Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Навчально-науковий інститут електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра кіберфізичних та інформаційно-вимірювальних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

здобувача вищої освіти Гончарова Владислава Костянтиновича
(П.І.Б.)

академічної групи 151м-21-1
(шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва спеціальності)

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Синтез та дослідження кіберфізичної системи класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи
(назва за наказом ректора)

Консультанти	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Керівник кваліфікаційної роботи	проф, Бубліков А.В.			
розділів:				
Синтез системи керування	проф., Бубліков А.В.			
Експериментальний розділ	проф., Бубліков А.В.			
Економічна частина	ст. викл., Яремчук І.О.			
Охорона праці	проф, Чеберячко Ю.І.			

Нормоконтролер	асист., Славінський Д.В.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувачем кафедри
кіберфізичних та інформаційно-вимірювальних систем
(повна назва)
_____ Бубліков А.В.
(підпис) (прізвище, ініціали)

«_____»_____2022 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

здобувача вищої освіти Гончарова В.К. **академічної групи** 151м-21-1
(прізвище та ініціали) (шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Синтез та дослідження кіберфізичної системи класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № _____.

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	1.10.22-06.10.22
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	07.10.22-14.10.22
Синтез системи	Розробка алгоритму керування та програмного забезпечення з людино-машинним інтерфейсом	15.10.22-5.11.22
Економічна частина	Економічне обґрунтування доцільність витрат на створення системи керування.	5.11.22-12.11.22
Охорона праці	Розробка організаційно-технічних заходів, щодо реалізації правил безпеки при експлуатації системи.	15.11.22-5.12.22

Завдання видано _____ Бубліков А. В.
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до атестаційної комісії _____

Прийнято до виконання _____ Гончаров В.К.
(підпис здобувача) (прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота містить 70 с., 36 рис., 12 табл., 1 дод.

Об'єкт дослідження – є ваговимірювальна платформа для зважування залізничних вагонів.

Предмет дослідження – методи та алгоритми класифікації залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи.

Мета дослідження – проведення класифікації залізничних типів вагонів за сигналами з ваговимірювальній платформи.

Проведений аналіз конструкції і процесу зважування на ваговимірювальній платформі залізничного вагона. За результатами сформована база знань конструктивних параметрів і на її основі розроблена імітаційна модель проїзду вагона по платформі для проведення досліджень з їх класифікації.

Результатами досліджень є отриманий алгоритм класифікації типів вагонів побудований на основі нейронної мережі Перцептрон, який враховує похибку визначення параметрів.

Розглянуто комплекс питань щодо економіки та охороні праці.

ABSTRACT

Qualification of the robot to revenge 70 p., 36 pics., 12 tab., 1 add.

The object of the follow-up is a vacant platform for the loading of rail cars.

The subject of the study is methods and algorithms for classifying saloon cars based on the analysis of signals from the railroad platform.

Meta follow-up - improving the accuracy of classification of rail car types for signals from the railroad platform.

Carrying out an analysis of the design and the process of launching on the wagon platform of a rail car. Based on the results, a base of knowledge of design parameters was formed and, on the basis of basis, a simulation model of the passage of a wagon along the platform was developed to carry out further classification.

As a result, we have subtracted the algorithm for classifying the types of carriages of impulses based on the neural network Perceptron, which is a kind of safe choice of parameters.

The complex of nutrition for economics and protection practices was reviewed.

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

FRAM – Function Resonantly Analysis Model

ЗМІСТ

Вступ

Використання інформаційно-вимірювальних систем дає можливість зважувати залізничні вагони і автопоїзди. Такі системи можуть використовуватись в залізничному транспорті під час завантаження, а також вбудовуватись в залізничну структуру.

Зацікавленість у подібних системах організацій показала, що зважування ваги по одній осі є популярними у різних напрямках: сільгоспвиробництво, металообробка, будівництво.

Організація встановив електронну залізничну ваговимірювальну платформу отримує контроль над перебігом вантажу, що призводить до зменшення виробничих витрат, а використання програмного забезпечення дозволяє ще більше контролювати усі етапи зважування ваги.

Актуальність теми полягає в тому, що автоматизація цих процесів дозволяє значно знизити час, затрачений на обробку, і зменшити кількість помилок при обробці даних.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Галузь промисловості

Для залізничного транспорту, що забезпечує вантажопотоки між виробниками та покупцями на різних етапах виробничо-технологічних та фінансових відносин важливо мати оперативні засоби контролю маси одиниць рухомого складу з вимогами, що відповідають специфіці вантажів, що перевозяться, і відповідно до існуючих стандартів. З багатьох причин виникає необхідність контролю маси рухомих об'єктів не тільки на початковому та кінцевому пунктах прямування, а й на проміжних фазах перевезень [1].

Компанія «Укрзалізниця» забезпечує 82% вантажних і майже 50% пасажирських перевезень, які здійснюються усіма видами транспорту. За обсягами вантажних перевезень українська залізниця займає четверте місце на Євразійському континенті, поступаючись, зокрема, залізницям Китаю та Індії[1]. Парк рухомого складу «Укрзалізниці» станом на 2017 рік складав: вантажні вагони – 65422 од., локомотиви – 2250 од., в тому числі: електровози 1469 од. та тепловози – 429 од.; пасажирські вагони – 5210 од. [2].

В результаті встановлення електронних залізничних ваг отримується контроль над оборотом вантажу, що призводить до зменшення виробничих втрат, а використання програмного забезпечення дозволяє ще ефективніше контролювати всі етапи вимірювання ваги і поведення класифікації вагонів.

1.2 Технологічній процес

Зважування вагонів є частиною логістичної системи управління товарних потоків залізничним шляхом від виробника до споживача. Виробник прикладає до вантажу документацію (тип, вага, країна, адреса та ін.) і проходячи через кордон або безпосередньо на підприємстві, вагони з вантажем перевіряються на зазначену виробником інформацію. Но однією з головних властивостей процесу навантаження є перевірка ступеню загрузки

вагону. Перевірка проводиться на ваговій платформі (рис. 1.1) завдяки встановленим тензометричним датчикам.

Зважування вагонів платформою виконується різними варіантами: осева і візкова (рис. 1.1), вагонна (рис. 1.2) і вимірюється за статикою (вагон стоїть на місці) або динамікою (вагон проїжджає по платформі).

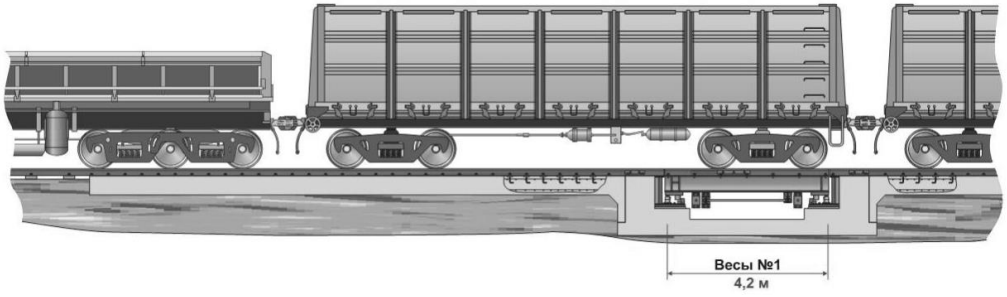

Рисунок 1.1 – Зважування ваги вагону візками в статиці або в динаміці

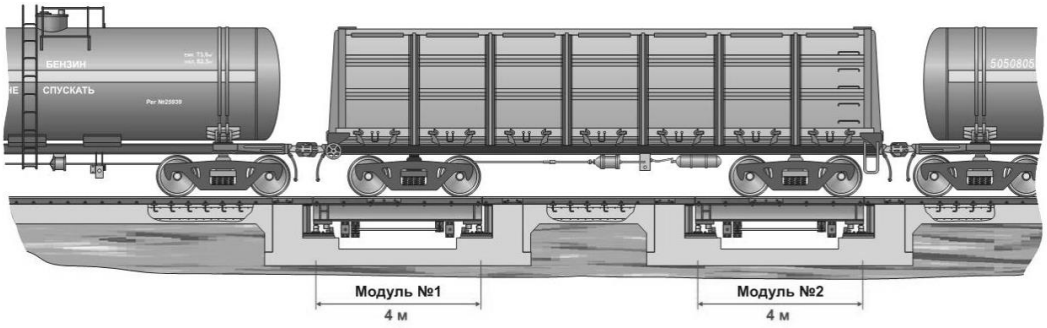

Рисунок 1.2 – Зважування ваги вагону цілком в статиці або динаміці

Вагова платформа розташовується на залізничному полотні, по якій проїжджає потяг. Встановлені датчики під тиском ваги деформуються, що призводить до зміни опору. Також змінюється і струм, який є інформаційним параметром, що характеризує зміну ваги на платформі. Зміна струму відображає заїзд чи з'їзд осі, візка вагону. Тому цю особистість технологічного процесу і буде використано в кваліфікаційній роботі для визначення типу вагону.

1.3 Об'єкт керування

Об'єктом дослідження у цій кваліфікаційній роботі є ваговимірювальна платформа для зважування залізничних вагонів.

1.3.1 Структура об’єкту керування

Функціональні елементи що формують структуру системи зважування вагонів на ваговимірювальній платформі зображена на рис. 1.3.

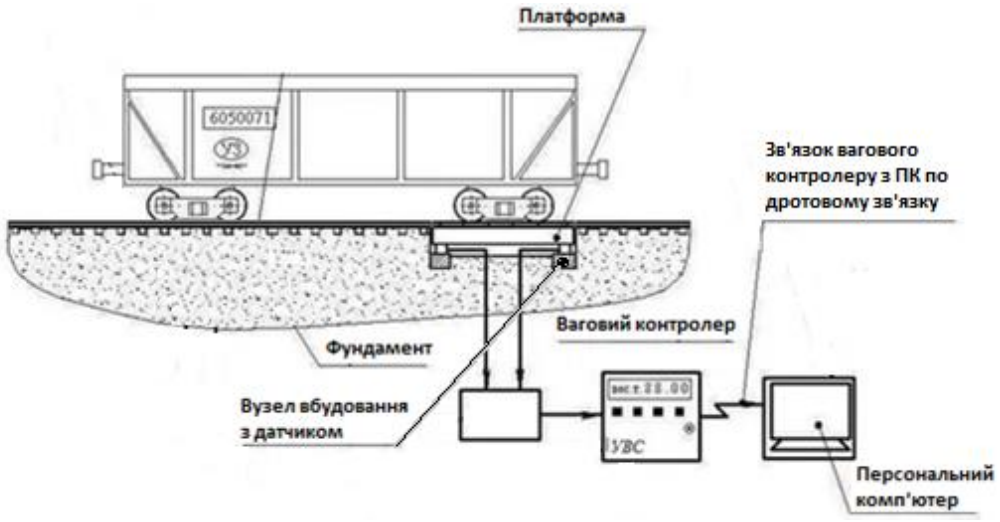

Рисунок 1.3 – Функціональна схема системи вимірювання ваги та ідентифікації типу вагону

Складова об’єкта керування складається з одного вагону що проїжджає по платформі, ваговимірювальної платформи, вагового контролеру та терміналом оператора (комп’ютера).

Оскільки по залізничному полотну проїжджають різні типи вагонів, необхідно сформувати базу знань фізичних параметрів вагонів для проведення експериментів і розробки моделі об’єкту керування. Тому надалі йдуть графічні зображення та фізичні параметри вагонів.

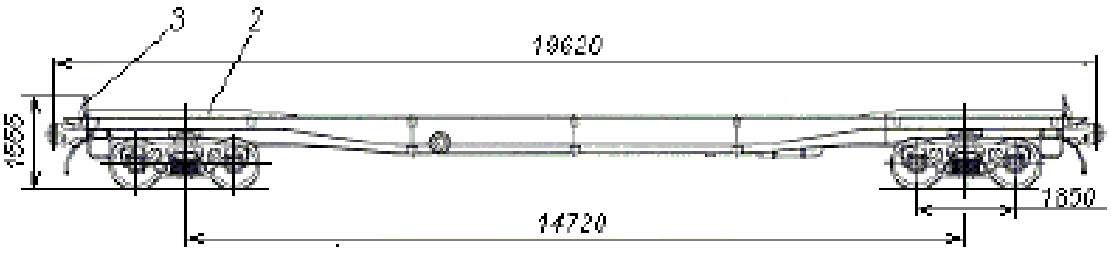

Рисунок 1.4 – Вагон-платформа 13-9004М

Вагон-платформа модифікації 13-9004М – вантажний вагон відкритого типу (рис. 1.4), призначений для перевезення довгомірних, штучних вантажів,

контейнерів та обладнання, що не потребують захисту від атмосферних впливів. Його конструктивні параметри наведені у табл. 1.1.

Таблиця 1.1 – Технічні параметри вагону 13-9004М

Вантажопідйомність, т	40
База вагона, мм	14720
Довжина по осях зчеплення автозчіпок, мм	19620
Довжина по кінцевих балках рами, мм	18400
Ширина, мм	2870
Кількість вісь, шт	4
Модель двовісного візка	18-100

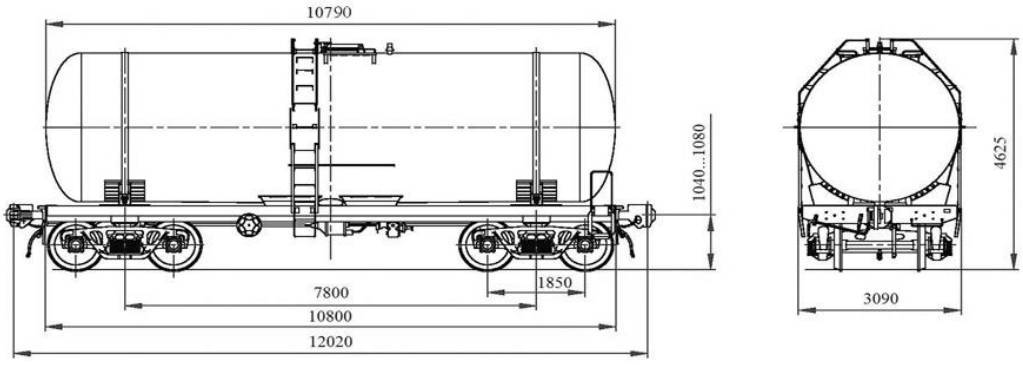

Рисунок 1.5 – Вагон-цистерна 15-776

Вагон-цистерна (рис. 1.5) призначена для перевезення світлих нафтопродуктів по всій мережі залізниць колії 1520 мм. Технічні параметри вагону наведені у таблиці 1.2.

Таблиця 1.2 - Технічні параметри вагону-цистерна 15-776

Вантажопідйомність, т	40
База вагона, мм	14720
Довжина по осях зчеплення автозчіпок, мм	19620

Продовження таблиці 1.2 – Технічні параметри вагону-цистерна 15-776

Довжина по кінцевих балках рами, мм	18400
Ширина, мм	2870
Кількість вісь, шт	4
Модель двовісного візка	18-100

Вагон хопер 19-7016 (рис. 1.6) призначений для безтарного перевезення зерна та аналогічних харчових продуктів, що вимагають захисту від атмосферних опадів, по всій мережі залізниць колії 1520 мм, із завантаженням через верхні люки та гравітаційним розвантаженням у міжрейковий простір через нижні розвантажувальні люки у спеціальні приймальні пристрої.

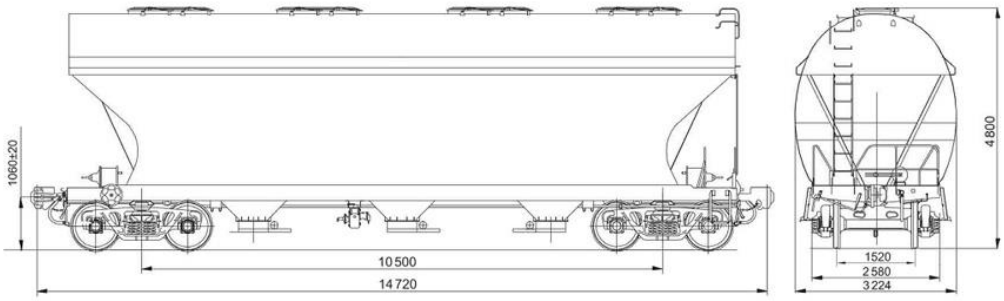

Рисунок 1.6 - Вагон хопер 19-7016

Технічні параметри вагону-хоперу наведено у таблиці 1.3.

Таблиця 1.3 - Технічні параметри вагону-хоперу 19-7016

Вантажопідйомність, т	70,2, т
База вагона, мм	10500
Довжина вагона по осям автозчепів, мм	14720
Ширина, мм	1520
Візок	18-7055

Напіввагон 12-7039 (рис. 1.7) призначений для перевезення штучних, сипких, у тому числі дрібнокускових, та пакетованих вантажів, який не вимагає захисту від атмосферних опадів по всій мережі залізниць із шириною колії 1520 мм з можливістю розвантаження через розвантажувальні люки, розміщені у підлозі вагона, або на вагоноперекидачу.

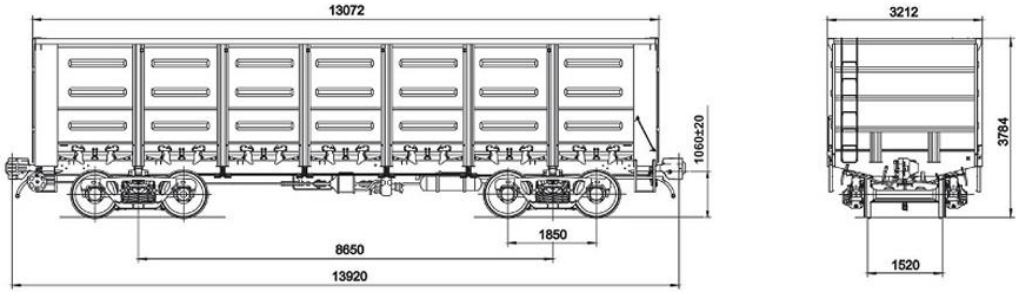

Рисунок 1.7 – Напіввагон 12-7039

Технічні параметри вагону наведені у таблиці 1.4.

Таблиця 1.4 - Технічні параметри напіввагону 12-7039

Вантажопідйомність, т	75,5, т
База вагона, мм	8650
Довжина вагона по осям автозчепів, мм	13920
Ширина, мм	1520
Візок	18-7033

Вагонні ваги 150ВВСД (рис. 1.8) на 150 т. призначені для статичного і динамічного зважування вагонів/потягів. Визначення ваги в статичному режимі відбувається вагонного і візкового. Зважування в русі відбувається без розчеплення 2-х, 4-х, 6-ти і 8-ми вісних вагонів і потяга в цілому, виключаючи вагу локомотива.

Рисунок 1.8 – Ваговимірювальна платформа модифікації 150ВВСД

Технічні характеристики ваговимірювальної платформи наведено у табл. 1.5.

Таблиця 1.5- Технічні характеристики ваговимірювальної платформи

Модифікація ваг	150ВВСД
Максимальна вантажопідйомність, т	150
Інтервали зважування	Одноінтервальні
Найбільша межа зважування, Max, т	Max = 150
Найменша межа зважування, Min, т	1
Ціна поділення = ціна повірочної поділки, кг	d = e = 50 кг.
Клас точності ваг згідно з ДСТУ EN 45501: 2017	Середній
Межі допустимої похибки при контролі в експлуатації в статиці, кг	від 1 т. до 25 т.: ± 50 понад 25 т. до 100 т.: ± 100 понад 100 т. до 150 т.: ± 150
Межі допустимої похибки при контролі в експлуатації в динаміці,%	0,5; 1; 2; 5
Швидкість руху вагона при зважуванні, км/год	до 10
Довжина, м	4,5

Встановлені датчики Zemic DHM9Bd10-C3-30t (рис 1.9) – цифрові тензометричні датчики на 30 тон. Мають високу зносостійкість, виконанні зі сталі з нікелевим покриттям і повно пило- та вологозахист (IP68). Інформація від датчика до вагового приладу передається у вигляді двійкового коду, що дозволяє використовувати ваговий прилад на відстані до 1 км. від ваг, не втрачаючи точності. Живіться від 6-15 В. Застосовується для автомобілів, вагонних ваг, систем зважування і інших вагових обладнань.

Рисунок 1.9 – Тензометричний датчик Zemic DHM9Bd10-C3-30t

Тензометричний датчик з’єднується з ваговим індикатором (рис. 1.10), в якому автоматично зберігаються данні перетворені з двійкового коду в десяткове число що зображується на цифровому екрані. В контролері стоїть акумулятор на 4А/год. з контролером розряду, має інтерфейс RS-232/485, кабель-інтерфейс USB і можливість під’єднання до мережі Ethernet. Пило-вологозахищена мембранна клавіатура й корпус за класом захисту IP65. Живиться від мережі змінного струму 187…240В.

Рисунок 1.10 – Ваговий термінал ВП-01МЦ

Таблиця 1.6 – Технічні параметри вагового процесору ВП-01МЦ

Напруга живлення терміналу	Від мережі	Від акумулятора
	~187…240В (±10%), 50 Гц	6В/4Агод
Робочий темп. діапазон	від -10℃ до +40℃	
Інтерфейс зв’язку з датчиками	RS422	
Ціна повірочних поділок	Від 5 до 100 кг	
Кількість повірочних поділок	До 10 000	
Дискретність	1/2/5/10/20/50/100 за вибором	
Інтерфейс зв’язку з ПК	RS232/485	
Ступінь захисту	Фактична IP54, можна розробити IP65	

1.3.2 Принцип функціонування об’єкту керування

Вагон проїжджаючи по платформі, вимушує вагою змінити свою форму тензометричні датчики. Змінюючи свою форму, датчики змінюють свої властивості – свій опір. Аналогово-цифровий перетворювач надсилає перетворений до двійкового коду сигнал, щодо зміни опору датчика, на ваговий термінал, в якому отриманий код перетворюється в значення в десятковій формі і зображується на терміналі. Отримане значення зберігається во внутрішнє сховище терміналу та надалі по дротовому (або бездротовому) зв’язку надсилається на термінал оператора.

1.4 Структура системи керування

Система класифікації типів вагонів на основі отриманих сигналів не має своєї структури, оскільки об’єктом керування є процес класифікації на основі зібраних даних у ваговому терміналі і терміналі оператора. Для розуміння всього процесу класифікації, наведена структура системи зважування ваги вагону проїжджаючого по платформі (рис. 1.11).

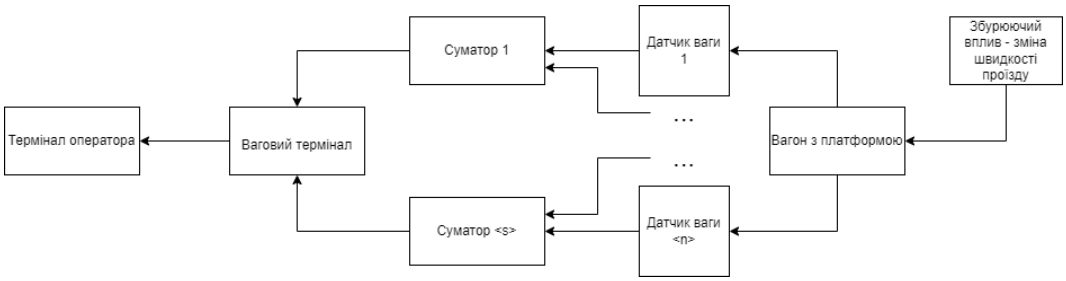

Рисунок 1.11 – Структура системи керування

До системи входить вагон що проїжджає по ваговимірювальній платформі, встановлені датчики ваги, сигнали яких з кожної сторони платформи надсилаються до суматора. Розраховане значення відправляється до вагового терміналу й далі до терміналу оператора. Збурюючи впливи які впливають на об’єкт керування, що розглядається у кваліфікаційній роботі, є зміна швидкості проїзду вагона по платформі.

1.5 Формулювання завдань дослідження

Проведення досліджень з класифікації залізничних типів вагонів проводиться за наступними поставленими завданнями:

– для проведення досліджень з класифікації типів вагонів створити імітаційну модель проїзду вагону по ваговимірювальній платформі;

– виконати дослідження за класифікацією типів вагонів з урахуванням їх конструктивних параметрів.

1.6 Висновки по розділу

Об'єктом дослідження є ваговимірювальна платформа для зважування залізничних вагонів. Предметом дослідження – процес класифікації залізничних типів вагонів за сигналами що надходять з ваговимірювальної платформи.

Розглянуті конструктивні параметри вагонів: кількість осей візка, відстань між осями одного візка, відстань між внутрішніми осями сусідніх візків. Розроблену систему верхнього рівня з ваговимірювальної платформи, вагового терміналу і персонального комп’ютера як місця оператора.

Розглянуто комплекс питань щодо проведення досліджень.

2 ТЕОРІТИЧНИЙ РОЗДІЛ. СТВОРЕННЯ МОДЕЛІ ІМІТАЦІЇ СИГНАЛУ З ВАГОВИМІРЮВАЛЬНОЇ ПЛАТФОРМИ

2.1 Модель об’єкта керування

Створення моделі імітації сигналу проїзду вагонного состава по ваговимірювальній платформі виконаємо на основі визначених інформаційних ознак (відстань між осями одного візка, відстань між внутрішніми осями візків та відстань від останньої осі візка до зчіпки вагона з двох сторін) вагонів, розглянутих у першому розділі кваліфікаційної роботи. Як результат роботи моделі очікується вектор значень вимірювальної ваги для проведення досліджень з класифікації типів залізничних вагонів.

Імітаційну модель буде розроблено в програмному забезпеченні Matlab з використанням Script’ів. У файлі generator_signal.m напочатку проведемо ініціалізацію змінних, що відповідають інформаційним ознакам вагонів:

```
% Напіввагон 12-7023 (двоосний), мм
sz_t1k1_1 = 1710; % відстань від зчіпки вагона до першої осі вагона
t1k1_t1k2_1 = 1850; % відстань від першої осі вагона до другої
t1k2_t2k1_1 = 6800; % відстань від другої осі вагона до третьої
t2k1_t2k2_1 = 1850; % відстань від третьої осі вагона до четвертої
t2k2_sz_1 = 1710; % відстань від четвертої осі вагона до зчіпки

%Вагон-хоппер 19-7016 (двоосний), мм
sz_t1k1_2 = 1185;
t1k1_t1k2_2 = 1850;
t1k2_t2k1_2 = 8650;
t2k1_t2k2_2 = 1850;
t2k2_sz_2 = 1185;

%Вагон-хоппер дозатор 19-789 (двоосний), мм
sz_t1k1_3 = 1185;
t1k1_t1k2_3 = 1850;
t1k2_t2k1_3 = 5350;
t2k1_t2k2_3 = 1850;
t2k2_sz_3 = 1185;
```

Розташування вагонів буде випадковим для підвищення рівня невизначеності перед проведенням досліджень по їх класифікації за сигналами. Тому створимо вектор випадкового розташування вагонів у залізничному составі:

```
N_w = 100;
type_w_1 = rand(N_w,1,'single');
for i = 1:N_w
   if(type_w_1(i) < 0.33)
      type_w(i) = 1;
   else
      if(type_w_1(i) > 0.66)
         type_w(i) = 2;
      else
         type_w(i) = 3;
      end
   end
end
```

У наведеному коді вище у тимчасову змінну type_w_1 генерується функцією rand(*) випадкові значення що лежать в діапазоні від нуля до одиниці. Певними умовами вибору до вектора type_w заноситься умовний ідентифікатор типу вагона від одиниці до трійки (одиниця відповідає напіввагону 12-7023 тощо).

Розробка імітаційної моделі повинна відповідати реальним фізичним процесам що проходять при проїзду состава по платформі. Тому для більш реалістичного створення імітаційної моделі розглядається такий фізичний вплив на зміну сигналу як центр ваги на візку та половину вагона. Но оскільки це лише імітаційна модель і створити ідеально реалістичний процес не є можливим, розглядати випадкове розподілення ваги на другій половині вагона як окрему складову не є доцільним. Тому другу частину вагової складової вагону будемо прирівнювати до першої. Відповідно, у наведеному коді нижче, центру ваги візка відповідає змінна Vaga_1k, а половині вагона – Vaga_w_z:

```
%Коефіцієнт розподілу ваги між колесами візка (через нерівномірність
розподілу може коливатися між 40 та 60% ваги першої половини вагону)
Vaga_1k = (rand(N_w,1,'single')*0.2)+0.4;
```

```
 %Вага половини вагонів, т. Межа зміни ваги від 12.5 т. до 36.5 т.
Vaga_w_z = ((rand(N_w,1,'single')*50)+23)*0.5;
```

Створені також додаткові змінні, які є технічною реалізацією моделі та не впливають на фізичний процес. Де змінна Conter_w відповідає конкретному типу вагона, Counter_cm – лічильник відстані у рамках одного вагону, Vaga_w_on – вектор моментів заїздів вагона на платформу.

```
%Лічильник вагонів
Conter_w = 1;

%Лічильник міліметрів у рамках одного вагону
Counter_cm = 0;

% Формування вектору заїздів вагонів на платформу
Vaga_w_on(1) = 0;
```

Формування сигналу імітаційної моделі складемо з двох частин: заїзд та з’їзд вагона з платформи. Цей процес (подія) також розкладається на підпроцеси (під події) з заїздом першої та другої осі першого візка, першої та другої осі другого візка. Всі події мають відносний зв’язок між собою, тому заїзд та з’їзд вагона буде розраховуватись згідно з їх фізичними зв’язками визначених у змінних sz_t1k1, t1k1_t1k2, t1k2_t2k1, t2k1_t2k2, t2k2_sz.

Наведений далі код програми, розраховує моменти заїзду осей вагонів впродовж певної кількості міліметрів зазначеної в змінній “i”, значення якої було взято з приблизно необхідної кількості міліметрів для проходження всього составу по платформі.

```
for i = 1:2000000
   %Завантаження параметрів поточного вагону
   if( type_w(Conter_w) == 1)
      sz_t1k1 = sz_t1k1_1;
      t1k1_t1k2 =  t1k1_t1k2_1;
      t1k2_t2k1 = t1k2_t2k1_1;
      t2k1_t2k2 = t2k1_t2k2_1;
      t2k2_sz = t2k2_sz_1;
   end
    if( type_w(Conter_w) == 2)
      sz_t1k1 = sz_t1k1_2;
      t1k1_t1k2 =  t1k1_t1k2_2;
```

```
        t1k2_t2k1 = t1k2_t2k1_2;
        t2k1_t2k2 = t2k1_t2k2_2;
        t2k2_sz = t2k2_sz_2;
      end
      if( type_w(Conter_w) == 3)
        sz_t1k1 = sz_t1k1_3;
        t1k1_t1k2 =  t1k1_t1k2_3;
        t1k2_t2k1 = t1k2_t2k1_3;
        t2k1_t2k2 = t2k1_t2k2_3;
        t2k2_sz = t2k2_sz_3;
    end

    Counter_cm = Counter_cm +1;

    Vaga_w_on(i+1) = Vaga_w_on(i);

    % Заїзд на платформу 1 колеса 1 візка
    if ( Counter_cm == sz_t1k1)
        Vaga_w_on(i+1) = Vaga_w_on(i) +
(Vaga_1k(Conter_w)*Vaga_w_z(Conter_w));
    end
    % Заїзд на платформу 2 колеса 1 візка
    if ( Counter_cm ==  sz_t1k1+t1k1_t1k2)
        Vaga_w_on(i+1) = Vaga_w_on(i) + ((1-
Vaga_1k(Conter_w))*Vaga_w_z(Conter_w));
    end
     % Заїзд на платформу 1 колеса 2 візка
    if ( Counter_cm == sz_t1k1+t1k1_t1k2+t1k2_t2k1)
        Vaga_w_on(i+1) = Vaga_w_on(i) +
(Vaga_1k(Conter_w)*Vaga_w_z(Conter_w));
    end
    % Заїзд на платформу 2 колеса 2 візка
    if ( Counter_cm ==   sz_t1k1+t1k1_t1k2+t1k2_t2k1+t2k1_t2k2)
        Vaga_w_on(i+1) = Vaga_w_on(i) + ((1-
Vaga_1k(Conter_w))*Vaga_w_z(Conter_w));
    end
    % Закінчення вагону
    if ( Counter_cm == sz_t1k1+t1k1_t1k2+t1k2_t2k1+t2k1_t2k2+ t2k2_sz)
        Conter_w = Conter_w + 1;
        Counter_cm = 0;
    end
    %Закінчення потягу
    if ( Conter_w == N_w+1)
        break;
```

```
   end
end
```

На початку роботи циклу for проводиться вибірка параметрів певного типу вагона згідно зі згенерованим масивом випадкового їх розташування. Кожну ітерацію циклу, збільшується значення лічильника Counter_cm, значення якого перевіряється з розрахунками умовних моментів заїзду осі. Якщо умова виконується, в вектор Vaga_w_on додається значення розрахованої ваги. Наприкінці вагону, лічильник Counter_cm скидає своє значення в нуль як і значення лічильника вагонів після досягнення їм значення визначеної в змінній N_w.

Другою частиною створення імітаційної моделі є формування сигналу з’їзду осей вагону з платформи. Потрібно зауважити, що відстань до початку заїзду першої осі першого візка це є фактично відстань від осі до зчіпки вагону, а з’їзд цієї осі з платформи дорівнюватиме довжині самої платформи. Тому для початку формування сигналу з’їздів, потрібно значення лічильника Counter_cm встановити на -4500, мм. Також, відповідно вектору Vaga_w_on, створимо вектор з’їздів Vaga_w_off.

```
%Лічильник сантиметрів у рамках одного вагону
Counter_cm = -4500;

% Формування вектору з'їздів вагонів з платформи
Vaga_w_off(1) = 0;
```

Також скидаємо значення лічильника вагонів на початкове.

```
%Лічильник вагонів
Conter_w = 1;
```

Код формування другої частини сигналу наведений далі.

```
for i = 1:2000000
   %Завантаження параметрів поточного вагону
   if( type_w(Conter_w) == 1)
      sz_t1k1 = sz_t1k1_1;
      t1k1_t1k2 =  t1k1_t1k2_1;
      t1k2_t2k1 = t1k2_t2k1_1;
      t2k1_t2k2 = t2k1_t2k2_1;
```

```
        t2k2_sz = t2k2_sz_1;
    end
     if( type_w(Conter_w) == 2)
        sz_t1k1 = sz_t1k1_2;
        t1k1_t1k2 =  t1k1_t1k2_2;
        t1k2_t2k1 = t1k2_t2k1_2;
        t2k1_t2k2 = t2k1_t2k2_2;
        t2k2_sz = t2k2_sz_2;
     end
     if( type_w(Conter_w) == 3)
        sz_t1k1 = sz_t1k1_3;
        t1k1_t1k2 =  t1k1_t1k2_3;
        t1k2_t2k1 = t1k2_t2k1_3;
        t2k1_t2k2 = t2k1_t2k2_3;
        t2k2_sz = t2k2_sz_3;
    end
    Counter_cm = Counter_cm +1;
    Vaga_w_off(i+1) = Vaga_w_off(i);
    % З'їзд з платформи 1 колеса 1 візка
    if ( Counter_cm == sz_t1k1)
        Vaga_w_off(i+1) = Vaga_w_off(i) - ((1-
Vaga_1k(Conter_w))*Vaga_w_z(Conter_w));
    end
    % З'їзд з платформи 2 колеса 1 візка
    if ( Counter_cm ==  sz_t1k1+t1k1_t1k2)
        Vaga_w_off(i+1) = Vaga_w_off(i) -
(Vaga_1k(Conter_w)*Vaga_w_z(Conter_w));
    end
     % З'їзд з платформи 1 колеса 2 візка
    if ( Counter_cm == sz_t1k1+t1k1_t1k2+t1k2_t2k1)
        Vaga_w_off(i+1) = Vaga_w_off(i) - ((1-
Vaga_1k(Conter_w))*Vaga_w_z(Conter_w));
    end
    % З'їзд з платформи 2 колеса 2 візка
    if ( Counter_cm ==   sz_t1k1+t1k1_t1k2+t1k2_t2k1+t2k1_t2k2)
        Vaga_w_off(i+1) = Vaga_w_off(i) -
(Vaga_1k(Conter_w)*Vaga_w_z(Conter_w));
    end
    % Закінчення вагону
    if ( Counter_cm == sz_t1k1+t1k1_t1k2+t1k2_t2k1+t2k1_t2k2+ t2k2_sz)
        Conter_w = Conter_w + 1;
        Counter_cm = 0;
    end
    %Закінчення потягу
```

```
   if ( Conter_w == N_w + 1)
      break;
   end
end
```

Останнім етапом створення імітаційного сигналу є об’єднання двох векторів в один, оскільки вони є частиною одного процесу, для подальшого використання в дослідженнях.

```
%Формування кінцевого вектору ваги з ваговимірювальної платформи
for i = 1:length(Vaga_w_off)
   if(i <= length(Vaga_w_on))
      Vaga_w(i) = Vaga_w_on(i) + Vaga_w_off(i);
   else
      Vaga_w(i) = Vaga_w_on(length(Vaga_w_on)) + Vaga_w_off(i);
   end
end
```

Сформований сигнал можна побачити на рис. 2.1.

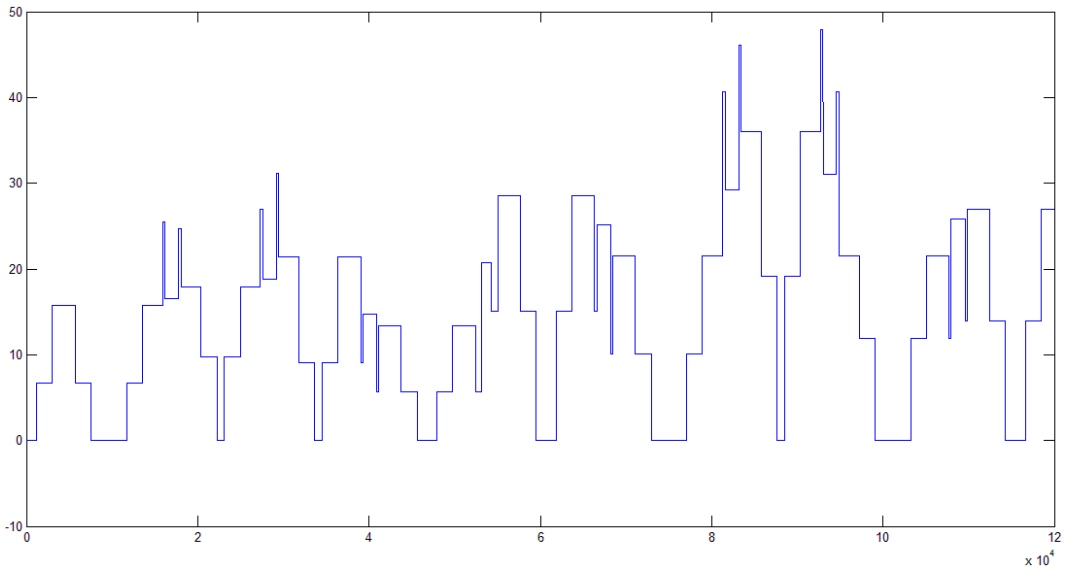

Рисунок 2.1 – Сигнал імітаційної моделі проїзду вагонів по платформі

Як можна побачити на рис. 2.1, напочатку перший стрибок в гору відповідає заїзду першої осі першого візка, другий – другої осі, відповідно стрибки вниз – з’їзд цих осей. Проміжок між попереднім і наступним стрибком – ситуація, коли платформа знаходиться між двома візками. Наступні два

стрибки мають той самий сенс, а третій символізує про заїзд на платформу візка наступного вагона. Стрибки вниз що йдуть після заїзду ще одного вагона, характерні ознаки з'їзду осей останнього візка попереднього вагона.

Створення імітаційної моделі проїзду вагонів по платформі є важливим інструментом для проведення досліджень з їх класифікації, тому потрібно розглянути усі аспекти що впливають на зміну сигналу. Вихідний сигнал з аналогових датчиків в дійсності не є ідеальним як на рис. 2.1, тому необхідно до моделі додати збурення.

Імітаційне збурення буде складатися з суми синусів за різними фазами (рис. 2.2), що дозволить створити випадкову складову, чого буде достатньо.

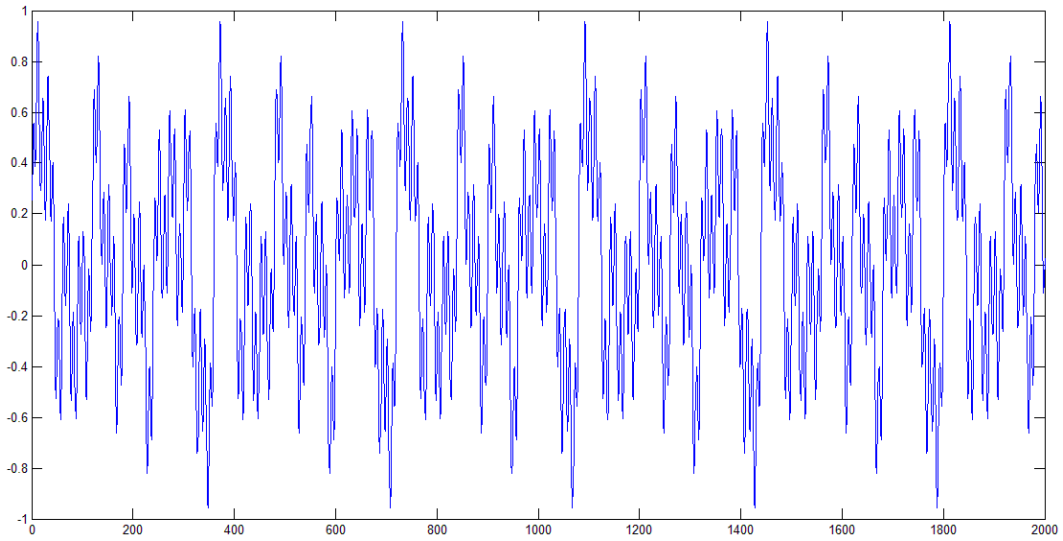

Рисунок 2.2 – Випадкова складова для імітації збурень проїзду вагона по платформі

Код нижче є реалізацією збурень і перехідний процес під впливом збурень наведено на рис. 2.3.

```
for i = 1:length(Vaga_w_off)
    Vaga_w_zb(i) =
Vaga_w(i)+((sin((i*2*pi)/10)+sin((i*2*pi)/30)+sin((i*2*pi)/60)+sin((i*2*pi)/90)+
sin((i*2*pi)/120))*0.25); end
```

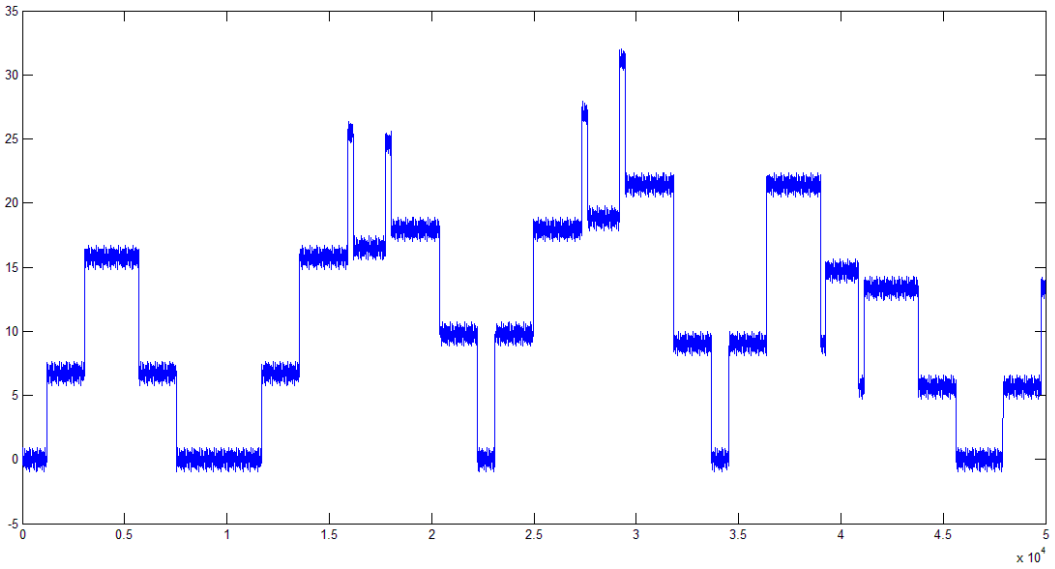

Рисунок 2.3 – Вихідний сигнал проїзду потяга по ваговимірювальної платформи зі збуреннями

Зараз вихідний сигнал розробленої імітаційної моделі знаходиться у системі міліметри/вага. Ця система не є подібною реальній, оскільки потяг рухається у часі зі швидкістю що змінюється в певному діапазоні. Зміна швидкості впливає на перехідний процес і змінює його. Тому створимо випадкову зміну швидкості за нормальним законом розподілу використовуючи функцію randn математичного пакету Matlab в змінну Speed_imp.

```
%Імітація зміни швидкості руху вагонів випадковим чином з періодом 10 с
Speed_imp = randn(100,1,'single')*2;
x=1:length(Speed_imp);
```

Швидкість потяга змінюється не так стрімко, як може згенерувати функція randn, тому необхідно провести інтерполяцію додав проміжні точки з шагом 1мс в вектор швидкості Speed для згладжування кривої (рис. 2.4).

```
%Інтерполяція вектору значень швидкості руху вагонів (крок - 1 мс)
i=1:0.0001:100;
Speed = interp1(x,Speed_imp,i);
```

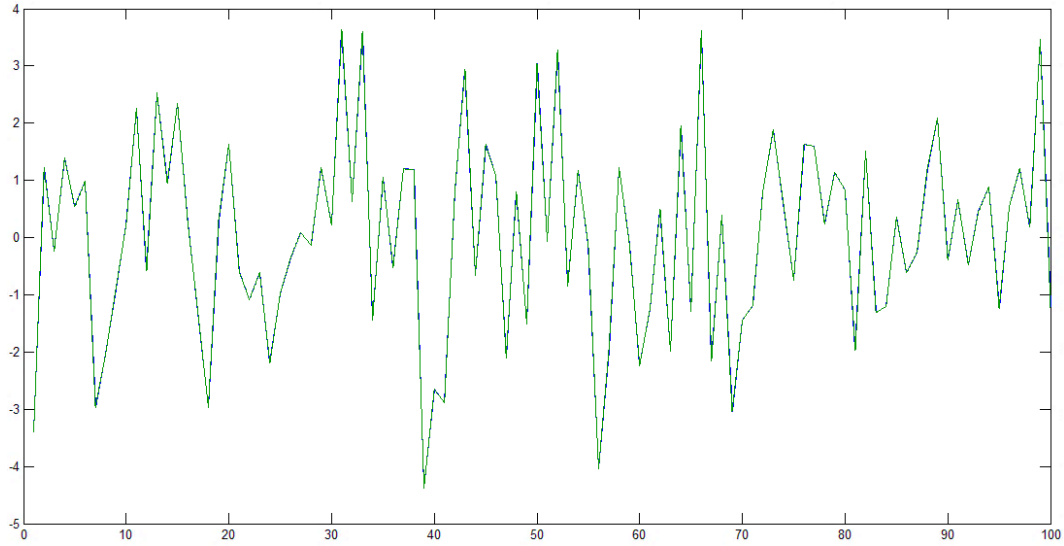

Рисунок 2.4 – випадкова (синя) і інтерпольована (зелена) швидкість

Перетворимо вектор Speed з системи кілометри/година до міліметри/мілі секунд розрахував по формулі t = S / V. За середню швидкість потяга взято 10 км/годину.

```
%Перехід від розмірності за віссю X від мм до мс за формулою t = S / V
WW=0;
%Перехід від км/год до мм/мс з додаванням середньої швидкості
Speed = (Speed +10)*(1000000/3600000);
for i=1:1000000
   %Розрахунок, скільки усього мм проїхав потяг через 1 мс
   WW = WW + Speed(i);
   if(WW>length(Vaga_w))
      break
   end
   Vaga_W(i) = Vaga_w(floor(WW));
end
```

Використана змінна WW є значенням поточного моменту зважування у часі, Vaga_W – вектор значень зважувань ваги у системі мілі секунди/вага. Перетворений перехідний процес наведено на рис. 2.5.

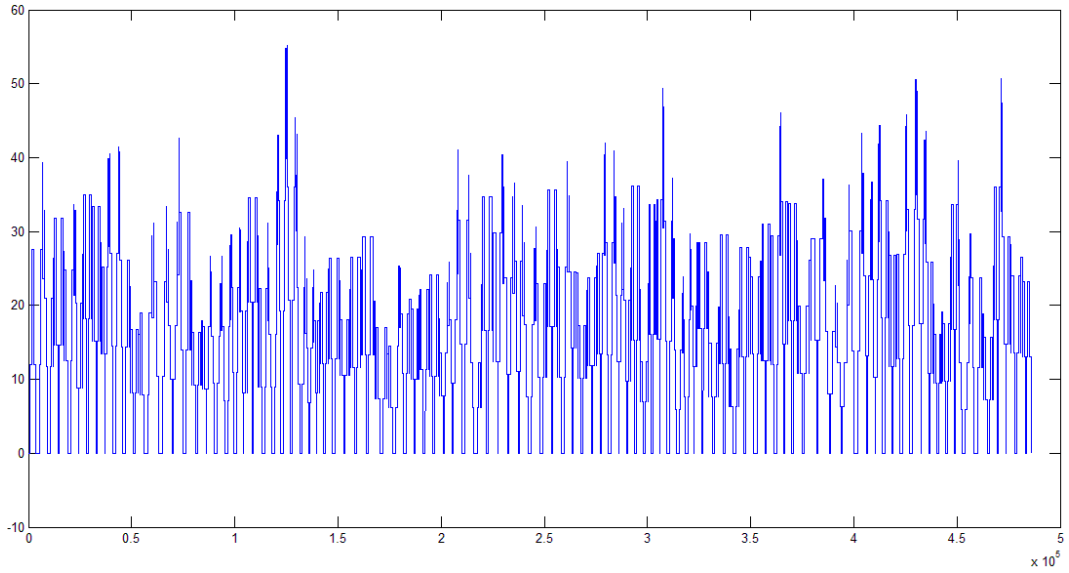

Рисунок 2.5 – Перетворений перехідний процес з мм у мс

Вихідний сигнал на рис. 2.5 відповідає очікуванням з імітаційної моделі проїзду потяга з вагонами по ваговимірювальної платформі. Надалі його будемо використовувати для проведення досліджень з класифікації типів залізничних вагонів.

2.2 Висновки до розділу

Цей розділ був присвячений розробці імітаційної моделі проїзду потяга з вагонами по ваговимірювальної платформі, з фіксацією зважувань ваги у сховище даних. Розглянута зміна швидкості потяга у часі і додана у модель як збурення. Надалі імітаційна модель буде використовуватись для проведення досліджень класифікації типів вагонів у третьому розділі кваліфікаційної роботи.

3 СИНТЕЗ ТА ДОСЛІДЖЕННЯ СИСТЕМИ АВТОМАТИЧНОГО КЕРУВАННЯ

3.1 Алгоритм та програма визначення конструктивних параметрів вагонів

У цьому розділі буде розглядатись алгоритм визначення конструктивних параметрів вагонів за отриманим вихідним сигналом розробленої моделі у другому розділі кваліфікаційної роботи. Вихідний сигнал моделі представлений як пройдений час вагонів відносно платформи за віссю x і вимірюється у мілі секундах, а за віссю y – зважена вага вагону в тонах. Звичайно у системах зважування дані збираються за часом, тому для визначення типу вагона за конструктивними параметрами необхідно перетворити вихідний сигнал до системи міліметри/вага. Для цього напишемо програму:

```
%Перехід від мс до мм
W=0;
for i=1:1000000
   %Розрахунок, скільки усього мм проїхав потяг через 1 мс
   W = W + Speed(i);
   if(i>length(Vaga_W))
      break
   end
   %Заповнення проміжних точок поточними значеннями ваги
   for j=1:ceil(Speed(i))-1
      Vaga_Wmm(ceil(W)-j) = Vaga_W(i);
   end
   Vaga_Wmm(ceil(W)) = Vaga_W(i);
end
```

Перетворений сигнал із системи мс/вага до мм/вага наведено на рис. 3.1. Надалі необхідно диференціювати сигнал та знайти моменти заїзду осей візка для визначення конструктивних параметрів:

```
Deriv_Vaga = [];
for i=2:length(Vaga_Wmm)
   Deriv_Vaga(i) = Vaga_Wmm(i) - Vaga_Wmm(i-1); end
```

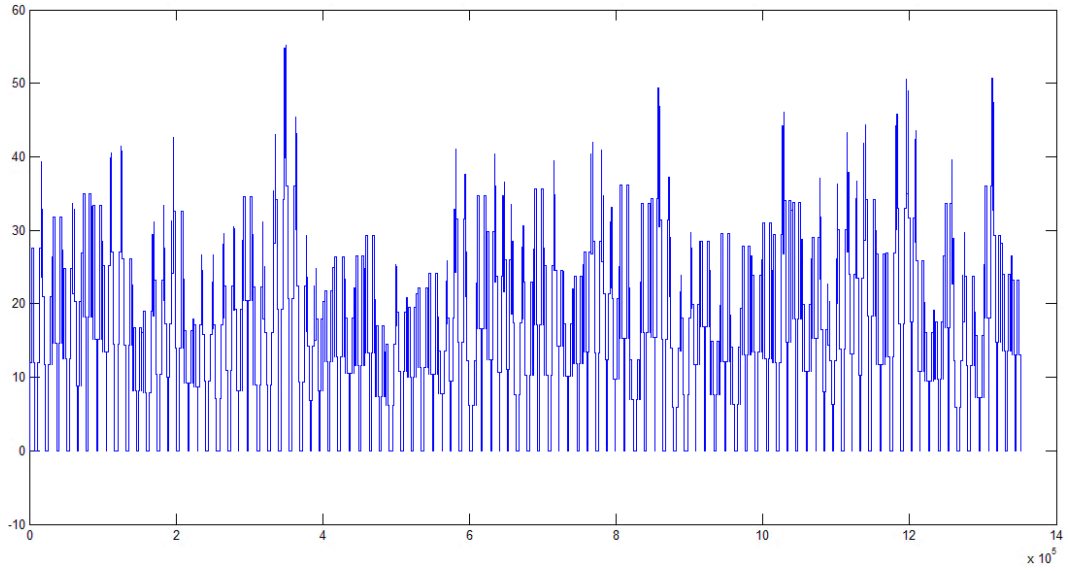

Рисунок 3.1 – Перетворений вихідний сигнал з імітаційної моделі з системи мс/вага до мм/вага

Графік диференційованого вихідного сигналу наведено на рис. 3.2 (наведена частина всього сигналу в інтервалі від 0 до $3.5 \cdot 10^5$). Бачимо що на графіку є диференціали на заїзд та з'їзд осі, скористаємось верхньою частиною для визначення параметрів.

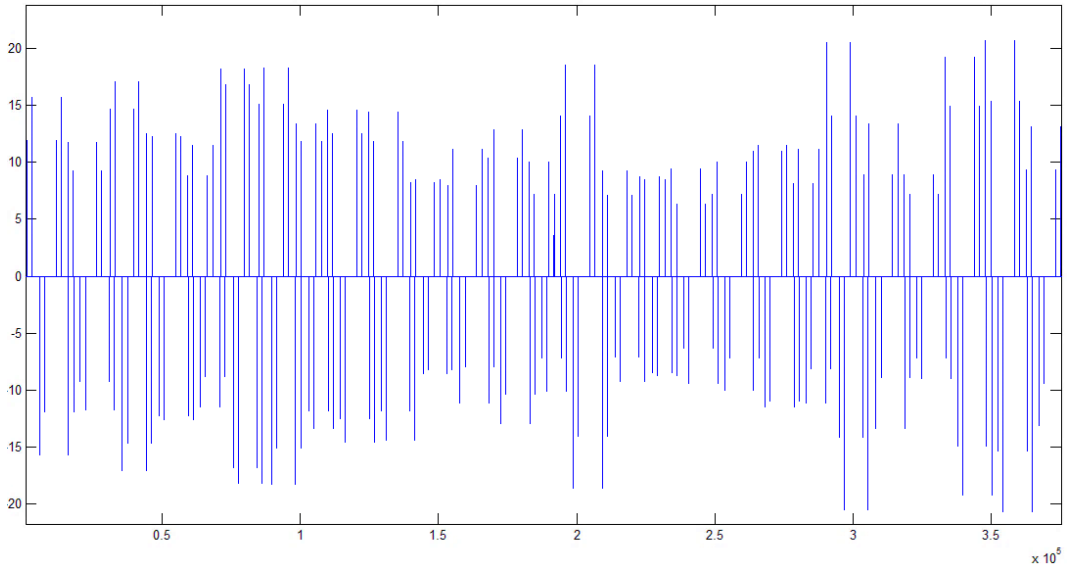

Рисунок 3.2 – Диференційований вихідний сигнал імітаційної моделі

При дослідженні диференційованого сигналу, було помічено що диференціал не досягає абсолютного нуля, тому визначимо діапазон

виникнення цих значень біля нуля скориставшись функцією std математичного пакету Matlab:

```
stdW = std(Deriv_Vaga);
```

На виході маємо stdW = 0.3067, значить що випадкові значення лежать в інтервалі 0 ± 0.3067. Оскільки в наведеному нижче алгоритмі за головний інформаційний критерій взятий заїзд осей, необхідно визначити мінімальне значення порога спрацювання лічильників сигналу (заїзду осей вагона) і воно дорівнюватиме значенню дисперсії похибки stdW помножену на десять:

```
level_Deriv_Vaga = 10*stdW;
```

Що дорівнюватиме 3.0668.

Надалі визначимо початкові змінні:

```
Counter_osn = 0;
Counter_1viz = 0;
Counter_2viz = 0;
Counter_w = 1;
Osn_w=[0];
Kol_1viz = [];
Kol_2viz = [];
buffosn = 0;
```

де Counter_osn – лічильник визначення осності вагонів, Counter_1viz – лічильник визначення відстані між осями візка, Counter_2viz – лічильник визначення відстані між внутрішніми осями сусідніх візків, Counter_w – лічильник кількості вагонів що проїхали по платформі, Osn_w – вектор визначеної кількості осей у вагона, Kol_1viz – вектор визначеної відстані між осями одного візка, Kol_1viz – вектор визначеної відстані мід осями сусідніх візків вагона, buffosn – буфер для запису кількості осей у вагона.

Визначивши всі початкові змінні, розрахував всі необхідні для алгоритму значення, далі наведено код програми що знаходить конструктивні параметри вагона за вихідним сигналом з імітаційної моделі:

```
T = length(Vaga_Wmm);
for i=2:T
```

```
   if Deriv_Vaga(i) > level_Deriv_Vaga
      Counter_osn = Counter_osn + 1;
   end
   if(Vaga_Wmm(i) < level_Deriv_Vaga && ...
         Vaga_Wmm(i-1) > level_Deriv_Vaga && ...
         Counter_osn ~= 0)
      buffosn = Counter_osn * 2;
   end
   % counter 2
   if Counter_osn == 1
      Counter_1viz = Counter_1viz + 1;
   end
   % counter 3
   if Deriv_Vaga(i) > level_Deriv_Vaga && ...
         (buffosn == 0 || Counter_osn < buffosn / 2 + 1)
      Counter_2viz = 0;
   end
   if Counter_osn ~= (buffosn / 2 + 1)
      Counter_2viz = Counter_2viz + 1;
   end
   if mod(Counter_osn, buffosn) == 0 && buffosn > 0
      Osn_w(end + 1) = buffosn / 2;
      Kol_1viz(end + 1) = Counter_1viz;
      Kol_2viz(end + 1) = Counter_2viz;
      Counter_osn = 0;
      buffosn = 0;
      Counter_1viz = 0;
      Counter_2viz = 0;
   end
end
```

Перший лічильник Counter_osn збільшує своє значення на одиницю, коли поточне значення ваги більше ніж мінімальне значення порога. Робота лічильника наведена на рис. 3.3.

При визначенні параметрів вагона, важливою частиною класифікації типу є визначення кількості осей візка. Передбачити скільки осей має візок неможливо, тому орієнтуючись на характерний признак зникнення рівня сигналу, що свідчить про відсутність жодної осі вагона на платформі, визначимо кількість осей візка і також кількість осей всього вагона. Додатково ще перевіряється умова кількості осей – вона не повинна дорівнюватиме нулю,

оскільки є моменти, коли вагон закінчився і очікується наступний. Якщо умова виконується, значення buffosn дорівнюватиме поточній кількості осей помноженій на два, що свідчить про кількість осей всього вагона.

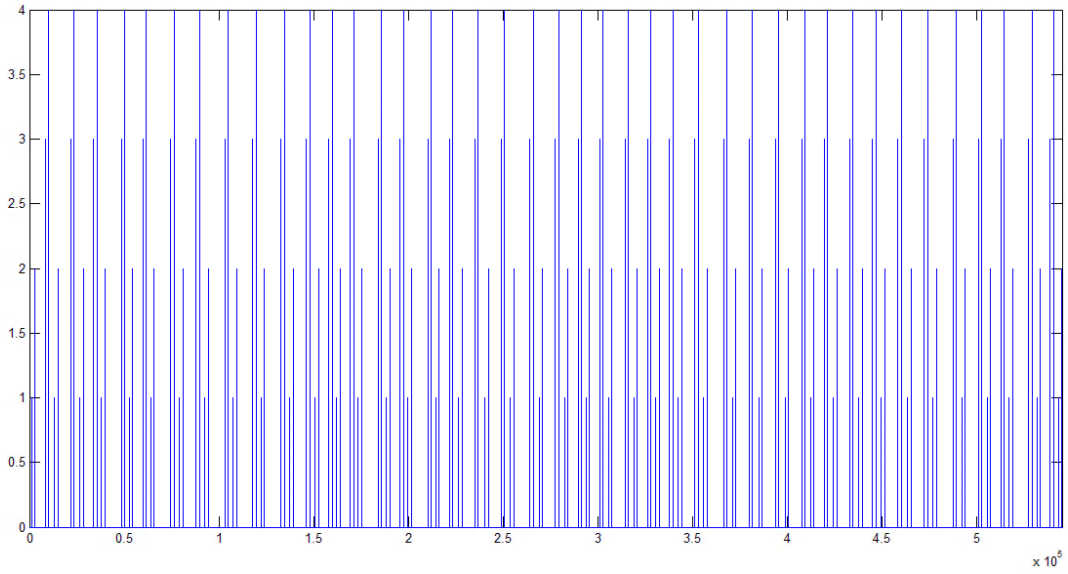

Рисунок 3.3 – Робота лічильника Counter_osn

Другий лічильник Counter_1viz - відстань між осями візка, збільшує своє значення на одиницю, коли значення лічильника кількості осей Counter_osn дорівнюватиме одиниці. Робота лічильника наведена на рис. 3.4.

Третій - лічильник відстані між внутрішніми осями візків, починає збільшувати своє значення на одиницю, завдяки умові з розрахунком відносної кількості осей: якщо поточне значення осей вагона не дорівнюватиме значенню buffosn / 2 + 1. Но є багато випадків, коли умова буде виконуватись і кінцеве значення не буде відповідати зазначеним параметрам вагона, тому потрібна додаткова умова скидання нарахувань лічильника: якщо поточна вага більше ніж мінімальне значення порога і значення buffosn дорівнюватиме нулю або лічильник осей менше ніж buffosn / 2 + 1, значення скидається в нуль. Це інтервальна умова, яка буде працювати і на більшій кількості осей візка. Робота лічильника наведена на рис. 3.5.

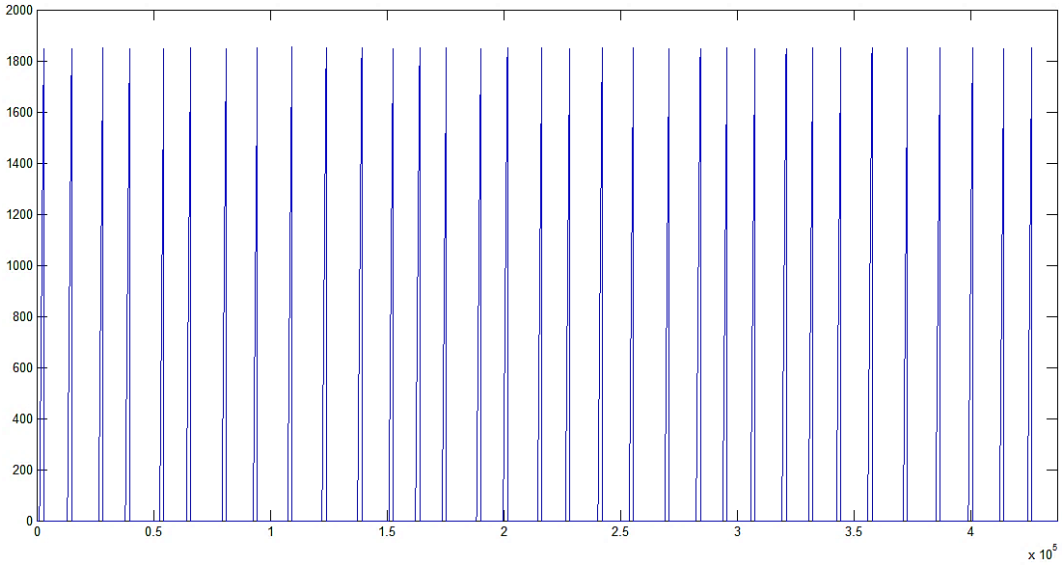

Рисунок 3.4 – Робота лічильника Counter_1viz

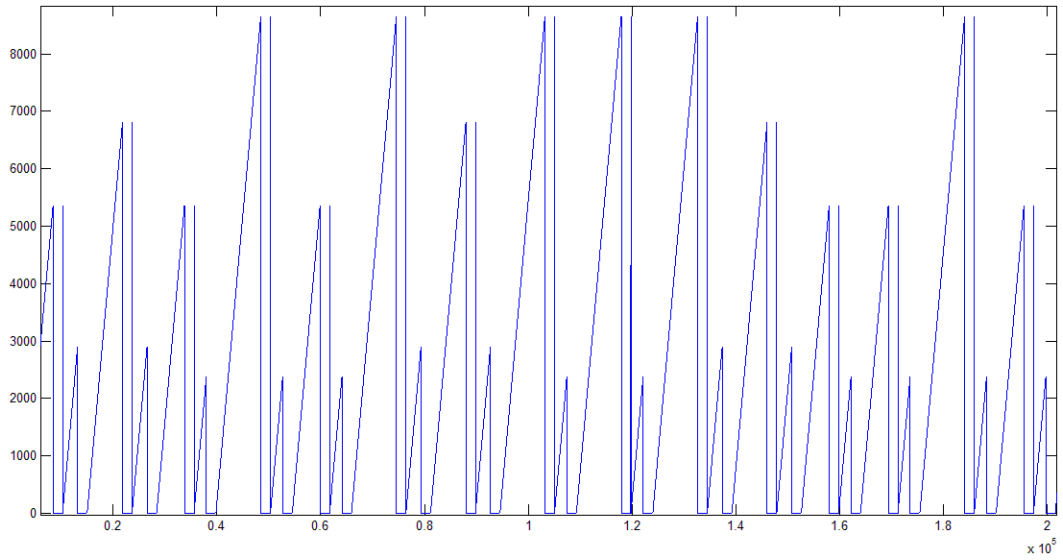

Рисунок 3.5 – Робота лічильника Counter_2viz

Після закінчення вагона, необхідно скинути всі лічильники в нуль. Скидання виконується за умовою, коли поточна кількість осей вагона Counter_osn дорівнюватиме значенню buffosn. Оскільки задачею стоїть провести класифікацію вагона за його параметрами, значення лічильників збережемо в відповідних параметрам векторах.

3.2 Застосування нейронної мережі для класифікації типів вагонів

При дослідженні сигналу з ваговимірювальної платформи розраховані значення параметрів вагона за сигналом лежать в певному діапазону відхилень. Для підвищення якості класифікації типів вагонів вирішено запобігти методам нейронних мереж з самонавчанням. Обрана модель нейронної мережі Перцептрон дозволяє проводити складну класифікацію типів незважаючи на свою простоту. Загальна структура обраної нейронної мережі розглянута на рис. 3.6.

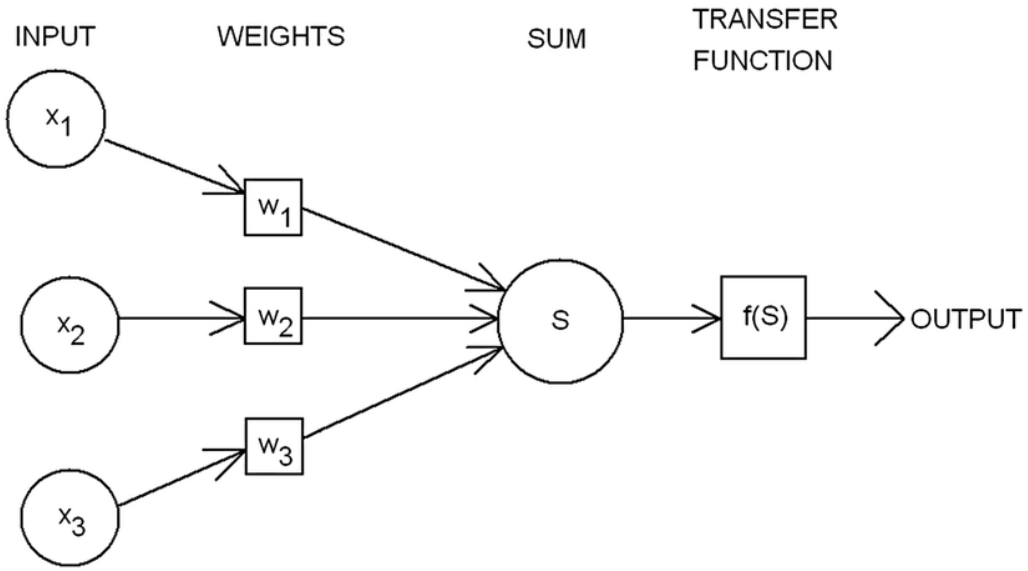

Рисунок 3.6 – Загальна структура Перцептрона

Нейронна мережа складається з трьох S, A і R слоїв. Де S (від анг. Sensors) – це значення з датчиків – вхідні параметри (на рис 3.6 це INPUT з x_1-x_3), A (від анг. Association) – шар що асоціює групу певних сенсорів зв'язаних між собою (на рис.3.6 це SUM) і R (від анг. React) – шар з активацією нейрона – TRANSFER FUNCTION, функція яка перетворює значення що надходить з A шару в значення що лежить в певному діапазоні.

Вага (weights) між S-A шарами э оцінка корисності інформаційного параметра що надходить з шару S.

Функції активації є різні, наприклад, сігмоіда, яка використовується як для посилення слабких сигналів так і не насичуватись від сильних (рис. 3.7а), ступінчаста (функція Хевісайда, рис. 3.7б) та інші.

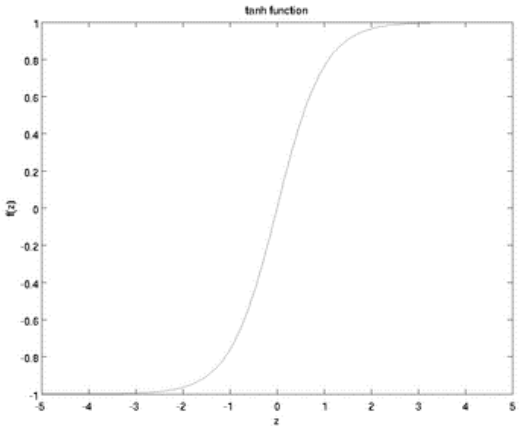

рис.3.7а - Сигмоіда

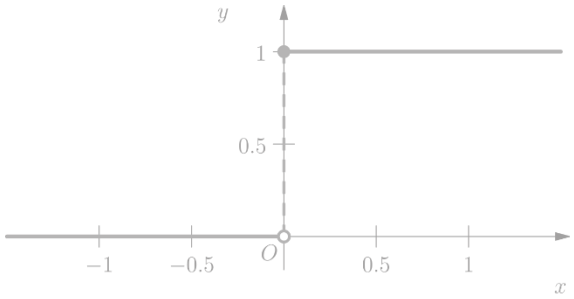

рис.3.7б – функція Хевісайда

Обрана нейронна мережа Перцептрон дозволить провести класифікацію типів вагонів за визначеними вихідними параметрами з імітаційної моделі за функцією активації Хевісайда. Тому виходячи з розглянутого матеріалу, розробимо програмний код.

Визначимо початкові змінні для частоти потраплянь конструктивних параметрів

```
N_Kol_1viz(1) = 0;
N_Kol_2viz(1) = 0;
N_Osn_w(1) = 0;
```

За першим інформаційним критерієм – відстань між осями одного візка

```
for i=1:length(Kol_1viz)
   for j=1:10000
      %Перше заповнення вектору N_Kol_1viz
      if(length(N_Kol_1viz) == j)
         N_Kol_1viz(j+1) = 0;
      end
      %Перевірка потрапляння значення критерію
      if(Kol_1viz(i) == j)
         N_Kol_1viz(j) = N_Kol_1viz(j)+1;
      end
```

```
   end
end
```

Результат дослідження частоти потраплянь за першим критерієм

```
>> t = [find(N_Kol_1viz>0); N_Kol_1viz(find(N_Kol_1viz))]

t =

        1847        1848        1849        1850        1851        1852        1853
           1           7          39          29          18           5           1
```

За другим інформаційним критерієм – відстань між внутрішніми осями візків вагона визначимо частоту потраплянь

```
for i=1:length(Kol_2viz)
   for j=1:10000
      %Перше заповнення вектору N_Kol_2viz
      if(length(N_Kol_2viz) == j)
         N_Kol_2viz(j+1) = 0;
      end
      %Перевірка потрапляння значення критерію
      if(Kol_2viz(i) == j)
         N_Kol_2viz(j) = N_Kol_2viz(j)+1;
      end
   end
end
```

Результат дослідження частоти потраплянь за другим критерієм

```
>> t = [find(N_Kol_2viz>0); N_Kol_2viz(find(N_Kol_2viz>0))]

t =

  Columns 1 through 7

       5347        5348        5349        5350        5351        5352        6797
          1           1           9           7           8           4           1

  Columns 8 through 14

       6798        6799        6800        6801        6802        8647        8648
          5           6          10           7           2           1           1

  Columns 15 through 18

       8649        8650        8651        8652
         10          14           9           4
```

За третім інформаційним критерієм – кількість осей візка

```
for i=1:length(Osn_w)
   for j=1:4
      %Перше заповнення вектору N_Kol_2viz
      if(length(N_Osn_w) == j)
         N_Osn_w(j+1) = 0;
```

```
        end
        %Перевірка потрапляння значення критерію
        if(Osn_w(i) == j)
            N_Osn_w(j) =  N_Osn_w(j)+1;
        end
    end
end
```

Результат дослідження частоти потраплянь за третім критерієм

```
>> t = [find(N_Osn_w>0); N_Osn_w(find(N_Osn_w>0))]

t =

     2
   100
```

Надалі знайдені частоти потраплянь за критеріями буде використано для активацій А шарів нейронної мережі.

При визначенні параметрів розробленим алгоритмом у п.п. 3.1 значення конструктивних параметрів лежать у певному інтервалі. Визначимо цей інтервал для всіх критеріїв.

Для першого критерія

```
%Оновлення функцій ваги (обрані прямокутної форми, ширина при цьому
%охоплює максимальний розкид значень інформаційного критерію)
%Для першого інформаційного критерію – відстань між осями візка
a_it=1;
b_it=1;
for i=1:length(N_Kol_1viz)-1
    if(N_Kol_1viz(i+1) ~= 0 && N_Kol_1viz(i) == 0)
        a_Kol_1viz(a_it) = i+1;
        a_it = a_it +1;
    end
    if(N_Kol_1viz(i+1) == 0 && N_Kol_1viz(i) ~= 0)
        b_Kol_1viz(b_it) = i;
        b_it = b_it +1;
    end
end
```

Результат знаходжень інтервалу за частотою потраплянь першого критерію

```
>> [a_Kol_1viz b_Kol_1viz]

ans =

        1847        1853
```

Додомо функцію ваги А елементу нейронної мережі за знайденим інтервалом в змінну A_Kol_1viz для розрахунку ваги шару R

```
for i=1:length(a_Kol_1viz)
   A_Kol_1viz(i).f =
@(x)(0.0.*(x<a_Kol_1viz(i)))+(1.0.*(x>=a_Kol_1viz(i)).*(x<=b_Kol_1viz(i)))+(0
.0.*(x>b_Kol_1viz(i)));
end
```

де функція f змінної A_Kol_1viz є прямокутною функцією активації.

За другим критерієм

```
%Для другого інформаційного критерію
%Визначення параметрів фунцій ваги А-елементів персептрону
a_it=1;
b_it=1;
for i=1:length(N_Kol_2viz)-1
   if(N_Kol_2viz(i+1) ~= 0 && N_Kol_2viz(i) == 0)
      a_Kol_2viz(a_it) = i+1;
      a_it = a_it +1;
   end
   if(N_Kol_2viz(i+1) == 0 && N_Kol_2viz(i) ~= 0)
      b_Kol_2viz(b_it) = i;
      b_it = b_it +1;
   end
end
```

Додомо функцію ваги А елементу нейронної мережі за знайденим інтервалом в змінну A_Kol_2viz для розрахунку ваги шару R

```
Функція активації  шару А для другого критерію
for i=1:length(a_Kol_2viz)
   A_Kol_2viz(i).f =
@(x)(0.0.*(x<a_Kol_2viz(i)))+(1.0.*(x>=a_Kol_2viz(i)).*(x<=b_Kol_2viz(i)))+(0
.0.*(x>b_Kol_2viz(i)));
end
```

За третім критерієм

```
%Для третього інформаційного критерію
%Визначення параметрів фунцій ваги А-елементів персептрону
a_it=1;
b_it=1;
for i=1:length(N_Osn_w)-1
   if(N_Osn_w(i+1) ~= 0 && N_Osn_w(i) == 0)
   a_Osn_w(a_it) = i+1;
   a_it = a_it +1;
   end
   if(N_Osn_w(i+1) == 0 && N_Osn_w(i) ~= 0)
   b_Osn_w(b_it) = i;
   b_it = b_it +1;
   end
end
```

Додомо функцію ваги А елементу нейронної мережі за знайденим інтервалом в змінну A_Osn_w для розрахунку ваги шару R

```
%Оновлення функцій ваги для А-елементів персептрону (3 критерій)
for i=1:length(a_Osn_w)
   A_Osn_w(i).f =
@(x)(0.0.*(x<a_Osn_w(i)))+(1.0.*(x>=a_Osn_w(i)).*(x<=b_Osn_w(i)))+(0.0.*(x
>b_Osn_w(i)));
end
```

Проведемо реалізацію класифікації типів вагонів. Визначимо більш інформаційні параметри вагону шляхом перемноження функцій ваги А шарів.

```
%Реалізація процедури класифікації типів вагонів
% Визначення активних ланцюжків для активації R-елементів перцептрону
%Цикл перебору вагонів
for i = 1:N_w
   %Цикли для визначення ваг різних ланцюжків персептрону (активація
      відбувається
   %шляхом перемноження функцій ваги А-елементів)
   for j = 1:length(A_Osn_w)
      for q = 1:length(A_Kol_2viz)
         for w = 1:length(A_Kol_1viz)
            if(A_Kol_2viz(q).f(Kol_2viz(i))*A_Kol_1viz(w).f(Kol_1viz(i))*
                  A_Osn_w(j).f(Osn_w(i)) == 1)
               R_matrix(i,1) = w;
               R_matrix(i,2) = q;
```

```
                R_matrix(i,3) = j;
            end
        end
    end
  end
end
```

Результатом досліджень інформаційного критерія є

```
>> R_matrix

R_matrix =

  Columns 1 through 15

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     3     3     2     3     1     2     2     1     3     3     2     3     3     1     3
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 16 through 30

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     2     1     3     3     3     1     2     3     3     3     3     3     1     1     2
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 31 through 45

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     3     2     2     2     2     3     3     2     3     2     1     1     3     3     2
     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 46 through 60

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     3     3     2     3     1     2     2     1     3     1     1     2
```

Бачимо що другий критерій в цій матриці і третій критерій за цім розділом є інформаційним за інші. Тому скористаємось їм для визначення типу вагона

```
% Активація R-елментів персептрону
 for i = 1:N_w
    if(R_matrix(i,1)==1 && R_matrix(i,2)==1 && R_matrix(i,3)==1)
       type_w_klass(i) = 3;
    end
    if(R_matrix(i,1)==1 && R_matrix(i,2)==2 && R_matrix(i,3)==1)
       type_w_klass(i) = 1;
    end
    if(R_matrix(i,1)==1 && R_matrix(i,2)==3 && R_matrix(i,3)==1)
       type_w_klass(i) = 2;
    end
 end
```

Всі класифіковані типи вагонів зберігаються у вектор type_w_klass для подальшого використовування у четвертому розділі з додаванням результатів класифікації у людино-машинний інтерфейс.

3.3 Висновки з розділом

У розділі було розроблено алгоритм визначення конструктивних параметрів залізничних вагонів з використанням методів перетворення сигналу з системи мілісекунди/вага до міліметри/вага. При перетворенні сигналу було помічено що значення параметрів лежать у певному діапазону від очікуваної величини. Тому за розробленим алгоритмом з використанням нейронної мережі Перцептрон, було проведено класифікацію типів залізничних вагонів з підвищенням рівня аналізу критеріїв. Оскільки похибка що може виникнути при перетворенні сигналу може лежати від локальної зони очікуваного значення значно далеко.

4 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

Необхідність спеціалізованого програмного забезпечення на робочому місці оператора є важливою складовою технологічного процесу. Програмне забезпечення повинно відповідати всім потребам оператора, який обробляє інформацію за наданим сигналом з вагового терміналу. Тобто, основними функціями є: вікно перегляду перехідного процесу, місце з вихідними параметрами системи (тип вагона), базу знань вагонів, проміжні графіки (лічильники, диференціальний сигнал), завантаження сигналу з зовні і збереження всіх визначених параметрів у файл. Виходячи з цього, розробимо людино-машинний інтерфейс, в якому для проведення класифікації типів вагонів буде використовуватись програмний код алгоритму класифікації типів вагонів, розроблений у третьому розділі кваліфікаційної роботи.

4.1 Створення графічної складової

Розробимо інтерфейс в інтегрованому пакеті guide математичного пакету Matlab. Головними критеріями розробки людино-машинного інтерфейсу для оператора є зручність і мінімалізм, оскільки ці критерії впливають на ефективність праці. Людино-машинний інтерфейс повинен мати активні і пасивні елементи взаємодії з оператором: інформаційні елементи для зображення всіх вихідних (або проміжних) параметрів для проведення додаткових досліджень, впливати на генерацію сигналу імітаційної моделі змінюючи кількість вагонів проїжджаючих по платформі. Також важними елементами кожного інтерфейсу з обробкою даних є можливість завантажувати сигнал з зовнішнього сховища – вагового терміналу – в програмне забезпечення і зберігати внутрішні визначені параметри у зовнішнє середовище – базу даних.

Виходячи з поставлених критеріїв була розроблена графічна складова людино-машинного інтерфейсу без функціоналу (рис 4.1).

Рисунок 4.1 – Розроблена графічна складова без функціоналу

На верхній частині інтерфейсу додано меню, в якому можна обрати дію «завантажити» або «зберегти» (рис. 4.2).

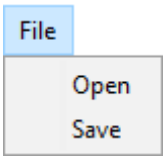

Рисунок 4.2 – Меню вибору з завантаженням сигналу або збереженням даних

Наступними розглянемо ряд елементів, які є активною частиною оператора це кнопки: «Згенерувати сигнал», «Визначити параметри» і поле для вводу «Кількість вагонів» (рис 4.3).

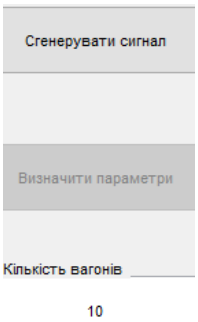

Рисунок 4.3 – Активні елементи оператора

Зображення перехідного процесу буде зображуватись на центральній частині інтерфейсу. Окрім перехідного процесу, можна також буде переглянути графіки лічильників з алгоритму визначення параметрів, натиснувши на відповідні кнопки поверх графіку. В верхньому правому боці поверх графіка є елемент «plot» що дозволить відкрити активний графік в окреме вікно з додатковими можливостями перегляду. Переглянути графік з його елементами взаємодії можна на рис. 4.4.

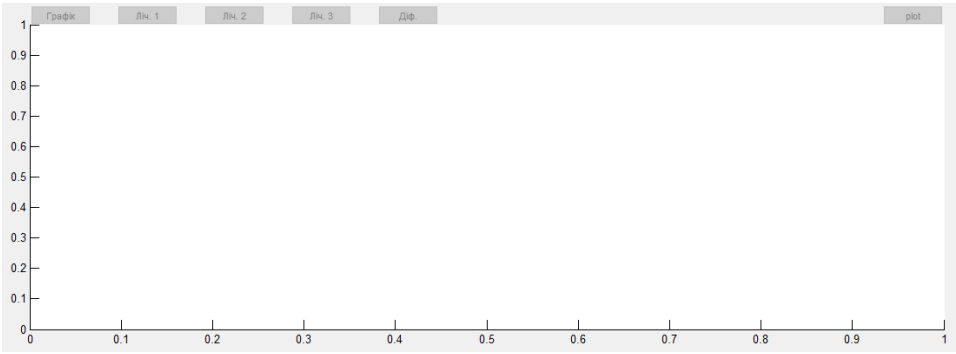

Рисунок 4.4 – Графік з елементами перемикання і кнопкою plot

Діючу базу знань параметрів вагонів можна подивитись на першій таблиці рис. 4.5, а в другій таблиці буде виводитись класифікований за алгоритмом тип вагона. Поруч з другою таблицею знаходяться значення діапазону змін величини виявленої у алгоритмі і розрахований допустимий мінімальний поріг.

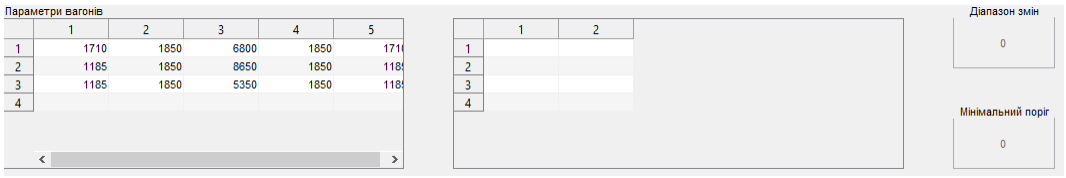

Рисунок 4.5 – Вхідні і вихідні параметри алгоритму класифікації типу вагонів

Останніми активними елементами інтерфейсу є вторинні кнопки: скидання стану програмного забезпечення до початкового і оновлюючий елемент, який є допоміжним для завантаження сигналу з зовнішнього середовища (рис. 4.6).

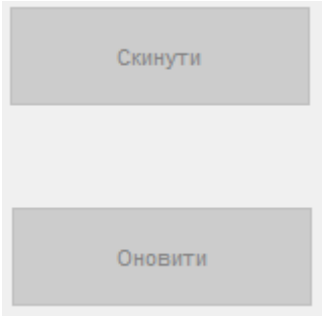

Рисунок 4.6 – Допоміжні кнопки скидання стану та оновлення графіку при завантаженні сигналу

Розроблена графічна частина інтерфейсу відповідає потребам візуалізації сигналу з імітаційної моделі.

4.2 Система взаємодії елементів

Всі графічні елементи мають вбудовану систему подій, які визиваються при конкретній дії людини пов’язаної з певним елементом. Самі елементи пов’язані зі структурою-обробником handles, з якого можна визвати той чи інший елемент для отримання параметру що він зберігає.

Однією з найпоширеніших подій є функція Callback,

function pushbutton1_Callback(hObject, eventdata, handles),

яка визивається в момент натискання на кнопку або зміні тексту у полі вводу і змінивши активний фокус тощо. В якості параметрів функції передається hObject – елемент що визиває цю подію, eventdata – зарезервована змінна розробниками пакету guide, handles – структура-обробник елементів. Тому параметри hObject і handles будуть часто використовуватись у розробці функціональної частини інтерфейсу.

Часто використовуються функції get і set. Функція get приймає елемент і назву параметра, значення якого потрібно отримати. Функція set – приймає елемент, назву параметра і значення яке потрібно встановити.

Для зміни внутрішньої структури handles використовується функція guidata, на вхід якої потрібно передати параметри handles і елемент, з якого ця функція визивається.

4.3 Розробка програмного забезпечення людино-машинного інтерфейсу

Розробленій графічній частині у п. п. 4.1 потрібно розробити функціональну складову, яка буде реагувати на натискання кнопок, або на взаємодію з системою файлів. Тому далі йде розглядання функціональної складової людино-машинного інтерфейсу.

Оскільки взаємодія між елементами є складною системою «під капотом», напочатку файлу з реалізацією подій йдуть необхідні початкові налаштування для коректної роботи, їх розглядати не будемо – вони статичні.

У функції pushbutton1_Callback зчитується значення змінної що відповідає кількості вагонів у змінну countw новоствореної структури custom. Вона э частиною структури handles. Потім countw передається у функцію fGeneratorSignal для формування вихідного сигналу імітаційної моделі. Вихідне значення fGeneratorSignal заноситься у змінну fgen структури custom. Функцією fTrainSpeed2 генерується випадкова складова – збурення – зміна швидкості потяга, в якості параметрів якої передається змінна fgen структури custom. Функцією fTrainSpeed2 на виході формується комірка з ще двох комірок: перша комірка - вихідний сигнал зі збуреннями, зберігається у змінну trspeed структури custom, в другій – нормалізована швидкість за часом, зберігається у змінну speed структури custom. Останніми діями обирається елемент зображення графіків – axes1 функцією обрання axes, скидається його стан функцією cla та заносяться нові дані в plot(). Оскільки в цій функції заносились (або змінюватимуться далі) данні, потрібно підтвердити зміну структури custom функцією guidata передавши в якості параметрів handles і hObject. Надалі структура handles і підструктура custom згадуватись не будуть, а розглядатимуться тільки змінні.

```
% функція генерації сигналу
function pushbutton1_Callback(hObject, eventdata, handles)
   handles.custom.countw = str2num(get(handles.edit1, 'String'));
   handles.custom.fgen = fGeneratorSignal(handles.custom.countw);
```

```
outfTrainSpeed2 = fTrainSpeed2(handles.custom.fgen);
handles.custom.trspeed = cell2mat(outfTrainSpeed2(1));
handles.custom.speed = cell2mat(outfTrainSpeed2(2));

axes(handles.axes1);
cla;
plot(handles.custom.trspeed);

guidata(hObject, handles);

set(handles.pushbutton1, 'Enable', 'off');
set(handles.pushbutton8, 'Enable', 'on');
set(handles.pushbutton10, 'Enable', 'on');
set(handles.pushbutton3, 'Enable', 'on');
set(handles.edit1, 'Enable', 'off');
```

У таблиці 4.1 розглянуто всі змінні що використані при розробці програмного забезпечення і стосуються графічного інтерфейсу.

Таблиця 4.1 – Використанні змінні

Назва змінної	**Призначення**
pushbutton1	Згенерувати сигнал
pushbutton3	Перемикач графіку у окремо вікно
pushbutton5	Перемикач поточного графіка на графік перехідного процесу
pushbutton6	Перемикач графіка на лічильник 1
pushbutton7	Перемикач графіка на лічильник 2
pushbutton8	Визначити параметри
pushbutton9	Перемикач графіка на лічильник 3
pushbutton10	Скинути стан ПЗ
pushbutton11	Перемикач графіка на діф. сигнал
pushbutton12	Оновити стан
edit1	Кількість вагонів
edit3	Діапазон змін
edit4	Мінімальний поріг
uitable1	База знань параметрів вагонів
uitable2	Класифіковані поточні вагони
Untitled_1	Головне меню
Untitled_2	Відкрити / завантажити файл
Untitled_3	Зберегти дані

Функція pushbutton8_Callback є активним елементом графічного інтерфейсу, що запускає алгоритм класифікації типів вагонів. У тілі функції визивається функція fDataForming3, в якості аргументів якій передаються змінні trspeed, speed і countw. На виході передається комірка з даними щодо проведення класифікації та проміжними результатами. Всі вони записуються у відповідні змінні. І далі використовуючи визначені параметри зі змінної outDataForming3 проводимо класифікацію вагонів за їх параметрами нейронною мережею perceptron. На виході функції fPerceptron формується вектор з комірками з назвами вагонів який присвоюється змінній wclass. Вектор wclass передається у таблицю uitable2 в якості аргумента для відображення його елементів на графічному інтерфейсі.

```
function pushbutton8_Callback(hObject, eventdata, handles)
   outDataForming3 = fDataForming3(handles.custom.trspeed,
handles.custom.speed, handles.custom.countw);

handles.custom.deriv_vaga = outDataForming3{1};
handles.custom.osn_w = outDataForming3{2};
handles.custom.kol_1viz = outDataForming3{3};
handles.custom.kol_2viz = outDataForming3{4};
handles.custom.counter1 = outDataForming3{7};
handles.custom.counter2 = outDataForming3{8};
handles.custom.counter3 = outDataForming3{9};
handles.custom.interval = outDataForming3{5};
handles.custom.minedge = outDataForming3{6};

handles.custom.wclass = fPerceptron({handles.custom.kol_1viz,
handles.custom.kol_2viz, ...
      handles.custom.osn_w, handles.custom.countw});
set(handles.uitable2, 'Data', handles.custom.wclass);


set(handles.edit3, 'String', handles.custom.interval);
set(handles.edit4, 'String', handles.custom.minedge);

set(handles.pushbutton6, 'Enable', 'on');
set(handles.pushbutton7, 'Enable', 'on');
set(handles.pushbutton9, 'Enable', 'on');
set(handles.pushbutton11, 'Enable', 'on');
```

```
guidata(hObject, handles);
```

Функція Untitled_2_Callback визивається в момент натискання кнопки з завантаженням файлу з сигналом. В тілі функції виконується uigetfile – вбудована в guide функція, яка приймає поширення файлу що потрібно знайти. Завантаження виконується з файлу з поширенням .mat, оскільки людино-машинний інтерфейс розробляється в Matlab, який має багато методів для перетворення, і оскільки структура збереження даних у кожному форматі різна, тому було прийнято рішення використати лише один.

На виході uigetfile надаються два параметри file і path, file – назва файлу с поширенням, path – шлях до цього файлу. Об’єднаєм ці дві змінні для імпортування даних функцією importdata. Дані з файлу завантажені.

```
function Untitled_2_Callback(hObject, eventdata, handles)
   [file, path] = uigetfile('*.mat');
   fullpath = strcat(path, file);
   handles.custom.trspeed = importdata(fullpath);

   guidata(hObject, handles);

   set(handles.pushbutton12, 'Enable', 'on');
```

Функція Untitled_3_Callback виконується при натисканні на кнопку збереження стану. В тілі функціє визивається функція uisave яка приймає тільки строчки, тому необхідно створити додаткову змінну і ініціалізувати її handles.custom. Потім в функцію uisave подаються назви змінної яку потрібно зберегти і назва файла в який буде зберігатись дані.

```
function Untitled_3_Callback(hObject, eventdata, handles)
   custom = handles.custom;
   uisave('custom', 'custom1');
```

Розглянуті функції є складними і потребують роз’яснень, інші розроблені функції є шаблонними або зрозумілими і наведені в додатку А.

Розроблений графічний інтерфейс з функціональною складовою наведено на рис 4.7, на рис. 4.8 зображений інтерфейс з активним графіком диференційованого сигналу.

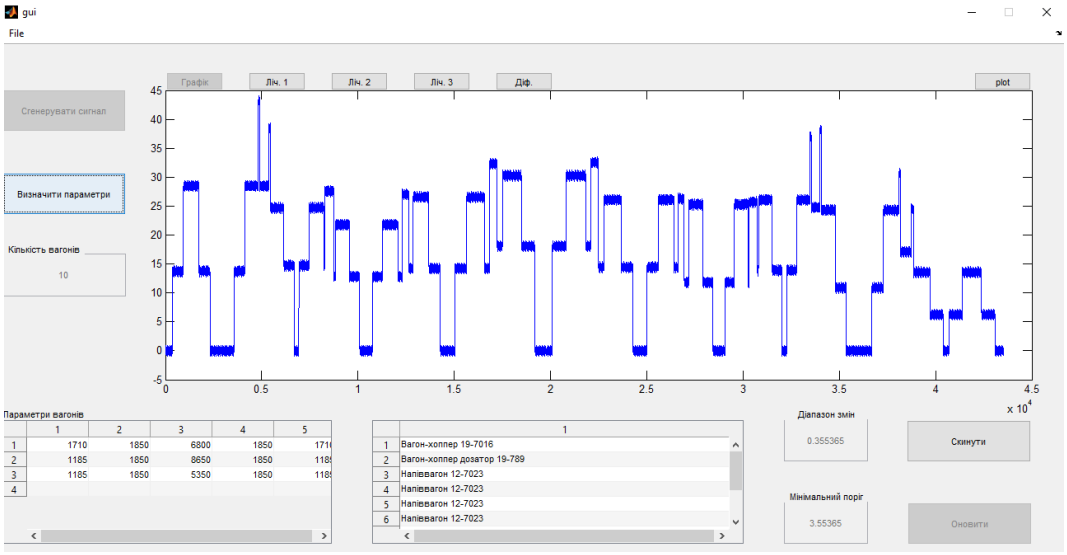

Рисунок 4.7 – Людино-машинний інтерфейс в робочому стані

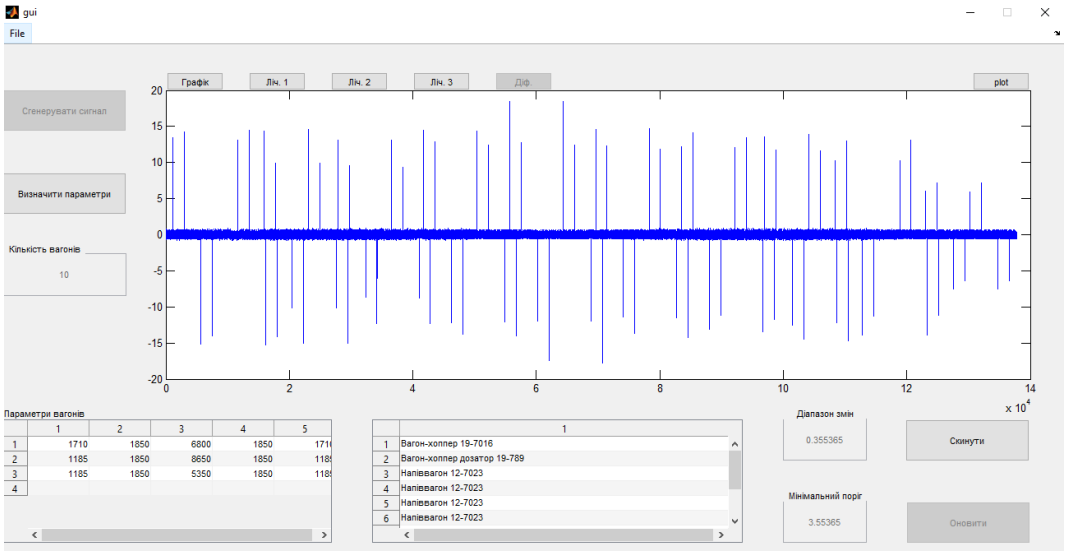

Рисунок 4.8 – Людино-машинний інтерфейс з активним графіком диференційованого сигналу

4.4 Висновки за розділом

У розділі було розроблено людино-машинний інтерфейс з використанням функцій і моделі розроблених у другому та третьому розділі кваліфікаційної роботи. На інтерфейсі є графік з можливостю перемкнути на проміжні параметри (лічильників або диференціального сигналу), можливість

переглянути в окремому вікні графік з додатковими можливостями перегляду, таблиця з бази знань параметрів вагонів, таблиця з вихідними даними класифікації вагонів. Є можливість завантажити з зовнішнього середовища сигнал з платформи, або зберегти всі параметри.

Інтерфейс відповідає потребам оператора для проведення класифікації типів вагонів за сигналом з платформи.

5 ЕКОНОМІКА

5.1 Обгрунтування доцільності автоматизації процесу класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи

Розвиток технічних, апаратних і програмних засобів, дає можливість підприємствам зменшувати витрати, підвищувати якість продукції, збільшувати швидкість виробництва, автоматизувати процеси та приймати безліч інших рішень.

У даному розділі кваліфікаційної роботи виконано економічне обґрунтування доцільності використання системи автоматичної класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи.

Автоматизована система забезпечує:

- зменшення електричних витрат на ідентифікацію типу проїжджаючого вагону по платформі нейронною мережею;

- спостереження за процесом керування за допомогою людино-машинного інтерфейсу;

- збереження отриманих даних.

5.2 Розрахунок капітальних витрат для автоматизації процесу класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи

Розрахуємо капітальні витрати, що пов’язані з виготовленням та впровадженням автоматизованої системи керування процесом класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи. Визначення проектних капітальних витрат виконується відповідно до:

$$К_{ПКВ} = С_{ОБ} + Д_{ТР} + М_{МН}, \quad (5.1)$$

де $К_{пкв}$ – проектні капітальні витрати (грн.);

$С_{об}$ – вартість основного та допоміжного обладнання (грн.);

$Д_{тр}$ – транспортно-заготівельні витрати (грн.);

$М_{мн}$ – витрати на монтаж і налагодження системи (грн.),

Вартість основного та допоміжного обладнання наведена в табл. 5.1.

Таблиця 5.1 – Перелік комплектуючих

№ п/п	Найменування виробів згідно проектних розробок	Кількість	Оптова ціна за од., грн.	Сума, грн.
1	Програмне забезпечення людино-машинного інтерфейсу	1	98780	98780
2	Ваговимірювальна платформа 150ВВСД з датчиками	1	640500	640500
3	Вагопроцесор (термінал)	1	9706	9706
Разом				748986

Витрати на транспортно-заготівельні і складські роботи визначаються в залежності від вартості обладнання, як 6 % від загальної вартості:

$$Д_{ТР} = С_{ОБ} * 0{,}06. \quad (5.2)$$

Витрати на транспортно-заготівельні і складські роботи складають:

$$Д_{ТР} = 748986 * 0{,}06 = 44939{,}16(\text{грн.}).$$

Вартість монтажно-налагоджувальних робіт приймаються на рівні 5 % від вартості обладнання:

$$М_{МН} = С_{ОБ} * 0{,}05. \quad (5.3)$$

Витрати на монтажно-налагоджувальні роботи складають:

$$М_{МН} = 748986 * 0{,}05 = 37449{,}3\ (\text{грн.}).$$

Капітальні витрати на придбання та налагодження обладнання складають:

$$К_{ПКВ} = 748986 + 44939,16 + 37449,3 = 831374,46\ (грн.)$$

Розрахунок експлуатаційних витрат для автоматизації процесу класифікації типів залізничних вагонів на основі аналізу сигналів ваговимірювальної платформи

Річні експлуатаційні витрати розраховуються як:

$$С_е = С_а + С_З + С_с + С_{РО} + С_{ее} + С_{ІНШ}, \quad (5.4)$$

де $С_е$– річні поточні витрати, пов’язані із застосуванням системи керування (грн.);

$С_а$ – амортизація основних фондів (грн.);

$С_З$ – заробітна плата обслуговуючого персоналу (грн.);

$С_с$ – відрахування на соціальні заходи (грн.);

$С_{РО}$ – витрати на технічне обслуговування та поточний ремонт обладнання (грн.);

$С_{ее}$ – вартість електроенергії,

$С_{ІНШ}$ – інші витрати.

Визначимо експлуатаційні витрати при впровадженні системи керування.

Залежно від групи, до якої віднесено той, чи інший об’єкт основних засобів, встановлено мінімально-допустимі строки їх амортизації.

Обладнання, розробленої в кваліфікаційній роботі системи керування, належить до 4 групи (машини та обладнання). Передбачуваний термін експлуатації системи становить 5 років.

При використанні методу прискореного зменшення залишкової вартості норма амортизації визначається як:

$$Н_а = \frac{2}{Т} * 100\ \%, \quad (5.5)$$

де $Н_а$ – норма амортизації (%),

Т – термін корисного використання об'єкта (років).

Амортизація основних фондів визначається як:

$$С_а = \frac{ПВ * Н_а}{100\ \%}, \quad (5.6)$$

де $С_а$ – річна амортизація основних фондів (грн.),

ПВ – первинна вартість ($ПВ = К_{ПКВ}$) (грн.)

Отже, норма амортизації для проектованої системи керування складає:

$$Н_а = \frac{2}{5} * 100\ \% = 40\ \%.$$

Сума амортизації для проектованої системи становить:

$$С_{ап} = 831374{,}46 * 0.4 = 332549.78\ (грн.),$$

де $С_{ап}$ – річна амортизація основних фондів проектної системи (грн.);

Номінальний річний фонд робочого часу одного працівника:

$$Т_{НР} = (Т_К - Т_{ВС} - Т_В) * Т_З, \quad (5.7)$$

де $Т_{НР}$ – номінальний річний фонд робочого часу одного працівника (год.);

$T_К$ – календарний фонд робочого часу ($T_К = 365$ днів),

$T_{ВС}$ – вихідні дні та свята ($T_{ВС} = 114$ (днів)),

$T_В$ – відпустка ($T_В = 21$ день);

$T_З$ – тривалість зміни ($T_З = 8$ год.).

Таким чином, річний фонд робочого часу працівника складе:

$$T_{НР} = (365 - 114 - 21) * 8 = 1840 \text{ (год.)}.$$

У процесі керування задіяний 1 оператор людино-машинного інтерфейсу, 1 технолог та 1 спеціаліст з електроустаткування.

Після впровадження проектованої системи керування штат персоналу не зміниться, отже заробітна плата і відрахування на соціальні заходи будуть однакові.

Розрахунок річного фонду заробітної плати виробничих робітників здійснюється у відповідності з формою, наведеною в табл. 5.2.

Відрахування на соціальні заходи визначаються як:

$$C_с = 0{,}22 * C_З. \quad (5.8)$$

де $C_{ЗП}$ – заробітна плата персоналу проектної системи керування (грн.);

Таблиця 5.2 – Розрахунок заробітної плати персоналу

№п/п	Професія	Число працюючих, чол	Годинна тарифна ставка, грн./год.	Номінальний річний фонд робочого часу	Основна заробітна плата, грн.	Додаткова заробітна плата	Всього заробітна плата, грн.
1	Оператор	1	35	1840	64400,3	8653,2	73053,5
2	Інженер	1	32	1840	58880,1	7426,6	56672,7

Продовження таблиці 5.2 – Розрахунок заробітної плати персоналу

3	Наладчик електроустаткування	1	29	1840	53360,9	6521,2	59882,1
Разом							189608,3

Відповідно до цього відрахування становлять:

$$C_c = 0{,}22 * 189608{,}3 = 41493{,}82 \text{ (грн.)}.$$

Витрати, пов'язані з ремонтом та технічним обслуговуванням нового обладнання, становлять 5 % від вартості капітальних вкладень, тобто:

$$C_{РТО} = 831374{,}46 * 0.05 = 41568{,}72$$

Вартість електроенергії, споживаної системою керування, розробленої у проекті:

$$C_{ее} = K_е * K_{РД} * T_З * T_е, \quad (5.9)$$

де $K_е$ – кількість електроенергії, спожите проектною системою керування (приймаємо $K_е = 0{,}53$ кВт · год.);

$K_{РД}$ – кількість робочих днів у році ($K_{РД} = 251$ день);

$T_е$ – тариф на електроенергію для підприємств (для користувачів електроенергії 2 класу тариф складає 2,26 грн/кВт без ПДВ, з урахуванням ПДВ тариф $T_е = 2{,}712$ грн.).

Таким чином вартість електроенергії становить:

$$C_{ееП} = 0{,}53 \cdot 251 \cdot 8 \cdot 2{,}712 = 2886{,}21 \text{ (грн.)}$$

де $C_{ееП}$ – вартість електроенергії споживаної проектною системою керування (грн.);

Інші витрати з експлуатації об’єкта проектування включають витрати з охорони праці, на спецодяг та інше згідно практиці, ці витрати визначаються в розмірі 4 % від річного фонду заробітної плати обслуговуючого персоналу:

$$С_{ІНШ} = 0{,}04 \cdot С_З. \quad (5.10)$$

Таким чином інші витрати становлять:

$$С_{ІНШП} = 0{,}04 \cdot 189608{,}3 = 7584{,}33 \text{ (грн.)}.$$

де $С_{ІНШП}$ – інші витрати проектної системи керування (грн.);

Річні експлуатаційні витрати становлять (5.4), (табл. 5.1):

$$С_П = 332549.78 + 189608{,}3 + 41493{,}82 + 41568{,}72 + 2886{,}21 + 7584{,}33 = 620691{,}16 \text{ (грн.)}$$

де $С_П$ – річні експлуатаційні витрати проектної системи керування (грн.);

Таблиця 5.1 – Експлуатаційні витрати

№ п/п	Назва покажчика	Проектний варіант, грн.
1	Амортизація	332549,78
2	Фонд заробітної плати	189608,3
3	Відрахування на соціальні виплати	41493,82
4	Ремонт та технічне обслуговування	41568,72
5	Електроенергія	2886,21
6	Інше	7584,33
7	Загалом	620691,16

5.3 Висновки по розділу

При впровадженні проектованої системи витрати на закупку комплектуючих елементів складає 748986 грн., транспортно-заготівельні і складські роботи – 44939,16 грн., монтажно-налагоджувальні роботи – 37449,3 грн., виходячи з цього капітальні витрати складають 831374,46 грн.

Річні експлуатаційні витрати, пов’язані з впровадженням системи розраховуються, як сума з амортизації системи – 332549,78, фонду заробітної плати – 189608,3, відрахування на соціальні виплати – 41493,82, ремонт та технічне обслуговування – 41568,72, витрати на електроенергію – 2886,21, та інші витрати складають – 7854,40. Виходячи з цього, експлуатаційні витрати складають - 620691,16 грн.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ПІДЧАС НАДЗВИЧАЙНИХ СИТУАЦІЙ

6.1 Аналіз сигналів платформи для визначення небезпеки травмування при посадці на потяг

Спробуємо описати процес відправлення потягу зі залізничної станції після завантаження потягу методом функціонального резонансного аналізу (FRAM). Цей метод використовує концепцію шести основних елементів (рис. 6.1) їх взаємодії, щоб отримати уявлення про виробничий процес. Основна увага приділяється виконанню роботи та можливій появі певного резонансу різних подій, що виникає внаслідок мінливості повсякденної діяльності. Опис моделі та програмне забезпечення наведено в табл. 6.1.

Після зупинки потягу на залізничній станції, провідники відкривають двері вагонів і залізниники можуть вийти чи зайти (рис. 6.2). Для відкриття дверей провідник керується сигналами про повну запинку потягу. Після закриття всіх дверей у вагонах потягу в машиніста на панелі управління з’являється відповідний сигнал. Далі машиніст повинен перевірити сигнал світлофору на залізничній колії. Після запиту до диспетчера на панелі управління з’являється сигнал, який дозволяє відправлення. Всі сигнали, які повідомляють машиніста про його можливі дії пов’язані з автоматичною системою управління рухом потягів, яка допомагає організовувати безпечний процес перевезення пасажирів.

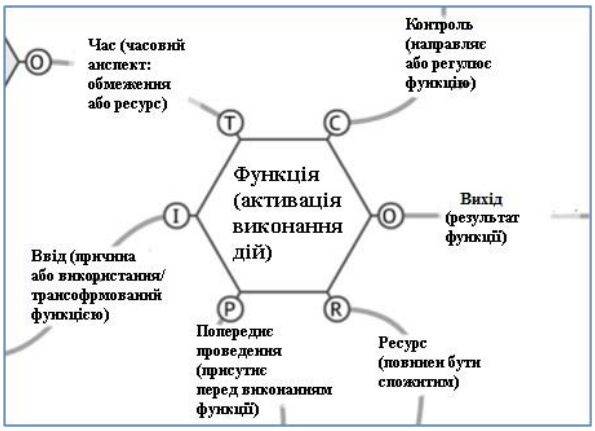

Рисунок 6.1 – Модель FRAM

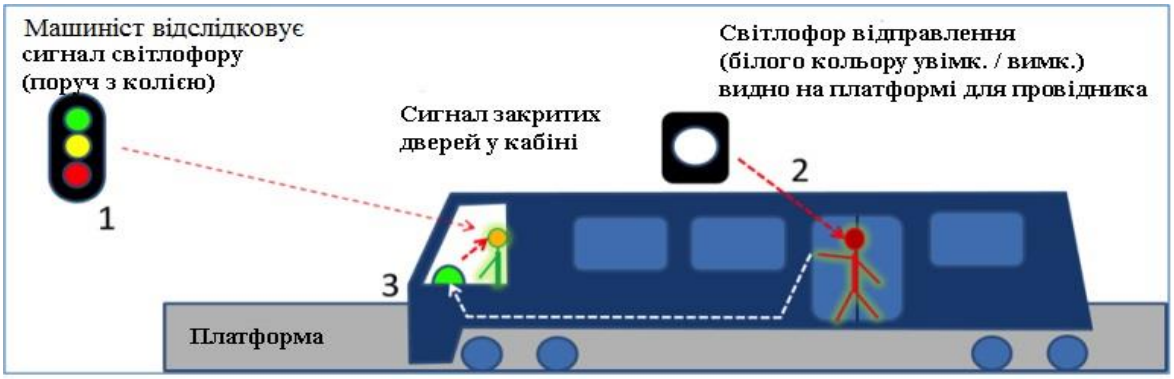

Рисунок 6.2 – Спрощена схема посадки і висадки з потягу

Таблиця 6.1 – Характеристика функціональних елементів FRAM

Параметри та символи		Опис
Input I	Вхідні дані	Вхідна інформація (або вхідні дані), яка характеризує функцію в процесі функціонування системи. Це є посилання на попередні функції. Вхідні дані запускають виконання або дію функції.
Output O	Вихід	Результат виконання функції (вихідні дані). Надає посилання на подальші функції. Представляє результат зміни технічного стану елементів системи (автомобіля).
Time T	Час	Час, необхідний для обробки (підготовки, діагностування, обслуговування елементів системи) функціональним підрозділом або підсистеми
Control C	Контроль	Обмеження, методи та процедури контролю. Вони визначають, як функція передає технічні дані та/оба контролюється
Precon-ditions P	Переду-мови	Системні умови, які визначають умови функціонування та, які повинні бути задоволені перед виконанням функції.
Resour-ces R	Ресурси	Ресурси, які потрібні або використовуються під час обробки функції або функціонування елементів системи (автомобіля).

Опис процесу в постановці задачі надає початкову інформацію для розробки моделі FRAM. На рис. 6.3 наведено першу ітерацію моделі. Програмне забезпечення автоматично створює фонові функції – контекстуальні елементи, які впливають на досліджуваний процес з точки зору мінливості і вважаються доречними для розуміння та аналізу процесу. Елементи різняться за кольором. Білий - це основні функції, інші кольори відображають другорядні додаткові процеси.

Отже, модель показує, що машиніст поїзда може почати рух, коли побачить зелене світло від світлофору з попередньою умовою сигналу закритих дверей та появи сигналу відправлення від диспетчера.

Зверніть увагу, що не всі вузли шестикутників використовуються, а лише ті, що мають значення для аналізу. Це випливає із збору інформації та тонкої настройки моделі з групою зворотного зв'язку. Модель завжди можна розширити додатковими функціями, щоб врахувати більше деталей, які можуть вплинути на кінцевий результат.

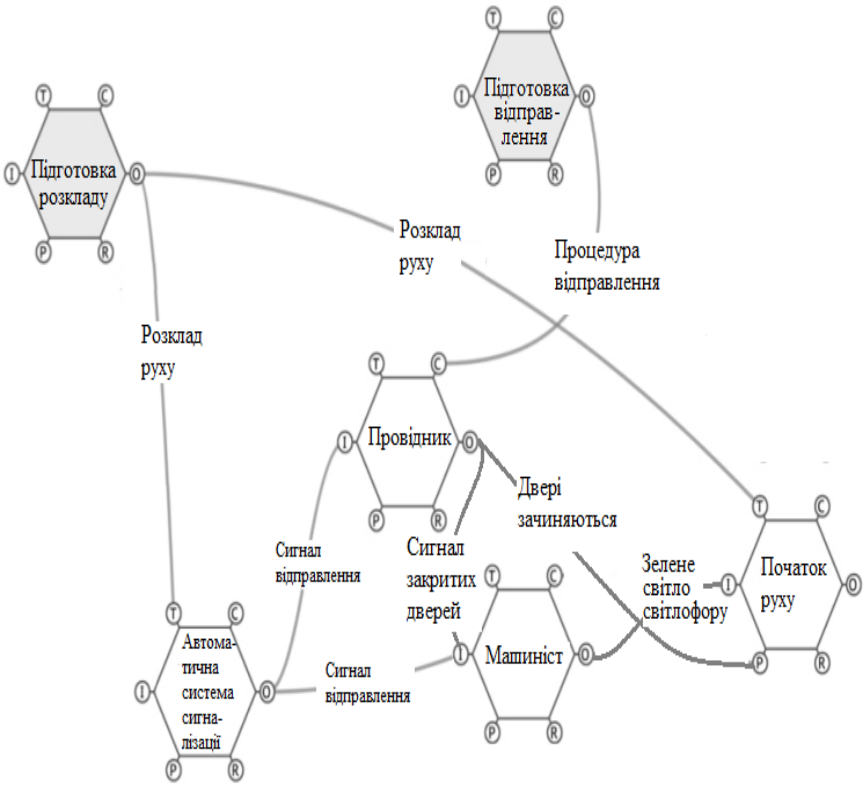

Рисунок 6.3 – Перша ітерація FRAM

Під час попередньої побудови моделі доречно ставити уточнюючі питання:

- Яка система активує сигнал світлофору і на основі чого?
- Яка система активує сигнал дозволу і на основі чого?
- Як аповязані обидва сигнали?
- Як виглядає процедура відправлення для кондуктора, а як для машиніста потягу?

Звісно запитань може бути більше (табл. 6.2), які необхідні для побудови остаточної моделі FRAM.

Отже, в результаті детального аналізу зупинки потяга і посадки пасажирів встановили, що сигнал світлофору активується безпосередньо через автоматичну систему сигналізації. В той же час диспетчер, який контролює рух потягів у режимі реального часу може вносити правки в ручному режимі для відповідного корегування виниклих нештатних проблем чи ситуацій. Це додаткова функція ручного введення в систему сигналізації. Сигнал для відкриття дверей потягу має дещо іншу активацію, ніж передбачалося в першій ітерації. Зокрема система активує сигнал, коли сигнал від світлофору стає зеленим. Майже одночасно провідник отримує «дозвіл», щоб розпочати процедуру посадки пасажирів. Провідник самостійно контролює час відправлення, щоб дотримуватись розкладу (передумова часу відправлення). А далі відбувається процедура закриття дверей, що призводить до появи відповідного сигналу у машиніста. Машиніст після повідомлення, що двері зачинені, запитує дозвіл почати рух, тому з’являється передумова зв’язок з диспетчером, який підтверджує дозвіл. На практиці машиніст контролює тільки сигнал «зачинені двері», рахуючи, що всі інші умови виконуються автоматично.

Таблиця 6.2 – Запитання для уточнення моделі

Вхідні дані	Що запускає функцію?
	На що функція діє чи змінюється?
Вихідні дані	Що таке результат чи результати функції?
	Чи потрібно комусь повідомляти?
	Чи потрібно щось збирати чи записувати / повідомляти? Якщо так, то де?
	Кому потрібна продукція? Хто використовуватиме те, що виробляється? Чи домовлялися ви з тим, хто цим користується, що це те, що їм потрібно?
Передумова	Що має бути на місці, щоб ви могли нормально виконувати цю функцію?
	Що робити, якщо передумови відсутні?
Ресурс	Які ресурси вам потрібні для виконання функції, такі як люди, обладнання, IT, електроенергія, будівлі тощо?
	Що робити, якщо ресурси відсутні?
Контроль	Чи є у вас якісь цілі щодо функції, наприклад, зробити щось за певний час (це контроль)?
	Яка мета цієї функції? Чому ми це робимо?
	Чи є у вас офіційні процедури чи інструкції щодо контролю функції?
	У вас є люди, такі як керівники, які контролюють функцію?
	Чи існують значення, що контролюють функцію?
	Чи контролюють функцію неофіційна практика роботи чи культура?
	Чи є у вас пріоритети, наприклад система сортування?
	Чи існують такі обмеження, як бюджет?
Час	Чи є якийсь час, пов’язаний із функцією?
	Чи є певний час, коли ви повинні виконувати цю функцію?
	Що станеться, якщо вас затримають - ви все одно виконаєте цю функцію чи ні, і яким є наслідок для наступних функцій?
	Час має лише чотири варіанти: занадто рано, занадто пізно, вчасно або взагалі немає.

Різний колір шестикутників допомагає читання моделі. Сірий відображає функції, пов'язані з системою підготовки. Жовтий – дозвільна система. Синій - це провідник, машиніст.

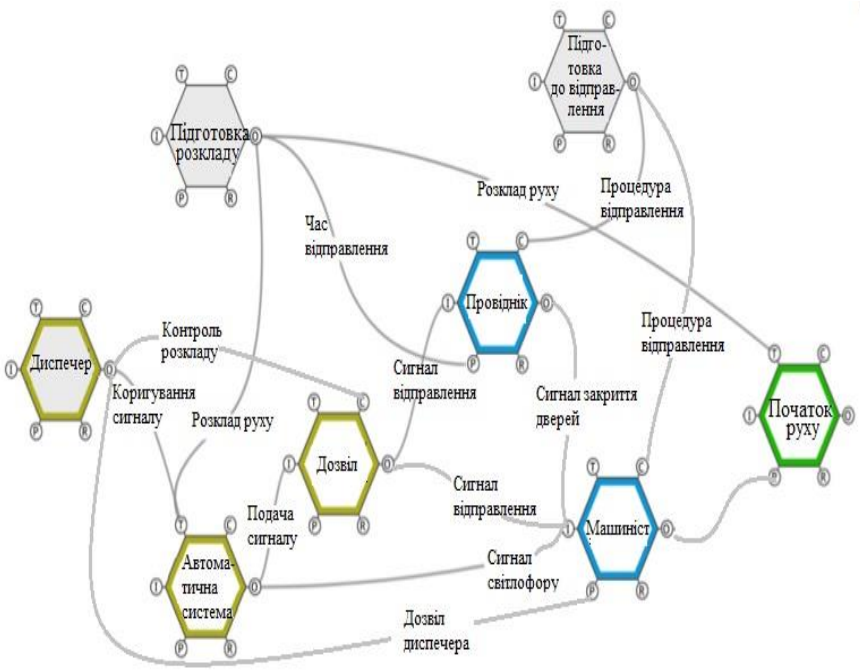

Рисунок 6.4 – Модель FRAM після уточнення

Після внесення змін з появою додаткової інформації та візуалізації всього процесу, з’явились деякі цікаві моменти. Перший, основним сигналом для машиніста є дозвіл диспетчера, який він отримує після повідомлення про закриті двері, тоді як необхідна інформація про зелене світло і сигнал відправлення інформація, який надходить пізніше. В принципі – це не викликає жодних проблем, доти коли виникає необхідність ручного управління диспетчера, коли може виникнути ситуація дублювання сигналів. Друге, виникають певні складнощі постійного контролю часу відправлення провідником.

Тобто у результаті досліджень обраний метод демонструє можливості використання FRAM для виявлення певних протиріч в системах, поєднання чи дублювання сигналів, які можуть призвести до аварійних ситуацій. Також він допомагає знайти шляхи щодо вдосконалення систем управління.

FRAM – це, перш за все, створення загальної моделі роботи, що виконується, під час розслідування. Це безпосередньо відокремлює його від традиційних способів розслідування інцидентів. Розслідування інцидентів спрямоване на конкретну ситуацію, в якій сталася подія. Дослідження FRAM

ставить конкретну ситуацію в контекст нормального виконання процесу. Ця різниця особливо актуальна при спробі вдосконалення. Потім варіанти вдосконалення можна також включити в ціле. FRAM сприяє обговоренню того, як виконується процес та які проблеми виникають щодня. Також можна зібрати багату картину контекстної інформації, яка може бути використана для подальшого роздуму.

ВИСНОВКИ

Об'єктом дослідження є ваговимірювальна платформа для зважування залізничних вагонів. Предметом дослідження – процес класифікації залізничних типів вагонів за сигналами що надходять з ваговимірювальної платформи.

Розглянуті конструктивні параметри вагонів: кількість осей візка, відстань між осями одного візка, відстань між внутрішніми осями сусідніх візків. Розроблену систему верхнього рівня з ваговимірювальної платформи, вагового терміналу і персонального комп’ютера як місця оператора.

Зміна швидкості потяга впливає на зміну перехідного процесу у системі, тому вона додана у модель як збурення.

Розглянуто комплекс питань щодо проведення досліджень.

Розроблена імітаційна модель проїзду потяга з вагонами по ваговимірювальної платформі, з фіксацією зважувань ваги у сховище даних.

Розроблено алгоритм визначення конструктивних параметрів залізничних вагонів з використанням методів перетворення сигналу. При перетворенні сигналу було помічено що значення параметрів лежать у певному діапазону від очікуваної величини. Тому за розробленим алгоритмом з використанням нейронної мережі Перцептрон, було проведено класифікацію типів залізничних вагонів з підвищенням рівня аналізу критеріїв.

Розроблено людино-машинний інтерфейс з використанням функцій і моделі розроблених у другому та третьому розділі кваліфікаційної роботи. На інтерфейсі є графік з можливостю перемкнути на проміжні параметри (лічильників або диференціального сигналу), можливість переглянути в окремому вікні графік з додатковими можливостями перегляду, таблиця з бази знань параметрів вагонів, таблиця з вихідними даними класифікації вагонів. Є можливість завантажити з зовнішнього середовища сигнал з платформи, або зберегти всі параметри.

Інтерфейс відповідає потребам оператора для проведення класифікації типів вагонів за сигналом з платформи.

При впровадженні проектованої системи витрати на закупку комплектуючих елементів складає 748986 грн., транспортно-заготівельні і складські роботи – 44939,16 грн., монтажно-налагоджувальні роботи – 37449,3 грн., виходячи з цього капітальні витрати складають 831374,46 грн.

Річні експлуатаційні витрати, пов’язані з впровадженням системи розраховуються, як сума з амортизації системи – 332549,78, фонду заробітної плати – 189608,3, відрахування на соціальні виплати – 41493,82, ремонт та технічне обслуговування – 41568,72, витрати на електроенергію – 2886,21, та інші витрати складають – 7854,40. Виходячи з цього, експлуатаційні витрати складають - 620691,16 грн.

Також були розглянуті питання щодо заходів з охорони праці

ДОДАТОК А

Програмний код функціоналу людино-машинного інтерфейсу

```
% функція ініціалізації інтерфейсу
function varargout = gui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
            gui_mainfcn(gui_State, varargin{:});

end

% функція яка спрацьовує при відкритті інтерфейсу
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);

function varargout = gui_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% функція що спрацьовує при створенні елементу edit1
function edit1_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', '10');

   if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
   end

% функція що спрацьовує при створенні елементу uitable1
function uitable1_CreateFcn(hObject, eventdata, handles)
```

```
    table = get(hObject, 'Data');
    table(1,1) = {1710};
    table(1,2) = {1850};
    table(1,3) = {6800};
    table(1,4) = {1850};
    table(1,5) = {1710};

    table(2,1) = {1185};
    table(2,2) = {1850};
    table(2,3) = {8650};
    table(2,4) = {1850};
    table(2,5) = {1185};

    table(3,1) = {1185};
    table(3,2) = {1850};
    table(3,3) = {5350};
    table(3,4) = {1850};
    table(3,5) = {1185};

    set(hObject, 'Data', table);

% функція що опрацьовує код в тілі після натискання на pushbutton1
function pushbutton1_Callback(hObject, eventdata, handles)
    handles.custom.countw = str2num(get(handles.edit1, 'String'));
    handles.custom.fgen = fGeneratorSignal(handles.custom.countw);
    outfTrainSpeed2 = fTrainSpeed2(handles.custom.fgen);
    handles.custom.trspeed = cell2mat(outfTrainSpeed2(1));
    handles.custom.speed = cell2mat(outfTrainSpeed2(2));

    axes(handles.axes1);
    cla;
    plot(handles.custom.trspeed);

    guidata(hObject, handles);

    set(handles.pushbutton1, 'Enable', 'off');
    set(handles.pushbutton8, 'Enable', 'on');
    set(handles.pushbutton10, 'Enable', 'on');
    set(handles.pushbutton3, 'Enable', 'on');
    set(handles.edit1, 'Enable', 'off');
```

```
% функція що спрацьовує при створенні елементу pushbutton1
function pushbutton1_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'Згенерувати сигнал');

% функція що спрацьовує при натисканні на елемент pushbutton3
function pushbutton3_Callback(hObject, eventdata, handles)
   figure(1);
   if strcmp(get(handles.pushbutton5, 'Enable'), 'off')
      plot(handles.custom.trspeed);
   elseif strcmp(get(handles.pushbutton6, 'Enable'), 'off')
      plot(handles.custom.counter1);
   elseif strcmp(get(handles.pushbutton7, 'Enable'), 'off')
      plot(handles.custom.counter2);
   elseif strcmp(get(handles.pushbutton9, 'Enable'), 'off')
      plot(handles.custom.counter3);
   elseif strcmp(get(handles.pushbutton11, 'Enable'), 'off')
      plot(handles.custom.deriv_vaga);
   end

 % функція що спрацьовує при створенні елементу pushbutton1
function pushbutton3_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'plot');
   set(hObject, 'Enable', 'off');

% функція що спрацьовує при натисканні на елемент pushbutton4
function pushbutton4_Callback(hObject, eventdata, handles)
   figure(2);
   plot(handles.custom.fgen)

% функція що спрацьовує при створенні елементу pushbutton4
   function pushbutton4_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'plot');

% функція що спрацьовує при натисканні на елемент pushbutton4
function pushbutton5_Callback(hObject, eventdata, handles)
   set(handles.pushbutton5, 'Enable', 'off');
   set(handles.pushbutton6, 'Enable', 'on');
   set(handles.pushbutton7, 'Enable', 'on');
   set(handles.pushbutton9, 'Enable', 'on');
   set(handles.pushbutton11, 'Enable', 'on');
   axes(handles.axes1);
   cla;
   plot(handles.custom.trspeed);
```

```
% функція що спрацьовує при натисканні на елемент pushbutton6
function pushbutton6_Callback(hObject, eventdata, handles)
    set(handles.pushbutton6, 'Enable', 'off');
    set(handles.pushbutton5, 'Enable', 'on');
    set(handles.pushbutton7, 'Enable', 'on');
    set(handles.pushbutton9, 'Enable', 'on');
    set(handles.pushbutton11, 'Enable', 'on');


    axes(handles.axes1);
    cla;
    plot(handles.custom.counter1);

% функція що спрацьовує при натисканні на елемент pushbutton7
function pushbutton7_Callback(hObject, eventdata, handles)
    set(handles.pushbutton5, 'Enable', 'on');
    set(handles.pushbutton6, 'Enable', 'on');
    set(handles.pushbutton7, 'Enable', 'off');
    set(handles.pushbutton9, 'Enable', 'on');
    set(handles.pushbutton11, 'Enable', 'on');
    axes(handles.axes1);
    cla;
    plot(handles.custom.counter2);

% функція що спрацьовує при натисканні на елемент pushbutton8
function pushbutton8_Callback(hObject, eventdata, handles)
    Vaga_W = handles.custom.trspeed;
    Speed = handles.custom.speed;
    count_wagons = handles.custom.countw;
    outDataForming3 = fDataForming3(Vaga_W, Speed, count_wagons);

    handles.custom.deriv_vaga = outDataForming3{1};
    handles.custom.osn_w = outDataForming3{2};
    disp(handles.custom.osn_w);
    handles.custom.kol_1viz = outDataForming3{3};
    disp(handles.custom.kol_1viz);
    handles.custom.kol_2viz = outDataForming3{4};
    disp(handles.custom.kol_2viz);
    handles.custom.counter1 = outDataForming3{7};
    handles.custom.counter2 = outDataForming3{8};
    handles.custom.counter3 = outDataForming3{9};
    handles.custom.interval = outDataForming3{5};
    handles.custom.minedge = outDataForming3{6};
```

```
    handles.custom.wclass = fPerceptron({handles.custom.kol_1viz,
handles.custom.kol_2viz, ...
    handles.custom.osn_w, handles.custom.countw});
    set(handles.uitable2, 'Data', handles.custom.wclass);

    set(handles.edit3, 'String', outDataForming3{5});
    set(handles.edit4, 'String', outDataForming3{6});

    set(handles.pushbutton6, 'Enable', 'on');
    set(handles.pushbutton7, 'Enable', 'on');
    set(handles.pushbutton9, 'Enable', 'on');
    set(handles.pushbutton11, 'Enable', 'on');

    guidata(hObject, handles);

% функція що спрацьовує при створенні елементу pushbutton8
function pushbutton8_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'String', 'Визначити параметри');
    set(hObject, 'Enable', 'off');

% функція що спрацьовує при створенні елементу pushbutton5
function pushbutton5_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Enable', 'off');
    set(hObject, 'String', 'Графік');

% функція що спрацьовує при створенні елементу pushbutton6
function pushbutton6_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Enable', 'off');
    set(hObject, 'String', 'Ліч. 1');

% функція що спрацьовує при створенні елементу pushbutton7
function pushbutton7_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Enable', 'off');
    set(hObject, 'String', 'Ліч. 2');

% функція що спрацьовує при натисканні на елемент pushbutton9
function pushbutton9_Callback(hObject, eventdata, handles)
    set(handles.pushbutton5, 'Enable', 'on');
    set(handles.pushbutton6, 'Enable', 'on');
    set(handles.pushbutton7, 'Enable', 'on');
    set(handles.pushbutton9, 'Enable', 'off');
    set(handles.pushbutton11, 'Enable', 'on');
    axes(handles.axes1);
    cla;
```

```
    plot(handles.custom.counter3);

% функція що спрацьовує при створенні елементу pushbutton9
function pushbutton9_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Enable', 'off');
    set(hObject, 'String', 'Ліч. 3');

% функція що спрацьовує при створенні елементу text1
function text1_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'String', 'Кількість вагонів')

% функція що спрацьовує при створенні елементу text1
function text2_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'String', 'Параметри вагонів')

% функція що спрацьовує при натисканні на елемент pushbutton10
function pushbutton10_Callback(hObject, eventdata, handles)
    axes(handles.axes1);
    cla;
    set(handles.pushbutton1, 'Enable', 'on');
    set(handles.pushbutton8, 'Enable', 'off');
    set(handles.edit3, 'String', '0');
    set(handles.edit4, 'String', '0');

    set(handles.pushbutton3, 'Enable', 'off');
    set(handles.pushbutton5, 'Enable', 'off');
    set(handles.pushbutton6, 'Enable', 'off');
    set(handles.pushbutton7, 'Enable', 'off');
    set(handles.pushbutton9, 'Enable', 'off');
    set(handles.pushbutton11, 'Enable', 'off');
    handles.custom = 0;
    guidata(hObject, handles);

% функція що спрацьовує при створенні елементу pushbutton10
function pushbutton10_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'String', 'Скинути');
    set(hObject, 'Enable', 'off');

% функція що спрацьовує при натисканні на елемент edit3
function edit3_Callback(hObject, eventdata, handles)
        %empty
```

```
% функція що спрацьовує при створенні елементу pushbutton9
function edit3_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', '0');
   set(hObject, 'Enable', 'off');

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
        %empty

function edit4_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', '0');
   set(hObject, 'Enable', 'off');

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function text3_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'Діапазон змін');

function text4_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'Мінімальний поріг');

function pushbutton11_Callback(hObject, eventdata, handles)
   set(handles.pushbutton5, 'Enable', 'on');
   set(handles.pushbutton6, 'Enable', 'on');
   set(handles.pushbutton7, 'Enable', 'on');
   set(handles.pushbutton9, 'Enable', 'on');
   set(handles.pushbutton11, 'Enable', 'off');
   axes(handles.axes1);
   cla;
   plot(handles.custom.deriv_vaga);

function pushbutton11_CreateFcn(hObject, eventdata, handles)
   set(hObject, 'String', 'Діф.');
   set(hObject, 'Enable', 'off');

function pushbutton8_ButtonDownFcn(hObject, eventdata, handles)
   set(hObject, 'Enable', 'off');
```

```
function Untitled_1_Callback(hObject, eventdata, handles)
    %empty

function Untitled_2_Callback(hObject, eventdata, handles)
    [file, path] = uigetfile('*.mat');
    fullpath = strcat(path, file);
    handles.custom.trspeed = importdata(fullpath);

    guidata(hObject, handles);

    set(handles.pushbutton12, 'Enable', 'on');

 function Untitled_1_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Label', 'File');

function Untitled_2_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Label', 'Open');

function pushbutton12_Callback(hObject, eventdata, handles)
    outTrainSpeed2 = fTrainSpeed2(handles.custom.trspeed);
    handles.custom.trspeed = outTrainSpeed2{1};
    handles.custom.speed = outTrainSpeed2{2};
    handles.custom.countw = str2num(get(handles.edit1, 'String'));
    guidata(hObject, handles);
    axes(handles.axes1);
    cla;
    plot(handles.custom.trspeed);
    set(hObject, 'Enable', 'off');
    set(handles.edit1, 'Enable', 'on');
    set(handles.pushbutton5, 'Enable', 'off');
    set(handles.pushbutton6, 'Enable', 'off');
    set(handles.pushbutton7, 'Enable', 'off');
    set(handles.pushbutton9, 'Enable', 'off');
    set(handles.pushbutton11, 'Enable', 'off');
    set(handles.pushbutton1, 'Enable', 'off');
    set(handles.pushbutton8, 'Enable', 'on');

function pushbutton12_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Enable', 'off')
    set(hObject, 'String', 'Оновити');

%функція зберігання при натисканні
function Untitled_3_Callback(hObject, eventdata, handles)
```

```
    custom = handles.custom;
    uisave('custom', 'custom1');

% функція зберігання при створенні
function Untitled_3_CreateFcn(hObject, eventdata, handles)
    set(hObject, 'Label', 'Save');
```

Посилання

1. Артемьев І. С. Автоматизация процессов идентификации железнодорожных подвижных единиц на основе гибридных нейроимунных моделей: автореф. дис. канд. техн. наук: 05.13.06. - Ростов-на-Дону, 2017. - 18 с.

2. Укрзалізниця // Вікіпедія URL: https://uk.wikipedia.org/wiki/Укрзалізниця#Рухомий_склад

3. Загальна інформація // УЗ URL: https://www.uz.gov.ua/about/general_information/

4. Разработка алгоритма идентификации подвижных единиц / Бережной А. С. ; Восточноукраинский Национальный Университет им. В. Даля

5. Розенблатт, Ф. Принципы нейродинамики: Перцептроны и теория механизмов мозга. - Мир, 1965

6. Д.С. Глеске Моделирование преобразователей информационно-измерительной системы взвешивания железнодорожных вагонов: Автоматизация и управление технологическими процессами и производствами наук: 01.03.02. - Тольятти, 2017. - 69 с.

6. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины, и определения. − М.: Госстандарт, 1992. – 54 с.

7. ГОСТ 19.201-78. ЕСПД. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. − М.: Госстандарт, 1982. – 128 с.

8. ГОСТ 19.202-78. ЕСПД. Единая система программной документации. Спецификация. Требования к содержанию и оформлению. - М.: Госстандарт, 1982. – 128 с.

9. ГОСТ 19.401-78. ЕСПД. Единая система программной документации. Текст программы. Требования к содержанию и оформлению. − М.: Госстандарт, 88 1982. – 128 с.

10. ГОСТ 19.402-78. ЕСПД. Единая система программной документации. Описание программы. - М.: Госстандарт, 1982. – 128 с.

11. ГОСТ 19.404-79. ЕСПД Единая система программной документации. Пояснительная записка. Требования к содержанию и оформлению. − М.: Госстандарт, 1982. – 128 с.

12. ГОСТ 19.701-90. ЕСПД. Единая система программной документации. Схема алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. − М.: Госстандарт, 1990. – 128 с

UNICHECK

Ім'я користувача:
Олег Бойко

ID перевірки:
1013292112

Дата перевірки:
13.12.2022 18:02:25 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.12.2022 18:03:10 EET

ID користувача:
100008838

Назва документа: **01_151м-21-1_Гончаров В К _ПЗС**

Кількість сторінок: **70** Кількість слів: **11315** Кількість символів: **80952** Розмір файлу: **1.52 MB** ID файлу: **1013050524**

9.91%
Схожість

Найбільша схожість: **6.06%** з джерелом з Бібліотеки (ID файлу: **1013040401**)

3.18% Джерела з Інтернету	109	Сторінка 72
7.73% Джерела з Бібліотеки	143	Сторінка 73

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	30