Міністерство освіти і науки України
Державний ВНЗ «Національний гірничий університет»

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

напрям підготовки *121 Інженерія програмного забезпечення*
(код і назва напряму підготовки)

спеціальність *Програмне забезпечення систем*
(код і назва спеціальності)

освітній рівень *магістр*
(назва освітнього рівня)

кваліфікація *інженер-програміст*
(назва кваліфікації)

на тему: *Обґрунтування програмного забезпечення структурно-параметричної ідентифікації для АСУ процесом крупного дроблення руд*

Виконавець:
студент *6* **курсу , групи** *121м-16-1*

(підпис) *Стадніченко І.О.* (прізвище та ініціали)

Керівники	Посада, прізвище, ініціали	Оцінка	Підпис
Проекту	*д.т.н. проф. Корнієнко В.І.*		
розділів:			
Спеціальний	*д.т.н. проф. Корнієнко В.І.*		
Економічний	*доц. Касьяненко Л.В.*		

Рецензент			

Нормоконтроль	*к.т.н. доц. Коротенко Л.М.*		

Дніпро
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

ЗАТВЕРДЖЕНО:
Завідувач кафедри

програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 ____ року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Програмна інженерія*
(код і назва спеціальності)

Студенту *121м-16-1* *Стадніченку Ігору Олеговичу*
(група) (прізвище та ініціали)

Тема дипломної роботи *Обґрунтування програмного забезпечення структурно-параметричної ідентифікації для АСУ процесом крупного дроблення руд*

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора Державного ВНЗ «НГУ» *№ 2127-л від 26.12.2017*

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – автоматизація керування процесами крупного дроблення руд.

Предмет досліджень – структурно-параметрична ідентифікація процесів крупного дроблення руд.

Мета НДР – обґрунтування методики структурно-параметричної ідентифікації процесів крупного дроблення руд, що забезпечує підвищення точності їх моделей.

Вихідні дані для проведення роботи – існуючі методи ідентифікації процесів крупного дроблення руд; основні параметри ідентифікації процесів крупного дроблення руд.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна результатів, що очікуються, полягає у:

- застосуванні композиційного методу структурно-параметричної ідентифікації складних об'єктів керування до визначення структури моделі процесу крупного дроблення руд;
- підвищенні точності визначення структури, оцінювання та оптимізації параметрів моделі складних об'єктів керування за рахунок застосування методів структурно-параметричної ідентифікації.

Практична цінність результатів полягає у розробленні програмного забезпечення для здійснення структурно-параметричної ідентифікації процесів крупного дроблення руд.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати повинні відповідати вимогам Закону України «Про національну програму інформатизації», Закону України «Про інноваційну діяльність», «Про затвердження Державної цільової науково-технічної програми використання в органах державної влади програмного забезпечення з відкритим кодом на 2012 - 2015 роки», що затверджено наказом Міністерства освіти України від 30 листопада 2011 р. №1269, ГОСТу 19.404-79, ГОСТ 19.402-78.

Результати досліджень мають бути подано у вигляді, що дозволяє безпосереднє використання та впровадження розробленої системи на практиці, її застосування у процесі керування технологічними об'єктами.

5. ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
Дослідження методів ідентифікації нелінійних динамічних процесів.	15.03.2017 - 01.05.2017
Обгрунтування методики структурно-параметричної ідентифікації процесів крупного дроблення руд.	02.05.2017 – 16.06.2017
Розробка програмного забезпечення на базі обраної методики структурно-параметричної ідентифікації процесів крупного дроблення руд.	17.06.2017 – 30.11.2017
Виконання розрахунку трудомісткості розробки програмного забезпечення, витрат на його створення й тривалості розробки.	01.12.2017 – 15.12.2017

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки підвищенню точності ідентифікації процесів дроблення крупних руд, що сприяє зменшенню експлуатаційних затрат при функціонуванні для АСУ складних об'єктів керування.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки спрощенню роботи персоналу, що забезпечує управління АСУ, і зниженню кількості прийнятих помилкових рішень.

7. ДОДАТКОВІ ВИМОГИ

Відповідність оформлення ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення, ЕДИНОЙ СИСТЕМЕ ПРОГРАММНОЙ ДОКУМЕНТАЦИИ (ЕСПД – ГОСТ 19.101-77, ГОСТ 19.102-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.201-78, ГОСТ 19.202-78, ГОСТ 19.401-78, ГОСТ 19.402-78, ГОСТ 19.404-79).

Завдання видав _____ *Корнієнко В.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Стадніченко І.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *15.03.2017р.*

Термін подання дипломного проекту до ДЕК *15.12.2017 р.*

Реферат

Пояснювальна записка: 123 с., 19 рис., 2 табл., 4 додатки, 69 джерел.

Об'єкт дослідження: автоматизація керування процесами крупного дроблення руд.

Мета магістерської роботи: обґрунтування методики структурно-параметричної ідентифікації процесів крупного дроблення руд, що забезпечує підвищення точності їх моделей.

Методи дослідження. При вирішенні поставлених завдань виконано аналіз і наукове узагальнення літературних джерел по вихідним посиланням досліджень для ідентифікації процесів крупного дроблення руд. Використовувався композиційний метод структурно-параметричної ідентифікації.

Наукова новизна отриманих результатів полягає в застосуванні композиційного методу структурно-параметричної ідентифікації складних об'єктів управління до визначення структури і параметрів процесів крупного дроблення руд. А також у підвищенні точності визначення структури, оцінювання та оптимізації параметрів моделі складних об'єктів керування.

Практичне значення роботи полягає в розробці програмного забезпечення для структурно-параметричної ідентифікації процесів крупного дроблення руд.

Галузь застосування. Розроблена методика може використовуватися в організаціях, що експлуатують складні об'єкти управління процесами рудопідготовки (дроблення і подрібнення).

Економічний ефект від впровадження результатів роботи очікується позитивним за рахунок підвищення ефективності ідентифікації процесів крупного дроблення руд, що сприяє зменшенню експлуатаційних витрат при функціонуванні складних об'єктів управління.

Значення роботи і висновки. Методика, що була розроблена, дозволяє підвищити ефективність ідентифікації процесів дроблення крупних руд за рахунок застосування методів структурно-параметричної ідентифікації.

Прогнози щодо розвитку досліджень. Дослідження процесів крупного дроблення руд на основі методів структурно-параметричної ідентифікації дозволить забезпечити визначення більш раціональних умов експлуатації, а також придбати нові можливості оптимізації параметрів складних об'єктів управління.

Список ключових слів: ПРОЦЕСИ КРУПНОГО ДРОБЛЕННЯ РУД, СТРУКТУРНО-ПАРАМЕТРИЧНА ІДЕНТИФІКАЦІЯ, СКЛАДНІ ОБ'ЄКТИ УПРАВЛІННЯ, ПАРАМЕТРИ, ОПТИМІЗАЦІЯ.

The abstract

Explanatory slip: 123 p., 19 fig., 2 tables, 4 application, 69 sources.

Object of research: automation of large ore crushing processes management.

The purpose of the degree project: substantiation the use of structural-parametric method to identify large ore crushing processes, provides an increase in the accuracy of their models.

Methods of research. At solution of tasks in view the analysis and scientific generalization of references on initial sending of researches is made, to identify large ore crushing processes. Composite method of complex control objects structural-parametric identification was used.

The scientific novelty of the results consists in the method of the composite structural and parametric identification complicated control facilities to determine the structure and parameters of processes coarse crushing ores. Also in increasing the accuracy of determining the structure, estimating and optimizing the parameters of the model of complex control objects.

The practical value of works consists in developing the software to implement the structural-parametric identification of large ore crushing processes.

Field of application. The developed method can be used in organizations that operate complex objects for controlling ore preparation processes (crushing and grinding).

The economic benefit of operation outcomes implantation is expected at the expense of increasing the efficiency of large ore crushing processes identification, which helps reduce operating costs in the operation of complex control objects.

Value of work and summary. The developed technique allows to increase the efficiency of large ore crushing processes identification through the use of structural-parametric identification methods.

The prognoses on development of researches. Research of large ore crushing processes on structural-parametric identification methods allow for the definition of a rational operating conditions, as well as opportunities to optimize the complex control objects parameters.

List of key words: LARGE ORE CRUSHING PROCESSES, STRUCTURAL-PARAMETRIC IDENTIFICATION, COMPLEX CONTROL OBJECTS, PARAMETERS, OPTIMIZATION.

ЗМІСТ

Список скорочень

АЗ – алгоритмічне забезпечення

АСУ – автоматична система управління

ГА – генетичний алгоритм

ВД – вихідні дані

ІЗ – інформаційне забезпечення

ЛДБ – лінійний динамічний блок

МГОА – метод групового обчислення аргументів

НМ – нейронна мережа

НСБ – нелінійний статичний блок

НМПП – нейронна мережа прямого поширення

ОУ – об'єкт управління

ПК – персональний комп'ютер

ПЗ – програмне забезпечення

ПВП – прямий випадковий пошук

ПЕОМ – персональна електроннно-обчислювальна машина

РБФ – радіальна базисна функція

СК – спіральний класифікатор

ЕА – еволюційний алгоритм

Anfis – адаптивна нейронна система нечіткого виведення

ARX – авторегресія с додатковим вхідним сигналом

ВСТУП

Актуальність роботи. Складні динамічні об'єкти управління (ОУ) мають нестаціонарні параметри, нелінійні залежності і стохастичні змінні, що обумовлює наявність у них різних динамічних режимів функціонування. До таких складних ОУ відносяться технологічні процеси рудопідготовки (дроблення і подрібнення). Пріоритетом наукових досліджень даних напрямків є розробка алгоритмів і створення автоматичних систем управління (АСУ) технологічними процесами і роботою устаткування. Для реалізації подібних систем необхідна розробка динамічних моделей керованих процесів, яка здійснюється за допомогою процедури їх ідентифікації.

Для управління складними ОУ використовується апріорна інформація у вигляді математичної моделі ОУ, як на стадії проектування, так і в процесі функціонування. Тому для оптимізації функціонування складних динамічних об'єктів актуальним є вирішення завдань ідентифікації, що дозволяє підвищити якість управління складними ОУ за рахунок підвищення точності оцінки їх стану та математичної моделі.

Зв'язок роботи з державними програмами, планами науково-дослідних робіт.

Робота виконана відповідно до закону України № 2623-14 від 11.07.2001 «Про пріоритетні напрями розвитку науки і техніки», «Державної комплексної програми розвитку України», затвердженої Постановою Кабінету Міністрів України, законом України № 3715-VI від 8.09.2011 « Про пріоритетні напрями інноваційної діяльності в Україні ».

Метою роботи та завданнями дослідження є:

- обґрунтування методики структурно-параметричної ідентифікації процесів крупного дроблення руд;
- підвищення точності оцінки стану складних об'єктів управління і їх математичної моделі;

- розробка програмного забезпечення, що реалізує запропоновану методику структурно-параметричної ідентифікації.

Об'єкт дослідження – автоматизація керування процесами крупного дроблення руд.

Предмет дослідження – структурно-параметрична ідентифікація процесів крупного дроблення руд.

Ідея роботи полягає в застосуванні методів структурно-параметричної ідентифікації в розробці динамічних моделей процесів функціонування складних об'єктів управління.

Методи дослідження. При вирішенні поставлених завдань виконано аналіз і наукове узагальнення літературних джерел по вихідним посиланням досліджень для ідентифікації процесів крупного дроблення руд. Використовувався композиційний метод структурно-параметричної ідентифікації.

Наукові положення, очікувані наукові результати.

Застосування методів структурно-параметричної ідентифікації нелінійних динамічних процесів дозволить підвищити точність визначення структури, оцінювання та оптимізації параметрів моделі процесу крупного дроблення руд. Збереження інформації в цифровому вигляді забезпечить можливість зручного її перегляду та об'єктивної оцінки на заданому інтервалі часу.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій роботи обґрунтована коректністю поставлених проблем і прийнятих рішень; обґрунтованістю вихідних посилок, що випливають з основ теорії нейронних мереж, достатнім обсягом і результатами експериментальних досліджень.

Наукова новизна отриманих результатів

1. Підвищення точності визначення структури, оцінки та оптимізації параметрів моделі процесу крупного дроблення руд за рахунок застосування методів структурно-параметричної ідентифікації.

Практичне значення отриманих результатів:

1. Розроблено програмне забезпечення, яке виконує структурно-параметричну ідентифікацію процесу крупного дроблення руд;

2. Показана можливість використання розробленого програмного забезпечення для задач оперативного контролю стану технологічних об'єктів в задачах інформаційного забезпечення технологічних процесів.

Особливий внесок здобувача. Автор самостійно сформулював мету, завдання дослідження, наукові положення і результати, виконав теоретичну і практичну частини роботи.

Апробація результатів магістерської роботи. Основні положення і результати були докладені та обговорені на студентській науковій конференції.

Структура і обсяг роботи. Робота складається з вступу, чотирьох розділів і висновків. Містить 123 сторінки друкованого тексту, в тому числі 77 сторінки тексту основної частини з 19 рисунками, списку використаних джерел з 69 найменувань на 7 сторінках, 4 додатки на 31 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ІДЕНТИФІКАЦІЇ ДИНАМІЧНИХ ПРОЦЕСІВ

1.1. Поняття об'єкта управління

Ключовим моментом теорії є створення математичної моделі, яка описує поведінку об'єкта управління в залежності від його стану, що управляють, і можливих збуджень (перешкод). Формальна математична близькість математичних моделей, що відносяться до об'єктів різної фізичної природи, дозволяє використовувати математичну теорію управління поза її зв'язками з конкретними реалізаціями, а також класифікувати системи управління за формальними математичним ознаками (наприклад, лінійні і нелінійні). У кожній технічній системі існує функціональна частина – об'єкт управління. Функції ОУ технічної системи полягають в сприйнятті керуючих впливів і зміні відповідно до них свого технічного стану. ОУ технічної системи не виконує функцій прийняття рішень, тобто не формує і не вибирає альтернативи своєї поведінки, а тільки реагує на зовнішні (керуючі або збуджуючі) впливи, змінюючи свої стани визначеним конструкцією чином.

1.2. Динамічні процеси в об'єктах управління

Ефективність управління визначається наявністю відповідних засобів, вірним об'єктам управління. Тому для складних ОУ, які характеризуються нестаціонарністю, стохастичністю і нелінійністю (включаючи хаотичну динаміку і фрактальну розмірність), актуальною є розробка засобів для їх ідентифікації і прогнозування. До складних ОУ можна віднести процеси рудопідготовки (дроблення і подрібнення).

При побудові систем оптимального управління необхідна інформація про динамічні характеристики об'єктів управління. Така інформація є набором відомостей, що дозволяють описати в явному вигляді динаміку об'єкта регулювання за допомогою математичної моделі. Якщо

характеристики об'єкта регулювання не змінюються, то можна раз і назавжди побудувати математичну модель або оптимальний регулятор. Якщо ж динамічні характеристики системи змінюються в часі, то побудова математичної моделі і відповідно оптимального регулятора здійснюється в процесі регулювання.

Побудова математичної моделі об'єкта регулювання називається ідентифікацією об'єкта регулювання незалежно від того, чи досліджуються структура і значення коефіцієнтів чи оцінюються параметри системи із заданою або обраною структурою.

Були запропоновані методики визначення властивостей і характеристик динамічних систем, що породжують відповідні сигнали (тимчасові реалізації). При цьому апарат нелінійної динаміки дозволяє оцінити структурні характеристики динамічної моделі нелінійної системи [1].

Нелінійний процес (динамічна система) може бути описаний за допомогою векторного рівняння потоку [2]:

$$\dot{x} = F(x, \rho) \tag{1}$$

або дискретного відображення Пуанкаре:

$$x[k+1] = F\{x[k], \rho\}; \; x[k] = \{x_1[k], ..., x_{d-1}[k]\}, \tag{2}$$

де F – нелінійна вектор-функція розмірності d;

x – вектор координат;

ρ – вектор параметрів порядку системи;

k – такт часу ($t = k \cdot T$);

T – період дискретизації.

Динамічні системи (1) і (2) мають в залежності від значень параметрів порядку чотири типи рішення [1-3]: рівновагу, коли після перехідного процесу система досягає стаціонарного стану, періодичне, квазіперіодичне рішення, а також хаос. Цим рішенням відповідають атрактори системи у вигляді стійкої рівноваги, граничного циклу, квазіперіодичного або хаотичного.

Системи (1) і (2) при зміні параметра ρ втрачають стійкість свого стану (режиму функціонування) і переходять в інший стан. Цей перехід називають біфуркацією [3].

Такий формальний підхід, наприклад, добре узгоджується з результатами теоретичних і експериментальних досліджень процесів крупного дроблення руд, як динамічних ОУ зі змінними структурами (розмірністю, режимом динаміки) і параметрами, які залежать від властивостей руди, конструктивних і технологічних змінних [4].

Ідентифікація таких складних ОУ традиційними способами вимагає великих витрат на експериментальні дослідження, тому доцільно використовувати методи нелінійної динаміки.

Відомо [1-3], що по одній тимчасовій реалізації можна визначити кореляційну ентропію, яка характеризує оцінку глибини точного прогнозу стану породжуючої системи, і в якому режимі вона знаходиться, а також кореляційну розмірність атрактора (порядок системи).

1.2.1. Аналіз методів ідентифікації процесу крупного дроблення руд

Найбільш ефективний підхід до вирішення завдань ідентифікації – це поєднання теоретичного і експериментального методу. Попередня теоретична оцінка дозволяє полегшити процес вимірювання. А результат експерименту допомагає уточнити математичний опис.

При ідентифікації складних об'єктів доцільно використовувати методи спрощення моделі, зниження її порядку, використання мінімізації. Однак ці

спрощення повинні забезпечувати збереження в використовуваній моделі істотних основних характеристик об'єкта в межах точності , що визначається вимогами розв'язуваної задачі.

В якості методів ідентифікації складних об'єктів управління великого поширення набули інтелектуальні методи.

Алгоритми методу групового обліку аргументів (МГОА) реалізують принцип самоорганізації моделей і відтворюють схему масової селекції [8-11]. МГОА фактично є методом пошуку закономірностей з автоматичним вибором структури і параметрів моделі, тобто реалізує як параметричну, так і структурну ідентифікацію, що є його основною перевагою. Метод ефективний в умовах малого обсягу експериментальних даних і високого рівня шумів [12].

Недоліком МГОА є те, що принцип реалізується при визначенні параметрів методу, призводить до його неадекватного застосування.

Алгоритм МГОА відносять до еволюційних алгоритмів (ЕА), загальним для яких є те, що вони моделюють процес біологічної еволюції: мутації структури і параметрів, їх схрещування (розмноження) і правило відбору. Це дозволяє виявляти їх сприятливі варіації, за допомогою яких будується послідовність поліпшених рішень [13-18].

ЕА (послідовний перебір і випадковий пошук) є методами глобальної оптимізації, які доцільно використовувати при структурній ідентифікації процесів крупного дроблення руд з різними динамічними режимами [7].

Серед ЕА найбільшого поширення набули генетичні алгоритми (ГА) [18-20], що представляють собою адаптивні методи пошуку, що основані на механізмах природного добору і успадкування на рівні генів. У них використовується еволюційний принцип виживання найбільш пристосованих особин (генів): розмноження найбільш пристосованих до зовнішнього середовища генів, а також виробництво генів з характеристиками, які були відсутні у генів попередніх поколінь. При цьому, основними операторами ГА є кросинговер, мутація, вибір батьків і селекція.

До переваг ГА відносять здатність знаходження глобального екстремуму, високу швидкодію при багатопроцесорній обробці і продуктивність. До недоліків – потреба у великому обсязі пам'яті і відносна низька швидкодія на фон-неймановських ЕОМ.

Для параметричної ідентифікації важливим є питання вибору типу базисних функцій. У МГОА, наприклад, в якості базисних функцій використовуються, як правило, поліноми Колмогорова-Габора. Розвитком поліноміального представлення функцій є нейронні мережі.

1.2.2. Нейронні мережі як метод ідентифікації складних ОУ

Нейронні мережі (НМ) являють собою обчислювальні структури, що моделюють процеси мозку і складаються з безлічі однакових елементів – штучних нейронів [21-26].

Обґрунтуванням ефективності використання НМ є доведення можливості представлення функції декількох змінних через суперпозицію функцій одного змінного, а також операцій їх складання і множення. Крім того, для НМ було доведено, що при нелінійної функції активації нейронів можна так побудувати мережу зв'язків і підібрати коефіцієнти w, щоб НМ як завгодно точно обчислювала будь-яку безперервну функцію Y від своїх входів X (рис. 1.1). Таким чином, НМ є універсальним та ефективним методом ідентифікації.

Архітектура НМ залежить від конкретної розв'язуваної задачі і характеризується кількістю шарів НМ, кількістю нейронів в кожному шарі і їх функціями активації, алгоритмами навчання мережі і т.д.

Архітектура багатошарової НМ прямого поширення (НМПП) складається з послідовно з'єднаних шарів, де нейрон кожного шару своїми входами пов'язаний з усіма нейронами попереднього шару, а виходами – наступного (див. рис. 1.1).

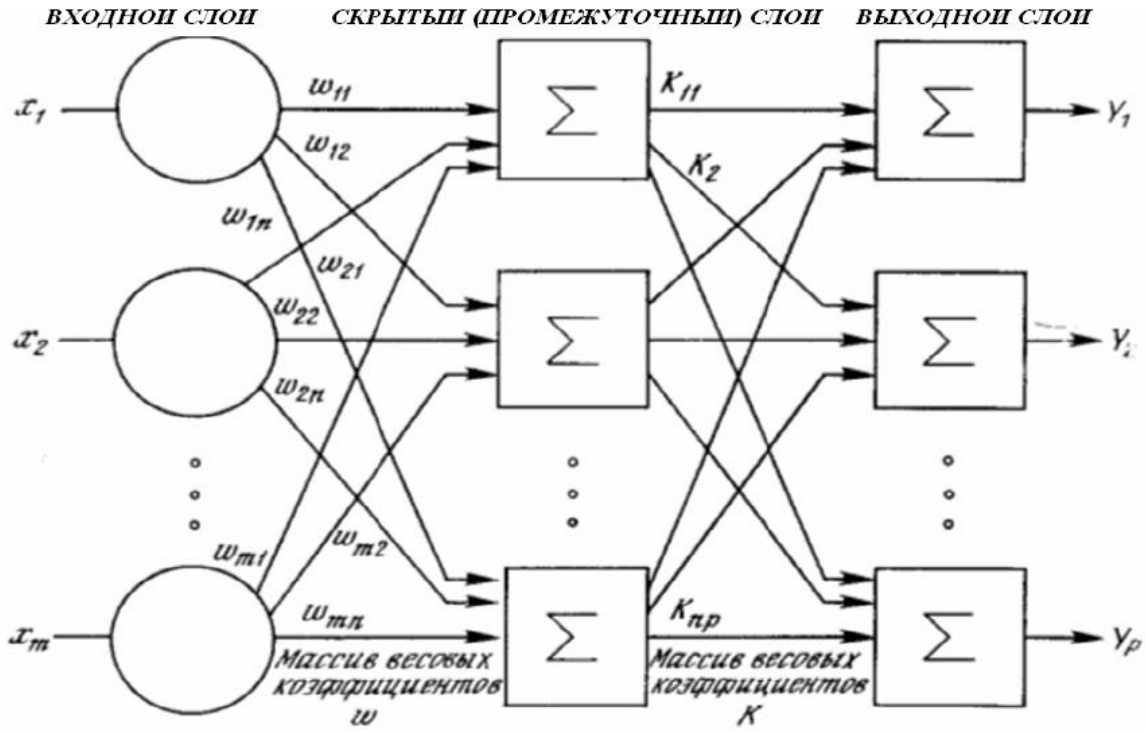


Рис. 1.1. Двошарова нейронна мережа прямого поширення.

Навчаються багатошарові НМ за допомогою алгоритму зворотного поширення помилки. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи.

НМ з радіальними базисними функціями (РБФ) – це двошарова мережа без зворотних зв'язків [21, 27], яка містить прихований (шаблонний) шар радіально симетричних нейронів і вихідний лінійний шар (рис. 1.2).

Навчання НМ з РБФ відбувається в два етапи. На першому етапі визначаються центри і відхилення для радіальних елементів, на другому – оптимізуються параметри лінійного вихідного шару.

Перевагою мереж з РБФ є те, що вони моделюють довільну нелінійну функцію за допомогою одного проміжного шару, тим самим, знімаючи проблему вибору числа шарів. Крім того, параметри лінійної комбінації в вихідному шарі можна оптимізувати за допомогою методів лінійної оптимізації, які швидко сходяться. До недоліків слід віднести погані екстраполюючі властивості мережі.

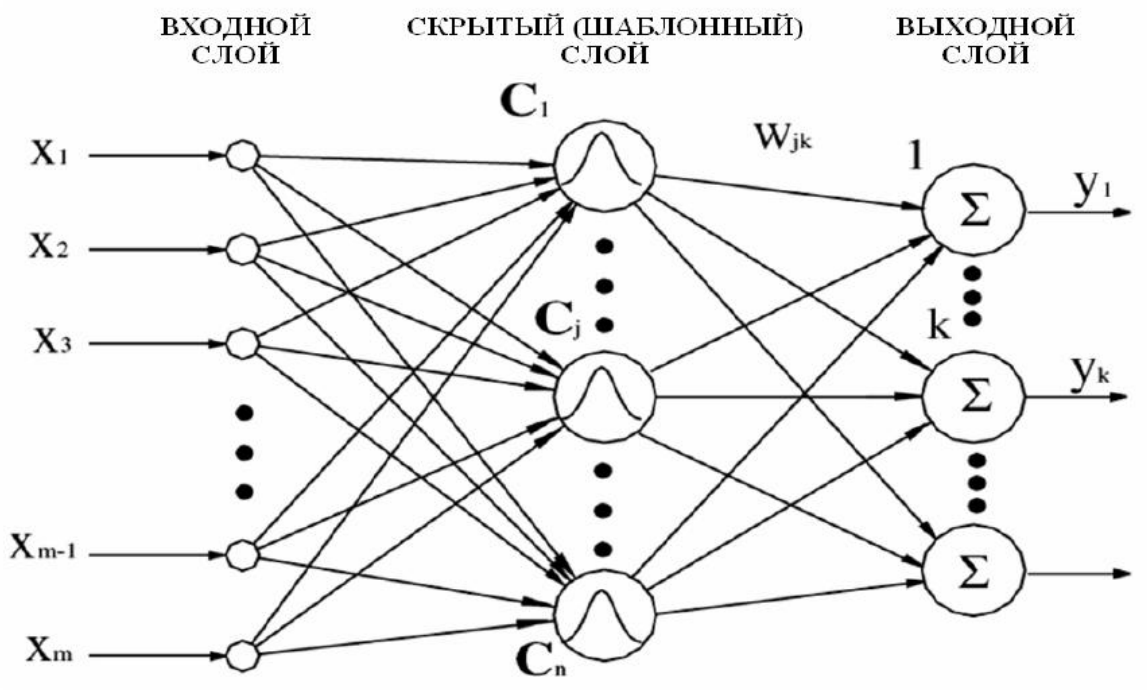


Рис. 1.2. Нейронна мережа з радіальними базисними функціями.

Для усунення недоліків НС запропоновані гібридні НС, варіантом яких є адаптивна нейронна система нечіткого виведення (Anfis – Adaptive neuro-fuzzy inference system), що реалізує систему нечіткого виводу Сугено у вигляді п'ятишарової НМПП сигналу (рис. 1.3).

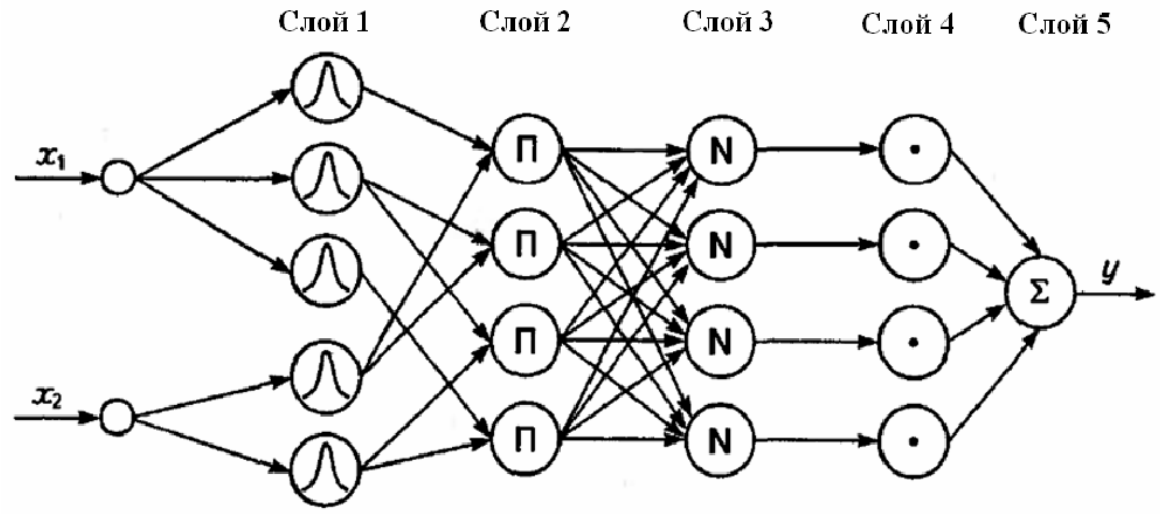


Рис. 1.3. Структура гібридної нейронної системи нечіткого виводу.

Призначення шарів наступні: шар 1 – терми вхідних змінних (x); шар 2 – антецеденти (посилки) нечітких правил; шар 3 – нормалізація степенів виконання правил; шар 4 – укладення правил; шар 5 – агрегування результату (y), отриманого за різними правилами.

Навчання гібридної мережі проводиться або за допомогою алгоритму зворотного поширення помилки, або комбінованим методом, розробленим для гібридних мереж.

1.3. Завдання структурно-параметричної ідентифікації

Сформулюємо задачу ідентифікації ОУ наступним чином. На основі експериментально отриманих множин функцій (часових рядів) збуджень, управлінь і виходів в умовах перешкод визначити структуру (узагальнену функцію) (структурна ідентифікація) і вектор параметрів (параметрична ідентифікація) моделі [31]:

$$Y[m + n] = \Phi\{Y[m], \xi[m], a[m], m\}, m = \overline{(1, M)}, \quad (3)$$

яка досить точно (в сенсі деякого критерію) апроксимує ОУ щодо вхідних і вихідних величин у всьому функціональному просторі.

Тут Y[m] – вектор (матриця) стану процесу;

ξ[m] – вектор (матриця) перешкод;

m – поточний такт часу (m = t / T), t – безперервний час;

T – період дискретизації;

M – час спостереження; n - глибина прогнозу.

Встановлено, що для вирішення задач ідентифікації технологічних процесів рудопідготовки доцільно застосовувати методи інтелектуальної обробки інформації [31].

1.3.1. Параметрична ідентифікація

Процедурою параметричної ідентифікації або оцінювання параметрів є визначення значень параметрів, що характеризують динаміку поведінки об'єкта, за допомогою певних способів обробки експериментальних даних в припущенні, що структура моделі досліджуваного об'єкта відома.

В даному випадку в прямих методах параметричної ідентифікації невідомі параметри моделі (3) визначаються на основі рішення системи рівнянь, одержуваних шляхом підстановки в модель (3) послідовностей експериментальних значень вхідних і вихідних величин ОУ [31].

При використанні в якості прогнозуючої моделі ОУ рекурентної гібридної або чіткої НМ для вирішення задачі параметричної ідентифікації (навчання) широко використовується градієнтний метод зворотного поширення помилки [21, 23]. Процес навчання при цьому (рис. 1.4) полягає в налаштуванні параметрів НМ за величиною помилки між відгуком моделі і необхідним по експериментальним вихідними даними (ВД). Звичайно, при цьому немає впевненості, що НМ навчилася найкращим чином, оскільки завжди існує можливість попадання алгоритму в локальний мінімум. Для цього використовуються спеціальні прийоми, що дозволяють «вибити» знайдене рішення з локального екстремуму. Якщо після декількох таких дій нейронна мережа збігається до того ж рішення, то можна зробити висновок про те, що знайдене рішення, швидше за все, оптимальне.

Така ідентифікація передбачає, що структура моделі (або ж діапазон її раціональних значень) відома, тобто вимагає виконання значних обсягів попередніх досліджень ОУ.

Рис. 1.4. Процес параметричної ідентифікації (навчання) НМ.

1.3.2. Структурно-параметрична ідентифікація

Процес структурно-параметричної ідентифікації (рис. 1.5) включає визначення структури, оцінку та оптимізацію параметрів моделі ОУ (3). Перші дві операції виконуються шляхом генерування (за допомогою базисних функцій) моделей-претендентів різної складності і налаштування їх параметрів з подальшою селекцією кращих з них за обраними критеріями. Результат – оптимальна структура моделі ОУ. Операція визначення оптимальних параметрів вирішується відомими методами оптимізації шляхом уточнення отриманих раніше значень параметрів за критеріями параметричної оптимізації на всій вибірці ВД [31].

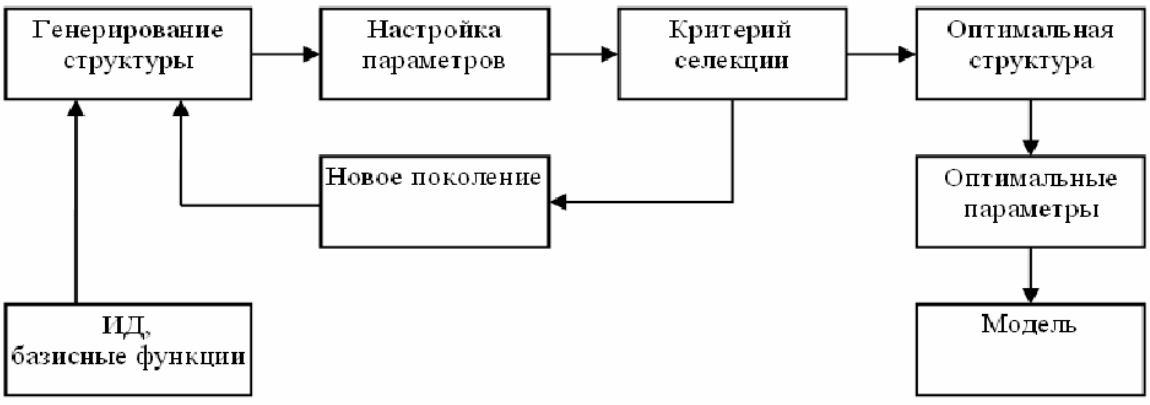


Рис. 1.5. Процес структурно-параметричної ідентифікації.

При структурній ідентифікації актуальними проблемами є вибір базисних функцій, в термінах яких здійснюється ідентифікація, а також вибір ефективного для конкретного випадку критерію селекції.

Методи глобальної оптимізації дозволяють вирішити структурно-параметричну ідентифікацію ОУ у вигляді НМ, як єдиного набору характеристик (параметрів). Використання комбінації методів глобальної і локальної оптимізації для задач структурно-параметричної ідентифікації ОУ, наприклад НМ і ГА, представляється найбільш перспективним.

При цьому, відкритим є питання доцільності проведення параметричної або структурно-параметричної ідентифікації для конкретного ОУ.

1.4. Визначення характеристик об'єкта управління

Ентропія Колмогорова K є найважливішою характеристикою руху у фазовому просторі довільної розмірності. Вона описує динамічну поведінку на атракторі і пропорційна швидкості втрати інформації про стан динамічної системи в часі. Для регулярного руху K-ентропія дорівнює нулю, для систем з детермінованим хаосом – позитивна і постійна. Ентропія K нескінченна в разі поведінки системи як білого шуму, що говорить про відсутність передбачуваності процесу.

Величина інформації про стан системи визначається як

$$K_k = -\sum_{i_0 \dots i_k} P_{i_0 \dots i_k} \ln P_{i_0 \dots i_k}, \qquad (4)$$

де $P_{i_0 \dots i_m}$ – спільна ймовірність того, що $x(t=0)$ знаходиться в осередку i_0 $x(t=T)$ – в осередку i_1, … , $x(t+kT)$ – в осередку i_k.

При цьому різниця $K_{k+1} - K_k$ є додатковою інформацією, яка необхідна для передбачення в якій комірці i_{k+1} буде, якщо раніше вона перебувала в осередках $i_1 \ldots i_k$.

Тоді К-ентропія дорівнює:

$$K = -\lim_{T\to 0}\lim_{\varepsilon\to 0}\lim_{N\to\infty}\frac{1}{NT}\sum_{k=0}^{N-1}(K_{k+1}-K_k) = -\lim_{T\to 0}\lim_{\varepsilon\to 0}\lim_{N\to\infty}\frac{1}{NT}\sum_{i_0\ldots i_N} P_{i_0\ldots i_N}\ln P_{i_0\ldots i_N} \quad (5)$$

де N – довжина тимчасової реалізації;

ε – розмір осередку фазового простору.

Для оцінки *К*-ентропії за експериментальними даними використовується величина кореляційної ентропії:

$$K_R = \lim_{\varepsilon\to 0}\lim_{k\to\infty}\ln\left[\frac{R_k(\varepsilon)}{R_{k+1}(\varepsilon)}\right] \le K, \quad (6)$$

де: $R_k(\varepsilon) = \lim_{N\to\infty}\frac{1}{N^2}\sum_{i,j}\upsilon[\varepsilon - \|x_i - x_j\|_k] = \sum_{i_1\ldots i_k} P^2_{i_1\ldots i_k}$ – узагальнений кореляційний інтеграл;

$\sum_{i,j}\upsilon[\varepsilon - \|x_i - x_j\|_k]$ – число пар i і j, для яких відстань $\|x_i - x_j\|_k < \varepsilon$; $\|x_i - x_j\|_k < \varepsilon$;

υ – ступінчаста функція Хевісайда $x_i = x[iT]$.

Додатково *К*-ентропія дозволяє визначити середній час, на який можна передбачити стан системи. При цьому, точне передбачення можливо тільки на інтервалі часу $T_{пр}$ такому, що $\varepsilon \cdot e^{KT_{пр}} = 1$ [2], тоді

$$T_{пр} = \frac{1}{K} \ln\left(\frac{1}{\varepsilon}\right) \tag{7}$$

Оцінка інтервалу передбачуваності за експериментальними даними виконується аналогічно виразу (7):

$$T_{Rпр} = \frac{1}{K_R} \ln\left(\frac{1}{\varepsilon}\right) \geq T_{пр} \tag{8}$$

Таким чином, за значенням кореляційного інтервалу передбачуваності визначається глибина точного прогнозу стану ОУ.

При цьому необхідний інтервал попередження (глибина прогнозу) визначається значенням суми періоду дискретизації та еквівалентного часу запізнювання системи.

Тоді з урахуванням (8) повинна виконуватися умова

$$T + \tau \leq T_{Rпр}, \tag{9}$$

з якого вибирається відповідне значення періоду дискретизації. Тут еквівалентний час запізнювання може бути визначено за максимумами взаємнокореляційних функцій вхідних і вихідний координат системи.

Іншим способом вибору періоду є статистичний підхід [5], згідно з яким з задовільною для практики точністю повинно виконуватися умова

$$T + \tau \leq 0,2 \cdot \tau_{кор} \tag{10}$$

де $\tau_{кор}$ – інтервал кореляції вихідний координати системи.

Домогтися виконання умови (10) можна шляхом послідовного збільшення періоду, що призводить до звуження спектра усередненого процесу щодо вихідного і, отже, розширенню його автокореляційної функції (тобто збільшення інтервалу кореляції усередненого процесу) [5].

Кореляційна розмірність атрактора характеризує складність атрактора динамічної системи, тобто характеризує мінімальну кількість відліків (глибину пам'яті) змінних, що входять в математичну модель системи.

Відстань між найближчими точками атрактора до і після біфуркацій знаходиться в універсальному відношенні [1]. Самоподобу такого явища описує фрактальна розмірність Хаусдорфа, яка є числом, що характеризує швидкість зростання числа осередків покриття даної множини при зменшенні розміру осередків:

$$D = -\lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log(\varepsilon)} \qquad (11)$$

де $N(\varepsilon)$ – кількість осередків.

Оцінка розмірності Хаусдорфа *D*, також як і оцінка *K*-ентропії може бути отримана за експериментальними даними [2].

Нехай траєкторія динамічної системи на атракторі описується як $x(t) = \left[x_1(t), ..., x_d(t)\right]$, і *d*-мірний фазовий простір розділено на осередки розміру ε^d. Тоді ймовірність попадання точки, що належить атрактору, в *i*-й осередок *(i = 1, 2, ...* $N(\varepsilon)$*)* обчислюється як:

$$p_i = \lim_{N \to \infty} \frac{N_i}{N}, \qquad (12)$$

де N_i – число точок в цьому осередку.

Чисельна оцінка розмірності D здійснюється відповідно до алгоритму Грассбергера-Прокаччі обчислення кореляційної розмірності:

$$D_R = \lim_{\varepsilon \to 0} \frac{\log(\sum_{i=0}^{N(\varepsilon)} p_i^2)}{\log \varepsilon} = \lim_{\varepsilon \to 0} \frac{\log R(\varepsilon)}{\log \varepsilon} \le D \tag{13}$$

де: $R(\varepsilon) = \lim_{N \to \infty} \frac{1}{N^2} \sum_{i,j} \upsilon[\varepsilon - \|x_i - x_j\|]$ – кореляційний інтеграл;

$\sum_{i=0}^{N(\varepsilon)} p_i^2$ – ймовірність того, що дві точки на атракторі лежать всередині осередку (ймовірність того, що дві точки атрактора розділені відстанню, меншою за ε).

Для визначення D_R будують залежність $\log R(\varepsilon)$ від $\log \varepsilon$ і шукають на ній лінійну ділянку, нахил якої і визначає шукане значення кореляційної розмірності D_R, яке далі використовується для визначення розмірності атрактора.

Розмірність d в фазовому просторі, починаючи з якої кореляційний розмір D_R перестає змінюватися, є мінімальним розміром вкладення атрактора, тобто найменшою цілої розмірністю фазового простору, яка містить весь атрактор. Тобто, розмірність фазового простору d визначається по залежності $D_R(d)$.

Разом з тим, з теореми про вкладення [2] випливає, що оцінка розмірності фазового простору визначається через оцінку розмірності атрактора D_R як:

$$d \ge 2D_R + 1 \tag{14}$$

При розрахунку кореляційної розмірності D_R однією з проблем є вибір величин обсягу експериментальних даних N і періоду дискретизації T [3]. Доцільно при цьому враховувати розмір тимчасового вікна $T_H = NT$ і брати до уваги існування обмеження на визначення максимальної величини кореляційної розмірності D_R:

$$D_{R\max} = \frac{2\lg N}{\lg(\varepsilon_{\max} / \varepsilon)}, \tag{15}$$

яке означає, що алгоритм розрахунку розмірності системи не може дати значення більше, ніж $D_{R\max}$ при заданому числі точок N. Тут $\varepsilon_{\max}$ – розмір атрактора.

Таким чином, обчислення розмірності D_R дозволяє реконструювати атрактор, а також визначити розмірність змінних математичної моделі ОУ (глибину пам'яті вхідних і вихідних змінних моделі).

1.5. Структурно-параметрична ідентифікація складного об'єкта управління

Процес структурно-параметричної ідентифікації включає операції визначення структури, оцінки та оптимізації параметрів моделі ОУ [31]. Перші дві операції вирішуються шляхом генерування (за допомогою базисних функцій) моделей-претендентів різної складності і налаштування їх параметрів з подальшою селекцією кращих з них за обраним критеріям (результат – оптимальна структура). Операція визначення оптимальних параметрів вирішується методами параметричної оптимізації шляхом уточнення отриманих раніше значень параметрів за критеріями регулярності на всій вибірці вихідних даних (результат – оптимальна модель).

При цьому актуальними проблемами є вибір базисних функцій, в термінах яких здійснюється ідентифікація, вибір способу генерування і селекції структур різної складності (метод структурної оптимізації), а також вибір методу параметричної оптимізації та ефективних критеріїв селекції та оптимізації.

Традиційно для апроксимації базисних функцій використовуються поліноми Лежандра, Колмогорова-Габора та ін. [9-10]. Коефіцієнти цих поліномів утворюють невідомі параметри, значення яких вибираються так, щоб найкраще відповідати спостережуваним часовими реалізаціями.

Більш продуктивним є використання нейронних мереж (НМ) або гібридних НМ з нечіткою логікою, які є універсальними та ефективними апроксіматорами.

Ефективним методом структурно параметричної ідентифікації є метод групового обчислення аргументів (МГОА) [9-10]. Його алгоритми реалізують схему масової селекції шляхом генерування моделей – приватних описів (схрещування) і відбору кращих з них (селекція). Але значною проблемою застосування МГОА є правильне співвідношення складності моделі з об'ємом навчальної вибірки.

Синтезована модель ОУ, яка правильно передає динаміку одного режиму функціонування може бути неадекватною до опису іншого режиму. Тому необхідна реалізація адаптивної ідентифікації ОУ в процесі функціонування системи управління.

Жорсткі вимоги до знання статистичних властивостей часових рядів експериментальних сигналів в системах управління обмежують можливості методів математичної статистики, теорії розпізнавання образів, теорії випадкових процесів та ін.

Багато реальних процесів не можуть бути адекватно описані за допомогою традиційних статистичних моделей, оскільки є суттєво нелінійними і мають або хаотичну, або квазіперіодичну, або змішану динаміку [1].

Для побудови та реалізації структури динамічної прогнозуючої моделі ОУ можуть використовуватися різні підходи. При цьому відомо [32], що нелінійна динамічна система (модель ОУ) може бути представлена шляхом композиції лінійного динамічного (ЛДЛ) та нелінійного статичного (НСЛ) ланок, наприклад, у вигляді моделі Вінера-Гаммерштайна (Wiener-Hammerstein), яка приведена на рис. 1.6.

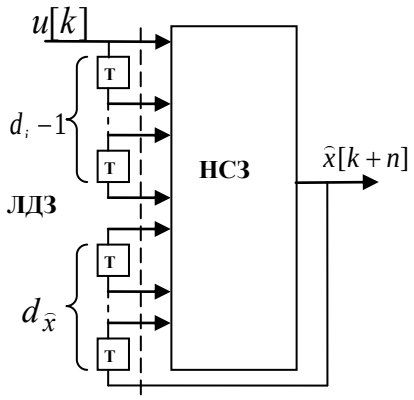


Рис. 1.6. Структура Вінера-Гаммерштайна прогнозуючої моделі нелінійного динамічного ОУ

Тут $u[k]$, $\hat{x}[k+n]$ – вхід процесу і оцінка прогнозу його виходу;

k, n – поточний такт часу і глибина прогнозу.

ЛДЛ є лініями затримки, величини яких (глибина пам'яті) визначаються розмірністю вхідних $d_i - 1$ і вихідних $d_{\hat{x}}$ змінних [33], а в якості НСЛ можуть використовуватися як традиційні засоби (поліноми Лежандра, Вольтерра, Колмогорова-Габора та ін.), так і інтелектуальні (НМ, гібридні НМ з нечіткою логікою та ін.).

Завдання ідентифікації ОУ формулюється таким чином [34]: на підставі експериментальної множини функцій (часових рядів) збуджень, управлінь і виходів в умовах перешкод визначити структуру (узагальнену функцію $F_{\hat{x}}$) і вектор параметрів $a_{\hat{x}}$ моделі ОУ, яка досить точно (в сенсі деякого критерію)

апроксимує його щодо вхідних і вихідних величин у всьому функціональному просторі.

Формування оцінки структури $F_{\hat{x}}$ (структурна ідентифікація) і параметрів $a_{\hat{x}}$ (параметрична ідентифікація) моделі ОУ здійснюється на основі векторів сигналів спостереження шляхом мінімізації прийнятого функціоналу $J\{F_{\hat{x}}, a_{\hat{x}}\}$.

У загальному випадку функціонал $J\{F_{\hat{x}}, a_{\hat{x}}\}$ полімодальний (має кілька локальних мінімумів), що вимагає використання методів глобальної оптимізації, серед яких найбільш ефективними є пошукові методи [20]. У них алгоритм пошуку оптимального рішення пов'язує такі рішення таким чином, щоб отримати нове краще рішення.

В алгоритмах прямого випадкового пошуку (ПВП) задаються напрямки пошуку і визначаються значення функціоналу J в точках $\Psi(j) \pm \gamma\varsigma$. Рішення полягає у виборі кроку в напрямку зменшення цього функціоналу:

$$\Psi(j+1) = \Psi(j) - \omega\varsigma\{J[\Psi(j) + \gamma\varsigma] - J\left[\Psi(j) - \gamma\varsigma\right]\}, \qquad (16)$$

де: $\Psi(j) \subset \{F_j, a_j\}$ – значення функції і параметрів на j-му кроці пошуку;

$\omega, \varsigma, \gamma$ – параметри, які визначають сфери прийняття рішення(ω), збору інформації (γ) і одиничний випадковий напрям(ς).

У загальному випадку параметри в (1) можуть змінюватися – адаптуватися до процедури пошуку і виду гіперповерхні прийнятого функціоналу.

Розвитком пошукових методів є еволюційні алгоритми, серед яких найбільш поширені генетичні алгоритми (ГА), що моделюють розвиток біологічної популяції на рівні геномів: мутації структури і параметрів $\delta\Psi$, їх схрещування (розмноження) [19]:

$$\Psi(j+1) = \Psi(j) + \delta\Psi(j) \tag{17}$$

і правило відбору, що дозволяє виявляти їх сприятливі варіації, за допомогою яких будується послідовність поліпшених рішень.

Більшість завдань, що вирішуються за допомогою ГА, мають один критерій оптимізації. Багатокритеріальна оптимізація (БО) заснована на знаходженні рішення, одночасно оптимізуючого більш ніж одну функцію. В цьому випадку шукається певний компроміс, в ролі якого виступає рішення, оптимальне в сенсі Парето. При БО, що використовує ГА вибирається не одна хромосома, що представляє собою оптимальне рішення в звичайному сенсі, а безліч хромосом, оптимальних в сенсі Парето. Користувач має можливість вибрати оптимальне рішення з цієї множини:

$$m \cdot \Psi(j+1) = m \cdot [\Psi(j) + \delta\Psi(j)], \tag{18}$$

де $m \geq 2$ – число розглянутих критеріїв.

Для ідентифікації ефективні зовнішні критерії, адекватні завданню побудови моделей з мінімальною дисперсією помилки прогнозу, які діляться на критерії регулярності і критерії незміщеності [9].

Критерій регулярності заснований на поділі даних на навчальну *A* і перевірочну *B* вибірки:

$$J_{рег} = \frac{\left\| x_B^*[d+n] - \hat{x}_B[d+n] \right\|}{\left\| x_B^*[d+n] \right\|}, \tag{19}$$

де: $x_B^*, \hat{x}_B$ – експериментальні та отримані за моделлю значення виходу ОУ;

d – глибина пам'яті,

n – глибина прогнозу.

Оптимізація моделі здійснюється на навчальній, а перевірка її ефективності (величини помилки) на перевірочної вибірці. Вся вибірка $N = A + B$.

Більш стійкі до перешкод критерії мінімуму зміщення (незміщеності). Наприклад, критерій незміщеності, заснований на аналізі рішень має вигляд:

$$J_{см} = \frac{\left\| \hat{x}_A[d+n] - \hat{x}_B[d+n] \right\|}{\left\| x^*[d+n] \right\|}, \qquad (20)$$

де $\hat{x}_A, \hat{x}_B$ – виходи моделей ОУ, які навчені на вибірках *A* і *B*, відповідно. Обчислення критерію $J_{см}$ здійснюється на всій вибірці N.

Критерій мінімуму зміщення дозволяє вибрати модель найменш чутливу до зміни безлічі точок, за якими вона отримана. Така модель повинна давати однакові результати на вибірках *A* і *B*. Тому, цей критерій рекомендується для вирішення завдань структурної ідентифікації.

Вибір методу локальної (параметричної) оптимізації визначається вибором типу базисних функцій.

Для прогнозуючих систем на базі НМ найкращі якості показує гетерогенна мережа, яка складається з прихованих шарів з нелінійною функцією активації нейронів і вихідного лінійного нейрона [35].

Базисна функція у вигляді НМ прямого поширення (НМПП) з прихованим шаром представляється як [35]:

$$\hat{x}[k+n] = \sum_{\theta \in P} F_l \{ \sum_{l \in Q} v_l[\theta] \cdot F_y (\sum_{m \in Q} v_{l,m}[\theta] \cdot y_m[k-\theta]) \}, \qquad (21)$$

де P – множина глибин пам'яті відповідних входів;

F_l – активаційна функція вихідного шару НМ;

Q – безліч входів нейронів;

l – порядковий номер входу вихідного шару НМ;

v_l – вагові коефіцієнти вихідного шару;

F_y – активаційна функція нейронів прихованого шару; m - порядковий номер входу НМ;

$v_{l,m}$ – вагові коефіцієнти зв'язку m-го входу і l-го нейрона;

y_m – вхід НМ.

Параметрами налаштування (навчання) цієї НМ є $\{v_l, v_{l,m}\} \subset a_{\hat{x}}$.

У загальному випадку структурними характеристиками НМ (6) є $\{T_s, P, F_y, F_l, r_s, M_{po}\} \subset F_{\hat{x}}$, де T_s – тип структури, $r_s \subset Q$ – розмір прихованого шару, M_{po} – метод параметричної оптимізації (функція навчання НМ).

До НМПП відносяться персептрони, каскадні НМ, вейвнети і ін.

Базисна функція у вигляді НС з радіальними базисними функціями (РБФ) представляється як [35]:

$$\hat{x}[k+n] = \sum_{\theta \in P} F_l\{\sum_{l,m \in Q} v_l \cdot F_y(\vartheta_l, \|y_m[k-\theta] - \upsilon_l\|)\}, \qquad (22)$$

де ϑ_l, υ_l – параметри РБФ l-го нейрона прихованого шару.

Параметрами налаштування НМ (7) є $\{v_l, \vartheta_l, \upsilon_l\} \subset a_{\hat{x}}$, а її структурними характеристиками – $\{T_s, P, F_y, F_l, r_s, M_{po}\} \subset F_{\hat{x}}$.

Теоретично, системи з нечіткою логікою та НМ еквівалентні один одному, проте на практиці у них є свої переваги і недоліки. У зв'язку з цим отримали розвиток гібридні НМ, в яких висновки робляться на основі апарату нечіткої логіки, а функції належності підлаштовуються з використанням алгоритмів навчання НМ. Такі системи не тільки

використовують апріорну інформацію, але можуть набувати нових знань і для користувача є логічно прозорими [21].

Базисна функція у вигляді гібридної НМ з нечіткою логікою (Anfis) представляється як [35]:

$$\hat{x}[k+n] = \sum_{\theta \in P} \sum_{m \in Q} \beta_m[\theta] \cdot \alpha_m[k-\theta], \quad (23)$$

де: $\beta_m[\theta] = U_m^{-1}(\alpha_m[\theta] / \sum_m \alpha_m[\theta])$;

$\alpha_m[k-\theta] = \underset{l,m \in Q}{Tn} \{L_{l,m}(y_m[k-\theta])\}$;

$U = U(a_U)$;

$L = L(a_L)$.

Тут U_m^{-1} – функція, зворотна функції належності проміжного виходу m мережі з параметрами a_U;

α_m – значення проміжного виходу;

Tn – довільна t – норма моделювання логічної операції «І»;

$L_{l,m}$ – функція належності нечіткого правила l входу m з параметрами a_L.

Параметрами налаштування НМ (8) є $\{a_U, a_L\} \subset a_{\hat{x}}$, а структурними характеристиками – $\{T_s, P, U_m, L_{l,m}, r_p, M_{po}\} \subset F_{\hat{x}}$, де $r_p \subset Q$ – кількість правил розкладання по входах.

Процес навчання НС представлений на рис. 1.7.

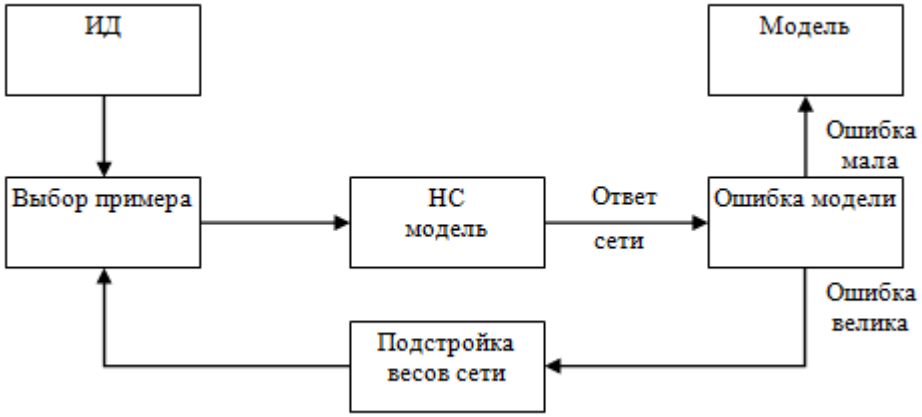


Рис. 1.7. Процес навчання НС.

Ідентифікація параметрів (навчання) НМПП (20) здійснюється за допомогою градієнтних алгоритмів навчання: алгоритмів методу сполучених градієнтів, алгоритмів зворотного поширення помилки або квазіньютонівських алгоритмів. При навчанні НМ з РБФ (21) спочатку визначаються центри і відхилення для радіальних елементів, після чого оптимізуються параметри лінійного вихідного шару. Навчання гібридної НМ (22) виконується аналогічно НМПП (20) шляхом оптимізації параметрів функцій належності за допомогою гібридного алгоритму або алгоритму зворотного поширення помилки [27]. Перевагою цих алгоритмів параметричного навчання НМ є їх простота і швидкодія, а недоліком – їх локальність.

Висновки розділу 1

Аналіз методів і алгоритмів ідентифікації технологічних процесів складних ОУ показав, що розвиток отримали параметричні методи ідентифікації ОУ. При цьому, вибір структур цих моделей, як правило, здійснюється евристично. Для ідентифікації процесів крупного дроблення руд перспективними є методи систем штучного інтелекту, зокрема нейронні мережі і системи з нечіткою логікою, які є універсальними та ефективними апроксіматорами.

Була запропонована методика структурно-параметричної ідентифікації процесів крупного дроблення руд, що включає процедури генерування структур, селекцію моделей і оптимізацію їх параметрів. При цьому для вирішення завдання ідентифікації використовуються алгоритми глобальної і локальної оптимізації.

РОЗДІЛ 2
ВИКОРИСТАННЯ ПРОГРАМНИХ ІНСТРУМЕНТІВ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ В ЗАДАЧАХ СТРУКТУРНО-ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ

2.1. Програмні пакети для математичної обробки даних

Формалізація завдання і застосування чисельних методів дозволяють використовувати добре вивчені прийоми рішення і стандартне (універсальне) математичне забезпечення ЕОМ. Застосування ЕОМ підвищує ефективність наукових досліджень, дозволяє проводити моделювання складних об'єктів і явищ.

Для автоматизації математичних розрахунків використовуються різноманітні обчислювальні засоби від мікрокалькуляторів до надпотужних супер ЕОМ. Проте, такі розрахунки залишаються складною справою. Більш того, застосування комп'ютерів внесло свої нові труднощі: перш ніж почати розрахунки, користувач повинен освоїти основи програмування на одному або декількох мовах програмування і чисельні методи розрахунків. Положення стало змінюватися після появи спеціалізованих програмних комплексів для автоматизації математичних та інженерно-технічних розрахунків.

Математичними системами, універсальними математичними пакетами (середовищами) називають пакети прикладних програм, що містять різноманітні інструменти для вирішення математичних завдань. Такі системи є програмними додатками, підтримуваними середовищем Windows і ресурсами самого ПК, а також дозволяють імпортувати документи з інших додатків в широкому діапазоні їх форматів.

До найбільш поширених математичних пакетів можна віднести MathCad, Maple та MatLab.

2.1.1. Математична система MathCad

MathCad є візуальним середовищем математичних обчислень, що дозволяє проводити різноманітні наукові та інженерні розрахунки, починаючи від елементарної арифметики і закінчуючи складними реалізаціями чисельних методів. Ця система відноситься до класу систем автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводом. Відрізняється легкістю використання і застосування для колективної роботи.

Відповідно до проблем реального життя система MathCad розрахована на ряд основних завдань:

- введення на комп'ютері різноманітних математичних виразів (для подальших розрахунків або створення документів, презентацій, Web-сторінок);
- проведення математичних розрахунків;
- підготовка графіків з результатами розрахунків;
- введення вихідних даних і виведення результатів в інших файлових форматах;
- підготовка Web-сторінок і публікація результатів в Інтернеті;
- отримання різної довідкової інформації з області математики.

MathCad містить сотні операторів і вбудованих функцій для вирішення різних технічних завдань. Програма дозволяє виконувати чисельні і символьні обчислення, проводити операції зі скалярними величинами, векторами і матрицями, автоматично переводити одні одиниці вимірювання в інші.

Основна відмінність Mathcad від аналогічних програм – це графічний, а не текстовий режим введення виразів. Для набору команд, функцій, формул можна використовувати як клавіатуру, так і кнопки на численних спеціальних панелях інструментів. Обчислення з введеними формулами

здійснюються за бажанням користувача: миттєво, одночасно з набором або по команді. Звичайні формули обчислюються зліва направо і зверху вниз (подібно читання тексту). Будь-які змінні, формули, параметри можна змінювати, спостерігаючи на власні очі відповідні зміни результату. Це дає можливість організації дійсності інтерактивних обчислювальних документів.

Слід зазначити можливість використання в розрахунках MathCad величин з розмірностями, причому можна вибрати систему одиниць: СІ, СГС, МКС, англійську, або побудувати власну. Результати обчислень, зрозуміло, також отримують відповідну розмірність. Користь від такої можливості важко переоцінити, оскільки значно спрощується відстеження помилок в розрахунках, особливо в фізичних та інженерних.

У середовищі MathCad фактично немає графіків функцій в математичному розумінні терміну, а є візуалізація даних, що знаходяться в векторах і матрицях (тобто здійснюється побудова як ліній, так і поверхонь по точках з інтерполяцією), хоча користувач може про це і не знати, оскільки у нього є можливість використання безпосередньо функцій однієї або двох змінних для побудови графіків або поверхонь відповідно. Так чи інакше, механізм візуалізації MathCad значно поступається такому в Maple, де досить мати тільки вид функції, щоб побудувати графік або поверхню будь-якого рівня складності. У порівнянні з Maple, графіка MathCad має ще такі недоліки, як: неможливість побудови поверхонь, заданих параметрично, з не прямокутної областю визначення двох параметрів; створення і форматування графіків тільки через меню, що обмежує можливості програмного керування параметрами графіки.

Відкрита архітектура програми в поєднанні з підтримкою технологій .NET і XML дозволяє легко інтегрувати MathCad практично в будь-які IT-структури і інженерні додатки, серед яких – SmartSketch , VisSim/Comm PE, Pro/ENGINEER .

2.1.2. Математична система Maple

Maple – система комп'ютерної математики, призначена головним чином для виконання аналітичних (символьних) обчислень, і має для цього один з найпотужніших в своєму класі арсенал спеціалізованих процедур і функцій (понад 3000). Володіє розвиненими графічними засобами. Має власну мову програмування. Ядро системи Maple використовується в ряді інших математичних систем, наприклад в MATLAB і MathCad, для реалізації в них символьних обчислень. Так само як і пакет Mathematica, СКМ Maple підтримує технологію MathML.

СКМ Maple – типова інтегрована програмна система. Вона об'єднує в собі:

- потужну мову програмування (вона же мова для інтерактивного спілкування з системою);
- редактор для підготовки і редагування документів і програм;
- сучасний багатовіконний інтерфейс з можливістю роботи в діалоговому режимі;
- потужну довідкову систему з багатьма тисячами прикладів;
- ядро алгоритмів і правил перетворення математичних виразів;
- чисельний і символьний процесори;
- систему діагностики;
- бібліотеки вбудованих і додаткових функцій;
- пакети функцій сторонніх виробників і підтримку деяких інших мов програмування і програм.

Maple має вхідну мову надвисокого рівня, орієнтовану на рішення математичних задач практично будь-якої складності. Вона служить для задавання системі питань або, кажучи інакше, задавання вхідних даних для подальшої їх обробки. Це мова інтерпретуючого типу і за своєю ідеологією нагадує BASIC. Має Maple і свою мову процедурного програмування –

Maple-мова. Ця мова має цілком традиційні засоби структурування програм: оператори циклів, оператори умовних і безумовних переходів, оператори порівняння, логічні оператори, команди управління зовнішніми пристроями, функції користувача, процедури і т. д. Він також включає в себе всі команди і функції вхідної мови, йому доступні всі спеціальні оператори та функції. Багато з них є досить серйозними програмами, наприклад символьне диференціювання, інтегрування, розкладання в ряд Тейлора, побудова складних тривимірних графіків і т. Д.

Мовою реалізації Maple є одна з найкращих і потужних універсальних мов програмування – C. На ньому написано ядро системи, що містить ретельно оптимізовані процедури. Більшість же функцій, які містяться в пакетах, написані на Maple-мові, завдяки чому їх можна модифікувати і навіть писати свої власні бібліотеки. Синтаксис структурних операторів мови Maple нагадує суміш Бейсика і Паскаля.

2.1.3. Математична система MATLAB

MATLAB являє собою основу сімейства продуктів MathWorks і є головним інструментом для вирішення широкого спектра наукових і прикладних задач, в таких областях як: моделювання об'єктів і розробка систем управління, проектування комунікаційних систем, обробка сигналів і зображень та ін. Так само в цій системі є однойменна інтерпретуюча мова програмування, яка використовується для моделювання процесів.

Мова MATLAB є високорівневою мовою програмування, що включає засновані на матрицях структури даних, широкий спектр функцій, інтегроване середовище розробки, об'єктно-орієнтовані можливості і інтерфейси до програм, написаних на інших мовах програмування. MATLAB в порівнянні з традиційними мовами програмування (C/C++, Java, Pascal, FORTRAN) дозволяє на порядок скоротити час вирішення типових завдань і значно спрощує розробку нових алгоритмів.

Програми, написані на MATLAB, бувають двох типів – функції і скрипти. Функції мають вхідні і вихідні аргументи, а також власний робочий простір для зберігання проміжних результатів обчислень і змінних. Скрипти ж використовують загальний робочий простір. Як скрипти, так і функції зберігаються у вигляді текстових файлів і компілюються в машинний код динамічно . Існує також можливість зберігати так звані pre-parsed програми – функції і скрипти, оброблені в вид, зручний для машинного виконання. У загальному випадку такі програми виконуються швидше звичайних, особливо якщо функція містить команди побудови графіків.

MATLAB надає зручні засоби для розробки алгоритмів, включаючи високорівневі з використанням концепцій об'єктно-орієнтованого програмування. У ньому є всі необхідні засоби інтегрованого середовища розробки, включаючи зневаджувач і профайлер. Функції для роботи з цілими типами даних полегшують створення алгоритмів для мікроконтролерів і інших додатків, де це необхідно.

У складі пакету MATLAB є велика кількість функцій для побудови графіків (рис. 2.1), в тому числі тривимірних, візуального аналізу даних і створення анімованих роликів.

Вбудоване середовище розробки дозволяє створювати графічні інтерфейси користувача з різними елементами управління, такими як кнопки, поля введення і іншими.

Програми MATLAB, як консольні, так і з графічним інтерфейсом користувача, можуть бути зібрані за допомогою компоненти MATLAB Compiler в незалежні від MATLAB програми або динамічні бібліотеки, для запуску на інших комп'ютерах, проте, потрібна установка вільно розповсюджуваного середовища MATLAB Compiler Runtime (MCR).

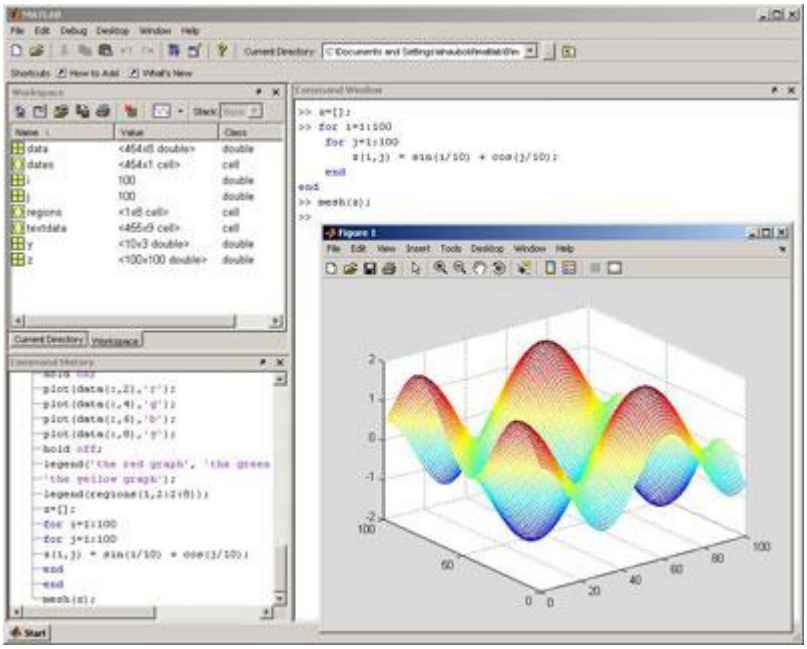


Рис. 2.1. Створення тривимірного графіка в середовищі MATLAB

Пакет MATLAB включає різні інтерфейси для отримання доступу до зовнішніх підпрограм, написаним на інших мовах програмування, даним, клієнтам і серверам, що спілкуються через технології Component Object Model або Dynamic Data Exchange, а також периферійних пристроїв, які взаємодіють безпосередньо з MATLAB. Багато з цих можливостей відомі під назвою MATLAB API.

Пакет MATLAB включає інтерфейс взаємодії з зовнішніми додатками, написаними на мовах C і FORTRAN. Здійснюється ця взаємодія через MEX-файли. Існує можливість виклику підпрограм, написаних на C або FORTRAN з MATLAB, ніби це вбудовані функції пакета. MEX-файли являють собою спільні бібліотеки, які можуть бути завантажені і виконані інтерпретатором, вбудованим в MATLAB. MEX-процедури мають також можливість викликати вбудовані команди MATLAB.

Інтерфейс MATLAB, що відноситься до загальних DLL, дозволяє викликати функції, що знаходяться в звичайних бібліотеках, що динамічно підключаються, прямо з MATLAB. Ці функції повинні мати C-інтерфейс. Крім того, в MATLAB є можливість отримати доступ до його вбудованих функцій через цей інтерфейс, що дозволяє використовувати функції пакета в

зовнішніх додатках, написаних на C. Ця технологія в MATLAB називається C Engine.

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

- Цифрова обробка сигналів, зображень та даних: DSP Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communication Toolbox, Filter Design Toolbox – набори функцій, що дозволяють вирішувати широкий спектр завдань обробки сигналів, зображень, проектування цифрових фільтрів і систем зв'язку.
- Системи управління : Control Systems Toolbox, Analysis and Synthesis Toolbox, Robust Control Toolbox, System Identification Toolbox, LMI Control Toolbox,Model Predictive Control Toolbox, Model-Based Calibration Toolbox – набори функцій, що полегшують аналіз і синтез динамічних систем, проектування, моделювання та ідентифікацію систем управління, включаючи сучасні алгоритми управління.
- Фінансовий аналіз: GARCH Toolbox, Fixed-Income Toolbox, Financial Time Series Toolbox, Financial Derivatives Toolbox, Financial Toolbox, Datafeed Toolbox – набори функцій, що дозволяють швидко і ефективно збирати, обробляти і передавати різну фінансову інформацію.
- Аналіз і синтез географічних карт, включаючи тривимірні: Mapping Toolbox.
- Збір і аналіз експериментальних даних: Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, Link for Code Composer Studio – набори функцій, що дозволяють зберігати і обробляти дані, отримані в ході експериментів, в тому числі в реальному часі. Підтримується широкий спектр наукового і інженерного вимірювального обладнання.

- Візуалізація і уявлення даних: Virtual Reality Toolbox – дозволяє створювати інтерактивні світи і візуалізувати наукову інформацію за допомогою технологій віртуальної реальності і мови VRML .
- Засоби розробки: MATLAB Builder for COM, MATLAB Builder for Excel, MATLAB Builder for NET, MATLAB Compiler, Filter Design HDL Coder – набори функцій, що дозволяють створювати незалежні програми з середовища MATLAB.
- Взаємодія з зовнішніми програмними продуктами: MATLAB Report Generator, Excel Link, Database Toolbox, MATLAB Web Server, Link for ModelSim – набори функцій, що дозволяють зберігати дані в різних видах таким чином, щоб інші програми могли з ними працювати.
- Бази даних: Database Toolbox – інструменти роботи з базами даних.
- Наукові і математичні пакети: Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed-Point Toolbox, Fuzzy Logic Toolbox, Genetic Algorithm and Direct Search Toolbox, OPC Toolbox, Optimization Toolbox, Partial Differential Equation Toolbox, Spline Toolbox, Statistic Toolbox, RF Toolbox. Ці набори спеціалізованих математичних функцій дозволяють вирішувати широкий спектр наукових і інженерних задач, включаючи розробку генетичних алгоритмів .
- Нейронні мережі: Neural Network Toolbox – інструменти для синтезу та аналізу нейронних мереж.
- Нечітка логіка: Fuzzy Logic Toolbox – інструменти для побудови і аналізу нечітких множин.
- Символьні обчислення: Symbolic Math Toolbox – інструменти для символьних обчислень з можливістю взаємодії з символьним процесором програми Maple .

На підставі результатів порівняльного аналізу пакетів для математичної обробки даних, для реалізації методики структурно-параметричної ідентифікації процесів крупного дроблення руд був обраний пакет MATLAB. Переваги обраного інструменту полягають в зручних засобах для розробки

алгоритмів, включаючи високорівневі з використанням концепцій об'єктно-орієнтованого програмування, і наявності інструментів для синтезу та аналізу нейронних мереж.

2.2. Нейронні мережі в MATLAB

Нейронні мережі (НМ) широко використовуються для вирішення різноманітних завдань. Серед областей застосування НМ, що розвиваються – обробка аналогових і цифрових сигналів, синтез і ідентифікація електронних ланцюгів і систем.

Прикладами застосування технології нейронних мереж для цифрової обробки сигналів є: фільтрація, оцінка параметрів, детектування, ідентифікація систем, розпізнавання образів, реконструкція сигналів, аналіз часових рядів і стиснення. Згадані види обробки застосовні до різноманітних видів сигналів: звукових, відео, мовних, зображення, передачі повідомлень, геофізичних, локаційних і інших.

Нейронні мережі – це новий напрямок в практиці створення технічних систем. Можливості нейронних мереж виконувати операції порівняння за зразком і класифікації об'єктів, недоступні для традиційної математики, дозволяють створювати штучні системи для вирішення завдань розпізнавання образів, діагностики захворювань, автоматичного аналізу документів і багатьох інших нетрадиційних додатків.

Нейронні мережі – це виключно потужний метод імітації процесів і явищ, що дозволяє відтворювати надзвичайно складні залежності. Нейронні мережі по своїй природі є нелінійними, в той час як протягом багатьох років для побудови моделей використовувався лінійний підхід. Крім того, у багатьох випадках нейронні мережі дозволяють подолати «прокляття розмірності», обумовлене тим, що моделювання нелінійних явищ в разі великого числа змінних вимагає величезної кількості обчислювальних ресурсів.

Інша особливість нейронних мереж пов'язана з тим, що вони використовують механізм навчання. Користувач нейронної мережі підбирає представницькі дані, а потім запускає алгоритм навчання, який автоматично налаштовує параметри мережі. При цьому від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати і готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніше, ніж, наприклад, при використанні традиційних методів.

Штучні нейронні мережі засновані на біологічній моделі нервової системи і складені з безлічі простих елементів, що діють паралельно. Як і в природі, функція нейронної мережі в значній мірі визначається зв'язками між елементами. Нейронну мережу можна навчати для виконання конкретної функції, регулюючи значення коефіцієнтів (ваг) зв'язку. Зазвичай штучні нейронні мережі налаштовуються або навчаються так, щоб конкретні входи перетворювалися в заданий цільовий вихід. Мережа налаштовується (навчається), грунтуючись на порівнянні сигналів виходу і цілі до тих пір, поки вихід мережі не буде відповідати меті. Щоб навчити мережу при такому керованому навчанні, як правило, використовується багато пар значень сигналів вхід/мета.

Основи теорії і технології застосування НМ широко представлені в пакеті MATLAB Neural Network Toolbox. NNT – це пакет розширення MATLAB, що містить засоби для проектування, моделювання, розробки та візуалізації нейронних мереж. У зв'язку з цим особливо слід відзначити GUI (Graphical User Interface – графічний інтерфейс користувача) для НМ – NNTool (рис. 2.2). Інтерфейс NNTool дозволяє створювати нейронні мережі з одним або двома шарами. У таблиці 2.1 описані всі доступні з пакета типи нейромереж.

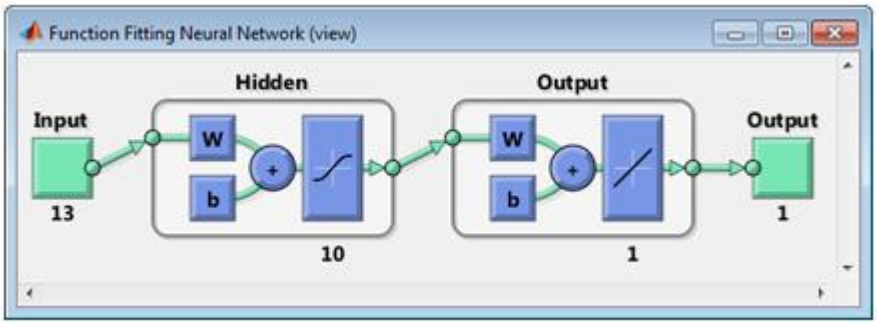


Рис. 2.2. Створення нейронної мережі через графічний інтерфейс NNTool.

Пакет забезпечує всебічну підтримку типових нейромережевих парадигм і має відкриту модульну архітектуру. Так само він містить функції командного рядка і графічний інтерфейс користувача для швидкого покрокового створення нейромереж.

Таблиця 2.1. Мережі, доступні для роботи з інтерфейсом NNTool.

№ п/п	Тип мережі	Назва мережі	Кількість шарів	Параметри
1	Competitive	Конкуруюча мережа	1	IW{1,1}, b{1}
2	Cascade-forward backprop	Каскадна мережа з прямим розповсюдженням сигналу і зворотним поширенням помилки	2	IW{1,1}, b{1}, LW{2,1}, IW{2,1}, b{2}
3	Elman backprop	Мережа Елмана зі зворотним поширенням помилки	2	IW{1,1}, b{1}, LW{2,1}, b{2}, LW{1,1}
4	Feed-forward backprop	Мережа з прямим розповсюдженням сигналу і зворотним поширенням помилки	2	IW{1,1}, b{1}, LW{2,1}, b{2}
5	Time delay backprop	Мережа з запізненням і зворотним поширенням помилки	2	IW{1,1}, b{1}, LW{2,1}, b{2}
6	Generalized regression	Узагальнена регресійна мережа	2	IW{1,1}, b{1}, LW{2,1}
7	Hopfield	Мережа Хопфилда	1	LW{1,1}, b{1}
8	Linear layer (design)	Лінійний шар (створення)	1	IW{1,1}, b{1}
9	Linear layer (train)	Лінійний шар (навчання)	1	IW{1,1}, b{1}
10	LVQ	Мережа для класифікації вхідних векторів	2	IW{1,1}, LW{2,1}
11	Perceptron	Персептрон	1	IW{1,1}, b{1}
12	Probabalistic	Імовірнісна мережа	2	IW{1,1}, b{1}, LW{2,1}
13	Radial basis (exact fit)	Радіальна базисна мережа з нульовою помилкою	2	IW{1,1}, b{1} LW{2,1}
14	Radial basis (fewer neurons)	Радіальна базисна мережа з мінімальним числом нейронів	2	IW{1,1}, b{1} LW{2,1}, b{2}
15	Self-organizing map	Самоорганізована мапа Кохонена	1	IW{1,1}

Примітки:

- позначення для шарів нейронної мережі: верхній індекс з двох символів застосовується для того, щоб вказати джерело сигналу (l) і пункт призначення (k); він використовується для позначення матриць ваг входу $IW^{k,l}$ і матриць ваг шару $LW^{k,l}$;
- для мереж 2, 3, 7 не забезпечується перегляд структурних схем;
- мережі 5, 9 допускають введення лінії затримання на вході;
- мережі 3 допускають введення лінії затримання в шарі;
- мережі з двома шарами мають послідовну структуру, коли вихід першого шару служить входом другого шару. Виняток становлять мережі 3, які допускають наявність зворотного зв'язку в першому шарі і передачу вхідного сигналу на входи обох шарів.

Перевага пакету MATLAB полягає в тому, що при його використанні користувач не обмежений моделями нейронних мереж і їх параметрами, жорстко закладеними в нейросимуляторі, а має можливість самостійно сконструювати ту мережу, яку вважає оптимальною для вирішення поставленого завдання.

2.3. MATLAB в умовах міжпроцесорної взаємодії

При створенні додатків засобами об'єктно-орієнтованого програмування доводиться приділяти велику увагу отриманню даних і генерації керуючих сигналів в реальному часі. Забезпечити відгук в реальному часі тим важче, чим швидше змінюється стан об'єктів. При тестуванні важливо не тільки отримання даних і генерація відгуку, часто доводиться будувати в реальному часі різні графіки. MATLAB надає потужні засоби для обробки графічної інформації. Звертаючись до функцій MATLAB, можна відносно просто виконувати складні математичні розрахунки і будувати графіки.

Завдяки застосуванню міжпроцесорного підходу вдається об'єднати керуючі модулі, що базуються на одній платформі, і засоби, які реалізують

математичні обчислення, які передбачають використання платформи MATLAB.

2.3.1. Міжпроцесорна взаємодія між MATLAB і Visual C++

У керуючих системах часто виникає задача обробки матриць, які формуються на основі даних, отриманих в результаті вимірювань. Організувати обробку матриць в середовищі C або VC++ досить важко. Данні можна перетворити в формат, придатний для обробки в MATLAB, однак при відсутності інтерфейсу між програмами не вдається здійснювати обробку в реальному часі.

Інтерфейс між VC++ та MATLAB дозволяє істотно підвищити рівень систем збору і аналізу даних. За допомогою програм, написаних на C або VC++, легко реалізувати низькорівневу взаємодію з апаратними засобами. Ці програми можуть використовуватися для передачі команд пристроїв і обміну даними з ними. При отриманні даних програми, написані на C або VC++, можуть звертатися до інструментальних засобів MATLAB, які, в свою чергу, виконають необхідну обробку даних. Результати обробки можна знову передати програмі на C або VC++, яка використовує їх для управління об'єктом.

Для створення незалежного додатка можна використовувати такі засоби:

- MATLAB Compiler;
- MATLAB C/C++ Math Library;
- MATLAB C/C++ Graphic Library.

MATLAB Compiler є основним інструментом розробки незалежних додатків. Він виступає в ролі інструменту, який дозволяє перетворювати файл M-коду у вихідний код C або C++. MATLAB C/C++ Math Library і MATLAB C/C++ Graphic Library є динамічно зв'язані бібліотеки. Вони надають розробникові вбудовані математичні і графічні функції MATLAB.

Використання засобів MATLAB в програмах на VC++ дозволяє також виконувати складні операції над багатовимірними матрицями. На рис. 2.3 представлений принцип взаємодії цих платформ. Перед зверненням до функцій MATLAB, призначеним для обробки матриць, програма, створена на VC++, повинна підготувати дані, зокрема зібрати інформацію, надану апаратними засобами, і створити файли даних в форматі ASCII. Після цього MATLAB отримує дані з файлів, підготовлених програмою на VC++, перетворює їх в властиві платформі типи, а потім виконує над ними операції. Результати передаються програмі на VC++ також через файли. Користуючись результатами, наданими MATLAB, програма на VC++ може здійснювати управління пристроями, підключеними до комп'ютера.

Рис. 2.3. Взаємодія засобів VC++ та MATLAB

Таким чином, обробка матриць здійснюється в два етапи. В першу чергу програма на VC++ готує дані (вони можуть бути згенеровані на апаратному рівні) і зберігає інформацію в файлах. Після цього виконання програми на VC++ призупиняється, а засоби MATLAB починають обробку даних, читаючи їх з файлів і виконуючи необхідні операції. Зрозуміло, що такий спосіб не забезпечує високої ефективності роботи. Зокрема, він не дозволяє управляти системами в реальному масштабі часу.

2.3.2. Міжпроцесорна взаємодія MATLAB і Java

Мова програмування Java використовується в різних областях, наприклад у виробництві, банківській сфері, комерційної діяльності та наукових дослідженнях. Java добре підходить для створення графічних користувацьких інтерфейсів і, що більш важливо, є об'єктно-орієнтованою мовою. Програми на Java транслюються в байт-код, що виконується віртуальною машиною Java (JVM) – програмою, яка обробляє байтовий код і передає інструкції обладнанню як інтерпретатор. Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина.

Ряд особливостей платформи безпосередньо впливає на швидкість виконання програм написаних на мові Java:

- застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,
- широке використання платформенно-орієнтованого коду (native-код) в стандартних бібліотеках,
- апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, що підтримується деякими процесорами фірми ARM).

Що є важливим для роботи в міжпроцесорній взаємодії.

Швидкий розвиток Java уможливило використання цієї мови для обміну даними між різними клієнтами мережі. Необхідність передавати мережею результати вимірювань і обробки даних виникає при вирішенні дуже багатьох завдань. MATLAB являє собою потужний інструмент аналізу даних в складних обчислювальних системах. Таким чином, організувавши взаємодію Java і MATLAB, можливо реалізовувати програми, які призначені для виконання складних дій. Для організації взаємодії Java і MATLAB

використовують динамічну бібліотеку JMatLink і утиліту MATLAB Builder JA.

Засоби JMatLink реалізовані у вигляді підкласу класу java.lang.Thread. Всі функції цієї бібліотеки використовують метод Matlab Engine або метод ActiveX. У програмі на Java функції, призначені для взаємодії з MATLAB, викликаються в окремих потоках. Синхронізація між потоками здійснюється за допомогою механізму очікування.

Всі функції JMatLink представляють собою переносні незалежні методи. Вони підтримуються в системах сімейства Windows і на різних платформах Unix. При перенесенні з однієї платформи на іншу змінювати вихідний код не потрібно.

MATLAB Builder JA дозволяє створювати класи Java з програм MATLAB. Ці Java-класи можуть бути інтегровані в програми Java і розгорнуті безпосередньо на настільних комп'ютерах або веб-серверах, які не мають встановленого MATLAB, за допомогою MATLAB Compiler Runtime (MCR), що поставляється разом з MATLAB Compiler.

При використанні спільно з MATLAB Compiler, утиліта створює компоненти, що розгортаються і роблять обчислення на основі MATLAB, візуалізацію даних, і реалізують графічні інтерфейси, доступні для кінцевих користувачів програм Java. Builder JA шифрує функції MATLAB і формує навколо них оболонку Java, створюючи імітацію стандартного класу Java. Java-класи, створені за допомогою MATLAB Builder JA, є портативними і можуть працювати на всіх платформах, що підтримуються MATLAB.

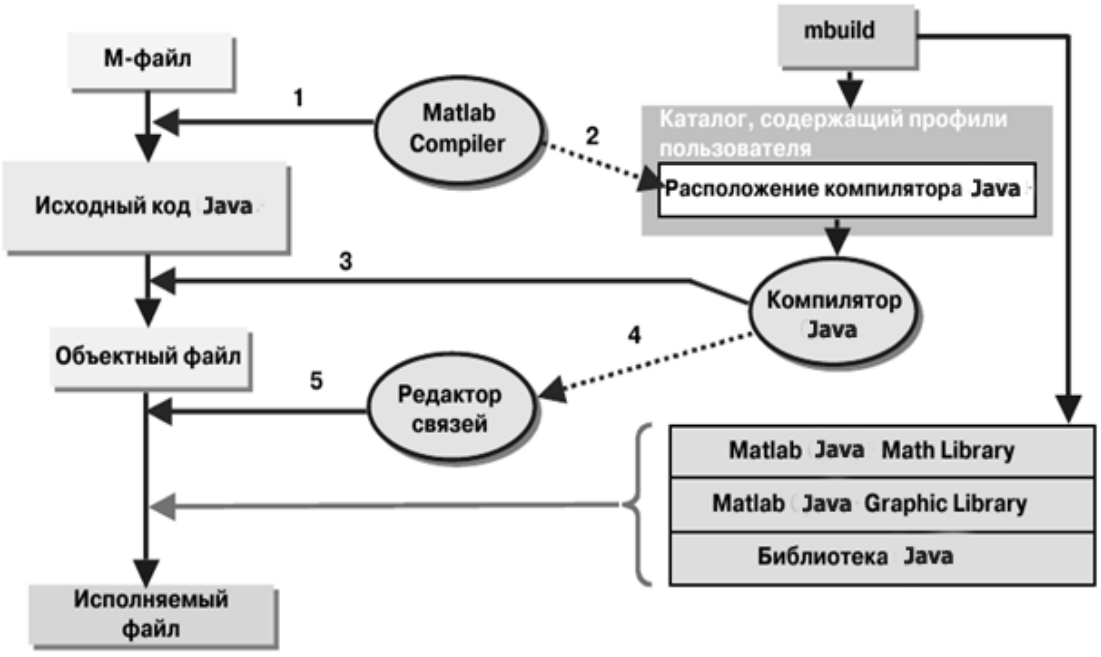
Ключовими особливостями MATLAB Builder JA є:

- розгортання класів Java для настільних комп'ютерів або web-сервісів;
- зміна масштабу, поворот і панорамне уявлення графіків MATLAB за допомогою інтерфейсу Web Figure;
- можливість портування класів, що не містять MEX-файли, для всіх підтримуваних платформ MATLAB;

- API для автоматичного перетворення між типами даних Java і MATLAB.

Основні етапи розробки незалежного Java-додатка показані на рис. 2.4:

1. MATLAB Builder JA викликається для перетворення М-коду у вихідний код Java;
2. MATLAB Compiler викликає компілятор Java, використовуючи дані про його розташування;
3. Компілятор Java перетворює вихідний текст програм в об'єктний код;
4. Після закінчення перетворення коду компілятор Java викликає редактор зв'язків;
5. Редактор зв'язків виконує компоновку, підключаючи до об'єктного коду необхідні бібліотеки, і генерує файл програми, що виконується.



Мал. 2.4. Основні етапи створення незалежного додатка

Виходячи з вимоги інтегрованості і кросплатформеності, для реалізації основних модулів програмного забезпечення структурно-параметричної ідентифікації та написання графічного інтерфейсу була вибрана мова програмування Java.

Висновки розділу 2

Для розв'язання задачі структурно-параметричної ідентифікації процесів крупного дроблення руд доцільно використовувати нейронну мережу. НМ дозволяють ефективно вирішувати багато завдань, і надають потужні гнучкі і універсальні механізми навчання, що є їх головною перевагою перед іншими методами. Також слід зазначити універсальність нейромережевих методів щодо вхідних даних.

Підхід міжпроцесорної взаємодії систем комп'ютерної алгебри та об'єктно-орієнтованих мов програмування дозволяє істотно розширити можливості систем управління складними ОУ і забезпечити виконання складних математичних обчислень в рамках подібних додатків. Завдяки застосуванню міжпроцесорного підходу пропонується об'єднати керуючі модулі програмного забезпечення структурно-параметричної ідентифікації, що базуються на платформі Java, і засоби, які реалізують математичні обчислення, що використовують платформу MATLAB.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТРУКТУРНО-ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ПРОЦЕСІВ КРУПНОГО ДРОБЛЕННЯ РУД

3.1. Методика структурно-параметричної ідентифікації ОУ

Ідентифікація процесу, як динамічної системи, полягає в отриманні або уточненні за експериментальними даними математичної моделі цього процесу, вираженої за допомогою того чи іншого математичного апарату [34]. Передбачається вивчення і зіставлення вхідних і вихідних процесів, так як завдання полягає у виборі відповідної математичної моделі. Модель повинна бути такою, що її реакція і реакція об'єкта на один і той же вхідний сигнал повинні бути, в даному випадку, близькими. Результати рішення задачі ідентифікації є вихідними даними для проектування процесів.

Структурно-параметрична ідентифікації включає в себе виділення об'єкта з навколишнього середовища, що взаємодіє з ним. а також оцінку і оптимізацію параметрів моделі ОУ. Перші дві операції вирішуються шляхом генерування (за допомогою базисних функцій) моделей-претендентів різної складності і настройки їх параметрів з подальшою селекцією кращих з них за обраними критеріями (оптимальна структура). Операція визначення оптимальних параметрів вирішується відомими методами оптимізації, шляхом уточнення отриманих раніше значень параметрів за критеріями регулярності на всій вибірці вихідних даних [9].

Для реалізації розглянутих вище підходів в роботі [35] запропоновано композиційний метод структурно-параметричної ідентифікації ОУ, який полягає в:

а) формуванні завдання ідентифікації з вибором методів і критеріїв структурної (глобальної) оптимізації, способів врахування обмежень, типу структури моделі, базисних функцій та методів параметричної оптимізації;

b) ідентифікації структури моделі ОУ на базі композиції методів глобальної оптимізації, які містять генерування структур моделей-претендентів (базисних функцій), і методів локальної оптимізації для параметричного навчання базисних функцій, а також селекцію кращих моделей за критеріями структурної оптимізації;

c) ідентифікації параметрів моделі оптимальної структури шляхом її навчання методами локальної параметричної оптимізації за критеріями регулярності на всій вибірці даних.

Композиційний метод ідентифікації динамічних представлений на рис. 3.1.

Рис. 3.1. Методика ідентифікації динамічного процесу.

Дана методика складається з етапів оцінки стану і характеристик ОУ і його структурно-параметричної ідентифікації.

Оцінка стану і характеристик ОУ включає:

- процедуру визначення режиму функціонування ОУ, що містить якісний аналіз виду часового сигналу, його спектра, кореляційної функції і вейвлет перетворення;
- обчислення кореляційної ентропії, яка є оцінкою ентропії Колмогорова і характеризує режим функціонування об'єкта управління [1];

- процедуру оцінки тимчасових характеристик ОУ, що містить обчислення кореляційного інтервалу передбачуваності (глибини прогнозу) процесу, який є оцінкою інтервалу точного прогнозування стану ОУ [1];
- визначення еквівалентного часу запізнювання в ОУ по взаємнокореляційним функціям каналів ОУ;
- вибір періоду дискретизації за значенням кореляційного інтервалу передбачуваності і інтервалу кореляції;
- обчислення кореляційної розмірності атрактора, яка характеризує розмірність Хаусдорфа [1];
- визначення розмірності вкладення атрактора (розмірності фазового простору – глибини пам'яті) ОУ по кореляційної розмірності атрактора;
- формування оцінки вектора стану ОУ.

Структурно-параметрична ідентифікація ОУ включає:

- формування завдання ідентифікації, яке здійснюється з вибором видів критеріїв структурної і параметричної ідентифікації, базисних функцій, методу структурної (глобальної) оптимізації та методів параметричної оптимізації, які визначаються вибором базисних функцій. Вищеописаний вибір здійснюється на етапі проектування системи ідентифікації процесів крупного дроблення руд і базується на апріорних (теоретичних і експериментальних) даних;
- структурну ідентифікацію, яка здійснюється за допомогою композиції методів глобальної і локальної оптимізації шляхом генерування структур моделей-претендентів (базисних функцій зі своїми структурними характеристиками) і селекцію кращих з них за критерієм структурної оптимізації;

- параметричну ідентифікацію, яка полягає у визначенні параметрів моделі оптимальної структури шляхом її навчання методом локальної параметричної оптимізації за критерієм регулярності.

3.2. Алгоритм структурно-параметричної ідентифікації процесів крупного дроблення руд

Критерієм параметричної ідентифікації моделі ОУ є величина помилки $J_{рег}$ (критерій регулярності). Її фізичний зміст полягає в тому, що вона визначає різницю між значеннями виходу моделі ОУ оптимальної структури і реального виходу ОУ. Таким чином, в реальному часі відбувається перевизначення (адаптація) оптимальних параметрів моделі ОУ оптимальної структури (рис. 3.2).

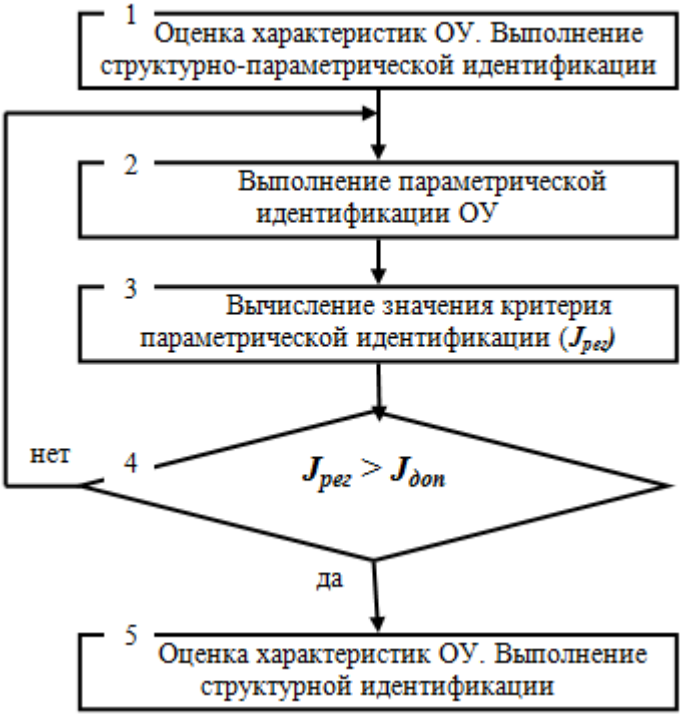


Рис. 3.2. Алгоритм структурної і параметричної ідентифікації.

Якщо значення помилки $J_{рег}$ для оптимальних значень параметрів виявляється більше допустимої ($J_{рег} > J_{доп}$), То це означає, що визначена

раніше оптимальна структура моделі ОУ вже не є такою (наприклад, через зміну режиму функціонування ОУ). Тому помилку доцільно враховувати при прийнятті рішення про необхідність переходу до оцінки характеристик ОУ і виконання його структурної ідентифікації (див. рис. 3.2). При цьому кроки 1 і 5 алгоритму на рис. 3.2 виконуються в автоматизованому режимі, а кроки 2-4 – в автоматичному.

Узагальнена схема структурної ідентифікації з використанням методів глобальної оптимізації: прямого випадкового пошуку (ПВП) і генетичного алгоритму (ГА) [19, 34] і базисних функцій у вигляді нейронних мереж (НМ) прямого поширення (НМПП), НМ з радіальними базисними функціями (РБФ), а також НМ нечіткого виведення Anfis [21] приведена на рис. 3.3.

Рис. 3.3. Схема алгоритму структурної ідентифікації.

Алгоритм глобальної оптимізації з використанням методу ПВП здійснюється наступним чином:

Крок 1. Формування початкових точок. Виконується визначення структурних характеристик моделі: типу базисної функції, кількості нейронів в прихованому шарі, типу функцій активації (приналежності) прихованого шару, а також типу алгоритму параметричної оптимізації. Кожна точка являє собою вектор структурних характеристик. При цьому налаштовуються опції для роботи ПВП: умови зупинки алгоритму ПВП, вибір методів, способів і властивостей пошуку.

Крок 2. Обчислення значення критерію структурної оптимізації для початкових умов. За допомогою алгоритмів локальної оптимізації базисних функцій визначаються значення виходу моделі ОУ, за якими обчислюються значення критерію структурної оптимізації.

Крок 3. Виконується умова зупинки алгоритму. Якщо так, то здійснюється перехід до кроку 7, інакше - перехід до кроку 4. При цьому, умови зупинки ПВП наступні:

- число ітерацій досягає заданого максимального значення;
- розмір осередку (точки) менше заданого значення;
- загальне число розрахунків значень цільової функції досягло максимально заданого значення;
- відстань між точкою, знайденої в успішному опитуванні, і точкою, знайденому в невдалому опитуванні буде менше заданого значення;
- зміни в цільовій функції для одного успішного опитування і подальшого успішного опитування будуть менше заданої величини.

Крок 4. Генерування нових точок відбувається за допомогою методів пошуку, налаштованих за крок 1.

Крок 5. Обчислення значення критерію для нових точок відбувається аналогічно обчисленням на кроці 2.

Крок 6. Селекція точок з кращими значеннями критерію, які заносяться в нову послідовність пошуку. Далі виконується перехід до кроку 3.

Крок 7. Формування моделі оптимальної структури. За допомогою алгоритму ПВП визначаються значення кращих точок (структурних характеристик), відповідних екстремуму критерію. Формується модель оптимальної структури.

Алгоритм глобальної оптимізації з використанням ГА здійснюється аналогічно ПВП:

Крок 1. Формування початкових особин. Кожна особина є вектор з елементами у вигляді структурних характеристик. На цьому етапі здійснюється настройка опцій ГА: умови зупинки, розміру популяції, а також формування генетичних операторів (мутації, відбору, кросинговеру, міграції та ін.)

Крок 2. Обчислення значення критерію структурної оптимізації для початкових умов.

Крок 3. Тут умови зупинки ГА наступні:

- число поколінь досягає заданого значення;
- минув заданий час роботи алгоритму;
- значення критерію кращої особини поточної популяції досягає екстремуму;
- немає поліпшення критерію в послідовності наступних один за одним поколінь, довжина якої задана на кроці 2;
- немає поліпшення для критерію протягом заданого інтервалу часу.

Крок 4. Генерування нових особин відбувається за допомогою генетичних операторів, налаштованих на кроці 1. Спочатку відбувається відбір батьків, потім кроссинговер батьків, схрещування і імовірнісна мутація нащадків. Тут також встановлюється частка мігруючих особин з однієї популяції в іншу.

Крок 5. Обчислення значення критерію для нових особин.

Крок 6. Селекція особин з найкращими значеннями критерію в нову популяцію.

Крок 7. За допомогою ГА визначаються значення кращих особин (структурних характеристик), відповідних екстремуму критерію. Формується модель оптимальної структури.

Узагальнена схема алгоритму параметричної ідентифікації з використанням базисних функцій у вигляді НМПП, НМ РБФ і НМ Anfis приведена на рис. 3.4.

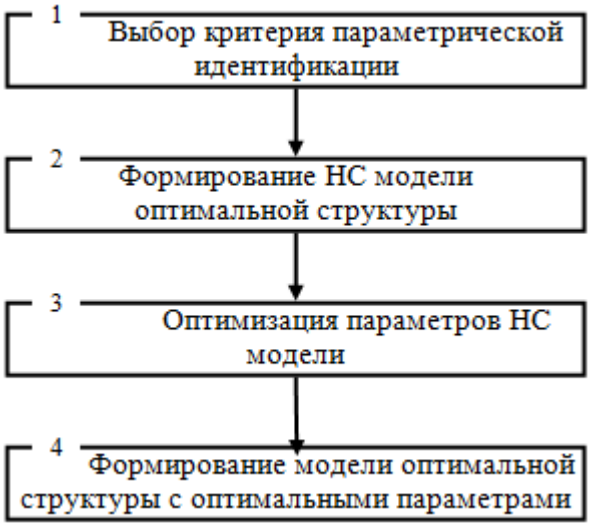


Рис. 3.4. Схема алгоритму параметричної ідентифікації.

Алгоритм локальної оптимізації з використанням в якості базисної функції НМПП здійснюється наступним чином:

Крок 1. Вибір виду критерію параметричної ідентифікації.

Крок 2. Формування моделі оптимальної структури. Створення НМПП з наступними структурними характеристиками: глибинами пам'яті, розміром прихованого шару, функцією активації нейронів прихованого і вихідного шару, а також методом параметричної оптимізації. Виконується настройка опцій для навчання НМ: умов зупинки навчання, формування масивів навчальних і перевірочних даних, функцій одновимірного пошуку та ін.

Крок 3. Оптимізація параметрів моделі (навчання НМ) шляхом визначення значень вагових коефіцієнтів відповідно до обраного на кроці 2 методом параметричної оптимізації. Умови зупинки навчання:

- кількість циклів (епох) навчання досягло максимально заданого значення;
- помилка зменшилася до заданого рівня;
- час навчання досяг максимально заданого значення;
- рівень перевищення помилки перевірочної вибірки в порівнянні з навчальною досяг максимально заданого значення.

Крок 4. Формування моделі оптимальної структури з оптимальними значеннями параметрів, відповідних мінімуму критерію параметричної оптимізації (вагових коефіцієнтів зв'язку вхідного і вихідного шарів.

Алгоритм локальної оптимізації з використанням в якості базисної функції НМ РБФ здійснюється аналогічно НМПП:

Крок 1. Вибір критерію параметричної ідентифікації.

Крок 2. Формування моделі оптимальної структури. Вибір структурних характеристик НМ РБФ: глибин пам'яті, максимального розміру прихованого шару, функції активації нейронів прихованого (РБФ) шару, функції активації нейронів вихідного шару, а також методу параметричної оптимізації. Виконується настройка опцій для навчання НМ: умов зупинки навчання, вибір значення відхилення (параметра впливу) і ін.

Крок 3. Умови зупинки навчання:

- помилка зменшилася до заданого рівня;
- кількість нейронів прихованого шару досягло максимально заданого значення.

Крок 4. Формування моделі оптимальної структури з оптимальними значеннями параметрів РБФ нейронів прихованого шару і вагових коефіцієнтів вихідного шару.

Алгоритм локальної оптимізації з використанням базисної функції НМ Anfis здійснюється наступним чином (див. Рис. 3.4):

Крок 1. Вибір критерію параметричної ідентифікації.

Крок 2. Генерація структури типу Sugeno з вибором наступних структурних характеристик: глибин пам'яті, функцій приналежності і кількості правил розкладання по входах.

Крок 3. Умови зупинки навчання:

- кількість циклів (епох) навчання досягло максимально заданого значення;
- помилка зменшилася до заданого рівня.

Крок 4. Формування моделі оптимальної структури з оптимальними значеннями параметрів функцій приналежності.

3.3. Логічна структура програмного продукту

Логічна структура додатку, що розробляється, містить набір функціонально-логічних модулів, що включають процедури і об'єкти, що представляють собою форми, вікна для заповнення вхідних даних, форми проміжних даних, графіків, а також вікна результатів і звітів.

При розробці додатка ідентифікації були дотримані базові принципи методології структурного підходу, що є принципами декомпозиції, згідно з якими при проектуванні програмного забезпечення здійснюється функціональна декомпозиція відповідних підсистеми і модулів, що виконують логічно різні функції.

Розроблене програмне забезпечення структурно-параметричної ідентифікації складається з наступних функціонально-логічних модулів:

1. **Math4JavaApp** – клас ініціалізації програми. Він містить ключовий метод *main(...),* який викликається JVM при першому зверненні до додатка. У цьому методі створюється головна форма, що відповідає за створення і відображення вікна введення вхідних параметрів для ідентифікації процесу.

2. **MainWindow** – клас для введення параметрів користувача, на основі яких виконується ідентифікація. При створенні екземпляра даного класу створюється вікно інтерфейсу користувача і контролери, які в подальшому використовуються для формування моделі і її передачі в міжпроцесорну обробку до MATLAB. Цей клас включає в себе такі методи:
    a) *initComponents(...)* – метод, створює та ініціалізує компоненти для інтерфейсу користувача;
    b) *collectData(...)* – метод, який формує об'єкт моделі, заснований на введених параметрах користувача, яка буде передана в міжпроцесорну обробку.
3. **Controller** – клас, відповідальний за створення міжпроцесорного «містка» і передачі по ньому моделі наданих даних.
4. **ResultWindow** – клас відповідальний за відображення звітів і результатів виконання ідентифікації. При завантаженні даного класу виконується ініціалізація інтерфейсу користувача для формування звіту. Звіт формується на даних, отриманих від MATLAB, після обчислень. ResultWindow містить такі методи:
    a) *initComponents(...)* – метод, створює та ініціалізує компоненти для інтерфейсу користувача;
    b) *setData(Object[] objects)* – цей метод приймає масив MATLAB структур, які містять в собі числові результати ідентифікації.

На додаток до цих методів, для зручності читання коду, було створено кілька вузьконаправлених допоміжних методів, які відповідають за розподіл даних в звіті. Так як код оперує досить абстрактними структурами, ці методи не дозволяють заплутатися у великих масивах однотипних даних.

3.4. Особливості програмної реалізації алгоритму

Алгоритм (рис. 3.5), є абстрактним поданням лінійного виконання процесу ідентифікації в створеному додатку. У ньому відображені основні кроки виконання поставленого завдання, а це отримання, формування та передача параметрів користувача, їх обробка та відображення результатів.

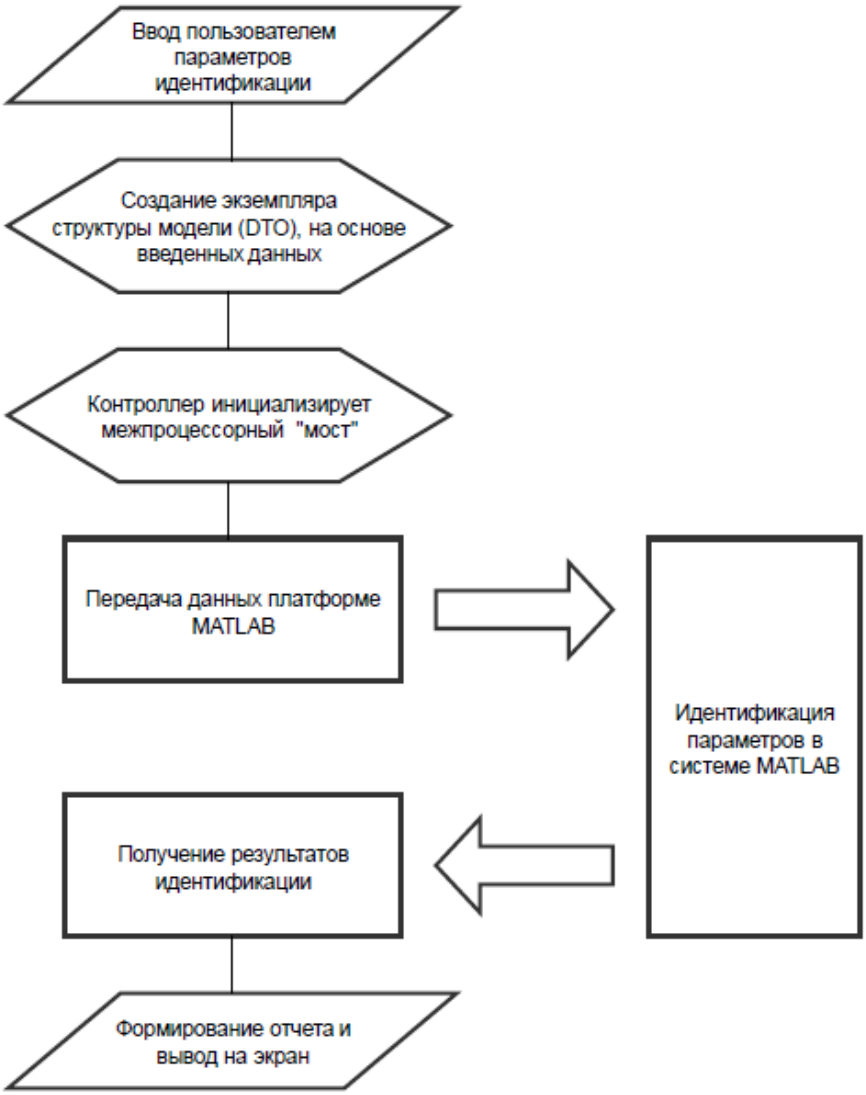


Рис. 3.5. Алгоритм повного циклу виконання ідентифікації.

Після введення користувачем параметрів, насамперед створюється модельний об'єкт, який компонує в собі всі ці параметри.

```
private MathModel collectData(NetworkArchitectureTypeEnum architectureType) {
    MathModel model = new MathModel();

    model.setArchitecture(networkArchitectureComboBox.getSelectedIndex()+1);
    model.setCount(Integer.valueOf(numberOfNeuronsTextField.getText()));
    model.setCriterion(criterionTypeComboBox.getSelectedIndex()+1);
    model.setMethod(optimizationMethodComboBox.getSelectedIndex()+1);
    model.setModelInd(identifiableObjectModelComboBox.getSelectedIndex());
    model.setOutData(outDataComboBox.getSelectedIndex());

    switch (architectureType) {
        case RBF:
           model.setTypeAlgorithm(0);
           model.setTypeFunc(0);
           break;
        default:
           model.setTypeAlgorithm(Integer.parseInt(
              learningAlgorithmTrFcnComboBox.getSelectedItem().toString()));
           model.setTypeFunc(Integer.parseInt(
              activationFunctionTFComboBox.getSelectedItem().toString()));
           break;
    }
    return model;
}
```

Для створення екземпляра класу JavaToMatLab Bridge, призначеного для взаємодії з М-функціями, був написаний контролер. У нього і передається модельний об'єкт.

```
...
Try {
    NetworkArchitectureTypeEnum networkArchitectureTypeEnum =
    networkArchitectureTypeMapper.map(
        networkArchitectureComboBox.getSelectedIndex());

    MathModel model = collectData(networkArchitectureTypeEnum);
    Object [] ob = controller.generate(model);

    new ResultWindow(ob);
}
catch (Exception e) {
    // handle exception
}
...
```

Після виконання коду MATLAB і отримання даних оптимізації з методу *generate(...),* дані передаються класу ResultWindow, відповідальному за формування результатів. Звіт будується за допомогою стандартних форм Java. Розподіл даних між допоміжними методами представлено нижче.

```
private void setData(Object[] objects) {
    if (objects != null) {
       double[] struct = ((MWNumericArray) objects[0]).GetDoubleData();
       jTextField1.setText(String.format("%.4f", struct[0]));
       jTextField2.setText(String.format("%.4f", struct[1]));
       jTextField3.setText(String.format("%.4f", struct[2]));
```

```
        activationFunctionTypeTextField.setText(
              String.format("%.4f", struct[3]));

        double opt = ((MWNumericArray) objects[1]).GetDouble();
        structuralOptimizationCriterionTextField.setText(
              String.format("%.4f", opt));

        double[] population = ((MWNumericArray) objects[4]).getDoubleData();
        setPopulation(population);

        double[] scores = ((MWNumericArray) objects[5]).GetDoubleData();
        setScores(scores);
    }
}
```

Наступні фрагменти коду ілюструють процес передачі даних в М-функцію, реалізований в методі *identify(...)* класу JavaToMatLab Bridge :

```
public void identify(Object [] lhs, Object [] rhs) throws MWException
{
    fMCR.invoke(Arrays.asList(lhs), Arrays.asList(rhs), disserSignature);
}
```

Масив rhs містить вхідні дані. Кількість вхідних параметрів визначається розміром виділеного масиву. Вхідні аргументи повинні передаватися як підкласи класу com.mathworks.toolbox.javabuilder.MWArray, або як масиви будь-якого типу, підтримуваного Java. Аргументи, що передаються як значення типів Java, конвертуються в масиви MATLAB відповідно до встановлених правил перетворення.

```
public Object[] identify(int nargout, Object... rhs) throws MWException
{
    Object[] lhs = new Object[nargout];
    fMCR.invoke(Arrays.asList(lhs),
         MWMCR.getRhsCompat(rhs, disserSignature), disserSignature);
    return lhs;
}
```

Параметр nargout містить число аргументів,що повертаються. Параметр rhs містить вхідні дані для передачі в М-функцію. Масив розміру nargout містить вихідні дані функції. Вихідні значення повертаються як підкласи класу com.mathworks.toolbox.javabuilder.MWArray. Кожен массив, що повертається, повинен звільнятися з допомогою виклику відповідного методу *dispose(...).*

3.5. Опис інтерфейсу програми

Після запуску програми на екрані з'являється форма, представлена на рис. 3.6. Вікно структурно-параметричної ідентифікації містить перелік полів для заповнення, в яких необхідно вказати:

1. метод оптимізації;
2. базисну функцію (архітектуру мережі). При виборі базисної функції стандартні дані вводяться автоматично і далі їх можна змінити;
3. кількість нейронів в прихованому шарі;
4. функції активації;
5. алгоритм навчання;
6. вихідні дані про об'єкт;
7. модель ідентифікованого об'єкта;
8. тип критерію.

Після того, як всі вихідні дані введені, потрібно натиснути кнопку «Виконати» і почнеться процес ідентифікації.

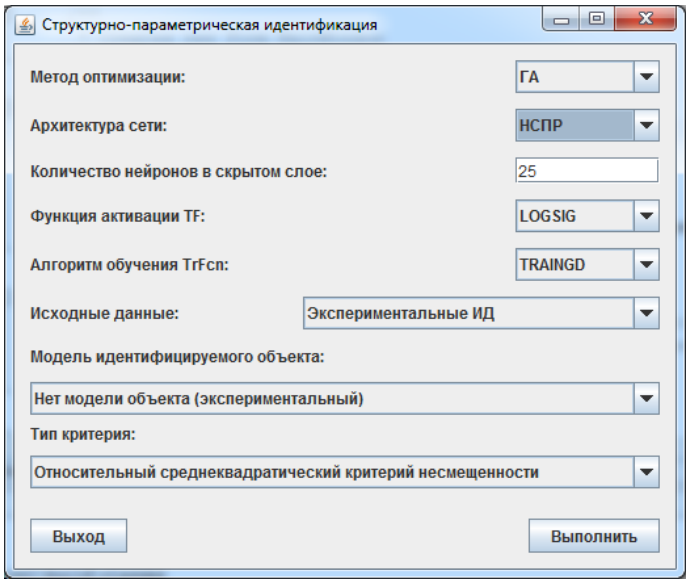


Рис. 3.6. Головне вікно програми структурно-параметричної ідентифікації.

Весь процес ідентифікації займає певний час. За прогресом виконання можна спостерігати за допомогою графіка, який відображається в додатковому вікні програми (рис. 3.7). Воно з'являється з початком виконання ідентифікації.

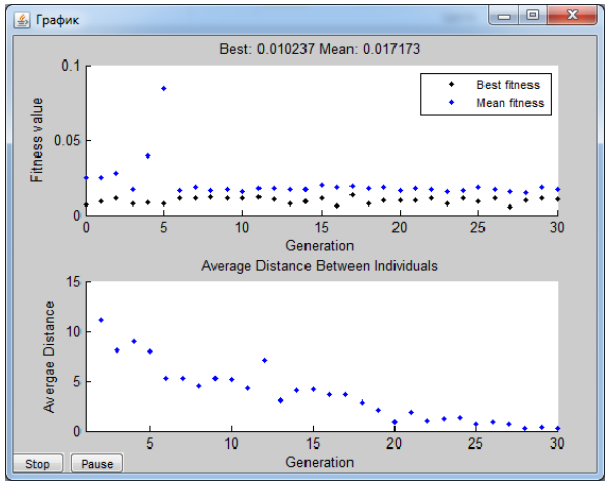


Рис. 3.7. Вікно графіка, що відображає процес ідентифікації.

По закінченню роботи програми на екрані з’явиться вікно результатів (рис. 3.8), в лівій частині якого знаходяться результати роботи програми, а в правій – результати генетичного алгоритму. Результати роботи являють собою структурні характеристики моделі і критерії структурної оптимізації за підсумками виконаної ідентифікації.

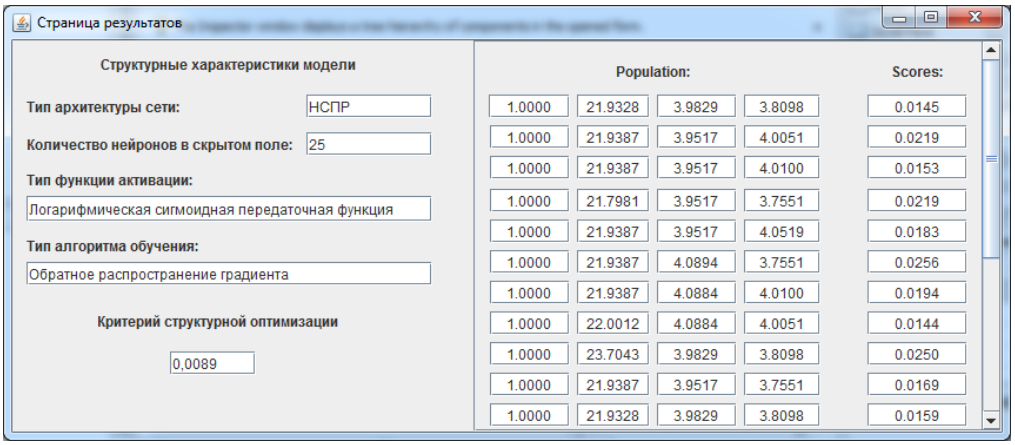


Рис. 3.8. Вікно результатів програми структурно-параметричної ідентифікації.

Висновки розділу 3

Запропоновано методику ідентифікації процесів крупного дроблення руд, яка складається з процедур оцінки стану і характеристик процесу і його структурно-параметричної ідентифікації. Розроблено алгоритми глобальної і локальної оптимізації моделі ідентифікації процесів крупного дроблення руд, які реалізують процедуру структурно-параметричної ідентифікації шляхом їх структурної та параметричної оптимізації, що дозволяє отримувати моделі підвищеної точності.

Розроблено програмне забезпечення для виконання структурно-параметричної ідентифікації процесів крупного дроблення руд, що реалізує запропоновану методику.

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Розрахунок трудомісткості і вартості розробки програмного продукту

Вхідні данні:

Q – передбачуване число операторів (1750);

q – коефіцієнт складності програми (1.2);

p – коефіцієнт корекції програми в ході її розробки (0.05).

$C_{пр}$ – середня годинна заробітна плата програміста з нарахуваннями(25).

$C_{мч}$ – вартість машинного часу ЕОМ (11 грн/год).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_О + t_И + t_А + t_П + t_{ОТ} + t_Д, \text{ чол.-г} \qquad (4.1)$$

де:

$t_О$ – витрати праці на підготовку й опис поставленої задачі (приймається 30);

$t_И$ – витрати праці на дослідження алгоритму рішення задачі;

$t_А$ – витрати праці на розробку блок-схеми алгоритму;

$t_П$ – витрати праці на програмування по готовій блок-схемі;

$t_{ОТ}$ – витрати праці на налагодження програми на ЕОМ;

$t_Д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється. В їх число входять ті оператори, які необхідно написати в процесі роботи над програмою з урахуванням можливих уточнень і вдосконалення алгоритму.

Умовне число операторів (підпрограм) обчислюється за формулою:

$$Q = q \times C \times (1 + p)\,, \qquad (4.2)$$

де:

Q – передбачуване число операторів (1750);

q – коефіцієнт складності програми (1.2);

p – коефіцієнт кореляції програми в ході її розробки (0.05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}2 \times 1750 \times (1 + 0{,}05) = 2205$$

Затрати праці на вивчення опису задачі $t_И$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_И = \frac{Q \times B}{80 \times K}\,, \text{ чол.-г} \qquad (4.3)$$

де:

B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років, він складає 1.2.

Приймемо збільшення витрат праці унаслідок недостатнього опису завдання не більше 50% (B = 1.5). З урахуванням коефіцієнта кваліфікації K = 1.2 отримуємо витрати праці на вивчення опису завдання:

$$t_И = \frac{2205 \times 1{,}5}{75 \times 1{,}2} = 37 \text{ чол.-г}$$

Витрати праці на розробку алгоритму розв'язання задачі визначаються за формулою:

$$t_A = \frac{Q}{20 \times K}\,, \text{ чол.-г} \qquad (4.4)$$

де: Q – умовне число операторів в програмі;

K – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення у формулу (4.4), отримаємо:

$$t_A = \frac{2205}{20 \times 1{,}2} = 92 \text{ чол.-г}$$

Витрати на складання програми по готовій блок-схемі

$$t_П = \frac{Q}{25 \times K}, \text{ чол.-г} \quad (4.5)$$

$$t_{ОТ} = \frac{2205}{25 \times 1{,}2} = 74 \text{ чол.-г}$$

Витрати праці на налагодження програми на ЕОМ складуть

$$t_{ОТ} = \frac{Q}{5 \times K}, \text{ чол.-г} \quad (4.6)$$

$$t_{ОТ} = \frac{2205}{5 \times 1{,}2} = 368 \text{ чол.-г}$$

З урахуванням коефіцієнта запасу 1.5 отримаємо

$$t^К_{ОТ} = 1{,}5 \times t_{ОТ}, \text{ чол.-г} \quad (4.7)$$

$$t^К_{ОТ} = 1{,}5 \times 368 = 552 \text{ чол.-г}$$

Витрати на підготовку документації визначаються за формулою

$$t_Д = t_{ДР} + t_{ДО}, \text{ чол.-г} \quad (4.8)$$

де:

$t_{ДР}$ – трудомісткість підготовки матеріалів та рукописи;

$t_{ДО}$ – трудомісткість редагування, друку та оформлення документації.

$$t_{ДР} = \frac{Q}{18 \times K}, \text{ чол.-г} \quad (4.9)$$

$$t_{ДО} = 0{,}75 \times t_{ДР}, \text{ чол.-г} \quad (4.10)$$

Підставляючи відповідні значення, отримаємо:

$$t_{ДР} = \frac{2205}{18 \times 1{,}2} = 101 \text{ чол.-г}$$

$$t_{ДО} = 0{,}75 \times 101 = 76 \text{ чол.-г}$$

$$t_Д = 101 + 76 = 177 \text{ чол.-г}$$

Повертаючись до формули (4.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 30 + 37 + 92 + 74 + 368 + 177 = 808 \text{ чол.-г}$$

4.2. Затрати на створення програмного забезпечення

Витрати на створення програмного забезпечення ($К_{ПО}$) включають витрати на заробітну плату розробників програми ($З_{зп}$), визначену множенням сумарної трудомісткості розробки ПЗ (t) на середню зарплату з нарахуваннями програміста та вартість машинного часу, необхідного для налагодження програми на ЕОМ ($З_{мв}$):

$$К_{ПО} = З_{зп} + З_{зм}, \text{ грн} \quad (4.11)$$

Заробітна плата розробників визначається за формулою:

$$З_{зп} = t \times С_{ПР}, \text{ грн} \quad (4.12)$$

де:

t – загальна трудомісткість розробки програми, чол.-г;

$С_{ПР}$ – середня годинна заробітна плата програміста з нарахуваннями.

З урахуванням того, що середня годинна зарплата програміста становить 25 грн/год, отримуємо:

$$З_{зп} = 808 \times 25 = 20\,200 \text{ грн} \quad (4.13)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{зм} = t_{от} \times С_{мч}, \text{ грн} \quad (4.14)$$

де:

$t_{от}$ – трудомісткість налагодження програми на ЕОМ, час;

$С_{мч}$ – вартість машинного часу ЕОМ (11 грн/год).

Підставивши у формулу відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{зм} = 368 \times 11 = 4048 \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 20\,200 + 4048 = 24\,248 \text{ грн}$$

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення додатку. Очікуваний період розробки програмного забезпечення:

$$T = \frac{t}{B_k \times F_p}, \text{ мес.} \qquad (4.15)$$

де:

B_k – число розробників (дорівнює 1),

F_p – місячний фонд робочого часу (при 40-годинному робочому тижні F_p = 176 годин).

Підставивши відповідні значення, отримаємо:

$$T = \frac{808}{1 \times 176} \approx 5 \text{ мес.}$$

4.3. Маркетингове дослідження ринку збуту розробленого програмного продукту

В даному дипломі розглянута розробка програмного забезпечення структурно-параметричної процесів крупного дроблення руд. В ході дослідження своєї теми, я прийшов до висновку, що подібного програмного продукту який реалізує виключно функціонал забезпечення структурно-параметричної ідентифікації процесів крупного дроблення руд не існує. Але існують програмні продукти, в функціонал яких входить розрахунок структурно-параметричної ідентифікації нелінійних динамічних процесом, з них, такі гіганти як:

1. MATLAB. Пакет прикладних програм для вирішення задач технічних обчислень і однойменний мову програмування, що використовується в цьому пакеті. MATLAB використовують більше 1 000 000 інженерних і наукових працівників, він працює на більшості сучасних операційних систем, включаючи Linux, Mac OS, Solaris

2. Microsoft Excel. Програма для роботи з електронними таблицями, створена корпорацією Microsoft для Microsoft Windows, Windows NT і Mac OS, Android А також, ОС IOS і Windows Phone. Вона надає можливості економіко-статистичних розрахунків, графічні інструменти і, за винятком Excel 2008 під Mac OS X, мова макропрограмування VBA (Visual Basic для додатка). Microsoft Excel входить до складу Microsoft Офіс і на сьогоднішній день Excel є одним з найбільш популярних додатків в світі.

Для порівняння я взяв продукт компанії MathWorks – MATLAB. За рахунок свого величезного функціоналу, при запуску доведеться трохи почекати, інтерфейс досить старий, 2000-х років, розробники активно працює над оновленнями самого функціоналу, а не інтерфейс, останнє оновлення було в березні 2006 року. Вартість ліцензії складає $2150

На даному етапі формування інформаційного ринку гостро відчувається відсутність науково обґрунтованої концепції та аргументованих підходів до системи купівлі-продажу таких інформаційних продуктів, як програмні засоби.

Аналіз існуючого ринку показує, що більшість виробників спеціалізується на декількох напрямах діяльності. Для країн з «перехідною економікою», зокрема для України, характерне прагнення багатьох підприємців працювати з різними товарами на будь-яких ринках через дефіцитність багатьох товарів і послуг. Це дає змогу успішно діяти на тому чи іншому ринку, навіть не знаючи добре особливостей самого ринку, специфіки товару і покупців, перспектив розвитку і т. п. В умовах ринку його вибір є ключовим.

Однак будь-який учасник ринку повинен знати: яке цільове призначення ринку; з кого і з чого складається ринок; які правила діють на ринку; які обмеження накладає ринок на бізнес; які можливості надає бізнесу ринок тощо.

Основними характеристиками будь-якого ринку є його місткість і насиченість товаром. Ступінь насиченості ринку товаром визначається його кількістю, наявною на ринку. Кількість товару, необхідного для задоволення попиту покупців, що може бути реалізований на ринку, визначає місткість ринку за даним видом (групою) товарів.

Місткість ринку для такого специфічного продукту, як ПЗ визначається кількістю і типом ЕОМ, установлених у країні, характером задач, що вирішуються, кваліфікацією користувачів тощо. Крім того, при оцінюванні місткості ринку в даному випадку необхідно враховувати, що ПЗ легко тиражується, витрати на тиражування неістотні порівняно з витратами на його розробку, рекламу і збут. Можливість тиражування ПЗ дає змогу швидко збільшувати насиченість ринку. Збільшення тиражів дасть можливість індустрії ПЗ досить швидко розвиватися при відносно низьких цінах, а також сприяти зростанню потенційної місткості внутрішнього ринку. Значна частина ПЗ на наш ринок потрапляє у складі апаратно-програмних комплексів на базі імпортних ЕОМ. Таке придбання не відбиває потреб у ПЗ ані в замовленнях організацій, ані в кількості програм, що реально експлуатуються. Нагромадження ПЗ у підприємств, установ та окремих програмістів здійснюється через їх немовби безкоштовне надходження, використання для придбання інших програм (як засіб обміну або платежу). У зв’язку із цим треба було б визначати місткість ринку ПЗ за їх офіційною вартісною оцінкою як продукції виробничо-технічного призначення. Недосконалість існуючої методики оцінювання місткості ринку підтверджується й тим, що більша частина ПЗ реалізується як науково-технічна продукція.

Таблиця 4.1. Розрахунок чистих грошових надходжень від розробки ПЗ

Показники, грн	По рокам						Всього за 5 років	Середнє за 5 років
	0	1	2	3	4	5		
1. Інвестиції в ПЗ (Придбання ПК)	24248	0	0	0	0	0	24248	4849,6
2. Витрати до впровадження ПЗ	–	170750	175750	183750	194750	202750	927750	185550
- на придбання ліцензій	–	2000					2000	400
- на щорічну діагностику ПК	–	750	750	750	750	750	3750	750
- на оплату праці дизайнера	–	72000	75000	78000	81000	85000	391000	78200
- на оплату праці програмісту	–	96000	100000	105000	113000	117000	531000	106200
3. Витрати після впровадження ПО	–	154825	142025	145075	147125	148175	737225	147445
- на придбання вимірюваного обладнання	–	4850					4850	970
- на щорічну діагностику ноутбука і обладнання	–	775	775	775	775	775	3875	775
- на оплату праці аналітика	–	144000	140000	143000	145000	147000	719000	143800
- на електроенергію	–	1200	1250	1300	1350	1400	6500	1300
4. Економія	–	15925	33725	38675	47625	54575	190525	38105
5. Амортизація	–	12350	12350	12350	12350	12350	61750	12350
6. Чисті грошові надходження	–	3575	21375	26352	35275	42225	128802	25760
7. Коефіцієнт дисконтування	–	0,870	0,736	0,658	0,572	0,497		
Дисконтні грошові надходження	–	3110	15732	17339	20177	20985	77343	15468

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Чиста поточна вартість доходів:

$$NPU = 77\ 343 – 24\ 248 = 53\ 095грн > 0$$

Строк окупності:

$$Т = \frac{24\ 248}{15\ 468} = 1{,}5 \text{ років}$$

Індекс прибутковості:

$$ІД = \frac{77\ 343}{24\ 248} = 3{,}18$$

Показник економічної ефективності (NPU – чиста поточна вартість доходів за роки реалізації впровадження (3-5 років)) складе 77 343 грн, тобто він відповідає умовам ефективності, тому що NPU > 0.

Середній термін окупності капіталовкладень складе 1,5 рік.

Індекс прибутковості за 5 років складе 3,18, тобто ІП > 1, проект варто прийняти.

Таким чином, показник ефективності свідчить про те, що дане впровадження є економічно вигідним.

Висновки до економічного розділу

При розрахунку трудомісткості розробки програмного забезпечення отримали значення рівні 808 чол.-г. Також були розраховані затрати на створення програмного забезпечення, які склали 24 248 грн. Загальна тривалість розробки програмного продукту складає приблизно 5 місяців, а також середній термін окупності проекту становить 1,5 рік, а індекс прибутковості за 5 років становить 3,18. Чиста вартість доходів за роки реалізації впровадження (3-5 років) складе 77 343 грн.

ВИСНОВКИ

В рамках даної магістерської роботи була запропонована методика структурно-параметричної ідентифікації процесів крупного дроблення руд, що включає процедури генерування структур, селекцію моделей і оптимізацію їх параметрів. Розроблено алгоритми глобальної і локальної оптимізації моделі процесу великого дроблення руд, які реалізують процедуру структурно-параметричної ідентифікації шляхом їх структурної та параметричної оптимізації, що дозволяє отримувати моделі підвищеної точності.

Також було розроблено програмне забезпечення для виконання структурно-параметричної ідентифікації процесу великого дроблення руд, що реалізує запропоновану методику. Завдяки застосуванню межпроцесорного підходу стало можливим об'єднати керуючі модулі програмного забезпечення структурно-параметричної ідентифікації, що базуються на платформі Java, і засоби, які реалізують математичні обчислення, що використовують платформу MATLAB.

ПРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузнецов С.П. Динамический хаос / С.П. Кузнецов – М.: Физматлит, 2002. – 296 с.

2. Шустер Г. Детерминированный хаос. Введение / Г. Шустер – М.: Мир, 1988. – 256 с.

3. Анищенко В.С. Знакомство с нелинейной динамикой / В.С. Анищенко // М.-Ижевск: ИКИ, 2002. – 172 с.

4. Корнієнко В.І. Автоматизація оптимального керування процесами дроблення і здрібнювання руд / В.І. Корнієнко. – Д.: Національний гірничий університет, 2013. – 191 с.

5. Ивахненко А.Г. Помехоустойчивость моделирования / А.Г. Ивахненко, В.С. Степашко. – К.: Наукова думка, 1985. – 214 с.

6. Сычев В. Фрактальный анализ. Программа Fractan 4.4 / В. Сычев. – http://impb.ru/~sychov/.

7. Марюта А.Н. Автоматическое управление технологическими процессами обогатительных фабрик / А.Н. Марюта, Ю.Г. Качан, В.А. Бунько. - М.: Недра, 1983.-277 с.

8. Ивахненко А.Г. Предсказание случайных процессов / А.Г. Ивахненко, В.Г. Лапа. - К.: Наукова думка, 1971. -416 с.

9. Ивахненко А.Г. Долгосрочное прогнозирование и управление сложными системами / А.Г. Ивахненко. - К.: Технпса, 1975. - 312 с.

10. Mueller J.-A. Self-Organising Data Mining. An Intelligent Approach To Extract Knowledge From Data / J.-A. Mueller, F. Lemke. - Berlin, Dresden, 1999. -225 p.

11. Ивахненко А.Г. Самоорганизация прогнозирующих моделей / А.Г. Ивахненко, И.А. Мюллер. - К.: Техшка, 1985; Берлин: ФЕБ Ферлаг Техник, 1984.-223 с.

12. Ивахненко А.Г. Помехоустойчивость моделирования / А.Г. Ивахненко, В.С. Степашко. - К.: Наук, думка, 1985. - 214 с.

13. Искусственный интеллект: применение в интегрированных производственных системах / Под ред. Э. Кьюсака. - М.: Машиностроение, 1991. - 544 с.

14. Фогель Л. Искусственный интеллект и эволюционное моделирование / Л. Фогель, А. Оуэне, М. Уолш. - М.: Мир, 1969. - 230 с.

15. Букатова И.Л. Эволюционное моделирование и его приложения / И.Л. Букатова. - М.: Наука, 1979. - 232 с.

16. Дюк В. Интеллектуальный анализ данных: Data Mining / В. Дюк, А. Самойленко. - СПб.: Питер, 2001. - 368 с.

17. Букатова И.Л. Эвоинформатика: теория и практика эволюционного моделирования / И.Л. Букатова, Ю.И. Михасев, А.М. Шаров. - М.: Наука, 1991. -206 с.

18. Емельянов В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В.В. Курейчик, В.М. Курейчик. - М: ФИЗМАТЛИТ, 2003. - 432 с. - ISBN 5-9221-0337-7.

19. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Вороновский Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.Н. - Харьков: Основа, 1997. - 112 с.

20. Holland J.H. Adaptation in natural and artificial systems. An introductory analysis with application to biology, control and artificial intelligence / J.H. Holland. - London: Bradford book edition, 1994. - 211 p.

21. Круглов В.В. Нечеткая логика и искусственные нейронные сети / В.В. Круглов, М.И. Дли, Р.Ю. Голунов. - М.: Физматлит, 2001. - 224 с.

22. Розенблатт Ф. Принципы нейродинамики. Персептрон и теория механизмов мозга / Ф. Розенблатт. - М.: Мир, 1965. - 480 с.

23. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика / Ф. Уоссермен - М.: Мир, 1992. - 237 с.

24. Фогель Л. Искусственный интеллект и эволюционное моделирование / Л. Фогель, А. Оуэне, М. Уолш. - М.: Мир, 1969. - 230 с.

25. Тархов Д.А. Нейронные сети: модели и алгоритмы / Д.А. Тархов. - М.: Радиотехника, 2005. - 256 с.

26. Головко В.А. Нейронные сети: обучение, организация и применение / В.А. Головко. - М.: ИПРЖР, 2001. - 256 с.

27. Нейронные сети. MATLAB 6 / Под общ. ред. В.Г. Потемкина. -М.: ДИАЛОГ-МИФИ, 2002. - 496с.

28. Прикладные нечеткие системы / Под ред. Т.Тэрано, К.Асаи, М.Сугэно. - М.: Мир, 1993. - 368 с.

29. Змитрович А.И. Интеллектуальные информационные системы / А.И. Змитрович. - Минск: НТОО «Тетри Системз», 1997. - 368 с.

30. Штовба С.Д. Проектирование нечетких систем средствами Matlab / С.Д. Штовба. - М.: Горячая линия - Телеком, 2007. - 288 с. - ISBN 5-93517-359-X.

31. Корниенко В.И. Интеллектуальные методы структурно-параметрической идентификации технологических процессов рудоподготовки / В.И. Корниенко, А.В. Пивоварова // Прнича електромехашка та автоматика. -2008. - Вип. 80. - С. 71-77. - ISSN 0201-7814.

32. Nelles O. Nonlinear System Identification: From Classical Approaches to Neural and Fuzzy Models / O. Nelles. – Berlin: Springer, 2001. – 785 p.

33. Малинецкий Г.Г. Современные проблемы нелинейной динамики / Г.Г. Малинецкий, А.Б. Потапов // М.: Эдиториал УРСС, 2000. – 336 с. – ISBN 5-8360-0110-3.

34. Справочник по теории автоматического управления / Под ред. А.А. Красовского. – М.: Наука, 1987. – 712 с.

35. Кузнецов Г.В. Композиційна структурно-параметрична ідентифікація нелінійних динамічних об’єктів керування / Г.В. Кузнецов, В.І. Корнієнко, О.В. Герасіна // Наукові вісті НТУУ КПІ. – 2009. – № 5. – С. 69-75.

36. Охорзин В.А. Прикладная математика в системе MATHCAD Учебное пособие. 3-е изд. – СПб.: Лань, 2009 - 352с.

37. Кирсанов М.Н. "Практика программирования в системе Maple" – М.: Издательский дом МЭИ, 2011. - 208с.

38. Дьяконов В.П. MATLAB и SIMULINK для радиоинженеров. — М.: «ДМК-Пресс», 2011. — 976 с. — ISBN 978-5-94074-492-4.

39. Таранчук В. Б. Основные функции систем компьютерной алгебры. — Минск: БГУ, 2013. — 59 с.

40. Ксу Д. Взаимодействие Matlab с ANSI C, Visual C++, Visual BASIC и Java. - М.: Питер, 2005.

41. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети - М.: Горячая линия – Телеком, 2015. - 284 с.

42. Гибридные модели прогнозирования коротких временных рядов/Л. А. Демидова, А. Н. Пылькин, С. В. Скворцов , Т. С. Скворцова - М.: Горячая линия – Телеком, 2015. - 208 с.

43. Белов В.В., Смирнов А.Е., Чистякова В.И. Распознавание нечётко определяемых состояний технических систем - М.: Горячая линия – Телеком, 2014. - 140 с.

44. Алгоритмы категорирования персональных данных для систем автоматизированного проектирования баз данных информационных систем/А.В. Благодаров, В.С.Зияутдинов, П.А. Корнев, В.Н. Малыш - М.: Горячая линия – Телеком, 2013. - 116 с.

45. Таганов А.И. Основы идентификации, анализа и мониторинга проектных рисков качества программных изделий в условиях нечеткости - М.: Горячая линия – Телеком, 2015. - 224 с.

46. Таганов А.И., Гильман Д.В. Методологические основы анализа и аттестации уровней зрелости процессов программных проектов в условиях нечеткости - М.: Горячая линия – Телеком, 2014. - 168 с.

47. Новые методы математического моделирования динамики и управления формированием компетенций в процессе обучения в вузе / А. А. Большаков, И. В. Вешнева, Л. А. Мельников и др. – М.: Горячая линия – Телеком, 2013. – 248 с.: ил.

48. Штовба С.Д. Проектирование нечетких систем средствами Matlab: Справочное издание - – М.: Горячая линия – Телеком, 2007. – 288 с.: ил.

49. Демидова Л.А., Кираковский В.В., Пылькин А.Н. Алгоритмы и системы нечеткого вывода при решении задач диагностики городских инженерных коммуникаций в среде Matlab – М.: Горячая линия – Телеком, 2005. – 365 с.: ил.

50. Усков А.А., Кузьмин А.В. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика - М.: Горячая линия – Телеком, 2004. - 143 с.

51. Минаев Ю.Н., Филимонова О.Ю., Бенамеур Л. Методы и алгоритмы индентификации и прогнозирования в условиях неопределенности в нейросетевом логическом базисе – М.: Горячая линия – Телеком, 2003. – 205с.: ил.

52. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы – М.: Горячая линия – Телеком, 2013. – 384с.: ил.

53. Демидова Л.А., Кираковский В.В., Пылькин А.Н. Принятие решений в условиях неопределенности – М.: Горячая линия – Телеком, 2015. – 289с.: ил.

54. Галушкин А.И. Нейронные сети: основы теории - М.: Горячая линия – Телеком, 2015. - 496 с.

55. Элементарное введение в технологию нейронных сетей с примерами программ// Р. Тадеусевич, Б. Боровик , Т. Гончаж, Б. Леппер - М.: Горячая линия – Телеком, 2011. - 408 с.

56. Гибридные модели прогнозирования коротких временных рядов//Л. А. Демидова, А. Н. Пылькин, С. В. Скворцов , Т. С. Скворцова

57. Демидова Л.А., Кираковский В.В., Пылькин А.Н. Алгоритмы и системы нечеткого вывода при решении задач диагностики городских инженерных коммуникаций в среде Matlab – М.: Горячая линия – Телеком, 2005. – 365 с.: ил.

58. Алгоритмы категорирования персональных данных для систем автоматизированного проектирования баз данных информационных систем //А.В. Благодаров, В.С.Зияутдинов, П.А. Корнев, В.Н. Малыш - М.: Горячая линия – Телеком, 2013. - 116 с.

59. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети - М.: Горячая линия – Телеком, 2015. - 284 с.

60. Усков А.А., Кузьмин А.В. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика - М.: Горячая линия – Телеком, 2004. - 143 с.

61. Минаев Ю.Н., Филимонова О.Ю., Бенамеур Л. Методы и алгоритмы индентификации и прогнозирования в условиях неопределенности в нейросетевом логическом базисе – М.: Горячая линия – Телеком, 2003. – 205с.: ил.

62. Нейрокомпьютеры: от программной к аппаратной реализации/ М.А. Аляутдинов, А.И. Галушкин, П.А. Казанцев, Г.П. Остапенко – М.: Горячая линия – Телеком, 2008. – 152 с.: ил.

63. Комашинский В.И., Смирнов Д.А. Нейронные сети и их применение в системах управления и связи – М.: Горячая линия – Телеком, 2003. – 94с.: ил.

64. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы – М.: Горячая линия – Телеком, 2013. – 384с.: ил.

65. Боровиков В.П. Популярное введение в современный анализ данных в системе STATISTICA: Учебное пособие для вузов (+ CD-Rom) – М.: Горячая линия – Телеком, 2015. – 288с.: ил.

66. Нейронные сети. STATISTICA Neural Networks: Методология и технологии современного анализа данных: Справочное издание/Под редакцией В.П. Боровикова – М.: Горячая линия – Телеком, 2008. – 392с.: ил.

67. Локтюхин В.Н., Челебаев С.В. Нейросетевые преобразователи импульсно-аналоговой информации: организация, синтез, реализация – М.:

Горячая линия – Телеком, 2008. – 144с.: ил. (Серия "Современная электроника") (реализация на ПЛИС фирмы Xilinx)

68. Салов В.О. Методичні рекомендації до виконання кваліфікаційних робіт випускників спеціальності 8.080403 «Програмне забезпечення автоматизованих систем» напряму 0804 «Комп'ютерні науки». Стандарт вищого навчального закладу (СТВНЗ-2070743-КР 2004). / В.О. Салов, М.О. Алексєєв, Г.М. Коротенко, Л.М. Коротенко; М-во освіти і науки України, Нац. гірн. ун-т. – Д.: НГУ, 2004.- 55с.

69. Вагонова О.Г. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун-т». – Д.: НГУ, 2012. – 11 с.

ДОДАТОК А

ОБҐРУНТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТРУКТУРНО-ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ДЛЯ АСУ ПРОЦЕСОМ КРУПНОГО ДРОБЛЕННЯ РУД

Текст програми

Аркушів 26

2018

Документ «Обґрунтування програмного забезпечення структурно-параметричної ідентифікації для АСУ процесом крупного дроблення руд». «Текст програми» призначений для розробки компонентів програмних засобів, призначених для реалізації методики структурно-параметричної ідентифікації складних об'єктів управління.

Даний документ містить текст програми. Цей документ призначений для розробника програмного забезпечення.

Зміст

Math4JavaApp.java

```
package diploma.sio;

public class Math4JavaApp {

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainWindow().setVisible(true);
            }
        });
    }
}
```

MainWindow.java

```
package diploma.sio;

import diploma.sio.dto.MathModel;
import diploma.sio.enums.NetworkArchitectureTypeEnum;
import diploma.sio.enums.map.NetworkArchitectureTypeMapper;

public class MainWindow extends javax.swing.JFrame {

    private final Controller controller;
    private final NetworkArchitectureTypeMapper
networkArchitectureTypeMapper;
    /**
     * Creates new form MainWindow
     */
    public MainWindow()
    {
        setTitle("Структурно-параметрическая идентификация");
        initComponents();

        controller = new Controller();
        networkArchitectureTypeMapper = new
NetworkArchitectureTypeMapper();
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    private void initComponents() {

        executeButton = new javax.swing.JButton();
        networkArchitectureLabel = new javax.swing.JLabel();
        optimizationMethodComboBox = new javax.swing.JComboBox();
        learningAlgorithmTrFcnComboBox = new javax.swing.JComboBox();
        networkArchitectureComboBox = new javax.swing.JComboBox();
```

```
        activationFunctionTFComboBox = new javax.swing.JComboBox();
        outDataComboBox = new javax.swing.JComboBox();
        numberOfNeuronsTextField = new javax.swing.JTextField();
        numberOfNeuronsLabel = new javax.swing.JLabel();
        activationFunctionTFLabel = new javax.swing.JLabel();
        learningAlgorithmTrFcnLabel = new javax.swing.JLabel();
        outDataLabel = new javax.swing.JLabel();
        optimizationMethodLabel = new javax.swing.JLabel();
        identifiableObjectModelLabel = new javax.swing.JLabel();
        identifiableObjectModelComboBox = new javax.swing.JComboBox();
        criterionTypeLabel = new javax.swing.JLabel();
        criterionTypeComboBox = new javax.swing.JComboBox();
        exitButton = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        executeButton.setText("Выполнить");
        executeButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                executeButtonPerformer(evt);
            }
        });

        networkArchitectureLabel.setText("Архитектура сети:");

        optimizationMethodComboBox.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "ГА", "ПСП" }));

        networkArchitectureComboBox.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "НСПР", "RBF", "ANFIS"
}));
        networkArchitectureComboBox.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                networkArchitectureTypeComboBoxPerformer(evt);
            }
        });

        outDataComboBox.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Экспериментальные
ИД", "Полигармонический авторегрессионный", "Полосовой авторегрес",
"Широкополосный авторегрес", "Хаотический (отображение Эно)" }));

        numberOfNeuronsLabel.setText("Количество нейронов в скрытом
слое:");

        activationFunctionTFLabel.setText("Функция активации TF:");

        learningAlgorithmTrFcnLabel.setText("Алгоритм обучения
TrFcn:");
        learningAlgorithmTrFcnLabel.setToolTipText("");

        outDataLabel.setText("Исходные данные:");
```

```
        optimizationMethodLabel.setText("Метод оптимизации:");

        identifiableObjectModelLabel.setText("Модель идентифицируемого
объекта:");

        identifiableObjectModelComboBox.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Нет модели объекта
(экспериментальный)", "ПИ с запаздыванием", "Нелинейное конечно-
разностное уравнение", "ПИД с запаздыванием + квадрат безинерц звено",
"Отображение Эно" }));

        criterionTypeLabel.setText("Тип критерия:");
        criterionTypeLabel.setToolTipText("");

        criterionTypeComboBox.setModel(new
javax.swing.DefaultComboBoxModel(new String[] { "Относительный
среднеквадратический критерий несмещенности", "Несмещенности по
эксперементальным и модельным данным", "Комплексный критерий",
"Критерий 1 по вариациям", "Критерий 2 по вариациям" }));

        exitButton.setText("Выход");
        exitButton.setToolTipText("");
        exitButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                exitButtonPerformer(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addComponent(networkArchitectureLabel)
                            .addComponent(learningAlgorithmTrFcnLabel)
                            .addComponent(activationFunctionTFLabel)
                            .addComponent(numberOfNeuronsLabel)
                            .addComponent(optimizationMethodLabel)
                            .addComponent(exitButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
144, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING, false)
```

```
.addComponent(activationFunctionTFComboBox,
javax.swing.GroupLayout.Alignment.TRAILING, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(numberOfNeuronsTextField,
javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(networkArchitectureComboBox,
javax.swing.GroupLayout.Alignment.TRAILING, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addGroup(layout.createSequentialGroup()
                                .addGap(10, 10, 10)
                                .addComponent(executeButton))
                            .addComponent(optimizationMethodComboBox,
javax.swing.GroupLayout.Alignment.TRAILING, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addComponent(learningAlgorithmTrFcnComboBox, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addComponent(criterionTypeComboBox, 0, 439,
Short.MAX_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(criterionTypeLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGap(366, 366, 366))
                    .addComponent(identifiableObjectModelComboBox, 0,
439, Short.MAX_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(outDataLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
70, Short.MAX_VALUE)
                        .addComponent(outDataComboBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 272,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(identifiableObjectModelLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 252,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 187, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(
                     javax.swing.GroupLayout.Alignment.TRAILING,
                         layout.createSequentialGroup()
             .addContainerGap()
        .addGroup(
                  layout.createParallelGroup(
                     javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(optimizationMethodComboBox,
                  javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(optimizationMethodLabel))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addGroup(
                layout.createParallelGroup(
                 javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(networkArchitectureComboBox,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(networkArchitectureLabel))
       .addPreferredGap(
            javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addGroup(
                layout.createParallelGroup(
                  javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(numberOfNeuronsTextField,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(numberOfNeuronsLabel))
       .addPreferredGap(
            javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addGroup(
                layout.createParallelGroup(
                      javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(activationFunctionTFComboBox,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(activationFunctionTFLabel))
        .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addGroup(layout.createParallelGroup(
                javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(learningAlgorithmTrFcnComboBox,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                   .addComponent(learningAlgorithmTrFcnLabel))
         .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addGroup(layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.BASELINE)
             .addComponent(outDataComboBox,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
                   .addComponent(outDataLabel))
         .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(identifiableObjectModelLabel)
         .addPreferredGap(
                javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(identifiableObjectModelComboBox,
                   javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(
              javax.swing.LayoutStyle.ComponentPlacement.RELATED)
          .addComponent(criterionTypeLabel)
       .addGap(9, 9, 9)
          .addComponent(criterionTypeComboBox,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(23, 23, 23)
  .addGroup(layout.createParallelGroup(
               javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(executeButton)
          .addComponent(exitButton))
            .addContainerGap(
           javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    pack();
}

private void executeButtonPerformer(
                                java.awt.event.ActionEvent evt) {
    try
    {
        NetworkArchitectureTypeEnum networkArchitectureTypeEnum =
                networkArchitectureTypeMapper.map(
                networkArchitectureComboBox.getSelectedIndex());
        MathModel model =collectData(networkArchitectureTypeEnum);
        Object[] ob = controller.generate(model);

        new ResultWindow(ob);
    }
    catch (Exception e)
    {
        e.printStackTrace();
        // handle exception
    }
}

private void exitButtonPerformer(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void networkArchitectureTypeComboBoxPerformer(
                              java.awt.event.ActionEvent evt) {
    NetworkArchitectureTypeEnum networkArchitectureTypeEnum =
    networkArchitectureTypeMapper.map(networkArchitectureComboBox
                    .getSelectedIndex());

    switch(networkArchitectureTypeEnum)
    {
        case NSPR:{
            setNSPR();
            break;
        }
        case RBF:{
            setRBF();
```

```
            break;
        }
        case ANFIS:{
            setAnfis();
            break;
        }
    }
}

private void setNSPR()
{
    activationFunctionTFComboBox.setEnabled(true);
    learningAlgorithmTrFcnComboBox.setEnabled(true);
    numberOfNeuronsTextField.setText("25");
    activationFunctionTFComboBox.setModel(
    new javax.swing.DefaultComboBoxModel(
          new String[] { "1", "2", "3", "4", "LOGSIG", "6", "7",
                   "8", "9", "10", "11", "12", "13" }));
    activationFunctionTFComboBox.setSelectedIndex(4);
    learningAlgorithmTrFcnComboBox.setModel(
    new javax.swing.DefaultComboBoxModel(
          new String[] { "TRAINGD", "2", "3", "4", "5",
                   "6", "7", "8", "9", "10", "11" }));
    outDataComboBox.setSelectedIndex(0);
    identifiableObjectModelComboBox.setSelectedIndex(0);
}

private void setRBF()
{
    activationFunctionTFComboBox.setEnabled(false);
    learningAlgorithmTrFcnComboBox.setEnabled(false);
    numberOfNeuronsTextField.setText("0");
    activationFunctionTFComboBox.removeAllItems();
    learningAlgorithmTrFcnComboBox.removeAllItems();
    outDataComboBox.setSelectedIndex(0);
    identifiableObjectModelComboBox.setSelectedIndex(0);
}

private void setAnfis()
{
    activationFunctionTFComboBox.setEnabled(true);
    learningAlgorithmTrFcnComboBox.setEnabled(true);
    numberOfNeuronsTextField.setText("0");
    activationFunctionTFComboBox.setModel(
new javax.swing.DefaultComboBoxModel(
new String[] { "14", "15", "16", "17", "18", "19", "20", "21" }));
    learningAlgorithmTrFcnComboBox.setModel(
new javax.swing.DefaultComboBoxModel(new String[] { "12", "13"}));
    outDataComboBox.setSelectedIndex(0);
    identifiableObjectModelComboBox.setSelectedIndex(0);
}




private MathModel collectData(
               NetworkArchitectureTypeEnum architectureType)
```

```
    {
        MathModel model = new MathModel();

        model.setArchitecture(networkArchitectureComboBox
                    .getSelectedIndex()+1);
        model.setCount(Integer.valueOf(
                    numberOfNeuronsTextField.getText()));
        model.setCriterion(criterionTypeComboBox
                   .getSelectedIndex()+1);
        model.setMethod(optimizationMethodComboBox
                   .getSelectedIndex()+1);
        model.setModelInd(identifiableObjectModelComboBox
                   .getSelectedIndex());
        model.setOutData(outDataComboBox.getSelectedIndex());

        switch (architectureType)
        {
            case RBF:
                model.setTypeAlgorithm(0);
                model.setTypeFunc(0);
            default:
                model.setTypeAlgorithm(Integer.parseInt(
        learningAlgorithmTrFcnComboBox.getSelectedItem().toString()));
                model.setTypeFunc(Integer.parseInt(
        activationFunctionTFComboBox.getSelectedItem().toString()));
                break;
        }

        return model;
    }

    private javax.swing.JComboBox activationFunctionTFComboBox;
    private javax.swing.JLabel activationFunctionTFLabel;
    private javax.swing.JComboBox criterionTypeComboBox;
    private javax.swing.JLabel criterionTypeLabel;
    private javax.swing.JButton executeButton;
    private javax.swing.JButton exitButton;
    private javax.swing.JComboBox identifiableObjectModelComboBox;
    private javax.swing.JLabel identifiableObjectModelLabel;
    private javax.swing.JComboBox learningAlgorithmTrFcnComboBox;
    private javax.swing.JLabel learningAlgorithmTrFcnLabel;
    private javax.swing.JComboBox networkArchitectureComboBox;
    private javax.swing.JLabel networkArchitectureLabel;
    private javax.swing.JLabel numberOfNeuronsLabel;
    private javax.swing.JTextField numberOfNeuronsTextField;
    private javax.swing.JComboBox optimizationMethodComboBox;
    private javax.swing.JLabel optimizationMethodLabel;
    private javax.swing.JComboBox outDataComboBox;
    private javax.swing.JLabel outDataLabel;
}
```

Controller.java

```
package diploma.sio;

import com.mathworks.toolbox.javabuilder.MWClassID;
import com.mathworks.toolbox.javabuilder.MWComplexity;
import com.mathworks.toolbox.javabuilder.MWException;
import com.mathworks.toolbox.javabuilder.MWNumericArray;
import diploma.sio.dto.MathModel;
import GA2.GA;

public class Controller
{
    public Object[] generate(MathModel model) throws MWException
    {
        GA gaInvoker = new GA();
        int[] o={1,4};
        MWNumericArray mass=MWNumericArray.newInstance(o,
MWClassID.DOUBLE, MWComplexity.REAL);
        mass.set(1,model.getArchitecture());
        mass.set(2,model.getCount());
        mass.set(3,model.getTypeFunc());
        mass.set(4,model.getTypeAlgorithm());
        Object[] ob = gaInvoker.GA(6, mass);
        gaInvoker.dispose();
        return ob;

    }
}
```

NetworkArchitectureTypeEnum.java

```
package diploma.sio.enums;

public enum NetworkArchitectureTypeEnum
{
    NSPR, RBF, ANFIS;
}
```

NetworkArchitectureTypeMapper.java

```
package diploma.sio.enums.map;

import diploma.sio.enums.NetworkArchitectureTypeEnum;

public class NetworkArchitectureTypeMapper implements
             Mapper<NetworkArchitectureTypeEnum>
{
    @Override
    public NetworkArchitectureTypeEnum map(int out)
    {
       NetworkArchitectureTypeEnum resultType =
                NetworkArchitectureTypeEnum.NSPR;
       switch (out)
       {
           case 0: resultType = NetworkArchitectureTypeEnum.NSPR;
```

```
                    break;
            case 1: resultType = NetworkArchitectureTypeEnum.RBF;
                     break;
            case 2: resultType = NetworkArchitectureTypeEnum.ANFIS;
                    break;
        }

        return resultType;
    }
}
```

MathModel.java

```
package diploma.sio.dto;

public class MathModel
{
    private int method;
    private int architecture;
    private int count;
    private int typeFunc;
    private int typeAlgorithm;
    private int outData;
    private int modelInd;
    private int criterion;

    public int getMethod()
    {
        return method;
    }

    public void setMethod(int method)
    {
        this.method = method;
    }

    public int getArchitecture()
    {
        return architecture;
    }

    public void setArchitecture(int architecture)
    {
        this.architecture = architecture;
    }

    public int getCount()
    {
        return count;
    }

    public void setCount(int count)
    {
        this.count = count;
    }

    public int getTypeFunc()
    {
```

```
        return typeFunc;
    }

    public void setTypeFunc(int typeFunc)
    {
        this.typeFunc = typeFunc;
    }

    public int getTypeAlgorithm()
    {
        return typeAlgorithm;
    }

    public void setTypeAlgorithm(int typeAlgorithm)
    {
        this.typeAlgorithm = typeAlgorithm;
    }

    public int getOutData()
    {
        return outData;
    }

    public void setOutData(int outData)
    {
        this.outData = outData;
    }

    public int getModelInd()
    {
        return modelInd;
    }

    public void setModelInd(int modelInd)
    {
        this.modelInd = modelInd;
    }

    public int getCriterion()
    {
        return criterion;
    }

    public void setCriterion(int criterion)
    {
        this.criterion = criterion;
    }
}
```

ResultWindow.java

```
package diploma.sio;

import com.mathworks.toolbox.javabuilder.MWNumericArray;

public class ResultWindow extends javax.swing.JFrame {

    private static final String ARCHITECTURE_TYPE_LABEL_TEXT_VALUE =
"Тип архитектуры сети:";


    public static void main(String[] args)
    {
        new ResultWindow().setVisible(true);
    }

    public ResultWindow()
    {
        setTitle("Страница результатов");
        initComponents();
    }

    public ResultWindow(Object[] objects)
    {
        initComponents();
        setData(objects);
    }

    private void setData(Object[] objects)
    {
        if (objects != null)
        {
            double[]
struct=((MWNumericArray)objects[0]).getDoubleData();
            jTextField9.setText(String.format("%.4f",struct[0]));
            jTextField10.setText(String.format("%.4f",struct[1]));
            jTextField7.setText(String.format("%.4f",struct[2]));

activationFunctionTypeTextField.setText(String.format("%.4f",struct[3]
));

            double opt=((MWNumericArray)objects[1]).getDouble();

structuralOptimizationCriterionTextField.setText(String.format("%.4f",
opt));

            double[]
population=((MWNumericArray)objects[4]).getDoubleData();
            setPopulation(population);

            double[]
scores=((MWNumericArray)objects[5]).getDoubleData();
            setScores(scores);
        }
    }

    private void setPopulation(double[] population)
```

```
{
    int i=0;
    //1
    jTextField1.setText(String.format("%.4f",population[0+i]));
    jTextField21.setText(String.format("%.4f",population[20+i]));
    jTextField41.setText(String.format("%.4f",population[40+i]));
    jTextField61.setText(String.format("%.4f",population[60+i]));
    //2
    i++;
    jTextField2.setText(String.format("%.4f",population[0+i]));
    jTextField22.setText(String.format("%.4f",population[20+i]));
    jTextField42.setText(String.format("%.4f",population[40+i]));
    jTextField62.setText(String.format("%.4f",population[60+i]));
    //3
    i++;
    jTextField3.setText(String.format("%.4f",population[0+i]));
    jTextField23.setText(String.format("%.4f",population[20+i]));
    jTextField43.setText(String.format("%.4f",population[40+i]));
    jTextField63.setText(String.format("%.4f",population[60+i]));
    //4
    i++;
    jTextField4.setText(String.format("%.4f",population[0+i]));
    jTextField24.setText(String.format("%.4f",population[20+i]));
    jTextField44.setText(String.format("%.4f",population[40+i]));
    jTextField64.setText(String.format("%.4f",population[60+i]));
    //5
    i++;
    jTextField5.setText(String.format("%.4f",population[0+i]));
    jTextField25.setText(String.format("%.4f",population[20+i]));
    jTextField45.setText(String.format("%.4f",population[40+i]));
    jTextField65.setText(String.format("%.4f",population[60+i]));
    //6
    i++;
    jTextField6.setText(String.format("%.4f",population[0+i]));
    jTextField26.setText(String.format("%.4f",population[20+i]));
    jTextField46.setText(String.format("%.4f",population[40+i]));
    jTextField66.setText(String.format("%.4f",population[60+i]));
    //7
    i++;
    jTextField7.setText(String.format("%.4f",population[0+i]));
    jTextField27.setText(String.format("%.4f",population[20+i]));
    jTextField47.setText(String.format("%.4f",population[40+i]));
    jTextField67.setText(String.format("%.4f",population[60+i]));
    //8
    i++;
    jTextField8.setText(String.format("%.4f",population[0+i]));
    jTextField28.setText(String.format("%.4f",population[20+i]));
    jTextField48.setText(String.format("%.4f",population[40+i]));
    jTextField68.setText(String.format("%.4f",population[60+i]));
    //9
    i++;
    jTextField9.setText(String.format("%.4f",population[0+i]));
    jTextField29.setText(String.format("%.4f",population[20+i]));
    jTextField49.setText(String.format("%.4f",population[40+i]));
    jTextField69.setText(String.format("%.4f",population[60+i]));
    //10
    i++;
    jTextField10.setText(String.format("%.4f",population[0+i]));
```

```
jTextField30.setText(String.format("%.4f",population[20+i]));
jTextField50.setText(String.format("%.4f",population[40+i]));
jTextField70.setText(String.format("%.4f",population[60+i]));
//11
i++;
jTextField11.setText(String.format("%.4f",population[0+i]));
jTextField31.setText(String.format("%.4f",population[20+i]));
jTextField51.setText(String.format("%.4f",population[40+i]));
jTextField71.setText(String.format("%.4f",population[60+i]));
//12
i++;
jTextField12.setText(String.format("%.4f",population[0+i]));
jTextField32.setText(String.format("%.4f",population[20+i]));
jTextField52.setText(String.format("%.4f",population[40+i]));
jTextField72.setText(String.format("%.4f",population[60+i]));
//13
i++;
jTextField13.setText(String.format("%.4f",population[0+i]));
jTextField33.setText(String.format("%.4f",population[20+i]));
jTextField53.setText(String.format("%.4f",population[40+i]));
jTextField73.setText(String.format("%.4f",population[60+i]));
//14
i++;
jTextField14.setText(String.format("%.4f",population[0+i]));
jTextField34.setText(String.format("%.4f",population[20+i]));
jTextField54.setText(String.format("%.4f",population[40+i]));
jTextField74.setText(String.format("%.4f",population[60+i]));
//15
i++;
jTextField15.setText(String.format("%.4f",population[0+i]));
jTextField35.setText(String.format("%.4f",population[20+i]));
jTextField55.setText(String.format("%.4f",population[40+i]));
jTextField75.setText(String.format("%.4f",population[60+i]));
//16
i++;
jTextField16.setText(String.format("%.4f",population[0+i]));
jTextField36.setText(String.format("%.4f",population[20+i]));
jTextField56.setText(String.format("%.4f",population[40+i]));
jTextField76.setText(String.format("%.4f",population[60+i]));
//17
i++;
jTextField17.setText(String.format("%.4f",population[0+i]));
jTextField37.setText(String.format("%.4f",population[20+i]));
jTextField57.setText(String.format("%.4f",population[40+i]));
jTextField77.setText(String.format("%.4f",population[60+i]));
//18
i++;
jTextField18.setText(String.format("%.4f",population[0+i]));
jTextField38.setText(String.format("%.4f",population[20+i]));
jTextField58.setText(String.format("%.4f",population[40+i]));
jTextField78.setText(String.format("%.4f",population[60+i]));
//19
i++;
jTextField19.setText(String.format("%.4f",population[0+i]));
jTextField39.setText(String.format("%.4f",population[20+i]));
jTextField59.setText(String.format("%.4f",population[40+i]));
jTextField79.setText(String.format("%.4f",population[60+i]));
//20
```

```
        i++;
        jTextField20.setText(String.format("%.4f",population[0+i]));
        jTextField40.setText(String.format("%.4f",population[20+i]));
        jTextField60.setText(String.format("%.4f",population[40+i]));
        jTextField80.setText(String.format("%.4f",population[60+i]));
    }

    private void setScores(double[] scores)
    {
        int i=0;
        jTextField81.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField82.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField83.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField84.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField85.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField86.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField87.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField88.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField89.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField90.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField91.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField92.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField93.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField94.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField95.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField96.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField97.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField98.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField99.setText(String.format("%.4f",scores[i]));
        i++;
        jTextField100.setText(String.format("%.4f",scores[i]));
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor
     */
```

```
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        dataPanel = new javax.swing.JPanel();
        structuralModelCharacteristics = new javax.swing.JLabel();
        architectureTypeLabel = new javax.swing.JLabel();
        architectureTypeTextField = new javax.swing.JTextField();
        numberOfNeuronsTextField = new javax.swing.JTextField();
        activationFunctionTypeTextField = new
javax.swing.JTextField();
        learningAlgorithmTypeTextField = new javax.swing.JTextField();
        structuralOptimizationCriterionTextField = new
javax.swing.JTextField();
        numberOfNeuronsLabel = new javax.swing.JLabel();
        activationFunctionTypeLabel = new javax.swing.JLabel();
        learningAlgorithmTypeLabel = new javax.swing.JLabel();
        structuralOptimizationCriterionLabel = new
javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        graphPanel = new javax.swing.JPanel();
        populationLabel = new javax.swing.JLabel();
        scoresLabel = new javax.swing.JLabel();
        jTextField101 = new javax.swing.JTextField();

        jTextField1.setText("1.0000");
        jTextField1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                jTextField1ActionPerformed(evt);
            }
        });


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        structuralModelCharacteristics.setText("Структурные
характеристики модели");

        architectureTypeLabel.setText("Тип архитектуры сети:");

        architectureTypeTextField.setText("НСПР");

        numberOfNeuronsTextField.setText("25");

        activationFunctionTypeTextField.setText("Логарифмическая
сигмоидная передаточная функция");
        activationFunctionTypeTextField.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                activationFunctionTypeTextFieldActionPerformed(evt);
            }
        });
```

```
        learningAlgorithmTypeTextField.setText("Обратное
распространение градиента");

        structuralOptimizationCriterionTextField.setText("0,0089");

        numberOfNeuronsLabel.setText("Количество нейронов в скрытом
поле:");

        activationFunctionTypeLabel.setText("Тип функции активации:");

        learningAlgorithmTypeLabel.setText("Тип алгоритма обучения:");

        structuralOptimizationCriterionLabel.setText("Критерий
структурной оптимизации");

        javax.swing.GroupLayout dataPanelLayout = new
javax.swing.GroupLayout(dataPanel);
        dataPanel.setLayout(dataPanelLayout);
        dataPanelLayout.setHorizontalGroup(

dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGroup(dataPanelLayout.createSequentialGroup()

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                    .addGroup(dataPanelLayout.createSequentialGroup()
                        .addGap(75, 75, 75)
                        .addComponent(structuralModelCharacteristics))
                    .addGroup(dataPanelLayout.createSequentialGroup()
                        .addGap(134, 134, 134)

.addComponent(structuralOptimizationCriterionTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, 73,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(dataPanelLayout.createSequentialGroup()
                        .addGap(72, 72, 72)

.addComponent(structuralOptimizationCriterionLabel)))
                .addContainerGap(71, Short.MAX_VALUE))
            .addGroup(dataPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                    .addComponent(numberOfNeuronsLabel)
                    .addComponent(architectureTypeLabel)
                    .addComponent(activationFunctionTypeLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.TRAILING)
                    .addComponent(numberOfNeuronsTextField,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE)
```

```
                    .addComponent(architectureTypeTextField,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE))
                .addGap(23, 23, 23))
            .addGroup(dataPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(learningAlgorithmTypeLabel)
                .addContainerGap(202, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
dataPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.TRAILING)
                    .addComponent(learningAlgorithmTypeTextField,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 308, Short.MAX_VALUE)
                    .addComponent(activationFunctionTypeTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 308, Short.MAX_VALUE))
                .addGap(23, 23, 23))
        );
        dataPanelLayout.setVerticalGroup(

dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGroup(dataPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(structuralModelCharacteristics)
                .addGap(18, 18, 18)

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(architectureTypeLabel)
                    .addComponent(architectureTypeTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(dataPanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(numberOfNeuronsTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(numberOfNeuronsLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(activationFunctionTypeLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(activationFunctionTypeTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 22, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(learningAlgorithmTypeLabel)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(learningAlgorithmTypeTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(22, 22, 22)
                .addComponent(structuralOptimizationCriterionLabel)
                .addGap(18, 18, 18)

.addComponent(structuralOptimizationCriterionTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(48, 48, 48))
        );


jTextField2.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField2.setText("1.0000");


jTextField3.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField3.setText("1.0000");



jTextField33.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField33.setText("21.9387");


jTextField34.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField34.setText("21.9328");


jTextField35.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField35.setText("21.9699");


jTextField36.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField36.setText("21.9543");


jTextField37.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField37.setText("21.9231");


jTextField38.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField38.setText("21.9492");


jTextField39.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField39.setText("21.9387");


jTextField40.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField40.setText("21.9328");
```

```
jTextField41.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField41.setText("3.9829");


jTextField42.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField42.setText("3.9517");



jTextField99.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField99.setText("0.0102");


jTextField100.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField100.setText("0.0160");

        populationLabel.setText("Population:");

        scoresLabel.setText("Scores:");


jTextField101.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jTextField101.setText("1.0000");

        javax.swing.GroupLayout graphPanelLayout = new
javax.swing.GroupLayout(graphPanel);
        graphPanel.setLayout(graphPanelLayout);
        graphPanelLayout.setHorizontalGroup(

graphPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(graphPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                    .addGroup(graphPanelLayout.createSequentialGroup()

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.TRAILING)

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING, false)
                                .addComponent(jTextField4)
                                .addComponent(jTextField5)
                                .addComponent(jTextField9)
                                .addComponent(jTextField10)
                                .addComponent(jTextField8)
                                .addComponent(jTextField7)
                                .addComponent(jTextField3)
                                .addComponent(jTextField6)
                                .addComponent(jTextField11)
                                .addComponent(jTextField12)
                                .addComponent(jTextField13)
                                .addComponent(jTextField14)
                                .addComponent(jTextField15)
                                .addComponent(jTextField16)
```

```
                                    .addComponent(jTextField17)
                                    .addComponent(jTextField18)
                                    .addComponent(jTextField19)
                                    .addComponent(jTextField20)
                                    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 68,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addComponent(jTextField101,
javax.swing.GroupLayout.DEFAULT_SIZE, 68, Short.MAX_VALUE))

                                .addComponent(jTextField42,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE)
                                .addComponent(jTextField41,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
graphPanelLayout.createSequentialGroup()
                            .addComponent(populationLabel)
                            .addGap(46, 46, 46)))

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                        .addGroup(graphPanelLayout.createSequentialGroup()

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING, false)
                                .addComponent(jTextField61,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                .addComponent(jTextField62)

javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField74,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField75,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField76,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField77,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField78,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE)
                                        .addComponent(jTextField79,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 65, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
                            .addGap(40, 40, 40)
```

```
.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                                .addComponent(jTextField82,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField81,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField83,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField84,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField85,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField86,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField87,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField88,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField89,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField90,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField91,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField92,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField93,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField94,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField95,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField96,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField97,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField98,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField99,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE)
                                .addComponent(jTextField100,
javax.swing.GroupLayout.DEFAULT_SIZE, 82, Short.MAX_VALUE))
                            .addContainerGap())

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE)
                                .addComponent(jTextField88,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jTextField68,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE)
                            .addComponent(jTextField89,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField69,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(graphPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE)
                            .addComponent(jTextField90,

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField47,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField48,

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField57,
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jTextField60,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addContainerGap(20, Short.MAX_VALUE))
        );

        jScrollPane1.setViewportView(graphPanel);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(dataPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 425, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(dataPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
        );

        pack();
    }

    private void jTextField1ActionPerformed(
                        java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void activationFunctionTypeTextFieldActionPerformed(
                                java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel activationFunctionTypeLabel;
    private javax.swing.JTextField activationFunctionTypeTextField;
    private javax.swing.JLabel architectureTypeLabel;
    private javax.swing.JTextField architectureTypeTextField;
    private javax.swing.JPanel dataPanel;
    private javax.swing.JPanel graphPanel;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JLabel learningAlgorithmTypeLabel;
    private javax.swing.JTextField learningAlgorithmTypeTextField;
    private javax.swing.JLabel numberOfNeuronsLabel;
    private javax.swing.JTextField numberOfNeuronsTextField;
    private javax.swing.JLabel populationLabel;
    private javax.swing.JLabel scoresLabel;
    private javax.swing.JLabel structuralModelCharacteristics;
    private javax.swing.JLabel structuralOptimizationCriterionLabel;
    private javax.swing.JTextField structuralOptimizCriterionField;
}
```

ДОДАТОК Б

ОБҐРУНТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТРУКТУРНО-ПАРАМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ДЛЯ АСУ ПРОЦЕСОМ КРУПНОГО ДРОБЛЕННЯ РУД

Перелік документів на компакт-диску

Аркушів 1

2018

Перелік документів на оптичному носії

Назва файлу	**Опис**
Пояснювальна записка	
Пояснювальна записка.docx	Пояснювальна записка
Додатки А-Г.docx	Додатки
Програма	
Diploma.zip	Архів. Містить вихідні коди програми, файли бази даних і програму в скомпільованій формі
Презентація	
Презентація_Стадніченко.pptx	Презентація дипломного проекту

ДОДАТОК В

ВІДГУК

на дипломну роботу магістра на тему:

«Обґрунтування програмного забезпечення структурно-параметричної ідентифікації для АСУ процесом крупного дроблення руд»

студента групи 121м-16-1 Стадніченка Ігора Олеговича

1. Метою дипломної роботи магістра є обґрунтування методики структурно-параметричної ідентифікації процесів крупного дроблення руд, що забезпечує підвищення точності їх моделей..
2. Актуальність даної теми обумовлена наявністю значних недоліків в традиційному підході до розробки динамічних моделей складних об'єктів управління, яка здійснюється за допомогою процедури їх ідентифікації.
3. Тема дипломної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 121 «Інженерія програмного забезпечення» галузі знань 12 «Інформаційні технології» – створення, дослідження і реалізація моделей і програмних засобів.
4. Наукова новизна результатів, які очікуються, полягає в підвищенні точності визначення структури, оцінювання та оптимізації параметрів моделі складних об'єктів керування за рахунок застосування методів структурно-параметричної ідентифікації.
5. Оригінальність технічних рішень при розробці програмного засобу полягає в застосуванні міжпроцесорного підходу, завдяки чому стало можливим об'єднати керуючі модулі програмного забезпечення структурно-параметричної ідентифікації, що базуються на платформі Java, і засоби, які реалізують математичні обчислення, що використовують платформу MATLAB.
6. Практична цінність дослідження полягає в розробці програмного забезпечення для здійснення структурно-параметричної ідентифікації процесів крупного дроблення руд.
7. Оформлення дипломної роботи магістра виконано на сучасному рівні і відповідає вимогам, що пред'являються до робіт даної кваліфікації. Ступінь самостійності виконання досить висока.
8. Дипломна робота магістра в цілому заслуговує оцінки «_____», а сам автор – присвоєння кваліфікації «інженер-програміст».

Керівник дипломної
роботи магістра, д.т.н.,
проф. кафедри ПЗКС _____ В.І. Корнієнко

ДОДАТОК Г

РЕЦЕНЗІЯ

на дипломну роботу магістра на тему:

«Обґрунтування програмного забезпечення структурно-параметричної ідентифікації для АСУ процесом крупного дроблення руд»

студента групи 121м-16-1 Стадніченка Ігора Олеговича

Складні динамічні об'єкти управління (ОУ) мають нестаціонарні параметри, нелінійні залежності і стохастичні змінні, що обумовлює наявність у них різних динамічних режимів функціонування. До таких складних ОУ відносяться рухомі об'єкти, технологічні процеси рудопідготовки (дроблення або подрібнення). Пріоритетом наукових досліджень даних напрямків є розробка алгоритмів і створення систем автоматичного управління технологічними процесами і роботою устаткування. Для реалізації подібних систем необхідна розробка динамічних моделей керованих процесів, яка здійснюється за допомогою процедури їх ідентифікації.

Тому для оптимізації функціонування складних динамічних об'єктів актуальним є вирішення завдань ідентифікації та прогнозування, що дозволяє підвищити якість управління складними ОУ за рахунок підвищення точності оцінки їх стану та математичної моделі.

Наукова новизна отриманих результатів полягає в підвищенні точності визначення структури, оцінювання та оптимізації параметрів моделі складних об'єктів керування за рахунок застосування методів структурно-параметричної ідентифікації.

Студент Стадніченко І.О. досить добре розібрався в специфіці застосування різноманітних інформаційних технологій: мови програмування Java, пакета математичних обчислень MATLAB, і кошти організації межпрограммного взаємодії MATLAB JA Builder.

Беручи до уваги вище викладене, можна зробити висновок, що дана робота цілком відповідає вимогам, що пред'являються до кваліфікаційних робіт рівня магістра.

З огляду на наукову новизну і ступінь опрацювання компонентів даної роботи, в цілому автор заслуговує оцінки «_____», а також присвоєння кваліфікації «інженер-програміст».

Рецензент _____ _____