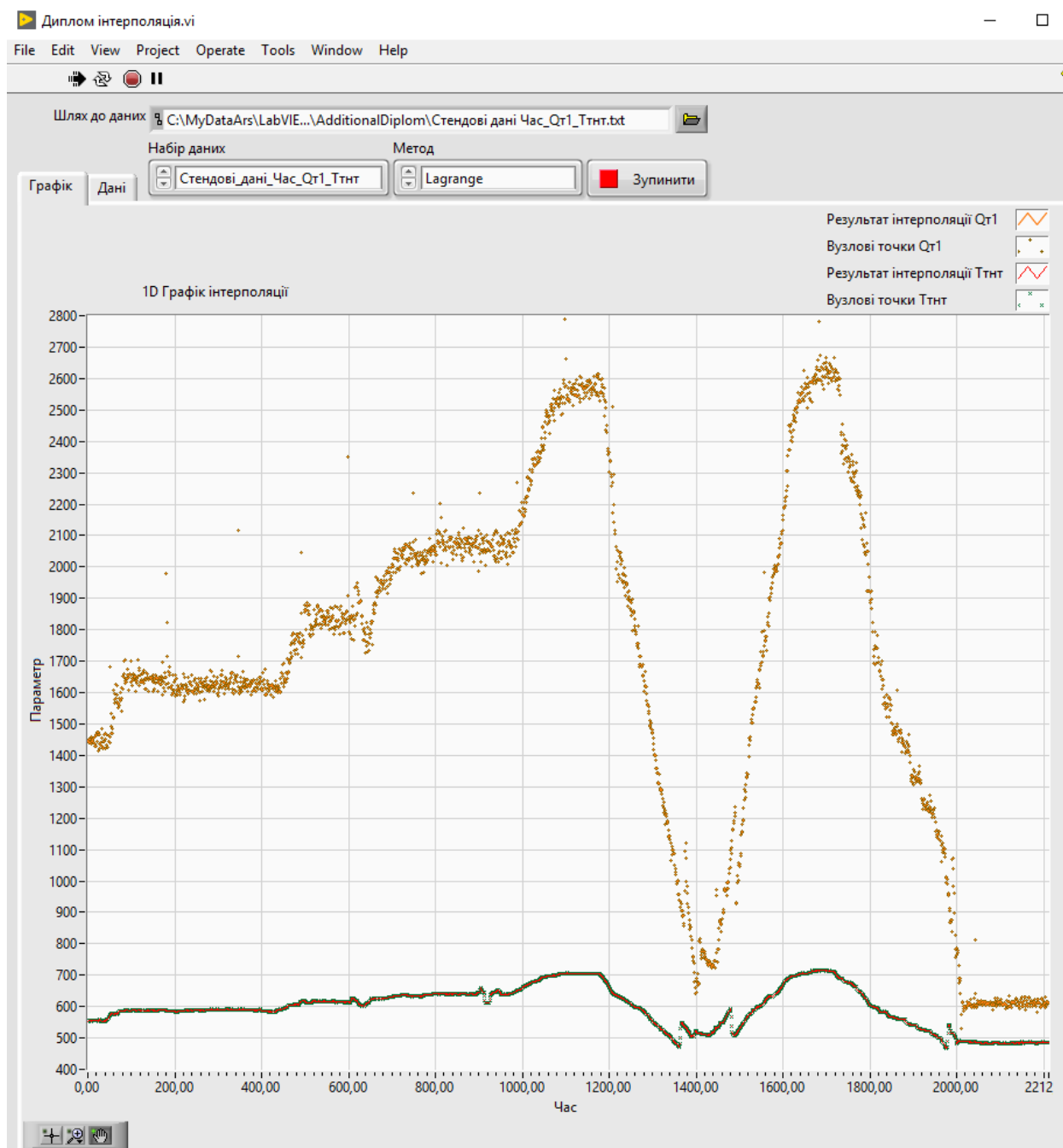


Рисунок 2.26 – Результат інтерполяції Лагранжу стенового набору даних

За результатами 2 частини досліджень була побудована порівняльна таблиця 2.4 для використаних методів інтерполяції. Для оцінки методів використовувалася шкала балів {1;2;3}, де «1» найгірша оцінка, «3» – найкраща.

Таблиця 2.4 – Порівняння методів інтерполяції для набору стендових даних

Критерій оцінки	Метод інтерполяції			
	лінійний	лагранж	сплайн	ерміт
Складність реалізації	3	2	2	1
Забезпечує пологість графіку	1	1	1	1
Демонструє найнижчу похибку	3	1	2	3
Відповідає реальному поведженню параметра	3	1	2	3
Сума балів	10	5	7	8
Сума балів по 2 дослідженням	21	14	20	20

В результаті проведеного дослідження на стендовому наборі даних було встановлено, що метод лінійної інтерполяції виявився кращим за інші при роботі з візуалізацією Big Data, також він виявився найшвидшим в реалізації та роботі з даними.

2.4 Висновки до розділу 2

В результаті проведеного дослідження для вдосконалення візуалізації даних було встановлено:

- чим більша кількість вузлових точок, тим менша похибка інтерполяції;
- для кожної функції та набору даних найкращий метод апроксимації відрізняється, тому необхідно знайти такий метод, що якнайкраще візуалізує функціональну залежність за найменшу кількість часу;
- метод найменших квадратів використовується коли дані експерименту були отримані зі значною похибкою. Графік функції демонструє поведження даних в середньому не проходячи через вузлові точки;

- чим простіша апроксимуюча функція, тим менше часу потрібно для розв'язку завдання апроксимації;
- маючи набір достовірних вузлових точок неможливо однозначно відновити функціональну залежність, отже похибка завжди присутня;
- в мові програмування LabVIEW багато вбудованих інструментів для вирішення задач апроксимації;
- при візуалізації, використання методів інтерполяції має сенс для параметрів з суттєвою дисперсією чи параметрів з низькою кількістю вузлових точок;
- інтерполяція методом сплайну забезпечує найкращу плавність графіків порівняно з іншими методами;
- для рішення завдання візуалізації Big Data найкраще підходить метод лінійної інтерполяції.

Результати дослідження є достовірними та відповідають висновкам з джерел дослідження [20]-[22].

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог до тестового програмного забезпечення для проведення випробувань на стендових системах реального часу

На основі проведеного дослідження в пункті 1.3.6 вдалося сформулювати вимоги до розроблюваного програмного забезпечення:

- можливість зчитування та збереження даних випробування;
- можливість візуалізації вимірюваних параметрів;
- можливість проведення обробки записаної бази даних за методами апроксимації;
- можливість усунення похибок датчиків на графіку під час проведення випробувань за рахунок усереднення параметрів.

3.2 Вибір середовища розробки

Для вибору середовища розробки було проведено порівняння двох програм LabVIEW та MatLab. Порівняння проводилося за таким планом:

- вступ;
- аналіз інтерфейсу користувача;
- аналіз програмних особливостей;
- переваги середовища розробки;
- недоліки середовища розробки.

3.2.1 Аналіз можливостей середовища розробки MatLab

MatLab був розроблений наприкінці 1970-х років Клівом Моулером, що обіймає посаду декана факультету комп'ютерних наук в Університеті Нью-Мексико, з метою позбавити студентів факультету від необхідності вивчення мови програмування Фортран та дати можливість використовувати програмні

бібліотеки Linpack та EISPACK. Незабаром мова поширилася серед інших університетів і дуже зацікавила вчених, працюючих у галузі прикладної математики. Вже тоді MatLab також широко використовувався у сфері освіти для викладання лінійної алгебри та чисельних методів [23].

MatLab – високорівнева мова програмування та одна з найстаріших, ретельно опрацьованих та перевірених часом систем автоматизації математичних розрахунків, яка побудована на розширеному поданні та застосування матричних операцій. Принципи та функціональні можливості програми з роботи з матрицями творці мови висловили у гасла «думай векторно» [23].

Інтерфейс користувача складається з наступних вікон (рис. 3.1) [23]:

- робочий стіл – основне вікно програми, містить рядок заголовка (назва програми), рядок меню, панель інструментів та внутрішні вікна;
- командне вікно – вікно введення команд, виведення результатів обчислень;
- історія команд – вікно хронологічного запису команд;
- відкрита папка – робочий каталог;
- робоча область – вікно відображення даних та змінних;
- панель запуску – вікно запуску програм.

Мова MatLab є високорівневою мовою програмування, що інтерпретується, включає засновані на матрицях структури даних, широкий спектр функцій, інтегроване середовище розробки, об'єктно орієнтовані можливості та інтерфейси до програм, написаних іншими мовами програмування [23].

Програми, створені за допомогою пакета MatLab, бувають двох типів – скрипти та функції. Функції мають вхідні та вихідні аргументи, власний робочий простір для зберігання проміжних результатів обчислень та змінних. Скрипти використовують загальне робоче простір.

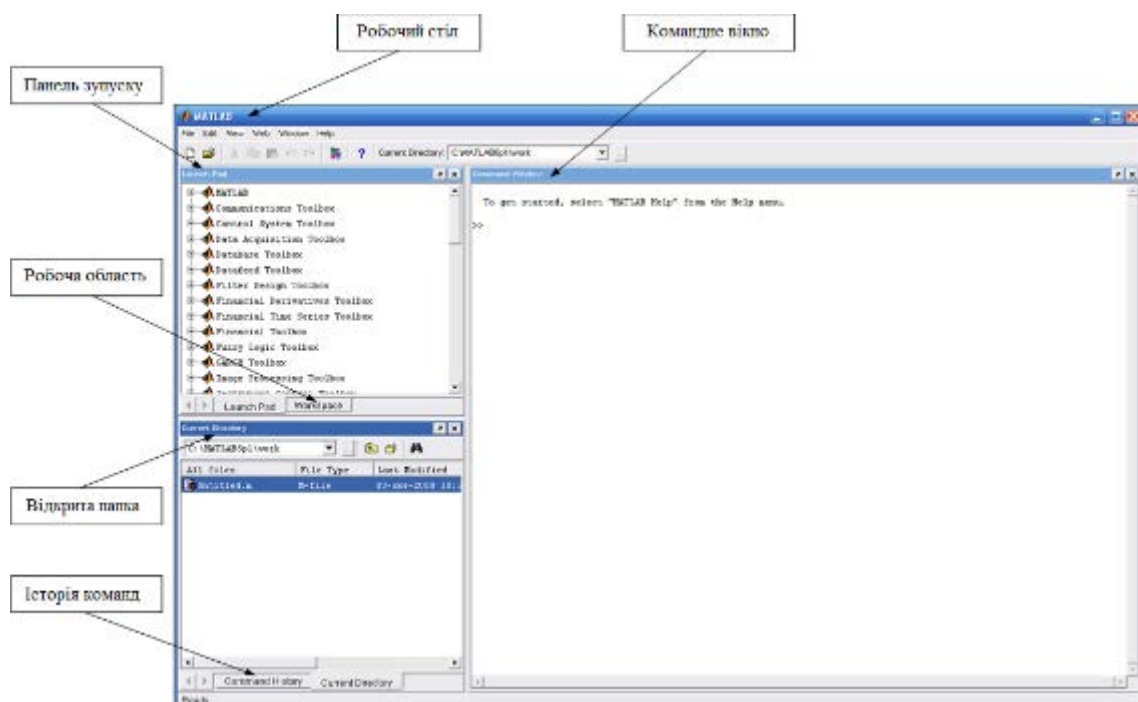


Рисунок 3.1 – Інтерфейс MatLab [23]

Як і скрипти, так і функції не компілюються в машинний код і зберігаються в вигляді текстових файлів. Існує також можливість зберігати pre-parsed програми – функції та скрипти, оброблені у вигляді зручному для машинного виконання. В загальному випадку такі програми виконуються набагато швидше звичайних [23].

Незважаючи на те, що спочатку MatLab призначалася виключно для обчислень, на додаток до вдосконалених обчислювальних було придбано ядро символічних перетворень, а також з'явилися бібліотеки, що дозволили MatLab розширити галузі застосування та забезпечили унікальні для математичних пакетів функції [23].

Для MatLab є можливість створювати спеціальні набори інструментів, що розширюють його функціональність. Набори інструментів являють собою колекції функцій, написаних на мові MatLab для розв'язання спеціалізованих завдань. Компанія Mathworks постачає набори інструментів, які використовуються в багатьох областях [23]:

- цифрове оброблення сигналів, зображень та даних: DSP Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communication Toolbox, Filter Design Toolbox – набори функцій, що дозволяють вирішувати широкий спектр задач обробки сигналів, зображень, проектування цифрових фільтрів та систем зв'язку;
- системи управління: Control Systems Toolbox, μ -Analysis and Synthesis Toolbox, Robust Control Toolbox, System Identification Toolbox, LMI Control Toolbox, Model Predictive Control Toolbox, Model-Based Calibration Toolbox – набори функцій, що полегшують аналіз та синтез динамічних систем, проектування, моделювання та ідентифікацію систем управління, включаючи сучасні алгоритми управління [23];
- фінансовий аналіз: GARCH Toolbox, Fixed-Income Toolbox, Financial Time Series Toolbox, Financial Derivatives Toolbox, Financial Toolbox, Datafeed Toolbox – набори функцій, що дозволяють швидко та ефективно збирати, обробляти та передавати різну фінансову інформацію;
- аналіз та синтез географічних карт, включаючи тривимірні: Mapping Toolbox [23];
- збір та аналіз експериментальних даних: Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, Link for Code Composer Studio – набори функцій, що дозволяють зберігати та обробляти дані, отримані в ході експериментів, у тому числі в реальному часі. Підтримується інтеграція широкого спектра наукового та інженерного вимірювального обладнання;
- набір інструментів для візуалізації та представлення даних: Virtual Reality Toolbox – дозволяє створювати інтерактивні світи та візуалізувати наукову інформацію з за допомогою технологій віртуальної реальності та мови VRML [23];
- засоби розробки: MatLab Builder for COM, MatLab Builder for Excel, MatLab Builder for NET, MatLab Compiler, Filter Design HDL Coder –

набори функцій, які дозволяють створювати незалежні програми з середовища MatLab [23];

- взаємодія із зовнішнім програмним забезпеченням: MatLab Report Generator, Excel Link, Database Toolbox, MatLab Web Server, Link for ModelSim – це набори функцій, що дозволяють зберігати дані в різних видах таким чином, щоб інші програми могли з ними працювати [23];

- інструменти роботи з базами даних: Database Toolbox;

- наукові та математичні пакети: Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed-Point Toolbox, Fuzzy Logic Toolbox, Genetic Algorithm and Direct Search Toolbox, OPC Toolbox, Optimization Toolbox, Partial Differential Equation Toolbox, Spline Toolbox, Statistic Toolbox, RF Toolbox – набори спеціалізованих математичних функцій, які дозволяють вирішувати широкий спектр наукових та інженерних задач, що включають розробку генетичних алгоритмів, розв'язання задач у похідних, комплексні проблеми, оптимізацію систем та інші [23];

- інструменти для синтезу та аналізу нейронних мереж: Neural Network Toolbox;

- інструменти для побудови та аналізу нечітких множин: Fuzzy Logic Toolbox [23];

- бібліотека символьних обчислень: Symbolic Math Toolbox – інструменти для символьних обчислень із можливістю взаємодії із символьним процесором програми Maple.

Перерахуємо сумісні з MatLab програми на рівні мови програмування [23]:

- GNU Octave;
- FreeMat;
- Maxima;
- Scilab;
- таблиці Excel;

– Maple.

Вартість ліцензії MatLab для комерційних організацій складає близько \$ 2880, бібліотеки візуального моделювання Simulink - \$4220, необхідні бібліотеки докуповуються окремо (наприклад, вартість бібліотеки Real-Time WindowsTarget для модуля Simulink складає \$3000) [23].

До складу MatLab включається бібліотека візуального програмування Simulink, що дозволяє побудувати логічну схему системи управління, використовуючи лише стандартні блоки. Після конструювання схеми можна детально проаналізувати її роботу. Сигнали, значення параметрів та атрибути схеми вказуються або безпосередньо на самій схемі, або введені з текстової консолі. Використання готових моделей дозволяє керувати набором даних та швидко перепрофілювати моделі. MatLab працює з різними типами даних: цілими, речовими, з фіксованою комою, матрицями, векторами, комплексними, а також працює з типами даних користувача [23].

MatLab надає широкі можливості для програмування. Бібліотека C Math є об'єктною та має понад 300 процедур обробки даних мовою C. При роботі з програмним продуктом можна використовувати як процедури самої MatLab, так і стандартні процедури мови C, використовуючи компілятор C Math, можна вбудовувати будь-які процедури MatLab у готові програми.

Бібліотеки MatLab відрізняються високою швидкістю чисельних обчислень. Матриці є основою автоматичного рішення та складання рівнянь стану динамічних об'єктів та систем. MatLab один з найбільш потужних універсальних інтегрованих пакетів комп'ютерної математики [23].

Компілятор підтримує велику кількість команд і пакетів MatLab. Компіляція кодів у MatLab вимагає хорошої підготовки програміста та глибокого знання принципів компіляції.

Компілятор MatLab вимагає наявності хоча б одного з наступних C/C++ компіляторів [24]:

– Lcc C, є лише C-компілятором;

- Borland C++ версії 5.3, 5.4, 5.5, 5.6;
- Microsoft Visual C/C++ версії 6.0, 7.0, 7.1.

MatLab – інженерний пакет високого рівня, ретельно налагоджений, широко використовуваний у багатьох сферах наукових досліджень, інженерних розрахунках та проектуванні систем автоматичного управління. Основним недоліком цієї програми є вартість ліцензії: набір платформи плюс бібліотека Simulink обійдеться комерційній організації приблизно \$7000 [24].

Пакет є пропрієтарним (комерційним) продуктом і не має відкритого програмного коду. Так навіть скомпільовані користувацькі програми зашифровані унікальним 1024-бітним ключем, таким чином, ці скомпільовані програми мовою Сі не можуть редагуватись [24].

Система MatLab не має розвинених засобів створення інтерфейсу користувача, якими володіють системи візуального програмування Microsoft Visual C++, Borland C++Builder та інш., тому створювана системою MatLab автономна програма не відповідає вимогам професійного програмування [24].

Система MatLab використовується в техніці, широко використовується в науці та освіті. Надає можливості для аналізу даних, що охоплюють практично всі області математики (матриці та лінійна алгебра, багаточлени та інтерполяція, математична статистика та аналіз, обробка даних, диференціальні рівняння, розріджені матриці, ціла арифметика). А також система MatLab застосовується для розробки алгоритмів, надаючи зручні засоби для розробки алгоритмів, що включають високо рівневі з використанням концепцій об'єктно-орієнтованого програмування, є необхідні засоби інтегрованого середовища розробки, включаючи налагоджувач та профайлер. Є функції для роботи з цілими типами даних, які полегшують створення алгоритмів для мікроконтролерів та інших програм, де це необхідно [24].

Цей інженерний пакет надає великі можливості візуалізації даних. До складу пакета входить достатня кількість функцій для побудови графіків, у

тому числі тривимірних, візуального аналізу даних та створення анімованих роликів.

Вбудоване середовище розробки надає можливість створювати користувацькі графічні інтерфейси з різними елементами управління, такими як кнопки, поля введення та іншими. За допомогою компонента MatLab-Compiler ці графічні інтерфейси можуть бути перетворені на самостійні програми [24].

Програма MatLab є однією з самих об'ємних програм комп'ютерної математики, що увібрала досвід, методи та правила математичних обчислень, накопичені за тисячі років розвитку математики. Додається до системи велика документація яку цілком можна розглядати як фундаментальний багатотомний електронний довідник по математичному забезпеченню.

До недоліків програми можна віднести [24]:

- система MatLab не має розвинених засобів створення інтерфейсу користувача, тому створювана системою MatLab автономна програма не відповідає вимогам професійного програмування;
- вартість MatLab включаючи Simulink більше 7000 долларів;
- вимагає наявності C – компілятора на ПК в додаток до вбудованого в MatLab;
- вузьконаправлена програма, що відмінно виконую завдання візуалізації та розрахунків даних, проте, поза межами цих областей немає розроблених бібліотек.

3.2.2 Аналіз можливостей середовища розробки LabVIEW

Лабораторія Virtual Instrument Engineering Workbench (LabVIEW) – це платформа для системного проектування та середовище розробки для мови програмування National Instruments. Графічна мова має назву «G». LabVIEW зазвичай використовується для збору даних, управління приладами та

промисловою автоматизацією в різноманітних операційних системах (ОС), включаючи Microsoft Windows, також різноманітні версії Unix, Linux, та MacOS [25].

Графічне середовище програмування LabVIEW інтегрує створення користувальницьких інтерфейсів (що називають лицьовими панелями) у цикл розробки. Програми-підпрограми LabVIEW називаються віртуальними інструментами (VI). Кожен VI складається з трьох компонентів: блок-схеми (рис. 3.2), лицьової панелі (рис. 3.3) та панелі підключень (рис. 3.4). Остання використовується для уявлення VI на блок-схемах інших викликаючих VI. Лицьова панель побудована з використанням елементів керування та індикаторів. Елементи керування – це входи, вони дозволяють користувачу вводити інформацію в VI. Індикатори це виходи – вони вказують або відображають результати на основі вхідних даних, що були передані в VI. Блок-схема містить графічний початковий код. Всі об'єкти що розміщені на лицьовій панелі, з'являються на блок-схемі як термінали. Блок-схема також містить структури та функції, які виконують операції з елементами керування та надають дані для індикаторів [25].

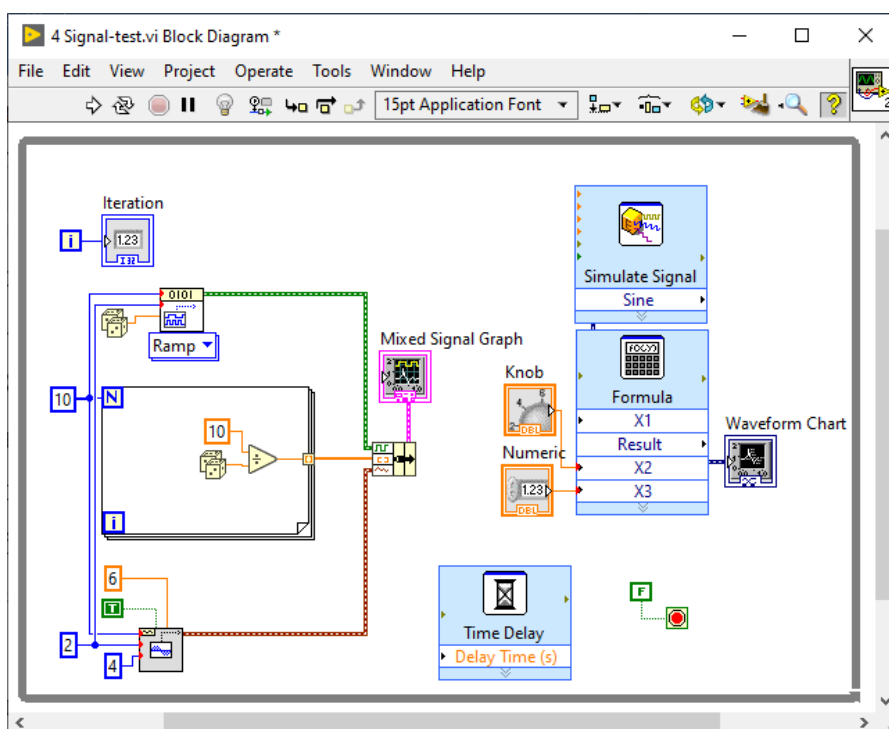


Рисунок 3.2 – Блок схема LabVIEW

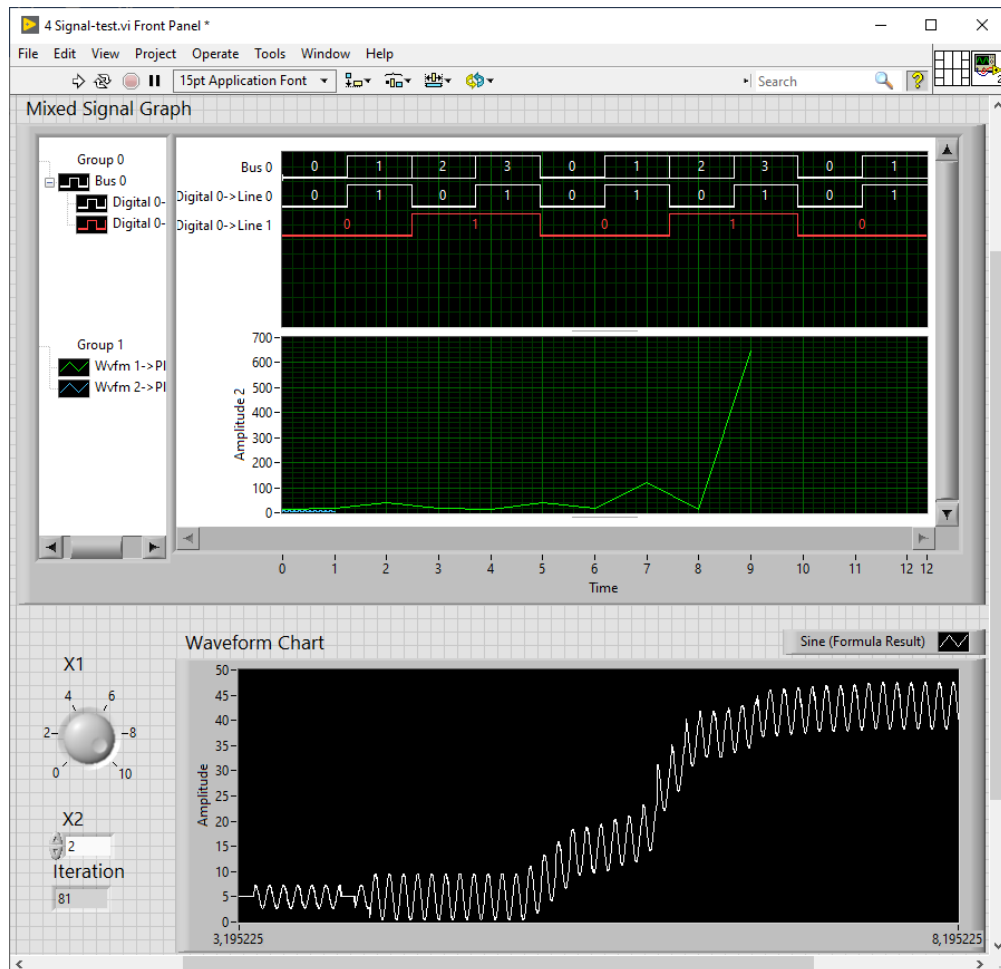


Рисунок 3.3 – Лицьова панель LabVIEW

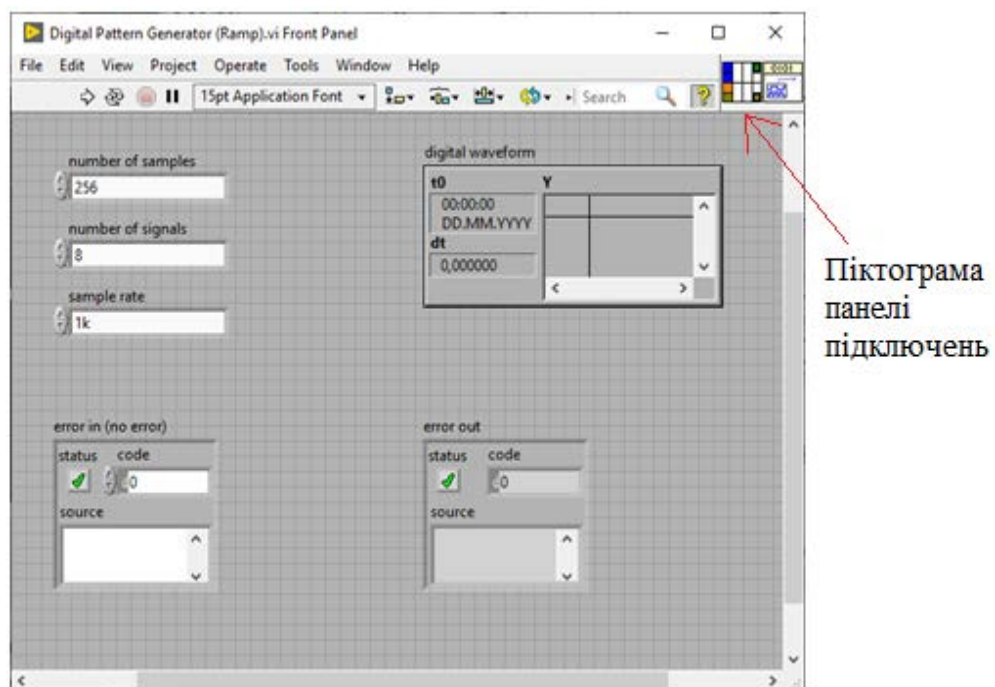


Рисунок 3.4 – Панель підключень LabVIEW

Структури та функції знаходяться на палітри функцій та можуть бути розміщені на блок-схемі. В сукупності елементи керування, індикатори, структури та функції називаються вузлами. Вузли з'єднуються між собою за допомогою дротів, наприклад, декілька елементів управління та індикатор можуть бути підключені до функції складання, для того, щоб індикатор відображав суму цих елементів управління. Таким чином, віртуальний інструмент може бути запущений як програма, з лицьовою панеллю, що слугує користувальницьким інтерфейсом, або при розміщенні в якості вузла на блок-діаграму, лицьова панель визначає входи та виходи для вузла через панель підключень. Це означає, що кожен VI можливо легко протестувати перед тим, як вбудувати його в якості підпрограми в більш об'ємну програму [25].

LabVIEW – це компілятор. Компіляція виконується у момент розробки, програма завжди готова до запуску, що значно прискорює налагодження програм. Графічний спосіб розробки також дозволяє програмістам створювати програми шляхом перетаскування віртуальних представлень лабораторного обладнання, з яким вони вже працювали [25].

Середовище програмування LabVIEW з наявними в великій кількості прикладами та технічною документацією спрощує створення невеликих додатків. Для складних алгоритмів або великого за об'ємом коду важливо, щоб програміст мав знання про спеціальний синтаксис LabVIEW та топологію управління пам'яттю. Найновіші системи розробки LabVIEW надають можливість створювати автономні додатки. Крім того, можливо створювати розподілені додатки, що обмінюються даними по моделі клієнт-сервер, таким чином їх легше реалізувати через паралельну природу мови G [26].

LabVIEW — універсальна мова програмування. Незважаючи на те, що цей продукт тісно пов'язаний з апаратним забезпеченням National Instruments, він не пов'язаний з конкретним персональним комп'ютером (ПК). Початкові тексти можливо переносити між підтримуваними системами, а програми будуть виглядати однаково на будь якій з них. Код, згенерований у LabVIEW

також може бути виконаний на Windows Mobile. Ця мова може бути з успіхом застосована для створення великих систем, для обробки текстів, зображень та роботи з базами даних [26].

LabVIEW — високорівнева мова з підтримкою включень низькорівневих модулів до програм, навіть, якщо ви хочете використовувати вставки коду на асемблері. Це можливо реалізувати за допомогою створення динамічно підключаємої бібліотеки (DLL) та виклику її з коду. Високорівнева мова дозволяє легко виконувати складні операції с даними, які в звичайній мові реалізуються за допомогою десятків рядків коду. Проте, деякі операції низькорівневих мов програмування, наприклад роботу з показниками, важко реалізувати у LabVIEW через його високорівневисть. LabVIEW включає основні конструкції управління, що мають аналоги в «традиційних» мовах [26]:

- змінні (локальні та глобальні);
- розгалуження (case structure);
- For – цикли з перевіркою завершення та без неї;
- While – цикли;
- групування операцій.

Віртуальні інструменти зберігаються в файлах з розширенням «.vi». VI – це так звані цеглинки з яких будується програма. Будь яка LabVIEW програма містить як мінімум один VI. Можливо провести аналогію з функцією в термінах мови «C», але в LabVIEW одна функція знаходиться в одному файлі. Також можливо створювати бібліотеки інструментів. Один VI може бути викликаний з іншого віртуального інструменту [26]. Так виглядає класичний приклад Hello, World! (рис. 3.5).

В основі LabVIEW лежить парадигма потоку даних. В прикладі константа та термінал індикатора з'єднані між собою лінією. Ця лінія має назву Wire, що перекладається як дріт.

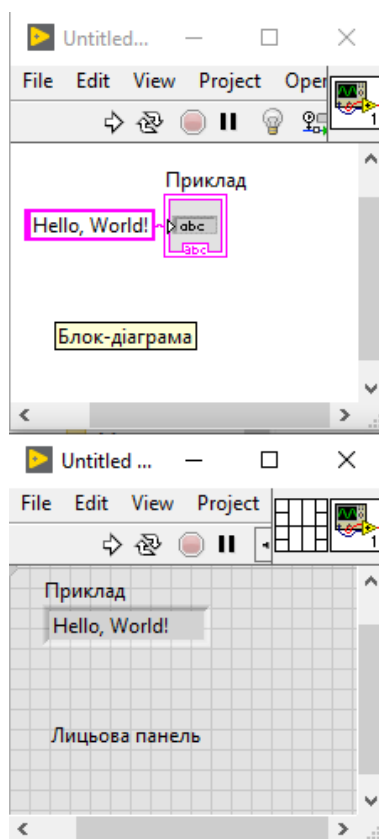


Рисунок 3.5 – Класичний приклад

По дротам передаються дані від одних елементів до інших. Ця концепція називається Data Flow. Сутність блок-діаграми – вузли (ноди), виходи одних вузлів підключені до входів інших вузлів. Вузли починають виконання тільки тоді, коли вони отримують всі необхідні для роботи дані. У прикладі (рис. 3.5) дві ноди. Одна з них константа. Цей вузол самодостатній – він починає виконуватися одразу. Другий вузол – індикатор, він відображає дані, що передає константа [26].

Далі наведено приклад складання та множення двох чисел. У традиційних мовах програмування ці операції реалізуються приблизно таким чином [26]:

- `int a, b, sum, mul;`
- `//...;`
- `sum = a + b;`
- `mul = a * b.`

Так операції виглядають у LabVIEW (рис. 3.6).

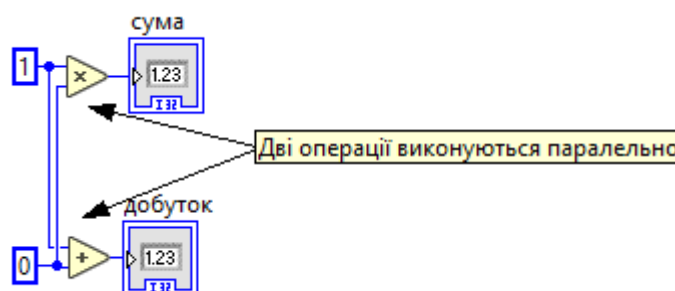


Рисунок 3.6 – Операції складання та множення

Додавання та множення автоматично виконуються паралельно. На двох процесорному ПК будуть автоматично задіяні обидва процесора. У розпорядженні розробника великий вибір інструментів для синхронізації потоків – семафори, черги, рандеву та інші [26].

Стандартна поставка LabVIEW включає в себе блоки для роботи з іні файлами, реєстром, функції для роботи з двійковими та текстовими файлами, математичні функції, інструменти для побудови графіків, інструменти для роботи с DLL, LabVIEW дозволяє працювати з ActiveX компонентами та .net. Починаючи з восьмої версії у LabVIEW з'явилася підтримка класів – мова стала об'єктно орієнтованою. Підтримка не є повною, але головні ознаки – успадкування та поліморфізм присутні. Функціональність мови можливо розширити додатковими модулями, наприклад NI Vision Toolkit – для обробки зображень і машинного зору та інші. За допомогою модуля Application Builder можливо згенерувати виконуваний exe-файл, проте для його виконання знадобиться LabVIEW Run-Time, додаток знаходиться у вільному доступі. За допомогою Internet Toolkit можливо працювати з серверами, з Database Connectivity Toolkit – з базами даних [26].

Розробка на LabVIEW пропонує прискорений підхід до графічного програмування, допомагає візуалізувати кожен аспект розроблюваного додатку, включаючи конфігурацію обладнання, дані вимірів та налагодження.

Така візуалізація дає змогу легко інтегрувати вимірювальне обладнання від будь-якого постачальника, представляти складну логіку на діаграмі, розробляти алгоритми аналізу даних та проектувати користувацькі інженерні інтерфейси [26].

На сьогодні LabVIEW використовується для наступних задач [27]:

- вимірювання фізичних систем за допомогою датчиків або виконуваних механізмів. На основі LabVIEW можливо створити індивідуальні вимірювальні рішення для візуалізації та аналізу реальних сигналів для прийняття рішень на основі даних [27];

- підтвердити або перевірити електронні конструкції. Використовуючи LabVIEW та NI або стороннє обладнання, можливо автоматизувати валідацію продукту у відповідності до жорстких вимог до терміну виходу на ринок та до продуктивності;

- розробка виробничих тестових систем. При роботі з LabVIEW, можливо створити гнучкі тестові додатки, що можуть керувати декількома приладами, та розроблювати користувацькі інтерфейси для оптимізації продуктивності виробничих тестів та експлуатаційних затрат [27];

- створення інтелектуальних машин або промислового обладнання. Платформа для розробки вбудованих систем NI включає в себе повний стек програмного забезпечення, інтегрованого та налаштованого обладнання та екосистему користувачів і IP;

- навчання студентів інженерних спеціальностей. В класі чи в лабораторії можливо навчати студентів за допомогою реальних систем, використовуючи LabVIEW, для досягнення кращих результатів та успішності [26].

Переваги LabVIEW в вимірюванні даних з фізичних систем з датчиками або виконуваними механізмами [27]:

- миттєве підключення до вимірювального обладнання. LabVIEW скорочує час розробки за рахунок візуального представлення обладнання.

Інтеграція сторонніх пристроїв до свого додатку прискорюється за рахунок великої бібліотеки драйверів та програмного інтерфейсу додатку (Application Programming Interface – API), прикладів та документації [27];

- графічний підхід. LabVIEW був розроблений спеціально для інженерів та вчених, за рахунок графічного підходу легше відображати свої думки у функціональний код, для інтуїтивної розробки додатків;

- миттєва аналітика. В LabVIEW вбудовано більше 1000 алгоритмів для візуалізації та аналізу сигналів, також існує можливість вбудовувати код із іншого програмного забезпечення, що дозволяє швидко переходити до аналізу нових даних;

- автоматизація збереження та звітів. LabVIEW пропонує функції управління файлами для збереження та обміну даними у стандартних форматах файлів, таких як двійковий, текстовий та табличний [27].

Недоліки LabVIEW перелічені нижче [28]:

а) ціна:

- 1) LabVIEW Base \$ 521.00 на рік;
- 2) LabVIEW Full \$ 4,105.00 на рік;
- 3) LabVIEW Professional \$ 6,845.00;
- 4) LabVIEW Community Edition безоплатна, проте з водяними

знаками для некомерційного використання;

б) не стандартизована мова. LabVIEW – запатентований продукт National Instruments. На відміну від поширених мов програмування, таких як C або Python, LabVIEW не керується та не визначається сторонніми комітетами по стандартам, такими як Американський національний інститут стандартів, Інститут інженерів електротехніки та електроніки, Міжнародна організація по стандартизації [28];

в) не текстова мова. Так як мова G не є текстовою, програмні інструменти, такі як управління версіями, паралельне порівняння та

відстеження змін коду версії, не можуть бути реалізовані так само, як для текстових мов програмування;

г) не підходить для простих програм. За допомогою модуля Application Builder можливо згенерувати виконуваний exe-файл що запуститься під будь-якою операційною системою, проте для його виконання знадобиться LabVIEW Run-Time, що важить 2-3 Гб. Це означає, що для надсучасної системи збору усіх можливих даних та для прикладу «Hello, World» знадобиться LabVIEW Run-Time. Коли проєкт великий, важить 10 Гб та більше, додаткові 3 Гб не дуже суттєві, проте, коли програма проста та важить декілька Кб, встановлення модуля виявляється непомірною витратою вільного місця [28].

З порівняння середовищ програмування LabVIEW та MatLab можна зробити висновок, що обидві програми чудово підходять для досліджень та розробки методів та засобів візуалізації для підвищення якості представлення вимірюваних параметрів, на системах стендових випробувань авіаційних двигунів. Обидві, мови програмування були наявні на виробництві. Вибір пав на LabVIEW, через те, що на ньому вже розроблялася система для проведення стендових випробувань Ostorus, також LabVIEW має більш розвинені засоби створення інтерфейсу користувача.

3.3 Розробка програмного забезпечення

Алгоритм за яким розроблювалося тестове програмне забезпечення для проведення випробувань на стендових системах реального часу зображено на рисунку 3.7.

Блок 1 — встановлення програмного забезпечення LabVIEW.

Блок 2 – перевірка працездатності прототипу підсистеми вимірювання температури та вологості, підключення підсистеми до комп'ютеру та налаштування передачі даних по COM порту.

Блок 3 – розробка програмного коду для читання даних з СОМ порту, реалізація ділення пакету даних на складові (значення вимірюваних параметрів).

Блок 4 – розробка програмного коду для представлення отриманих даних на графіках, реалізація функції очистки графіку, алгоритму усереднення вимірюваних параметрів температури та вологості та розробка інтерфейсу користувача на лицьовій панелі головного вікна.

Блок 5 – розробка програмного коду для створення бази даних, запису сирих вимірюваних даних до бази та збереження змін у базі даних.

Блок 6 – перевірка працездатності функцій програмного забезпечення та кастомізація інтерфейсу користувача на лицьовій панелі.

Блок 7 – розробка програмного коду для читання записаної раніше бази даних та представлення даних у зручному вигляді для алгоритмів.

Блок 8 – розробка програмного коду для розрахунку отриманих даних за методами апроксимації даних.

Блок 9 – розробка програмного коду для відображення графіків апроксимованих значень, відображення вузлових точок за якими виконувалась апроксимація, відображення даних отриманих під час роботи алгоритмів.

Блок 10 – перевірка працездатності функцій читання бази даних, розрахунку даних за методами апроксимації, візуалізації даних у вікні пост-обробки програмного забезпечення та кастомізація інтерфейсу користувача на лицьовій панелі.

Блок 11 – перевірка всіх функцій тестового програмного забезпечення для проведення випробувань на стендових системах реального часу.

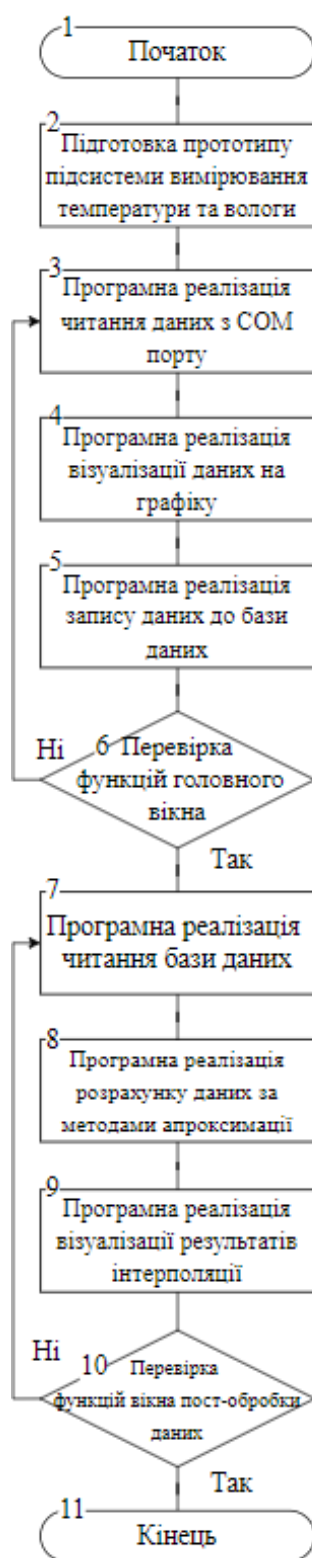


Рисунок 3.7 – Алгоритм розробки тестового програмного забезпечення

Далі наведено UML діаграму взаємодії стендових служб для проведення випробувань авіаційних двигунів (рис. 3.8).

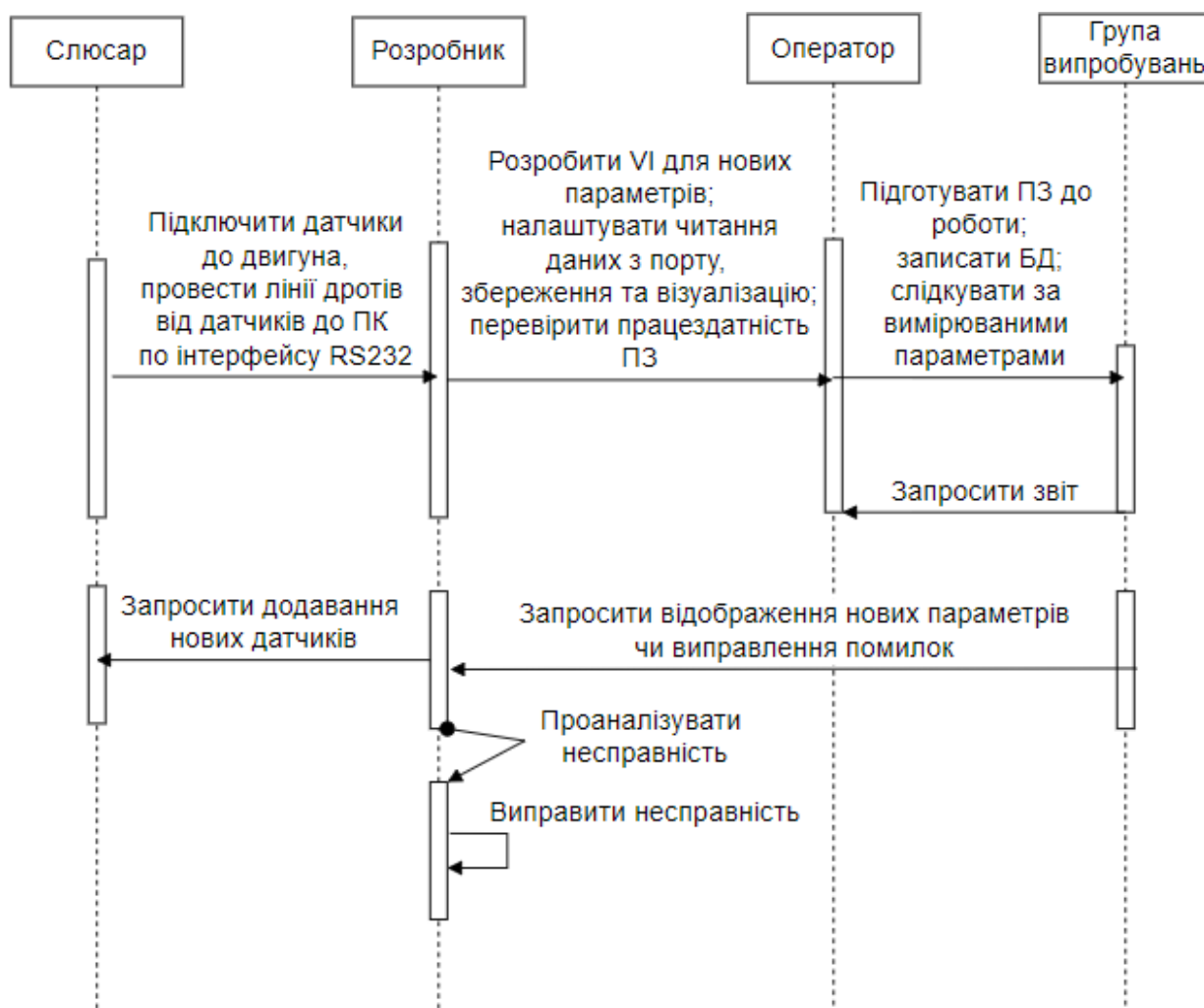


Рисунок 3.8 – Діаграма взаємодії стендових служб

3.4 Архітектура тестової системи збору даних

Перед тим як модифікувати існуюче програмне забезпечення для проведення випробувань на стендових системах реального часу – Ostopus, необхідно довести ефективність досліджених методів апроксимації у тестовій системі збору даних реалізувавши у ній вимоги до програмного забезпечення перелічені в підрозділі 3.1. Після підтвердження їх ефективності, інженер – розробник програмного забезпечення Ostopus додасть функції з методами апроксимації у програмний код проєкту. Архітектура тестового програмного забезпечення зображена на рисунку 3.9.

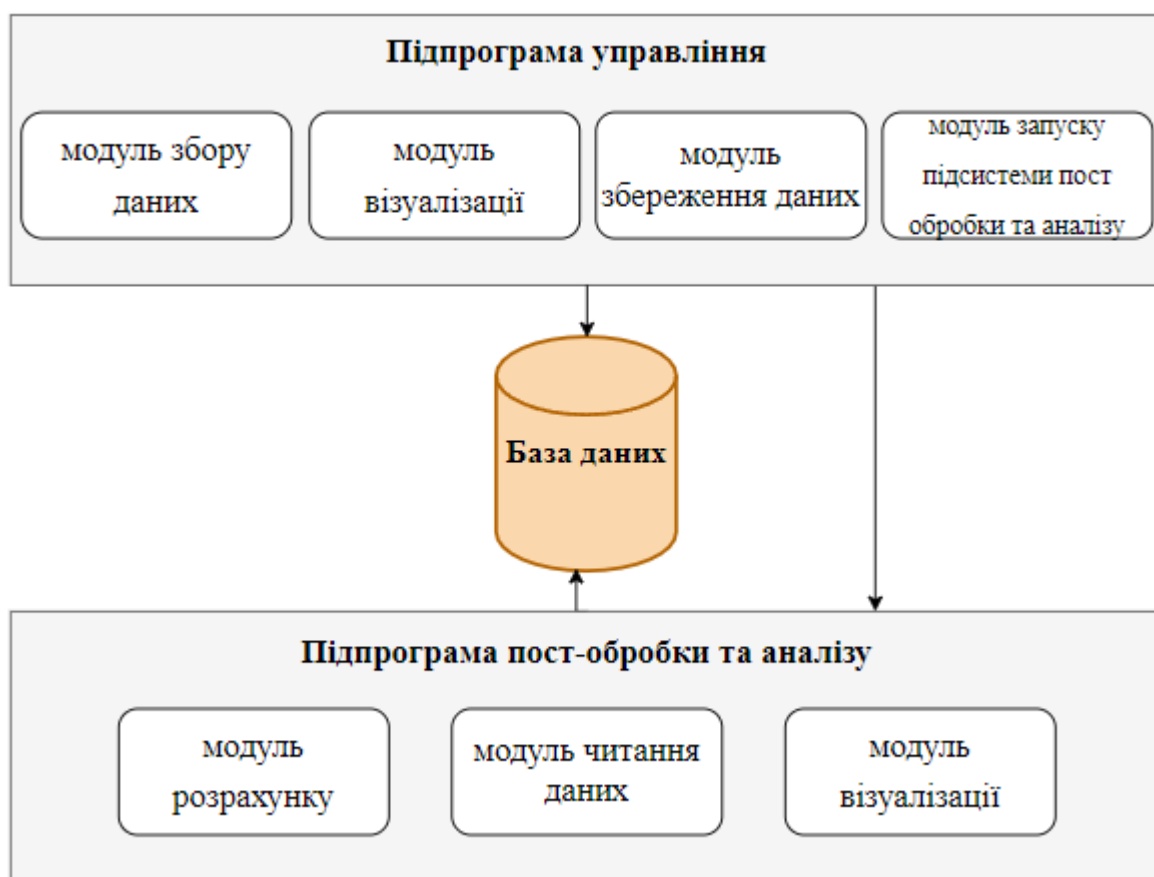


Рисунок 3.9 – Архітектура тестового ПЗ

Тестова підпрограми управління використовувалась у пункті 2.3.1 під час збору даних для дослідження з прототипу підсистеми вимірювання температури та вологості. В її функції входить налаштування з'єднання з COM портом, читання даних з порту, контроль запису бази даних, взаємодія з розробником через діалогові вікна з описом помилок при виконанні проєкту, зупинка читання даних з порту та звільнення оперативної пам'яті на комп'ютері.

Тестова підпрограма управління складається з наступних модулів:

- модуль збору даних. Виконує налаштування COM порту, читання буферу вводу-виводу, конвертує зчитані строкові дані у целочисельний тип;
- модуль візуалізації. Виконує об'єднання сирих та усереднених даних у масив перед виводом на графік, візуалізує дані, має функцію очистки

графіку для спрощення процесу тестування тестової системи збору даних та функцію усереднення параметрів;

- модуль збереження даних. Створює файл бази даних з всіма необхідними полями, виділяє дисковий простір під базу даних, записує зчитані дані до файлу;
- модуль запуску підпрограми пост-обробки та аналізу. Налаштовує з'єднання з залежною підпрограмою, відкриває, запускає та конфігурує VI.

База даних це файл з розширенням .tdms, що являє собою книгу Excel з фіксованою кількістю сторінок до якої записуються дані за допомогою модуля збереження даних у реальному часі.

Тестова підпрограма пост-обробки та аналізу використовувалася у підрозділі 2.3. В її функції входить читання бази даних записаної підпрограмою управління, чи бази даних записаної за допомогою програмного забезпечення для проведення випробувань на стендових системах реального часу – Octavo. Також ця підпрограма відображає результати розрахунків на розробленій лицьовій панелі.

Тестова підпрограма пост-обробки та аналізу складається з наступних складових:

- модуль розрахунку. перетворює дані у необхідний формат для подальшого проведення апроксимації;
- модуль читання даних. Відкриває записану базу даних, зчитує з неї дані, закриває базу;
- модуль візуалізації. Виконує об'єднання та перетворення даних у необхідний формат для візуалізації.

3.5 Висновки до розділу 3

В 3 розділі було виконано:

- а) проаналізовані вимоги до тестового програмного забезпечення;

б) проаналізовані середовища розробки MatLab та LabVIEW за таким планом:

- 1) вступ;
 - 2) аналіз інтерфейсу користувача;
 - 3) аналіз програмних особливостей;
 - 4) переваги середовища розробки;
 - 5) недоліки середовища розробки;
- в) розроблено алгоритм розробки програмного забезпечення;
- г) розроблено діаграму взаємодії стендових служб;
- д) розроблено архітектуру програмного забезпечення.

4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Алгоритм роботи тестового програмного забезпечення для проведення випробувань на стендових системах

Алгоритм роботи програмного забезпечення зображено на рисунках 4.1 – 4.2.

Блок 1 — запуск проєкту на виконання.

Блок 2 — очікування активації користувачем кнопки «пост-обробка».

Блок 3 — написання коду з реалізацією функцій відкриття вікна пост-обробки даних, запуску його на виконання, очікування завершення роботи з відкритим вікном, закриття вікна пост-обробки даних.

Блок 4 — написання коду, за допомогою якого підпрограма управління створює шлях до директорії в яку необхідно помістити базу даних, створює назву бази, створює файл бази даних.

Блок 5 — написання коду, за допомогою якого проводиться відкриття файлу бази даних, створюються групи та канали для запису в базу, виділяється вільний простір на дисковому просторі, налаштовується запис, проводиться запис до бази даних.

Блок 6 — написання коду для збереження, закриття бази та виводу діалогового вікна з помилками, що виникли про роботі бази даних.

Блок 7 — Ініціалізація одномірного масиву нульових значень.

Блок 8 — написання коду, для об'єднання вимірюваних даних, та даних отриманих після усереднення, даних про відносний час на графіку в один двомірний масив для запису до бази даних.

Блок 9 — написання коду, який виконує конфігурацію послідовного порту, налаштування з'єднання з послідовним СОМ портом, очищує буфер вводу-виводу.

Блок 10 — написання коду, що перевіряє кількість даних в байтах, що надійшли до буферу вводу-виводу.

Блок 11 — читання даних з COM порту, перетворення зчитаних даних з типу string до типу int.

Блок 12 — перевірка, чи була натиснута кнопка зупинки головного циклу підпрограми управління. У разі активації, завершує читання та запис даних до бази.

Блок 13 — написання коду, для зупинки з'єднання з COM портом та виводу діалогового вікна з помилками, що виникли про роботі з послідовним портом.

Блок 14 — написання коду, для проведення усереднення вимірюваних параметрів за вказаним користувачем інтервалом часу.

Блок 15 — написання коду для об'єднання усіх отриманих даних в кластер значень, та виводу на графік.

Блок 16 — написання коду для отримання відносних значень часу з графіку, перевірка роботи графіка.

Блок 17 — написання коду для очистки даних на графіку шляхом заміни історії графіку на кластер нульових значень.

Блок 18 — програмна реалізація запуску вікна пост-обробки даних або завершення роботи тестової системи проведення випробувань.

Блок 19 — написання коду, що виконує функції відкриття обраної бази даних, читання даних з неї та закриття бази даних.

Блок 20 — написання коду, що дозволяє вибирати між заданими заздалегідь наборами даних для демонстрації можливостей роботи програмного забезпечення, та набором «інтерполяція баз даних», що містить налаштування для проведення інтерполяції при роботі зі збереженими базами даних.

Блок 21 — підготовка даних, налаштування діапазонів значень та частоти роботи методів для інтерполяції.

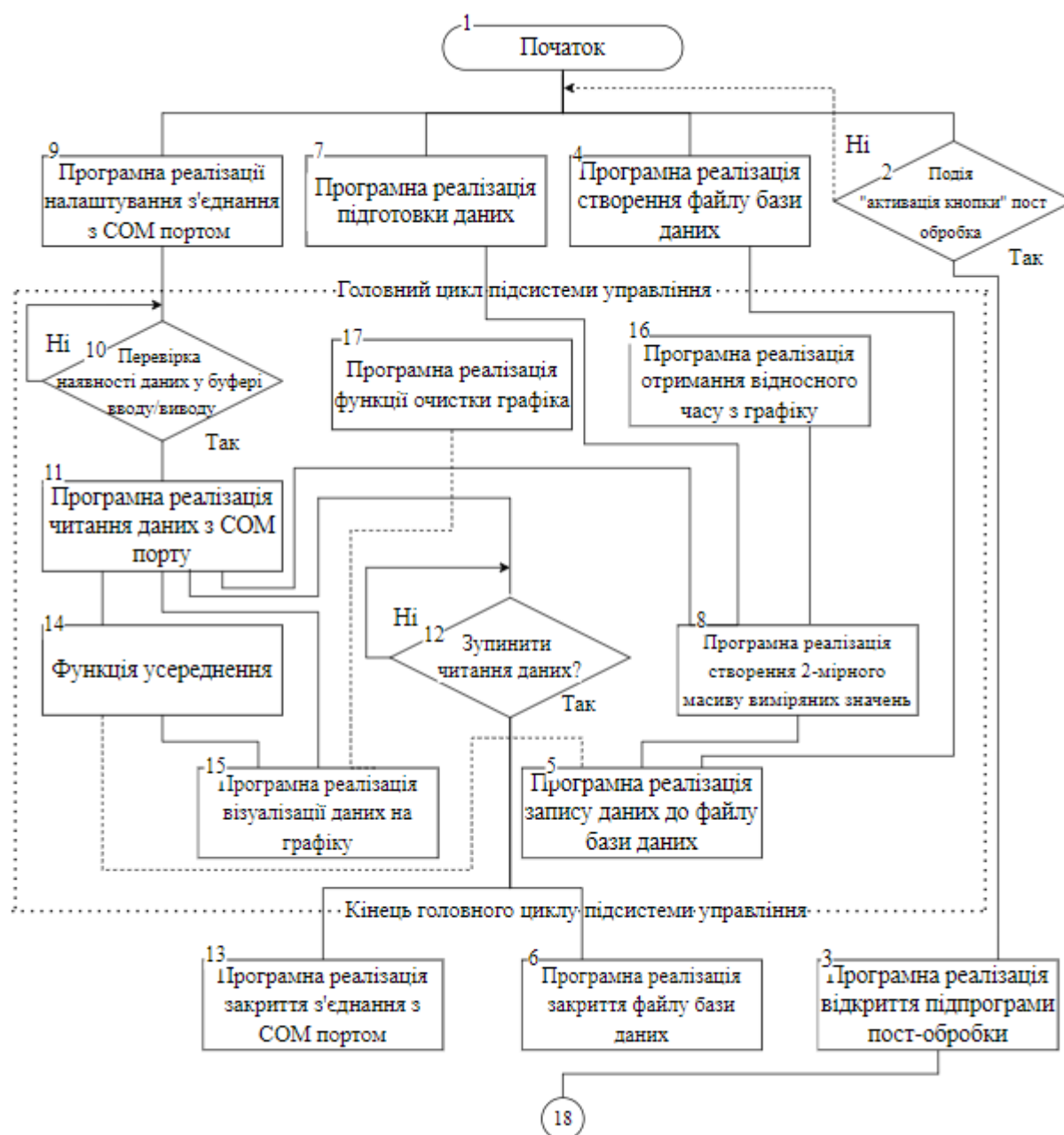


Рисунок 4.1 – Алгоритм роботи програмного забезпечення

Блок 22 — написання коду для виводу вузлових значень, побудованих точок інтерполяції та графіку на екран.

Блок 23 — реалізація можливості завершення роботи з підпрограмою пост-обробки даних за активацією відповідної кнопки.

Блок 24 — написання коду для перетворення даних у необхідний вид для роботи функції апроксимації методом найменших квадратів.

Блок 25 — написання коду для виводу вузлових значень та графіку апроксимації на екран.

Блок 26 — завершення роботи підпрограми пост-обробки та управління за допомогою кнопки на екрані.

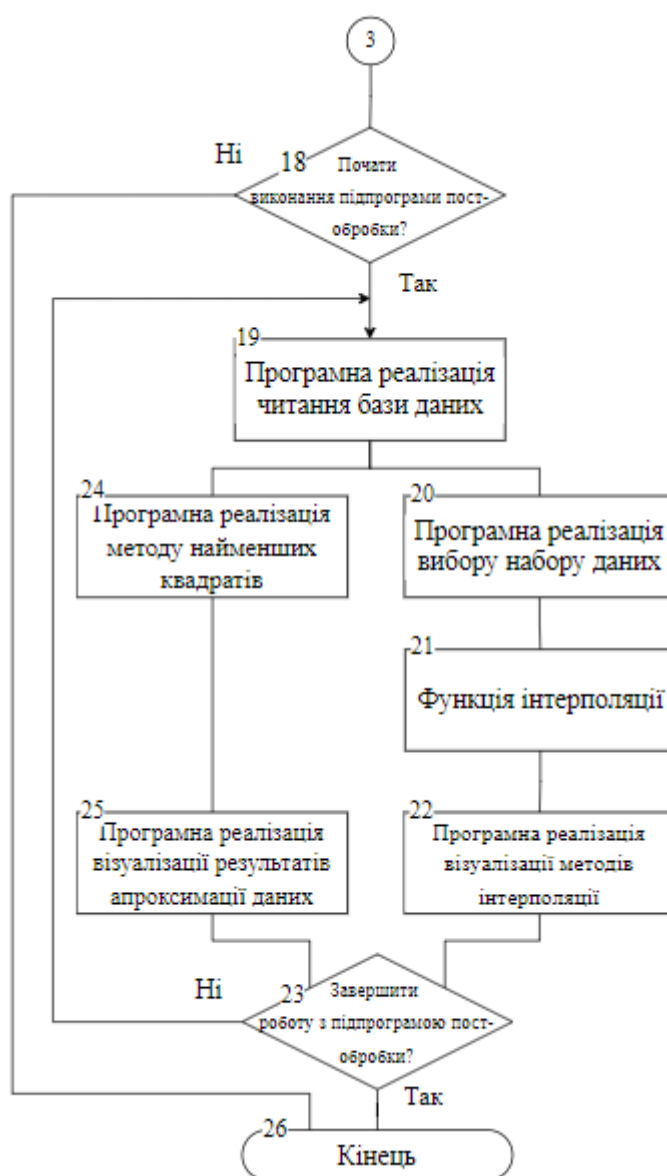


Рисунок 4.2 – Продовження алгоритму роботи програмного забезпечення

4.2 Алгоритм роботи оператора з тестовою системою збору даних

Роботу оператора з тестовою системою збору даних необхідно проводити за алгоритмом зображеним на рисунках 4.3 – 4.4.

Блок 1 — запуск тестової системи збору даних.

Блок 2 — налаштування кількості точок за якими буде проводитися усереднення параметрів, вказання назви для бази даних.

Блок 3 – запустити виконання програмного забезпечення.

Блок 4 – перевірка працездатності програми на надходження даних з СОМ порту, спостереження за вимірюваними параметрами, при необхідності використання функцій очистки графіка, активація алгоритму усереднення, або зміна кількості точок для усереднення.

Блок 5 –запустити запис даних до бази.

Блок 6 – завершити запис до бази даних за допомогою перемикача.

Блок 7 –зупинити виконання ПЗ відповідною кнопкою.

Блок 8 – завершити роботу з ПЗ, або зробити аналіз записаної бази даних.

Блок 9 – відкрити вікно пост-обробки даних, вказати шлях до створеної бази даних, вказати частоту інтерполяції.

Блок 10 – запустити виконання програмного забезпечення для обробки даних.

Блок 11 – при необхідності вибрати іншу базу даних, частоту інтерполяції чи метод для аналізу, можливо використовувати вкладенки для спостереженнями за сирими та інтерпольованими даними.

Блок 12 – завершити роботу, або вибрати іншу базу для аналізу.

Блок 13 – зупинити виконання підпрограми пост-обробки та аналізу.

Блок 14 – завершити виконання тестової системи збору даних.

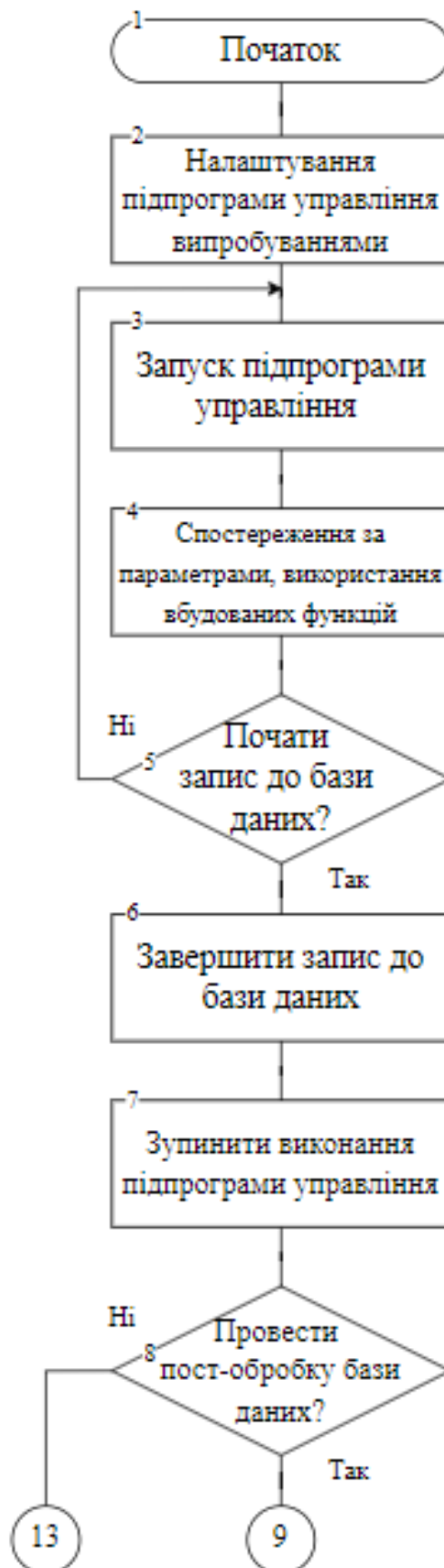


Рисунок 4.3 – Алгоритм роботи оператора з тестовою системою збору даних

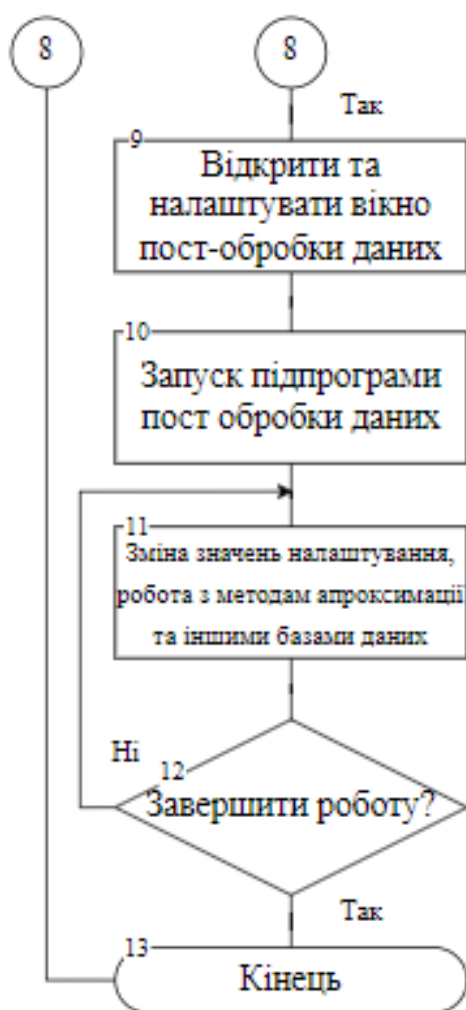


Рисунок 4.4 – Продовження алгоритму роботи оператора з тестовою системою збору даних

4.2 Алгоритм роботи розробника з тестовою системою збору даних

Роботу розробника з тестовою системою збору даних необхідно проводити за таким алгоритмом (рис. 4.5).

Блок 1 — відкрити проєкт.

Блок 2 — почати роботи згідно технічного завдання.

Блок 3 — дослідити який параметр необхідно вимірювати, обрати плату, що зможе виконати це завдання, налаштувати режими роботи плати для конкретного параметра.

Блок 4 — розробити VI для читання даних з плати, перевірити коректність отриманих даних за допомогою порівняння з еталонним значенням, перетворити дані в зручний формат для передачі та візуалізації.

Блок 5 — налаштувати канал передачі даних від VI для читання даних з плати до main VI проєкту, налаштувати запуск розробленого VI під час запуску ПЗ.

Блок 6 — модифікувати функції модуля візуалізації (очистки, усереднення, відображення на графіку та на індикаторах), для роботи з новим параметром.

Блок 7 — модифікувати групу та назви каналів у модулі збереження даних, додати новий параметр до масиву записуваних параметрів.

Блок 8 — додати новий параметр на графік підпрограми пост-обробки, вивести параметр на індикатори, до модуля розрахунку, розраховані дані вивести на графік та індикатори.

Блок 9 — запустити ПЗ, перевірити усі функції.

Блок 10 — вибрати номер порту до якого надходять значення.

Блок 11 — змінити директорію для створення файлу з базою даних, на коректну.

Блок 12 — запустити ПЗ, перевірити виконання всіх функцій.

Блок 13 — зберегти, скопіювати проєкт та завершити роботу.

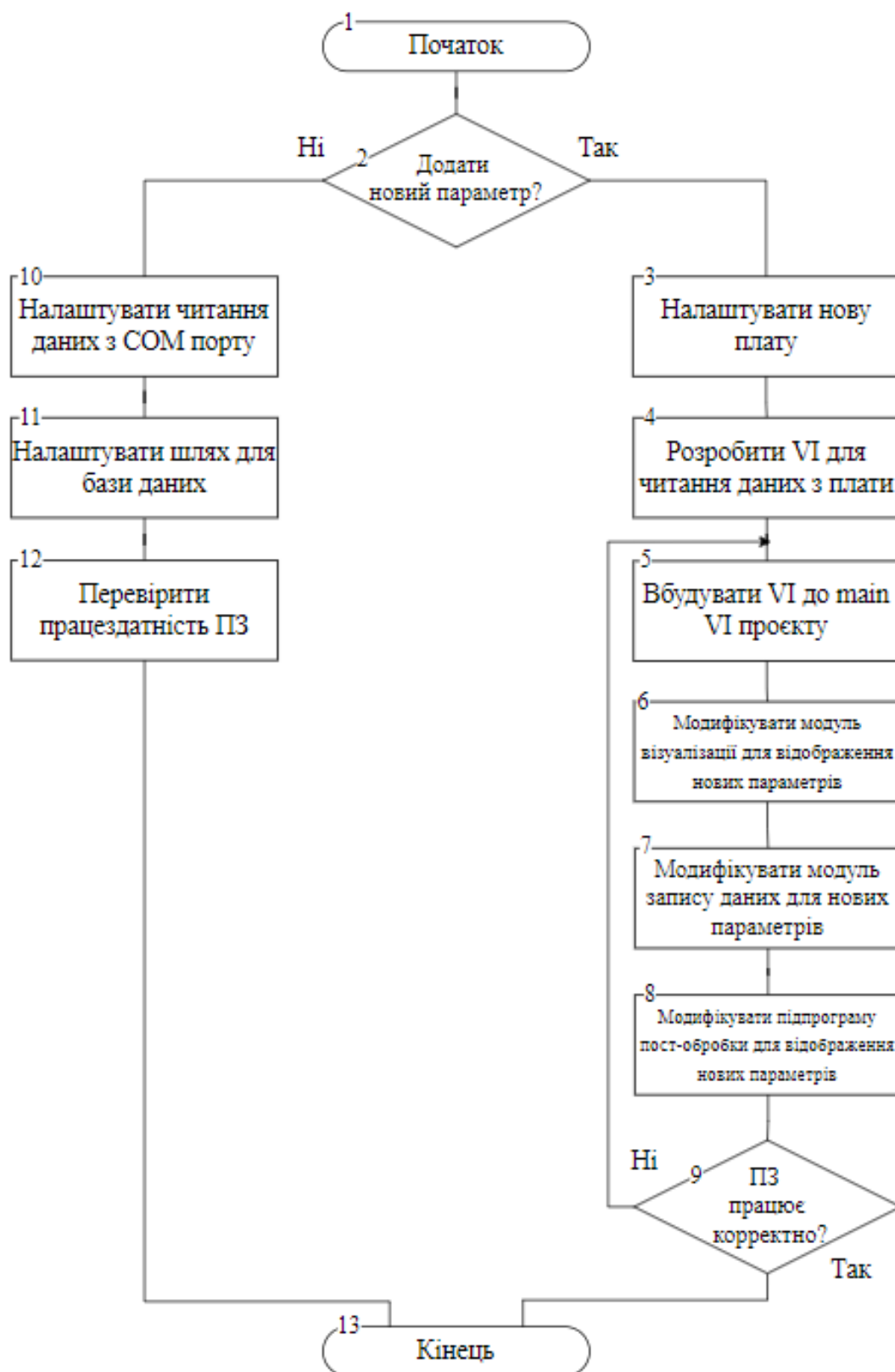


Рисунок 4.5 – Алгоритм роботи розробника з тестовою системою збору даних

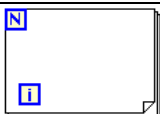
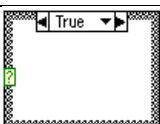
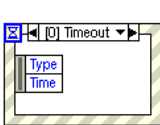
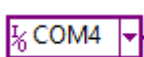
4.3 Функції які використовувалися в розробці програмного забезпечення

Перелік усіх використаних у роботі функцій, що являють собою VI з бібліотеки LabVIEW ізображено в таблиці 4.1.

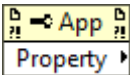
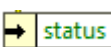
Таблиця 4.1 – Перелік використаних VI [29]

Назва VI	Піктограма	Опис функції
1	2	3
Оператори		
АБО		Обчислює логічне АБО вхідних даних. Обидва вхідні значення мають бути логічними значеннями, числовими значеннями або кластерами помилок. Якщо обидва вхідні дані мають значення FALSE, функція повертає FALSE. В іншому випадку він повертає TRUE
AND		Обчислює логічне AND вхідних даних. Обидва вхідні дані мають бути логічними значеннями, числовими значеннями або кластерами помилок. Якщо обидва вхідні дані TRUE, функція повертає TRUE
Більше?		Повертає TRUE, якщо x більше за y. В іншому випадку ця функція повертає FALSE. Можливо змінити режим порівняння цієї функції
Інкремент		Додає 1 до вхідного значення
Дорівнює?		Повертає TRUE, якщо x дорівнює y. В іншому випадку ця функція повертає FALSE
Множення		Повертає добуток вхідних даних
Індикатори		
Індикатор числа		Відображає на лицьовій панелі приладу значення, що надходять до цієї функції
Графіки та діаграми форми хвилі		Відображає дані на графіку, які зазвичай збираються з постійною швидкістю
Граф XY		Це декартовий графічний об'єкт загального призначення, який зображує багатозначні функції, такі як кругові форми або форми хвилі з різною часовою базою. На графіку XY відображається будь-який набір точок, з рівномірною вибіркою чи ні

Продовження таблиці 4.1 [29]

1	2	3
Логічні елементи керування та індикатори		Дозволяє використовувати логічні елементи керування та індикатори, щоб вводити та відображати логічні значення (TRUE/FALSE) за допомогою таких об'єктів, як кнопки, перемикачі та світлодіодні індикатори
Елементи керування та індикатори рядка та шляху		Функція надає можливість керування та індикації для введення або відображення шляхів до файлів або папок
Елементи керування та індикатори вкладинок		Функція надає можливість керування та індикації для введення або відображення шляхів до файлів або папок
Цикли та структури		
Цикл While		Повторює код у своїй під діаграмі, доки не виникне конкретна умова. Цикл While завжди виконується принаймні один раз
Цикл FOR		Виконує свою під діаграму n разів, де n — значення, підключене до терміналу лічильника (N). Термінал ітерації (i) надає поточний підрахунок ітерацій циклу, який коливається від 0 до n-1
Структура умови		Містить одну або кілька під діаграм або умов, одна з яких виконується під час виконання структури. Значення, підключене до селектора регістру, визначає, який регістр потрібно виконати
Структури подій		Чекає, поки не станеться подія, а потім виконує відповідний випадок для обробки цієї події. Структура подій має одну або кілька під діаграм або випадків подій, саме один з яких виконується, коли структура виконується для обробки події
Функції		
Керування ім'ям ресурсу VISA		Необхідний щоб указати ресурс, для якого буде відкритий сеанс VISA, а також для підтримки сеансу та класу
Налаштування послідовного порту VISA		Ініціалізує послідовний порт, визначений ім'ям ресурсу VISA, що вказаний при налаштуванні
Очистка буферу VISA		Функція очистки буфера введення-виведення
Читати VISA		Зчитує вказану кількість байтів з пристрою або інтерфейсу, зазначеного в імені ресурсу VISA, і повертає дані в буфер читання

Продовження таблиці 4.1 [29]

1	2	3
Закрити VISA		Закриває сеанс пристрою або об'єкт події, визначений ім'ям ресурсу VISA
Вузел властивостей		Одержує (читає) та/або встановлює (записує) властивості посилання. Вузол властивостей використовується щоб отримати або встановити властивості та методи для локальних або глобальних екземплярів програми, VI та об'єктів. Також можливо використовувати вузол властивостей для доступу до приватних даних класу LabVIEW
Затримка виконання		Чекає вказану кількість мілісекунд і повертає значення мілісекундного таймера, очікування введення змушує поточний потік передати контроль над центральним процесором
Пошук та розділення рядка		Розділяє один рядок на два під рядки за вказаним розділовим символом
Конвертація рядка до числа		Перетворює числові символи в рядку, починаючи зі зміщення, в десяткове ціле число і повертає його у вигляді числа
Групування		Збирає кластер з окремих елементів
Розгрупування за назвою		Повертає елементи кластера, імена яких були вказані
Обробник помилок		Вказує, чи сталася помилка. Якщо сталася помилка, цей VI повертає опис помилки та за бажанням відображає діалогове вікно
Ініціалізація масиву		Створює n-вимірний масив, у якому кожен елемент ініціалізується значенням елемента
Заміна підмножини у масиві		Замінює елемент або під масив у масиві в точці, яка була вказана в індексі
Індекс з масиву		Повертає елемент або під масив n-вимірного масиву за індексом
Побудувати масив		Об'єднує кілька масивів або додає елементи до n-вимірного масиву. Також можливо використовувати функцію заміни підмножини масиву, щоб змінити існуючий масив
Транспонувати масив		Переставляє елементи 2D-масиву таким чином, що 2D-масив [i,j] стає транспонованим масивом [j,i]
Розмірність масиву		Повертає кількість елементів у кожному вимірі масиву
Шаблон рампи		Генерує масив, що містить шаблон наростання. Забезпечує можливість вручну вибрати поліморфний екземпляр, який буде використовуватися

Продовження таблиці 4.1 [29]

1	2	3
Інтерполяція		Виконує одновимірну інтерполяцію за допомогою вибраного методу на основі таблиці пошуку, визначеної значеннями X та Y
Відкриття TDMS		Відкриває файл .tdms для читання або запису байтів даних. Можливо використовувати цю функцію, щоб створити новий файл або замінити наявний файл
Встановити інформацію про канал TDMS		Визначає інформацію про канал вихідних даних, які потрібно записати у вказаний файл .tdms. Інформація про канал включає розташування даних, назву групи, назви каналів, тип даних і кількість вибірок
Резервування розміру файлу TDMS		Попередньо виділяє дисковий простір для запису та запобігає фрагментації на рівні файлової системи
Налаштування асинхронного запису TDMS		Виділяє буфери та налаштовує значення тайм-ауту для асинхронного запису. Значення часу очікування застосовується до всіх наступних асинхронних записів
Асинхронний запис TDMS		Записує дані у вказаний файл .tdms асинхронно. Ця функція може ініціювати кілька асинхронних записів, які відбуваються у фоновому режимі
Закрити TDMS		Закриває файл .tdms, відкритий за допомогою функції TDMS Advanced Open, і звільняє додатковий дисковий простір, який резервує функція TDMS Reserve File Size
Читання TDMS		Читає вказаний файл .tdms і повертає дані у форматі, визначеному типом даних. Якщо дані мають інформацію про масштабування, цей ВП автоматично масштабує дані
Усереднення		Обчислює середнє значення в наборі точок вхідних даних, визначених довжиною вибірки. Якщо кількість значень менша за довжину вибірки
Об'єднання рядків		Об'єднує вхідні рядки та одновимірні масиви рядків в один вихідний рядок. Для вхідних даних масиву ця функція об'єднує кожен елемент масиву
Конвертація рядка до шляху		Перетворює рядок, що описує шлях у стандартному форматі для поточної платформи, у шлях
Створення шляху		Створює новий шлях шляхом додавання імені або відносного шляху до існуючого шляху
Вибір		Повертає значення, підключене до входу t або входу f, залежно від значення s. Якщо s має значення TRUE, ця функція повертає значення, підключене до t. Якщо s має значення FALSE, ця функція повертає значення, підключене до f

Продовження таблиці 4.1 [29]

1	2	3
Апроксимація		Повертає лінійну підгонку набору даних (X, Y) за допомогою методу найменшого квадрата, найменшого абсолютного залишку або біквадрата
Вузол зворотного зв'язку		Зберігає дані від одного виконання VI або ітерації циклу та передає до наступного
Відкрити VI		Повертає посилання на VI, користувацький елемент керування або глобальну змінну, визначену рядком імені або шляхом до розташування VI на диску
Вузол виклику		Викликає метод або дію над посиланням. Більшість методів мають пов'язані параметри
Закрити посилання		Закриває посилання, пов'язане з відкритим об'єктом VI, відкритим екземпляром програми або об'єктом .NET або ActiveX
Максимум та мінімум масиву		Повертає максимальне та мінімальне значення, знайдені в масиві, разом з індексами для кожного значення
Перетворити час у дату		Перетворює значення позначки часу або числове значення в рядок дати та часу. Функція інтерпретує позначку часу та числові значення як незалежну від часового поясу кількість секунд, що пройшли з 00:00, п'ятниці, 1 січня 1904 року, за всесвітнім часом
Власноруч розроблені VI		
Усереднення		Функція використовується для проведення усереднення за визначеною кількістю точок. Приймає значення типу double для даних, що будуть осереднюватись, та integer, для визначення кількості точок, за якими буде проведено усереднення. Повертає розраховане значення типу double
Отримати дату		Функція використовується для представлення всесвітнього координованого часу в формат UTC+2, або UTC+3. Приймає значення часу на нульовому меридіані та розраховує час в Україні. Також функція робить перехід на зимовий час з 30 на 31 жовтня, або літній час з 27 на 28 березня

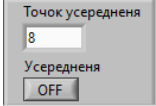
Змінні що є у символьних мовах програмування при розробці тестової системи збору даних не використовувалися. У графічній мові програмування LabVIEW замість змінних використовуються дроти. Коли вузол, VI чи функція закінчує виконання, значення, що вона повертає зберігається у дроті до того

моменту, як ці дані зможуть потрапити у наступний елемент програми. Після цього, значення дротів видаляються.

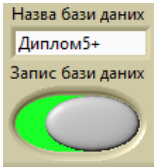
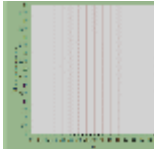
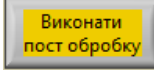
4.4 Огляд розробленого інтерфейсу та функцій, що виконують програмне забезпечення

При розробці використовувались вбудовані функції LabVIEW з яких були створені власні функції для досягнення завдань поставлених в меті роботи, та у вимогах до програмного забезпечення. Перелік існуючих функцій зображено в таблиці 4.2.

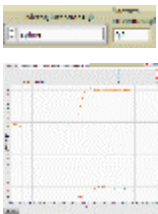
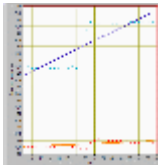
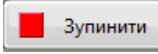
Таблиця 4.2 – Перелік функцій ПЗ

Назва	Вигляд елемента управління чи індикації на лицьовій панелі	Тип елемента управління чи індикації	VI з яких складається функція	Тип отриманих та повернутих значень	Призначення
1	2	3	4	5	6
Функції у інтерфейсі підпрограми управління					
Очистка графіка		Кнопка	Вбудовані VI (вузол властивостей)	Приймає: boolean, повертає: 1D array of cluster	Очистка історії вимірюваних параметрів
Усереднення вимірюваних параметрів		Текстове поле + кнопка	Розроблене VI (усереднення)	Приймає: double, boolean, integer; повертає: double	Усереднює фіксовану кількість отриманих значень

Продовження таблиці 4.2

1	2	3	4	5	6
Запис до бази даних		Текстове поле + повзунний перемикач	Вбудовані VI (відкриття TDMS, встановити інформацію про канал TDMS, резервування розміру файлу TDMS, Налаштування асинхронного запису TDMS, асинхронний запис TDMS, закрити TDMS конвертація рядка до шляху, створення шляху)	Приймає: boolean, integer, 2D array of double; повертає: tdms файл	Створює новий файл бази даних з розширенням .tdms, конфігурує файл, записує до нього дані випробувань, закриває файл
Візуалізація даних		Графік	Вбудовані VI (Графіки та діаграми форми хвилі, групування)	Приймає: cluster of double; повертає: cluster of double	Візуалізує вимірювані параметри на графіку
Зупинка читання даних		Кнопка	Вбудовані VI (розгруповання за назвою, оператор АБО)	Приймає: boolean; повертає: boolean	Завершує головний цикл підпрограми проведення випробувань
Виконати пост-обробку даних		Кнопка	Вбудовані VI (відкрити VI, вузол виклику, закрити посилання)	Приймає: подія активації кнопки «Виконати пост обробку»; Повертає: посилання на VI	Відкриває файл Diplom_Interpolate.vi та запускає його на виконання, після завершення роботи з обробкою даних, закінчує виконання програмного забезпечення для проведення випробувань
Дата		Індикатор текстове поле	Розроблене VI (отримати дату)	Приймає: string of time; повертає: string of time	Використовується для відображення поточного часу
Функції у інтерфейсі підпрограми пост-обробки та аналізу					
Відкрити базу даних		Текстове поле + кнопка	Вбудовані VI (створення шляху, відкрити TDMS, читання TDMS, закрити TDMS)	Приймає: шлях до файлу; повертає: 2D array of double	Використовується для вибору бази даних для читання та отримання даних з бази даних

Продовження таблиці 4.2

1	2	3	4	5	6
Запуск		Кнопка	Вбудовані VI (Логічні елементи керування та індикатори)	Приймає: boolean; повертає: boolean	Використовується для запуску підпрограми пост-обробки на виконання
Інтерполяція		Графік + випадаючий список + текстове поле	Вбудовані VI (граф XY, інтерполяція, групування, шаблон рампи, будувати масив)	Приймає: 2D array of double; повертає: cluster of double	Виконує інтерполяцію значень з бази даних
Апроксимація		Графік + індикатор текстове поле	Вбудовані VI (апроксимація, групування, транспонувати масив, будувати масив, граф XY)	Приймає: 2D array of double; повертає: cluster of double	Виконує апроксимацію значень з бази даних
Зупинити виконання		Кнопка	Вбудовані VI (Логічні елементи керування та індикатори)	Приймає: boolean; повертає: boolean	Використовується для зупинки програмного забезпечення

Далі зображено розроблену лицьову панель підпрограми проведення випробувань (рис. 4.6) та підпрограми пост-обробки даних (рис. 4.7).

Перелічені підпрограми можуть працювати незалежно одна від одної, також реалізована функція виконання пост-обробки даних що дозволяє відкривати підпрограми обробки не завершуючи виконання підпрограми проведення випробувань.

Далі продемонстрований інтерфейс підпрограми управління:

– функція очистки графіку. Приклад роботи зображено на рисунку 4.8, жовтим прямокутником виділено елементи керування функцією на лицьовій панелі;

– функція усереднення вимірюваних параметрів. Приклад роботи зображено на рисунку 4.9, жовтим прямокутником виділено елементи керування функцією на лицьовій панелі;

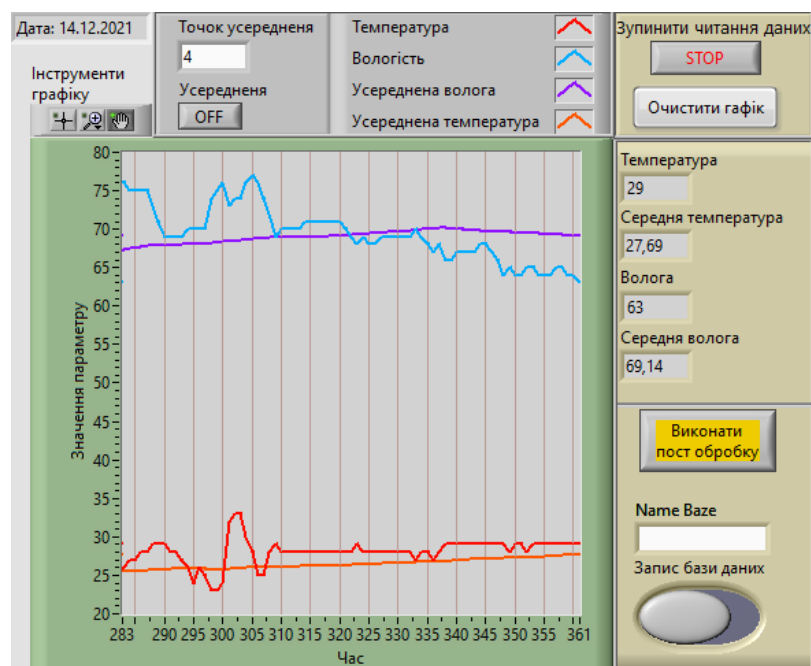


Рисунок 4.6 – Лицьова панель підпрограми управління

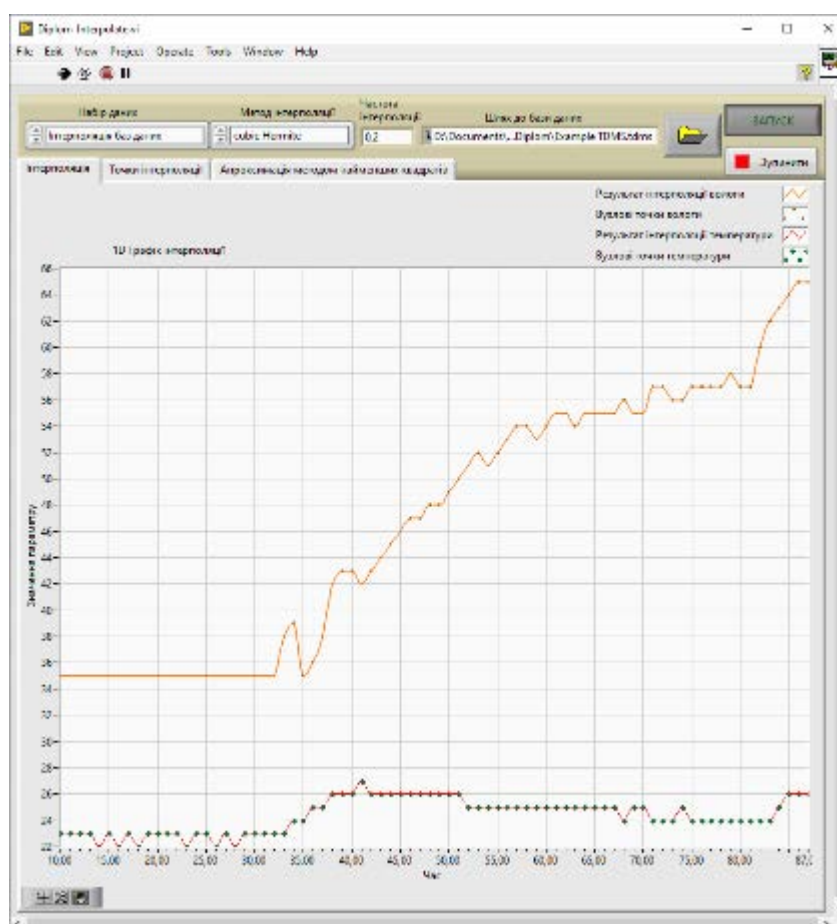


Рисунок 4.7 – Розроблена лицьова панель підпрограми пост-обробки даних

- функція запису до бази даних. Приклад роботи зображено на рисунку 4.10, жовтим прямокутником виділено елементи керування функцією на лицьовій панелі;
- функція зупинки читання даних. Елемент керування виділений жовтим прямокутником на рисунку 4.11;
- функція виконати пост-обробку даних. Елемент керування виділений жовтим прямокутником на рисунку 4.12.

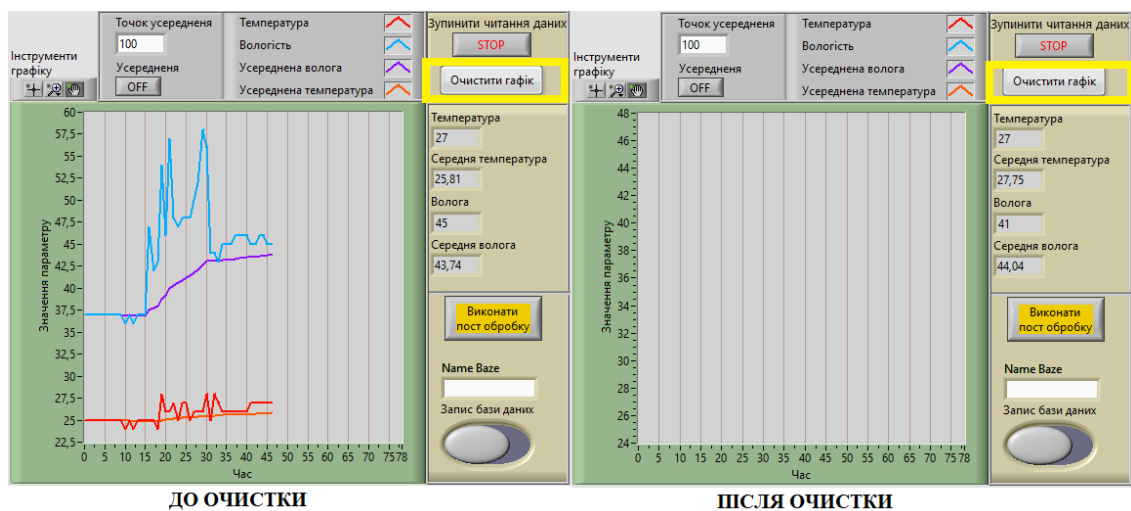


Рисунок 4.8 – Приклад роботи функції очистки

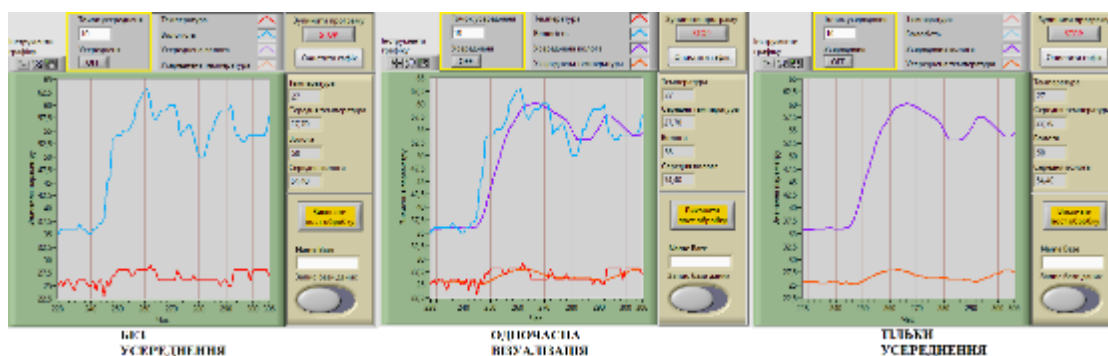


Рисунок 4.9 – Приклад роботи функції усереднення

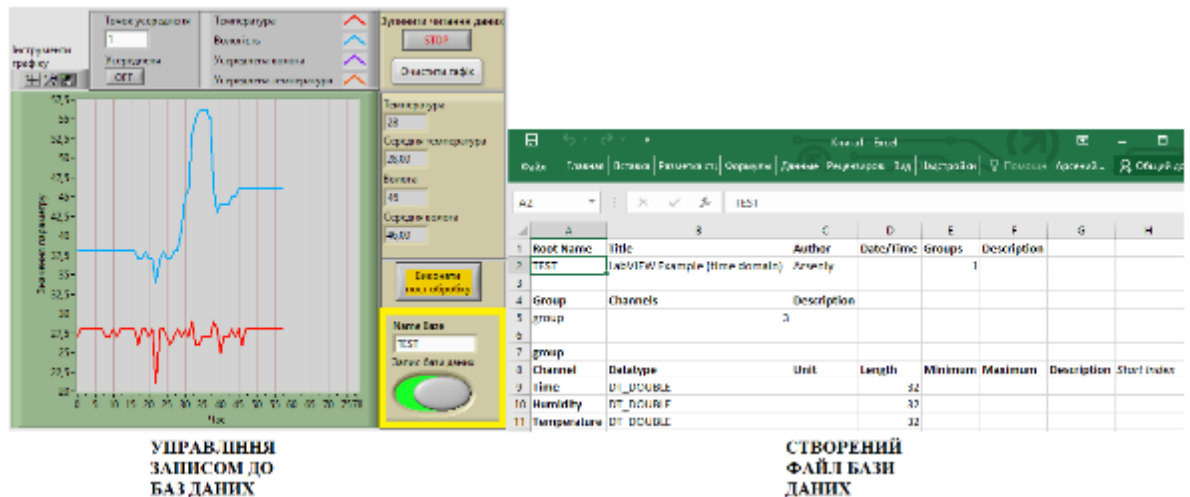


Рисунок 4.10 – Приклад роботи функції запису до бази даних

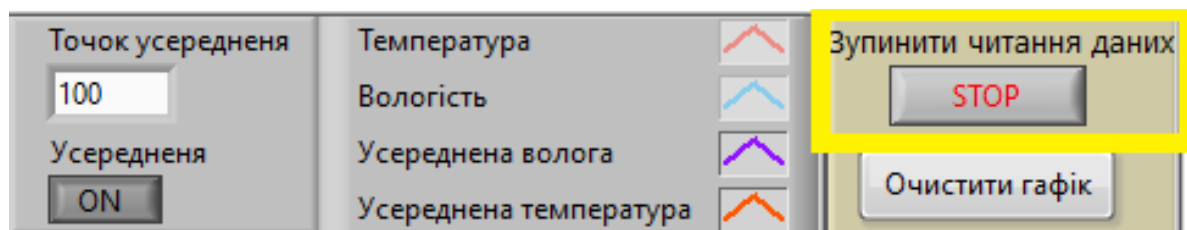


Рисунок 4.11 – Функція зупинки читання даних

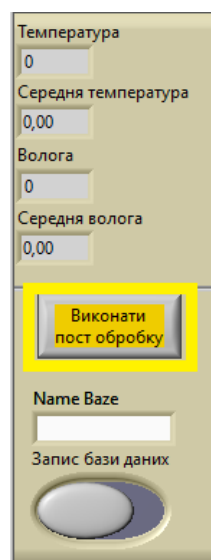


Рисунок 4.12 – Функція виконати пост-обробку даних

Далі зазначено перелік функції підпрограми пост-обробки даних та їх опис:

- функція відкрити базу даних. Реалізація функції зображена на рисунку 4.13, жовтим прямокутником виділено елементи керування функцією;
- інтерполяція. Елементи керування функцією виділено жовтим прямокутником, що зображені на рисунку 4.14;
- функція запуск. Елемент керування функцією виділено жовтим прямокутником, зображено на рисунку 4.15;
- функція зупинити виконання. Реалізація функції зображена на рисунку 4.16, жовтим прямокутником виділено елемент керування функцією.

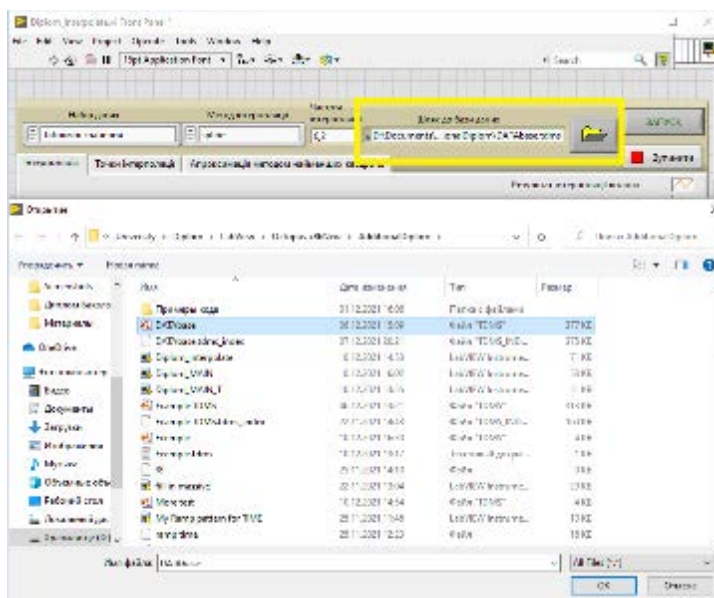


Рисунок 4.13 – Приклад роботи функції відкриття бази даних

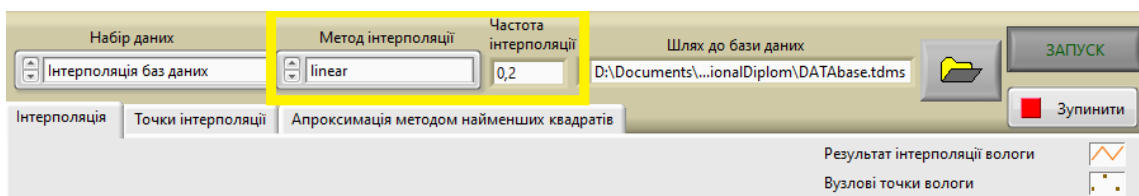


Рисунок 4.14 – Функція налаштування інтерполяції

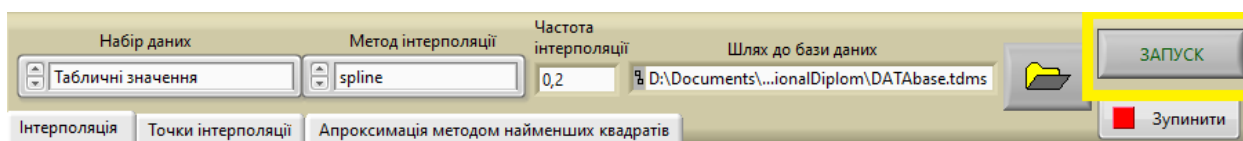


Рисунок 4.15 – Функція запуск

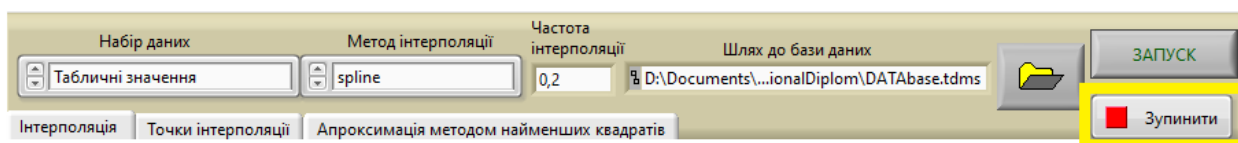


Рисунок 4.16 – Функція зупинити виконання

Обидві підпрограми можуть працювати незалежно, проте, для більшої зручності була реалізована можливість викликати пост-обробку VI, напряду з VI управління.

Також, реалізована можливість перемикавання вкладинок, для зручного представлення результатів методів апроксимації. Можливо використовувати для навігації по лицьовій панелі підпрограми пост-обробки даних між вкладинками «Інтерполяція», «Точки інтерполяції», «Апроксимація методом найменших квадратів». Приклад роботи зображено на одному наборі даних, для однієї бази даних на рисунках 4.17 – 4.19.

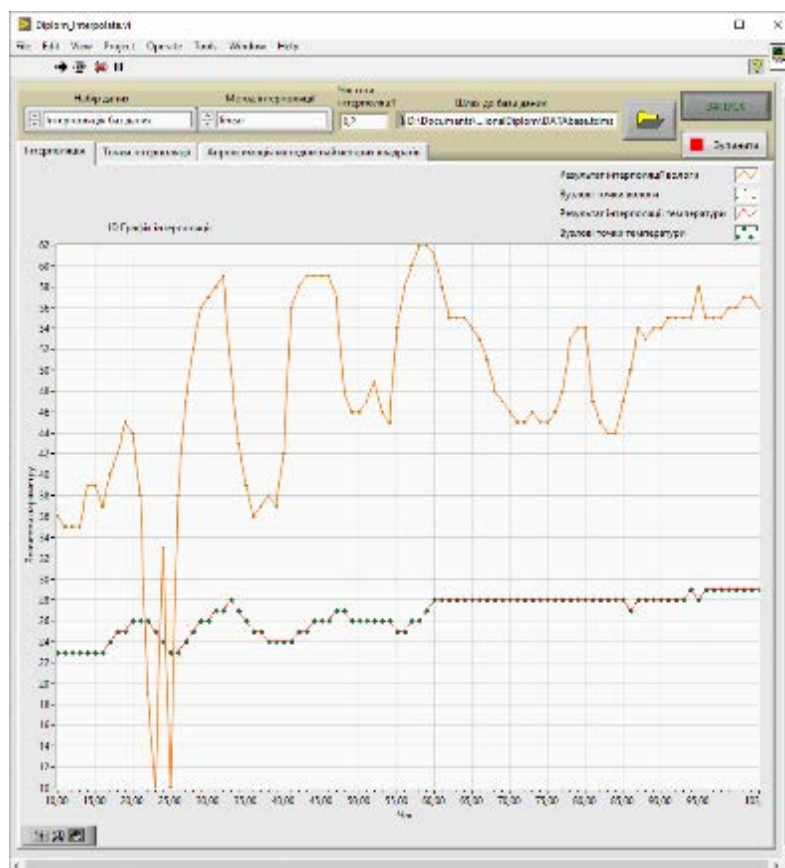


Рисунок 4.17 – Вкладка «Інтерполяція»

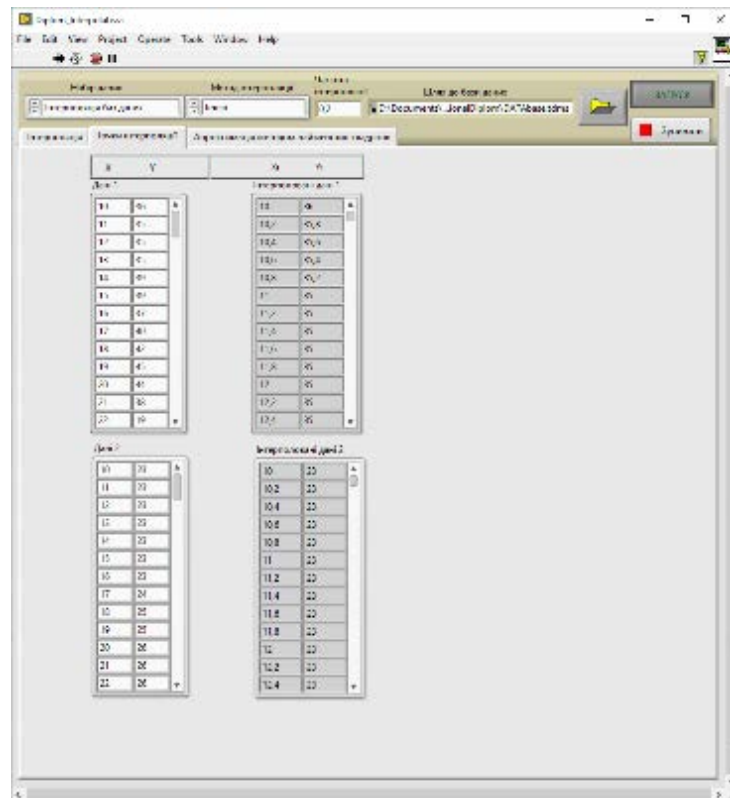


Рисунок 4.18 – Вкладка «Точки інтерполяції»

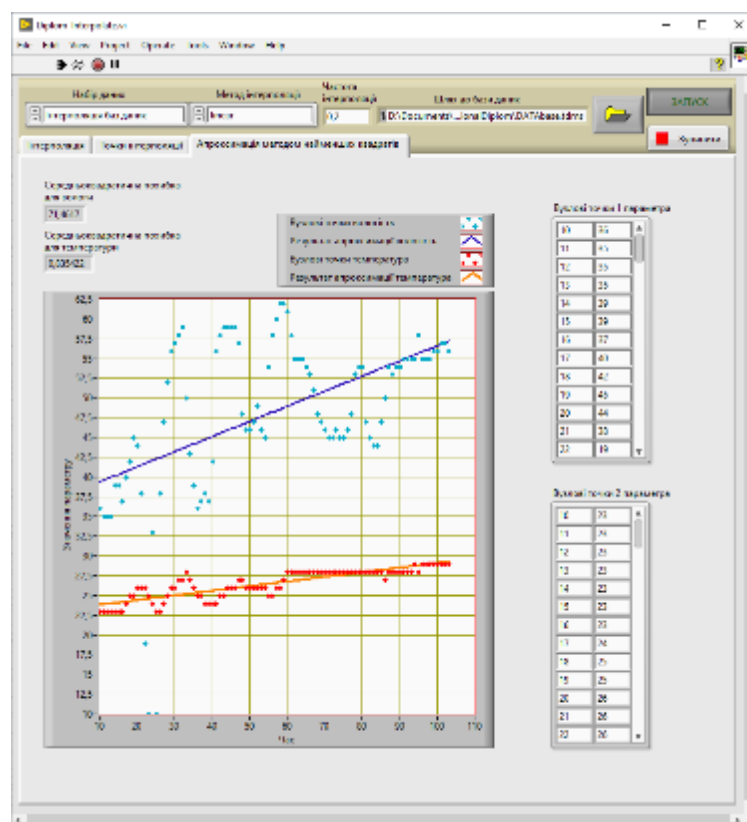


Рисунок 4.19 – Вкладка «Апроксимація методом найменших квадратів»

4.5 Висновки до розділу 4

Розроблена тестова система проведення випробувань підтвердила ефективність методів апроксимації для покращення візуалізації параметрів під час проведення досліджень. Результати експериментів були продемонстровані розробнику ПЗ Ostorus. Інженером-розробником було внесено досліджені в роботі методи інтерполяції до підпрограми пост-обробки та аналізу даних (рис. 4.20) програмного забезпечення для проведення стендових випробувань Ostorus.

Частота	AutoConfig						
16	ЛОЖЬ						
Канал	Линия	Тип данных	Тип	Минимум	Максимум	Пределы	Недатчика
0x00		I16	Линейный	0	25000		
0x01		I16	Линейный	0	25000		
0x02		I16	Линейный	0	10000		
0x03		I16	Линейный				
0x04		I16	Линейный	0	500		
0x05		I16	Линейный	0	500		
0x06		I16	Линейный	0	500		
0x07		I16	Линейный	0	500		
0x20		I16	Линейный	-100	100		
0x21		I16	Линейный	-100	100		
0x22		I16	Линейный	0	100		
0x23		I16	Коды	0	125		
0x24		I16	Полином		180		
0x25		I16	БлижТочки	0	100		
0x26		I16	Линейный		253		
0x27		I16	Формула		273		
0x10		I16	К-ДСТУ60584-1		40		
0x11		I16	L-ГОСТ8.585	0	40		
0x12		I16	R-ДСТУ60584-1	0	40		

Буде

Частота	AutoConfig						
16	ЛОЖЬ						
Канал	Линия	Тип данных	Тип	Минимум	Максимум	Пределы	Недатчика
0x00		I16	Линейный	0	25000		
0x01		I16	Линейный	0	25000		
0x02		I16	Линейный	0	10000		
0x03		I16	Линейный				
0x04		I16	Линейный	0	500		
0x05		I16	Линейный	0	500		
0x06		I16	Линейный	0	500		
0x07		I16	Линейный	0	500		
0x20		I16	Линейный	-100	100		
0x21		I16	Линейный	-100	100		
0x22		I16	Сплайн	0	100		
0x23		I16	Коды	0	125		
0x24		I16	Полином		180		
0x25		I16	Ближ точки	0	100		
0x26		I16	Сплайн		253		
0x27		I16	Эрмит		273		
0x10		I16	Лагранж		40		
0x11		I16	Формула	0	40		
0x12		I16	Линейный	0	40		

Стало

Рисунок 4.20 – Результат додавання методів інтерполяції до ПЗ Ostorus

Також, нові методи інтерполяції були внесені до файлу «Руководство оператора», що зображено на рисунках 4.21 – 4.22.

Результат використання методу інтерполяції сплайном зображено на рисунку 4.23. Інтерполяція проводилася на базі даних з випробувального стенду №14, дані були зібрані з маслоагрегата двигуна аі35. Жовтими колами виділено частини графіку що були покращенні під час обробки даних.

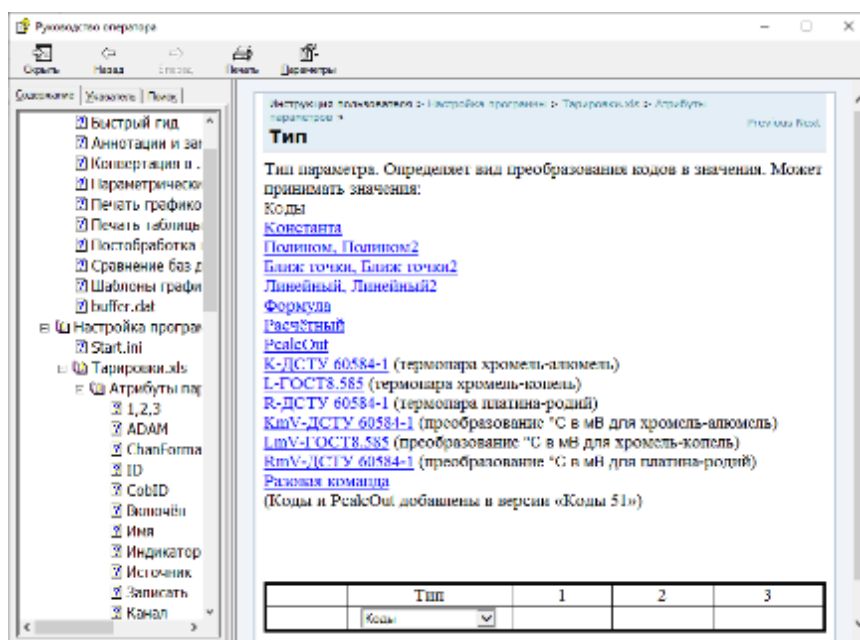


Рисунок 4.21 – Керівництво оператора Ostorpus минула версія

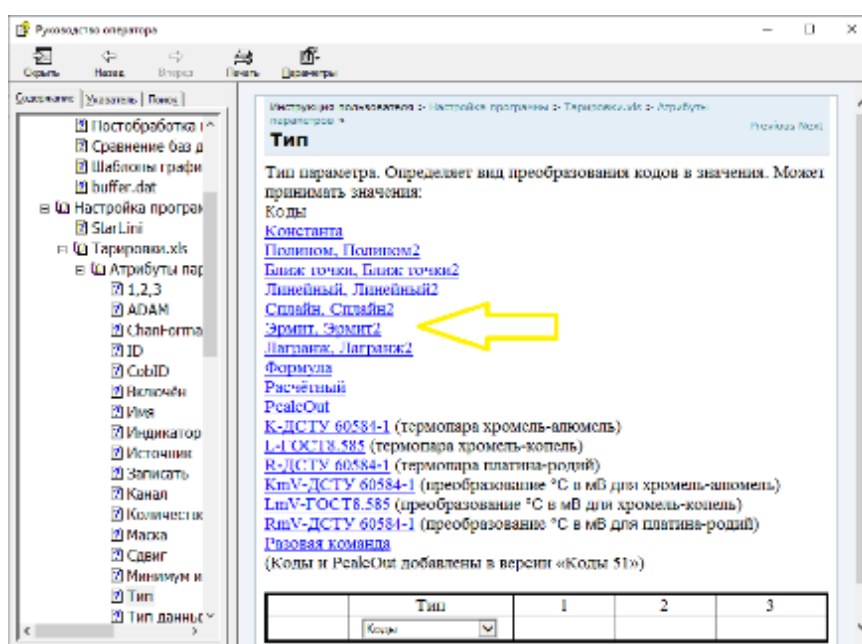


Рисунок 4.22 – Керівництво оператора Ostorpus поточна версії

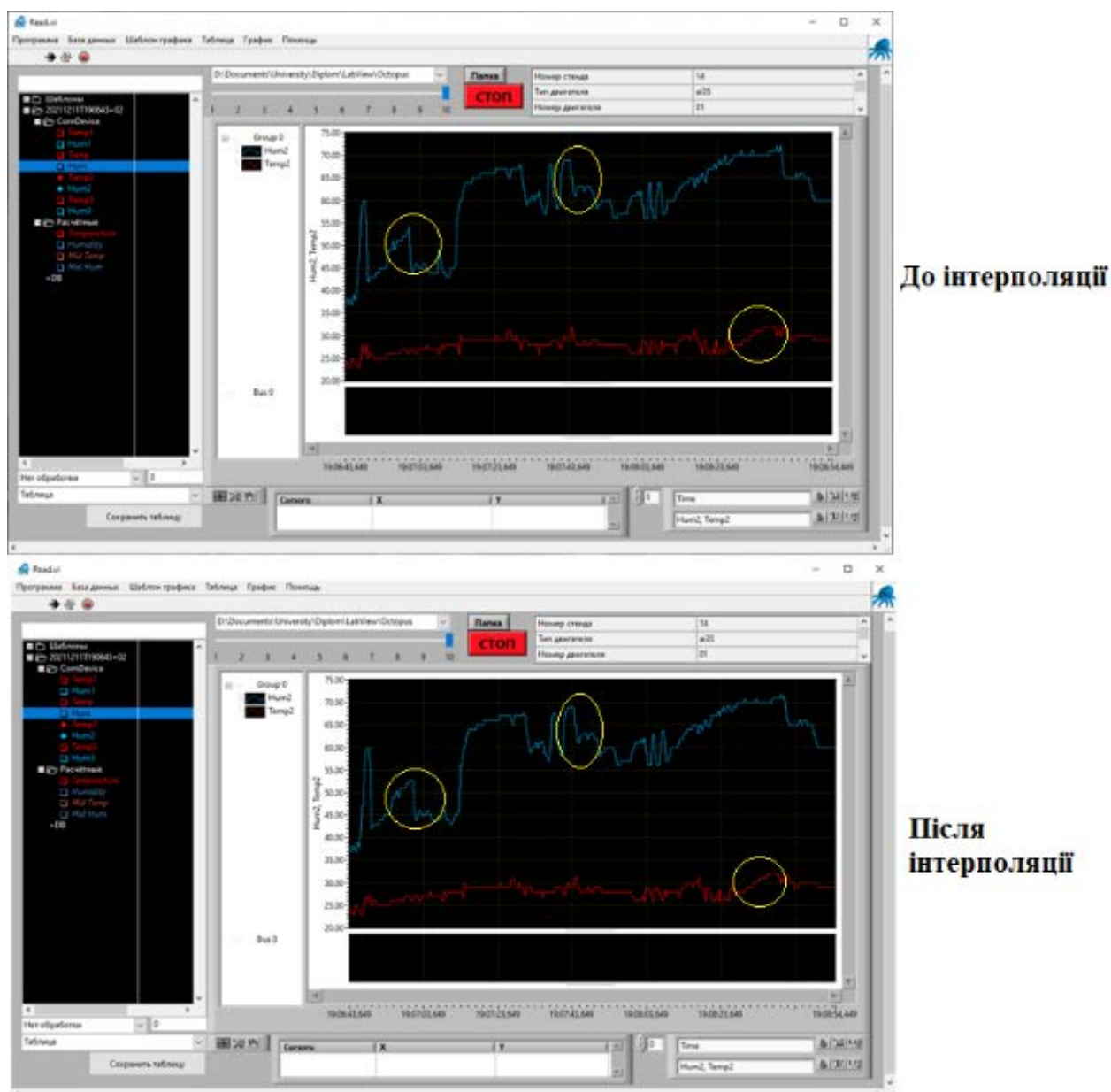


Рисунок 4.23 – Використання методів інтерполяції на стендовій базі даних

Функція усереднення була додана до підпрограми візуалізації Ostorus, результат зображено на рисунках 4.24 – 4.25. Зчитування даних проводилося за допомогою ПЗ Ostorus.

Завдяки функції усереднення даних вдалося вдосконалити процес підготовки даних перед їх інтерполяцією. Завдяки усередненню виключаються помилкові значення в отриманих даних, що підвищує якість набору даних для розрахунків за методами апроксимації.

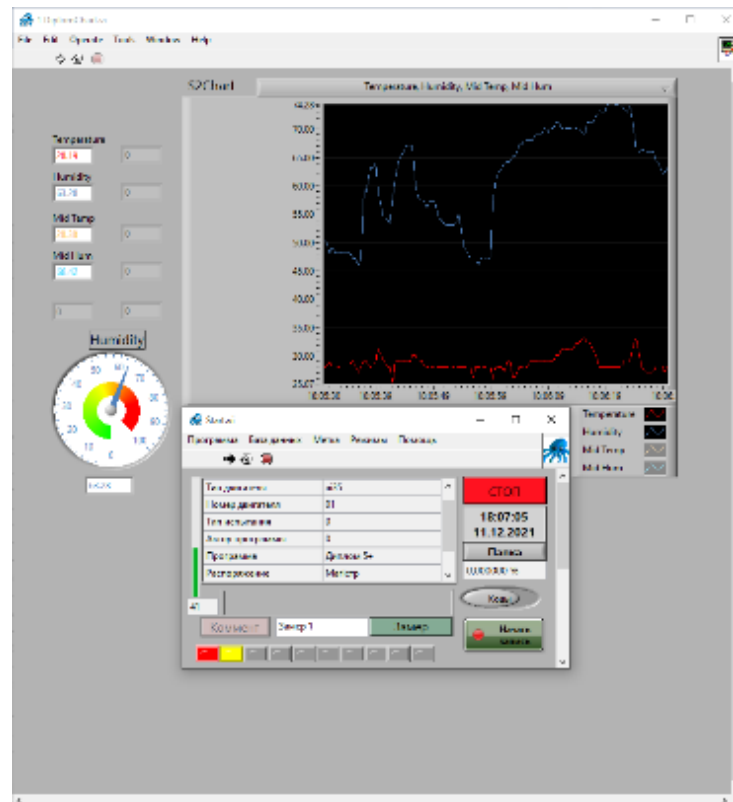


Рисунок 4.24 – Візуалізація даних без усереднення параметрів

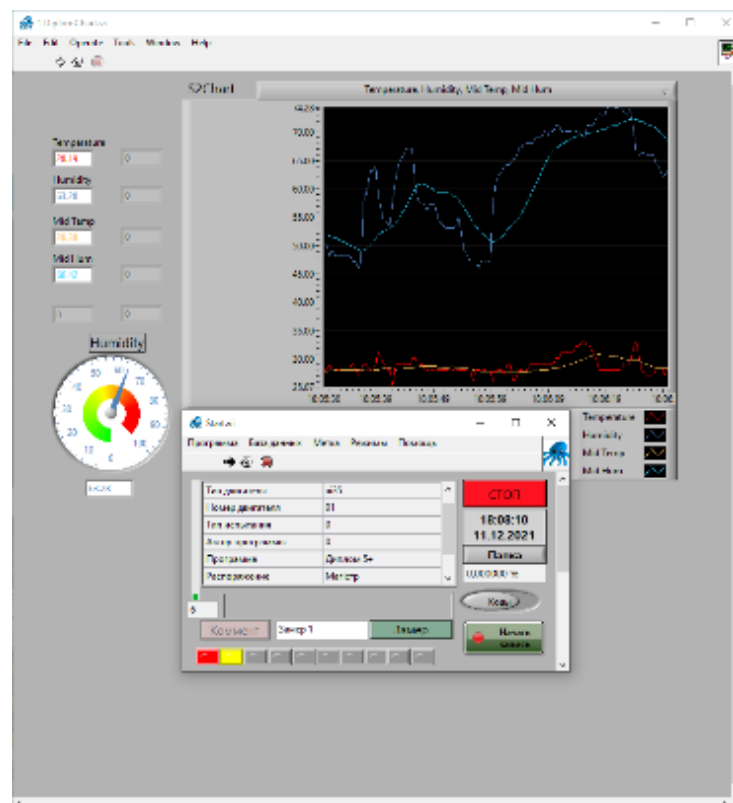


Рисунок 4.25 – Візуалізація даних з функцією усереднення

5 КЕРІВНИЦТВО ПРОГРАМІСТА

5.1 Призначення і умови використання тестової системи збору даних

Функціональне призначення: тестування методів інтерполяції.

Експлуатаційне призначення: тестова система збору даних доцільно використовувати для збору, збереження, візуалізації даних, та обробки вимірних параметрів за методами інтерполяції на випробувальних стендах. ПЗ може працювати з даними через послідовний інтерфейс RS-232, що є розповсюдженим способом передачі даних у багатьох сферах промисловості та у лабораторних дослідженнях.

5.2 Характеристики програми

Програма має 12 функцій, які виконують основний функціонал програми:

- очистка графіка. Очистка історії вимірюваних параметрів;
- усереднення вимірюваних параметрів. Усереднює фіксовану кількість отриманих значень;
- запис до бази даних. Створює новий файл бази даних з розширенням .tdms, конфігурує файл, записує до нього дані випробувань, закриває файл;
- візуалізація даних. Візуалізує вимірювані параметри на графіку;
- зупинка читання даних. Завершує головний цикл підпрограми проведення випробувань;
- виконати пост-обробку даних. Відкриває файл Diplom_Interpolate.vi та запускає його на виконання, після завершення роботи з обробкою даних, закінчує виконання програмного забезпечення для проведення випробувань;

- дата. Використовується для відображення поточного часу;
- відкрити базу даних. Використовується для вибору бази даних для читання та отримання даних з бази даних;
- запуск. Використовується для запуску підпрограми пост-обробки на виконання;
- інтерполяція. Виконує інтерполяцію значень з бази даних;
- апроксимація. Виконує апроксимацію значень з бази даних;
- зупинити виконання. Використовується для зупинки програмного забезпечення.

5.3 Вхідні і вихідні дані

Вхідними даними для тестової системи збору даних є дані з датчиків підключених до системи стендових випробувань, що передають дані за COM портом. У дослідженні використовувалися дані температури та вологості навколишнього середовища.

Вихідними даними для тестової системи збору даних є графіки отриманих параметрів перерахованими за методами інтерполяції та файл бази даних з записаними до нього даними.

5.4 Встановлення програмно-апаратного комплексу

Для встановлення тестової системи збору даних необхідно виконати наступні дії:

- встановити проєкт ПЗ на ПК;
- встановити модуль мови LabVIEW Run-Time engine x86, не нижче 17 (LVRTE2017SP1_f3Patchstd.exe) на сайті National Instruments;
- встановити Microsoft Excel (не нижче 2007);

- підключити лінію передачі вимірюваних даних до ПК через COM порт;
- налаштувати номер порту для читання даних та директорію для запису бази даних.

5.5 Алгоритм роботи розробника з тестовою системою збору даних

Розробник повинен виконувати налаштування програми та додавання нових параметрів до неї у відповідності до цього додатка.

Для того, щоб налаштувати програмне забезпечення необхідно:

- відкрити проєкт програмного забезпечення для проведення випробувань на стендах реального часу (рис. 5.1);
- у проєкті відкрити файл Diplom_MAIN.vi (рис. 5.2);
- відкрити блок діаграму ПЗ як показано на рисунку 5.3, або комбінацією клавіш Ctrl + E;
- на блок діаграмі вибрати необхідний COM порт зі списку (рис. 5.4);

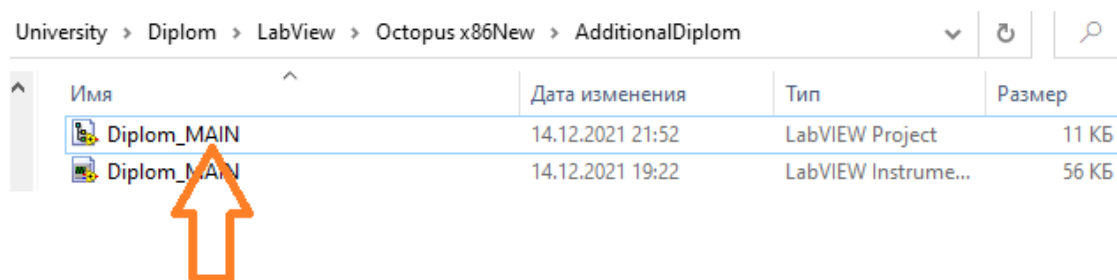


Рисунок 5.1 – Відкрити проєкт

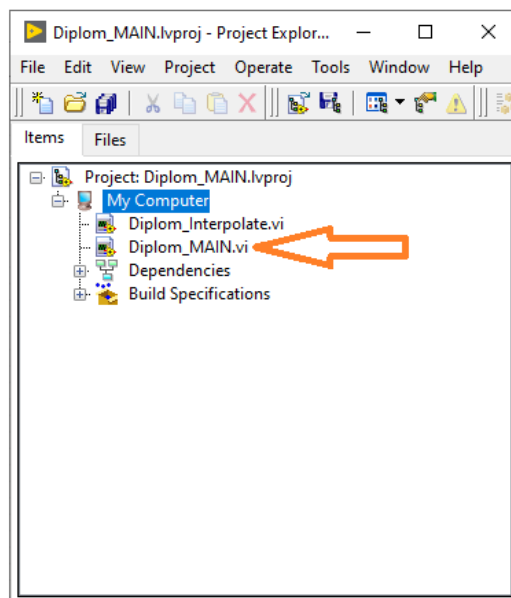


Рисунок 5.2 – Відкрити файл

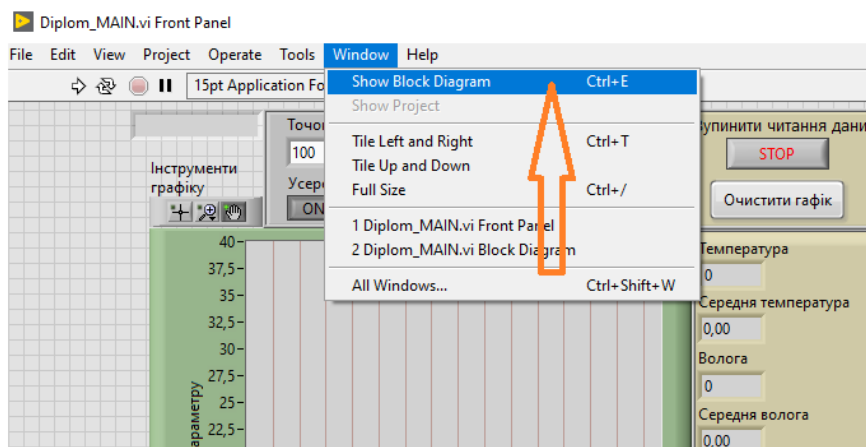


Рисунок 5.3 – Відкрити блок діаграму

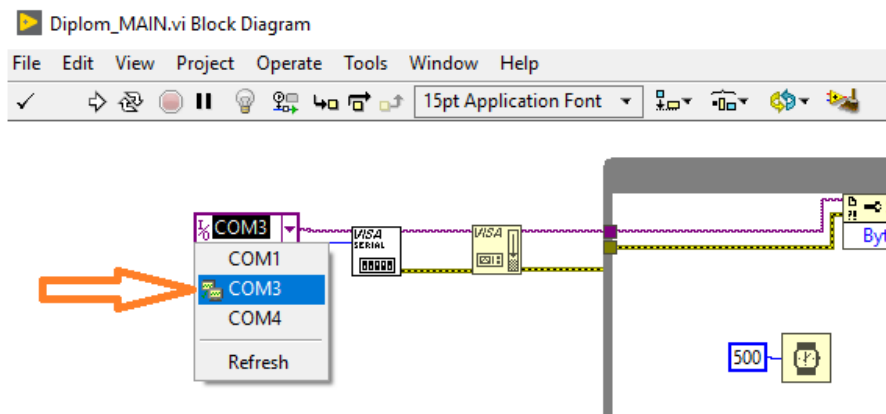


Рисунок 5.4 – Вибрати COM порт

- на блок діаграмі вказати необхідну директорию, в яку буде записуватись файл бази даних (рис. 5.5);
- зберегти проєкт Ctrl + S, закрити проєкт, програмне забезпечення готове до використання.

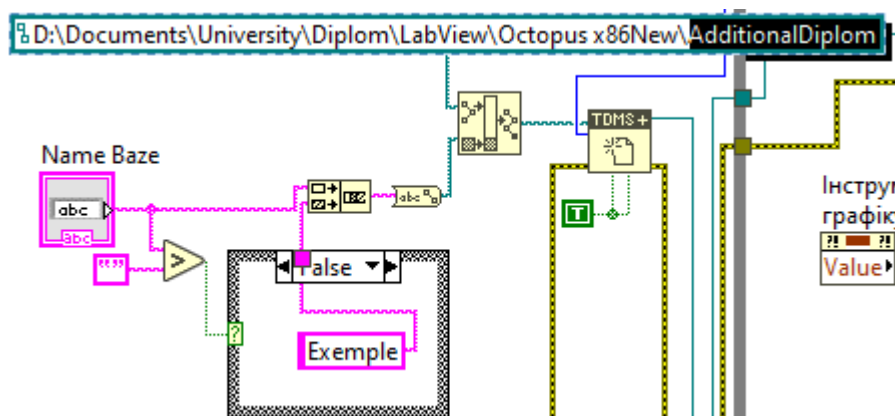


Рисунок 5.5 – Вказати директорию

Для того, щоб додати новий параметр необхідно:

- відкрити проєкт програмного забезпечення для проведення випробувань на стендах реального часу (рис. 5.1);
- у проєкті відкрити файл Diplom_MAIN.vi (рис. 5.2);
- створити у проєкті новий VI та зберегти його(рис. 5.6);

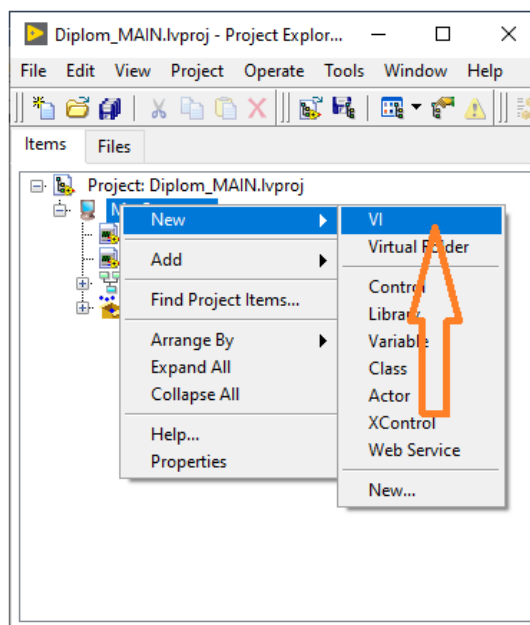


Рисунок 5.6 – Створити новий файл

- у проєкті вибрати пункт «Build Specifications», «Source File Settings», вказати створений VI, далі вибрати «Destination» з випадаючого списку вибрати «Support Directory» (рис. 5.7);
- у проєкті вибрати пункт «Build Specifications», «Source File», вибрати створений VI, натиснути на стрілочку, що вказує праворуч, файл повинен додатися до поля «Always included» (рис. 5.8);

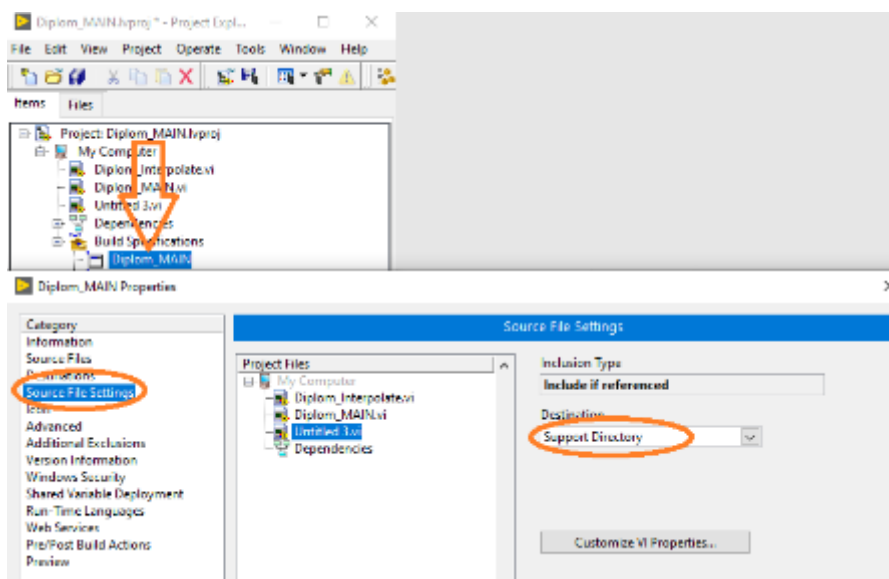


Рисунок 5.7 – Налаштувати Source File Settings

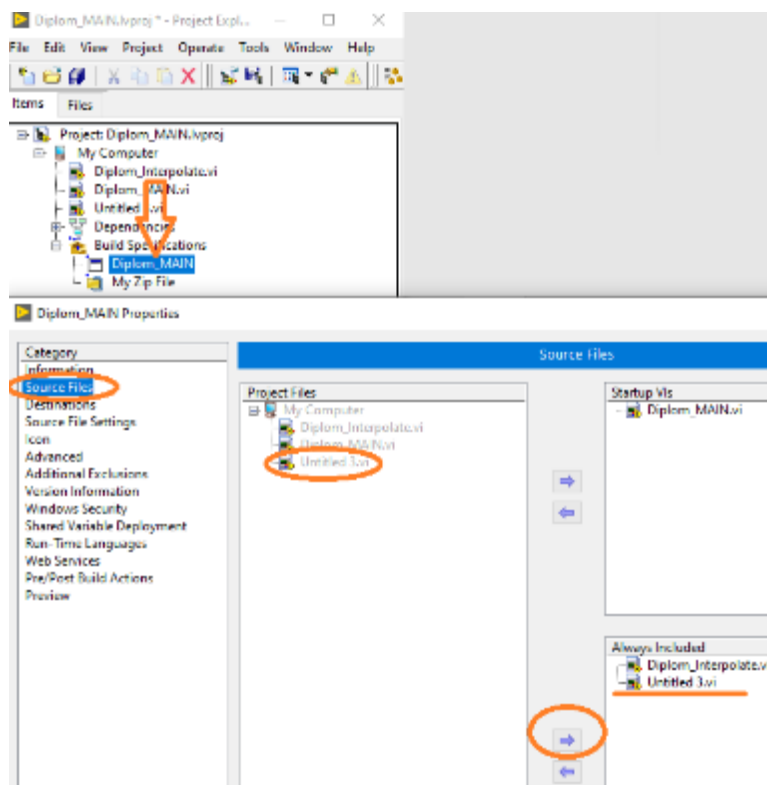


Рисунок 5.8 – Налаштувати Source File

- у тому самому вікні натиснути кнопку «Build» (рис. 5.9). Після цих дій VI буде додано до проєкту, VI можливо буде викликати з Diplom_MAIN.vi та передати на нього, чи з нього значення;
- далі необхідно дослідити який параметр потрібно вимірювати, обрати плату, що зможе виконати це завдання, налаштувати режими роботи плати для конкретного параметра;
- розробити код у створеному VI для читання даних з плати, перевірити коректність отриманих даних за допомогою порівняння з еталонним значенням, перетворити дані в зручний формат для передачі та візуалізації;
- налаштувати канал передачі даних від VI для читання даних з плати до Diplom_MAIN.vi, налаштувати запуск розробленого VI під час запуску ПЗ;
- модифікувати функції модуля візуалізації (очистки, усереднення, відображення на графіку та на індикаторах), для роботи з новим параметром;
- модифікувати групу та назви каналів у модулі збереження даних, додати новий параметр до масиву записуваних параметрів;
- додати новий параметр на графік підпрограми пост-обробки, вивести параметр на індикатори, до модуля розрахунку, розраховані дані вивести на графік та індикатори;
- запустити ПЗ, перевірити виконання всіх функцій;
- зберегти, скомпілювати проєкт та завершити роботу.

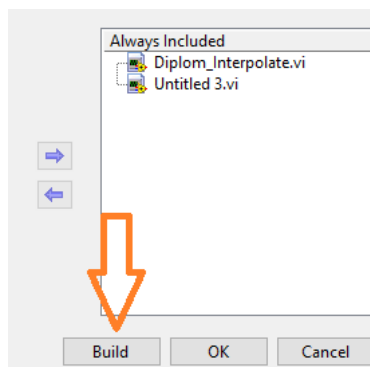


Рисунок 5.9 – Оновити проєкт

5.6 Умови використання тестової системи збору даних

Щоб тестова система збору даних працювала коректно, необхідно дотримуватись таких самих експлуатаційних вимог, які використовуються для персональних комп'ютерів, а саме:

- температура повітря у навколишньому середовищі 0С — +50С;
- відносна вологість, не повинна перевищувати норму (45 – 60%);
- необхідно безперервне підключення до живлення 220 В;
- також, обов'язкове встановлення LabVIEW Run-Time модуля;
- також, обов'язкова наявність Microsoft Excel (не нижче 2007).

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Призначення тестової системи збору даних

Призначається для проведення випробувань на стендових системах реального часу, що забезпечує вищу точність візуалізації стендових параметрів на графіках та діаграмах для покращення якості випробувань.

6.2 Умови використання тестової системи збору даних

Щоб тестова система збору даних працювала правильно, необхідно дотримуватись таких вимог:

- температура повітря у навколишньому середовищі 0С — +50С;
- відносна вологість, не повинна перевищувати норму (45 – 60%);
- необхідно безперервне підключення до живлення 220 В;
- обов'язкове встановлення LabVIEW Run-Time модуля;
- встановлення та первинне налаштування виконує програміст.

6.3 Експлуатація тестової системи збору даних

Для експлуатації тестової системи збору даних користувач повинен дотримуватися наступного порядку дій.

Запустити тестову систему збору даних подвійним кліком миші на піктограму програмного забезпечення у директорії проєкту (рис. 6.1).

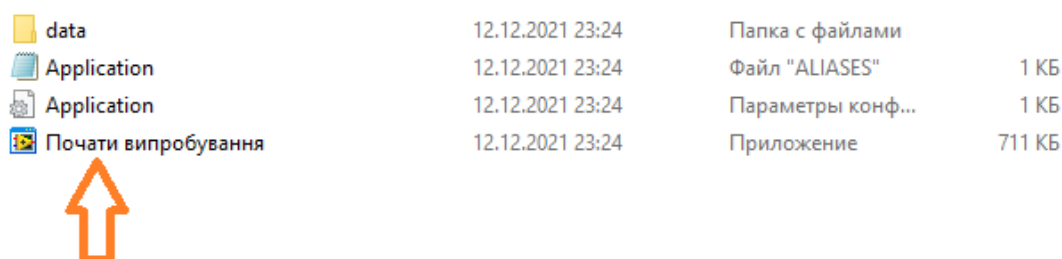


Рисунок 6.1 – Запустити програмне забезпечення

Після відкриття програми налаштувати кількість точок за якими буде проводитися усереднення параметрів, вказати назву для бази даних (рис. 6.2).

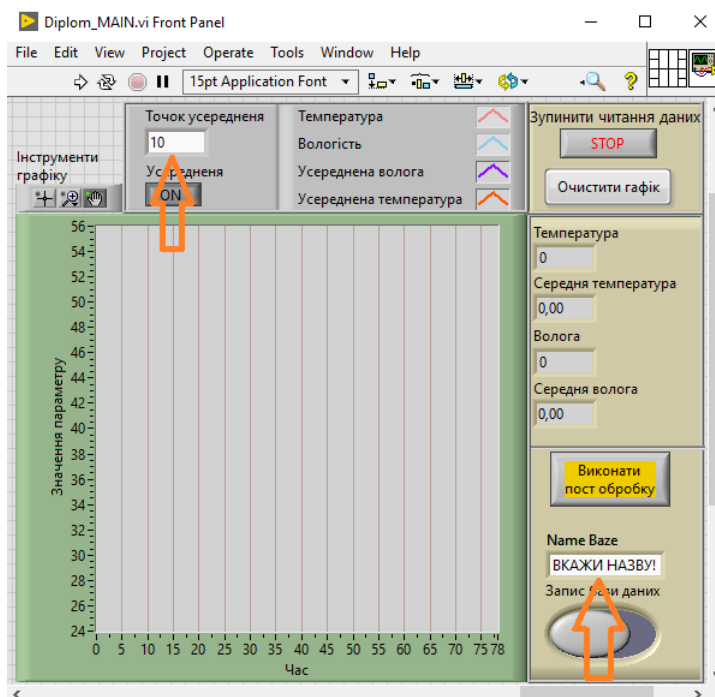


Рисунок 6.2 – Налаштування перед роботою

Після завершення етапу налаштування, необхідно запустити програму на виконання (рис. 6.3).

За командою, запустити запис бази даних, не вимикати запис бази до завершення випробувань (рис. 6.4). Якщо не вказати назву, буде автоматично створена база з назвою Example, з якої можливо дістати записану базу.

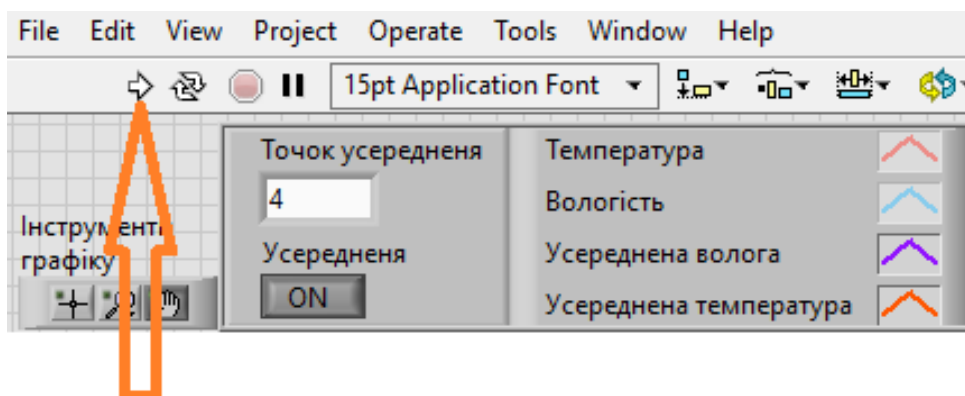


Рисунок 6.3 – Запуск програмного забезпечення

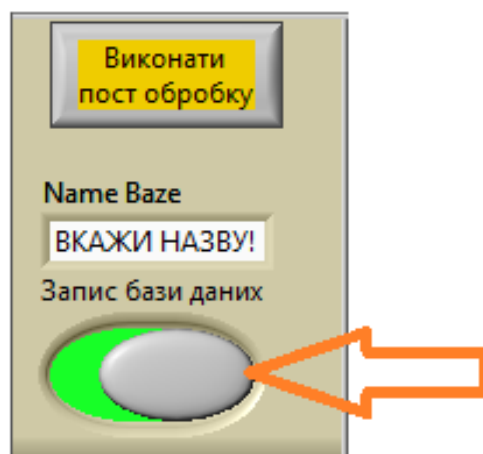


Рисунок 6.4 – Увімкнення запису бази даних

Далі, необхідно перевірити працездатність програмного забезпечення. Якщо графік будується та виводиться на екран, значення параметрів змінюються, то програма працює вірно.

Після завершення випробувань, завершити запис бази даних, зупинити виконання ПЗ відповідною кнопкою (рис. 6.5).

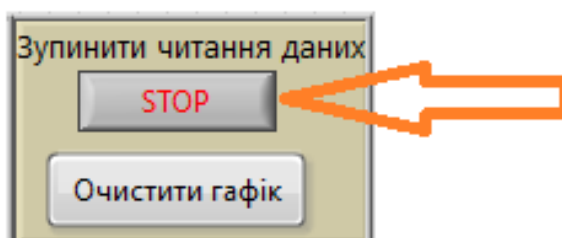


Рисунок 6.5 – Завершити читання даних

Далі можливо закрити програму червоним колом на панелі інструментів, або через хрестик у правому верхньому куті програми. Або, можливо продовжити роботу з програмою, натиснувши кнопку «виконати пост-обробку» (рис. 6.6).

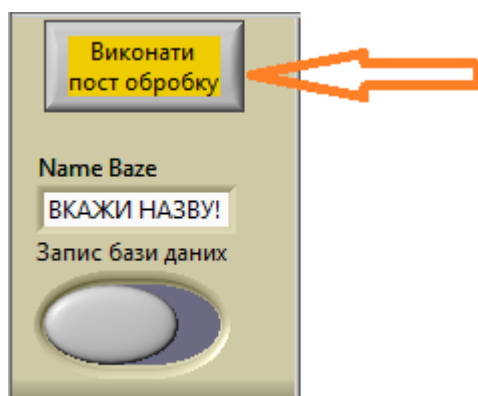


Рисунок 6.6 – Відкрити вікно пост-обробки даних

У відкритому вікні пост-обробки даних, необхідно виконати налаштування перед початком роботи, а саме, вказати набір використовуваних даних, шлях до створеної бази даних, вказати частоту інтерполяції (рис. 6.7).

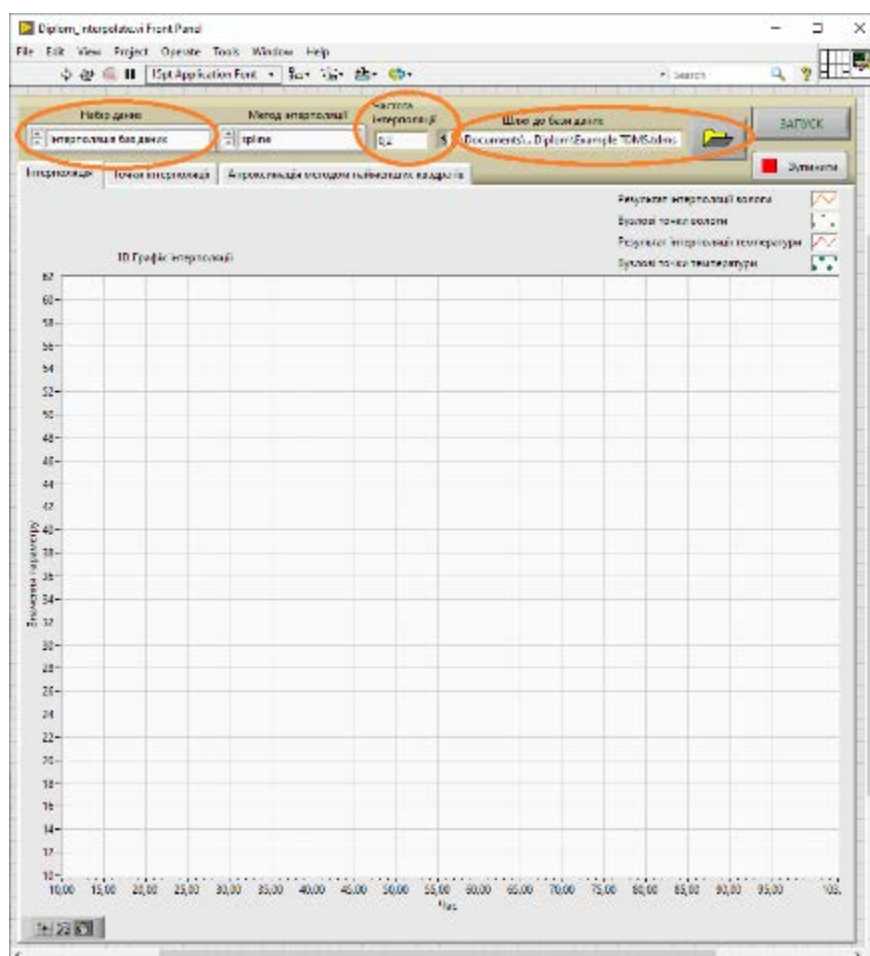


Рисунок 6.7 – Налаштування перед роботою з базою даних

Після налаштування натиснути кнопку «Запуск». Програма буде запущена, під час роботи програми можна змінювати метод інтерполяції, частоту інтерполяції, досліджувану базу даних (рис. 6.8).

Програма для обробки даних може працювати окремо від програми запису даних, можливість запуску однієї з іншої реалізована для економії часу оператора. Також можна використовувати вкладенки у лівому верхньому куті графіку, щоб продивлятися точки інтерполяції, результат методів інтерполяції та апроксимації методом найменших квадратів на основі відкритої бази даних (рис. 6.9).

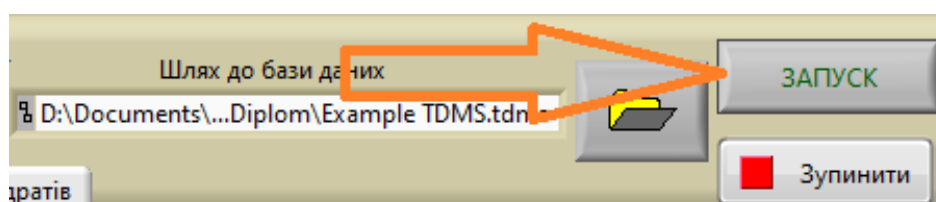


Рисунок 6.8 – Запуск програми обробки даних

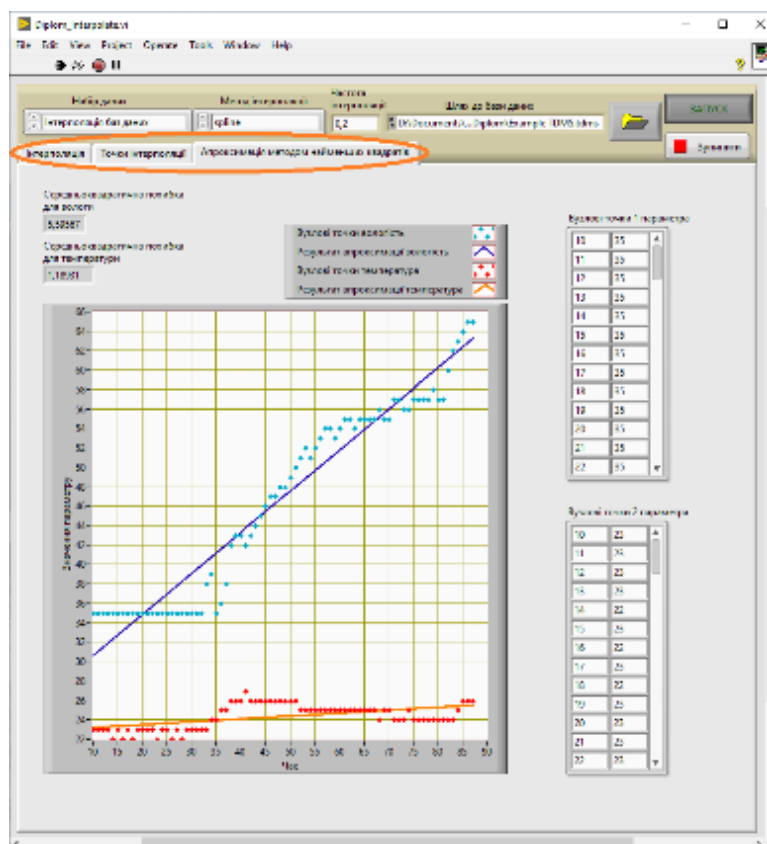


Рисунок 6.9 – Перегляд вкладинок

Після виконання роботи необхідно натиснути кнопку «Зупинити» це завершить роботу з тестовою системою проведення випробувань (рис. 6.10). Закрити вікно хрестиком у правому верхньому куті програми.

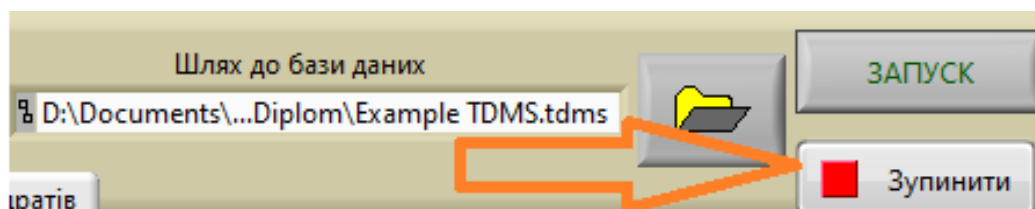


Рисунок 6.10 – Завершити роботу з даними

6.4 Повідомлення тестової системи проведення випробувань

Дані не надходять до програмного забезпечення по СОМ порту (рис. 6.11).

Рішення: перевірити підключення кабелю до USB порту комп'ютера, витягти та вставити кабель у той же роз'єм, перезапустити програму. У разі, якщо не допомогло, звернутися до програміста.

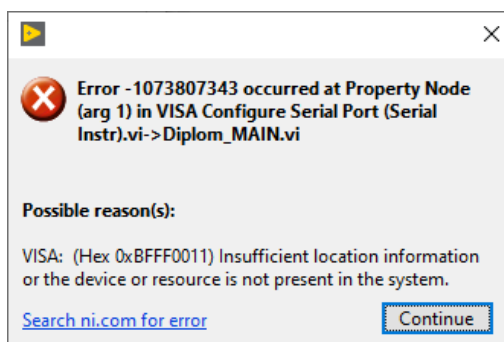


Рисунок 6.11 – Помилка читання даних з порту

Помилка читання бази даних при відкритті бази з програми пост-обробки даних (рис. 6.12). Рішення: не завершений запис даних до бази даних, потрібно закрити вікно роботи з даними, завершити запис до бази, потім

можливо передивлятися записану базу. У разі, якщо не допомогло, звернутися до програміста.

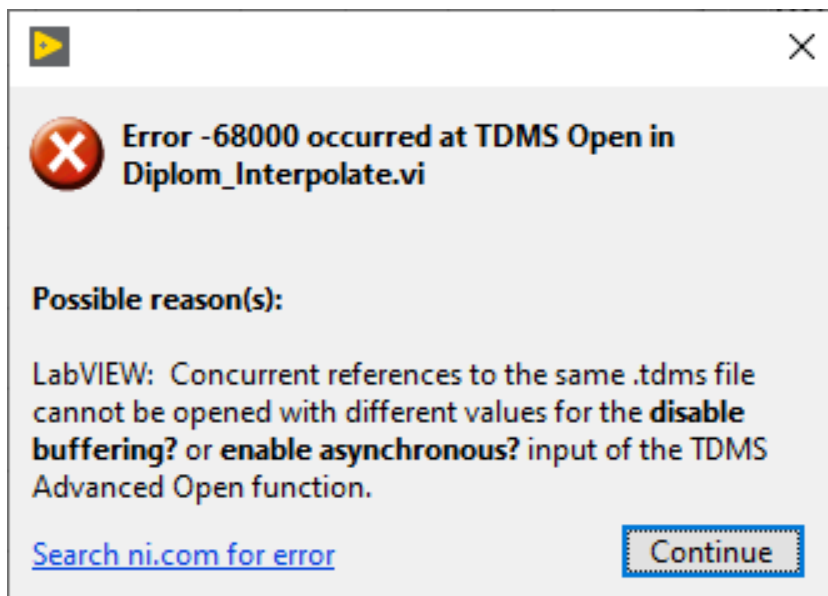


Рисунок 6.12 – Помилка читання бази даних

ВИСНОВКИ

В результаті виконаної роботи було проведено аналіз існуючих на підприємстві стендових систем збору даних, досліджено дані випробувань авіаційних двигунів та методи обробки цих даних, розроблено тестове ПЗ для візуалізації даних випробувань.

В якості мови програмування для розробки ПЗ обрано мову G. В якості середовища розробки було обрано середовище розробки LabVIEW. Було описано, обґрунтовано та продемонстровано основні рішення з вдосконалення візуалізації вимірюваних параметрів у програмному забезпеченні для проведення стендових випробувань Ostopus.

Наукова новизна роботи полягає у тому, що отримали подальший розвиток методи обробки та візуалізації параметрів авіаційних двигунів за рахунок удосконалення функції інтерполяції даних для більш наочного представлення інформації.

Практична цінність роботи полягає в тому, що впровадження розробленого ПЗ дозволить модернізувати підсистему візуалізації Ostopus для правильної інтерпретації даних випробувань фахівцем.

В майбутньому планується розширення функціональних можливостей тестового ПЗ за рахунок додавання списку каналів отримання даних та списку підтримуваних плат, для подальшого встановлення цього ПЗ на всі стенди підприємства.

Результати роботи було представлено на Щорічній науково-практичній конференції НУЗП Тиждень науки – 2021, 19-23 квітня 2021 р.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Важинський, С. Е. Методика та організація наукових досліджень: навч. посіб. / С. Е. Важинський, Т. І. Щербак. – Суми: СумДПУ імені А. С. Макаренка, 2016. – 260 с.
2. Сбор данных там, где начинается аналитика [Электрон. ресурс]. – Режим доступа: <https://www.intel.ru/content/www/ru/ru/analytics/data-collection.html>.
3. Митришкин, Ю. В. Имитационное моделирование систем управления плазмой в токамаке со стендом реального времени / Ю. В. Митришкин, Т.К. Джумакаев, П. С. Коренев // Имитационное моделирование. Теория и практика : 7-я Всероссийская научно-практическая конференция, Москва, 21-23 октября 2015 г. : сб. науч. ст. – Москва: ИПУ РАН, 2015. – С. 224-229.
4. Ратникова, Н. А. Техническое описание изделия ГАММА / Н. А. Ратникова // Актуальные вопросы исследований в авионике: теория, обслуживание, разработки: II Всероссийская научно-практическая конференция, Воронеж, 11–13 февраля 2015 г. : сб. науч. ст. – Воронеж: ВУНЦ ВВС «ВВА», 2015. – С. 21–22.
5. Комиссаров, И.В. Система автоматизированная информационно-измерительная ПБ-2 / И.В. Комиссаров // Описание программного обеспечения №24608189.00094.000 ПА – Запорожье, 2011 – 26 с.
6. Никольский, С. Особенности национальной автоматизации / С. Никольский // Компьютерра. – 2006. – № 6. – С. 16–32.
7. QNX momentics tool suite [Electronic resource]. – Access mode: <https://blackberry.qnx.com/content/dam/bbcomv4/qnx/software-solutions/embedded-software/qnx-momentics-tools-suite/pdf/QNX-Momentics-Product-Brief-v5.pdf>.

8. Octopus: a tool for the application of time-dependent density functional theory / [A. Castro, H. Appel, M. Oliveira et al.] // Physica status solidi. – 2006. – Vol. 243. – P. 2465-2488.
9. Овчинников, С. В. Введение в теорию теплообмена: Теплопроводность в твердых телах : учеб.-метод. пособие / С. В. Овчинников. – Саратов : СГУ имени Н. Г. Чернышевского, 2015. – 104 с.
10. Косенко, С. Новая серия датчиков дифференциального давления от Omron / С. Косенко, О. Михеев// Вестник электроники. – 2014. – № 2. – С. 44–50.
11. Преобразователи давления измерительные АИР – 10Н [Электрон. ресурс]. – Режим доступа: <http://tekua.com.ua/uploads/2/files/1c1749eec5817e355c28b469b1346947.pdf>
12. Датчики давления МИДА-13П-М [Электрон. ресурс]. – Режим доступа: <http://www.midaus.com/images/docs/13pm.pdf>
13. МИДА-ДИ-13П Датчики избыточного давления микроэлектронные [Электрон. ресурс]. – Режим доступа: <http://xn--90anlhdbcfdd.xn--80adxhks/catalog/mida-di-13p/>
14. Иванов, В.И. Аналого – цифровые и цифроаналоговые преобразователи: методические указания к выполнению лабораторной работы по дисциплине «Электротехника, электроника и схемотехника» / В.И. Иванов. – Курск: ЮЗГУ, 2016. –19 с.
15. Блок сбора и передачи информации БСПИ-3Л-МР-Н [Электрон. ресурс]. – Режим доступа: https://www.electronpribor.ru/files/products/to_bspi_3l_mr_n.pdf
16. PCI-1747U [Electronic resource]. – Access mode: https://www.advantech.ru/products/1-2mlk8i/pci-1747u/mod_80e54ea9-46aa-4dc0-949c-1cd3cbde172c
17. PCI-1747U product manual [Electronic resource]. – Access mode: <https://www.proxis.ua/files/documents/PCI-1747U-UM.pdf>

18. Marz, N. Big data: principles and best practices of scalable realtime data systems / N. Marz, J. Warren. – Frechen : MITP, 2016. – 328 p.
19. Верес, О. М. Класифікація методів аналізу великих даних / О. М. Верес, Р. М. Оливко // Вісник Національного університету "Львівська політехніка". – 2017. – № 827. – С. 84–92.
20. Степанов, М.М. Аппроксимация функций : методические указания к лабораторным работам по дисциплине «Информатика» / М. М. Степанов, Н. Н. Потапова, Т.В. Ерещенко. – Волгоград : ВолгГАСУ, 2012. – 34 с.
21. Методические указания к решению задач по интерполяции функций : уч-метод пособие / [А.Л. Калашников, Г.В. Потёмин, А.М. Федоткин и др.]. – Нижний Новгород: Нижегородский госуниверситет, 2016. – 36 p.
22. Colalongo, L. Hermite interpolation method (HiM): compact surface potential MOSFET model based on the Hermite polynomial / L. Colalongo, A. Richelli, Z. Kovacs-Vajna // *Electronics Letters*. – 2016. – Vol. 52. – P. 1490-1492.
23. Колкер, А.Б. Обоснование выбора программного обеспечения для робототехники / А.Б. Колкер, Д.А. Ливенец, А.И. Кошелева // *Автоматика і програмна інженерія*. – 2012. – №1. – С. 64.
24. Elali, T. Discrete systems and digital signal processing with MATLAB / T. Elali. – Hoboken : CRC Press, 2016. – 620 p.
25. Laboratory virtual instrument engineering workbench (LABVIEW) / [S. R. Kurkute, K. P. Arbuj, C. R. Dargude et al.] // *International Journal of Modern Embedded System*. – 2017. – Vol. 5. – P. 17–20.
26. Travis, J. LabVIEW for everyone : graphical programming made easy and fun / J. Travis, K. Jim. – Upper Saddle River, NJ: Prentice Hall, 2006. – 1032 p.
27. How do I use LabVIEW to measure physical systems with sensors or actuators? [Electronic resource]. – Access mode: <https://www.ni.com/ru-ru/shop/labview/how-do-i-use-labview-to-measure-physical-systems.html>.

28. Hardware integration with LabVIEW [Electronic resource]. – Access mode: <https://www.ni.com/ru-ru/innovations/white-papers/13/hardware-integration-with-ni-labview.html>.
29. Blume, P. The LabVIEW style book / P. Blume. – The LabVIEW style book : Prentice Hall, 2010. – 372 p.

ДОДАТОК А
Технічне завдання

Вступ

Назва продукту: тестове програмне забезпечення для проведення випробувань на стендових системах збору даних реального часу.

Програмне забезпечення забезпечить кращу візуалізацію вимірюваних параметрів, таким чином підвищить точність проведення випробувань авіаційних двигунів.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему «Дослідження та розробка програмного забезпечення для системи стендових випробувань авіаційних двигунів», затверджене наказом по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Функціональне призначення: тестування методів інтерполяції.

Експлуатаційне призначення: тестова система збору даних доцільно використовувати для збору, збереження, візуалізації даних, та обробки вимірюваних параметрів за методами інтерполяції на випробувальних стендах. ПЗ може працювати з даними через послідовний інтерфейс RS-232, що є розповсюдженим способом передачі даних у багатьох сферах промисловості та у лабораторних дослідженнях.

A.3 Вимоги до програми чи програмному виробу

A.3.1 Вимоги до функціональних характеристик

Розроблюване ПЗ повинно виконувати такі функції:

- можливість зчитування та збереження даних випробування;
- можливість візуалізації вимірюваних параметрів;
- можливість проведення обробки записаної бази даних за методами апроксимації;
- можливість усунення похибок датчиків на графіку під час проведення випробувань за рахунок усереднення параметрів.

A.3.2 Вимоги до нефункціональних характеристик

Розроблюване ПЗ повинно дотримуватися таких умов:

- програмне забезпечення повинно працювати без помилок;
- програмне забезпечення повинно бути універсальним, з можливістю використання у різних галузях;
- можливість додавання нових параметрів для читання, збереження, візуалізації та обробки;
- просте налаштування програми користувачем, в ідеальному випадку – без втручання програміста;
- повинно завантажуватися на офісних комп'ютерах;
- низьке енергоспоживання.

A.3.3 Умови експлуатації

Умови експлуатації наступні:

- температура повітря у навколишньому середовищі 0С — +50С;
- відносна вологість, не повинна перевищувати норму (45 – 60%);

- необхідно безперервне підключення до живлення 220 В;
- обов'язкове встановлення LabVIEW Run-Time модуля;
- встановлення та первинне налаштування виконує програміст.

A.3.4 Вимоги маркування й упакування

Тестове програмне забезпечення поставляється на флешці або CD диску, та поширюється мережею інтернет.

A.3.5 Вимоги транспортування й збереження

Вимоги до транспортування та збереження співпадають з вимогами до транспортування та збереження електричних приладів.

A.4 Вимоги до програмної документації

Програмна документація повинна бути написана українською мовою та складатися з таких документів:

- технічне завдання;
- керівництво користувача;
- керівництво програміста.

A.5 Порядок контролю й приймання

Дипломний проєкт допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедрою і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Заголовный файл

```

<Item Name="My Computer" Type="My Computer">
    <Property                                Name="server.app.propertiesEnabled"
Type="Bool">true</Property>
    <Property                                Name="server.control.propertiesEnabled"
Type="Bool">true</Property>
    <Property Name="server.tcp.enabled" Type="Bool">>false</Property>
    <Property Name="server.tcp.port" Type="Int">0</Property>
    <Property Name="server.tcp.serviceName" Type="Str">My Computer/VI
Server</Property>
    <Property Name="server.tcp.serviceName.default" Type="Str">My
Computer/VI Server</Property>
    <Property                                Name="server.vi.callsEnabled"
Type="Bool">true</Property>
    <Property                                Name="server.vi.propertiesEnabled"
Type="Bool">true</Property>
    <Property                                Name="specify.custom.address"
Type="Bool">>false</Property>
    <Item Name="Diplom_Interpolate.vi" Type="VI"
URL="../../Diplom_Interpolate.vi"/>
    <Item Name="Diplom_MAIN.vi" Type="VI" URL="../../Diplom_MAIN.vi"/>
    <Item Name="Dependencies" Type="Dependencies">
        <Item Name="vi.lib" Type="Folder">
            <Item Name="BuildHelpPath.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/BuildHelpPath.vi"/>
            <Item Name="Check Special Tags.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Check Special Tags.vi"/>
            <Item Name="Clear Errors.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Clear Errors.vi"/>
            <Item Name="Convert property node font to graphics
font.vi" Type="VI" URL="/&lt;vilib&gt;/Utility/error.llb/Convert property node
font to graphics font.vi"/>
            <Item Name="Details Display Dialog.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Details Display Dialog.vi"/>
            <Item Name="DialogType.ctl" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/DialogType.ctl"/>
            <Item Name="DialogTypeEnum.ctl" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/DialogTypeEnum.ctl"/>
            <Item Name="Error Code Database.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Error Code Database.vi"/>

```

```

        <Item          Name="ErrWarn.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/ErrWarn.ctl"/>
        <Item          Name="eventvkey.ctl"        Type="VI"
URL="/&lt;vilib&gt;/event_ctls.llb/eventvkey.ctl"/>
        <Item          Name="Find          Tag.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Find Tag.vi"/>
        <Item  Name="Format  Message  String.vi"  Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Format Message String.vi"/>
        <Item  Name="General  Error  Handler  Core  CORE.vi"
Type="VI"  URL="/&lt;vilib&gt;/Utility/error.llb/General Error Handler Core
CORE.vi"/>
        <Item  Name="General  Error  Handler.vi"  Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/General Error Handler.vi"/>
        <Item  Name="Get  String  Text  Bounds.vi"  Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Get String Text Bounds.vi"/>
        <Item  Name="Get  Text  Rect.vi"          Type="VI"
URL="/&lt;vilib&gt;/picture/picture.llb/Get Text Rect.vi"/>
        <Item          Name="GetHelpDir.vi"        Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/GetHelpDir.vi"/>
        <Item  Name="GetRTHostConnectedProp.vi"    Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/GetRTHostConnectedProp.vi"/>
        <Item Name="Longest Line Length in Pixels.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Longest Line Length in Pixels.vi"/>
        <Item          Name="LVBoundsTypeDef.ctl"    Type="VI"
URL="/&lt;vilib&gt;/Utility/miscctls.llb/LVBoundsTypeDef.ctl"/>
        <Item          Name="LVRectTypeDef.ctl"      Type="VI"
URL="/&lt;vilib&gt;/Utility/miscctls.llb/LVRectTypeDef.ctl"/>
        <Item          Name="NI_AALBase.lvlib"      Type="Library"
URL="/&lt;vilib&gt;/Analysis/NI_AALBase.lvlib"/>
        <Item          Name="NI_AALPro.lvlib"       Type="Library"
URL="/&lt;vilib&gt;/Analysis/NI_AALPro.lvlib"/>
        <Item          Name="NI_Gmath.lvlib"       Type="Library"
URL="/&lt;vilib&gt;/gmath/NI_Gmath.lvlib"/>
        <Item          Name="NI_PtbyPt.lvlib"      Type="Library"
URL="/&lt;vilib&gt;/ptbypt/NI_PtbyPt.lvlib"/>
        <Item  Name="Not  Found  Dialog.vi"        Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Not Found Dialog.vi"/>
        <Item Name="Search and Replace Pattern.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Search and Replace Pattern.vi"/>
        <Item  Name="Set  Bold  Text.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Set Bold Text.vi"/>

```

```

        <Item      Name="Set      String      Value.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Set String Value.vi"/>
        <Item      Name="Simple      Error      Handler.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Simple Error Handler.vi"/>
        <Item              Name="subTimeDelay.vi"              Type="VI"
URL="/&lt;vilib&gt;/express/express                                execution
control/TimeDelayBlock.llb/subTimeDelay.vi"/>
        <Item              Name="TagReturnType.ctl"              Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/TagReturnType.ctl"/>
        <Item      Name="Three Button Dialog CORE.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Three Button Dialog CORE.vi"/>
        <Item      Name="Three Button Dialog.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Three Button Dialog.vi"/>
        <Item      Name="Trim      Whitespace.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Trim Whitespace.vi"/>
        <Item      Name="VISA Configure Serial Port"      Type="VI"
URL="/&lt;vilib&gt;/Instr/_visa.llb/VISA Configure Serial Port"/>
        <Item      Name="VISA Configure Serial Port (Instr).vi"
Type="VI"      URL="/&lt;vilib&gt;/Instr/_visa.llb/VISA Configure Serial Port
(Instr).vi"/>
        <Item      Name="VISA Configure Serial Port (Serial
Instr).vi"      Type="VI"      URL="/&lt;vilib&gt;/Instr/_visa.llb/VISA Configure Serial
Port (Serial Instr).vi"/>
        <Item              Name="whitespace.ctl"              Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/whitespace.ctl"/>
        </Item>
        <Item              Name="lvanlys.dll"              Type="Document"
URL="/&lt;resource&gt;/lvanlys.dll"/>
        </Item>
        <Item      Name="Build Specifications"      Type="Build">
        <Item      Name="Diplom_MAIN"      Type="EXE">
        <Property              Name="App_copyErrors"
Type="Bool">true</Property>
        <Property              Name="App_INI_aliasGUID"
Type="Str">{B3894F7B-5461-4BE8-A1AB-3DA6DA9FFB72}</Property>
        <Property      Name="App_INI_GUID"      Type="Str">{0A17884B-
AE58-4E6D-AC30-BBAEB6431123}</Property>
        <Property              Name="App_serverConfig.httpPort"
Type="Int">8002</Property>
        <Property              Name="Bld_autoIncrement"
Type="Bool">true</Property>

```

```

        <Property                                Name="Bld_buildCacheID"
Type="Str">{D04FCA44-83BC-4E92-BC5D-96C341C60FD5}</Property>
        <Property                                Name="Bld_buildSpecName"
Type="Str">Diplom_MAIN</Property>
        <Property                                Name="Bld_excludeInlineSubVIs"
Type="Bool">true</Property>
        <Property                                Name="Bld_excludeLibraryItems"
Type="Bool">true</Property>
        <Property                                Name="Bld_excludePolymorphicVIs"
Type="Bool">true</Property>
        <Property                                Name="Bld_localDestDir"
Type="Path">../builds/NI_AB_PROJECTNAME</Property>
        <Property                                Name="Bld_localDestDirType"
Type="Str">relativeToCommon</Property>
        <Property                                Name="Bld_modifyLibraryFile"
Type="Bool">true</Property>
        <Property                                Name="Bld_previewCacheID"
Type="Str">{70D80FAD-50B6-480F-9B7F-95FC2FBB59C8}</Property>
        <Property                                Name="Bld_version.build"
Type="Int">2</Property>
        <Property                                Name="Bld_version.major"
Type="Int">1</Property>
        <Property                                Name="Destination[0].destName"
Type="Str">Application.exe</Property>
        <Property                                Name="Destination[0].path"
Type="Path">../builds/NI_AB_PROJECTNAME/Application.exe</Property>
        <Property                                Name="Destination[0].preserveHierarchy"
Type="Bool">true</Property>
        <Property                                Name="Destination[0].type"
Type="Str">App</Property>
        <Property                                Name="Destination[1].destName"
Type="Str">Support Directory</Property>
        <Property                                Name="Destination[1].path"
Type="Path">../builds/NI_AB_PROJECTNAME/data</Property>
        <Property                                Name="DestinationCount"
Type="Int">2</Property>
        <Property                                Name="Source[0].itemID"
Type="Str">{637E2267-E507-45D5-8041-8D14BCA394E9}</Property>
        <Property                                Name="Source[0].type"
Type="Str">Container</Property>
        <Property                                Name="Source[1].destinationIndex"
Type="Int">0</Property>

```

```

        <Property      Name="Source[1].itemID"      Type="Ref">/My
Computer/Diplom_MAIN.vi</Property>
        <Property      Name="Source[1].properties[0].type"
Type="Str">Run when opened</Property>
        <Property      Name="Source[1].properties[0].value"
Type="Bool">false</Property>
        <Property      Name="Source[1].propertiesCount"
Type="Int">1</Property>
        <Property      Name="Source[1].sourceInclusion"
Type="Str">TopLevel</Property>
        <Property      Name="Source[1].type"
Type="Str">VI</Property>
        <Property      Name="Source[2].destinationIndex"
Type="Int">1</Property>
        <Property      Name="Source[2].itemID"      Type="Ref">/My
Computer/Diplom_Interpolate.vi</Property>
        <Property      Name="Source[2].sourceInclusion"
Type="Str">Include</Property>
        <Property      Name="Source[2].type"
Type="Str">VI</Property>
        <Property Name="SourceCount" Type="Int">3</Property>
        <Property      Name="TgtF_fileDescription"
Type="Str">Diplom_MAIN</Property>
        <Property      Name="TgtF_internalName"
Type="Str">Diplom_MAIN</Property>
        <Property      Name="TgtF_legalCopyright"
Type="Str">Copyright © 2021 </Property>
        <Property      Name="TgtF_productName"
Type="Str">Diplom_MAIN</Property>
        <Property      Name="TgtF_targetfileGUID"
Type="Str">{21E54967-BE66-435B-9D94-F01ABFAA46A2}</Property>
        <Property      Name="TgtF_targetfileName"
Type="Str">Application.exe</Property>
        <Property      Name="TgtF_versionIndependent"
Type="Bool">true</Property>
    </Item>
    <Item Name="My Zip File" Type="Zip File">
        <Property      Name="Absolute[0]"
Type="Bool">false</Property>
        <Property      Name="BuildName"      Type="Str">My      Zip
File</Property>
        <Property Name="Comments" Type="Str"></Property>

```

```

        <Property                                Name="DestinationID[0]"
Type="Str">{61BE4129-1157-4D19-B6C5-C455BF268392}</Property>
        <Property                                Name="DestinationItemCount"
Type="Int">1</Property>
        <Property                                Name="DestinationName[0]"
Type="Str">Destination Directory</Property>
        <Property                                Name="IncludedItemCount"
Type="Int">1</Property>
        <Property    Name="IncludedItems[0]"    Type="Ref">/My
Computer</Property>
        <Property                                Name="IncludeProject"
Type="Bool">true</Property>
        <Property                                Name="Path[0]"
Type="Path">../../builds/Diplom_MAIN/My Zip File/Diplom_MAIN.zip</Property>
        <Property                                Name="ZipBase"
Type="Str">NI_zipbasedefault</Property>
    </Item>
</Item>
</Item>
</Project>

```

Б.2 Фрагмент коду програмного забезпечення

```

<Project Type="Project" LVVersion="17008000">
    <Item Name="My Computer" Type="My Computer">
        <Property                                Name="seer.app.propertiesEnabled"
Type="Bool">true</Property>
        <Property                                Name="seer.control.propertiesEnabled"
Type="Bool">true</Property>
        <Property Name="sver.tcp.enabled" Type="Bool">false</Property>
        <Property Name="sver.tcp.port" Type="Int">0</Property>
        <Property Name="sever.tcp.servceName" Type="Str">My Computer/VI
Server</Property>
        <Property    Name="erver.tcp.seviceNme.default"    Type="Str">My
Computer/VI Serer</Property>
        <Property Name="server.vi.callsnabled" Type="Bool">true</Property>
        <Property                                Name="server.vi.proprtiesEabled"
Type="Bool">true</Property>
        <Property Name="specify.cstom.address" Type="Bool">false</Property>
        <Item    Name="Diplom_Interpolate.vi"    Type="VI"
URL="../../Diplom_Interpolate.vi"/>
    </Item>
</Project>

```

```

        <Item Name="Deenencies" Type="Dependencies">
            <Item Name="vi.lib" Type="Folder">
                <Item          Name="BuidHelpPth.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/BuidelpPath.vi"/>
                <Item      Name="Check      Special      Tags.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Check Special Tags.vi"/>
                <Item          Name="Clear          Errors.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Clear Errrs.vi"/>
                <Item Name="Cnvert propety node font to graphics
font.vi" Type="VI" URL="/&lt;vilib&gt;/Utility/error.llb/Conert property node
font to grapics font.vi"/>
                <Item      Name="Detils      Display      Dialog.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Details Display Dialog.vi"/>
                <Item          Name="DialogType.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/DialogType.ctl"/>
                <Item          Name="DialogTeEnum.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/DgTypeEnum.ctl"/>
                <Item      Name="Error      Code      Datse.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Error Code Dase.vi"/>
                <Item          Name="Errrn.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/ErrWarn.ctl"/>
                <Item          Name="evevkey.ctl"          Type="VI"
URL="/&lt;vilib&gt;/event_ctls.llb/eventvkey.ctl"/>
                <Item      Name="Find          Tag.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Find Tag.vi"/>
                <Item      Name="Format      Message      String.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Format Message String.vi"/>
                <Item Name="General Error Haler Core CRE.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/General Error Hanr Core CORE.vi"/>
                <Item      Name="General      Error      Hanr.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Genl Error Handler.vi"/>
                <Item      Name="Get      Strng      Text      Bonds.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Get Sring Text Bonds.vi"/>
                <Item      Name="Get      Text      Rect.vi"          Type="VI"
URL="/&lt;vilib&gt;/picture/picture.llb/Get Text Ret.vi"/>
                <Item          Name="GetelDir.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/GetelpDr.vi"/>
                <Item          Name="GetRTstConctedrop.vi"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/GetRTHostonecteProp.vi"/>
                <Item Name="Longest Line Legth in Piels.vi" Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Longest Line Length in Pixels.vi"/>

```

```

        <Item          Name="VBundsTpeDef.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/miscctls.llb/LBundsTpeDef.ctl"/>
        <Item          Name="LVectTpeDef.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/miscctls.llb/LRectTypeef.ctl"/>
        <Item          Name="NI_ALase.lvlib"          Type="Library"
URL="/&lt;vilib&gt;/Analysis/NI_AALBase.lvlib"/>
        <Item          Name="NI_ALPro.llib"          Type="Library"
URL="/&lt;vilib&gt;/Analysis/NI_AALPro.lvlib"/>
        <Item          Name="I_Gath.lvlib"          Type="Library"
URL="/&lt;vilib&gt;/gmath/NI_Gmath.lvlib"/>
        <Item      Name="Not      Found      Diaog.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Not Fond Dialog.vi"/>
        <Item      Name="Search and Replace Pattern.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.lb/Sarch and lace ttern.vi"/>
        <Item      Name="Set      Bold      Text.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Set Bold Text.vi"/>
        <Item      Name="Set      Strg      Val.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Set String Value.vi"/>
        <Item      Name="Sile      Error      Haer.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Simple Error Handler.vi"/>
        <Item          Name="subTay.vi"          Type="VI"
URL="/&lt;vilib&gt;/expss/expss
control/TimeDelayBlock.llb/subTiay.vi"/>
        <Item          Name="TagReturnTpe.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/TagrnTpe.ctl"/>
        <Item      Name="Three Bon Dialog CORE.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Three Bun Dia CORE.vi"/>
        <Item      Name="Three Butn Dialog.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Three Button Dialog.vi"/>
        <Item      Name="Trim      Whitace.vi"      Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/Trim Whitce.vi"/>
        <Item          Name="white.ctl"          Type="VI"
URL="/&lt;vilib&gt;/Utility/error.llb/whipace.ctl"/>
        </Item>
        <Item          Name="lvnlys.dll"          Type="Document"
URL="/&lt;resource&gt;/lvnlys.dll"/>
        </Item>
        <Item      Name="Build Specifications"      Type="Buld">
            <Item      Name="Diplom_Interpolate"      Type="EXE">
                <Property          Name="App_copyErrors"
Type="Bool">true</Property>

```

```

        <Property                                Name="App_INI_aliasGUID"
Type="Str">{214A8C31-0398-479F-9178-1D9792A}</Property>
        <Property  Name="App_INI_GUID"  Type="Str">{45B9B194-
682F-4A85-86C3-27279A926B37}</Property>
        <Property                                Name="App_serverConfig.httpPort"
Type="It">8002</Property>
        <Property                                Name="Bld_atoIncement"
Type="Bool">true</Property>
        <Property  Name="Bld_bCacheID"  Type="Str">{25B6BC74-
1A18-43A5-A84B-9029E437C0D7}</Property>
        <Property                                Name="Bld_buame"
Type="Str">Diplom_Interpolate</Property>
        <Property                                Name="Bld_excludSubVIs"
Type="Bool">true</Property>
        <Property                                Name="Bld_eLibraryItems"
Type="Bool">true</Property>
        <Property                                Name="Bld_excluymorphicVIs"
Type="Bool">true</Property>
        <Property                                Name="Bld_lDestDir"
Type="Path">../builds/NI_AB_PROJECTNAME/Diplom_Interpolate</Property>
        <Property                                Name="Bld_locatDirType"
Type="Str">relativeToCommon</Property>
        <Property                                Name="Bld_modifyaryFile"
Type="Bool">true</Property>
        <Property  Name="Bld_prevacheID"  Type="Str">{4B2D39FC-
15EB-4D5E-B701-0622F0A718AF}</Property>
        <Property                                Name="Bld_vern.build"
Type="Int">1</Property>
        <Property                                Name="Bld_vers.major"
Type="Int">1</Property>
        <Property  Name="Destinat          [0].destName"
Type="Str">Application.exe</Property>
        <Property                                Name="Destion[0]path"
Type="Path">../builds/NI_AB_PROJECTNAME/Diplom_Interpolate/Application.exe</P
roperty>
        <Property                                Name="Destination[0].presearchy"
Type="Bool">true</Property>
        <Property                                Name="Destin[0].type"
Type="Str">App</Property>
        <Property  Name="Destination[1tName"  Type="Str">Support
Directory</Property>

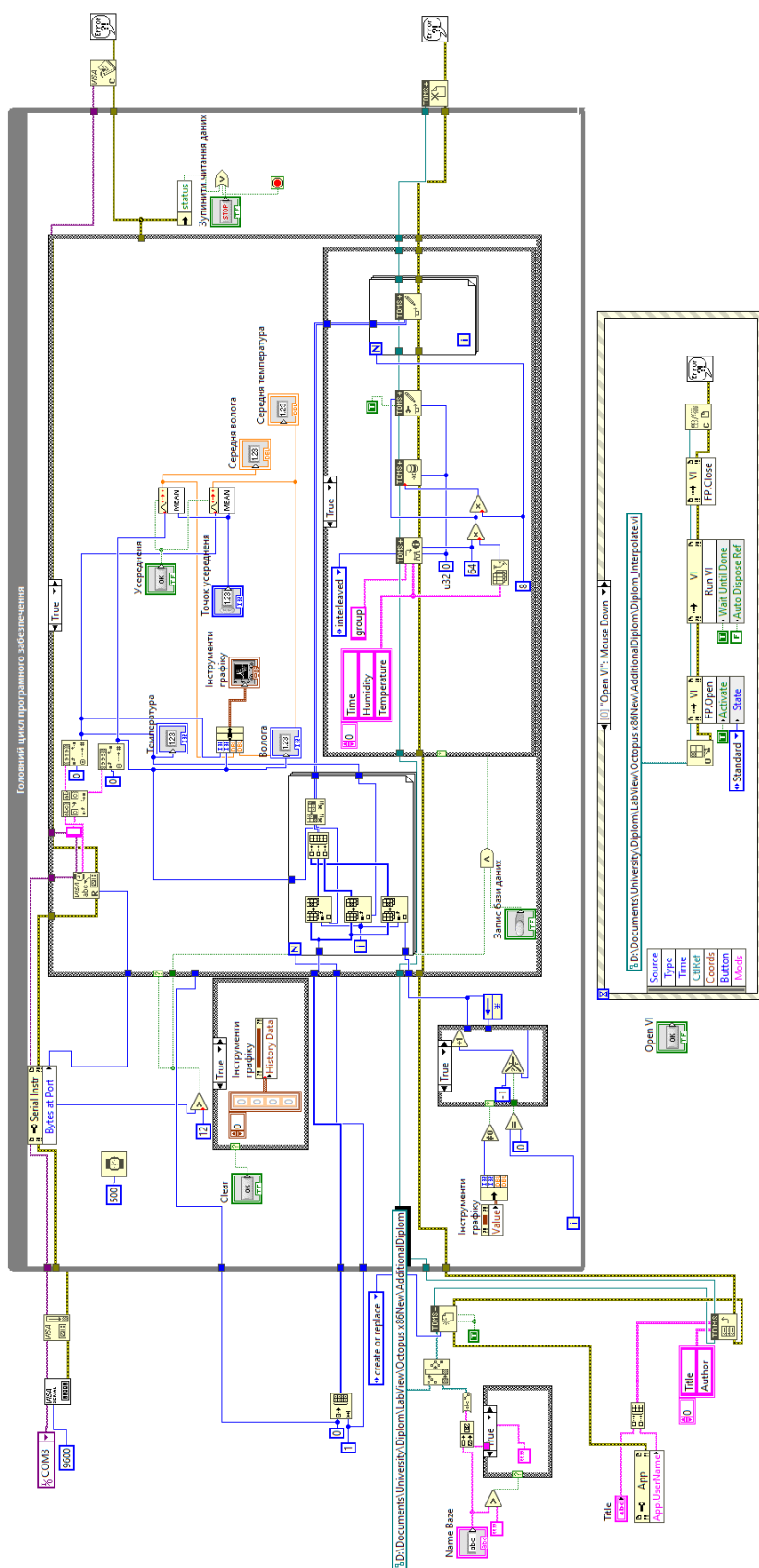
```

```

        <Property                                Name="Desttion[1].path"
Type="Path">../builds/NI_AB_PROJECTNAME/Diplom_Interpolate/data</Property>
        <Property Name="DesttionCount" Type="Int">2</Property>
        <Property                                Name="Source[0].itemID"
Type="Str">{3325941F-A86D -9073C1966}</Property>
        <Property                                Name="Source[0].type"
Type="Str">Container</Property>
        <Property                                Name="Source[1].destiIndex"
Type="Int">0</Property>
        <Property Name="Source[1].itemID" Type="Ref">/My
Computer/Diplom_Inate.vi</Property>
        <Property                                Name="Source[1].sourceInn"
Type="Str">TopLevel</Property>
        <Property Name="Source[1].type" Type=" " </Property>
        <Property Name="SourceCount" Type="Int">2</Property>
        <Property                                Name="TgtF_fileDeption"
Type="Str">Diplom_Interpolate</Property>
        <Property                                Name="TgtF_iname"
Type="Str">Diplom_Interpolate</Property>
        <Property Name="TgtF_legalght" Type="Str">Copyright ©
2021 </Property>
        <Property                                Name="TgtF_proName"
Type="Str">Diplom_Interpolate</Property>
        <Property Name="TgtF_targetUID" Type="Str">{D1CA6F49-
C9C4-4C5E-90F3-37DED2FBC563}</Property>
        <Property                                Name="TgtF_targeeName"
Type="Str">Application.exe</Property>
        <Property                                Name="TgtF_versionInddent"
Type="Bool">true</Property>
    </Item>
</Item>
</Item>
</Project>
    self.hasvem2 = int(obj_elem.get("HasrItem"), 0)
    vartype2 = obj_elem.get("Vartype")
    if vartype2 isnot None:
        self.vartype2 = int(vartype2, 0)

```

Б.3 Блок діаграма підпрограми управління



ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра Дослідження та розробка програмного забезпечення для системи стендових випробувань авіаційних двигунів

Виконав
ст. гр. КНТ – 210м

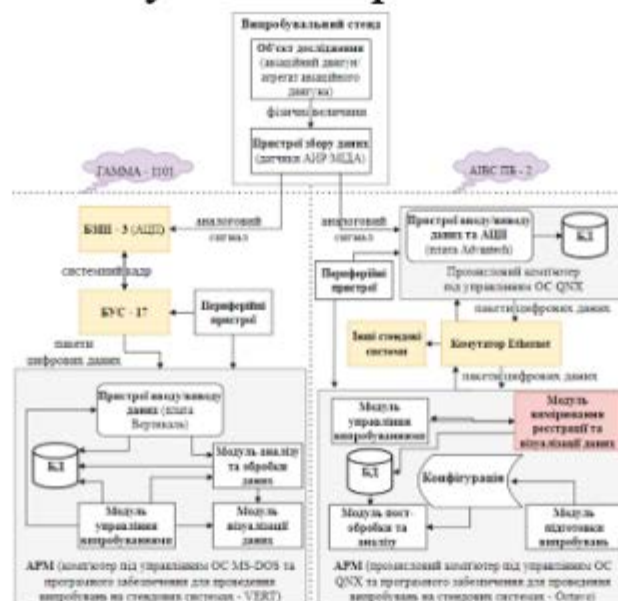
А.С. Архипов

Керівник
к. т. н., доцент

А.В. Пархоменко

Рисунок В.1 – Слайд №1

Актуальність роботи



Архітектура систем стендових випробувань ГАММА та АІВС

Рисунок В.2 – Слайд №2

Актуальність роботи

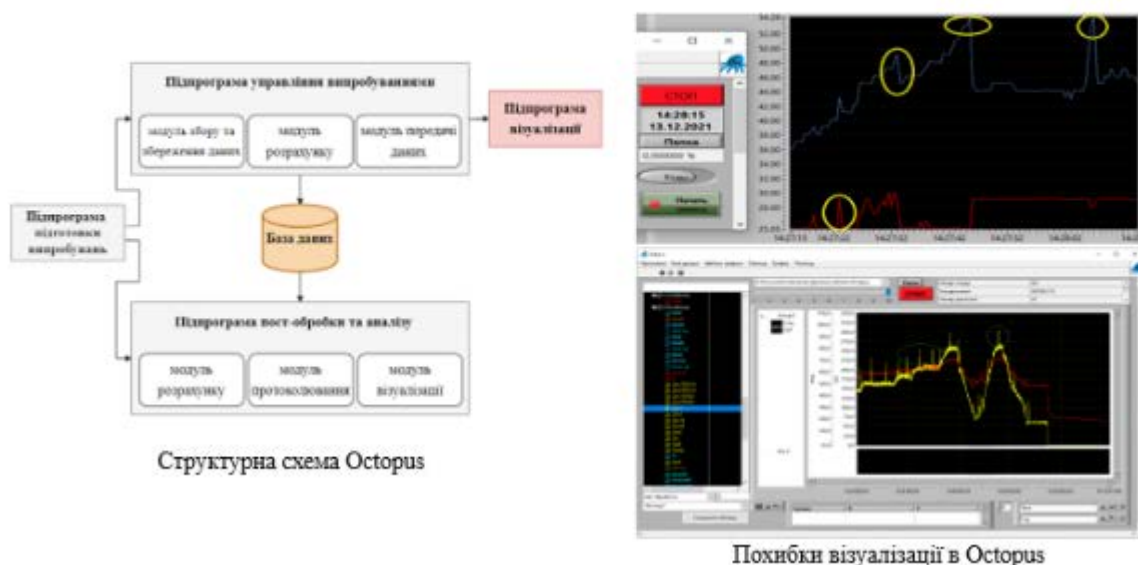


Рисунок В.3 – Слайд №3

Мета та задачі роботи

Мета роботи – дослідження та програмна реалізація методів та засобів візуалізації даних для підвищення якості представлення вимірюваних параметрів на системах стендових випробувань авіаційних двигунів.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

- зібрати набори даних для дослідження, дослідити бази даних випробувань авіаційних двигунів;
- розробити інструмент для тестування методів візуалізації даних, протестувати методи, обрати відповідні для вирішення поставленої задачі;
- визначити вимоги до ПЗ, розробити його архітектуру та вибрати середовище програмування;
- розробити тестове програмне забезпечення для проведення стендових випробувань, реалізувавши в ньому досліджені методи.

Рисунок В.4 – Слайд №4

Аналіз даних випробувань авіаційних двигунів

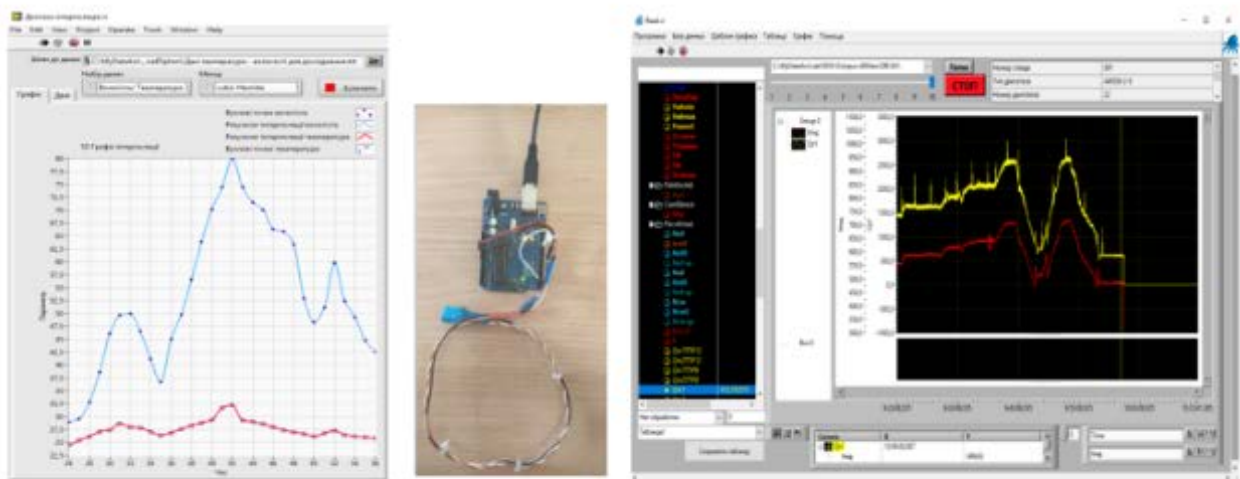


Методи аналітики Великих даних

5

Рисунок В.5 – Слайд №5

Набори даних для дослідження



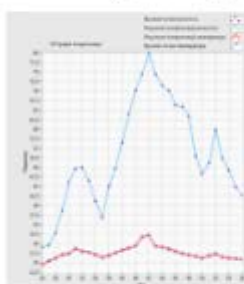
6

Рисунок В.6 – Слайд №6

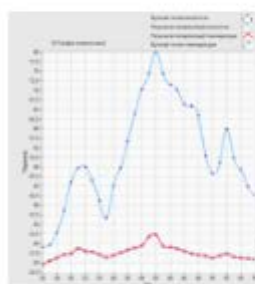
Аналіз методів апроксимації

Набір експериментальних даних

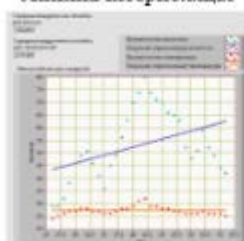
Набір стендових даних



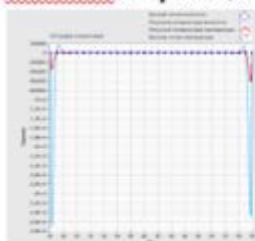
Лінійна інтерполяція



Сплайнова інтерполяція



Метод найменших квадратів



Інтерполяція Лагранжу



Порівняння лінійної та сплайнової інтерполяції на ділянці 1



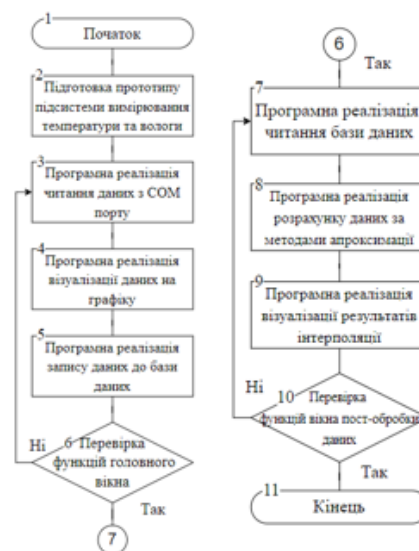
Лінійна інтерполяція

Рисунок В.7 – Слайд №7

Алгоритм розробки та вимоги до ПЗ

Розроблюване програмне забезпечення має виконувати наступні завдання:

- можливість зчитування та збереження даних випробування;
- можливість візуалізації вимірюваних параметрів;
- можливість проведення обробки записаної бази даних за методами апроксимації;
- можливість усунення похибок датчиків на графіку під час проведення випробувань за рахунок усереднення параметрів.



Алгоритм розробки тестового програмного забезпечення 8

Рисунок В.8 – Слайд №8

Розробка ПЗ в середовищі LabVIEW

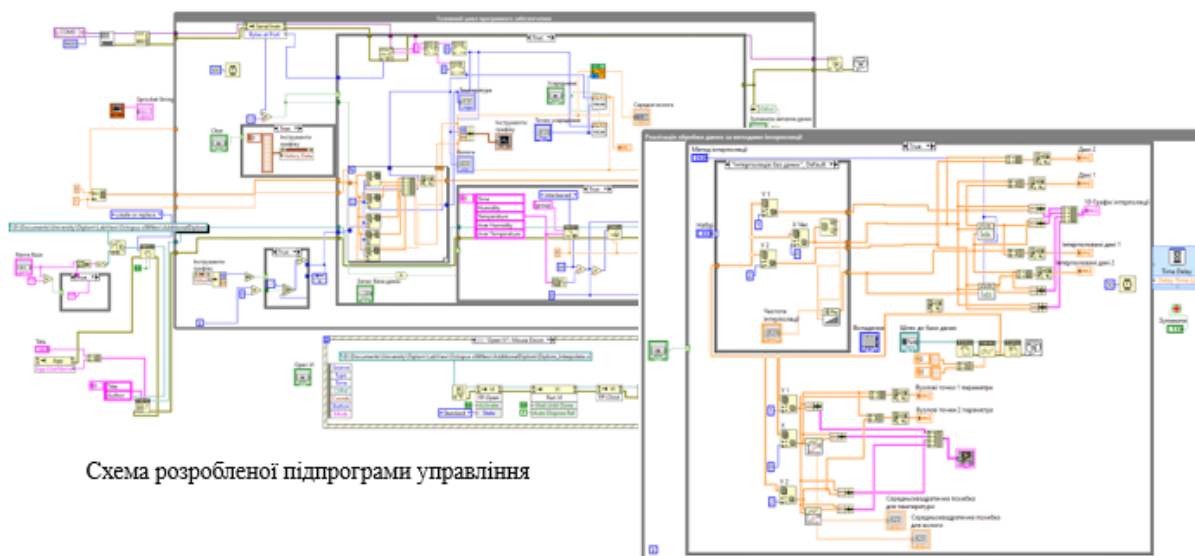
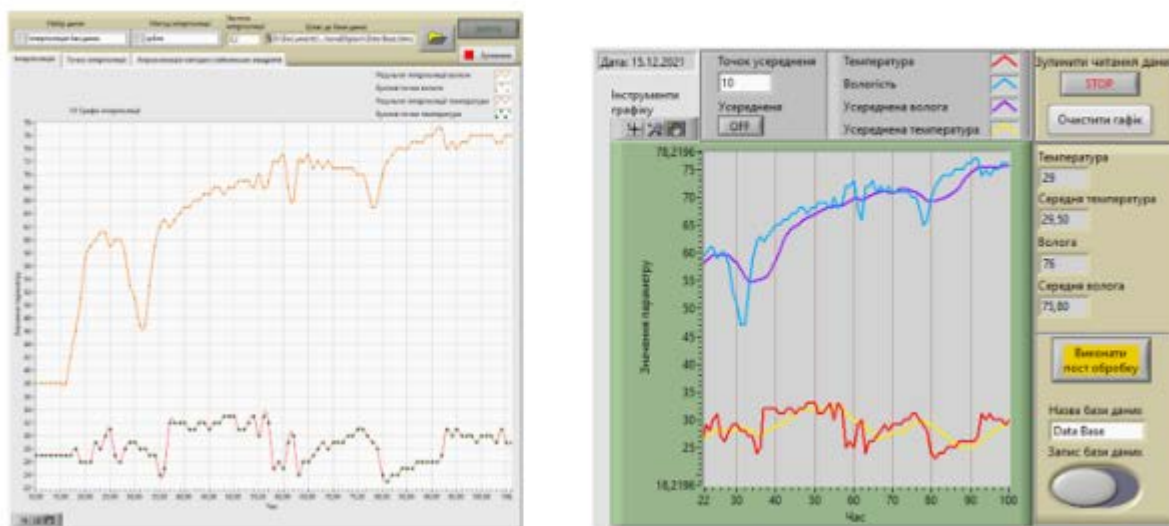


Схема розробленої підпрограми управління

Схема розробленої підпрограми пост обробки 11

Рисунок В.11 – Слайд №11

Інтерфейс розробленого ПЗ



Лифцева панель підсистеми пост-обработки та аналізу

Лифцева панель підсистеми управління 12

Рисунок В.12 – Слайд №12

База даних розробленого ПЗ

Asset Name	Title	Author	Date/Time	Groups	Description
Data Base	LABVIEW Example (time domain)	Author			
Group	Channels				
group					
Channel	Subtype	Unit	Length	Minimum	Maximum
Time	DT_SOURCE		97		
Humidity	DT_SOURCE		97		
Temperature	DT_SOURCE		97		
Asset Humidity	DT_SOURCE		97		
Asset Temperature	DT_SOURCE		97		

Time	Humidity	Temperature	Asset Humidity	Asset Temperature
1	38	27	38	27
2	38	27	38	27
3	38	27	38	27
4	38	27	38	27
5	38	27	38	27
6	38	27	38	27
7	38	27	38	27
8	38	27	38	27
9	38	27	38	27
10	38	27	38	27
11	38	27	38	27
12	38	27	38	27
13	38	27	38	27
14	38	27	38	27
15	38	27	38	27
16	38	27	38	27
17	38	27	38	27
18	38	27	38	27
19	38	27	38	27
20	38	27	38	27
21	38	27	38	27
22	38	27	38	27
23	38	27	38	27
24	38	27	38	27
25	38	27	38	27
26	38	27	38	27
27	38	27	38	27
28	38	27	38	27
29	38	27	38	27
30	38	27	38	27
31	38	27	38	27
32	38	27	38	27
33	38	27	38	27
34	38	27	38	27
35	38	27	38	27
36	38	27	38	27
37	38	27	38	27
38	38	27	38	27
39	38	27	38	27
40	38	27	38	27
41	38	27	38	27
42	38	27	38	27

Лист з інформацією про базу

Лист з даними

13

Рисунок В.13 – Слайд №13

Висновок

В результаті виконаної роботи було проведено аналіз існуючих на підприємстві стендових систем збору даних, досліджено дані випробувань авіаційних двигунів та методи обробки цих даних, розроблено тестове ПЗ для візуалізації даних випробувань.

Наукова новизна роботи полягає у тому, що отримали подальший розвиток методи обробки та візуалізації параметрів авіаційних двигунів за рахунок удосконалення функції інтерполяції даних для більш наочного представлення інформації.

Практична цінність роботи полягає в тому, що впровадження розробленого ПЗ дозволить модернізувати підсистему візуалізації Ostorus для правильної інтерпретації даних випробувань фахівцем.

В майбутньому планується розширення функціональних можливостей тестового ПЗ за рахунок додавання списку каналів отримання даних та списку підтримуваних плат, для подальшого встановлення цього ПЗ на всі стенди підприємства.

Результати роботи було представлено на Щорічній науково-практичній конференції НУЗП Тиждень науки – 2021, 19-23 квітня 2021 р.

14

Рисунок В.14 – Слайд №14