МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В. Н. КАРАЗІНА

ТЕХНОЛОГІЇ НЕЙРОННИХ МЕРЕЖ І НЕЧІТКОГО МОДЕЛЮВАННЯ В СИСТЕМАХ УПРАВЛІННЯ

Методичні вказівки
до проведення практичних занять для здобувачів вищої освіти
другого (магістерського) рівня за спеціальністю 174 «Автоматизація,
комп’ютерно-інтегровані технології та робототехніка»

У двох частинах

Частина 1

Електронний ресурс

Харків – 2025

УДК 658.5.011.56:658.589(075.5)
Т 38

Рецензенти:

Р. М. Тріщ – доктор технічних наук, професор, завідувач кафедри мехатроніки та електротехніки Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут»;

О. О. Литвин – доктор фізико-математичних наук, професор, завідувач кафедри ХТЛПіД Навчально-наукового інституту «Українська інженерно-педагогічна академія».

Затверджено до розміщення в мережі Інтернет рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 11 від 25 червня2025 року)

Т 38 Технології нейронних мереж і нечіткого моделювання в системах управління: методичні вказівки до проведення практичних занять для здобувачів вищої освіти другого (магістерського) рівня за спеціальністю 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». У 2-х ч. Ч. 1 [Електронний ресурс] / уклад. Г. І. Канюк, Т. Ю. Василець. – Харків : ХНУ імені В. Н. Каразіна, 2025. – (PDF 132 с.)

Методичні вказівки по проведенню практичних занять з дисципліни «Технології нейронних мереж і нечіткого моделювання в системах управління» спрямовані на відпрацювання отриманих знань і накопичення досвіду з розрахунку інтелектуальних систем управління. Метою практичних занять першої частини курсу є набуття студентами практичних навичок синтезу нейромережевих систем управління з використанням пакету системи MATLAB.

Видання призначене здобувачам вищої освіти освітнього ступеню «магістр», що навчаються за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» зі спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка».

УДК 658.5.011.56:658.589(075.5)

ЗМІСТ

Практичне заняття 1

ВИВЧЕННЯ GUI-ІНТЕРФЕЙСУ NNTOOL ПАКЕТУ NEURAL NETWORK TOOLBOX СИСТЕМИ MATLAB

1.1 Мета заняття

Ознайомлення з графічним інтерфейсом **NNTool** пакету **Neural Network Toolbox** в середовищі MATLAB. Вивчення основних операцій з використання інтерфейсу, таких як створення та налаштування нейронних мереж, вибір архітектури, підготовка вхідних та цільових даних для навчання мережі. Оволодіння навичками роботи з інструментами для навчання нейронних мереж та їх аналізу.

1.2 Опис GUI-інтерфейсу NNTool

Одним з найбільш зручних інструментів для розрахунку і проектування нейронних мереж є пакет прикладних програм (ППП) **Neural Network Toolbox**, що функціонує під управлінням ядра системи MATLAB. ППП **Neural Network Toolbox** служить засобом, який допомагає користувачеві розвивати методи проектування і розширює область застосування нейронних мереж.

До складу програмних продуктів фірми MathWorks, що функціонують під управлінням ядра системи MATLAB, включені різні інструментальні засоби організації діалогу з користувачем. Як правило, це GUI-інтерфейси. Не є виключенням і пакет по нейронним мережам **Neural Network Toolbox**, до складу якого входить інструментальний засіб **NNTool**. Цей графічний інтерфейс дозволяє, не звертаючись до командного вікна системи MATLAB, виконувати створення, навчання, моделювання, а також імпорт і експорт нейронних мереж і даних, використовуючи тільки інструментальні можливості GUI-інтерфейсу. Такі інструменти мають певні обмеження, зокрема, інтерфейс **NNTool** допускає роботу тільки з простими одношаровими і двошаровими нейронними мережами, але при цьому є виграш в часі і ефективності побудови мережі.

Виклик GUI-інтерфейсу **NNTool** можливий або командою **nntool** з командного рядка, або з вікна запуску додатків **Launch Pad** за допомогою опції **NNTool** з розділу **Neural Network Toolbox**. Після виклику на екрані терміналу з'являється вікно **Network/Data Manager** (Управління мережею/даними) (рис.1.1).

У вікні наявні такі області і кнопки:

Input Data (Вхідні дані) – набір даних, які подаються на вхід нейронної мережі під час навчання.

Target Data (Цільові дані) – послідовність цілей, тобто правильні (очікувані) виходи. Цільові дані використовуються для порівняння з передбаченими результатами мережі під час навчання.

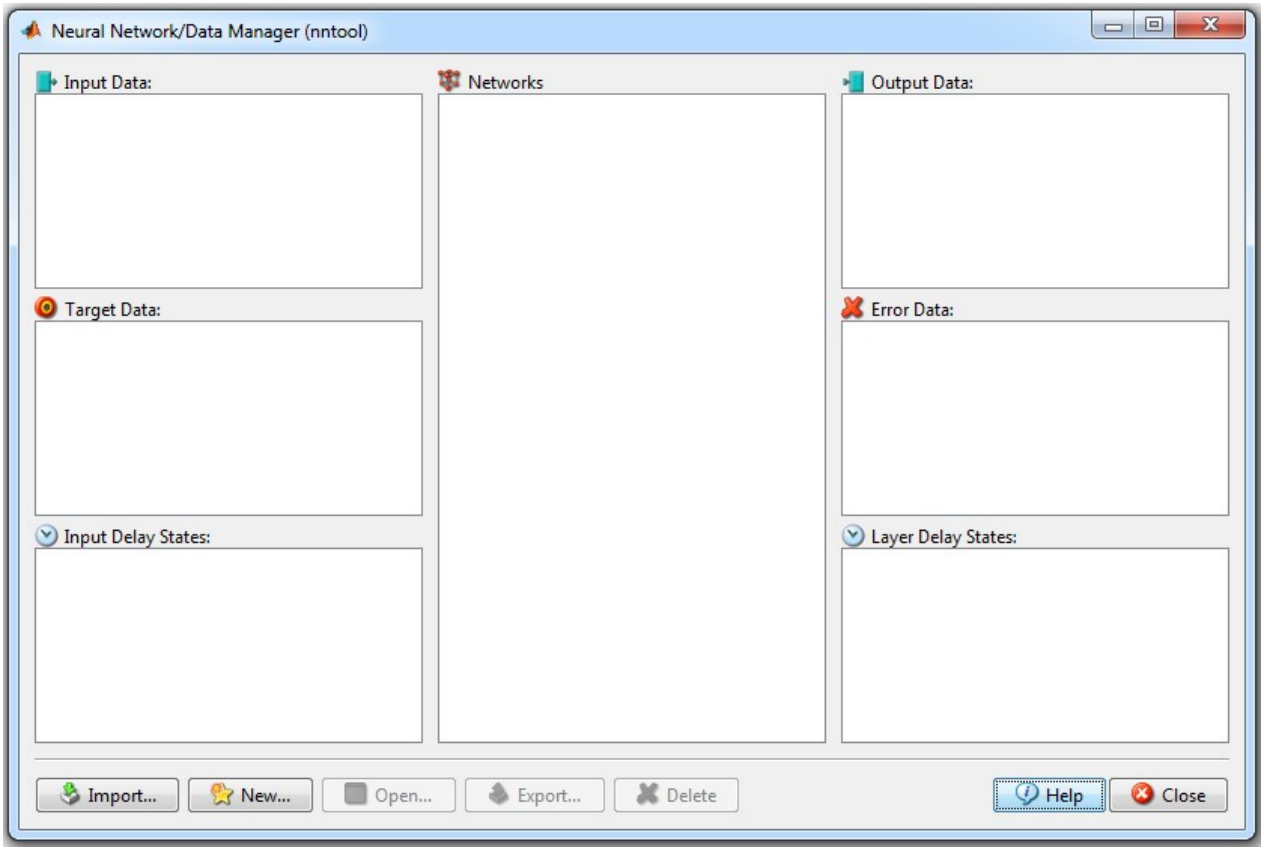


Рис.1.1. Вікно **Network/Data Manager** (Управління мережею/даними), що викликається функцією **nntool**

Networks (Мережі) – список створених нейромереж.

Outputs Data (Вихідні дані) – значення, які мережа генерує як результат обробки вхідних даних під час навчання або тестування. Вони порівнюються з **Target Data** (цільовими даними) для обчислення помилки.

Error Data (Дані помилки) – відображається помилка між вихідними даними мережі та цільовими даними.

Input Delay State (Стан затримки для вхідних даних) – у цьому полі зберігається інформація про затримки вхідних даних. Це важливий параметр для рекурентних нейронних мереж **(RNN)**, які мають елементи пам'яті та використовують затримки вхідних сигналів.

Layer Delay State (Стан затримки для шарів) – у цьому полі показується інформація про те, як внутрішні стани кожного шару мережі змінюються в залежності від попередніх вхідних даних і попередніх станів мережі. Цей параметр також важливий для рекурентних нейронних мереж, де мережа має внутрішні стани або пам'ять, яка зберігається між етапами обробки даних.

Import... (Імпорт...) **–** кнопка виклику вікна для імпорту або завантаження даних **Import to Network/Data Manager** (рис.1.2);

New... (Новий...) – кнопка виклику вікна **Create Network or Data**, яке має дві закладки:

- закладка **Network -** створення нової нейронної мереж (рис. 1.3):
- закладка **Data -** формування даних (рис.1.4);

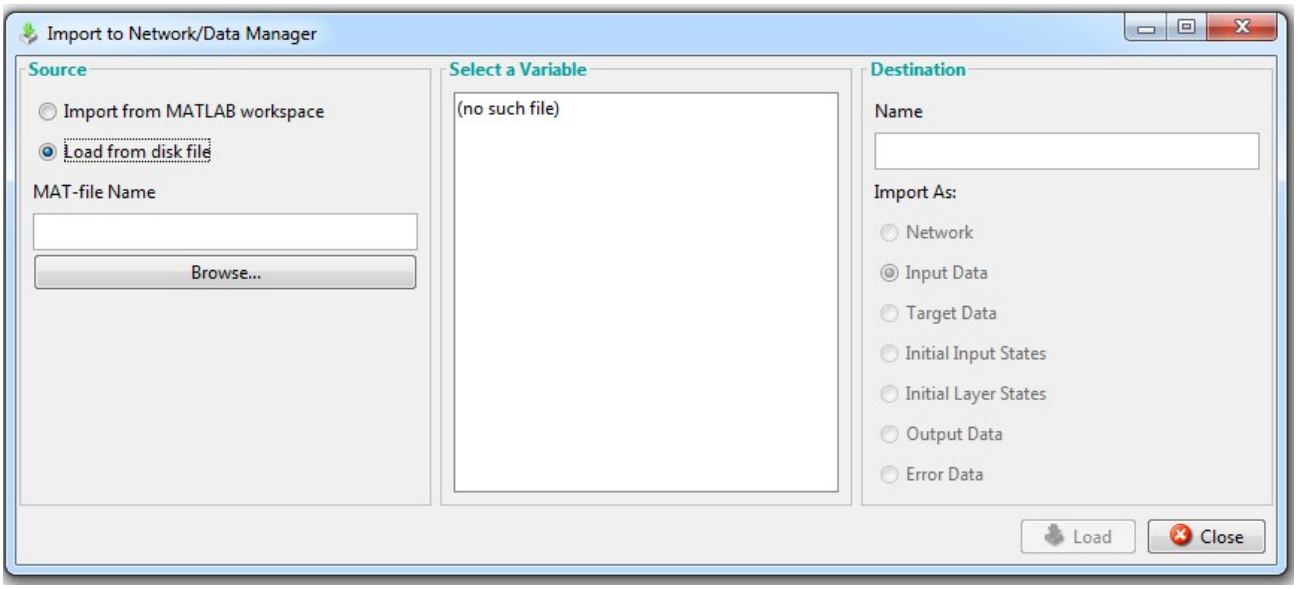


Рис.1.2. Вікно для імпорту або завантаження даних
Import to Network/Data Manager

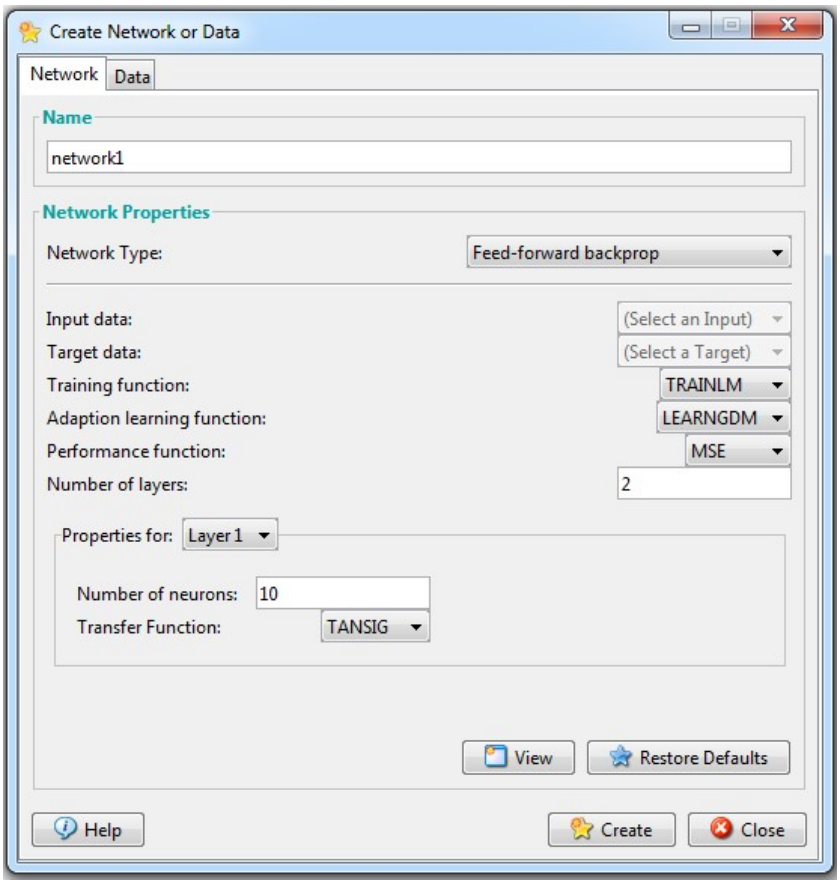


Рис.1.3. Вікно створення нової нейронної мережі або формування даних
Create Network or Data (закладка **Network)**

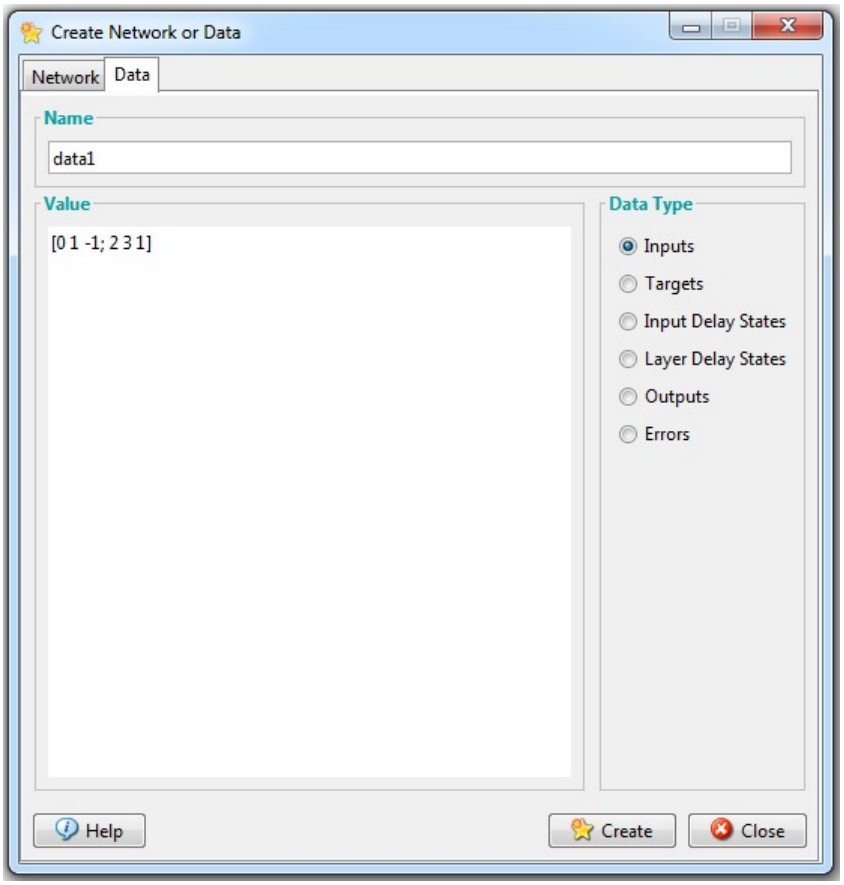


Рис.1.4. Вікно створення нової нейронної мережі або формування даних
Create Network or Data (закладка **Data)**

Open… (Відкрити…) – кнопка дозволяє відкрити раніше збережені об'єкти: мережу, дані.

Export... (Експорт…) – кнопка виклику вікна **Export from Network/Data Manager** для експорту даних в робочу область **Matlab** або запису даних у файл.

Delete (Видалити) – використовується для видалення мережі або обраних змінних.

Кнопка **Export...** стає активною тільки після створення і активізації даних, що відносяться до послідовностей входу, мети, виходу.

Кнопки **Open..** і **Delete..** стають активними після створення і активізації нейронної мережі.

Help (Допомога) – кнопка виклику вікна підказки **Network/Data Manage Help,** в якому описуються правила роботи з диспетчером **Neural Network/Data Manager** при створенні нейронної мережі.

Розглянемо більш детально призначення і способи роботи з вікнами.

Щоб створити нейронну мережу, необхідно виконати наступні операції.

- Сформувати послідовності входів і цілей (натиснути на кнопку **New…**, у вікні **Create Network or Dana**, що відкрилось, вибрати закладку **Data** (див. рис.1.4), або завантажити їх з робочої області системи MATLAB або з файлу (натиснути на кнопку **Import…**, відкриється вікно **Import to**

Network/Data Manager (див. рис.1.2), завантажити усі необхідні дані).

- Створити нову нейронну мережу (натиснути на кнопку **New…**, у вікні **Create Network or Dana**, що відкрилось, вибрати закладку **Network**), або завантажити її з робочої області системи MATLAB або з файлу (кнопка **Import…**, вікно **Import to Network/Data Manager**).

Вікно **Import to Network/Data Manager.** Це вікно (див. рис.1.2) включає 3 поля.

Source (Джерело) – поле для вибору джерела даних. Це або робоча область системи MATLAB (кнопка вибору **Import from MATLAB Workspace),** або файл (кнопка вибору **Load from disk file**).

Якщо вибрана перша кнопка, то в полі **Select a Variable** можна бачити всі змінні робочої області і, вибравши одну з них, наприклад **Inp_Train_N**, можна призначити її в поле **Destination** (Призначення) як послідовність входу **Inputs** (Входи).

Якщо вибирається кнопка **Load from disk file**, то активізується поле **MAT-file Name** і кнопка **Browse**, що дозволяє почати пошук і завантаження файлу з файлової системи.

Вікно **Create Network or Data,** закладка **Network** (рис.1.3). Це вікно включає поля для завдання параметрів створюваної мережі. Залежно від типу мережі кількість полів і їх назви змінюються.

Звернемося до опису полів.

Name (Ім'я) – стандартне ім'я мережі **network1**, що присвоюється GUI-інтерфейсом **NNTool**; в процесі створення нових мереж порядковий номер змінюватиметься автоматично. Користувач може задати будь-яке ім'я.

Network Type (Тип мережі) – список мереж, доступних для роботи з інтерфейсом **NNTool**.

Input Data (Вхідні дані)– набір вхідних значень, які подаються на нейронну мережу ;

Target Data (Цільові дані) – послідовність цілей, тобто правильні відповіді, які нейромережа має навчитися прогнозувати.

Training function (Функція навчання) – список функцій навчання.

Adaption learning function (Функції настройки для режиму адаптації) – список функцій настройок.

Performance function (Функція якості навчання) – список функцій оцінки якості навчання.

Number of layers (Кількість шарів) – кількість шарів нейронної мережі.

Properties for (Властивості) – список шарів: Layer 1 (Шар 1), Layer 2 (Шар 2).

Number of neurons (Кількість нейронів) – кількість нейронів в шарі.

Transfer function (Функція активації) – функція активації шару.

Вікно **Create Network or Data**, закладка **Data** (рис.1.4). Це вікно включає 2 області редагування тексту для запису імені даних (область **Name**), що вводяться, і

введення самих даних (область **Value**), а також 6 кнопок для вказівки типу даних, що вводяться.

Розрізняють наступні типи даних:

Inputs (Входи) – послідовність значень входів;

Targets (Цілі) – послідовність значень мети;

Input Delay States (Стани ЛЗ входу) – початкові умови лінії затримки на вході;

Layer Delay States (Стани ЛЗ шару) – початкові умови лінії затримки в шарі;

Outputs (Виходи) – послідовність значень виходу мережі;

Errors (Помилки) – різниця значень цілей і виходів.

Як правило, користувач задає тільки послідовності входу і мети, тобто типи даних **Inputs** і **Targets**. При цьому слід пам'ятати, що при адаптації нейронної мережі дані повинні бути представлені у вигляді масиву осередків.

Вікно **Export or Save from Network/Data Manager** (рис. 1.5). Це вікно дозволяє передати дані з робочої області GUI–інтерфейсу **NNTool** в робочу область системи MATLAB або записати їх у вигляді файлу на диску. В даному випадку вибрана змінна **Neural_Model,** яка належить до класу **network object** і описує нейронну мережу. Після того, як ця змінна експортована в робочу область, можна, наприклад, побудувати модель нейронної мережі в системі Simulink за допомогою оператора **gensim**.

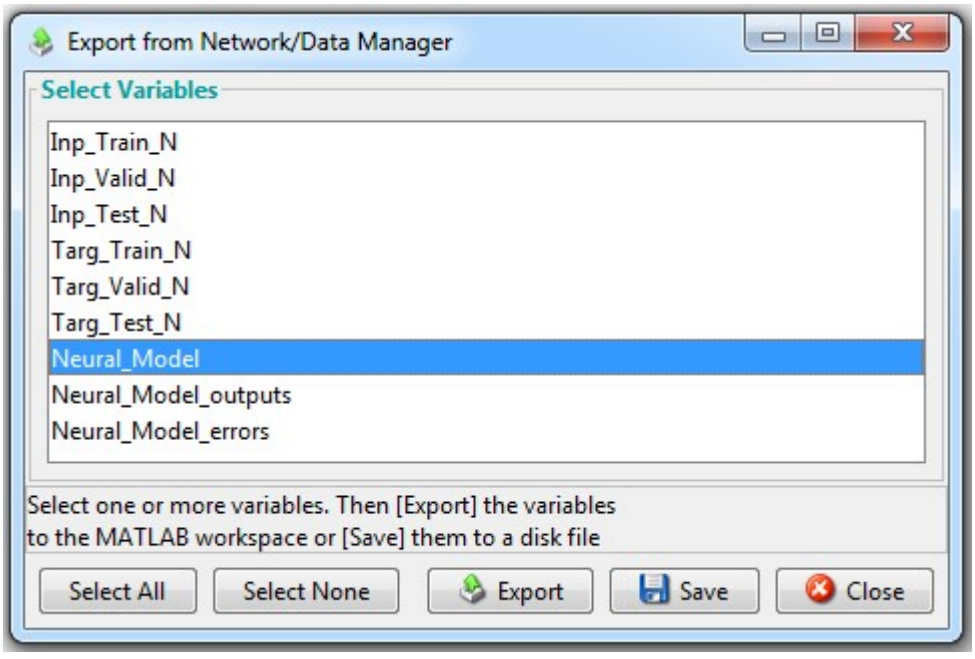


Рис.1.5. Вікно для експорту або запису даних у файл
Export from Network/Data Manager

Діалогова панель **Network: Neural Model** (рис. 1.6). Вона відкривається при натисканні на кнопку **Open** тільки у тому випадку, коли у вікні **Neural Network/Data Manager** виділена створена мережа і кнопка **Open** стає активною.

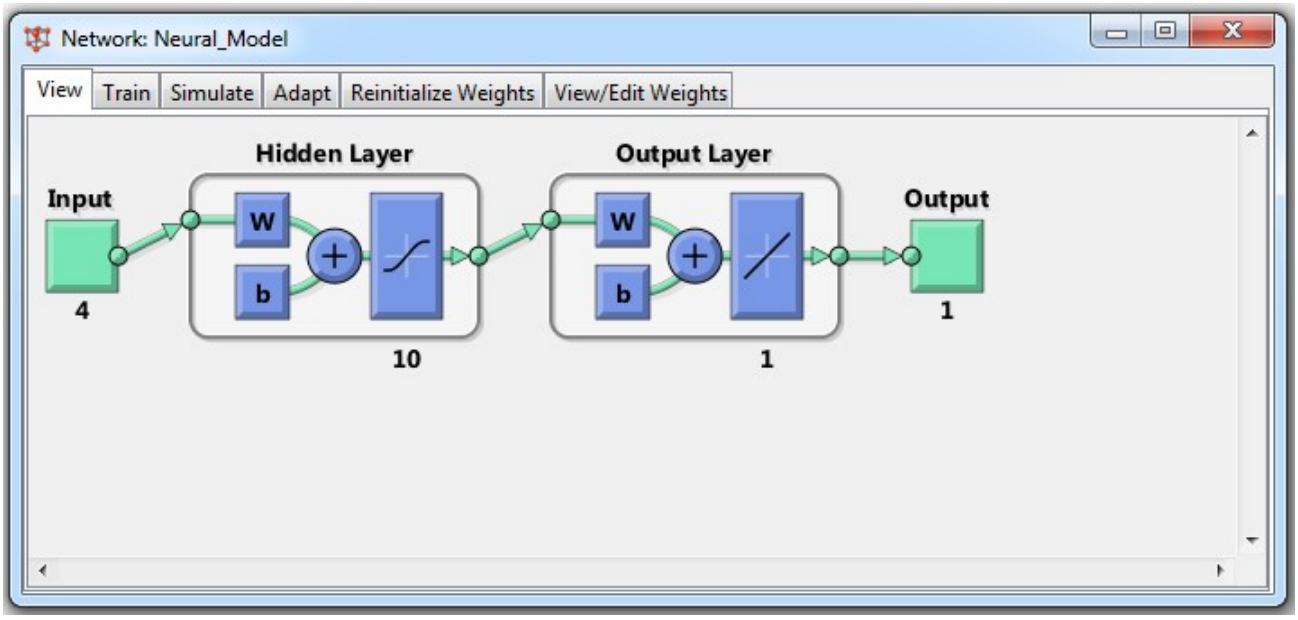


Рис.1.7. Діалогова панель **Network: Neural Model**

Панель має 6 закладок:

1. **View** (Переглянути). Заладка **View** дозволяє переглядати архітектуру нейронної мережі**.** Після вибору цієї опції, MATLAB відобразить візуалізацію мережі, включаючи:

- кількість шарів (вхідний, прихований, вихідний);
- кількість нейронів в кожному шарі;
- структуру та з'єднання між шарами (як нейрони взаємодіють один з одним).

2. **Train** (Навчати). Закладка **Train** використовується для навчання нейронної мережі. У закладки **Train** є в свою чергу 2 закладки: **Train Info** (рис. 1.8) і **Train Parameters** (рис.1.9).

Закладка **Train Info** має два поля: **Training Data** і **Training Results**.

Поле **Training Data** (Навчальні дані). Це поле показує вихідні (вхідні та цільові) дані, які використовуються для навчання нейронної мережі.

- **Inputs** (Вхідні дані) – це набір даних, які подаються на вхід нейронної мережі під час навчання.
- **Targets** (Цільові значення) – це правильні (очікувані) відповіді для кожного вхідного зразка.

Ці дані використовуються під час тренування для порівняння передбачених виходів із правильними відповідями.

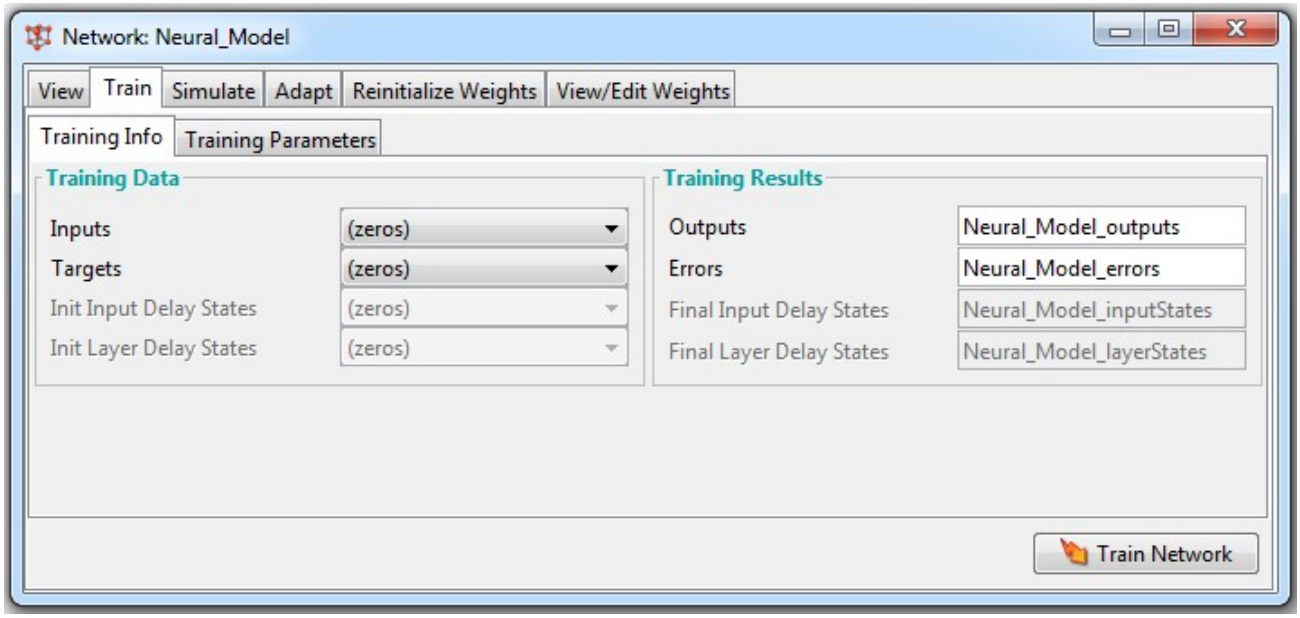


Рис. 1.8. Закладка **Train Info** панелі **Network: Neural Model**

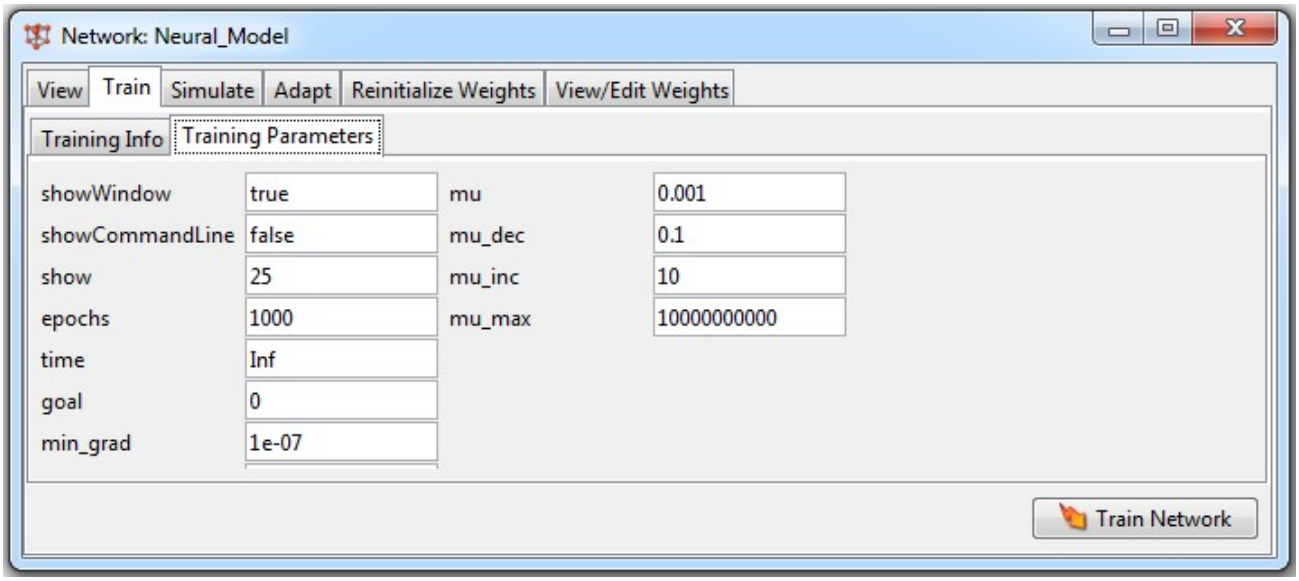


Рис. 1.9. Закладка **Train Parameters** панелі **Network: Neural Model**

Поле **Training Results** (Результати навчання). Це поле показує результати роботи мережі після тренування.

- **Outputs** (Вихідні дані) – це передбачені значення, які мережа згенерувала для кожного вхідного прикладу. Вони порівнюються з цільовими значеннями (Targets), щоб визначити точність навчання.
- **Error** (Помилка) – це різниця між цільовими (Targets) і отриманими (Outputs) значеннями.Помилка використовується для оцінки точності навчання. У багатьох випадках використовується середньоквадратична помилка (MSE – Mean Squared Error), яка показує, наскільки передбачені результати відхиляються від правильних. Якщо Error є

малою, це означає, що мережа добре навчилася. Якщо Error великий, можливо, потрібно змінити параметри навчання або структуру мережі.

Закладка **Train Parameters** дає змогу налаштувати параметри навчання для нейронної мережі, які визначають поведінку процесу навчання.

Значення параметрів:

- **show Window** (Показувати вікно). Цей параметр визначає, чи має MATLAB показувати вікно прогресу під час навчання. Якщо цей параметр увімкнений, під час навчання, то зможна бачити вікно з відображенням ходу навчання, включаючи інформацію про кількість епох, поточну помилку та інші показники.
- **show CommandLine** (Показувати командний рядок). Цей параметр визначає, чи має MATLAB виводити інформацію про хід навчання в командному рядку. Якщо увімкнений, під час кожної епохи навчання буде виводитися статистика, наприклад, поточний рівень помилки та інші метрики.
- **show** (Показувати). Цей параметр дозволяє вибрати рівень деталізації виведеної інформації. Цей параметр дозволяє зручніше налаштовувати кількість виведених даних під час навчання. Наприкла, якщо show = 25, то це означає, що на кожній епосі навчання MATLAB показуватиме інформацію про перші 25 зразків з навчальних даних (вхідні дані, цільові значення, виходи мережі, помилки).
- **epochs** (Кількість епох). Це максимальна кількість епох (циклів) навчання, після яких процес буде припинений, якщо досягнуто мети (**goal**) або інший критерій зупинки. Кожна епоха є одним циклом проходу по всіх навчальних даних. Наприклад**:** якщо встановити epochs = 1000, мережа буде тренуватися максимум 1000 епох, якщо інші умови зупинки не будуть виконані раніше**.**
- **goal** (Мета). Цей параметр визначає бажану мету для середньоквадратичної помилки (Mean Squared Error, MSE) або іншої функції помилки, яку мережа повинна досягти під час навчання. Якщо помилка мережі стає меншою або рівною цій меті, навчання припиняється. Наприклад**:** якщо встановити goal = 0.01, мережа припинить навчання, коли середня помилка на тестових даних стане ≤ 0.01.
- **min_grad** (Мінімальний градієнт). Цей параметр визначає мінімальний градієнт, при якому навчання припиняється. Якщо значення градієнта (швидкість зміни ваг) падає нижче цього порогу, це вказує на те, що навчання майже завершене, і алгоритм може припинити процес навчання, щоб уникнути надмірних обчислень. Наприклад**:** Якщо встановити min_grad = 1e-6, навчання припиниться, якщо величина градієнта стане меншою ніж 1e-6.
- **max_fail** (Максимальна кількість неуспішних спроб). Цей параметр визначає максимальну кількість неуспішних спроб (епох), коли мережа не покращує свої результати (наприклад, помилка не зменшується), після

яких навчання буде припинено. Нап**риклад**: Якщо встановити max_fail = 6, то навчання припиниться після шести епох, якщо помилка не покращується.

- **mu** (Параметр **mu**). Цей параметр визначає початкову величину для параметра адаптивного навчання (**mu**)**,** який використовується в методі Левенберга-Маркуарда (Levenberg-Marquardt) та інших адаптивних методах. Це важливий параметр для керування швидкістю коригування ваг під час навчання. Наприклад**:** значення mu = 0.01 може визначати початкову швидкість адаптації, що впливає на швидкість навчання.
- **mu_dec** (Зменшення **mu**). Цей параметр визначає, наскільки зменшується параметр **mu** після кожної епохи навчання, коли помилка покращується. Нап**риклад**: Якщо встановити mu_dec = 0.1, то на кожній ітерації, коли помилка зменшується, значення **mu** буде зменшуватися в 10 разів.
- **mu_inc** (Збільшення **mu**). Цей параметр визначає, на скільки збільшується **mu** під час навчання, якщо помилка не покращується. Наприклад: якщо встановити mu_inc = 10, параметр **mu** буде збільшуватися в 10 разів при кожному погіршенні результату.
- **mu_max** (Максимальне **mu**). Цей параметр встановлює максимальне значення для параметра **mu**. Це допомагає обмежити його зростання і запобігає надмірному збільшенню швидкості навчання, що може привести до нестабільності. Нап**иклад**: Якщо встановити mu_max = 100, значення **mu** не буде перевищувати 100.

*Загальний опис **Train Parameters** :*

- **Show Window / Show CommandLine / Show**: Налаштування для виведення інформації під час навчання.
- **Epochs / Goal / Min_grad / Max_fail**: Параметри, які визначають, коли припинити навчання.
- **Mu / Mu_dec / Mu_inc / Mu_max**: Параметри для налаштування адаптивного методу навчання (як, наприклад, метод Левенберга-Маркуарда).

Ці параметри дають змогу налаштувати настройку процесу навчання для досягнення кращих результатів або для оптимізації часу навчання.

3. **Simulate** (Моделювати). Закладка **Simulate** дозволяє симулювати роботу вже навченого нейронної мережі на нових даних, тобто отримувати прогнози або результати мережі на тестових або нових вхідних даних.

Використовується для:

- перевірки якості роботи мережі після навчання;
- прогнозування результатів на нових вхідних даних;
- оцінки точності моделі

4. **Adapt** (Адаптувати). Закладка **Adapt** (рис.3.11) дає змогу адаптувати мережу до нових даних, не проводячи повного циклу навчання. Це може бути корисно,

коли у вас є нові дані, і ви хочете "оновити" мережу, навчаючи її лише на частині даних або використовуючи певний підхід до навчання, як у онлайн навчанні**.**

Адаптація часто використовується в ситуаціях, коли модель потребує швидкого реагування на зміни у даних або коли необхідно адаптувати мережу до нових умов без перепідготовки з нуля

5. **Reinitialize Weight** (Перевстановити ваги). Ця опція дозволяє повернути початкові значення ваг нейронної мережі. Після її вибору вага всіх нейронів буде перезаписана випадковими значеннями, або значеннями, визначеними згідно з методами ініціалізації, встановленими для вашої мережі (наприклад, випадкові значення з певним розподілом, згідно з вибраною стратегією).

6. **View/Edit Weight** (Переглянути/Редагувати ваги). Ця опція відкриває вікно, де можна переглядати та редагувати ваги нейронної мережі вручну**.**

1.3 Контрольні питання по темі заняття

1. Яке призначення пакету прикладних програм **Neural Network Toolbox** системи MATLAB?
2. Перечисліть функції, які дозволяє виконувати інтерфейс **NNTool**.
3. Які обмеження має інтерфейс **NNTool**?
4. Як виконується виклик GUI-інтерфейсу **NNTool?**
5. Поясніть призначення кнопок вікна **Neural Network/Data Manager .**
6. Перечисліть операції, які операції необхідно виконати, щоб створити нейронну мережу.
7. Які дані вводяться за допомогою вікна **Create Network or Data** (закладка **Data)**?
8. Перечисліть і поясніть призначення полів, які має вікно **Create Network or Data** (акладка **Network)**.
9. Які поля має вікно **Import to Network/Data Manager**?
10. Яке призначення має вікно **Export from Network/Data Manager**?
11. Які закладки має панель **Network: Neural Model ?**

Практичне заняття 2

ПОБУДОВА НЕЙРОМЕРЕЖЕВИХ МОДЕЛІЙ БАГАТОМАСОВИХ ЕЛЕКТРОМЕХАНІЧНИХ СИСТЕМ З ВИКОРИСТАННЯМ GUI-ІНТЕРФЕЙСУ NNTOOL

2.1 Мета заняття

Набуття практичних навичок по застосовувати графічного інтерфейсу **NNTool** для вирішення конкретних інженерних задач - побудови нейромережевих моделей багатомасових електромеханічних систем, що передбачає вибір відповідної архітектури нейронних мереж, налаштування параметрів, проведення навчання та оцінки результатів.

2.2 Методика синтезу нейромережевої моделі двомасової електромеханічної системи

Особливості роботи з відповідними вікнами розглянемо на прикладі створення нейронної мережі для вирішення завдання ідентифікації двомасової електромеханічної системи управління електроприводом механізмом підйому мостового крана з урахуванням кінцевої жорсткості підйомного канату. Потім оцінимо точність отриманої нейромоделі за допомогою порівняння модельних значень із значеннями, отриманими шляхом моделювання двомасової електромеханічної системи в Simulink.

У якості привідного двигуна застосовано двигун постійного струму незалежного збудження. Якірна обмотка живиться від тиристорного підсилювача потужності. Систем управління електроприводом побудована за принципом підлеглого регулювання і містить два контури: контур регулювання струму і контур регулювання швидкості із зворотним зв’язком по ЕДС двигуна. Замкнений контур струму настроєний на модульний оптимум, а контур ЕДС – на симетричний оптимум.

Двомасової системи підлеглого регулювання швидкості з внутрішнім контуром струму і зовнішнім контуром ЕРС описується системою диференціальних рівнянь (2.1). Призапису системи прийняті наступні позначення: $\omega_{\text{д}}$, $\omega_{\text{м}}$ – кутові швидкості двигуна і механізму; $M_{\text{д}}$ – обертаючий момент двигуна; $M_{\text{ст}}$ – момент статичного навантаження; $M_{\text{пр}}$ – момент пружності; $P = \dfrac{dM_{\text{д}}}{dt}$ – ривок; $U_{\text{зе}}$, $E_{\text{зе}}$ – напруга і ЕДС завдання; $U_{\text{зз}E}$ – напруга зворотного зв’язку по ЕДС; U_{ie} – вихідна напруга інтегратора ПІ регулятора ЕДС; $T_{\mu\text{т}}$ – мала некомпенсуєма постійна часу контуру струму; T_a – електромеханічна постійна часу електродвигуна; $k_{\text{д}}$ – коефіцієнт посилення

двигуна: $k_{\text{д}} = \dfrac{1}{k\Phi_{\text{н}}}$; $k\Phi_{\text{н}}$ – добуток конструктивного коефіцієнта і магнітного потоку двигуна; $k_{\text{т}}$ – коефіцієнт посилення зворотного зв'язку по току; $k_{\text{н}}$ – коефіцієнт посилення зворотного зв'язку по напрузі; β – коефіцієнт внутрішнього в'язкого тертя; c_{12} – коефіцієнт жорсткості передачі; $J_{\text{дΣ}}$ – сумарний момент інерції двигуна і жорстко пов'язаних з ним елементів системи; $J_{\text{м}}$ – момент інерції механізму; $k_{\text{пе}}$, k_{ie} – коефіцієнти підсилення пропорційної і інтегральної частин регулятора ЕДС.

$$\begin{cases} \dfrac{d\omega_{\text{м}}}{dt} = \dfrac{1}{J_{\text{м}}}M_{\text{пр}} + \dfrac{\beta}{J_{\text{м}}}\omega_{\text{д}} - \dfrac{\beta}{J_{\text{м}}}\omega_{\text{м}} - \dfrac{1}{J_{\text{м}}}M_{\text{ст}}; \dfrac{dM_{\text{пр}}}{dt} = c_{12}\omega_{\text{д}} - c_{12}\omega_{\text{м}}; \\ \dfrac{d\omega_{\text{д}}}{dt} = \dfrac{1}{J_{\text{дΣ}}}M_{\text{д}} - \dfrac{1}{J_{\text{дΣ}}}M_{\text{пр}} - \dfrac{\beta}{J_{\text{дΣ}}}\omega_{\text{д}} - \dfrac{\beta}{J_{\text{дΣ}}}\omega_{\text{м}}; \\ \dfrac{dM_{\text{д}}}{dt} = \dfrac{1}{k_{\text{д}}}P; \\ \dfrac{dP}{dt} = \dfrac{k_{\text{ре}}}{2T_{\mu T}^2 k_{\text{т}}}E_{\text{зе}} - \dfrac{k_{\text{ре}}}{2T_{\mu T}^2 k_{\text{т}}}U_{\text{ззе}} - \dfrac{k_{\text{д}}}{2T_{\mu T}^2}M_{\text{д}} - \dfrac{1}{T_{\mu T}}P + \dfrac{1}{T_{\mu T}^2 k_{\text{т}}}U_{\text{ie}}; \\ \dfrac{dE_{\text{зе}}}{dt} = \dfrac{1}{T_a}U_{\text{зе}} - \dfrac{1}{T_a}E_{\text{зе}}; \\ \dfrac{dU_{\text{ззе}}}{dt} = \dfrac{k_{\text{н}}}{T_a k_{\text{д}}}\omega_{\text{д}} - \dfrac{1}{T_a}U_{\text{ззе}}; \\ \dfrac{dU_{\text{ie}}}{dt} = k_{\text{ie}}E_{\text{зе}} - k_{\text{ie}}U_{\text{ззе}}. \end{cases} \tag{2.1}$$

Значення параметрів $k_{\text{пе}}$, k_{ie} – можуть бути отримані з виразу передатної функції регулятора ЕДС:

$$W_{\text{ре}}(p) = \frac{T_{\text{м}}k_{\text{т}}}{2T_{\text{це}}k_{\text{н}}R_{\Sigma}} \cdot \frac{4T_{\text{це}}p + 1}{4T_{\text{це}}p} = \frac{T_{\text{м}}k_{\text{т}}}{2T_{\mu E}k_{\text{н}}R_{\Sigma}} + \frac{T_{\text{м}}k_{\text{т}}}{8T_{\mu E}^2 k_{\text{н}}R_{\Sigma}}\frac{1}{p} = k_{\text{пе}} + \frac{k_{\text{ie}}}{p},$$

$$k_{\text{пе}} = \frac{T_{\text{м}}k_{\text{т}}}{2T_{\mu E}k_{\text{н}}R_{\Sigma}},\ k_{\text{ie}} = \frac{T_{\text{м}}k_{\text{т}}}{8T_{\mu E}^2 k_{\text{н}}R_{\Sigma}}.$$

$T_{\text{м}}$ – електромеханічна постійна часу системи ТП-Д; при записі системи (2.1) замість $T_{\text{м}}$ записано вираз $T_{\text{м}} = J_{\Sigma}\dfrac{R_{\Sigma}}{\left(k\Phi_{\text{н}}\right)^2}$, де $J_{\Sigma} = J_{\Sigma\text{д}} + J_{\text{м}}$ - сумарний момент інерції електроприводу: $J_{\Sigma} = J_{\Sigma\text{д}} + J_{\text{м}}$.

Алгоритмічна схема двомасової системи зображена на рис.2.1.

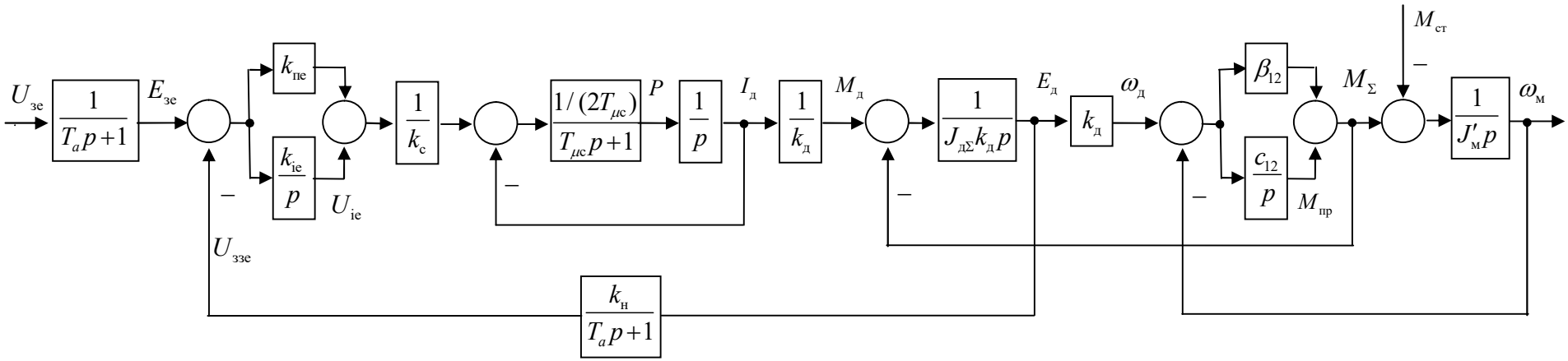


Рис.2.1. Алгоритмічна схема двомасової системи.

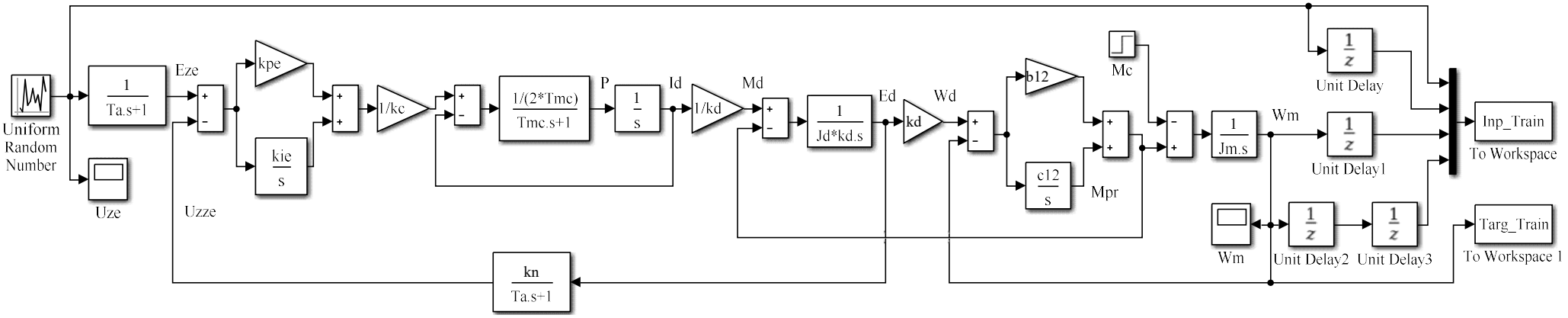


Рис.2.2. Схема моделі двомасової системи, розроблена в Simulink

Перш ніж приступити до розробки нейрмережевої моделі двомасової системи сформуємо послідовності значень входів **Inputs** і цілей **Targets**, а також контрольні **Validation** і тестові **Testing** послідовностей. Доцільно сформувати ці послідовності в робочій області системи MATLAB, а потім імпортувати їх в робочу область інтерфейсу **NNTool**. Для цього створимо у вікні Simulink схему моделі системи (рис.2.2), а в редакторові-відладчику **m-файлів** сформуємо файл початкових даних системи (рис.2.3) і збережемо його на диску під іменем «Dat_Kran». За допомогою команд меню **Run** введемо початкові дані в робочу область MATLAB.

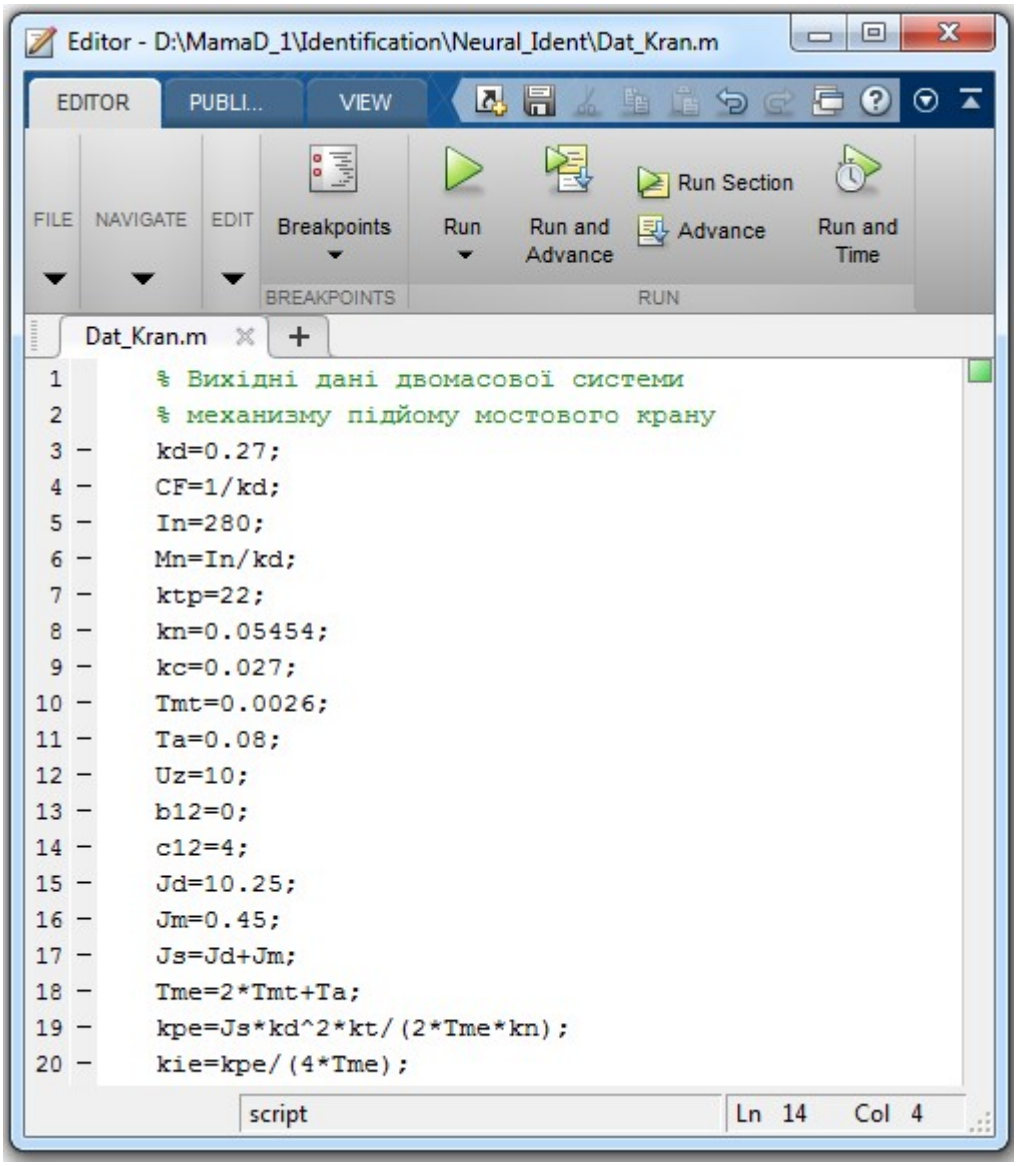


Рис. 2.3. Файл вихідних даних системи

Для побудови моделі динамічного об'єкту, необхідно вхідну послідовність задати на основі поточного значення вхідного сигналу об'єкту, і ряду попередніх значень вхідного і вихідного сигналів. Порядки затримок по вхідному і вихідному сигналах заздалегідь вибираються на підставі апріорних значень про об'єкт ідентифікації (якщо такі є) і досвіду дослідника, а потім уточнюються експериментально в процесі побудови моделі об'єкту.

У даному прикладі сформуємо вхідну послідовність (**Inputs**) на основі поточного значення вхідного сигналу системи $U_{зе}(k)$ і вхідного сигналу, затриманого на один шах дискретності $U_{зе}(k-1)$, а також два затриманих на один і два кроки вихідних сигналів, тобто $\omega_{м}(k-1)$ і $\omega_{м}(k-2)$ відповідно. Вихідним сигналом (**Targets**) є швидкість механізму $\omega_{м}(t)$.

Для формування затриманих сигналів використовуємо блоки **Unit Delay** (див. рис. 2.2). У вікні завдання параметрів блоку задамо наступні параметри:

Initial condition [початкове значення вхідного сигналу]. Залишимо значення 0, встановлене за умовчанням.

Sample time [такт дискретності]. При завданні величини такту дискретності необхідно врахувати наступне. Успіх тренування нейронної мережі в значній мірі залежить від довжини навчальної вибірки $N_{в}$ і такту дискретності Δt, що визначає інтервал між двома послідовними моментами знімання даних. При збільшенні Δt знижується точність обчислення і різниця між помилкою навчання і помилкою, отриманою на контрольній і тестовій множині. Зменшення Δt викликає необхідність відповідного збільшення $N_{в}$ і, як наслідок, значно збільшується час тренування мережі, при цьому істотного зниження помилки навчання мережі не спостерігається. Для даного прикладу встановимо такт дискретності $\Delta t = 0,05c$.

У схемі моделі рис.2.2 використано два блоки **To Workspase** і **To Workspase1** для запису послідовності значень входів **Inputs (**в даному випадку $U_{зе}(k)$, $U_{зе}(k-1)$, $\omega_{м}(k-1)$, $\omega_{м}(k-2)$) і цілей **Targets** (швидкості механізму $\omega_{м}(t)$) відповідно в робочу область системи MATLAB. У вікні завдання параметрів блоків слід задати наступні параметри:

Variable name [ім'я змінної]. Ім'я масиву, в який записуватимуться дані. Встановимо **Inp_Train** і **Targ_Train** відповідно.

Sample time [такт дискретності]. Встановимо таке ж значення, як і для блоку **Unit Delay**, тобто $\Delta t = 0,05c$.

Save format [формат даних, що зберігаються]. З пропонованого списку вибираємо формат **Array** – масив. Дані зберігаються як масив, в якому число рядків визначається числом розрахункових точок в часі, а число стовпців – розмірністю вектора, що подається на вхід блоку. У даному прикладі масив **Inp_Train** містить 4 стовпці, а масив **Inp_Train –** один стовпець.

Решту параметрів блоку залишимо без зміни.

Як джерело сигналу в схемі рис.2.2 використаний блок **Uniform Random Number** – джерело сигналу з рівномірним розподілом. У вікні завдання параметрів даного блоку слід задати наступні параметри:

Minimum [мінімальний рівень сигналу].

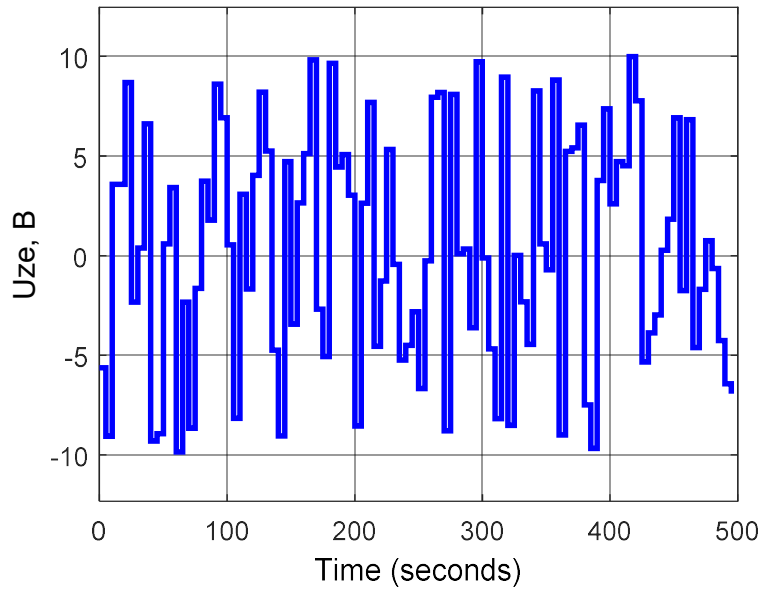
Maximum [максимальний рівень сигналу].

Мінімальна і максимальна величина напруги завдання $U_{зе}(k)$ вибирається при

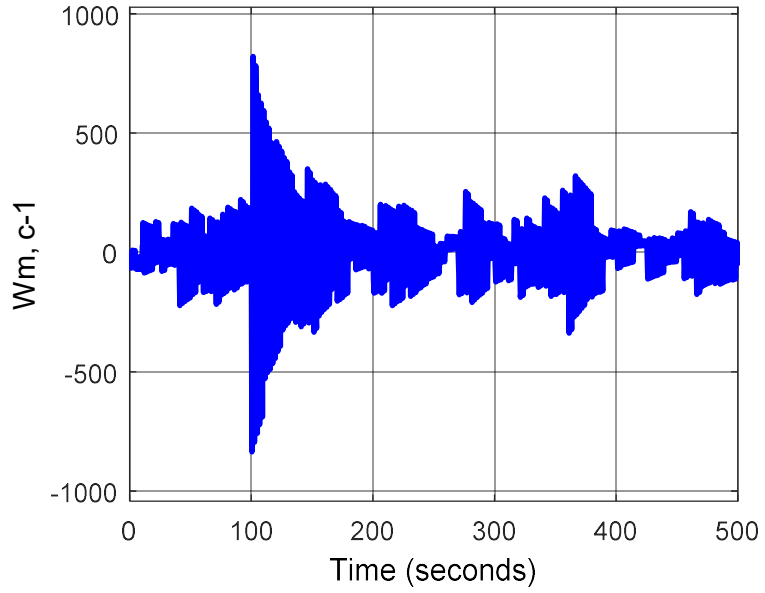
синтезі системи підлеглого регулювання. Приймаємо дані параметри $+10B$ і $-10B$ відповідно.

Sample time [такт дискретності]. Для отримання представницької вибірки необхідно правильно задати значення інтервалу ідентифікації, тобто тривалість стрибків завдань. Величина їх залежить від параметрів об'єкту управління, оскільки тренувальні дані повинні містити тільки фази прискорень. У даному прикладі приймаємо величину **Sample time** рівною $5c$.

Задамо у вікні **Simulink** моделі системи час моделювання $500c$ і промоделюємо систему за допомогою команди меню **Simulation\Start**. Масиви **Inp_Train** і **Inp_Train** міститимуть 10001 рядок. Графіки вхідного $U_{зе}(k)$ і вихідного $\omega_{м}(t)$ сигналів системи показані на рис.2.4.



а)



б)

Рис.2.4. Графіки вхідного $U_{зе}(k)$ (а) і вихідного $\omega_{м}(t)$ (б) сигналів системи

Аналогічно сформуємо контрольні **Validation** і тестові **Testing** послідовності. Для цього в блоках **To Workspase** і **To Workspase1** (рис.2.2) в полі **Variable name** змінимо імена масивів, в який записуватимуться дані на **Inp_Valid** і **Targ_Valid**, а потім на **Inp_Test** і **Targ_Test** відповідно. Час моделювання системи встановимо в обох випадках $250c$. Після моделювання системи масиви **Inp_Valid**, **Targ_Valid**, **Inp_Test** і **Targ_Test** міститимуть 5001 рядок.

У вікні **Workspase** системи MATLAB виводяться імена всіх змінних, які знаходяться в робочій області, у тому числі і масивів **Inp_Train** і **Inp_Train, Inp_Valid**, **Targ_Valid**, **Inp_Test** і **Targ_Test** .

Збережемо всі масиви у файлі з назвою **Dat_Inp_Targ** і розмістимо файл в папці **Neural_Ident** каталога **Identification** на диску **E**. Команда **save** дозволяє зберегти вміст робочої області в двійковому MAT-файлі, який можна потім викликати командою **load**. Для збереження масивів в командному вікні системи MATLAB слід задати команду:

```
save E:\Identification\Neural_Ident\Dat_Inp_Targ Inp_Train Inp_Valid Inp_Test
Targ_Train Targ_Valid_ Targ_Test
```

Для того, щоб можна було використовувати отримані масиви при навчанні нейронної мережі, їх необхідно спочатку транспонувати. З цією метою може бути використаний редактор-відладчик **m**-файлів системи MATLAB (рис.2.5).

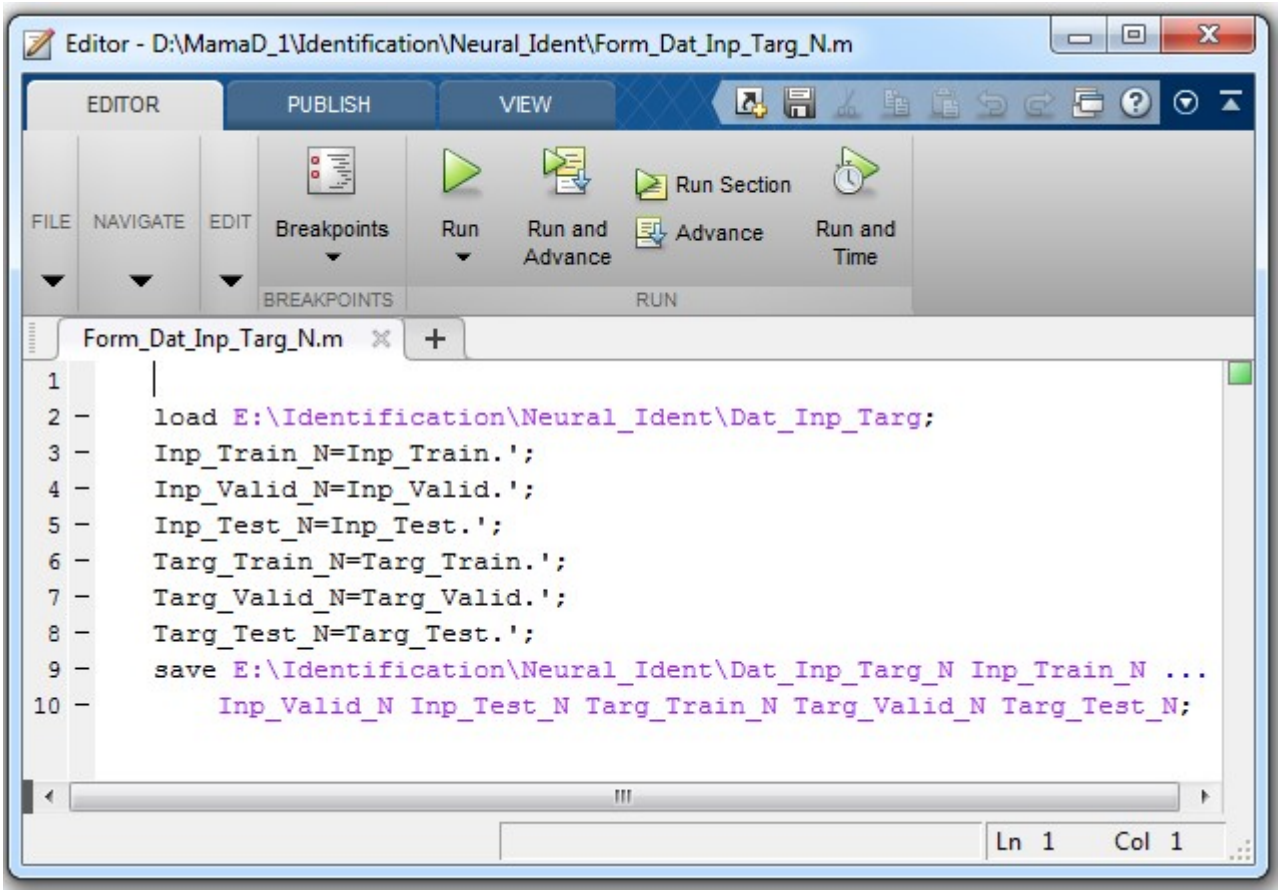


Рис.2.5. Текст m-файлу формування масивів входів і цілей використовуваних для навчання нейронної мережі

У вікні відладчика приведений текст **m**-файлу формування транспонованих масивів навчальних, перевірочних і тестових даних **Inp_Train_N**, **Targ_Train_N**, **Inp_Test_N** і **Targ_Test_N**, **Inp_Valid_N**, **Targ_Valid_N**. Вказані масиви збережені у файлі **Dat_Inp_Targ_N**.

Масиви **Inp_Train_N, Inp_Test_N Inp_Valid_N** містять по 4 рядки, а масиви **Targ_Train_N**, **Targ_Test_N**, **Targ_Valid_N** – мають 1 рядок. Кількість стовпців масивів відповідає розмірам відповідних послідовностей.

Завантажимо послідовності входів і цілей в робочу область GUI-інтерфейсу **NNTool**, використовуючи вікно **Import to Network/Data Manager** (рис. 2.6), що викликається при натисканні на кнопку **Import у вікні Network/Data Manager**.

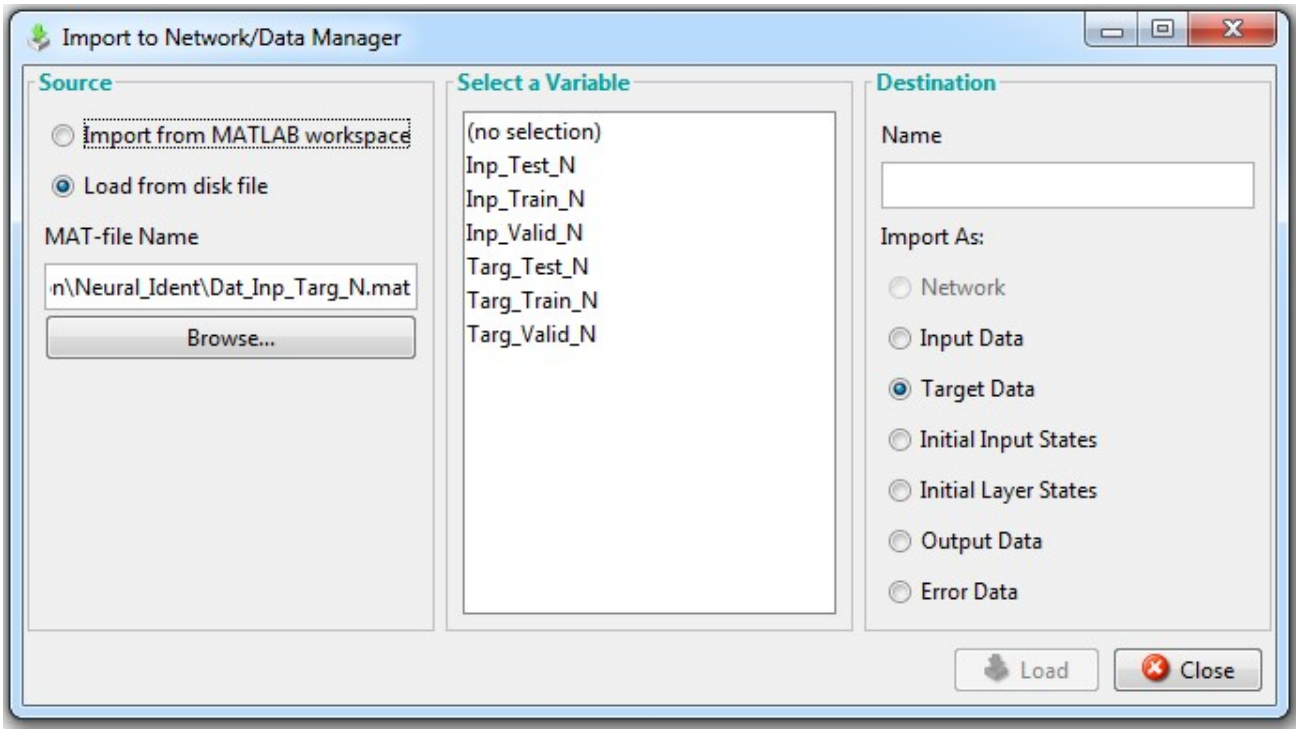


Рис.2.6. Вікно для завантаження даних в робочу область GUI-інтерфейсу **NNTool**

Створимо нейронну мережу. Для цього у вікні **Create Network or Data** , на закладці **Network** (див. рис.1.3 практичного заняття 1) виберемо нейронну мережу типу **Feed-forward backprop** з прямою передачею сигналу і із зворотним розповсюдженням помилки. Присвоюємо мережі ім'я **Neural_Model**.

З випадаючого меню для **Input data** і **Target data** виберемо **Inp_Train_N** і **Targ_Train_N**.

При ідентифікації найбільш важливим питанням є вибір кількості нейронів прихованого шару (тобто шару 1). При малій кількості нейронів мережа не може виконувати поставлене завдання, а при великій спостерігається явище перенавчання і зростає об'єм обчислень. Встановимо в полі **Number of neurons** кількість нейронів 10 (як показали дослідження, для даного завдання ідентифікації оптимальні значення нейронів прихованого шару знаходиться в межах $8 \div 12$ при цьому помилка навчання, а також помилки на контрольній і тестовій множині не перевищують $2 \cdot 10^{-2}$.

У другому (вихідному) шарі кількість нейронів 1.

Використовуємо функції активації (**Transfer function):** гіперболічного тангенса (**tansig**) – в першому шарі, лінійну (**purelin**) – в другому шарі.

Як навчальну функцію виберемо **trainlm**, що відповідає алгоритму Левенберга-Марквардта.

При натисненні на кнопку **Create** відбувається створення мережі з вказаними параметрами. Назва мережі з'являється в списку нейронних мереж вікна **Neural Network\Data Manager**.

Схема мережі показана на рис.1.7 практичного заняття 1. Схема мережі з'являється при натисненні на кнопку **View** у вікні **Create Network or Data** або при виборі закладки **View** в діалоговій панелі **Network: Neural Model**, яке детально описано вище.

Виконаємо ініціалізацію мережі, для чого в діалоговій панелі **Network: Neural Model** виберемо закладку **Reinitialize Weigts** (рис.2.7). Відкриється діалогова панель, показана на рис.2.7.

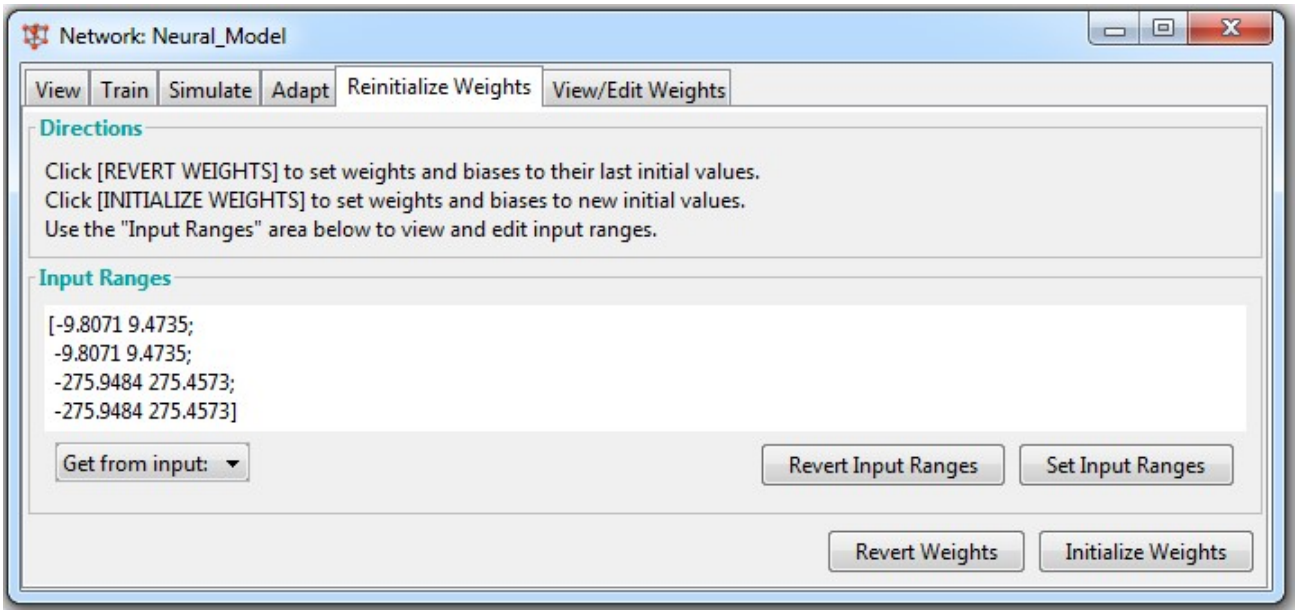


Рис.2.7. Закладка **Reinitialize Weigts** діалогової панелі панелі **Network: Neural Model**

Діапазони значень початкових даних виберемо по входах **Inp_Train_N** з випадаючого меню **Get from input**. Для введення встановлених діапазонів і ініціалізації вагів треба скористатися кнопками **Set Input Ranges** (Встановити діапазони для вхідних даних) і **Initialize Weights** (Ініціалізувати ваги). Якщо потрібно повернутися до колишніх діапазонів, то слід вибрати кнопки **Revert Input Ranges** (Повернути діапазонидіапазони для вхідних даних) і **Revert Weights** (Повернути ваги).

Потім виконується навчання мережі, для чого в діалоговій панелі **Network: Neural Model** вибирається закладка **Train**. Як зазначалось в методичних вказівках до практичного заняття 1, закладка в свою чергумає дві закладки: **Training Info** (Інформація про навчальні послідовності) (рис.1.8) і **Training Parameters** (Параметри

навчання) (рис.1.9), детально описані вище. Застосовуючи ці закладки, встановлюємо імена послідовностей входу і мети, а також параметри процедури навчання.

Тепер можна приступити до навчання мережі (кнопка **Train Network** на закладці **Train Parameters**). Процес та результати навчання наочно демонструються за допомогою діалогового вікна **Neural Network Training (nntraintool)**, показаного на рис..2.8. Наведене вікно - це інтерфейс для навчання нейронних мереж у Neural Network Toolbox в MATLAB. Детальний опис вікна дається в навчально-методичному посібнику до дисципліни.

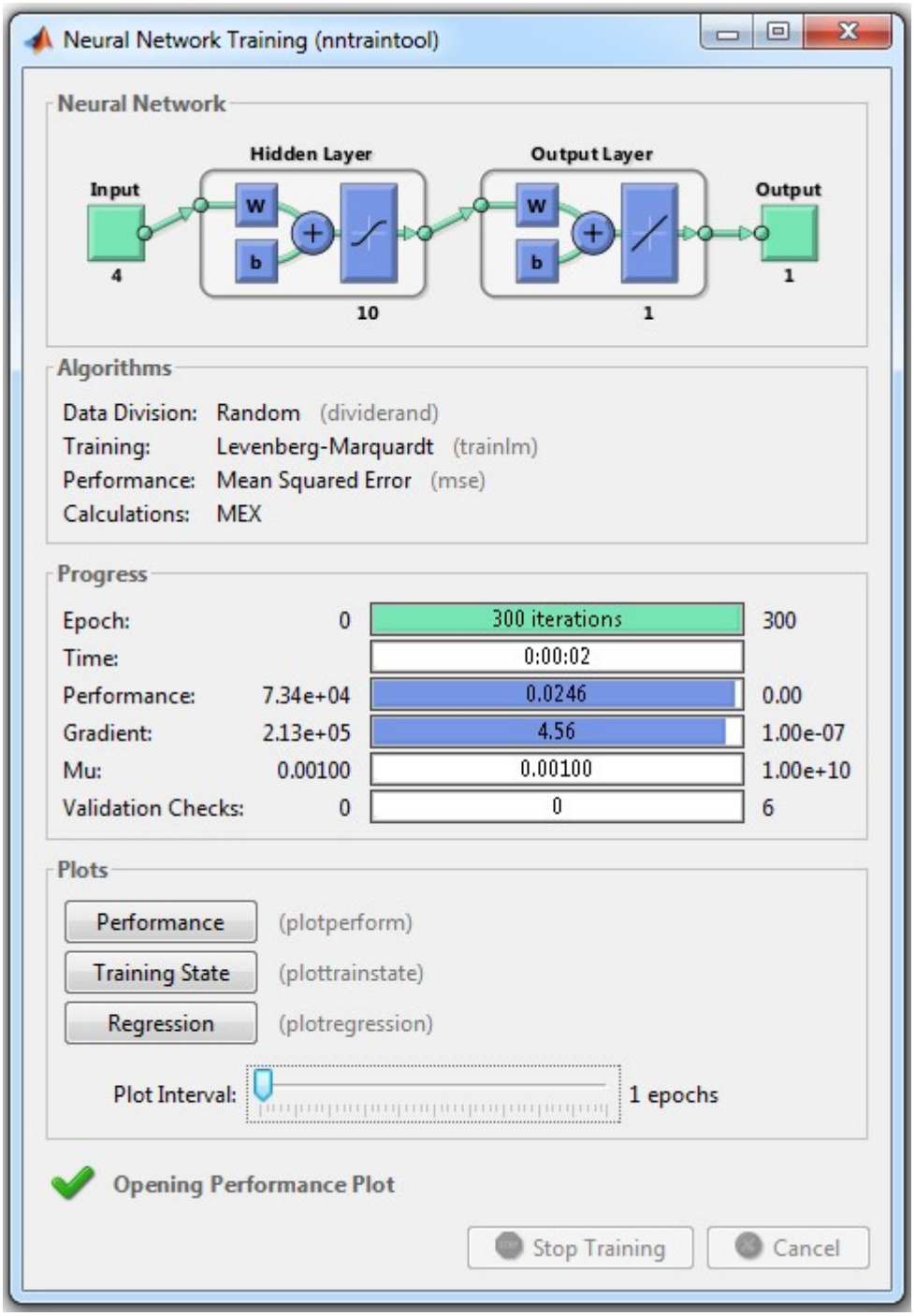


Рис. 2.8. Вікно **Neural Network Training**, що відображає хід навчання та результати навчання нейронної мережі

При натисканні на кнопку **Performance** відкривається вікно **Neural Network Training Performance**, показане на рис. 2.9. У вікні відображаються графік **Mean Squared Error (mse)**, які показують, як змінюється середньоквадратична помилка на навчальних, валідаційних (перевірочних) та тестових наборах даних під час кожної ітерації навчання. Графіки MSE показують, як змінюється точність мережі під час навчання і чи є ознаки перенавчання (overfitting), коли помилка на валідаційних та тестових даних починає зростати. Метою є мінімізувати MSE на усіх наборах даних для досягнення найкращої моделі.

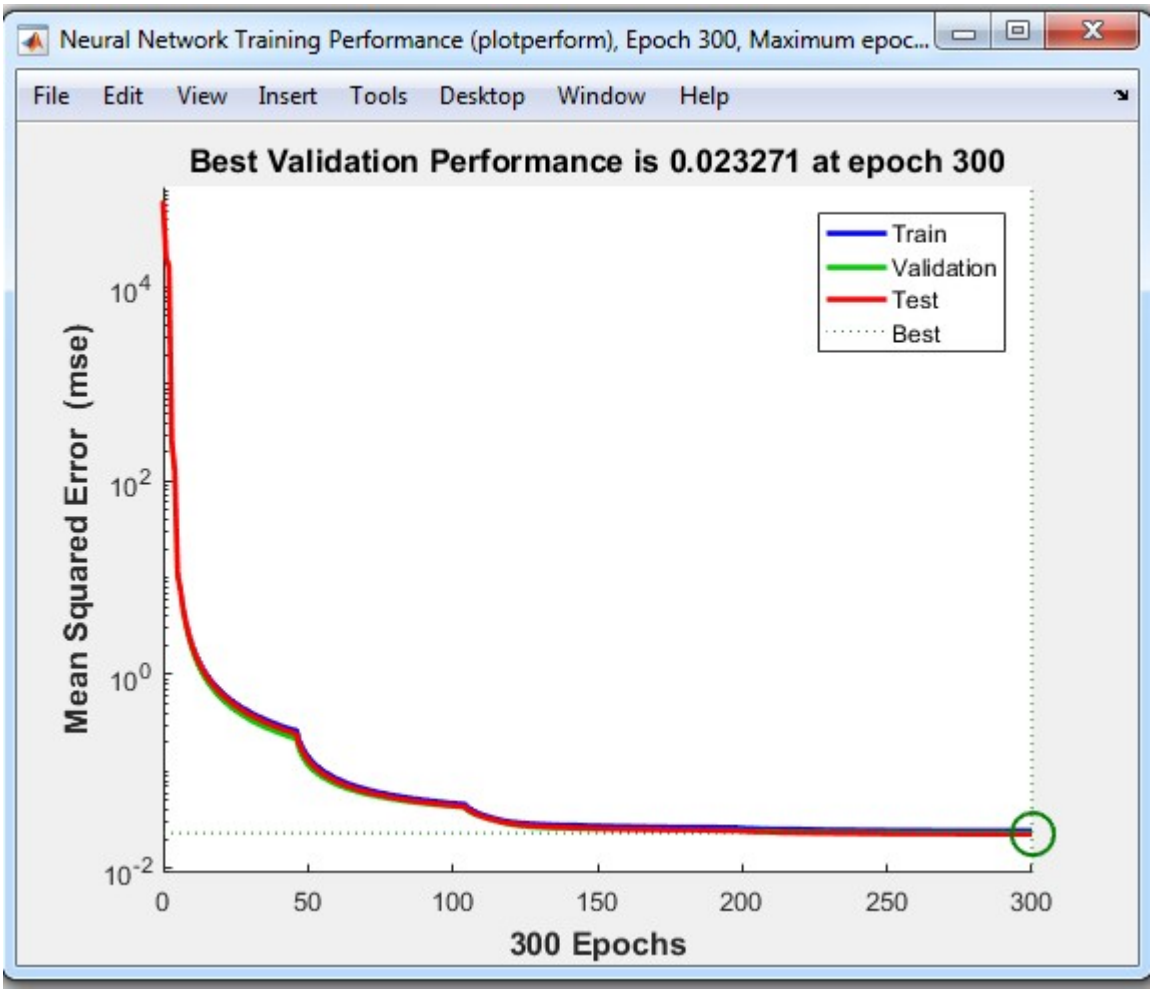


Рис.2.9. Вікно контролю процесу навчання нейронної мережі

При натисканні на кнопку **Training State** відкривається вікно **Neural Network Training State** (рис. 2.10**)** в якому відображається інформацію про стан навчання нейронної мережі на певний момент часу під час процесу навчання.

Графік Gradient показує зміну значення градієнта під час навчання нейронної мережі. Градієнт в контексті навчання нейронних мереж є вектором, який вказує напрямок та міру найшвидшого зростання функції втрати (або помилки) мережі в точці навчання. Градієнт використовується для оновлення вагових коефіцієнтів мережі під час оптимізації.

Графік mu показує зміну значення параметра mu під час навчання нейронної мережі. Параметр mu відноситься до швидкості навчання та регуляризації мережі, і його зміна вказує на те, як адаптується швидкість навчання під час тренування.

Зазвичай, під час навчання нейронної мережі використовуються методи оптимізації, такі як стохастичний градієнтний спуск (SGD) або його модифікації. Параметр mu в цьому контексті часто вказує на швидкість навчання (learning rate) та регуляризацію.

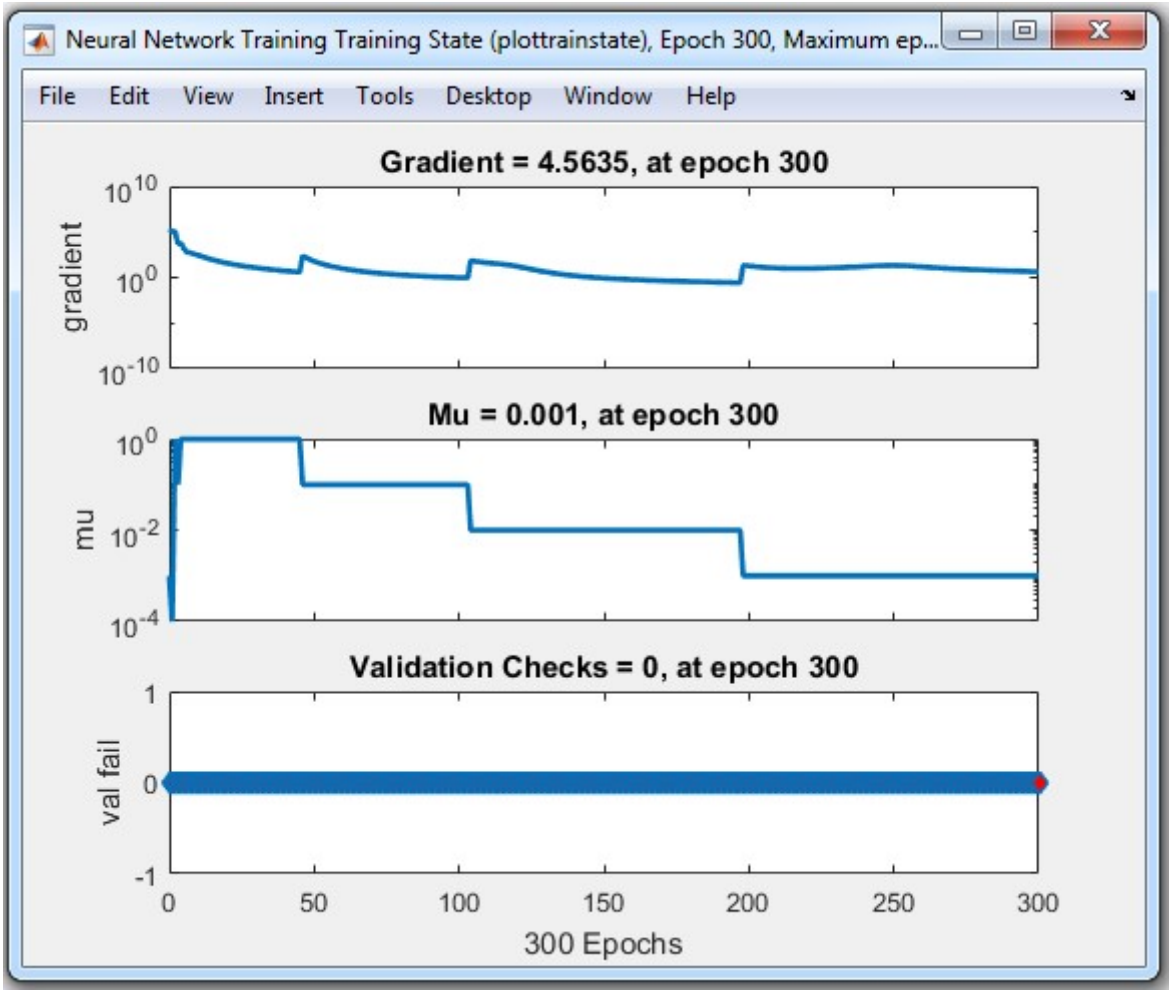


Рис.2.10. Вікно, що відображається інформацію
про стан навчання нейронної мережі

Графік Validation Failure (або **val fail**) **(помилка підтвердження**) показує, як змінюється величина, що відображає кількість невдач (помилок) під час перевірки (валідації) нейронної мережі під час навчання. Головною метою валідації є визначення, наскільки добре навчена мережа узгоджується з набором даних, які не використовувалися під час навчання, і чи існують ознаки перенавчання.

При натисканні на кнопку **Regression** відкривається вікно **Neural Network Training Regression** (рис. 2.11), у якому показані графіки та інформація, які допомагають в оцінці та аналізі продуктивності нейронної мережі під час навчання. Для наочної оцінки якості навчання приведені графіки регресії для Training, Validation та Test наборів даних.

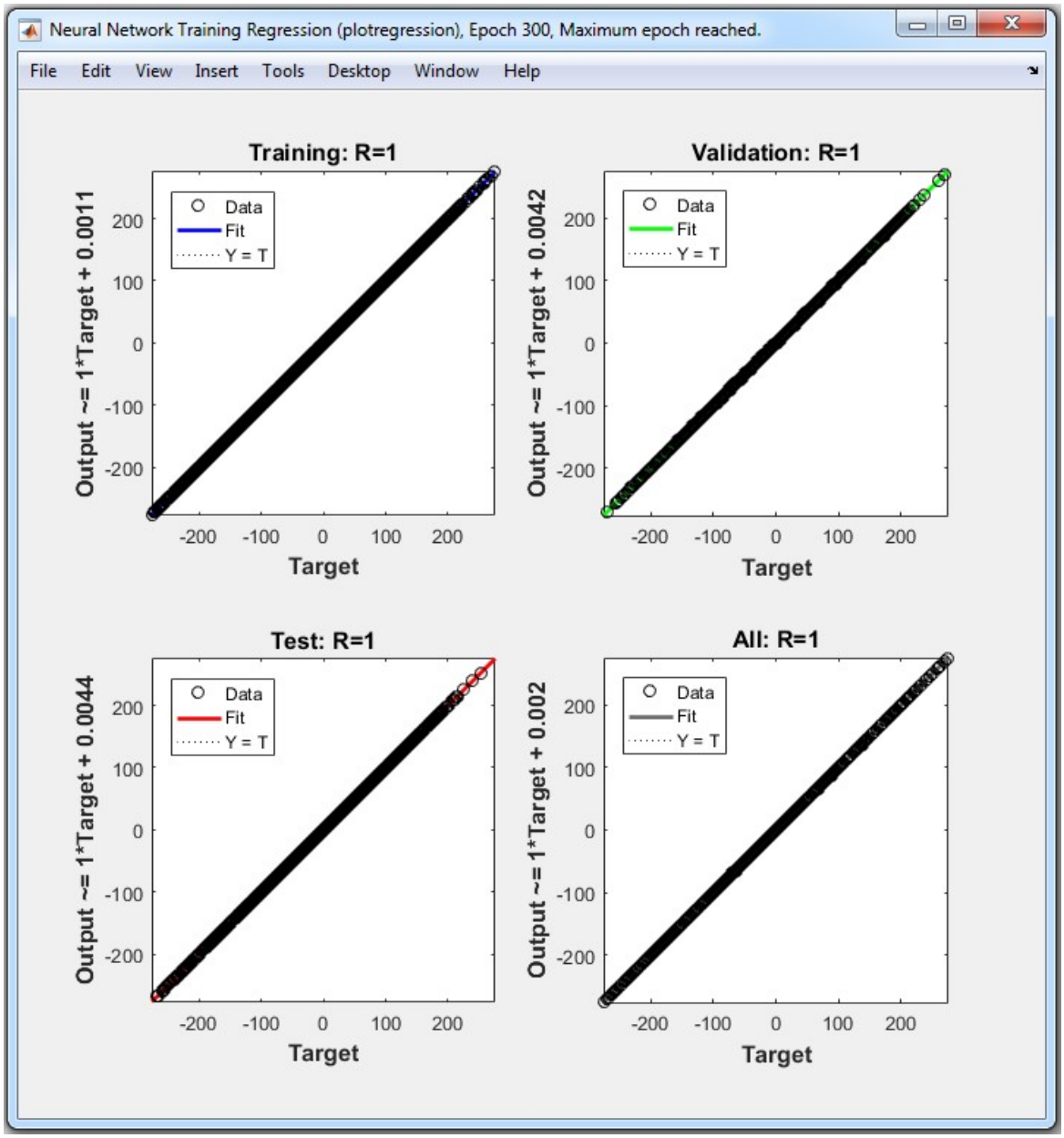


Рис. 2.11. Вікно **Neural Network Training Regression**

Training показують, наскільки точно модель відповідає навчальним даним, тобто даним, на яких вона була навчена.

Графіки **Validation** демонструють, наскільки точно модель працює на наборі даних для перевірки. Дані для перевірки використовуються для валідації результатів моделі та визначення, наскільки вона загальніє на нових даних.

Графіки **Test** включають тестові дані, які незалежно від використовувалися для оцінки точності моделі на даних, які модель раніше не бачила. Тестові дані надають об'єктивну оцінку роботи моделі на нових даних.

Графіки **All** показують регресійні результати для всіх трьох наборів даних: Training, Validation та Test. Графіки All об'єднують і відображають прогнози моделі та фактичні значення для всіх трьох наборів даних на одному графіку з метою порівняння. Ці графіки забезпечують користувача загальним зоровим порівнянням точності моделі на різних наборах даних і допомагають визначити, чи існують різниці в її поведінці на тренувальних, валідаційних та тестових даних.

Результат тренування мережі залежить від початкового значення вагів нейронної мережі w_{ij}. і кількості циклів (епох) навчання $N_{\text{ц}}$ (епох). Для досягнення глобального мінімуму процес навчання необхідно повторювати багато разів при різних початкових значеннях w_{ij} (встановлюється на закладці **Reinitialize Weigts** діалогової панелі **Network: Neural Mode** (див. рис.2.7) натисненням на кнопку **Initialize Weights**) і величині $N_{\text{ц}}$ (встановлюється на закладці **Train Parameters** панелі **Network: Neural Model** (див. рис.1.9 практичного заняття 1)). У даному завданні для кожного варіанту мережі (тобто кількості нейронів в першому шарі) вибиралося декілька десятків початкових точок розрахунку. Кількість циклів навчання, після закінчення яких помилка навчання переставала зменшуватися, складало $100 \div 130$, при цьому помилка навчання складала $\approx 2 \cdot 10^{-2}$

Відповідні ваги і зсуви можна побачити, а за потреби скоригувати, якщо на панелі **Network: Neural Mode** вибрати закладку **View/Edit Weights** (рис.2.12).

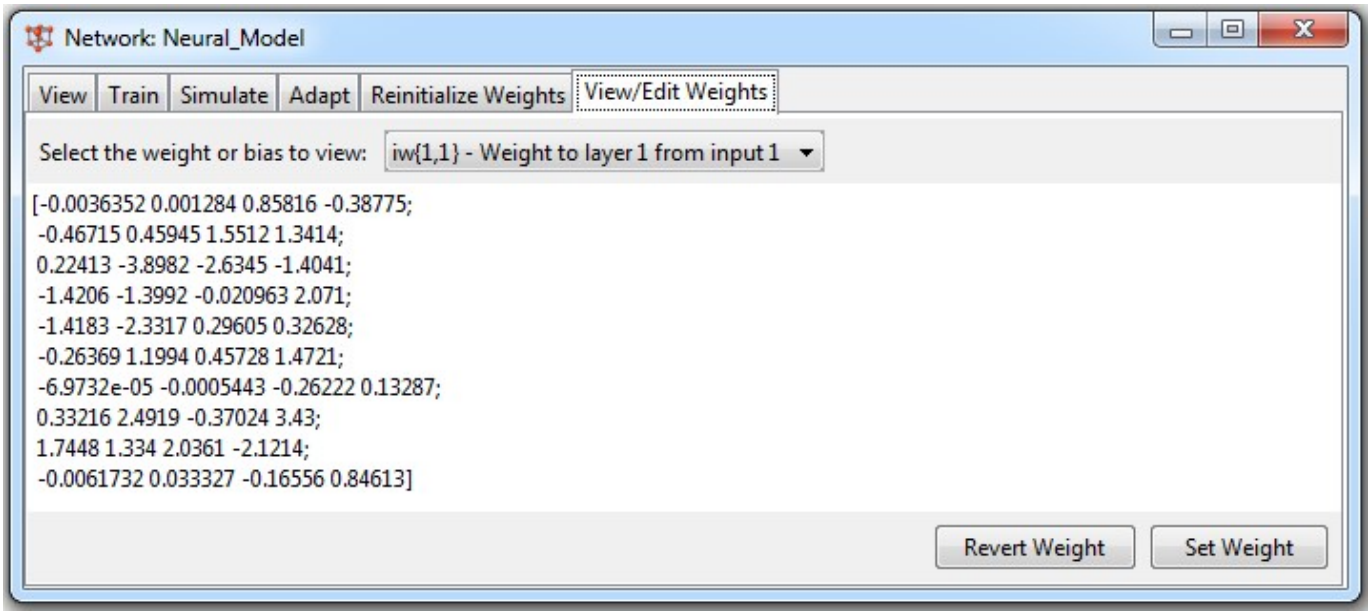


Рис.2.12. Вікно для перегляду вагів і зсувів мережі

Для зручності роботи можна експортувати створену нейронну мережу в робочу область системи MATLAB і отримати інформацію про ваги і зсуви безпосередньо в робочому вікні системи.

Результати навчання можна проглянути, використовуючи вікно **Neural Network/Data Manager** (рис.2.13). Активізуючи імена послідовностей виходу або помилок **Neural_Model_outputs** або **Neural_Model_errors** подвійним клацанням

миші, можна проглянути результати у відповідних вікнах, що відкриваються (рис.2.14).

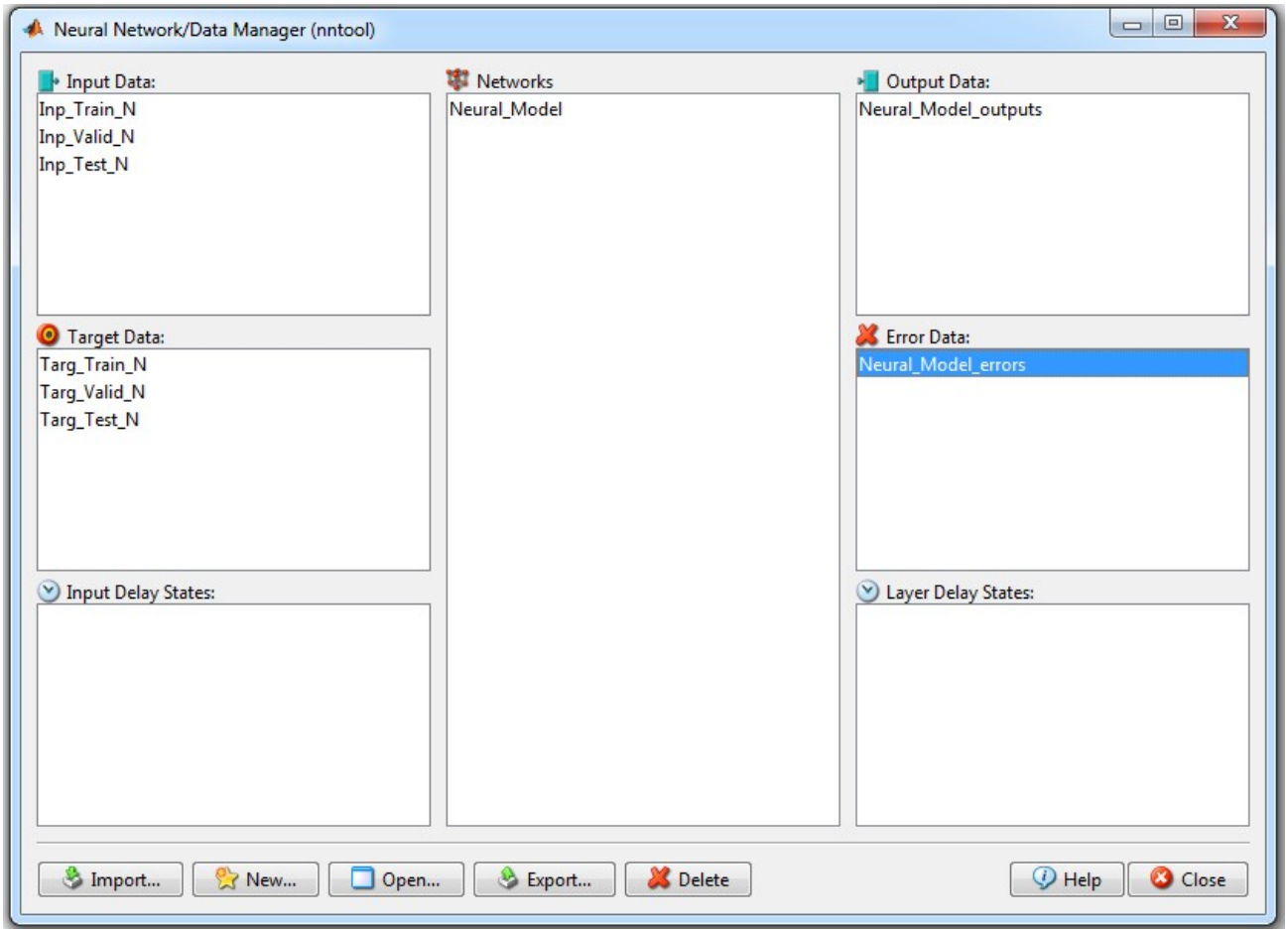


Рис.4.13. Вікно **Network/Data Manager** з активізованим ім'ям помилки **Neural_Model_errors**

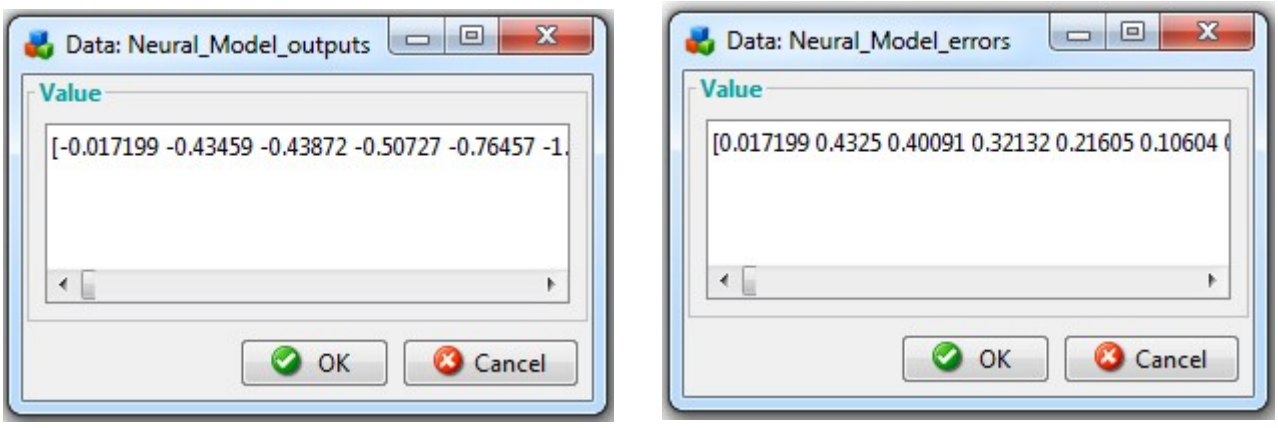


Рис.2.14. Вікна для перегляду значень сигналу на виході і помилки мережі

Запишемо дані з робочої області GUI-інтерфейсу **NNTool** у вигляді файлу на диску. Для цього у вікні рис.2.13 активізуємо кнопку **Export**. У вікні **Export from**

Network/Data Manager (рис.2.15), що відкрилося, виділимо всі змінні (кнопка **Select All**) і збережемо (кнопка **Save**) у файлі з ім'ям **NN_Model_and_Dat.mat**.

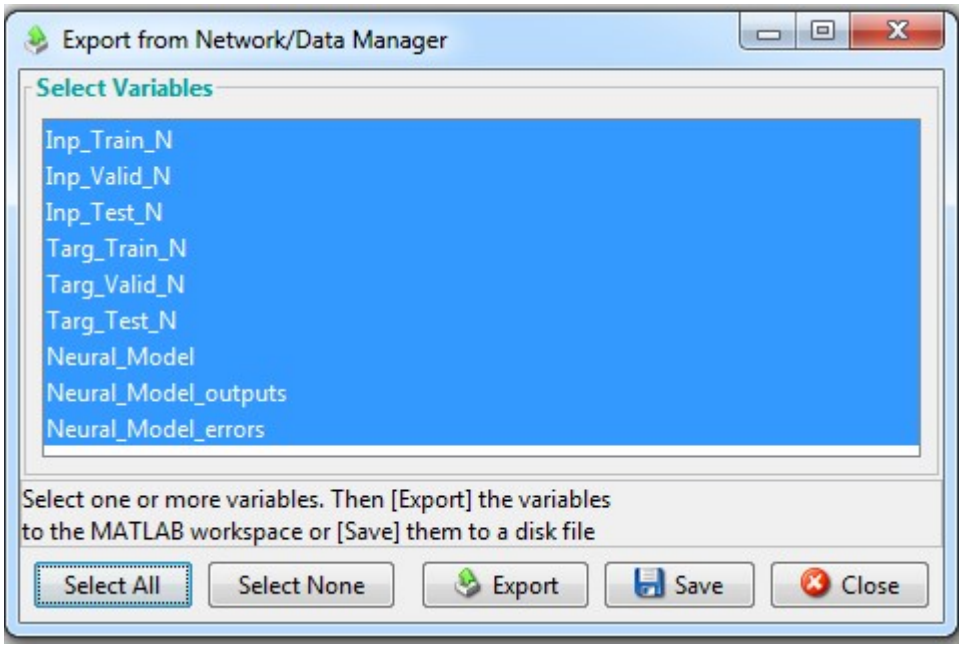


Рис.2.15. Вікно **Export or Save from Network/Data Manager**

Для наочності побудуємо графіки вхідного **System Input** і вихідного **System Output** сигналів двомасової системи, вихідного сигналу побудованої нейромеоежевої моделі **NN Output** і графік миттєвого значення помилки навчання мережі **Error** для перших 2001 значень вказаних сигналів, що відповідає часу $100c$ (рис.2.16). Лістинг файлу в редакторі/відладчику **m**-файлів показаний на рис.2.16. Відповідні підписи у вікні рис.2.17 виконані з використанням опцій меню **Edit\Figure Properties** і **Insert\X Label**, **Insert\Title**.

З аналізу рисунка виходить, що миттєве значення погрішності навчання не перевищує $1,5c^{-1}$.

Тепер можна побудувати модель нейронної мережі в системі **Simulink** за допомогою оператора **gensim(Neural_Model)** (рис.2.18). Ця схема, на відміну від ілюстративних схем, є повною мірою функціональною схемою і може бути застосована для моделювання нейронної мережі.

Для перегляду структури мережі слід активізувати елементи мережі (рис.2.18). Кожен подальший елемент з'являється в окремому вікні при активізації попереднього подвійним клацанням миші. З даних елементів в системі **Simulink** може бути побудована схема мережі, показана на рис.2.19

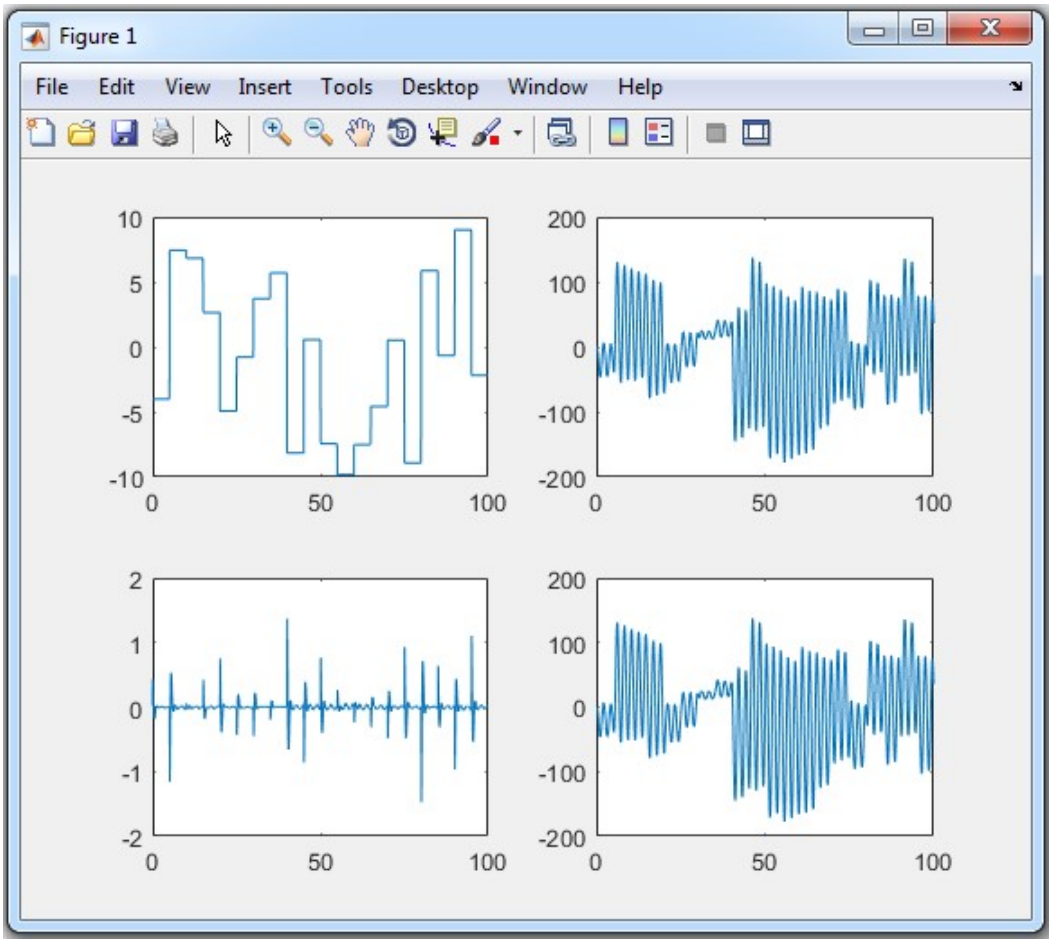


Рис.2.16. Результати тренування мережі

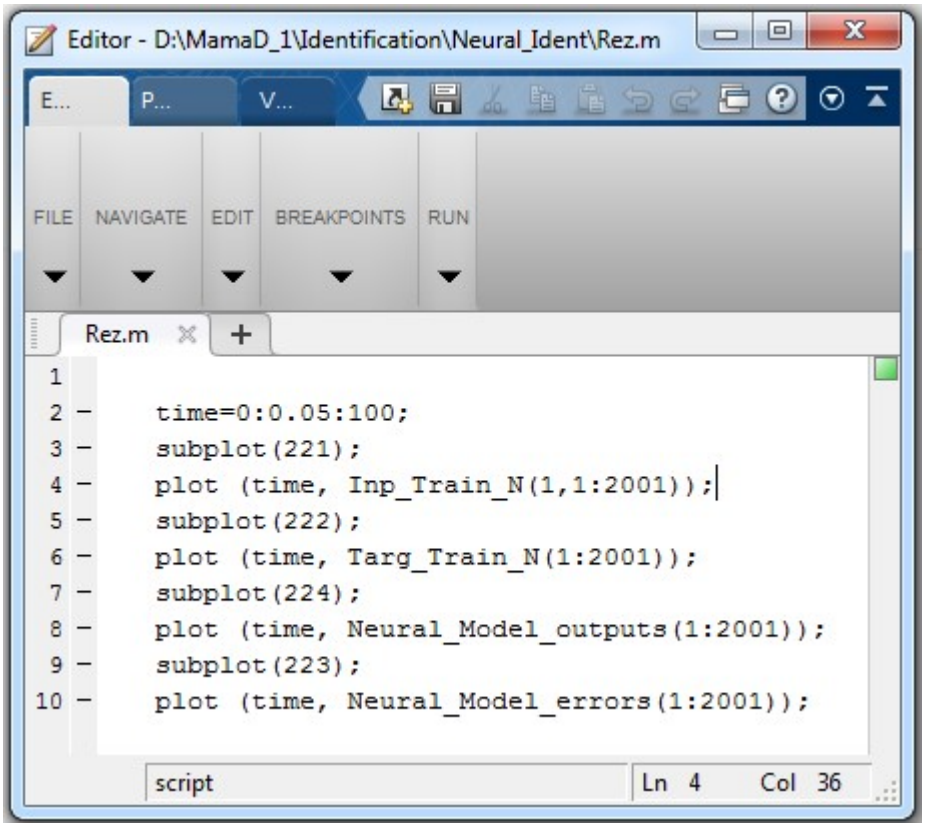


Рис.2.17. Лістинг файлу побудови графіків рис.2.16

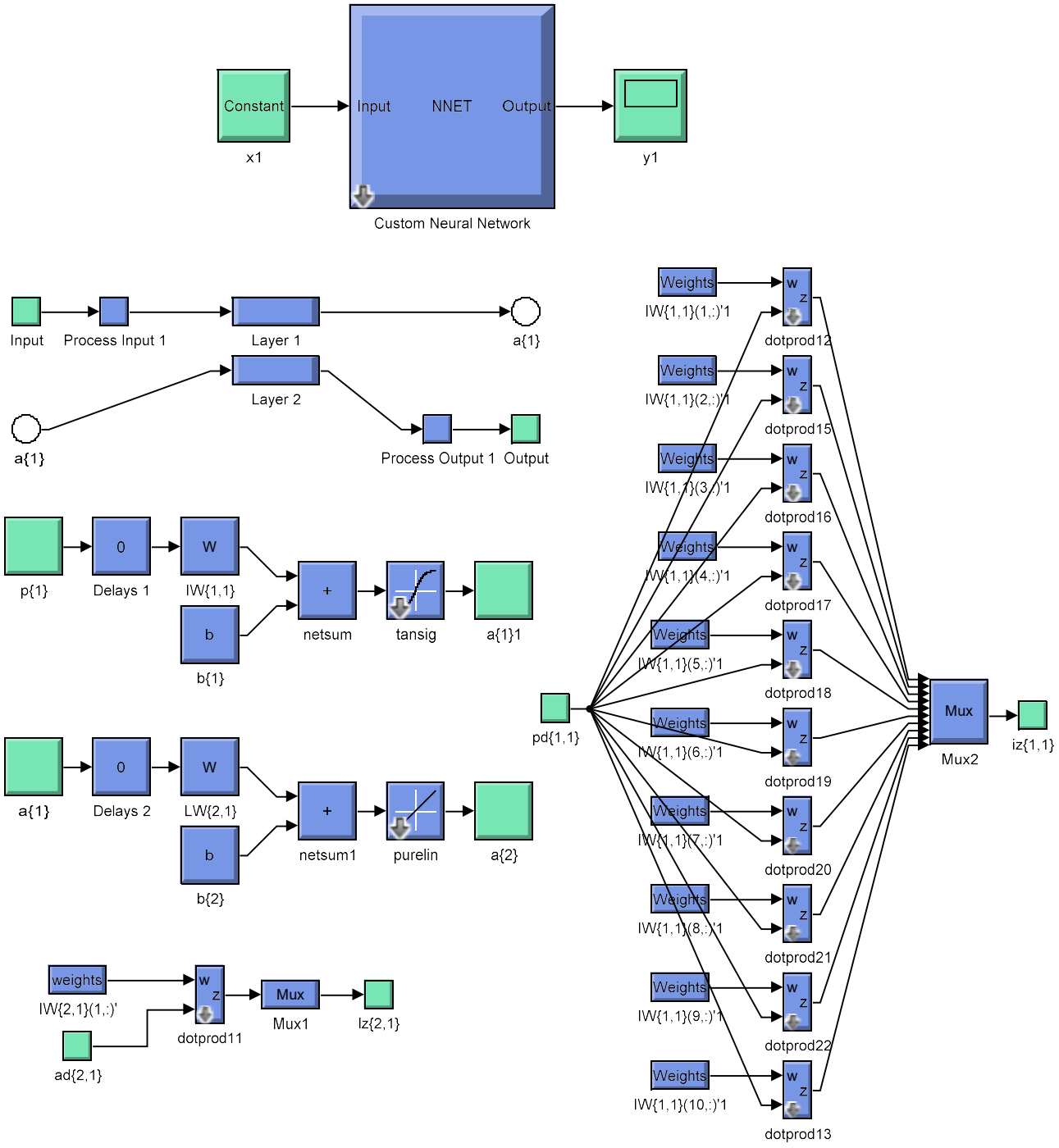


Рис.2.18. Згорнута модель нейронної мережі з прямою передачею сигналу і моделі елементів нейронної мережі, реалізовані в Simulink

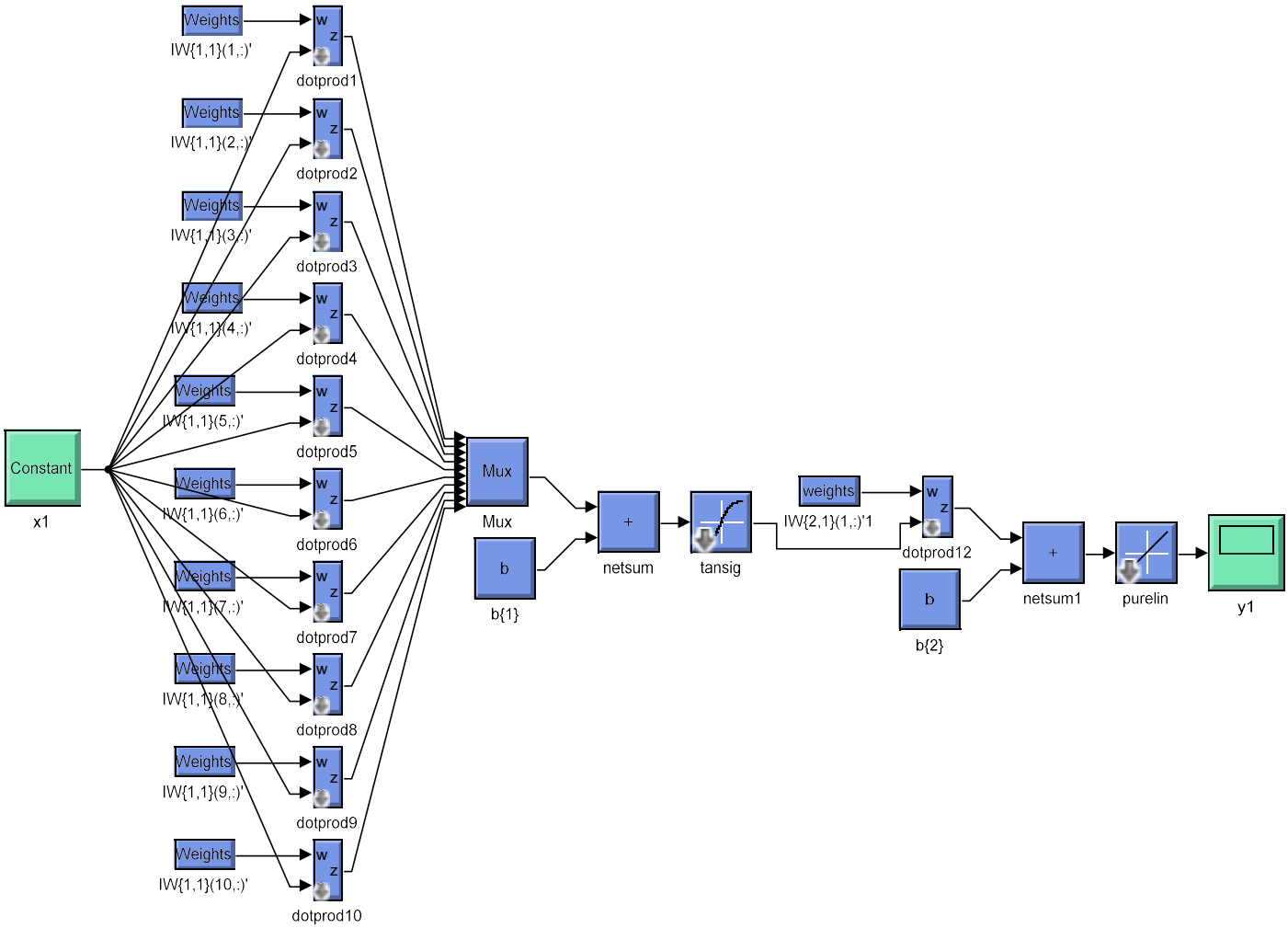


Рис.2.19. Розгорнута модель нейронної мережі з прямою передачею сигналу

Дана мережа є статичною (статична мережа характеризується тим, що в її складі немає елементів запізнювання і зворотних зв'язків). Мережа використовує 1 вектор входу з 4 елементами, має 2 шари з 10 нейронами в першому (прихованому) шарі і 1 нейроном в другому (вихідному) шарі. Використовувані функції активації: гіперболічного тангенса (**tansig)** – в першому шарі, лінійна (**purelin**) – в другому шарі.

Виконаємо перевірку адекватності побудованої нейромережевої моделі. Для цього створимо у вікні Simulink схему, показану на рис.2.20. Схема складається з моделі двомасової електромеханічної системи управління (див. рис.2.2, і блоку **Neural Network**.

Для набуття затриманих значень вхідної і вихідної змінних використані блоки **Unit Delay** (можна використовувати блоки **Memory**).

Перед початком моделювання слід завантажити в робочу область MATLAB початкові дані двомасової електромеханічної системи управління механізмом під-йому мостового крана (файл **Dat_Kran**). Результати моделювання представлені на рис.2.21 і рис.2. 22.

На рис.2.21 приведені графіки вихідної координати системи – швидкості механізму $\omega_{\text{м}}(t)$. Графік, зображений червоною лінією відповідає заданому значенню швидкості $\omega_{\text{м}_\text{з}}(t)$, синьою лінією – швидкості на виході моделі двомасової системи $\omega_{\text{м}_\text{дс}}(t)$ і зеленою пунктирною лінією – вихідній координаті побудованої нейромережевої моделі $\omega_{\text{м}_\text{нм}}(t)$. Як бачимо, графіки $\omega_{\text{м}_\text{дс}}(t)$ і $\omega_{\text{м}_\text{нм}}(t)$ практично співпадають.

На рис.2.22 зображений графік різниці вказаних швидкостей: $\varepsilon(t) = \omega_{\text{м}_\text{дс}}(t) - \omega_{\text{м}_\text{нм}}(t)$, тобто графік помилки ідентифікації. З аналізу графіків виходить, що при зміні швидкості механізму в межах від $+140c^{-1}$ до $-140c^{-1}$ значення $\varepsilon(t)$ знаходиться в межах від $+2{,}5c^{-1}$ до $-2{,}5c^{-1}$ тобто помилка ідентифікації не перевищує 2%.

Таким чином, перевірка побудованої нейромережевої моделі двомасової системи показує достатньо високий ступінь її адекватності реальним даним, що дозволяє зробити висновок про можливість її практичного використання для вирішення завдань ідентифікації. Розглянутий підхід є перспективним напрямом для побудови і використання відповідних нейромережевих моделей динамічних об'єктів і систем за відсутності апріорних даних про їх структуру і параметри.

2.3 Завдання для самостійного виконання

Варіанти завдань до побудови нейромережевих моделей двомасових і трьохмасових електромеханічних систем з використанням GUI-інтерфейсу NNTool наведено в додатку А.

2.4 Контрольні питання по темі заняття

1. Наведіть методику синтезу нейромережевої моделі двомасової електромеханічної системи.
2. Як вибирається кількість нейронів прихованого шару?
3. Як вибирається кількість циклів навчання?
4. За допомогою якого оператора можна побудувати модель нейронної мережі в системі Simulink?
5. Які блоки використовуються для набуття затриманих значень вхідної і вихідної змінних?

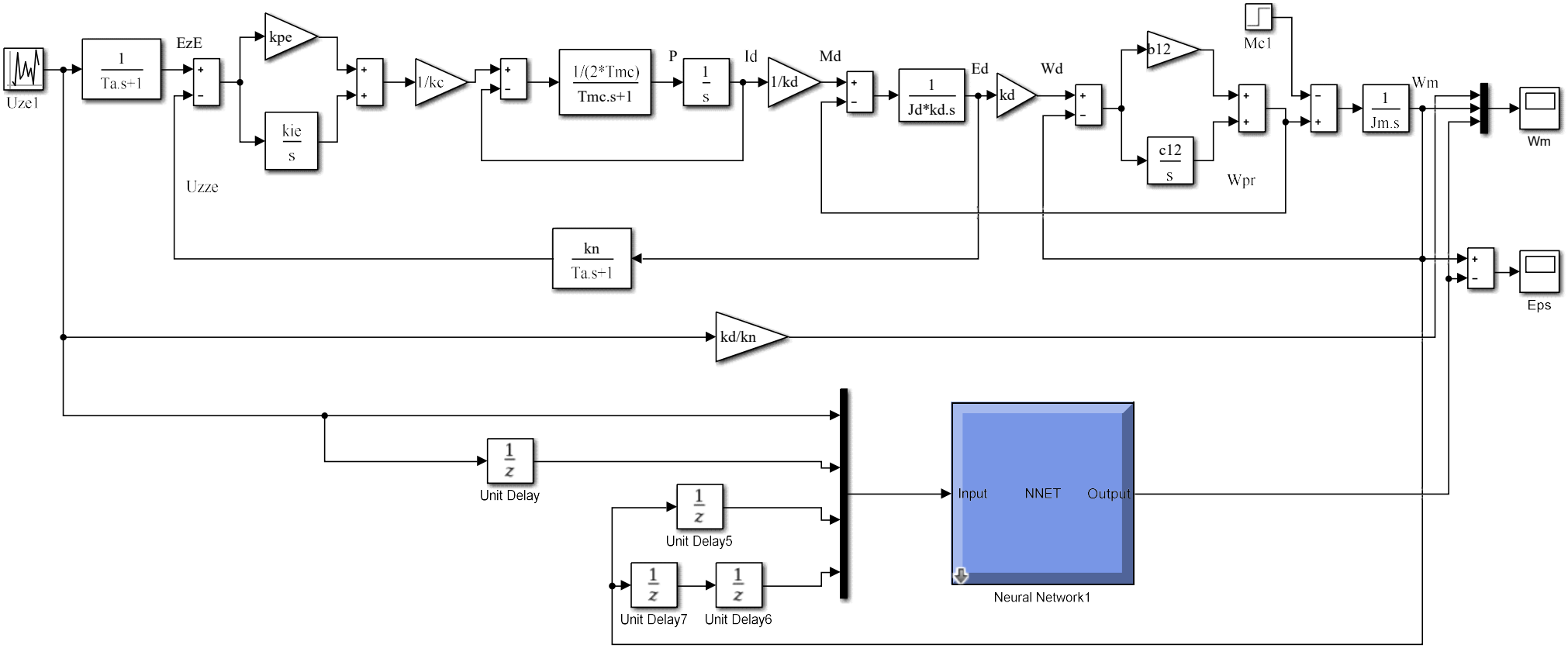


Рис.2.20. Схема для перевірки адекватності побудованої нейромережевої моделі двомасової системи

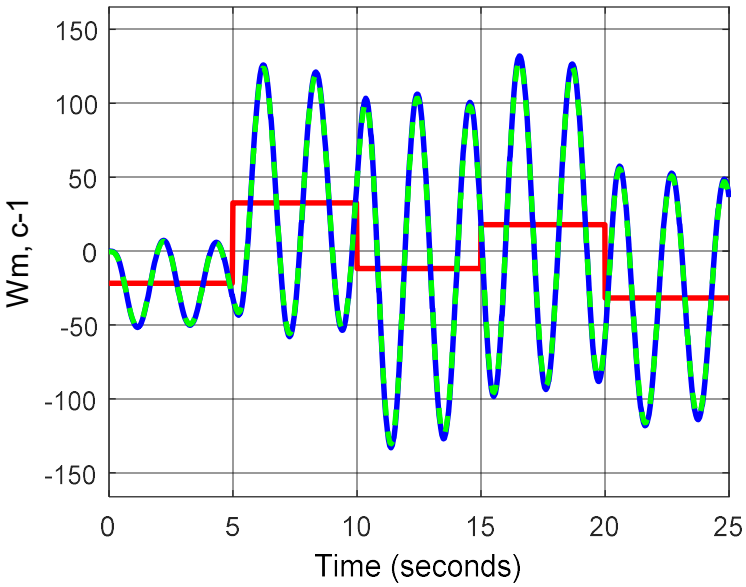


Рис.2.21. Графіки вихідної координати двомасової електромеханічної системи

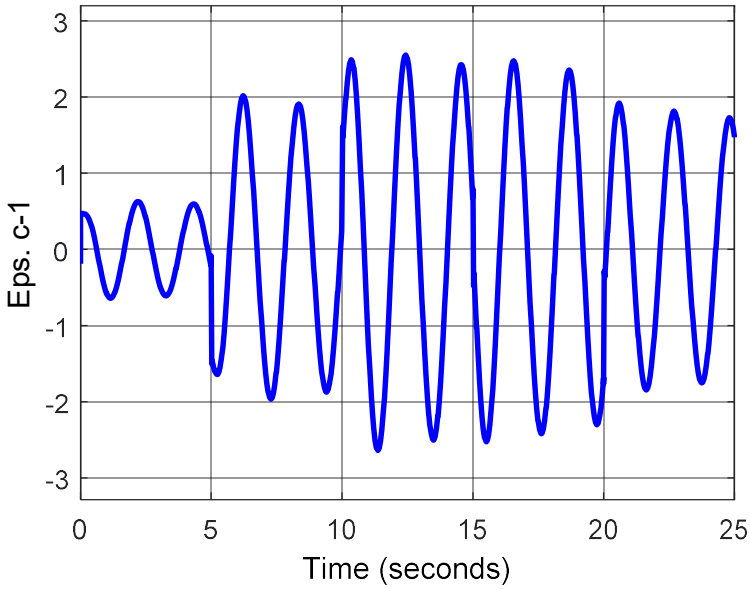


Рис.2.22. Графік помилки ідентифікації

Практичне заняття 3

СИНТЕЗ НЕЙРОРЕГУЛЯТОРА З ПРОГНОЗОМ NN PREDICTIVE CONTROLLER З ВИКОРИСТАННЯМ СИСТЕМИ MATLAB

3.1 Мета заняття

Ознайомлення з порядком синтезу нейрорегулятора з прогнозом **NN Predictive Controller** у середовищі MATLAB. Вивчення методики створення, налаштовування та навчання нейронної мережі для прогнозування динаміки об’єкта керування, а також використання отриманої моделі для реалізації предиктивного управління. Заняття спрямоване на набуття практичних навичок розробки інтелектуальних систем управління, заснованих на нейронних мережах, для застосування в автоматизованих системах керування складними об’єктами.

3.2 Регулятори, реалізовані в пакеті прикладних програм Neural Network Toolbox системи MATLAB

Синтез нейромережевих систем управління доцільно проводити з допомогою пакета прикладних програм Neural Network Toolbox системи MATLAB. .

В пакеті прикладних програм Neural Network Toolbox системи MATLAB реалізовано 3 нейрорегулятора:

- регулятора з прогнозом (**NN Predictive Controller**);
- регулятора на основі моделі авторегресії з ковзаючим середнім (**NARMA-L2 Controller**);
- регулятора на основі еталонної моделі (**Model Reference Controller**).

Застосування нейронних мереж для вирішення завдань управління дозволяє виділити 2 етапи проектування:

- етап ідентифікації керованого об'єкту;
- етап синтезу закону управління.

На етапі ідентифікації розробляється модель керованого об'єкту у вигляді нейронної мережі, яка на етапі синтезу використовується для синтезу регулятора. Для кожного з трьох регуляторів використовується одна і та ж процедура ідентифікації, проте етапи синтезу істотно розрізняються.

При управлінні з прогнозом модель керованого об'єкту використовується для того, щоб передбачити його майбутню поведінку, а алгоритм оптимізації застосовується для розрахунку такого управління, яке мінімізує різницю між бажаними і дійсними змінами виходу моделі.

При управлінні на основі моделі авторегресії з ковзаючим середнім регулятор є достатньо простою реконструкцією моделі керованого об'єкту.

При управлінні на основі еталонної моделі регулятор – це нейронна мережа,

яка навчена управляти об'єктом так, щоб він відстежував поведінку еталонної моделі. При цьому модель керованого об'єкту активно використовується при настройці параметрів самого регулятора.

Динамічні моделі нейромережевих регуляторів розміщені в спеціальному розділі **Control Systems** набору блоків **Neural Network Blocksets** (рис.3.1).

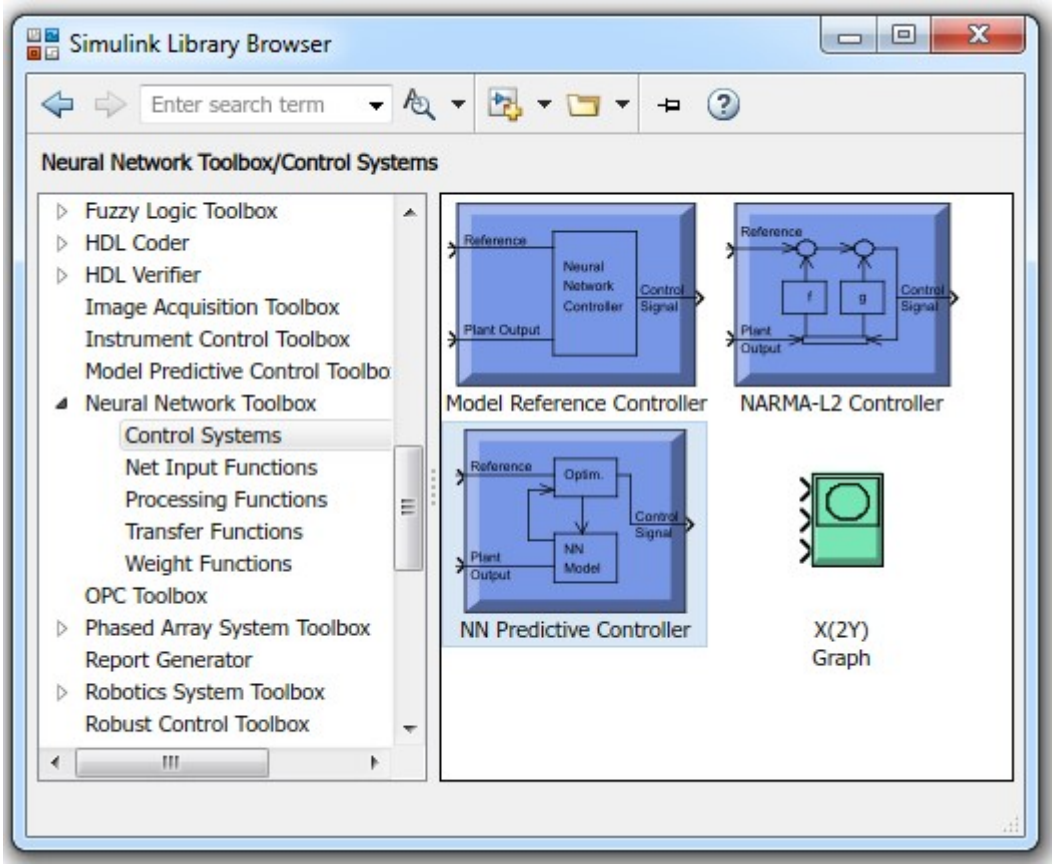


Рис.3.1. Блоки **Neural Network Blocksets** розділу **Control Systems** системи SIMULINK

Регулятор з прогнозом. Цей регулятор використовує модель керованого об’єкту у вигляді нейронної мережі для того, щоб передбачити майбутню реакцію об’єкту на випадкові сигнали управління. Алгоритм оптимізації обчислює управляючі сигнали, що мінімізують різницю між бажаною і дійсною зміною сигналу на виході моделі і таким чином оптимізують керований процес. Побудова моделі об’єкту управління виконується автономно з використанням нейронної мережі, яка навчається в груповому режимі з використанням одного з алгоритмів навчання. Регулятор, що реалізує такий алгоритм, потребує значного об’єму обчислень, оскільки для розрахунку оптимального закону управління оптимізація виконується на кожному такті управління.

Регулятор NARMA-L2. Цей регулятор вимагає найменшого об'єму обчислень. Даний регулятор – це просто деяка реконструкція нейромережевої моделі керованого об'єкту, отриманої на етапі автономної ідентифікації. Обчислення в реальному часі пов’язані тільки з реалізацією нейронної мережі.

Регулятор на основі еталонної моделі. Необхідний об'єм обчислень для цього регулятора порівнянний з попереднім. Проте архітектура регулятора з еталонною моделлю вимагає навчання нейронної мережі об'єкту управління і нейронної мережі регулятора. При цьому навчання нейронної мережі регулятора виявляється достатньо складним, оскільки навчання засноване на динамічному варіанті методу зворотного розповсюдження помилки. Достоїнством регуляторів на основі еталонної моделі є те, що вони застосовні до різних класів об'єктів управління.

3.3 Принцип побудови регулятора з прогнозом NN Predictive Controller

Регулятор з прогнозом NN Predictive Controller, реалізований в **ППП Neural Network Toolbox**, використовує модель нелінійного керованого об'єкту у вигляді нейронної мережі для того, щоб передбачати його майбутню поведінку. Крім того, регулятор обчислює сигнал управління, який оптимізує поведінку об'єкту на заданому інтервалі часу.

3.3.1 Ідентифікація об'єкту управління

Схема підсистеми ідентифікації показана на рис.3.2. Вона включає модель об'єкту управління у вигляді нейронної мережі, яка повинна бути навчена в автономному режимі так, щоб мінімізувати помилку між реакціями об'єкту і моделі $e = y - y_n$ на послідовність пробних сигналів u.

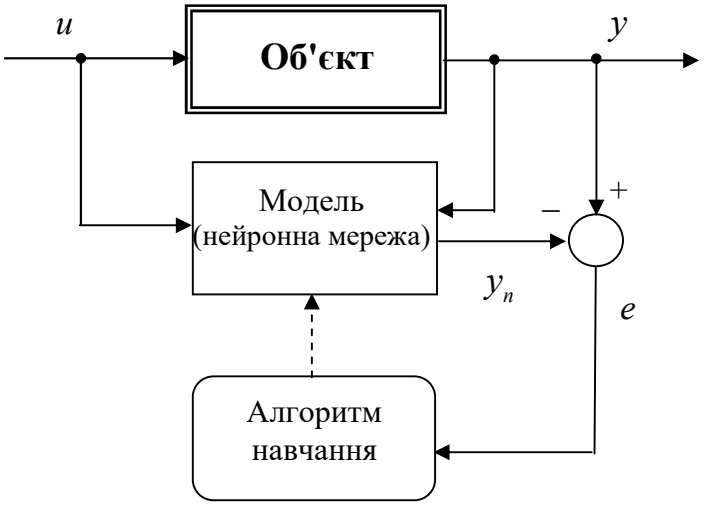


Рис.3.2. Схема підсистеми ідентифікації

Нейронна мережа регулятора представлена на рис.3.3. Вона має 2 шари нейронів і використовує лінії затримки (ЛЗ), щоб запам'ятати попередні значення входів і виходів об'єкту з метою передбачити майбутнє значення виходу.

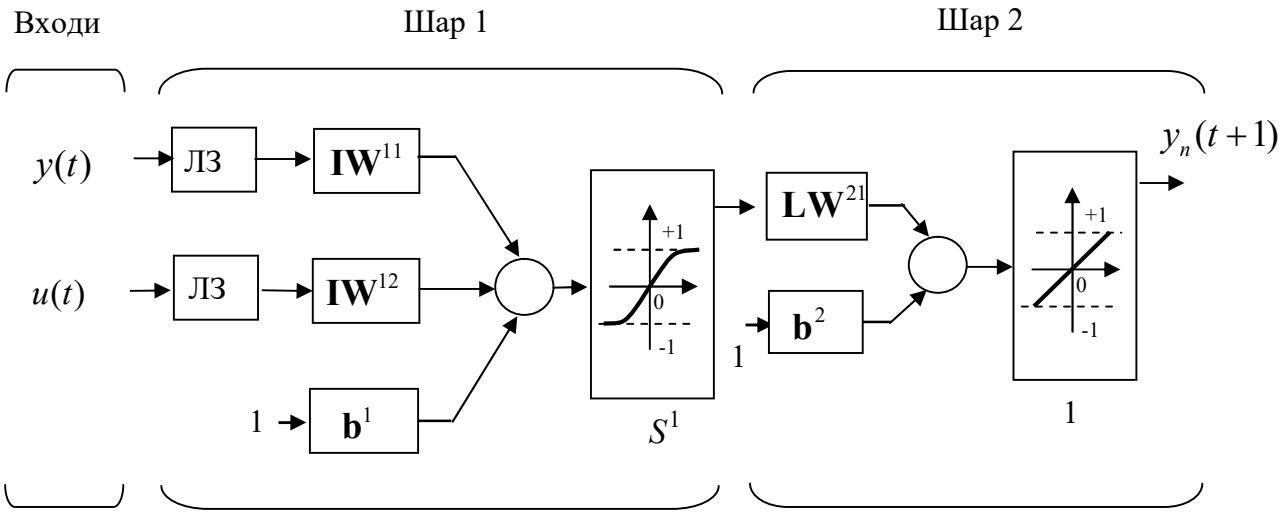


Рис.3.3. Нейронна мережа регулятора керованого об'єкту

Настройка параметрів цієї мережі виконується автономно методом групового навчання, використовуючи дані, отримані при випробуваннях об'єкту. Для навчання мережі може бути використаний будь-якому з навчальних алгоритмів для нейронних мереж. Найбільш ефективним є алгоритм Левенберга-Марквардта. У ППП Neural Network Toolbox даний алгоритм реалізований у вигляді М-функції **trainlm**.

3.3.2 Схема управління з прогнозом

Управління з прогнозом використовує принцип горизонту, що віддаляється, коли нейромережева модель керованого об'єкту передбачає реакцію об'єкту на певному інтервалі часу в майбутньому. Прогноз використовується програмою чисельної оптимізації для того, щоб обчислити управляючий сигнал, який мінімізує критерій якості

$$J = \sum_{j=N_1}^{N_2} \left[y_r(t+j) - y_n(t+j) \right]^2 + \sum_{j=1}^{N_u} \left[u'(t+j-1) - u'(t+j-2) \right]^2, \quad (3.1)$$

де константи N_1, N_2 і N_u задають межі, усередині яких обчислюються помилки стеження і потужність управляючого сигналу. Змінна u' описує пробний управляючий сигнал; y_r – очікувана, а y_n – дійсна реакція моделі даної системи. Величина ρ визначає внесок, який вносить потужність управління в критерій якості. Структурна схема на рис.3.4 ілюструє процес управління з прогнозом. Регулятор складається з нейромережевої моделі керованого об'єкту і блоку оптимізації. Блок оптимізації визначає значення, які мінімізують критерій якості управління, а відповідний управляючий сигнал управляє процесом.

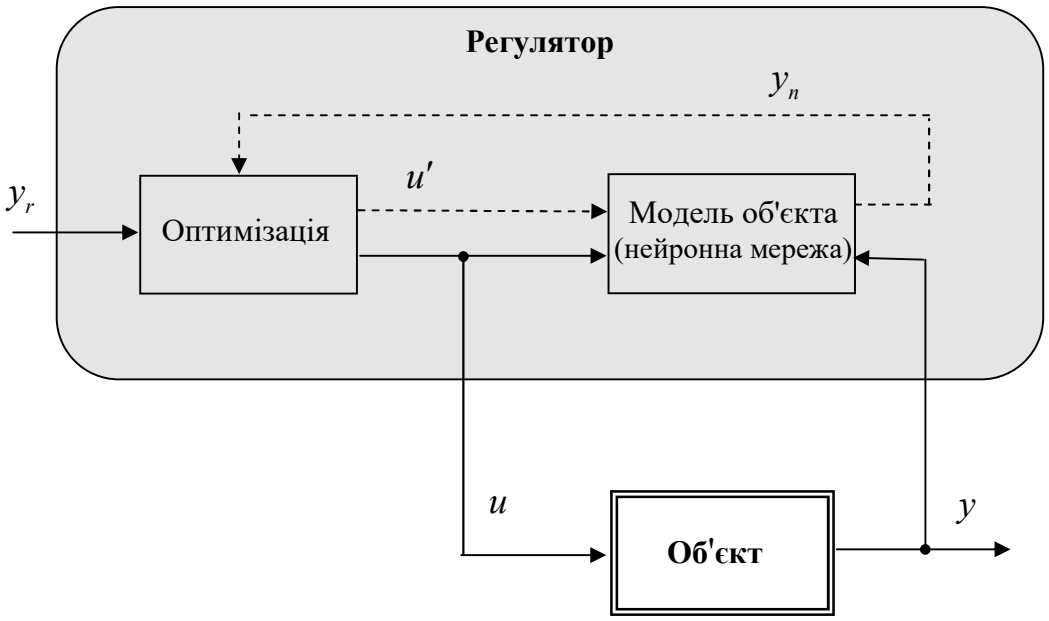


Рис.3.4. Структурна схема системи з регулятором,
що використовує принцип прогнозу

3.4 Динамічні характеристики двомасової системи

Синтез нейромережевих систем розглянемо на прикладі управляння двомасовою електромеханічною системою механізму підйому мостового крану. Математичну модель двомасової системи наведено у розділі 2.2 практичного заняття 2.

Для оцінки ефективності застосування нейрорегулятора доцільно виконати моделювання двомасової системи і потім порівняти з результатами моделювання нейромережевої системи.

Дослідження динамічних характеристик двомасової системи слід виконувати з застосуванням пакету прикладних програм MATLAB. Розрахунок може бути виконаний з використанням рівнянь стану системи, а також моделі Simulink системи, зображеної на рис.3.5. Графіки перехідних процесів перших чотирьох змінних стану системи по задаючій і збурюючій діях приведені на рис.3.6 і рис.3.7. Для формування вхідного сигналу $U_з$ і сигналу збурення $M_{ст}$ використані блоки Uniform Random, які формують ступінчастий сигнал з випадковою амплітудою, що знаходиться у встановлених межах. Інтервал, впродовж якого сигнал залишається незмінним, встановлено 10 с.

Як видно з графіків, перехідні процеси мають характер слабозатухаючих коливань, що недопустимо за умовами експлуатації мостового крану.

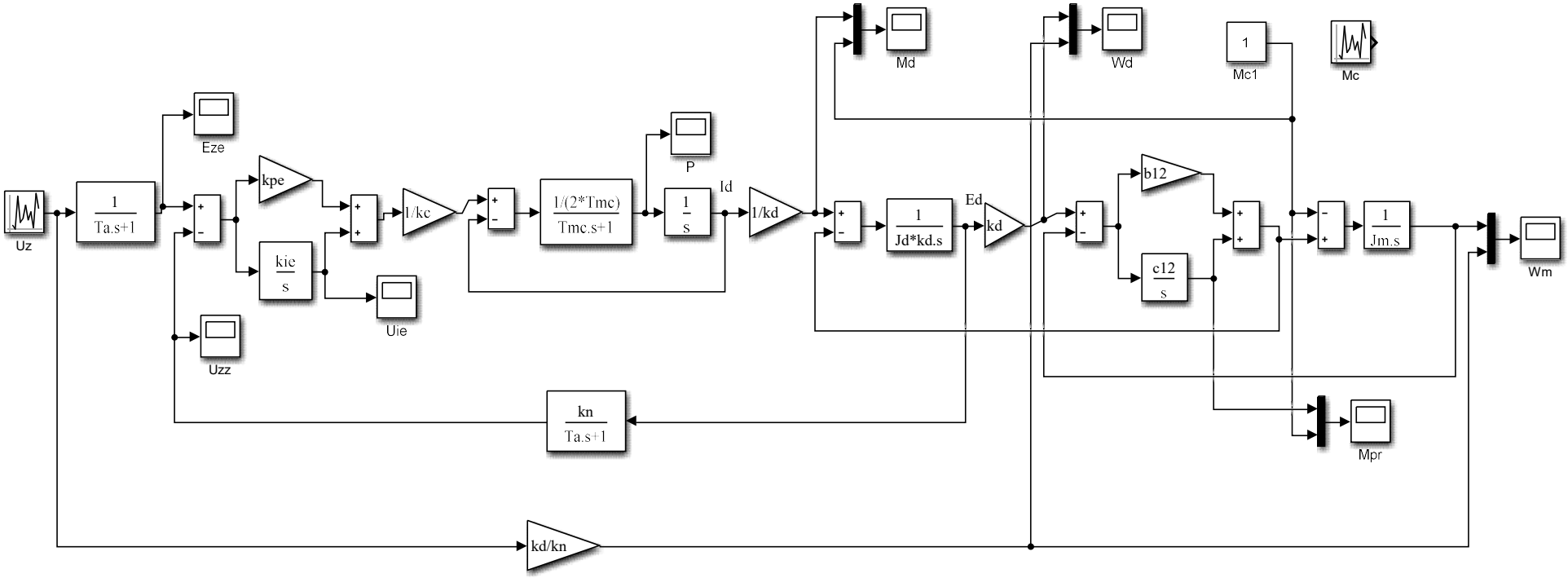
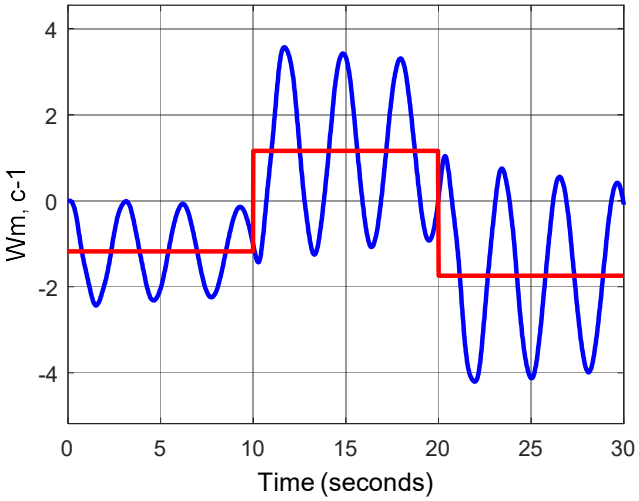
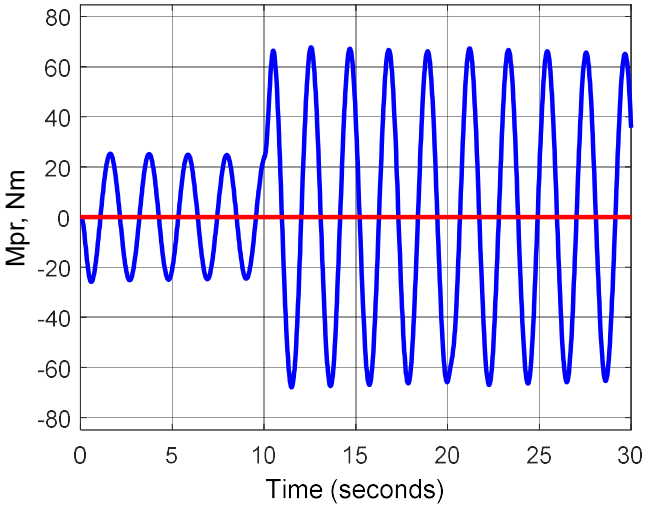


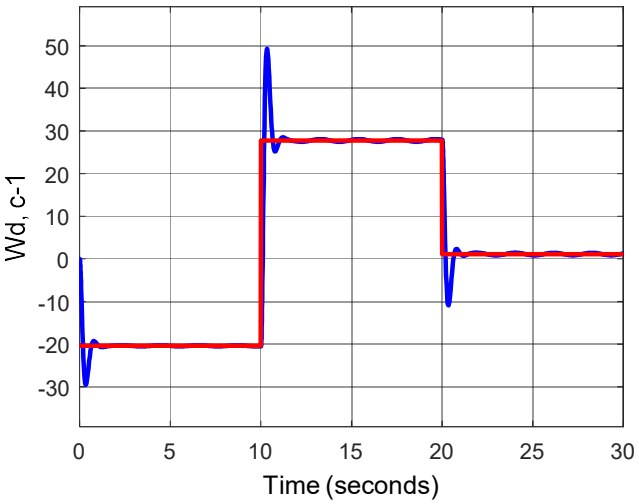
Рис. 3.5. Схема моделі двомасової системи, розроблена в Simulink системи MATLAB



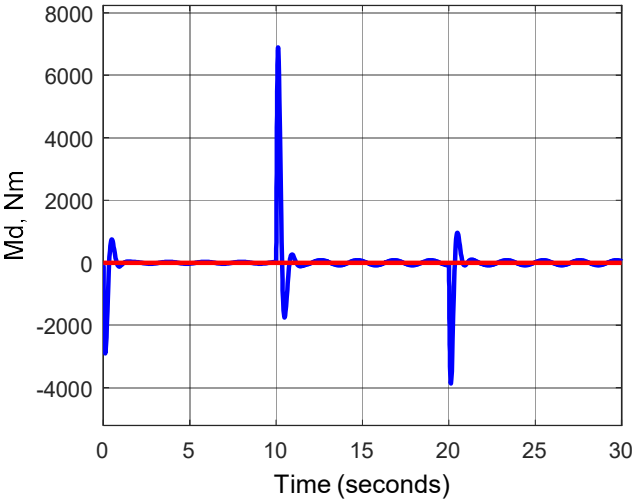
швидкість механізму по задаючій дії



момент пружності по задаючій дії

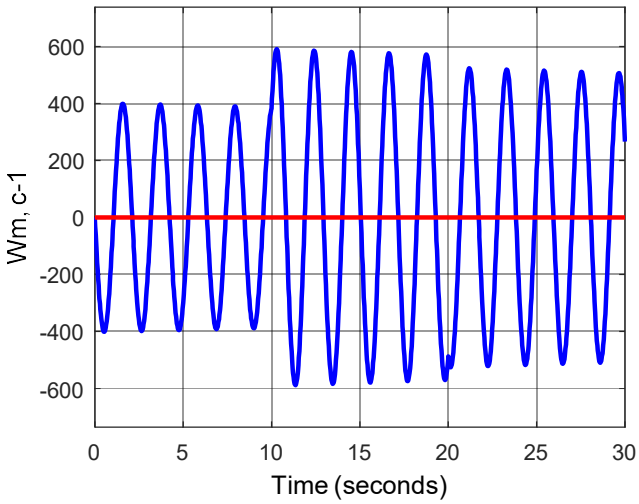


швидкість двигуна по задаючій дії

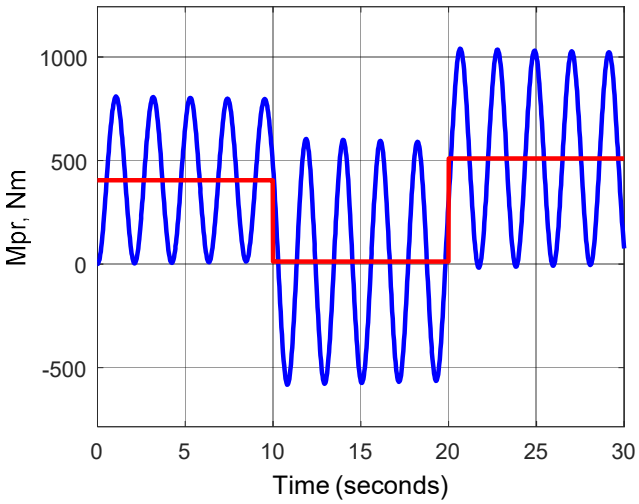


момент двигуна по задаючій дії

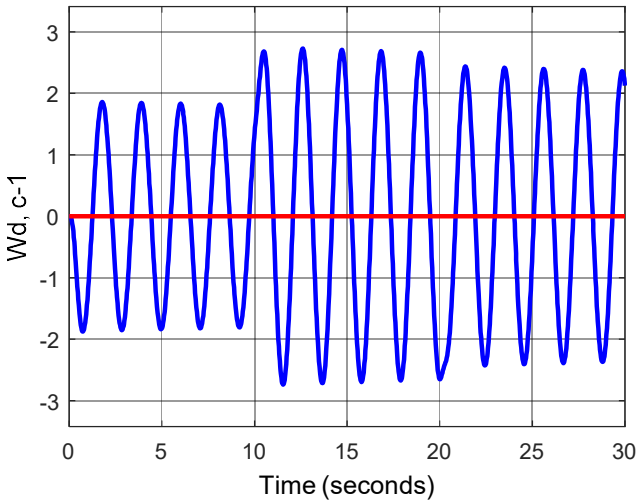
Рис.3.6. Графіки перехідних процесів двомасової системи по задаючій дії



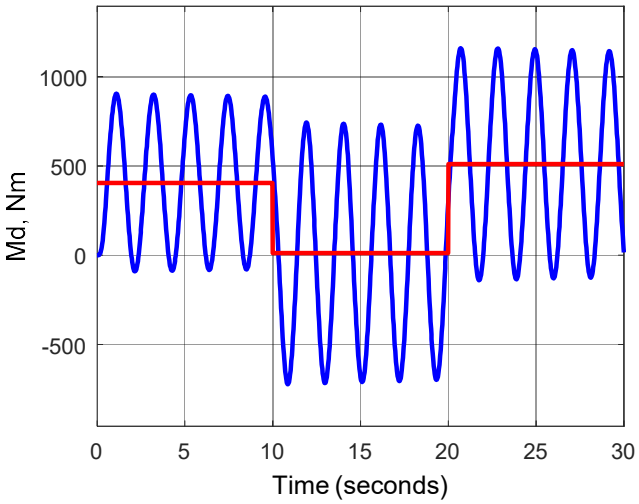
швидкість механізму по збурюючій дії



момент пружності по збурюючій дії



швидкість двигуна по збурюючій дії



момент двигунапо збурюючій дії

Рис.3.7. Графіки перехідних процесів двомасової системи по збурюючій дії

3.5 Методика синтезу нейрорегулятора NN Predictive Controller

При синтезі нейрорегулятора NN Predictive Controller використовуються наступні файли, розміщені в каталозі **toolbox/nnet/nncontrol** системи MATLAB.

Функції одновимірної оптимізації:

Csrchbac – пошук із зворотним прогоном;

Csrchbre – метод Брента, об'єднуючий методи золотого перетину і квадратичної інтерполяції;

Csrchcha – метод кубічної інтерполяції Чараламбуса;

Csrchgol – метод золотого перетину;

Csrchhyb – гібридний метод бісекції і кубічної інтерполяції.

Функції для синтезу управління з прогнозом:

Calcjjdjj – обчислення функціонала якості і його градієнта;

Predopt – оптимізація регулятора з прогнозом;

Dyduvar – обчислення приватних похідних виходу по входу.

Моделі Simulink.

Predcstr – GUI – додаток для регулятора з прогнозом;

Prest3sim2 – нейромережева модель управління використовувана M-функцією deport для прогнозу процесу в майбутньому.

Допоміжні функції:

Nncontrolutil – утиліта, яка забезпечує доступ до приватних функцій у середовищі Simulink;

Nnident.m – головна функція для ідентифікації об'єкта, розташована в каталозі **private.** Вона надає інтерфейс GUI для користувача, виконує генерацію навчальної вибірки, створює та тренує нейронну мережу.

Слід зазначити, що в методичних рекомендаціях до практичних робіт дається скорочений опис вікон, що використовуються при синтезі нейрорегуляторів. Повний опис наведено в навчально-методичному посібнику до даної дисципліни.

У вікні системи Simulink формується схема системи із структурою, показаною на рис. 3.8.

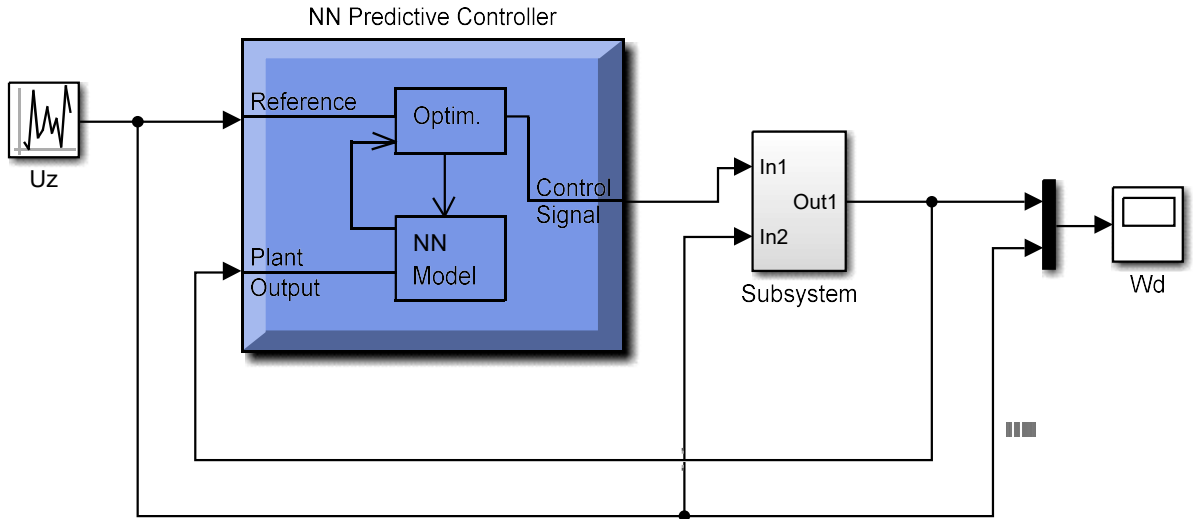


Рис. 3.8. Схема системи управління з нейрорегулятором NN Predictive Controller

Ця структура включає блок керованого об'єкту **Subsystem** і блок регулятора **NN Predictive Controller**, а також блоки генерації еталонного ступінчастого сигналу з випадковою амплітудою **Random Reference**, блок побудови графіків. Особливість цієї структури полягає в тому, що вона виконується не тільки функції блок-схеми системи SIMULINK, але і функції графічного інтерфейсу користувача GUI.

Структурна схема нейрорегулятора **NN Predictive Controller** представлена на рис.3.9. Ця схема відображається у відокремленому вікні, яке вибирається через пункт меню **Look Under Mask** блоку NN Predictive Controller.

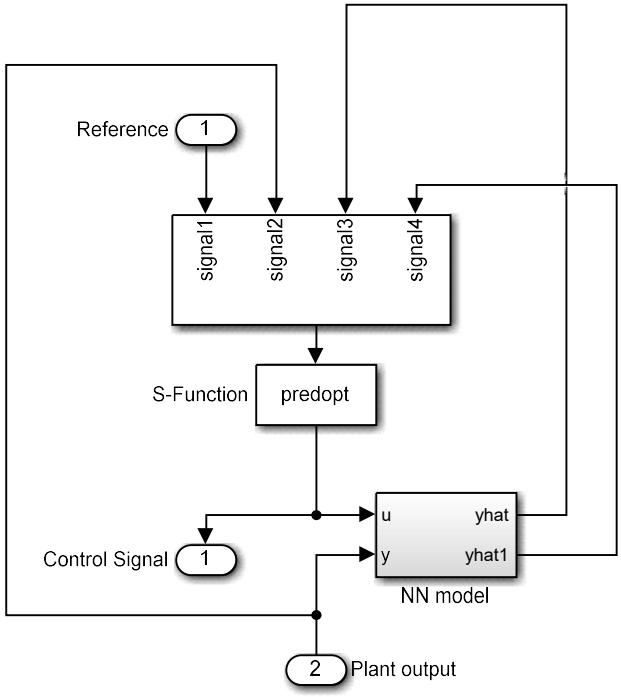


Рис. 3.9.Структурна схеманейрорегулятора

Процес створення нейрорегулятора починається з активації блоку **NN Predictive Controller**. З'являється вікно, показане на рис. 3.10. Це вікно виконує функції графічного інтерфейсу користувача.

У рамці вікна рис. 3.10 відображається повідомлення, що вказує на подальші кроки: "*Perform plant identification before controller configuration*". Це означає, що перед конфігуруванням регулятора необхідно спершу провести ідентифікацію керованого об'єкту, тобто створити його нейромережеву модель, використовуючи спеціальну процедуру **Plant Identification**.

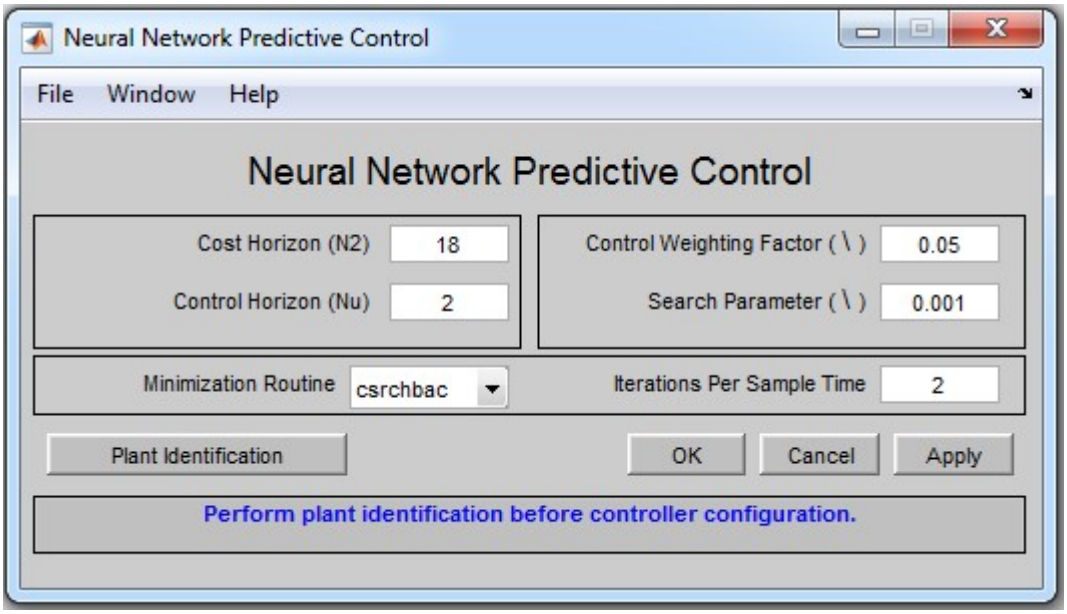


Рис. 3.10. Вікно завдання параметрів регулятора

Вікно **Plant Identification** зображено на рис.3.11. Ця структура є універсальною і може бути використана для розробки нейромережевих моделей для будь-якого динамічного об'єкту, який описаний за допомогою моделі SIMULINK. В завданні, що розглядається, моделлю є двомасова система.

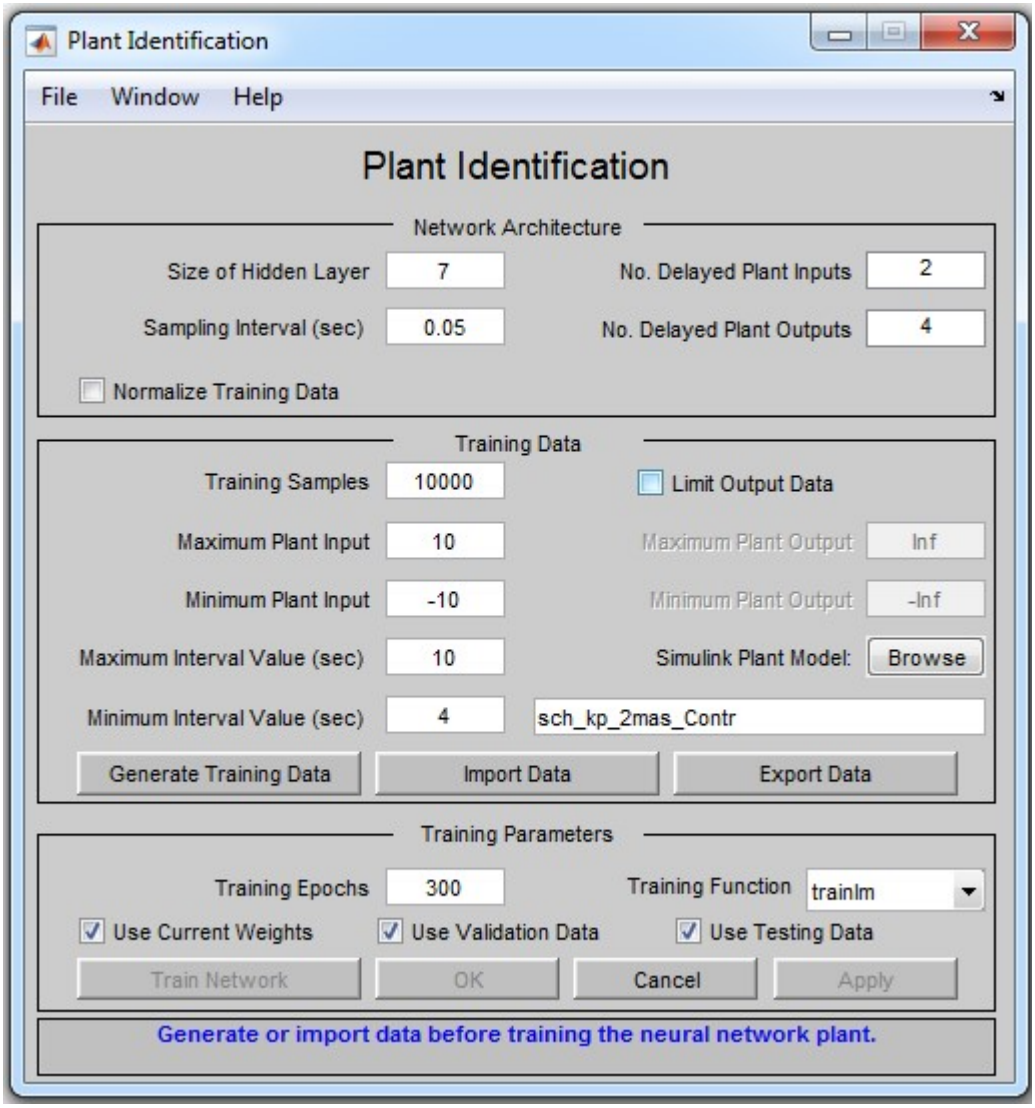


Рис. 3.11. Вікно ідентифікації керованої системи

Процедура ідентифікації дозволяє побудувати нейронну мережу, яка відтворюватиме динаміку керованого об'єкту. Оскільки ця модель має застосовуватися під час налаштування регулятора, її слід створити перед розпочатком процесу налаштування регулятора.

Процедура ідентифікації вимагає завдання ряду параметрів, які задаються з використанням вікна **Plant Identification**. Дане вікно містить кілька секцій.

Опис вікна Plant Identification.

Секція Network Architectuer (Архітектура нейронної мережі). Параметри, що адаються в цій секції визначають архітектуру нейронної мережі, яка використовується для ідентифікації і моделювання динаміки системи. Вибір правильних параметрів секції Network Architecture є ключовим етапом в ідентифікації системи, оскі-

льки від них залежить здатність моделі адекватно відтворювати динаміку досліджуваної системи.

Size of Hidden Layer (розмір прихованого шару). Параметр Size of Hidden Layerвказує на кількість нейронів у прихованому шарі нейронної мережі, яку ви використовуєте для моделювання або ідентифікації об'єкта або системи.

Sampling Interval (sec) (інтервал вибірки в секудах). Параметр Sampling Interval (sec) визначає інтервал між двома послідовними моментами знімання даних.

No. Delayed Plant Inputs (кількість затриманих входів системи). Параметр No. Delayed Plant Inputs вказує на кількість запізнених (затриманих) входів системи, які враховуються при ідентифікації динаміки системи.

No. Delayed Plant Outputs (кількість затриманих виходів системи). Параметр No. Delayed Plant Outputs вказує на кількість запізнених (затриманих) виходів системи, які враховуються при ідентифікації динаміки системи.

Normalize Training Data (нормалізація даних навчання). Вікна контролю нормування навчальних даних. Normalize Training Data вказує на необхідність нормалізації даних під час навчання моделі системи.

Секція Training Data (дані навчання). Training Data - це набір даних, які використовуються для навчання моделі ідентифікації системи. Ці дані представляють собою вхідні та вихідні сигнали системи, і їх використовують для створення математичної моделі системи.

Training samples (навчальні зразки). Довжина навчальної вибірки (кількість точок знімання інформації). Навчальна вибірка включає в себе вхідні та вихідні дані, які використовуються для створення моделі системи.

Maximum Plant Input (максимальне значення вхідного сигналу). Параметр Maximum Plant Input вказує на максимальне значення вхідного сигналу (величини, яка керує системою) під час ідентифікації динаміки системи. Цей параметр встановлює верхню межу діапазону значень вхідного сигналу, які будуть використовуватися під час навчання моделі.

Minimum Plant Input (мінімальне значення вхідного сигналу). Параметр Minimum Plant Input вказує на мінімальне значення вхідного сигналу (величини, яка керує системою) під час ідентифікації динаміки системи. Цей параметр встановлює нижню межу діапазону значень вхідного сигналу під час навчання моделі.

Maximum Interval Value (sec) (максимальне значення інтервалу). Параметр Maximum Interval Value (sec) вказує на максимальну довжину інтервалу часу в секундах, протягом якого використовуються вхідні та вихідні дані для ідентифікації динаміки системи, тобто максимальний інтервал ідентифікації. Цей параметр дозволяє обмежити часовий період, на який поширюється навчальний набір даних, використовуваний під час ідентифікації.

Minimum Interval Value (sec) (мінімальне значення інтервалу). Параметр Minimum Interval Value (sec) вказує на мінімальну довжину інтервалу часу в секундах, протягом якого використовуються вхідні та вихідні дані для ідентифікації динаміки системи, тобто мінімальний інтервал ідентифікації. Цей параметр дозволяє

обмежити часовий період, на який поширюється навчальний набір даних, використовуваний під час ідентифікації.

Limit Output Data (обмеження вихідних даних). Вікно контролю, що дозволяє обмежити об'єм вихідних даних, тільки при включеному вікні будуть доступні 2 наступних вікна редагування тексту.

Maximum Plant Output. Максимальне значення вихідного сигналу.

Minimum Plant Output. Мінімальне значення вихідного сигналу.

Параметри Maximum Plant Output і Minimum Plant Output вказують на обмеження вихідних даних, які використовуються під час ідентифікації динаміки системи. Ці параметри можуть бути використаними для вибору конкретного діапазону вихідних даних для аналізу та навчання моделі системи.

Simulink Plant Model (Simulink модель обєкту). Завдання моделі Simulink з вказівкою вхідних і вихідних портів, використовуваних при побудові нейромережевої моделі керованої системи. Вказується модель об'єкта або системи, яка використовується для ідентифікації.

За допомогою кнопки **Browser** вибирається модель Simulink об’єкту управління; в даній роботі це схема моделі, показана на рис. 3.12.

Generate Training Data (генерація навчальних даних). Кнопка запуску процесу генерації навчальної послідовності, тобто створення навчальних даних для ідентифікації динаміки системи на основі параметрів та моделі системи. Ця функція дозволяє симулювати дані для навчання моделі, коли реальні навчальні дані не доступні.

Import Data. Імпорт навчальної послідовності з робочої області або файлу даних.

Export Data. Експорт даних, що згенерували, в робочу область або MAT – файл.

Секція Training Parameters (параметри навчання). Training Parameters - це набір налаштувань, які визначають, як саме буде проводитися процес навчання моделі системи на основі навчальних даних. Ці параметри дозволяють керувати процесом навчання ідентифікаційної моделі та впливати на його результати.

Training Epochs (епохи навчання). Training Epochs вказують на кількість ітерацій (повних циклів навчання) для навчання моделі або алгоритму ідентифікації. Кожна епоха представляє собою одну ітерацію, під час якої модель навчається на підготовчих даних та оновлює свої внутрішні параметри, щоб покращити свою точність.

Training function (навчальна функція). Training Function вказує на функцію, яка використовується для навчання нейромережевої моделі об'єкта або системи. Ця функція визначає, яким чином модель навчається на навчальних даних та які параметри моделі оптимізуються під час навчання.

Use Current Weights (використовувати поточні ваги). Вікно контролю, що дозволяє підтвердити використання поточних вагів нейронної мережі.

Параметр Use Current Weights вказує, чи слід використовувати поточні ваги для ідентифікації системи під час навчання моделі. Цей параметр впливає на початкову точку або ініціалізацію процесу навчання.

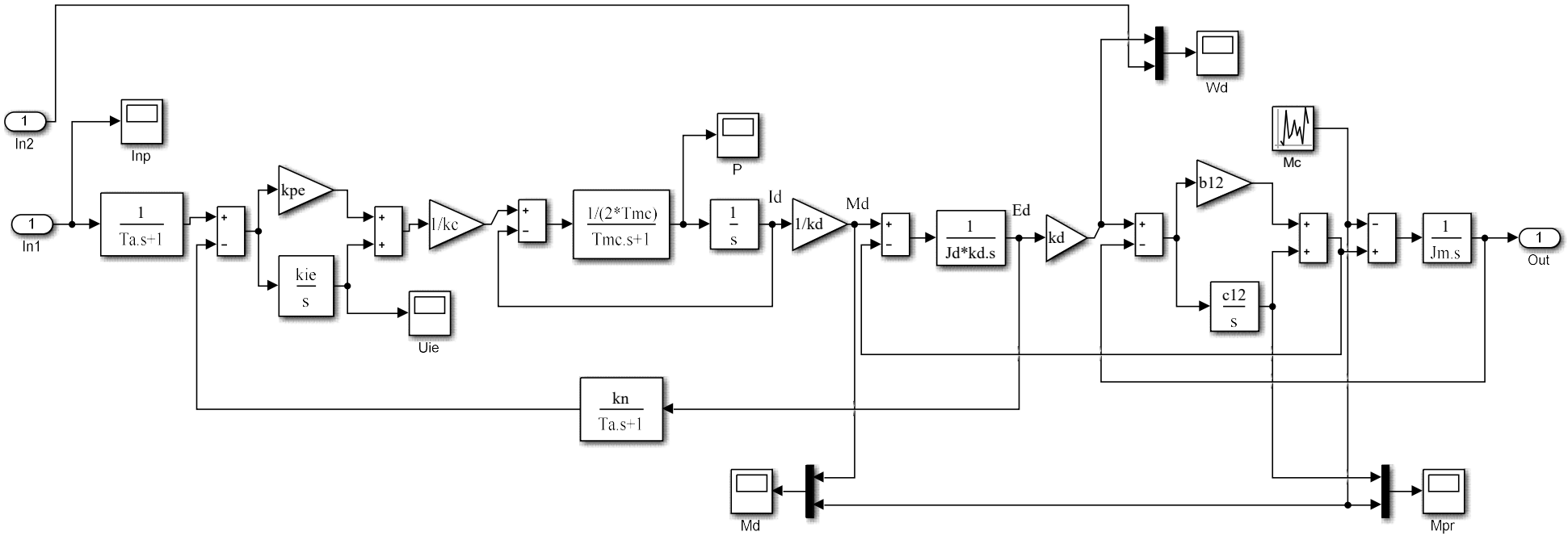


Рис. 3.12. Модель об'єкту управління нейрорегулятора

Use Validation Data (використовувати дані валідації). Вікно контролю, що дозволяє вказати, чи слід використовувати окремий набір даних для валідації моделі під час навчання ідентифікаційної системи. Валідація - це процес оцінки якості побудованої моделі на основі навчальних даних.

Use Testing Data (використовувати тестові дані). Вікно контролю, що дозволяє вказати, чи слід використовувати використовувати окремий набір тестових даних для оцінки якості побудованої моделі системи після її навчання. Тестові дані - це дані, які модель не бачила під час навчання або валідації.

Виклик процедури **Generate Training Data** призведе до запуску програми для створення навчальної послідовності. Ця програма генерує навчальні дані шляхом впливу послідовності випадкових ступінчастих сигналів на модель Simulink керованого об'єкту. Графіки входового та вихідного сигналів об'єкту управління відображаються на екрані (див. рис. 3.13). Після завершення процесу створення навчальної послідовності користувач має можливість прийняти (**Accept Data**) або відхилити (**Reject Data**) згенеровані дані.

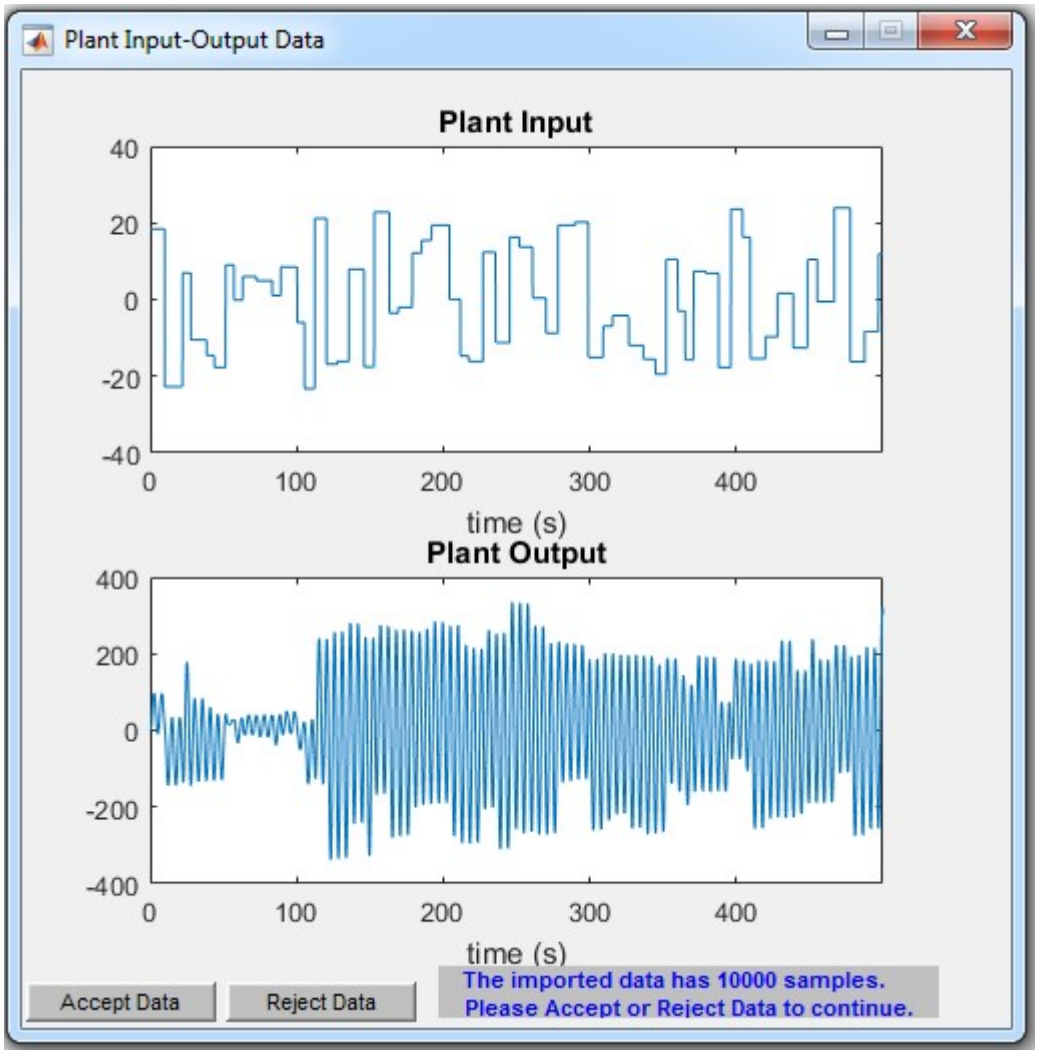


Рис. 3.13. Графіки вхідного і вихідного сигналів при
генерації навчальної послідовності

Якщо дані прийняті, програма відкриє модифіковане вікно **Plant Identification** (див. рис.3.14). У цьому вікні деякі елементи стають недоступними, а кнопка Generate Training Data замінюється кнопкою Erase Generate Data, яка дає можливість видалити згенеровані дані.

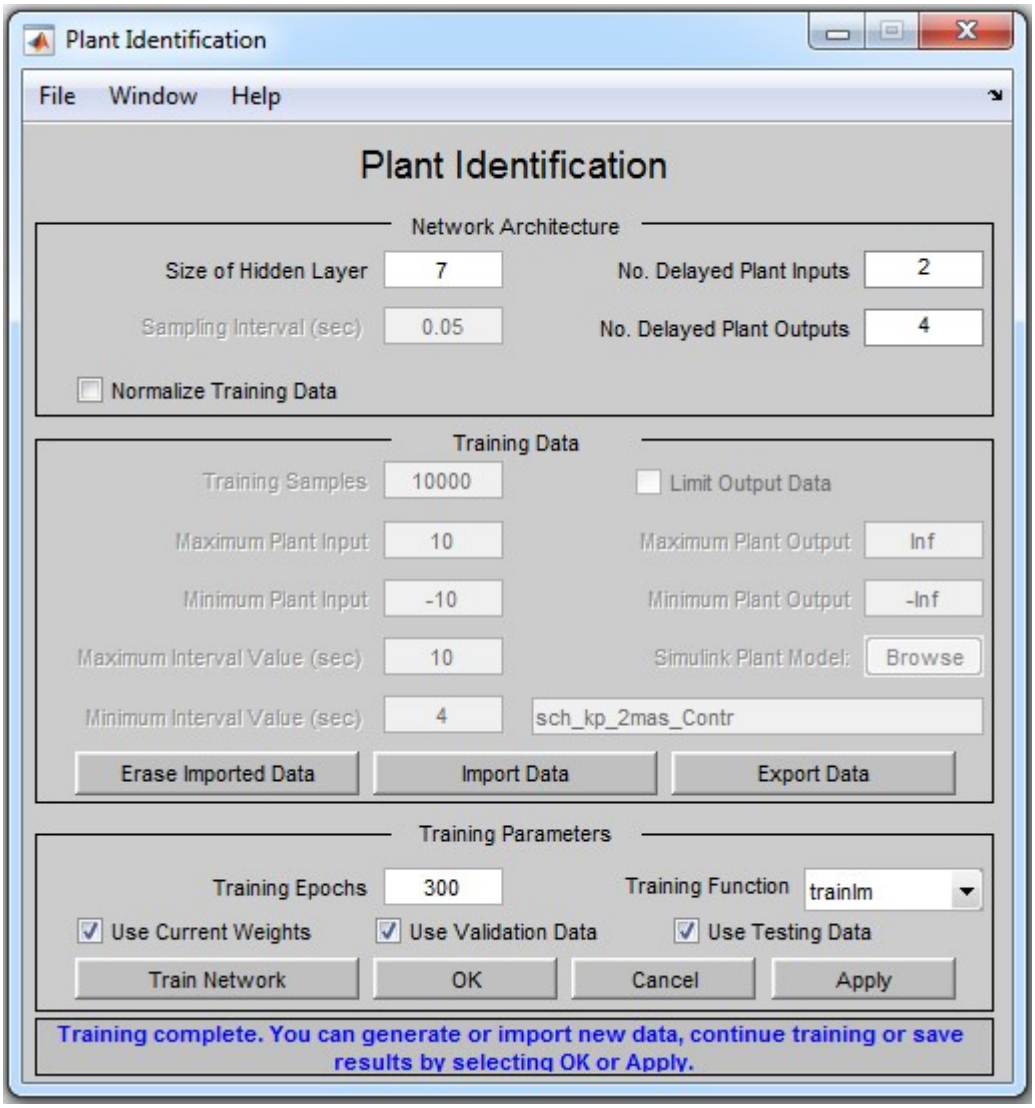


Рис.3.14. Вид вікна Plant Identification після генерації навчальної послідовності

Після натискання на кнопку **Train Network** відбувається створення мережі з прямим розподілом сигналу за допомогою М-функції **newff**. Ця функція не лише створює мережу з ім'ям netn, але й ініціалізує її ваги та зсуви, готуючи нейронну мережу до навчання. Модель нейронної мережі може бути відображена у системі Simulink за допомогою оператора **gensim(netn)** (див. рис. 3.15).

Елементи нейронної мережі відповідають заданим параметрам: розмір прихованого шару, кількість затримок на вході моделі та кількість затримок на виході моделі. Кожен наступний елемент стає доступним у новому вікні після подвійного клацання на попередньому елементі. На основі цих елементів у системі Simulink можна побудувати схему мережі, як показано на рис. 3.16.

Ця мережа є статичною (статичні мережі не містять елементів затримки та зворотних зв'язків). Вона використовує один вектор входу з шістьма елементами, має два шари: задана кількість нейронів у першому (прихованому) шарі та один нейрон у другому (вихідному) шарі. Активаційні функції: гіперболічний тангенс (tansig) у першому шарі та лінійна функція (purelin*)* у другому шарі.

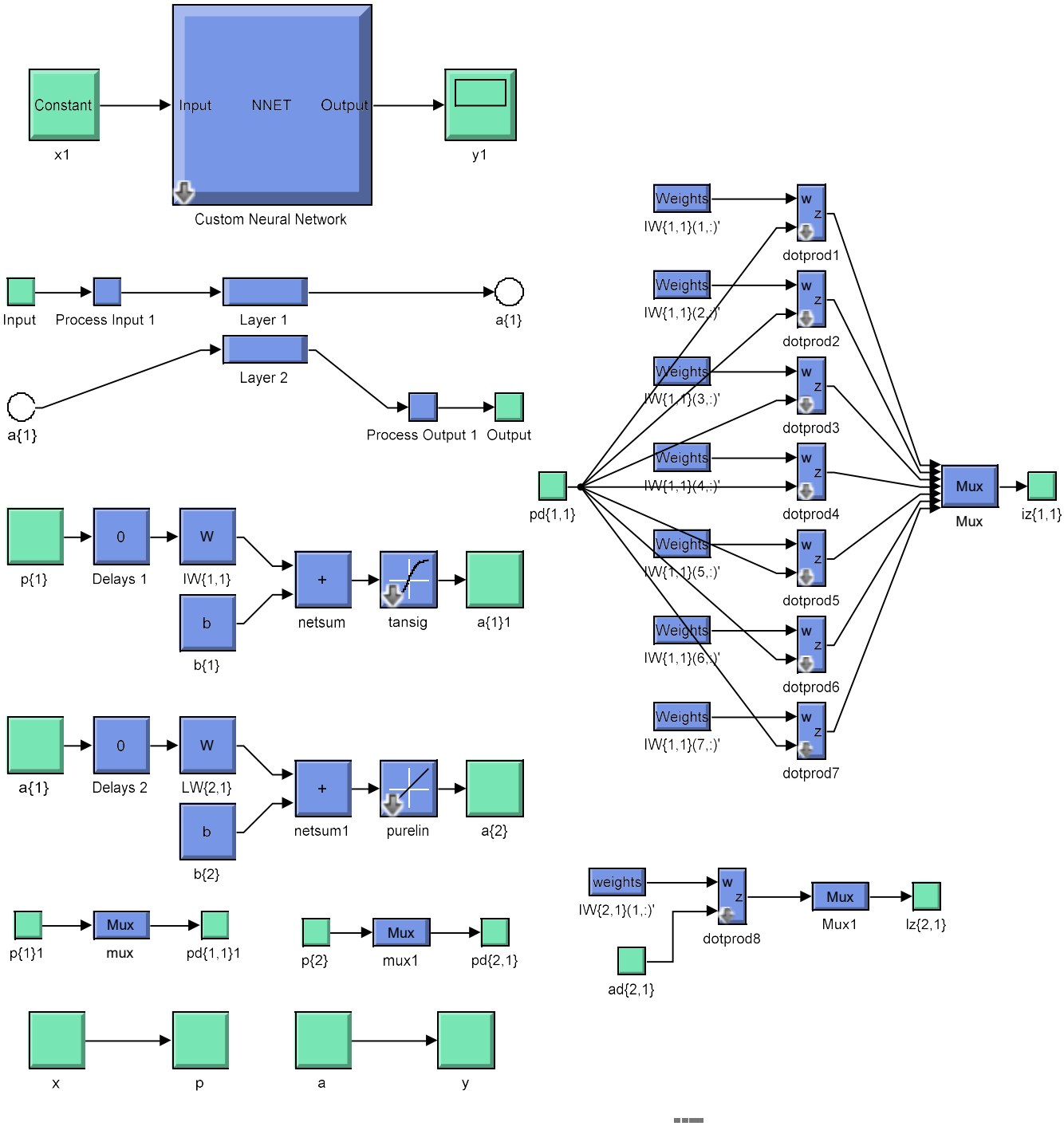


Рис. 3.15. Моделі елементів мережі з прямою передачею сигналу, реалізовані в Simulink

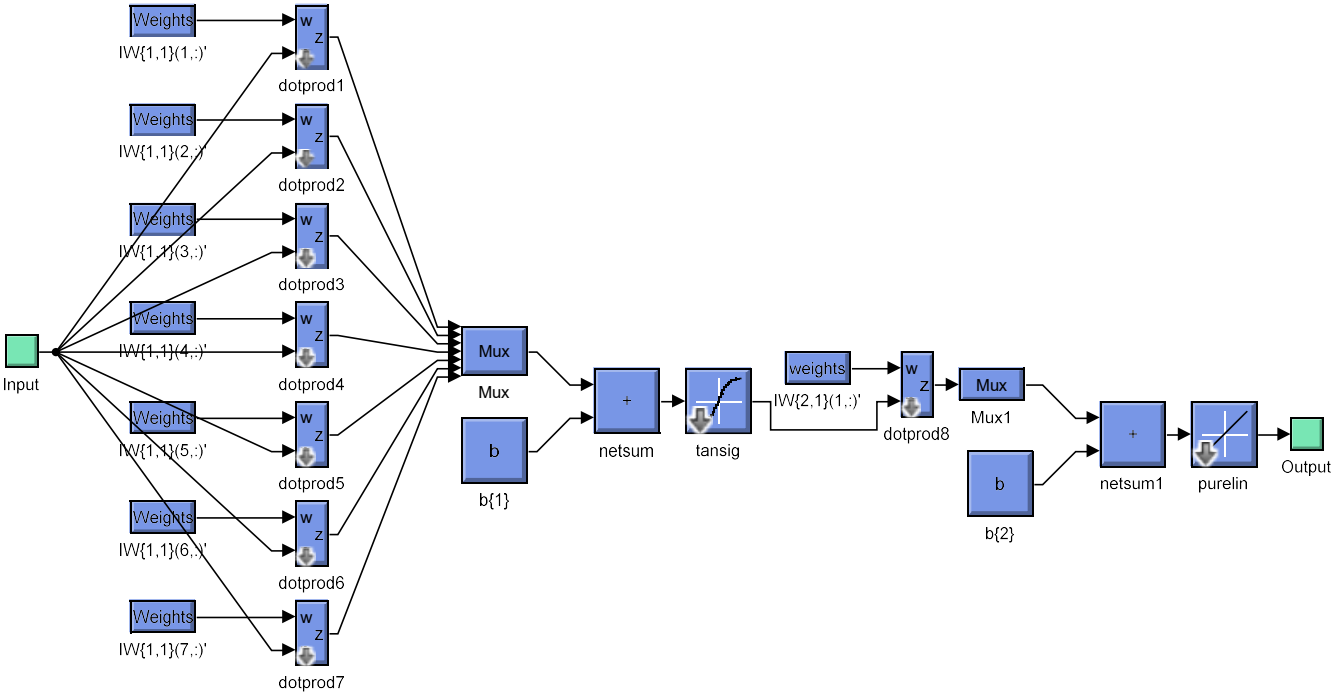


Рис. 3.16. Модель статичної мережі з прямою передачею сигналу, побудована в Simulink

Після створення мережі починається процес її навчанні. Процес та результати навчання наочно демонструються за допомогою діалогового вікна **Neural Network Training (nntraintool)**, показаного на рис..3.17. Наведене вікно - це інтерфейс для навчання нейронних мереж у Neural Network Toolbox в MATLAB. Воно є потужним інструментом для навчання і налаштування нейронних мереж з візуалізацією та можливістю моніторингу процесу навчання.

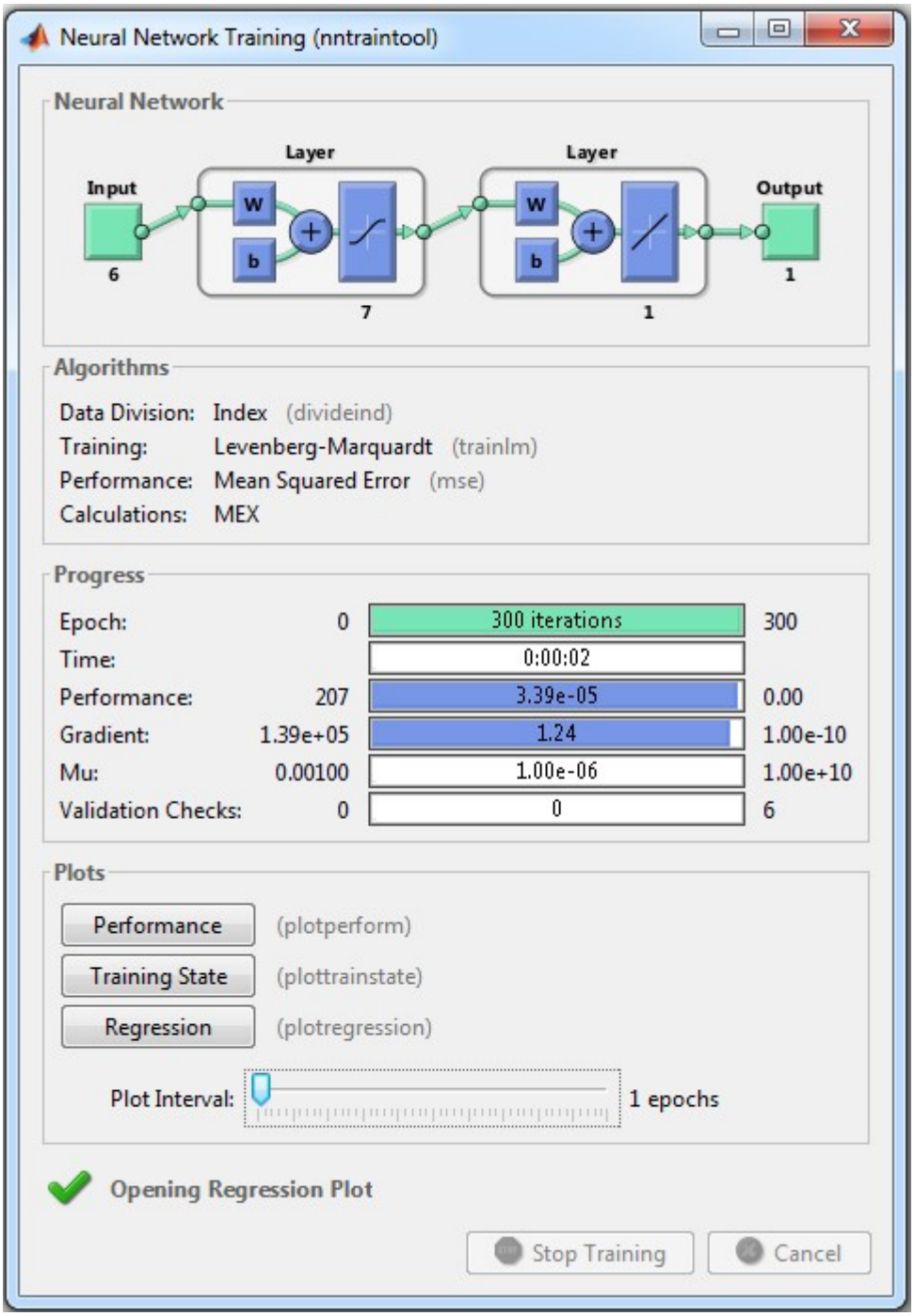


Рис. 3.17. Головне вікно, що відображає хід навчання
та результати навчання нейронної мережі

Опис вікна Neural Network Training (nntraintool)

Вікно Neural Network Training (nntraintool) має наступні секції.

Секція Neural Network (нейронна мережа). У секції наводиться структура згенерованої нейронної мережі.

Секція Algorithms (алгоритми). Секція призначена для відображення різних алгоритмів та параметрів, які використовуються під час навчання нейронної мережі.

Data Division (ділення даних). Параметр Data Division: Index (поділ даних: індекс) вказує на конкретні індекси прикладів даних, які використовуються для на-

вчання, перевірки та тестування нейронної мережі. Цей параметр використовується для визначення, як саме дані розділяються на навчальні, валідаційні та тестові набори під час навчання мережі. Якщо ділення даних не відбувається, то рядок Data Division у вікні Neural Network Training (nntraintool) відсутній.

Training (навчання). Функція навчання, яка визначає, як нейронна мережа оновлює свої ваги та зсуви під час навчання. Популярні функції включають в себе Backpropagation, Levenberg-Marquardt, Bayesian Regularization та інші.

Встановлення Training: Levenberg-Marquardt (trainlm) вказує на використання алгоритму тренування, відомого як алгоритм Левенберга-Марквардта (Levenberg-Marquardt, позначається як trainlm), для навчання нейронної мережі в середовищі MATLAB. Алгоритм Левенберга-Марквардта - це широко використовуваний метод оптимізації для навчання нейронних мереж. Цей метод комбінує в собі ідеї з методу найменших квадратів і методу Ньютона для оптимізації функції втрати мережі. Основна ідея алгоритму Левенберга-Марквардта полягає в швидкому спаді значення функції втрати і пошуку локального мінімуму шляхом оновлення вагових коефіцієнтів мережі. Він добре справляється зі складними задачами навчання, де інші методи можуть справлятися гірше.

Performance (виконання). Термін Performance вказує на показник якості навчання нейронної мережі на основі обраної функції втрат (loss function) та даних для перевірки (validation dataset). Цей показник відображає, наскільки добре нейронна мережа працює під час навчання на навчальних та валідаційних даних і допомагає визначити, коли навчання мережі має бути завершено.

Параметр Performance: Mean Squared Error (mse) вказує на використання середньоквадратичної помилки (MSE) для оцінки точності нейронної мережі під час навчання. Середньоквадратична помилка (Mean Squared Error, MSE) є однією з найпоширеніших метрик у задачах регресії. Вона обчислюється як середнє значення квадратів різниці між прогнозованими значеннями моделі (у цьому випадку нейронної мережі) та відомими або фактичними значеннями у наборі даних.

Calculations (розрахунки). Параметр відноситься до кількості обчислень або розрахунків, які виконуються під час навчання нейронної мережі. Цей параметр може бути важливим для моніторингу продуктивності та визначення часу, який потрібен для завершення процесу навчання.

Параметр **Calculations: MEX** вказує на використання MEX-функцій під час навчання нейронної мережі в середовищі MATLAB. MEX (MATLAB Executable) - це механізм, який дозволяє виконувати оптимізований обчислювальний код, написаний на C, C++ або Fortran, безпосередньо в середовищі MATLAB.

Секція Progress (прогрес). У секції вказується на інформація про прогрес навчання нейронної мережі під час процесу навчання. Містить інформацію, яка дозволяє в режимі реального часу відстежувати прогрес та якість навчання моделі. Це допомагає контролювати процес навчання та вживати необхідні дії для поліпшення якості моделі.

Epoch (епоха). Вказує на один повний прохід через весь навчальний набір даних під час навчання нейронної мережі. Під час кожної епохи навчання всі приклади з навчального набору даних проходять через нейронну мережу один раз, і ваги мережі оновлюються відповідно до результатів навчання на цій епохі. Епохи допомагають відстежувати процес навчання та роблять його більш керованим.

Time (час). Вказує на час, який вже було витрачено на процес навчання нейронної мережі. Цей параметр служить для відстеження тривалості процесу навчання і може бути корисним для оцінки часових аспектів навчання моделі.

Performance (виконання). Ввказує на показник якості навчання нейронної мережі на основі обраної функції втрат (loss function) та даних для перевірки (validation dataset). Цей показник відображає, наскільки добре нейронна мережа працює під час навчання на навчальних та валідаційних даних. Для задач регресії зазвичай використовується середньоквадратична помилка (Mean Squared Error, MSE), яка визначає середню квадратичну помилку між прогнозованими значеннями і справжніми значеннями цільової змінної.

Gradient (градієнт). Вказує на градієнт функції втрат (loss function) відносно параметрів нейронної мережі під час навчання. Градієнт - це вектор, який вказує на те, як потрібно змінювати ваги та зсуви (biases) нейронної мережі, щоб мінімізувати функцію втрат і покращити її ефективність.

Під час навчання нейронної мережі використовується метод зворотного поширення помилки (backpropagation), і градієнт відіграє важливу роль у цьому процесі. Градієнт обчислюється для кожного параметра (ваги та зсуви) і показує, в якому напрямку і наскільки слід змінити ці параметри, щоб зменшити значення функції втрат.

Mu. Відноситься до тау-параметра, який використовується в алгоритмах оптимізації для регулювання швидкості навчання нейронної мережі.

Validation Checks (перевірки підтвердження). Вказується кількість перевірок (checks) якості моделі, які виконуються під час навчання нейронної мережі з використанням набору даних для перевірки (validation dataset). Validation Checks допомагає контролювати процес навчання та визначити, коли навчання слід припинити або зупинити, щоб уникнути перенавчання.

Секція Plots (графіки). Містить графіки та візуалізацію прогресу навчання нейронної мережі під час процесу навчання. Ця секція допомагає візуально аналізувати якість та хід навчання мережі.

Кнопка **Performance** використовується для відображення показників якості під час навчання нейронної мережі. При натисканні на цю кнопку відкривається вікно **Neural Network Training Performance**, показане на рис. 3.18.

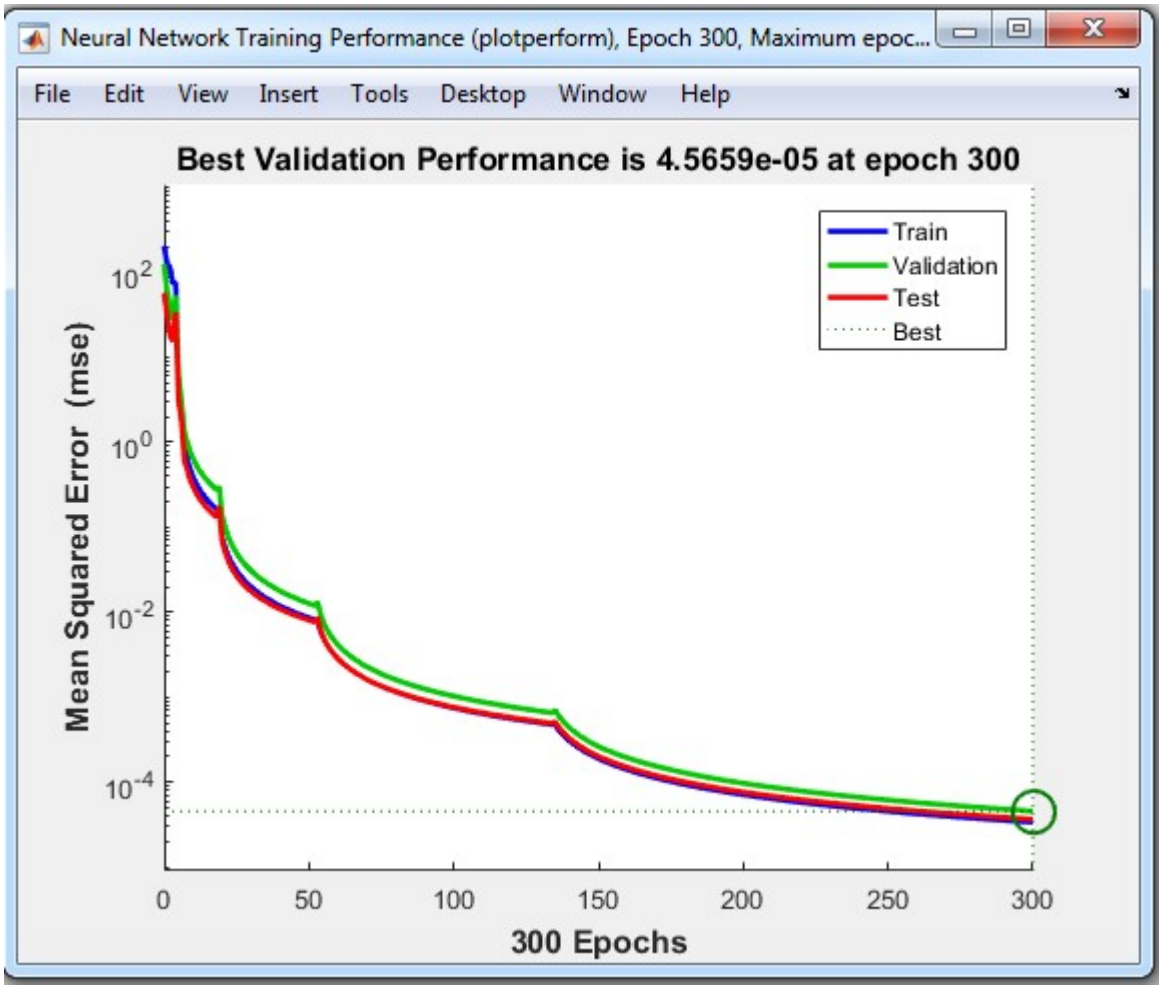


Рис. 3.18. Вікно контролю процесу навчання

У вікні Neural Network Training Performance відображаються графіки **Mean Squared Error (mse)**, які показують, як змінюється середньоквадратична помилка на навчальних, валідаційних (перевірочних) та тестових наборах даних під час кожної ітерації навчання. Графіки MSE показують, як змінюється точність мережі під час навчання і чи є ознаки перенавчання (overfitting), коли помилка на валідаційних та тестових даних починає зростати. Метою є мінімізувати MSE на усіх наборах даних для досягнення найкращої моделі.

Кнопка **Training State (стан навчання).** При натисканні на кнопку Training State відкривається вікно **Neural Network Training State** (рис. 3.19**)** в якому відображається інформацію про стан навчання нейронної мережі на певний момент часу під час процесу навчання.

Опис вікна Neural Network Training State

У вікні Neural Network Training State відображаються наступні графіки.

Графік Gradient показує зміну значення градієнта під час навчання нейронної мережі. Градієнт в контексті навчання нейронних мереж є вектором, який вказує напрямок та міру найшвидшого зростання функції втрати (або помилки) мережі в точці навчання. Градієнт використовується для оновлення вагових коефіцієнтів мережі під час оптимізації.

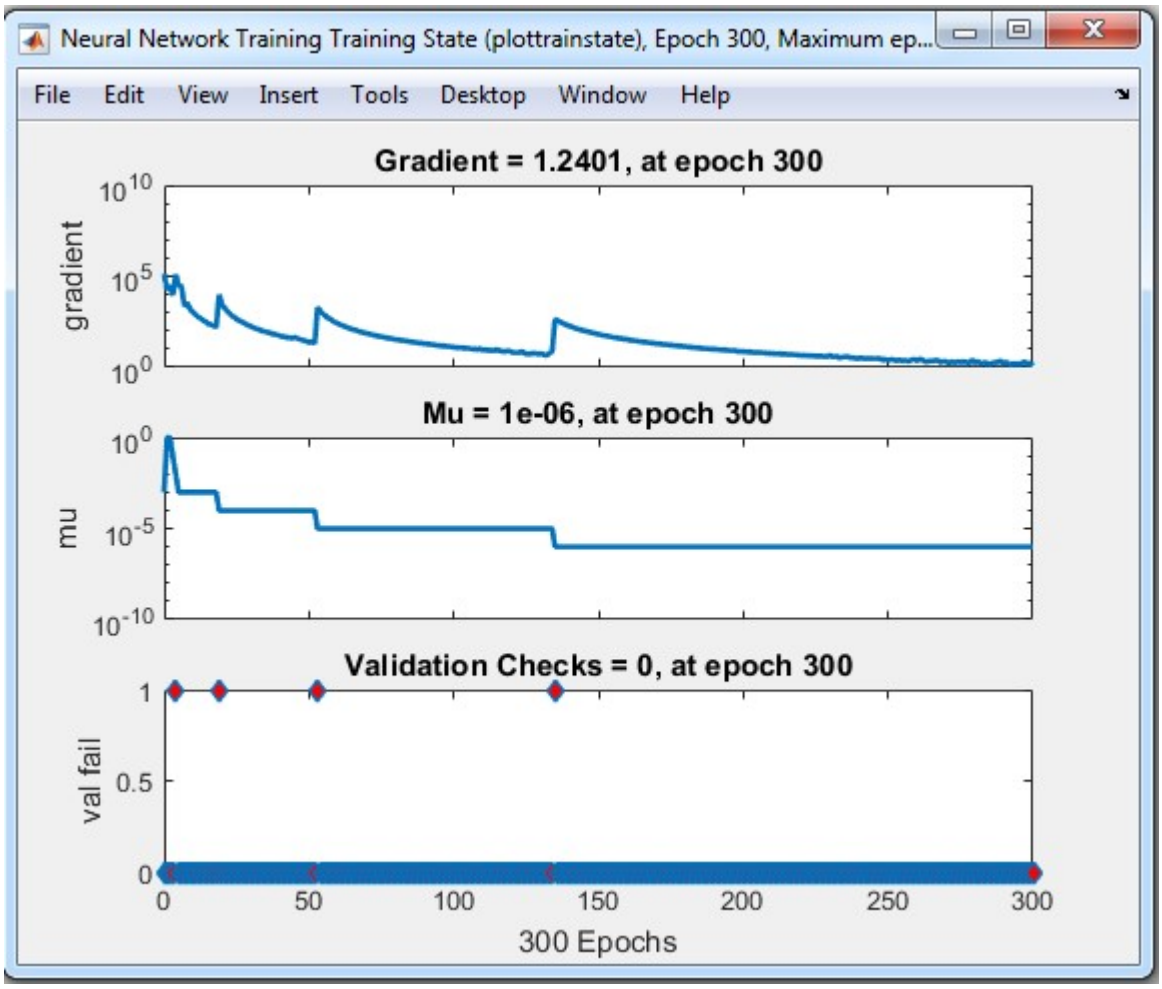


Рис.3.19. Вікно Neural Network Training State

Графік mu показує зміну значення параметра mu під час навчання нейронної мережі. Параметр mu відноситься до швидкості навчання та регуляризації мережі, і його зміна вказує на те, як адаптується швидкість навчання під час тренування.

Графік Validation Failure (або **val fail**) **(помилка підтвердження**) показує, як змінюється величина, що відображає кількість невдач (помилок) під час перевірки (валідації) нейронної мережі під час навчання.

Продовження опису вікна Neural Network Training (nntraintool)

Rнопка **Regression (регресія)** використовується для вибору та налаштування параметрів навчання нейронної мережі для завдань регресії.

При натисканні на кнопку **Regression** відкривається вікно **Neural Network Training Regression** (рис. 3.20), у якому показані графіки та інформація, які допомагають в оцінці та аналізі продуктивності нейронної мережі під час навчання. Для наочної оцінки якості навчання приведені графіки регресії для Training, Validation та Test наборів даних.

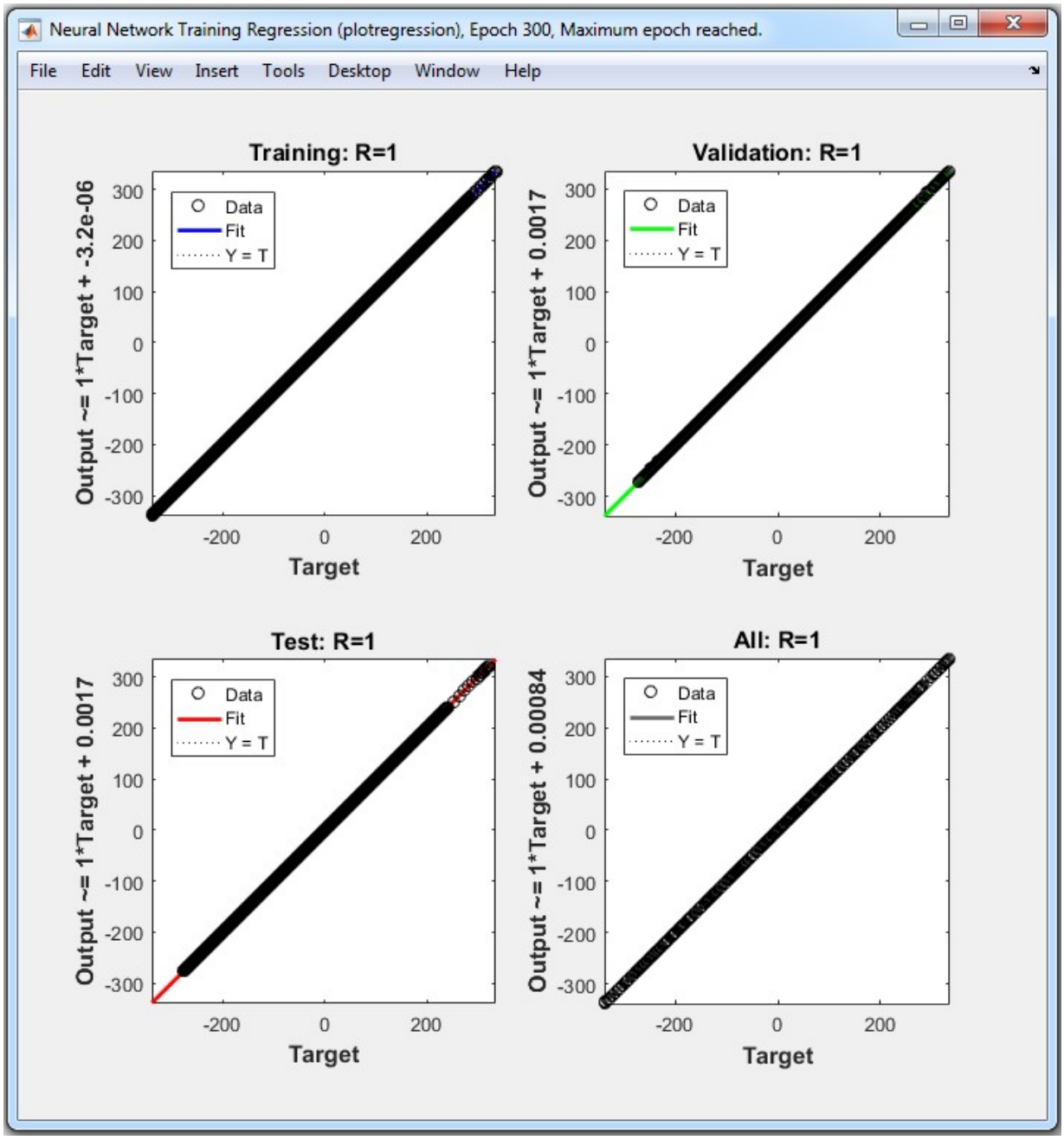


Рис. 3.20. Вікно Neural Network Training Regression

Опис вікна Neural Neural Network Training Regression

Графіки **Training** показують, наскільки точно модель відповідає навчальним даним, тобто даним, на яких вона була навчена.

Графіки **Validation** демонструють, наскільки точно модель працює на наборі даних для перевірки. Дані для перевірки використовуються для валідації результатів моделі та визначення, наскільки вона загальніє на нових даних.

Графіки **Test** включають тестові дані, які незалежно від використовувалися для оцінки точності моделі на даних, які модель раніше не бачила. Тестові дані надають об'єктивну оцінку роботи моделі на нових даних.

Графіки **All** показують регресійні результати для всіх трьох наборів даних: Training, Validation та Test. Графіки All об'єднують і відображають прогнози моделі

та фактичні значення для всіх трьох наборів даних на одному графіку з метою порівняння. Ці графіки забезпечують користувача загальним зоровим порівнянням точності моделі на різних наборах даних і допомагають визначити, чи існують різниці в її поведінці на тренувальних, валідаційних та тестових даних.

Графіки **Data (дані)** для Training, Validation, Test та All даних у вікні Neural Network Training Regression представляють собою графічну інформацію про те, як навчальна модель нейронної мережі відповідає даним на різних наборах даних. Ці графіки показують на горизонтальній вісі фактичні значення (дані з набору даних), а на вертикальній вісі - відповідні прогнозовані значення, отримані від моделі нейронної мережі, тобто кожній парі **з** тренувальних, валідаційних та тестових даних відповідає одна точка (у вікні Neural Network Training Regression позначено кружечками) з координатою по осі абсцис, яка дорівнює бажаному значенню, а по осі одинат – значеенню на виході штучної нейронної мережі. Якщо всі точки лежать близько до ідеальної (діагональної) лінії (це графіки $Y = T$, позначені точками у вікні Neural Network Training Regression), то це означає, що модель добре передбачає дані. Якщо є велика розсіюваність точок, це може свідчити про недосконаліть моделі або шум у даних.

Коефіцієнт кореляції R у вікні **Neural Network Training Regression (plotregression)** є мірою кореляції між фактичними та прогнозованими значеннями у завданні регресії. Цей коефіцієнт допомагає визначити, наскільки добре модель регресії відповідає фактичним даним і наскільки точним є її прогноз.

Кореляційний коефіцієнт R близький до 1 вказує на високу точність моделі регресії, тоді як значення близьке до 0 або від'ємне вказує на низьку точність. Цей коефіцієнт дуже корисний для візуальної оцінки точності моделі та порівняння прогнозованих та фактичних значень у завданнях регресії.

Продовження опису вікна Neural Network Training (nntraintool)

Plot Interval (інтервал графіка). Цей параметр контролює частоту оновлення графіків та відображення результатів під час навчання нейронної мережі для задачі регресії. Plot Interval визначає, через скільки ітерацій навчання (або епох) має бути відображена інформація на графіках.

Кнопки **Stop Training** і **Cancel** у вікні Neural Network Training (nntraintool) в середовищі MATLAB використовуються для керування процесом навчання нейронної мережі, але вони мають різні функції:

Кнопка **Stop Training (зупинка навчання)** призначена для негайної зупинки процесу навчання нейронної мережі. Вона припиняє навчання без можливості продовження або збереження поточного стану моделі. Використовується, коли ви бачите, що навчання триває дуже довго, ви визнали, що навчання не дає задовільних результатів, або ви просто хочете негайно припинити процес навчання з будь-якої причини.

Кнопка **Cancel (скасувати)** використовується для припинення процесу навчання, але зазвичай вона надає можливість підтвердження та збереження поточного стану моделі. Вона дозволяє вам контролювати ітерації навчання, перевіряти ре-

зультати, змінювати параметри та вирішувати, коли припинити навчання з можливістю зберегення результатів навчання.

Після завершення навчання результати відображаються на графіках у вікні **Training data for NN Predictive Control**, яке показано на рис.3.21.

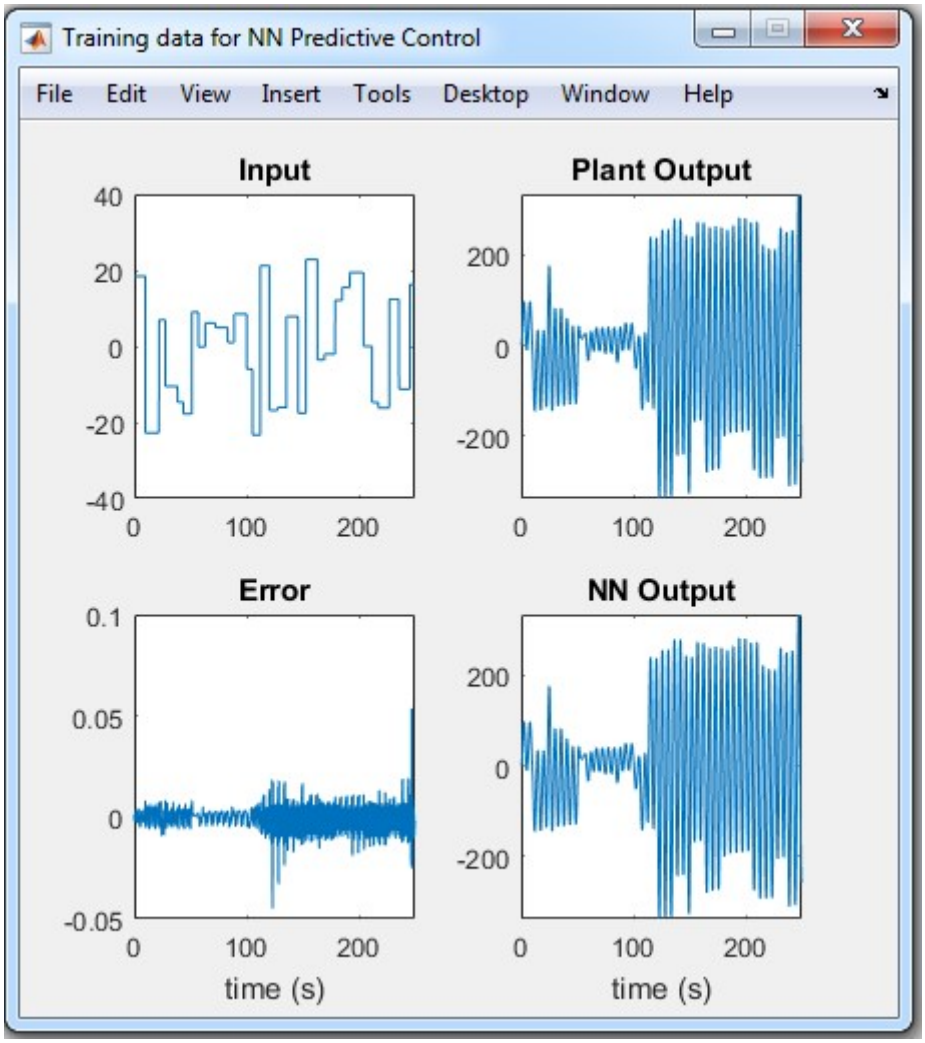


Рис.3.21. Результати тренування мережі

Опис вікна Training data for NN Predictive Control

Input (вхід). Графік Input відображає вхідні сигнали, які подаються на систему і нейромережеву модель під час навчання для генерації відповіді.

Plant Output (вихід системи). Графік Plant output відображає фактичні вихідні дані системи (Plant) під час навчання. Ці дані представляють реальну динаміку системи під впливом вхідних сигналів.

NN Output (вихід нейромережі). Графік NN Output відображає вихідні дані, що згенеровані нейромережевою моделлю під час навчання. Ці дані представляють прогнозовану відповідь моделі, яка намагається відтворити динаміку системи.

Error (Помилка). Графік Error показує різницю між прогнозованими відповідями нейромережевої моделі (NN Output) і фактичними відповідями системи (Plant

Output). Ця помилка вказує на те, наскільки добре модель відтворює динаміку системи під час навчання.

На рис.3.22 і рис.3.23 показані вікна **Validation data for NN Predictive Control** і **Testing data for NN Predictive Control**, в яких наведені аналогічні графіки, побудовані в результаті перевірки на контрольній і текстовій множині.

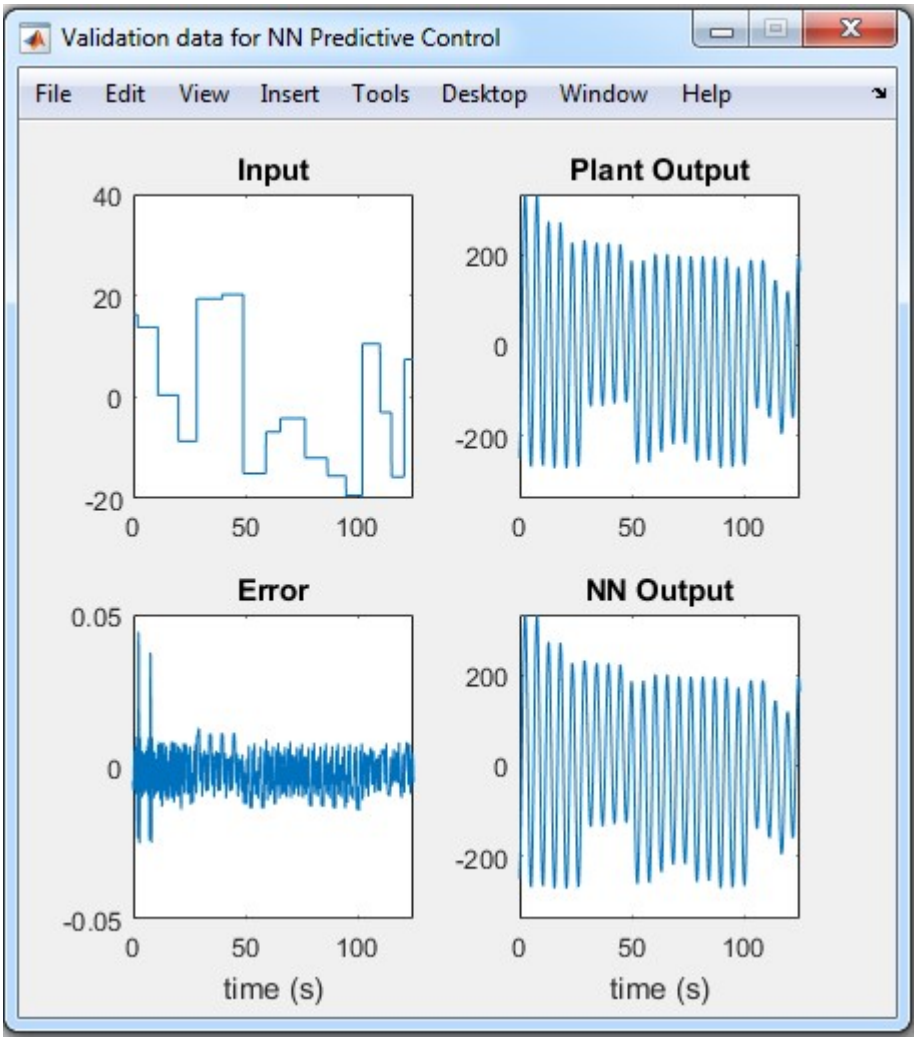


Рис.3.22. Результати перевірки мережі на контрольній множині

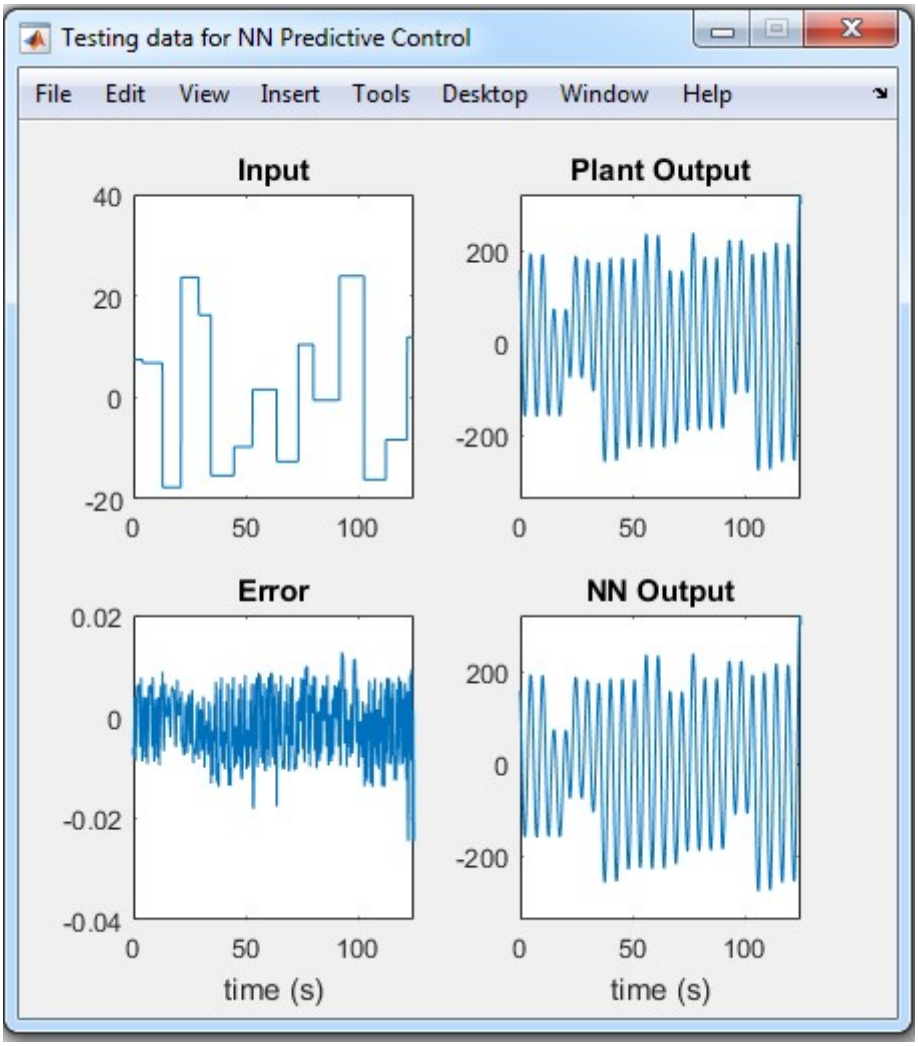


Рис.3.23. Результати перевірки мережі на тестовій множині

Продовження опису вікна Plant Identification.

Поточний стан відмічений у вікні Plant Identification повідомленням «*Training complete. Tou can generate or import new data, continue training or save results by selecting OK or Apply» (Навчання завершене. Можна згенерувати або імпортувати нові дані, продовжити навчання або зберегти отримані результати, вибравши кнопки OK або Apply*).

Коли навчання завершено, М-функція **Nnident** формує динамічну мережу **netn2** із визначеною кількістю затримок на вході та виході моделі., не змінюючи при цьому набутих значень вагів і зсувів нейронів шарів. Згорнута схема динамічної мережі і моделі елементів мережі показані на рис.3.24.

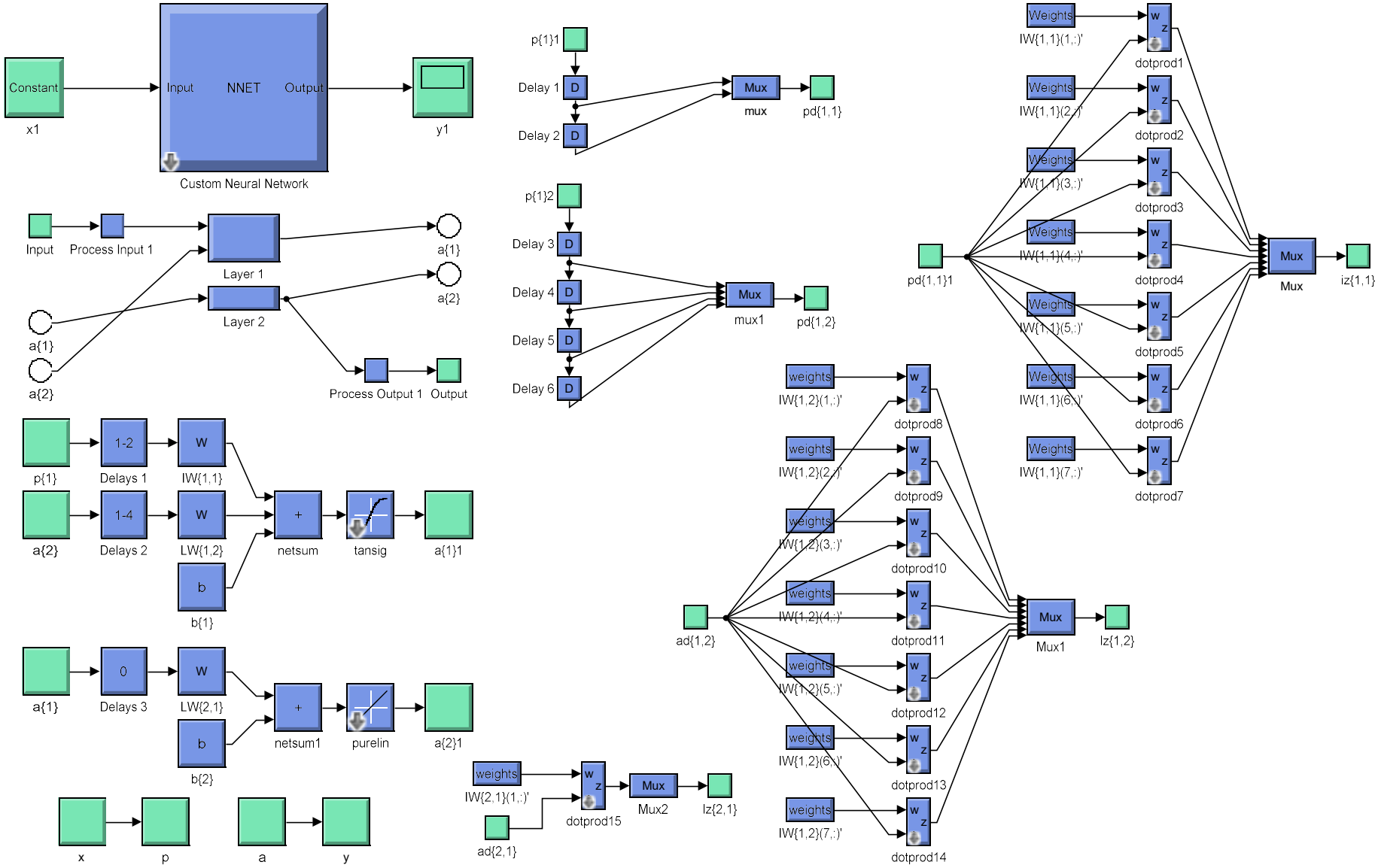


Рис.3.24. Моделі елементів динамічної мережі з елементами затримок, реалізовані в Simulink

Модель динамічної нейронної побудована в системі Simulink за допомогою оператора **gensim(netn2).** Кожен подальший елемент з'являється в окремому вікні при активізації попереднього подвійним клацанням миші. З даних елементів в системі Simulink побудована схема динамічної мережі, показана на рис.3.25.

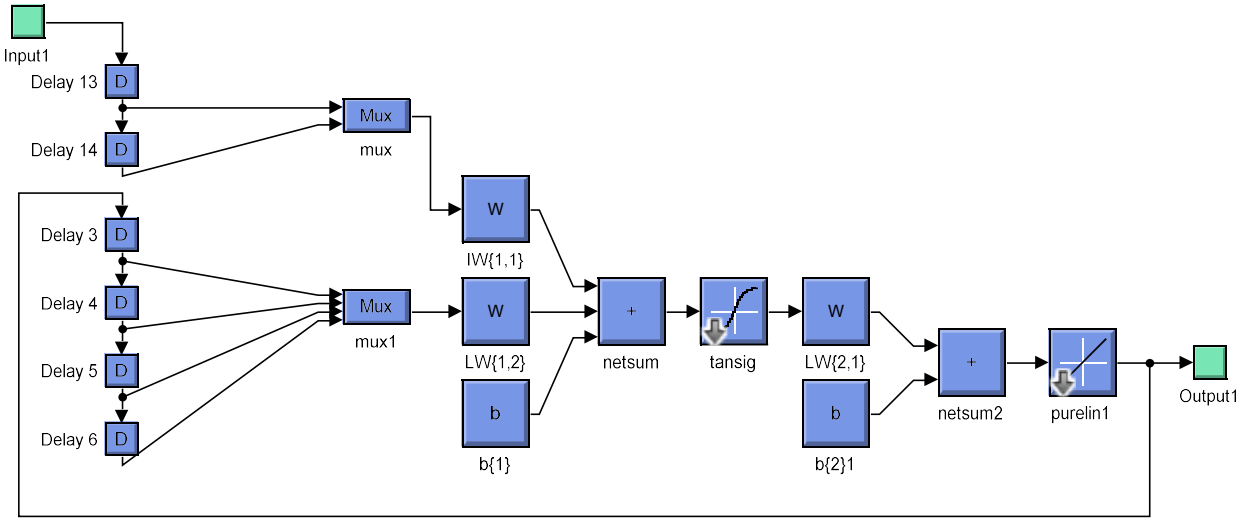


Рис.3.25. Модель динамічної мережі з елементами затримок, побудованої в Simulink

Слід зазначити, що для побудови моделей мереж **netn** і **netn2** треба перервати виконання програми шляхом попереднього встановлення **Breakpoint** у відповідному місці функції Nnident.m, оскільки після закінчення виконання зазначеної функції побудова мереж стає неможливою.

В результаті параметри нейромережевої моделі керованого об'єкту вводяться в блок **NN Predictive Controller** системи Simulink. У системі Simulink дана мережа представляється у вигляді структурної схеми, показаної на рис.3.26.

Блоки Matrix Gain і Matrix Gain 1 відповідають матрицям IW{1,1} і LW{1,2} відповідно. Блоки Constant (B1) Constant1 (B2) відносяться до зсувів нейронів першого і другого шарів. Елементи затримок моделюються за допомогою блоків Discrete State Space 1 і Discrete State Space 2

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{u}(n)\ ;$$

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n).$$

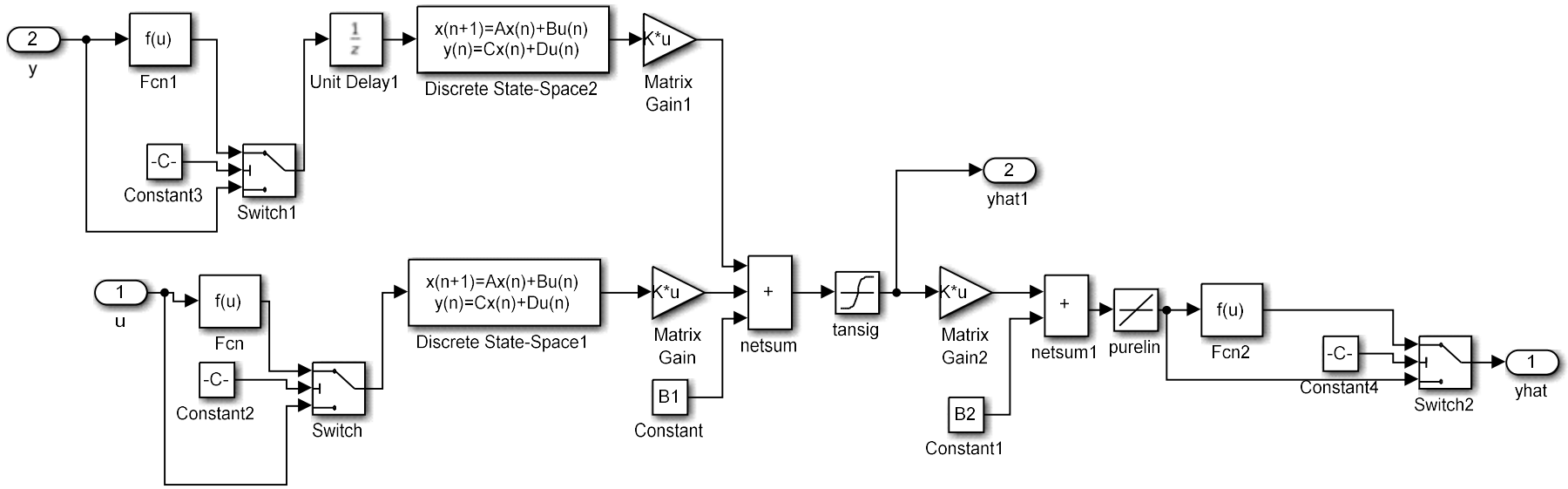


Рис. 3.26. Структурна схема нейромережевої моделі об'єкту регулювання

Для випадку $N_i = 2$ і $N_j = 5$ чисельні значення матриць **A**, **B**, **C** і **D** вказаних блоків наступні.

Discrete State Space 1

$$\mathbf{A} = \begin{vmatrix} 0 \end{vmatrix}, \quad \mathbf{B} = \begin{vmatrix} 1 \end{vmatrix}, \quad \mathbf{C} = \begin{vmatrix} 0 \\ 1 \end{vmatrix}, \quad \mathbf{D} = \begin{vmatrix} 1 \\ 0 \end{vmatrix}.$$

Discrete State Space 2

$$\mathbf{A} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}, \quad \mathbf{B} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix}, \quad \mathbf{C} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}, \quad \mathbf{D} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}.$$

У системі Simulink формується також схема **ptest3sim2**, показана на рис.3.27. Після активізації блоку Subsystem відкривається вікно з схемою рис.3.28. Дана схема теж є нейромережевою моделлю об'єкту управління, що має додаткові виходи, і використовується М-функцією predopt для прогнозу процесу в майбутньому.

Перетворена структурна схема нейрорегулятора, отримана в результаті об'єднання схем рис.3.9 і рис.3.26 показана на рис. 3.29.

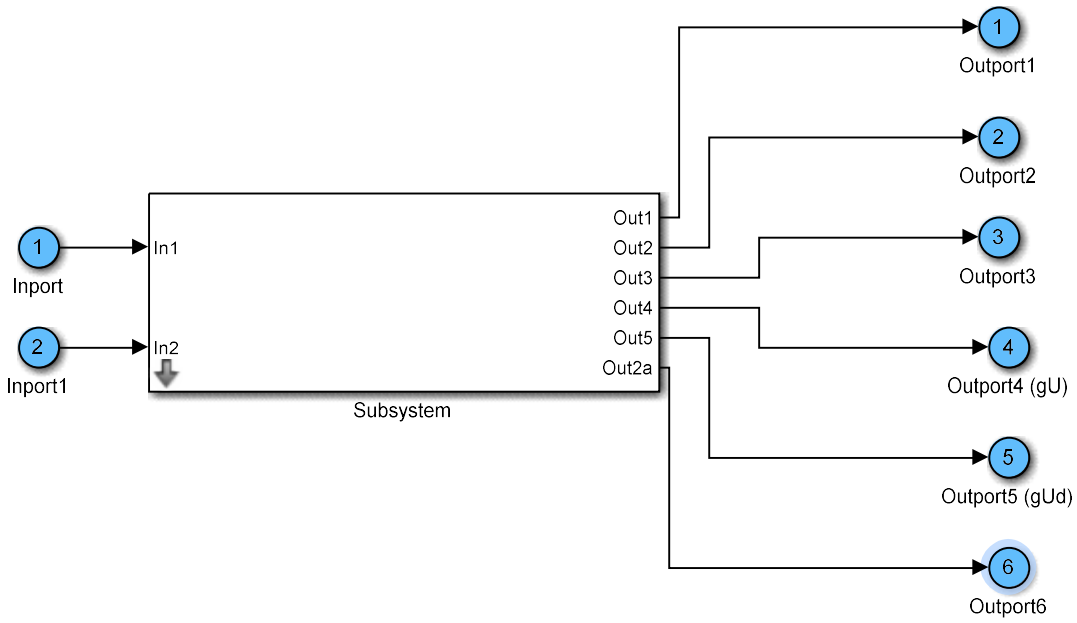


Рис. 3.27.Схема ptest3sim2, використовувана М-функцією predopt для прогнозу процесу в майбутньому

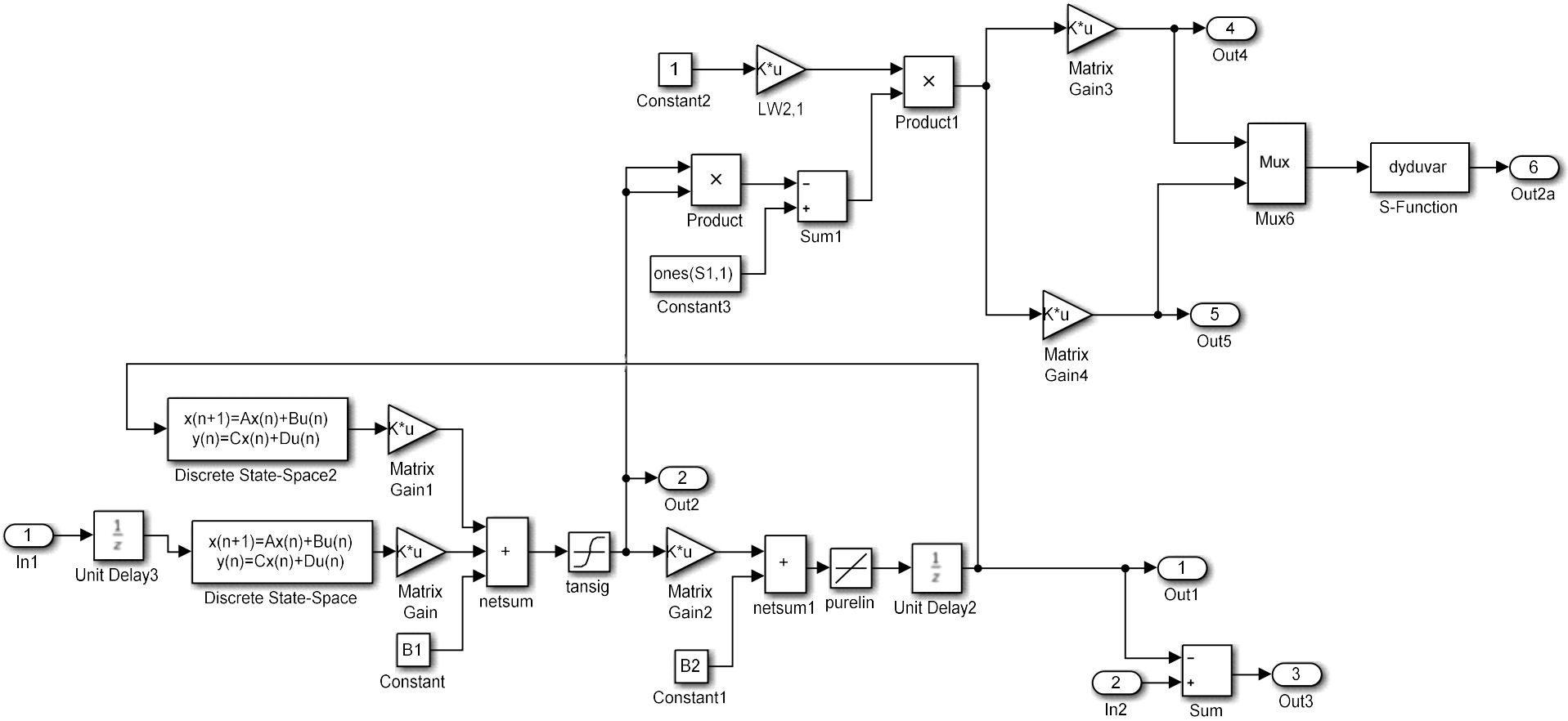


Рис. 28. Схема блоку Subsystem системи ptest3sim2

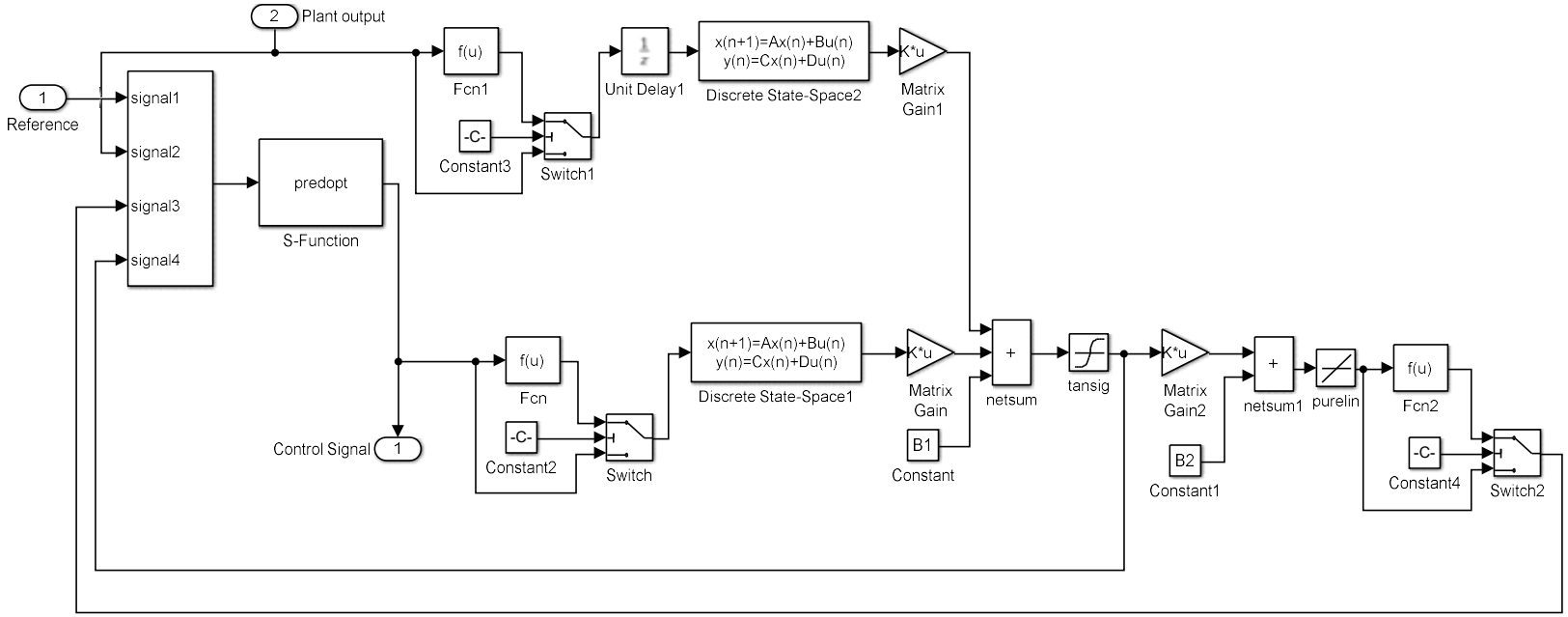


Рис.3.29. Перетворена структурна схема нейрорегулятора

Після створення нейромережевої моделі керованого об'єкта здійснюється повернення до вікна **Neural Network Predictive Controller** (рис. 3.10), де задаються параметри оптимізації.

Продовження опису вікна Neural Network Predictive Controller

Const Horizon (N2). Верхня межа підсумовування в показнику якості (нижня межа N_1 фіксована і рівна 1). Цей параметр визначає горизонт прогнозу, тобто кількість кроків у майбутньому, на які алгоритм прогнозуваного керування буде робити прогнози.

Control Horizon (Nu). Верхня межа підсумовування при оцінці потужності управління N_u. Цей параметр грає важливу роль при плануванні оптимального керування системою, і він визначає кількість кроків у майбутньому, на які буде розрахована потужність управління в рамках контролю.

Control Weighting Factor (ρ). Коефіцієнт ваги для потужності управління. Цей параметр дозволяє вам налаштувати важливість мінімізації керуючих сигналів в обчисленнях контролера.

Search parameter (α). Параметр одновимірного пошуку, задаючий поріг зменшення показника якості. Цей параметр використовується для контролю процесу оптимізації і для визначення, коли оптимізація може бути завершена, оскільки зменшення показника якості досягло заданого порогу.

Minimization Routine. Вибір процедури одновимірного пошуку.

Цей параметр визначає, який алгоритм оптимізації буде використовуватися для налаштування параметрів контрольної стратегії. Вибір підходящого алгоритму оптимізації може суттєво впливати на якість та ефективність контрольної системи.

Iterations Per Sample Time. Параметр изначає кількість ітерацій оптимізації, які виконуються протягом кожного відрізку часу (Sample Time) при розрахунку оптимального керування системою. Цей параметр дозволяє встановити, скільки ітерацій оптимізації виконується в межах кожного відрізку часу, і впливає на швидкість та точність оптимізації контролера. Зазвичай, більша кількість ітерацій дозволяє отримати більш точну оптимальну стратегію керування, але водночас може збільшити обчислювальну складність та вимоги до ресурсів.

Після установки параметрів оптимізації вони вводяться в блок NN Predictive controller системи Simulink.

3.6 Вибір параметрів нейрорегулятора NN Predictive Controller

При синтезі регулятора варіюються значення параметрів N_2, N_u і ρ (при цьому N_1 фіксоване та дорівнює 1), а також параметра одновимірного пошуку α, який визначає поріг зменшення показника якості та кількість ітерацій на один такт дискретності γ. Додатково задається процедура одновимірного пошуку.

Дослідження показали, що параметри N_u, ρ і α мають незначний вплив на результати синтезу, тому були обрані значення: N_u=2, $\rho = 0{,}05$, $\alpha = 0{,}001$. Як процедуру одновимірного пошуку використано **csrchbac**.

Параметри N_2 і γ значно впливають на роботу регулятора. Збільшення цих значень підвищує точність, проте істотно зростає обсяг обчислень на кожному такті дискретності. Оптимальні значення для розв'язуваної задачі знаходяться в межах $N_2 = 15 \div 25$, $\gamma = 2 \div 3$.

При ідентифікації об'єкта керування ключовим питанням є вибір кількості нейронів у прихованому шарі N. Недостатня кількість нейронів унеможливлює виконання поставлених завдань, а надмірна кількість призводить до перенавчання та збільшення обсягу обчислень. Для даної задачі оптимальне значення $N = 7 \div 12$, при цьому помилки навчання, а також контрольної та тестової вибірках ε знаходиться в межах $10^{-3} \div 10^{-5}$.

Успіх навчання мережі суттєво залежить від довжини навчальної вибірки $N_{\text{в}}$ та такту дискретності Δt, який визначає інтервал між двома послідовними моментами збору даних. Оптимальні значення: $N_{\text{в}} = 10000$, $\Delta t = 0{,}05 \div 0{,}1\ c$. Збільшення Δt знижує точність обчислень і зменшує різницю між помилкою на навчальній множині та помилками на контрольній і тестовій множинах. Зменшення Δt вимагає збільшення $N_{\text{в}}$, що призводить до значного зростання часу навчання без суттєвого зменшення помилки.

Для отримання репрезентативної вибірки важливо правильно встановити мінімальне та максимальне значення інтервалу ідентифікації. Ці параметри залежать від характеристик об'єкта **Subsystem**. У даному прикладі обрані межі: $t_{\min} = 4 \div 5\ c$, $t_{\max} = 10 \div 20\ c$.

Під час синтезу нейромережевої моделі системи задається кількість елементів затримки на вході z_1 та виході z_2 моделі. Найкращі результати досягнуті при $z_1 = 2$, $z_2 = 2 \div 5$.

Результат навчання мережі також залежить від початкових значень ваг w_{ij} та кількості навчальних циклів $N_{\text{ц}}$. Для досягнення глобального мінімуму процес навчання слід повторювати багаторазово з різними початковими значеннями w_{ij} та різними значеннями $N_{\text{ц}}$. У даному прикладі для кожної конфігурації мережі було об-

рано кілька сотень початкових точок, а кількість навчальних циклів, після яких помилка переставала зменшуватись, становила $300 \div 600$.

Як навчальну функцію використано **trainlm**.

3.7 Розрахунок динамічних характеристик систем з нейрорегулятором NN Predictive Controller

Для побудови графіків перехідних процесів двомасової системи з нейрорегулятором NN Predictive Controller використовуємо моделі Simulink, приведені на рис. 3.8 або 3.30.

На рис. 3.31 і рис. 3.32 приведені графіки перехідних процесів основних змінних стану синтезованої системи. З аналізу отриманих характеристик виходить, що реакція системи на ступінчасті дії з випадковою амплітудою цілком задовільна, має коливальний характер з невеликим перерегулюванням

Отже, синтезований нейромережевий регулятор з прогнозом NN Predictive Controller може бути використаний для управління двомасовою електромеханічною системою, що розглядається у наведеному прикладі

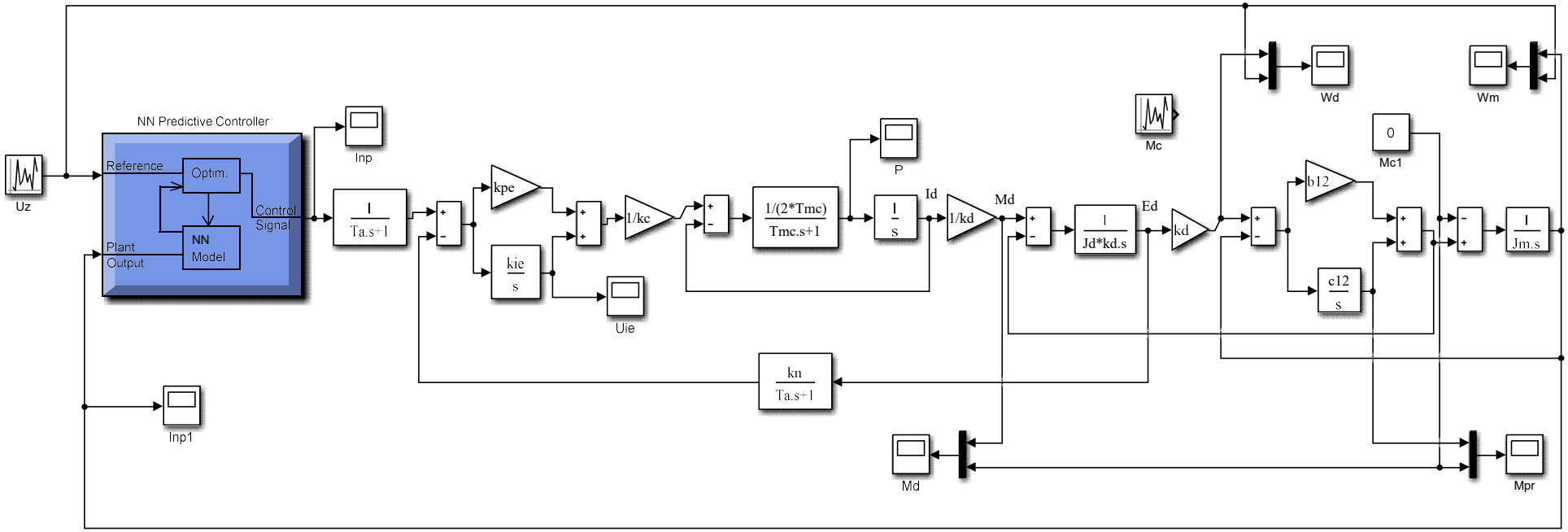
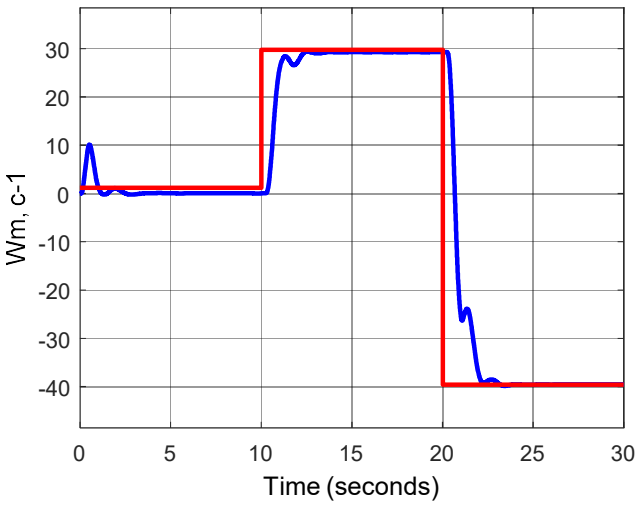
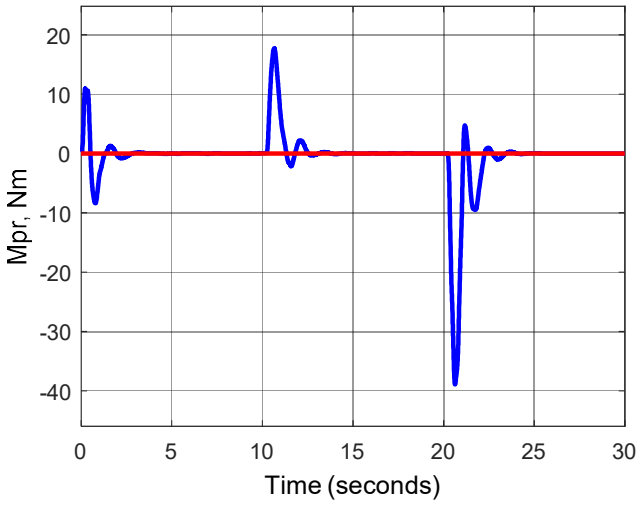


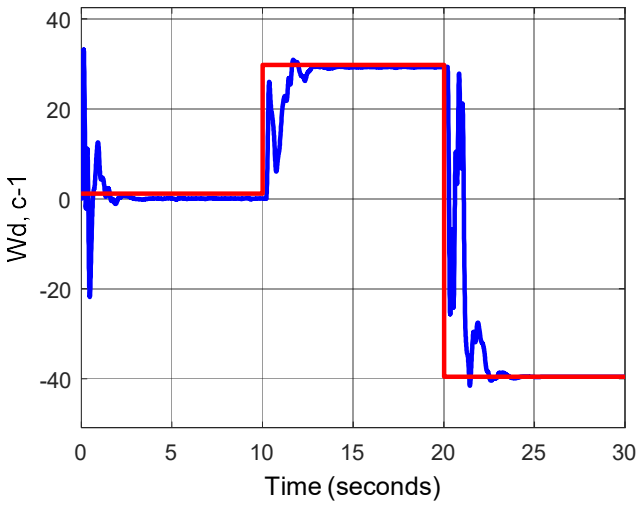
Рис. 3.30. Схема моделі двомасової системи з нейрорегулятором



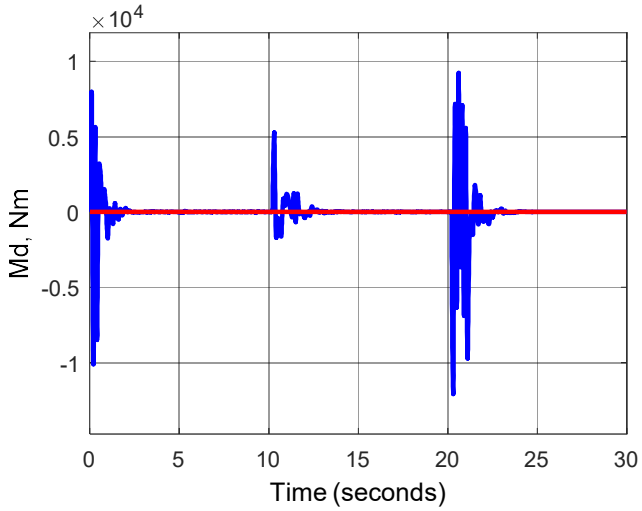
швидкість механізму по задаючій дії



момент пружності по задаючій дії

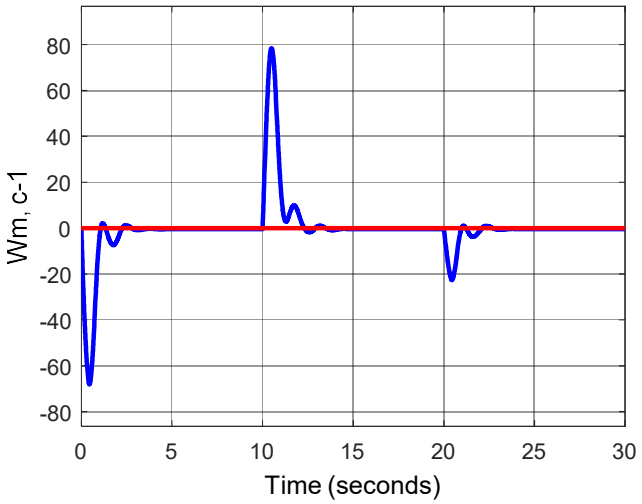


швидкість двигуна по задаючій дії

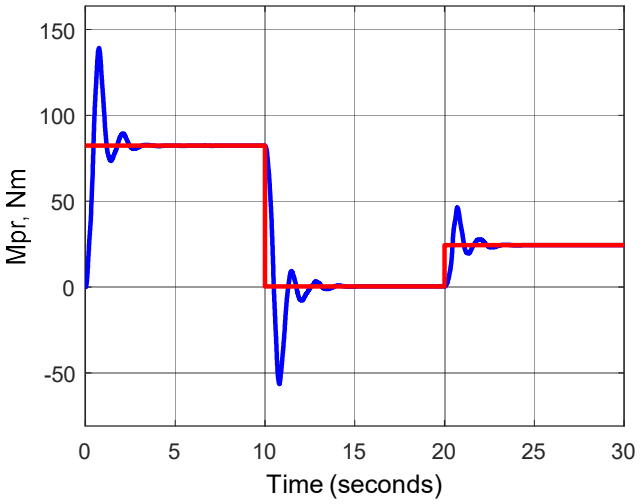


момент двигуна по задаючій дії

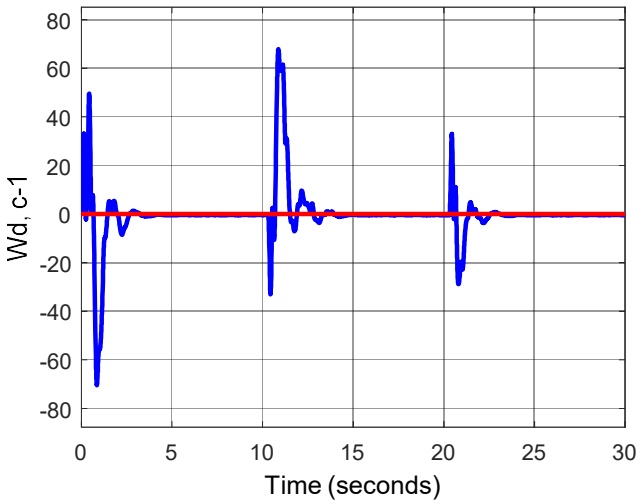
Рис.3.31. Графіки перехідних процесів двомасової системи з нейрорегулятором NN Predictive Controller по задаючій дії



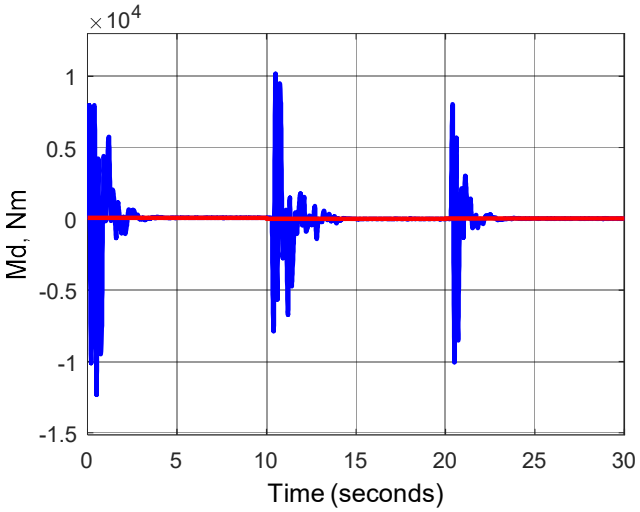
швидкість механізму по збурюючій дії



момент пружності по збурюючій дії



швидкість двигуна по збурюючій дії



момент двигуна по збурюючій дії

Рис.3.32. Графіки перехідних процесів двомасової системи системи з нейрорегулятором NN Predictive Controller по збурюючій дії

3.8 Завдання для самостійного виконання

Варіанти завдань для синтезу нейрорегулятора **NN Predictive Controller** для двомасових і трьохмасових електромеханічних систем наведено в додатку А.

3.9 Контрольні питання по темі заняття

1. Перечисліть регулятори реалізовані в пакеті прикладних програм Neural Network Toolbox системи MATLAB.
2. Поясніть принцип дії регулятора з прогнозом **NN Predictive Controller**.
3. Поясніть структуру нейронних мереж, які використовуються в регуляторі.
4. Наведіть структурну схему системи управління з нейрорегулятором **NN Predictive Controller**.
5. Перечисліть основні етапи синтезу нейромережевої системи управління, побудованої на основі регулятора. NN Predictive Controller.
6. Як виконується формування навчальної послідовності?
7. Як відбувається контроль процесу навчання мережі?
8. Як вибираються параметри нейрорегулятора?
9. Які функції виконує нейронна мережа в регуляторі?

Практичне заняття 4

СИНТЕЗ НЕЙРОРЕГУЛЯТОРА НА ОСНОВІ МОДЕЛІ АВТОРЕГРЕСІЇ З КОВЗАЮЧИМ СЕРЕДНІМ NARMA-L2 CONTROLLER З ВИКОРИСТАННЯМ СИСТЕМИ MATLAB

4.1 Мета заняття

Ознайомлення з порядком синтезу нейрорегулятора нейрорегулятора на основі моделі авторегресії з ковзаючим середнім **NARMA-L2 Controller** у середовищі MATLAB. Вивчення особливості методики керування на основі моделі авторегресії з ковзаючим середнім. Набуття навичок побудови нейромережевої моделі об'єкту керування, здійснення навчання нейронної мережі та реалізації керування на її основі. Заняття спрямоване на набуття практичних навичок у використанні **NARMA-L2 Controller** для підвищення ефективності управління скуладними динамічними системами.

4.2 Принцип побудови нейрорегулятора на основі моделі авторегресії з ковзаючим середнім NARMA-L2 Controller

Нейромережевий регулятор NARMA-L2 використовує як модель керованого об'єкту модель нелінійної авторегресії з ковзаючим середнім (Nonlinear Autoregressive-Moving Average – NARMA-L2). При синтезі даного регулятора будується дискретна нелінійна модель нелінійного об'єкту управління як авторегресійна модель з ковзаючим середнім, або NARMA-модель у формі

$$y(k+d) = N[y(k), y(k-1),..., y(k-n+1),\ u(k),\ u(k-1),...,u(k-m+1)]\ , \qquad (4.1)$$

де $y(k)$ – вихід моделі; d – число тактів прогнозу; $u(k)$ – вхід моделі.

На етапі ідентифікації будується нейронна мережа для NARMA-моделі вигляду (4.1).

Якщо потрібно спроектувати стежачу систему, яка забезпечує рух по заданій траєкторії

$$y(k+d) = y_r(k+d)\,,$$

то це означає, що необхідно сформувати регулятор наступного виду:

$$u(k) = G[y(k), y(k-1),..., y(k-n+1), y_r(k+d),\ u(k-1),...,u(k-m+1)]\,.$$

Хоча такий регулятор за допомогою нейронної мережі і може бути сформований, проте в процесі мінімізації середньоквадратичної помилки він вимагає надмірних обчислень, оскільки використовує динамічний варіант методу зворотного роз-

повсюдження помилки. Для практичного вирішення завдання стеження Нарендра (Narendra) і Макхопадхаї (Mukhopadhyay) запропонували наближені NARMA-модель з виділеною складовою управління. Така модель регулятора, що іменується моделлю NARMA-L2, має вигляд:

$$y(k+d) = f[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)] +$$
$$+g[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)]u(k).$$

Перевага цієї форми полягає в тому, що тепер поточне управління можна безпосередньо обчислити, якщо відома бажана траєкторія y_r, передісторія управління $\{u(k-1),...,u(k-m+1)\}$, а також передуючі і поточне значення виходу $\{y(k), y(k-1),..., y(k-n+1)$:

$$u(k) = \frac{y_r(k+d) - f[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)]}{g[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)]}. \quad (4.2)$$

Безпосереднє застосування цього співвідношення для реалізації регулятора утруднене, оскільки управління залежить від поточного значення виходу, тому рівняння (4.2) модифікується таким чином:

$$u(k+1) = \frac{y_r(k+d) - f[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)]}{g[y(k), y(k-1),..., y(k-n+1),\ u(k-1),...,u(k-m+1)]}, \quad (4.3)$$

але при цьому параметр прогнозу повинен задовольняти умові $d \geq 2$.

На рис.4.1 показана структура відповідного регулятора у вигляді нейронної мережі. Тут слід звернути увагу на ділянки мережі, які виконують апроксимацію нелінійних операторів g і f у вигляді виходів $\hat{g} = a^2(t)$ і $\hat{f} = a^4(t)$. Входами регулятора є сигнали $y(t+1)$ і $u(t+1)$ (останній реалізований у вигляді зворотного зв'язку), а також еталонний сигнал $y_r(t+2)$. Блоки затримки здійснюють запам'ятовування відповідних послідовностей входу і виходу, а потім використовуються двошарові нейронні мережі, які формують оцінки нелінійних операторів і обчислюють сигнал управління у формі (4.3).

Загальна структурна схема системи з регулятором NARMA-L2 показана на рис.4.2. На схемі явним чином виділена еталонна модель, яка задає бажану траєкторію для виходу керованого об'єкту.

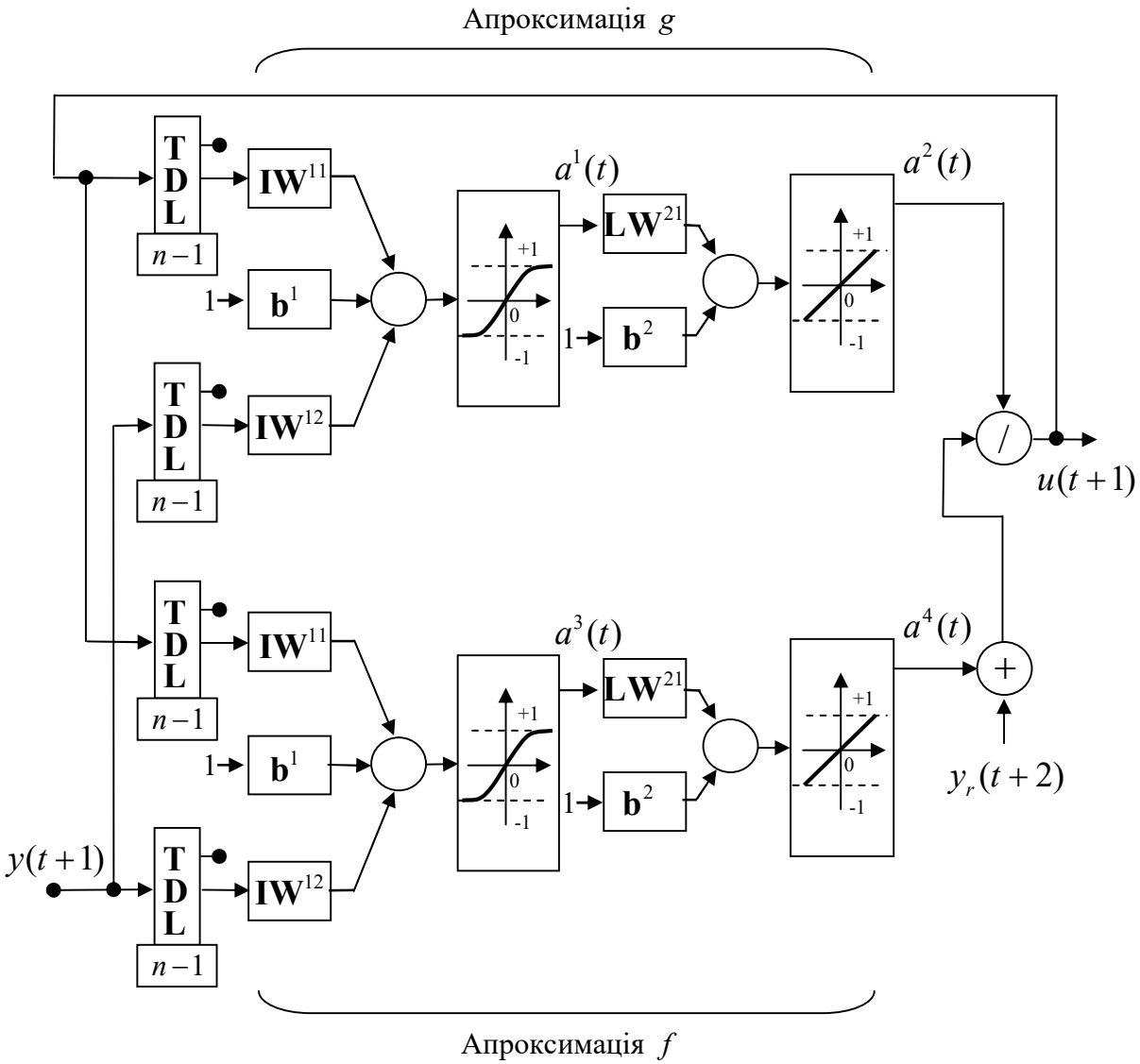


Рис.4.1. Структура NARMA-L2 регулятора у вигляді нейронної мережі

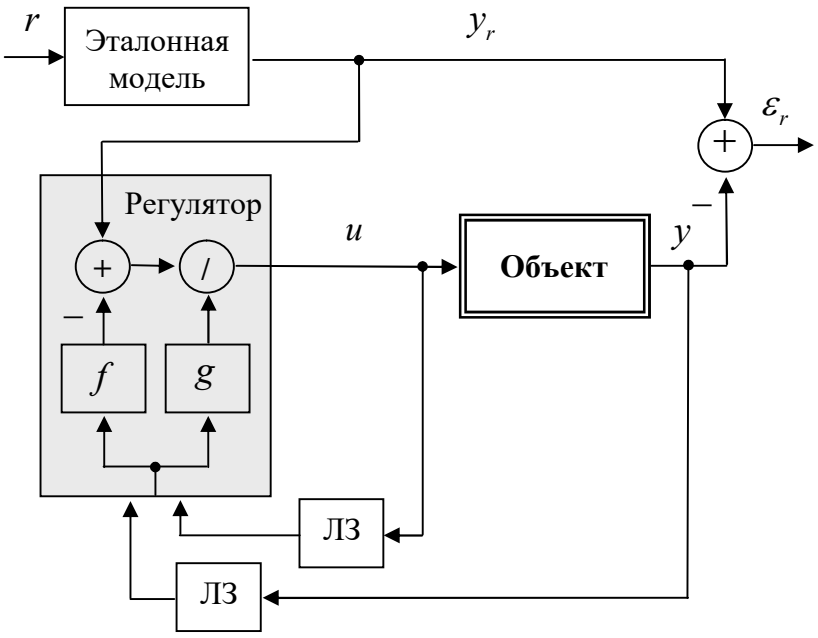


Рис.4.2. Структурна схема системи з регулятором NARMA-L2

4.3 Методика синтезу нейрорегулятора NARMA-L2 Controller

При синтезі нейрорегулятора **NARMA-L2 Controller** використовуються наступні файли, розміщені в каталозі toolbox/nnet/nncontrol системи SIMULINK.

nncontrolutil – підтримка, що забезпечує можливість звернення до приватних функцій з системи SIMULINK;

sfunxy2 – функція для виведення графіків;

nnident.m – функція, використовувана при ідентифікації об'єкту управління (ця функція використовується при побудові нейромережевої моделі об'єкту управління при синтезі всіх регуляторів, реалізованих в ППП Neural Network Toolbox системи MATLAB).

У вікні системи SIMULINK формується схема системи із структурою, показаною на рис.4.3. Ця структура аналогічна схемі рис. 3.8, наведеній у практичному занятті 3, проте замість блоку нейрорегулятора **NN Prediction Controller** використовується блок **NARMA-L2 Controller**. Схема блоку Subsystem відповідає рис. 3.12 практичного занятті 3.

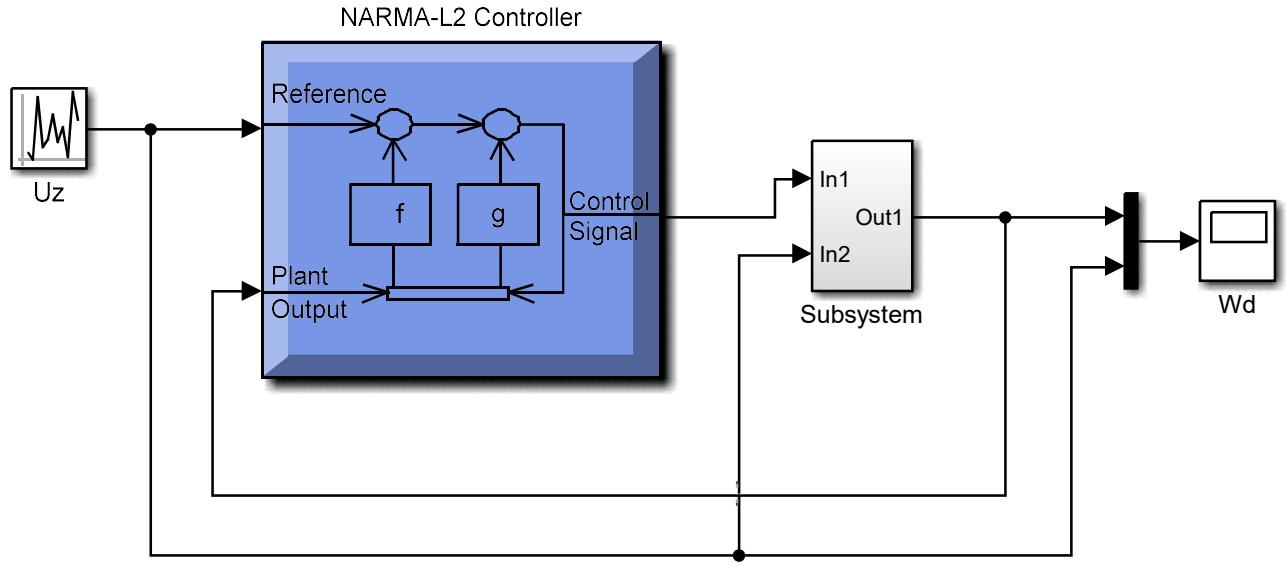


Рис.4.3. Схема системи управління з нейрорегулятором NARMA-L2 Controller

Процес синтезу нейрорегулятора починається шляхом активізації блоку **NARMA-L2 Controller**. З'являється вікно **Plant Identification NARMA-L2**, показане на рис.4.4.

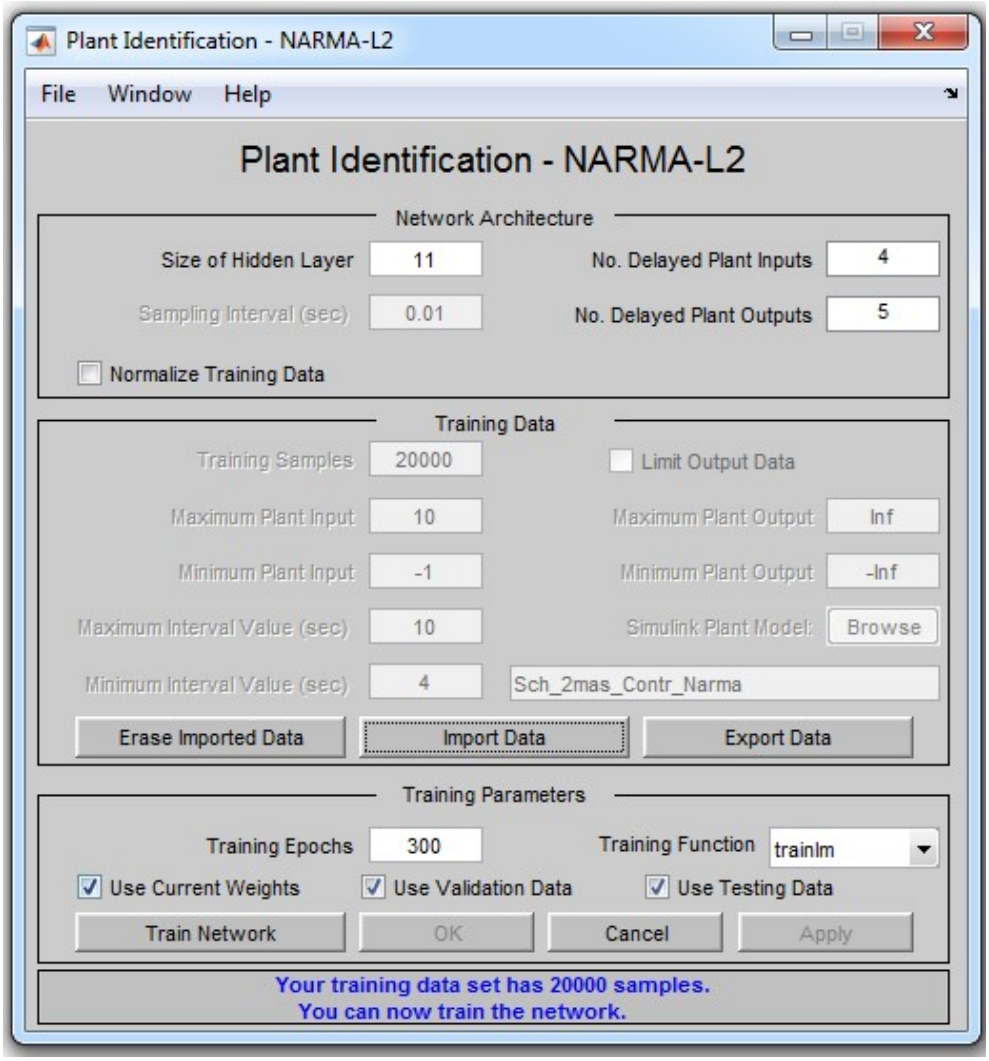


Рис.4.4. Вікно ідентифікації керованого об'єкту Plant Identification NARMA-L2

Як указувалося вище при описі порядку синтезу нейрорегулятора NN Predictive Controller, це вікно універсальне і може бути використано для побудови нейромережевих моделей для будь-якого динамічного об'єкту, який описаний моделлю SIMULINK. Для побудови вікна і проведення процедури ідентифікації в системі MATLAB використовується функція **nnident.m.**

Параметри, які задаються при проведенні процедури ідентифікації, схема моделі об’єкту управління, порядок генерації навчальної послідовності аналогічні тім, які описані в практичному занятті 3 при синтезі нейрорегулятора **NN Prediction Controller**.

Вибір процедури **Generate Training Data** призведе до запуску програми для створення навчальної послідовності. Графіки входового та вихідного сигналів об'єкту управління відображаються на екрані (рис.4.5). Після завершення процесу створення навчальної послідовності користувач має можливість прийняти (**Accept Data**) або відхилити (**Reject Data**) згенеровані дані.

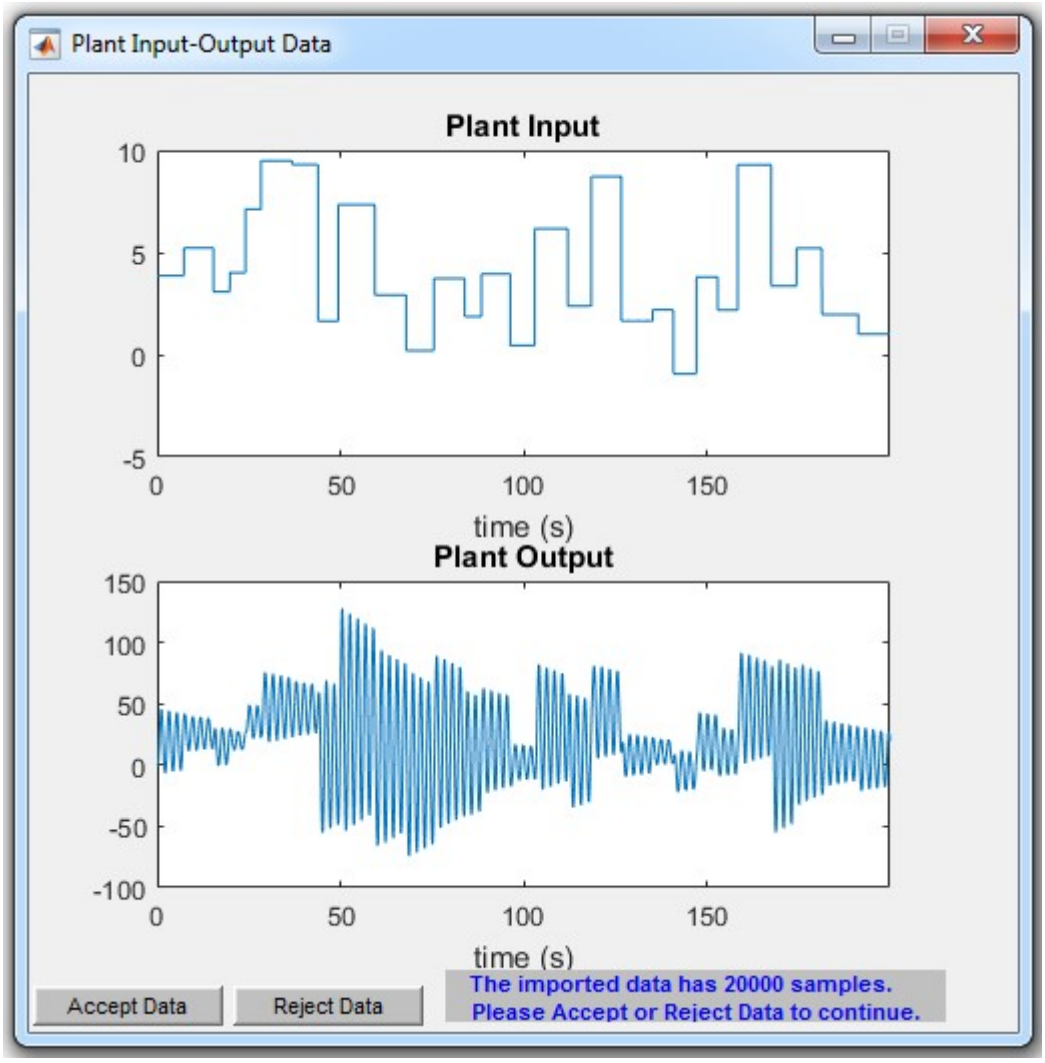


Рис. 4.5. Графіки вхідного і вихідного сигналів при
генерації навчальної послідовності

Якщо дані прийняті, програма відкриє модифіковане вікно **Plant Identification**. У цьому вікні деякі елементи стають недоступними, а кнопка **Generate Training Data** замінюється кнопкою **Erase Generate Data**, яка дає можливість видалити згенеровані дані.

Після натиснення на кнопку **Train Network.** відбувається створення і ініціалізація мережі **netn** з прямою передачею сигналу за допомогою М-функції **newff**. При цьому структура мережі істотно відрізняється від структури мережі, яка використовується в нейрорегуляторі NN Predictive Controller.

На рис.4.6 показані моделі елементів нейронної мережі, побудовані за допомогою оператора **gensim(netn).**

Елементи нейронної мережі, відповідають наступним параметрам, заданим у вікні ідентифікації: розмір прихованого шару $S = 11$, кількість елементів запізнювання на вході моделі $N_i = 4$ і кількість елементів запізнювання на виході моделі $N_j = 5$. Кожен подальший елемент з'являється в окремому вікні при активізації попереднього подвійним клацанням миші. З даних елементів в системі Simulink побудована схема мережі, показана на рис.4.7.

Дана мережа не має елементів затримки, тобто є статичною. Мережа використовує 1 вектор входу з 8 елементами. На перші N_j входів подаються сигнали $y(k), y(k-1),..., y(k-N_j+1)$ (у даному випадку $y(k)$, $y(k-1)$), $y(k-2)$, $y(k-3)$, $y(k-4)$), на наступні (N_i-1) входів подаються сигнали $u(k-1),...,u(k-N_i+1)$ (у даному випадку $u(k-1)$, $u(k-2)$, $u(k-3)$). Мережа має 6 шарів з 11 нейронами в першому і третьому шарах і 1 нейроном в другому, четвертому, п'ятому і шостому шарах. Використовувані функції активації: гіперболічного тангенса (**tansig**) – в першому і третьому шарі, лінійна (**purelin**) – в другому, четвертому, п'ятому і шостому шарах.

Після створення мережі **netn** відбувається її перетворення за допомогою наступних операторів:

```
netn.numInputs=2;
netn.numInputs=3;
netn.inputs{2}.size=netn.inputs{1}.size;
netn.inputs{2}.range=netn.inputs{1}.range;
netn.inputs{3}.range=minmax(ptr{3,1});
netn.biasConnect(5:6)=0;
netn.layers{5}.netInputFcn='netprod';
netn.inputConnect(3,2)=1;
netn.inputConnect(5,3)=1;
netn.layerConnect(6,2)=1;
netn.layerConnect(3,2)=0.
```

В результаті формується мережа, елементи якої показані на рис.4.8 Схема перетвореної мережі, побудована з використанням елементів, приведених на рис.4.8, показана на рис.4.9.

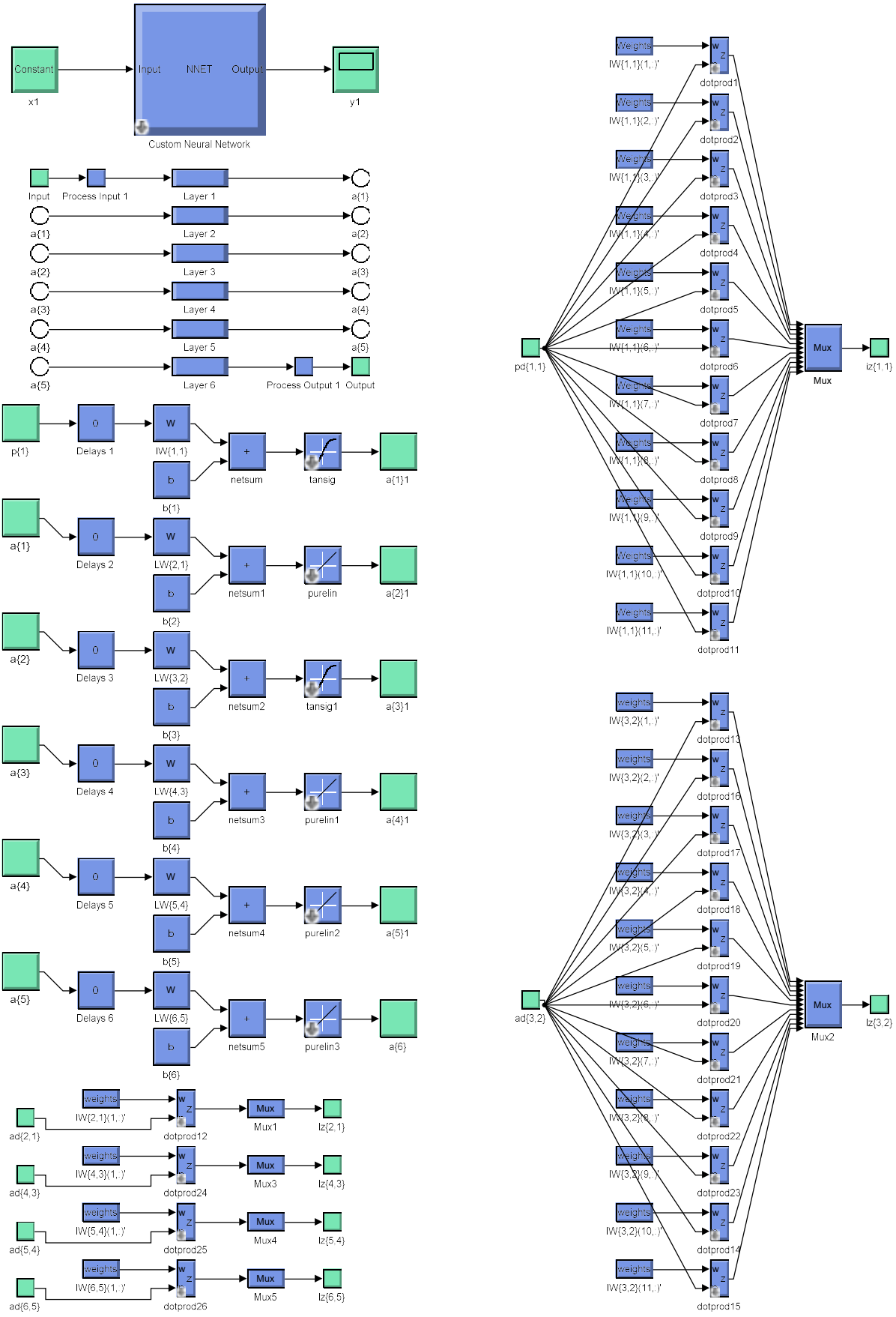


Рис.4.6. Моделі елементів мережі **netn** з прямою передачею сигналу нейрорегулятора **NARMA-L2 Controller**

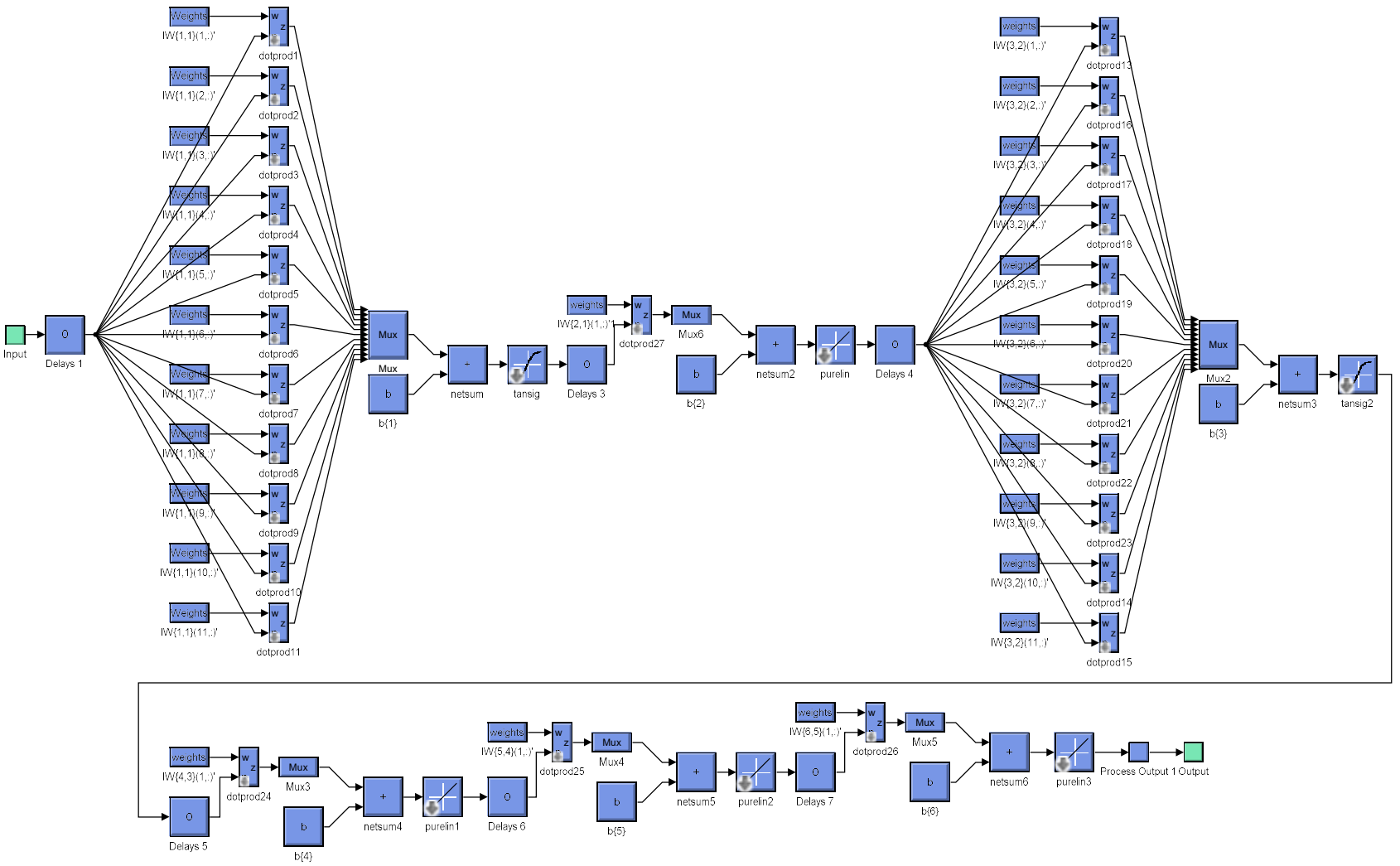


Рис.4.7. Модель статичної мережі **netn** з прямою передачею сигналу нейрорегулятора **NARMA-L2 Controller**

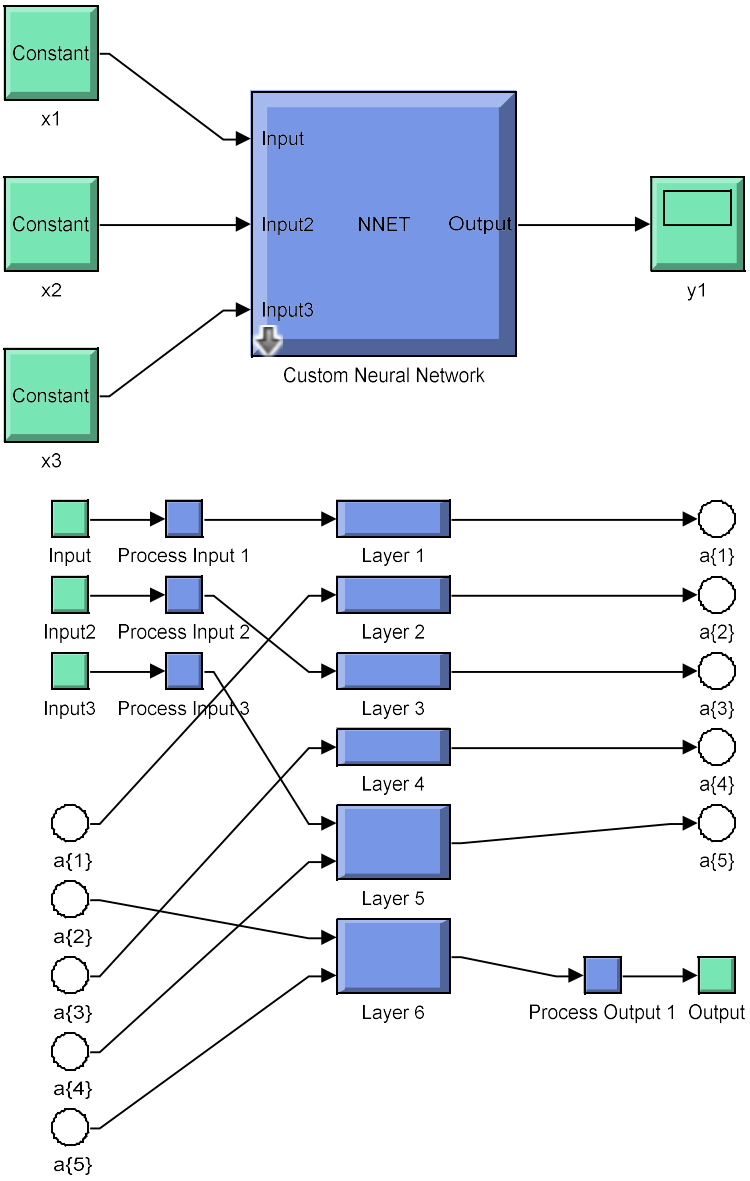


Рис.4.8. Моделі елементів перетвореної мережі **netn** регулятора

Мережа використовує 3 вектори входу з 8 елементами в першому і другому векторах і 1 елементом в третьому векторі. Другий вектор входу формується так само, як і перший, описаний вище. На третій вхід подаються сигнали $u(k)$. Замість лінійної функції активації (**purelin**) в п'ятому шарі встановлюється функція активації **netprod**, що виконує функцію поелементного добутку зважених входів.

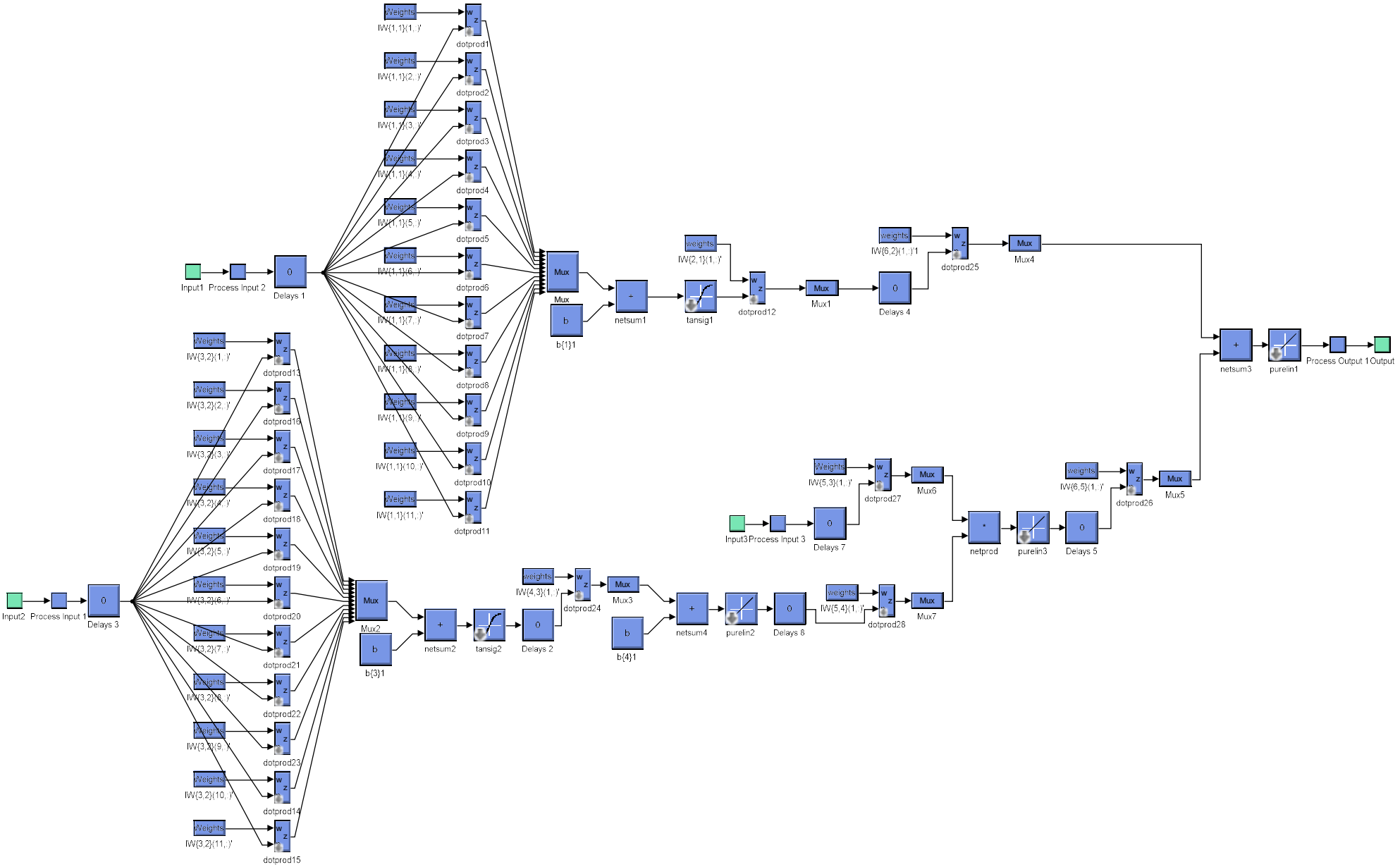


Рис.4.9. Модель перетвореної мережі **netn** регулятора NARMA-L2 Controller з трьома векторами входу

Після створення мережі виконується процес її навчанні. Вектори входу представляються як числові масиви вибірок у форматі **double**, що відповідає груповому представленню даних. Навчання здійснюється з використанням функції **trainlm**, відповідної алгоритму Левенберга-Марквардта. Процес та результати навчання наочно демонструються за допомогою діалогового вікна **Neural Network Training (nntraintool)**, показаного на рис.4.10.

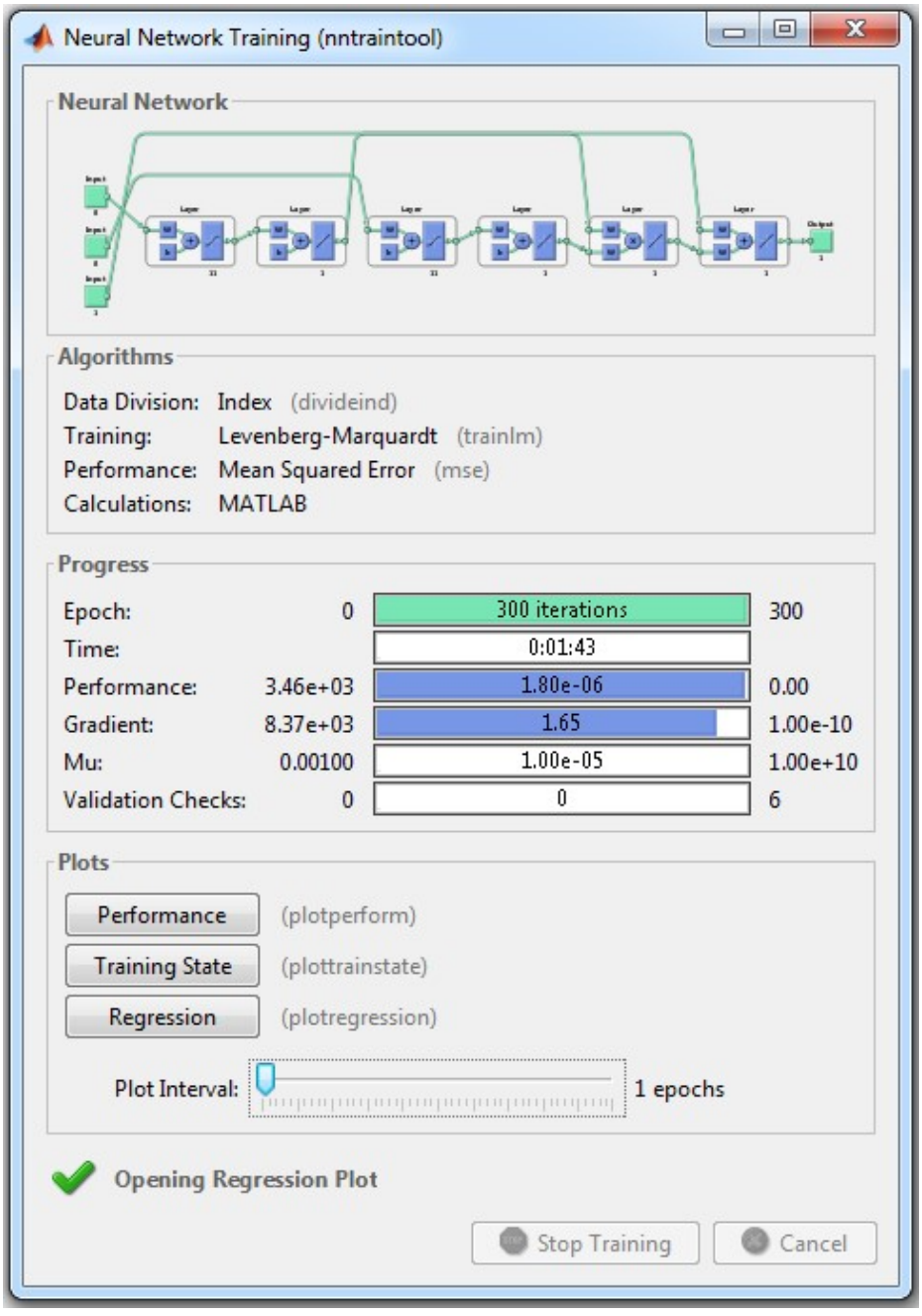


Рис. 4.10. Ввікно Neural Network Training , що відображає хід навчання та результати навчання нейронної мережі

Динаміка зміни помилки навчання, а також перевірки на контрольній і текстовій множинах відбивається у вікні,зображеному на рис.4.11.

Результати навчання, а також результати тестування на контрольній і тестовій множині відображаються на графіках, представлених на рис.4.12 – рис.4.14.

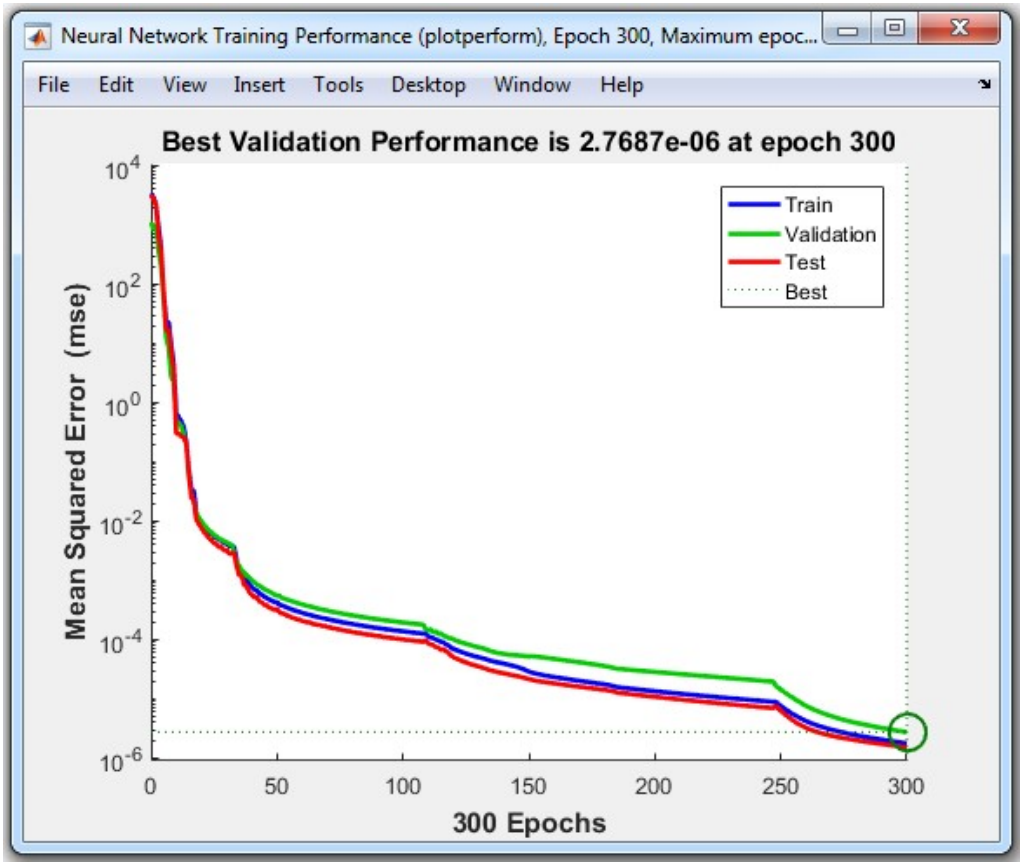


Рис.4.11. Вікно контролю процесу навчання нейронної мережі нейрорегулятора NARMA-L2 Controller

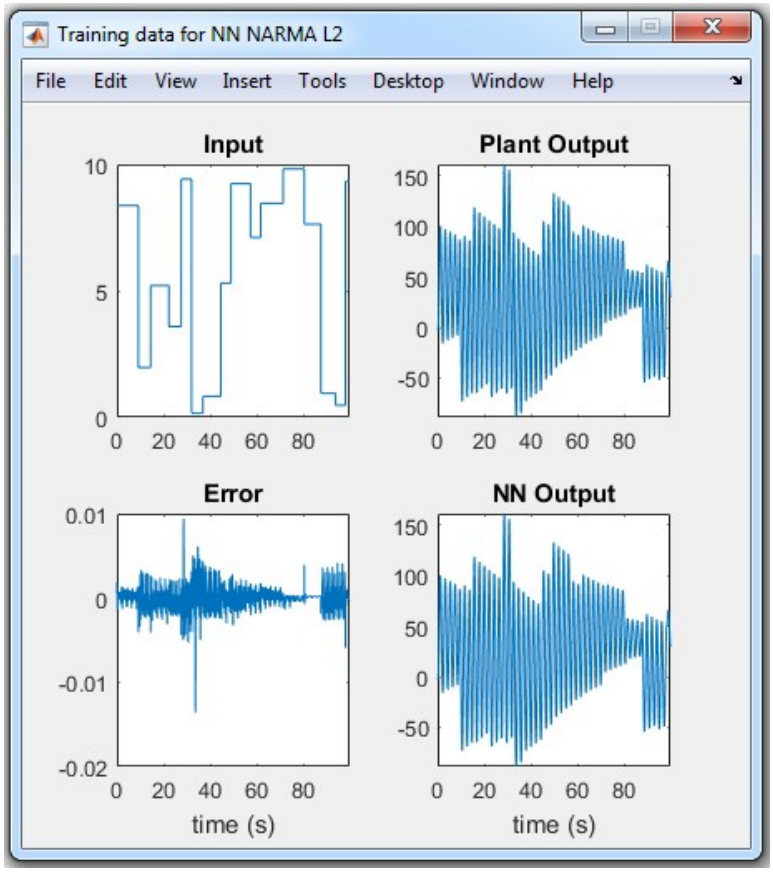


Рис.4.12. Результати тренування мережі нейрорегулятора NARMA-L2 Controller

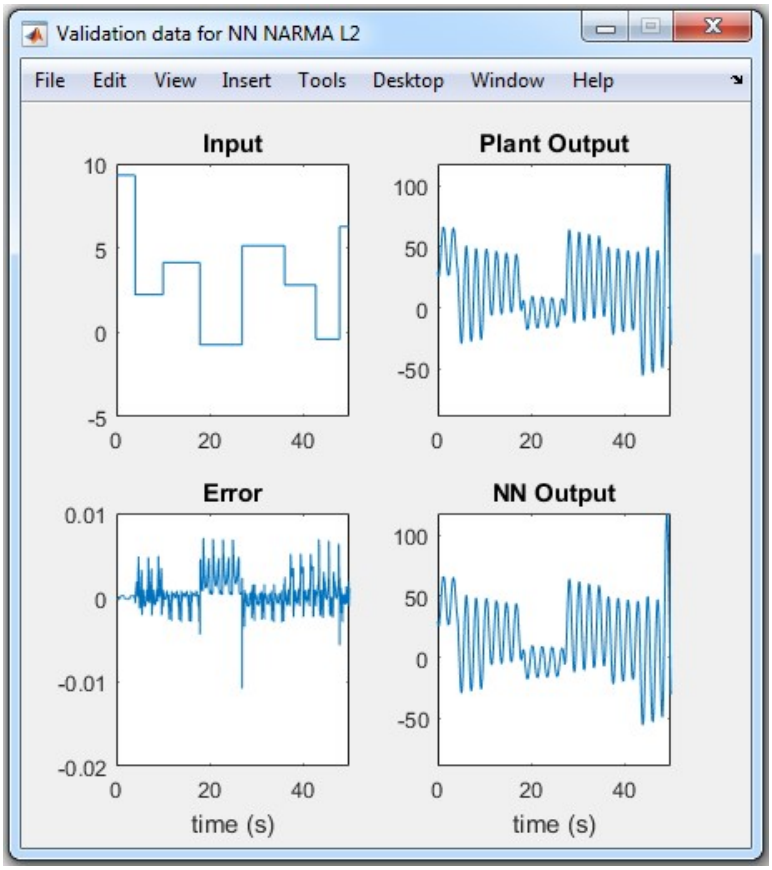


Рис.4.13. Результати тестування мережі нейрорегулятора NARMA-L2 Controller на контрольній множині

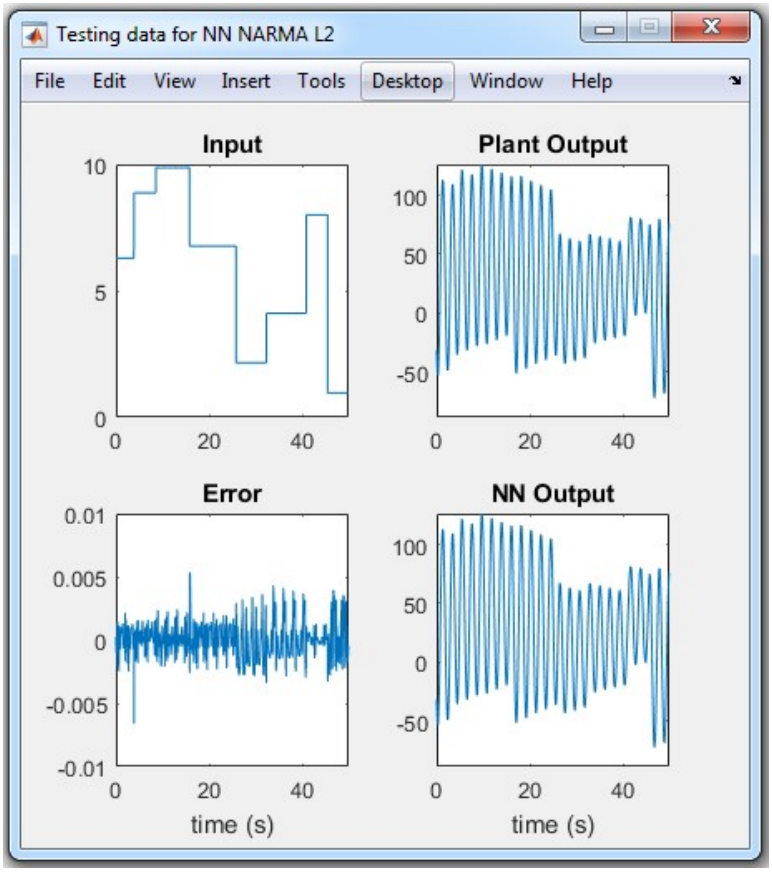


Рис.4.14. Результати тестування мережі нейрорегулятора NARMA-L2 Controller на тестовій множині

Після закінчення процесу навчання числові значення елементів матриць вагів **IW{1,1}, IW{3,2}, IW{5,3}, LW{2,1}, LW{4,3}, LW{5,4}, LW{6,5}, LW{6,2}** і зсувів **b{1}, b{2}, b{3}, b{4}** вводяться в блок NARMA-L2 Controller системи Simulink. У системі Simulink дана мережа представляється у вигляді структурної схеми, показаної на рис.4.15. Параметр **K** блоків **Matrix Gain** відповідає:

Matrix Gain IW1_1=netn.IW{1,1}; Matrix Gain1 IW3_2=netn.IW{3,2};
Matrix Gain2 LW2_1=netn.LW{2,1}; Matrix Gain3 LW2_1=netn.LW{4,3};
Matrix Gain5 LW4_3=netn.LW{6,2};
Matrix Gain8 LW6_5*LW5_4*IW5_3=netn.LW{6,5}*netn.LW{5,4}*netn.IW{5,3}.

Значення зсувів присвоюються параметру **Constant value** блоків **Constant**

B1=netn.b{1}; B2=netn.b{2}; B3=netn.b{3}; B4=netn.b{4};

Елементи затримок моделюються за допомогою блоку **Discrete State Space** аналогічно розглянутому вище для нейрорегулятора NN Predictive Controller.

4.4 Вибір параметрів нейрорегулятора NARMA-L2 Controller

При синтезі регулятора NARMA-L2 Controller, як і регулятора NN Predictive Controller, найбільш важливим питанням є вибір кількості нейронів першого і третього шарів S. При малій кількості нейронів мережа не може виконувати поставлене завдання, а при великій спостерігається явище перенавчання і зростає об'єм обчислень. Для даного нейрорегулятора оптимальні значення $S = 10 \div 14$, при цьому помилка навчання, а також помилка на контрольній і тестовій множині ε не перевищує $10^{-3} \div 10^{-5}$.

Успіх тренування мережі в значній мірі залежить від довжини навчальної вибірки N_B і такту дискретності Δt, що визначає інтервал між двома послідовними моментами знімання даних. Оптимальними у вирішуваній задачі є: $N_B = 20000$, $\Delta t = 0{,}1\ c$. При збільшенні Δt знижується точність обчислюється і різниця між помилкою навчання і помилкою, отриманою на контрольній і тестовій множині. Зменшення Δt викликає необхідність відповідного збільшення N_B і, як наслідок, значно збільшується час тренування мережі, при цьому істотного зниження ε не спостерігається.

Для отримання представницької вибірки необхідно правильно задати максимальне і мінімальне значення інтервалу ідентифікації. Величина їх залежить від параметрів об'єкту управління; у даному завданні прийнято $t_{\min} = 4 \div 5\ c$, $t_{\max} = 10 \div 20\ c$.

Кількість елементів запізнювання на вході N_i і виході N_j моделі варіювалися в межах $N_i = 1 \div 4$, $N_j = 2 \div 5$.

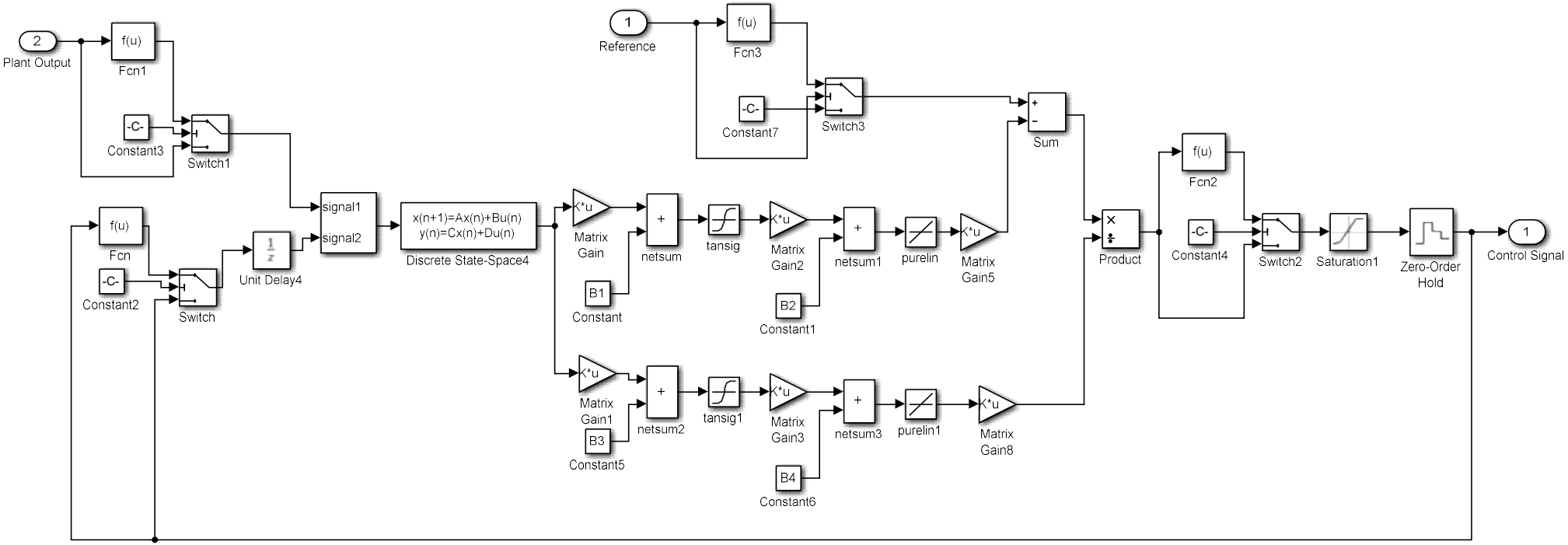


Рис.4.15. Структурна схема нейронної мережі нейрорегулятора NARMA-L2 Controller в системі Simulink

Кількість циклів навчання $N_{\text{ц}}$, після закінчення яких помилка навчання перестала зменшуватися, складало $300 \div 600$.

Як навчальна функція використана функція **trainlm**.

Як оптимальні параметри прийняті наступні: $S = 11$; $N_i = 4$, $N_j = 5$; $N_{\text{ц}} = 300$. Криві, що характеризують процес навчання нейронної мережі нейрорегулятора NARMA-L2 Controller, приведені вище на рис.4.11 – рис.4.14, відповідають вказаним параметрам.

Не дивлячись на те, що отримані помилки мають той же порядок, що і помилки навчання мережі нейрорегулятора NN Predictive Controller, перехідні процеси системи з нейрорегулятором NARMA-L2 Controller мають коливальний характер.

4.5 Завдання для самостійного виконання

Варіанти завдань для синтезу нейрорегулятора **NARMA-L2 Controller** для двомасових і трьохмасових електромеханічних систем наведено в додатку А.

4.6 Контрольні питання по темі заняття

1. .Поясніть принцип дії регулятора на основі моделі авторегресії з ковзаючим середнім **NARMA-L2 Controller**.
2. Поясніть структуру нейронних мереж, які використовуються в регуляторі.
3. Наведіть структурну схеми систему управління з нейрорегулятором **NARMA-L2 Controller**
4. Перечисліть основні етапи синтезу нейромережої системи управління, побудованої на основі регулятор **NARMA-L2 Controller**.
5. Як виконується формування навчальної послідовності?
6. Як відбувається контроль процесу навчання мережі?
7. Як вибираються параметри нейрорегулятора?
8. Які функції виконують нейронні мережі в регуляторі?
9. Скільки шарів містить нейронна мережа нейрорегулятора **NARMA-L2 Controller**?

Практичне заняття 5

СИНТЕЗ НЕЙРОРЕГУЛЯТОРА З ЕТАЛОННОЮ МОДЕЛЛЮ MODEL REFERENCE CONTROLLER З ВИКОРИСТАННЯМ СИСТЕМИ MATLAB

5.1 Мета заняття

Ознайомлення з методикою синтезу нейрорегулятора **Model Reference Controller** у середовищі MATLAB. Засвоєння принципи адаптивного керування на основі еталонної моделі. Вивчення методики створення, налаштовування та навчання нейронної мережі для реалізації регулятора, а також аналізу його ефективність у керуванні динамічними системами. Заняття спрямоване на набуття практичних навичок проектування інтелектуальних систем керування для складних технічних об’єктів

5.2 Принцип побудови нейрорегулятора з еталонною моделлю Model Reference Controller

При реалізації системи управління з еталонною моделлю використовуються 2 нейронні мережі: для регулятора і для моделі об'єкту управління. Мета навчання регулятора полягає в тому, щоб рух об'єкту управління відстежував вихід еталонної моделі.

Структурна схема, що пояснює принцип побудови системи управління з еталонною моделлю, показана на рис.5.1.

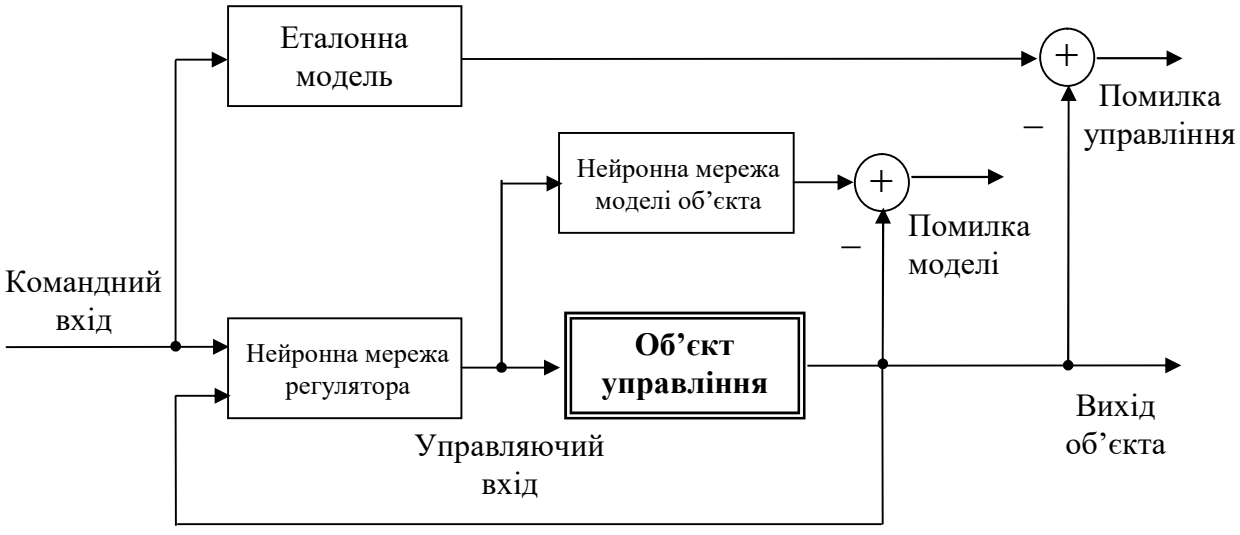


Рис.5.1. Структурна схема системи управління з еталонною моделлю

У ній слід виділити еталонну модель, яка задає бажану траекторію руху об'єкту управління, а також нейронні мережі, які реалізують регулятор і модель об'єкту управління.

Архітектура нейронних мереж регулятора і об'єкту управління надані на рис.5.2. Мережі мають два шари нейронів і містять лінії затримок, що використовуються для формування входів нейронних мереж

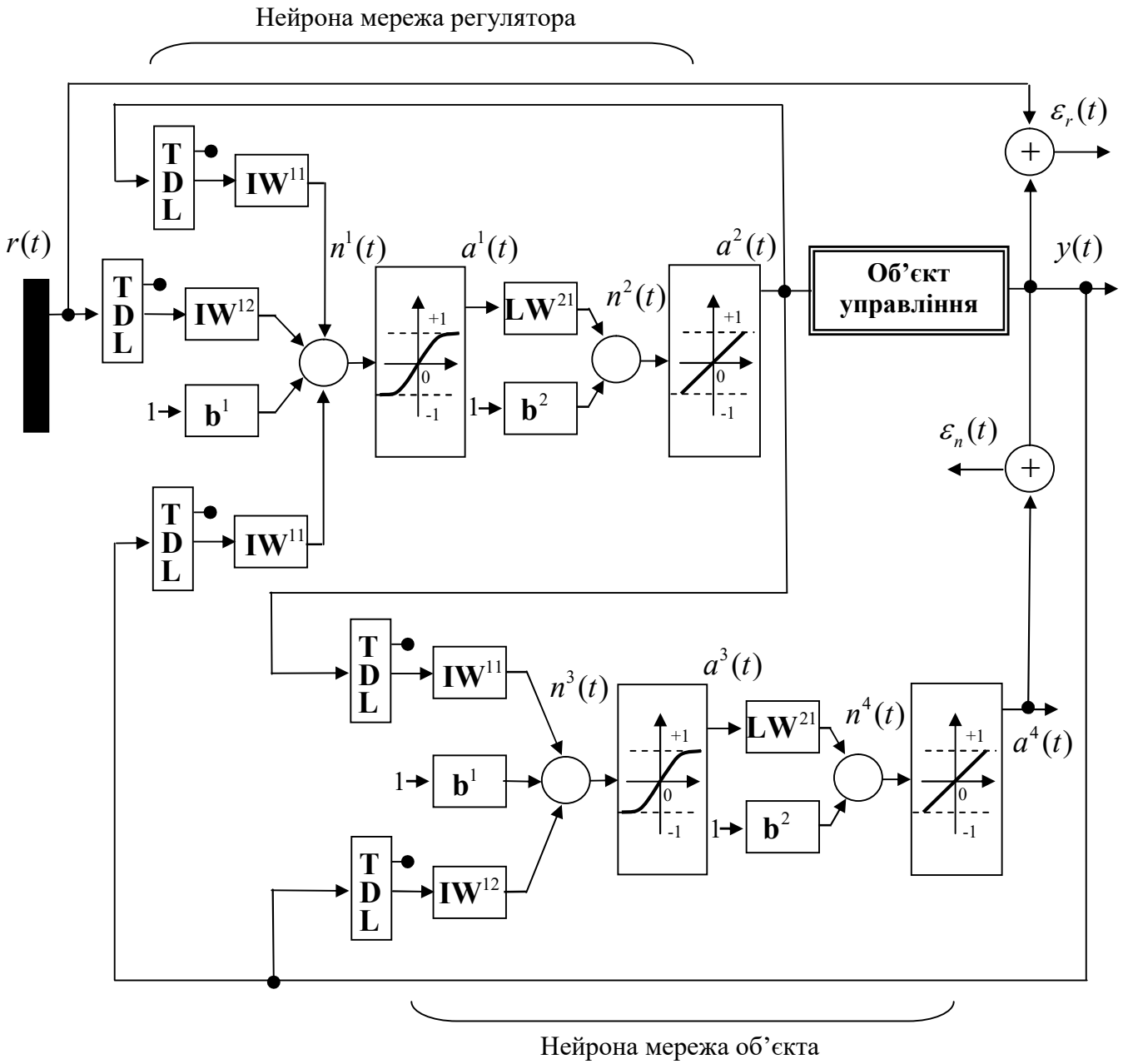


Рис.5.2. Схема включення нейронних мереж регулятора і об'єкту управління в системі управління з еталонною моделлю

5.3 Методика синтезу нейрорегулятора Model Reference Controller

При синтезі нейрорегулятора **Model Reference Controller** використовуються наступні файли, розміщені в каталозі **toolbox/nnet/nncontrol**.

Навчальні функції нейронних мереж:

srchbacxc – процедура одновимірного пошуку на основі перебору з поверненням;

trainbfgc – модифікована процедура алгоритму BPGS для розрахунку системи управління з еталонною моделлю;

nnmodref.m – основна функція, використовувана при синтезі нейрорегулятора Model Reference Controller: забезпечує GUI користувача, генерацію навчальної вибірки, створення і тренування мережі регулятора.

Функції **sfunxy2, nncontrolutil, nnident.m** – описані вище.

У вікні системи SIMULINK формується схема системи із структурою, показаною на рис.5.3. Ця структура аналогічна схемі рис. 3.8, наведеному у практичному занятті 3, проте замість блоку нейрорегулятора **NN Prediction Controller** використовується блок **Model Reference Controller**. Схема блоку Subsystem відповідає рис. 3.12 (див. практичне заняття 3).

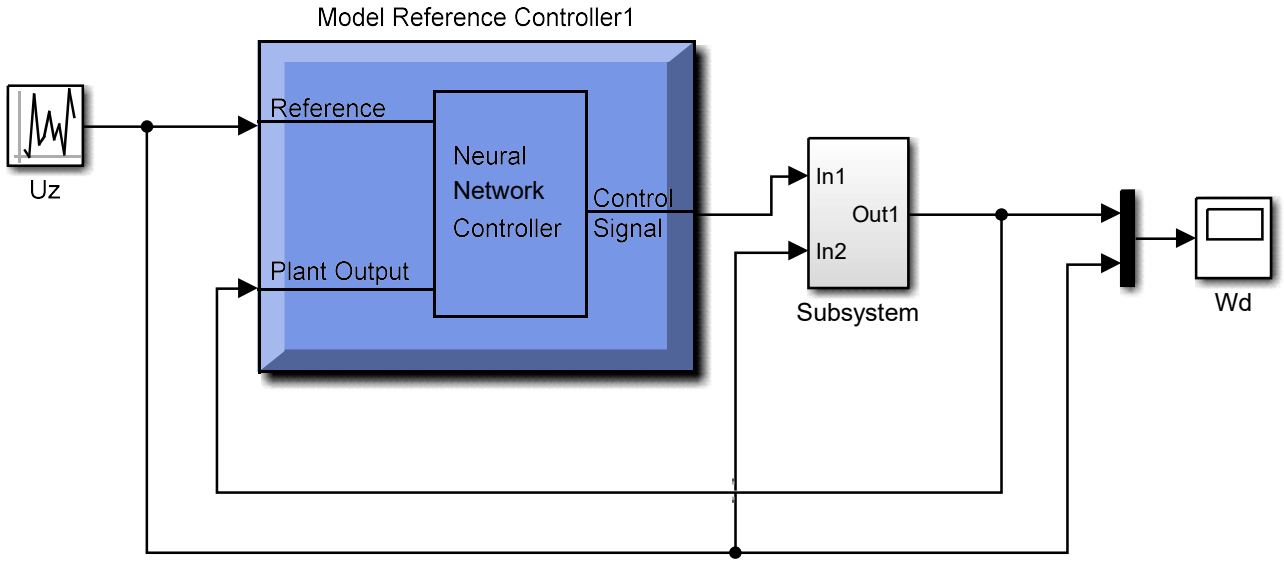


Рис.5.3. Схема системи управління з нейрорегулятором Model Reference Controller

Структурна схема нейрорегулятора Model Reference Controller показана на рис.5.4. Дана схема з'являється в окремому вікні при виборі пункту меню **Look Under Mask** блоку **Model Reference Controller**.

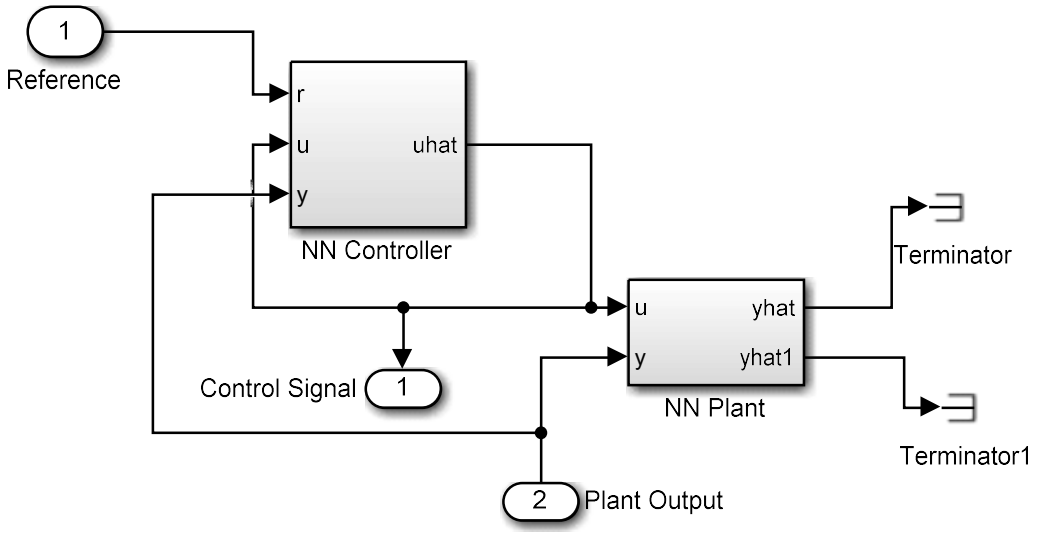


Рис.5.4. Структурна схема нейрорегулятора Model Reference Controller

Процес синтезу нейрорегулятора починається шляхом активізації блоку **Model Reference Controller**. З'являється вікно, показане на рис.5.5.

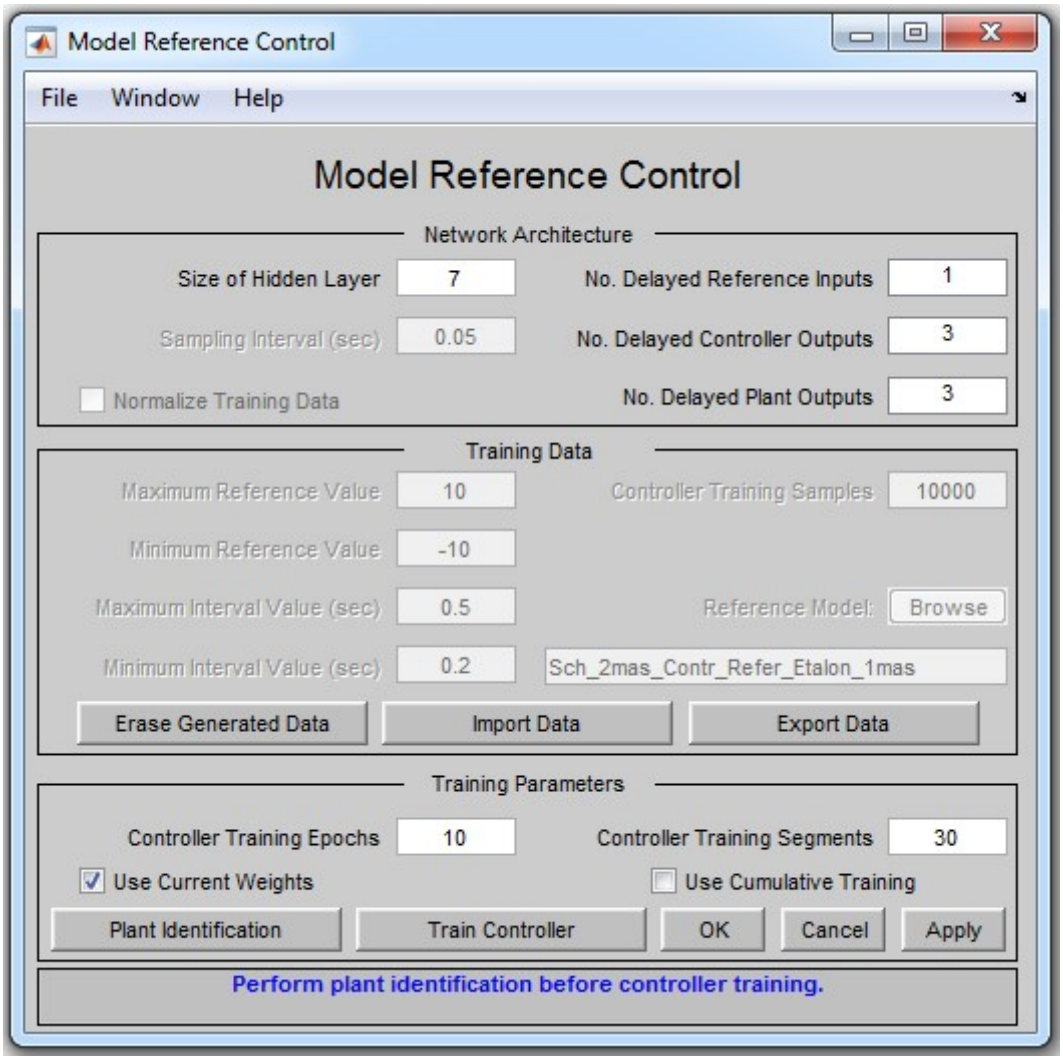


Рис.5.5. Вікно завдання параметрів нейрорегулятора Model Reference Controller

Особливість даної системи управління полягає в тому, що виконується побудова двох нейронних мереж: моделі об'єкту управління і самого регулятора. Спочатку виконується побудова моделі об'єкту управління, для чого у вікні, показаному на рис.5.5 активізується кнопка **Plant Identification**. При цьому відкривається вікно **Plant Identification**, і виконується побудова нейромережевої моделі об'єкту управління так само, як і при синтезі регулятора **NN Predicteve Controller**.

Числові значення елементів матриць вагів **IW{1,1}, LW{2,1}** і зсувів **b{1}, b{2}** нейронної мережі моделі об'єкту управління заносяться в пам'ять машини і використовуються потім при побудові нейронної мережі регулятора. Після завершення побудови нейромережевої моделі керованого об'єкту відбувається повернення до вікна завдання параметрів **Model Reference Control** (рис.5.5).

У вікні **Model Reference Control** вводиться параметр **Controller Training Segments**, який задає кількість сегментів (або ділянок часу), на які розбивається період навчання нейронного регулятора під час синтезу за еталонною моделлю.

Це означає, що замість того, щоб проводити навчання на одному безперервному періоді симуляції, навчальний процес ділиться на кілька окремих сегментів. Для кожного з цих сегментів обчислюється похибка (відхилення від поведінки еталонної моделі), а отримані дані використовуються для коригування параметрів нейронної мережі.

Основні міркування щодо використання цього параметру:

- *Покращення стабільності навчання:* Розбиття навчального періоду на менші частини може допомогти краще врахувати змінність динаміки системи протягом часу.
- *Локалізація помилок:* Аналіз помилки на окремих сегментах дозволяє виявити проблемні ділянки або нелінійності в поведінці об’єкта керування.
- *Гнучкість***:** Значення цього параметру може бути підібране експериментально залежно від характеру системи та вимог до якості регулювання. Занадто велика кількість сегментів може ускладнити процес оптимізації, а занадто мала – не врахувати всі особливості динаміки.

Таким чином, **Controller Training Segments** – це інструмент для тонкого налаштування процесу навчання нейронного регулятора, який дозволяє адаптувати модель до змінних умов роботи системи та підвищити якість регулювання.

Для навчання нейронної мережі регулятора необхідно згенерувати навчальні дані. Як еталонну модель приймаємо модель одномасової системи. Для генерації навчальної послідовності активізується кнопка **Generate Training Data**. Ці дані у вигляді графіків з'являться у вікні **Input-Output Data NN Model Reference Control.** Необхідно підтвердити або відкинути ці дані. Якщо дані прийнятні, то у вікні **Model Reference Control** вибирається кнопка **Train Controler** (Навчити регулятор).

Після натиснення на кнопку **Train Controler** відбувається створення і ініціалізація мережі динамічної мережі із заданим числом затримок по входу і виходу моделі і регулятора **netn.** На рис.5.6 показані моделі елементів нейронної мережі, по-

будовані за допомогою оператора **gensim(netn).** З даних елементів в системі Simulink побудована схема мережі, показана на рис.5.7.

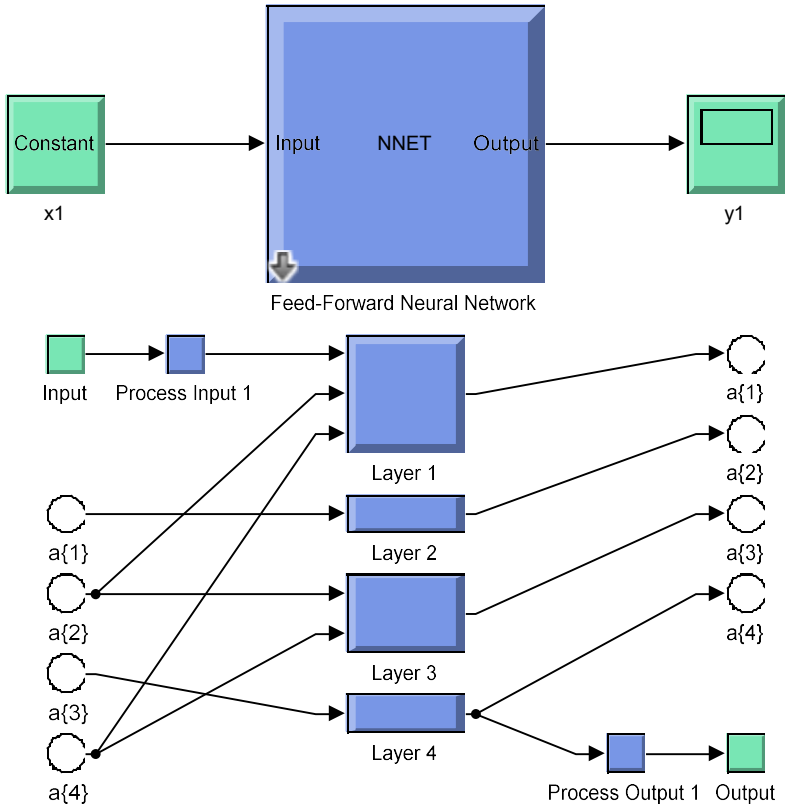


Рис.5.6. Моделі елементів динамічної мережі нейрорегулятора Model Reference Controller

Мережа має 4 шари. Параметри першого і другого шару задаються у вікні **Model Reference Control**. У даному випадку розмір першого шару $S_1 = 7$, у другому шарі є 1 нейрон. Параметри третього і четвертого шарів відповідають параметрам нейромережевої моделі об'єкту управління, отримані в результаті виконання процедури ідентифікації (в даному випадку: розмір третього шару $S_3 = 10$, четвертого – $S_4 = 1$). Використовувані функції активації: гіперболічного тангенса (**tansig**) – в першому і третьому шарі, лінійна (**purelin**) – в другому і четвертому шарах. Кількість елементів запізнювання на вході регулятора N_{rc}, кількість елементів запізнювання на виході регулятора N_{ic} і кількість елементів запізнювання на виході моделі об'єкту N_{jc} задаються у вікні **Model Reference Control** (див. рис.5.5). У даному прикладі $N_{rc} = 1, N_{ic} = 3, N_{jc} = 3$.

Елементам матриць вагів і зсувів третього і четвертого шару динамічної мережі присвоюються відповідні значення матриць вагів і зсувів першого і другого шару мережі, що відповідає нейромережевої моделі об'єкту управління, отриманій при виконанні процедури ідентифікації.

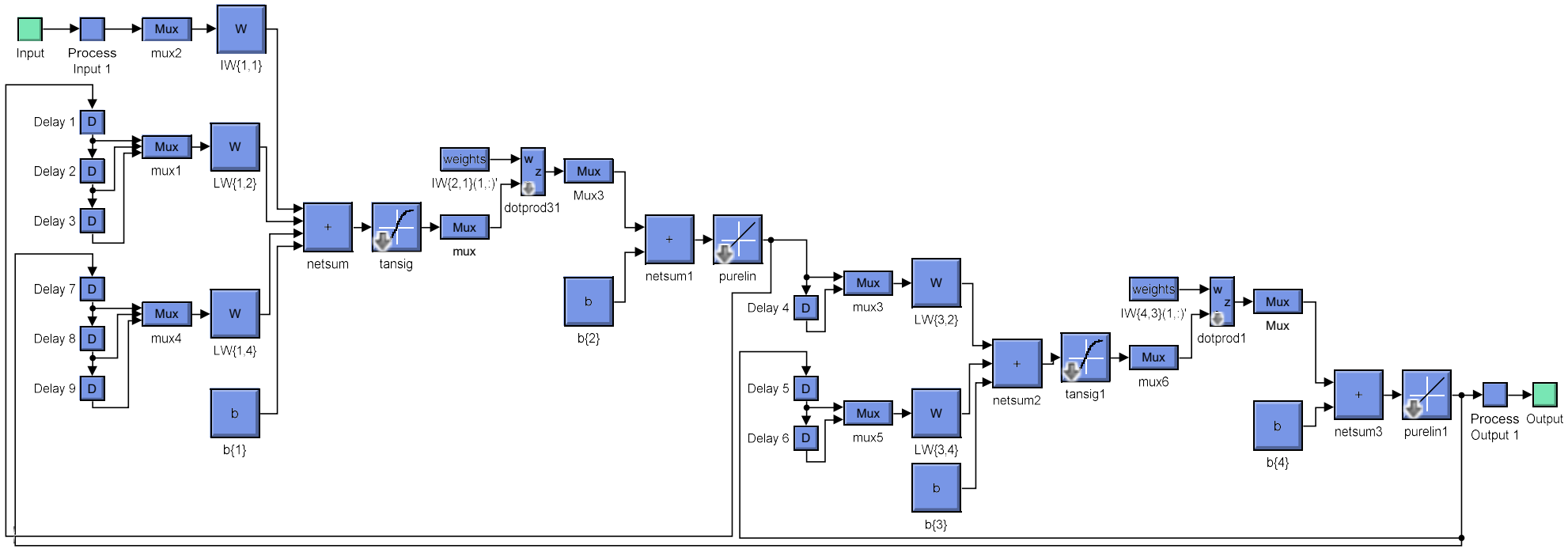


Рис.5.7. Модель динамічної мережі, що формується при синтезі нейрорегулятора Model Reference Controller

Після створення мережі виконується процес її навчання. Параметр навчання вагів і зсувів третього і четвертого шарів встановлюється рівним 0, унаслідок чого вони залишаються незмінними в процесі тренування, а змінюються ваги і зсуви першого і другого шарів, тобто параметри нейромережевої моделі нейрорегулятора.

Процес та результати навчання наочно демонструються за допомогою діалогового вікна **Neural Network Training (nntraintool)**, показаного на рис..5.8.

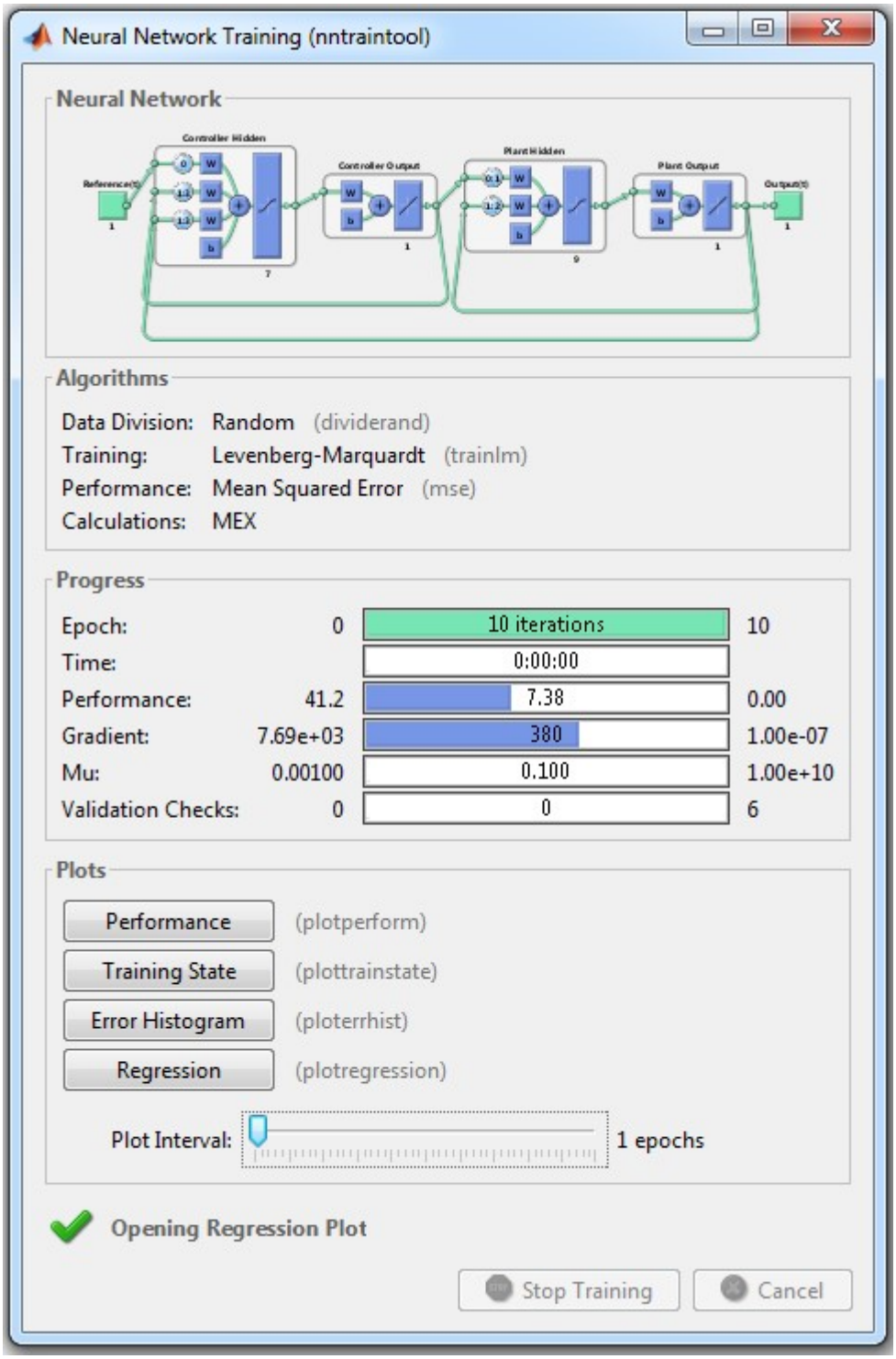


Рис. 5.8. Вікно, що відображає хід навчання та результати навчання нейронної мережі при тренувані нейронної мережі регулятора

При натисканні на кнопку **Performance** відкривається вікно **Neural Network Training Performance**, показане на рис. 5.9. У вікні відображаються графік **Mean Squared Error (mse)**, який показує, як змінюється середньоквадратична помилка навчальння.

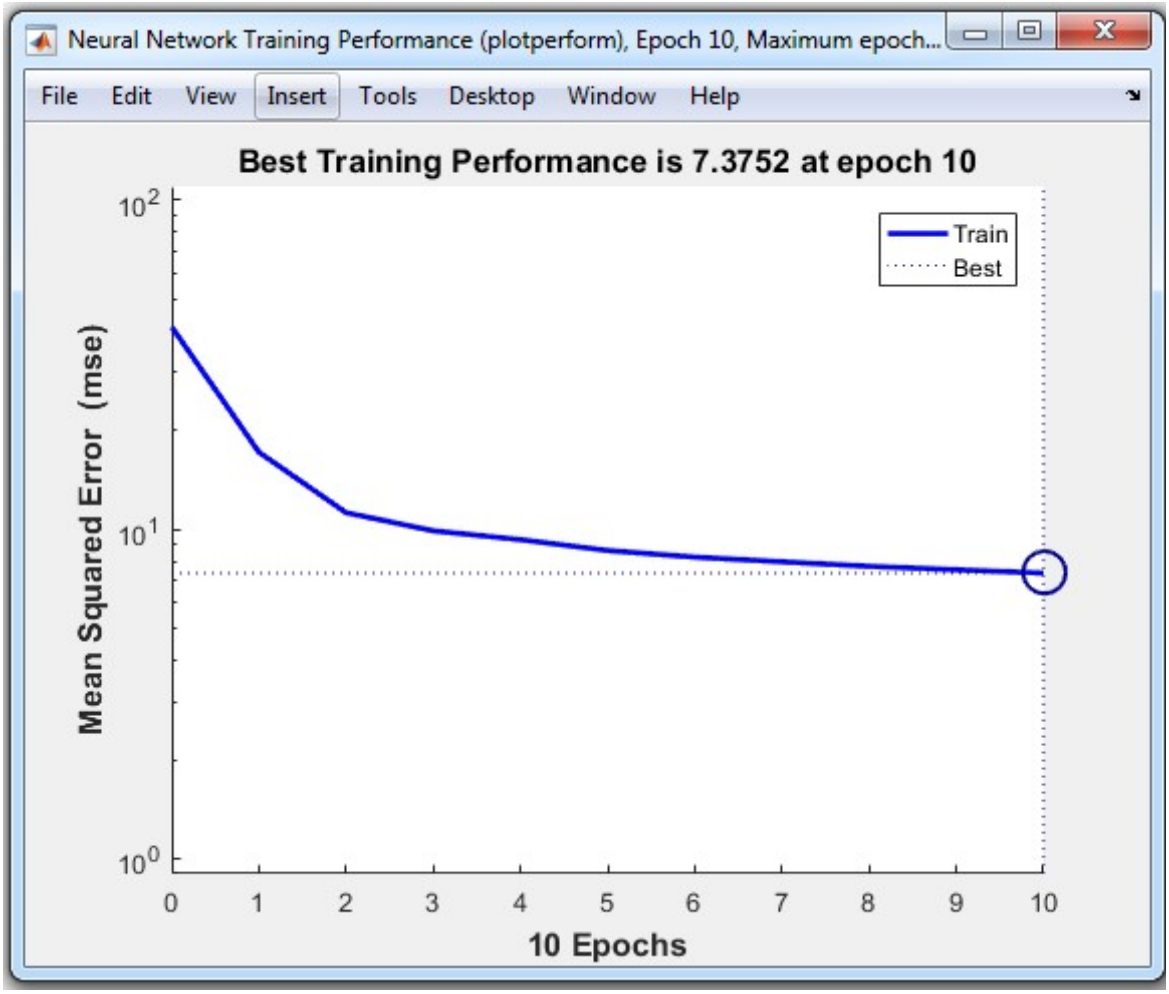


Рис.5.9. Вікно контролю процесу навчання нейронної мережі нейрорегулятора Model Reference Controller

При натисканні на кнопку **Training State** відкривається вікно **Neural Network Training State** (рис. 5.10**)** в якому відображається інформацію про стан навчання нейронної мережі на певний момент часу під час процесу навчання.

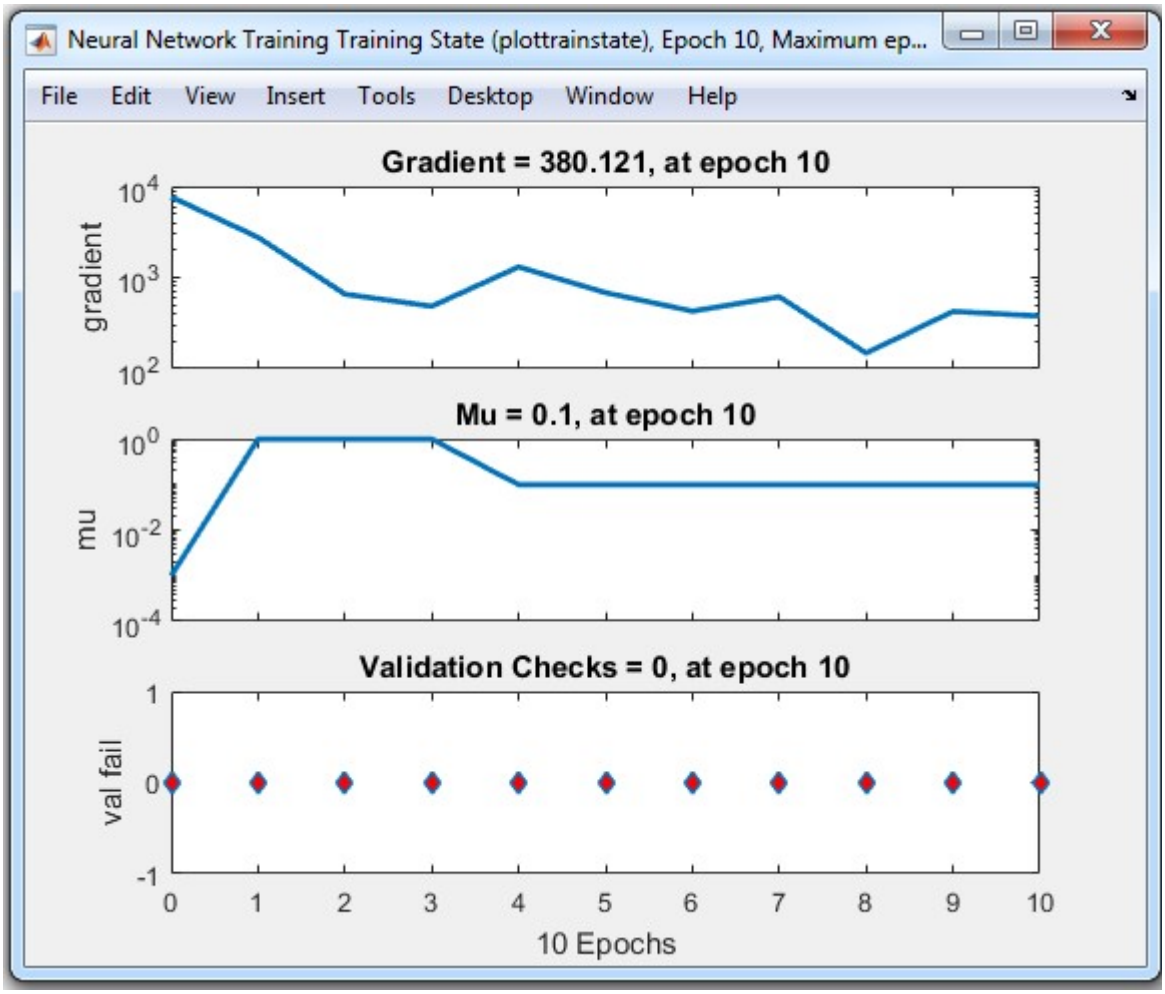


Рис.510. Вікно Neural Network Training State

Кнопка **Error Histogram** у вікні **Neural Network Training** ***(nntraintool)*** відображає графічне подання розподілу помилок, які виникли під час навчання нейронної мережі. Іншими словами, вона показує гістограму, де по осі *X* відкладаються значення помилок (різниця між бажаними виходами, що задає еталонна модель, і фактичними виходами мережі), а по осі *Y* – кількість зразків, для яких відповідна помилка зустрічається.

Основні моменти:

- *Аналіз якості навчання:* Гістограма дає змогу швидко оцінити, чи більшість помилок невеликі і розподілені симетрично навколо нуля, що є ознакою успішного навчання.
- *Виявлення відхилень:* Якщо гістограма показує значну кількість зразків з великими помилками або нерівномірний розподіл, це може сигналізувати про проблеми в процесі навчання або потребу в налаштуванні параметрів мережі.
- *Контроль за адаптацією регулятора:* У контексті синтезу нейрорегулятора з еталонною моделлю, аналіз гістограми помилок допомагає переконатися, що регулятор наближає поведінку системи до бажаної (еталонної) динаміки.

Таким чином, натискання кнопки **Error Histogram** відкриває додаткове вікно з гістограмою помилок, що дозволяє користувачу в режимі навчання отримати візуальну інформацію про розподіл помилок та, за необхідності, скорегувати процес навчання чи архітектуру нейронної мережі (рис.5.11).

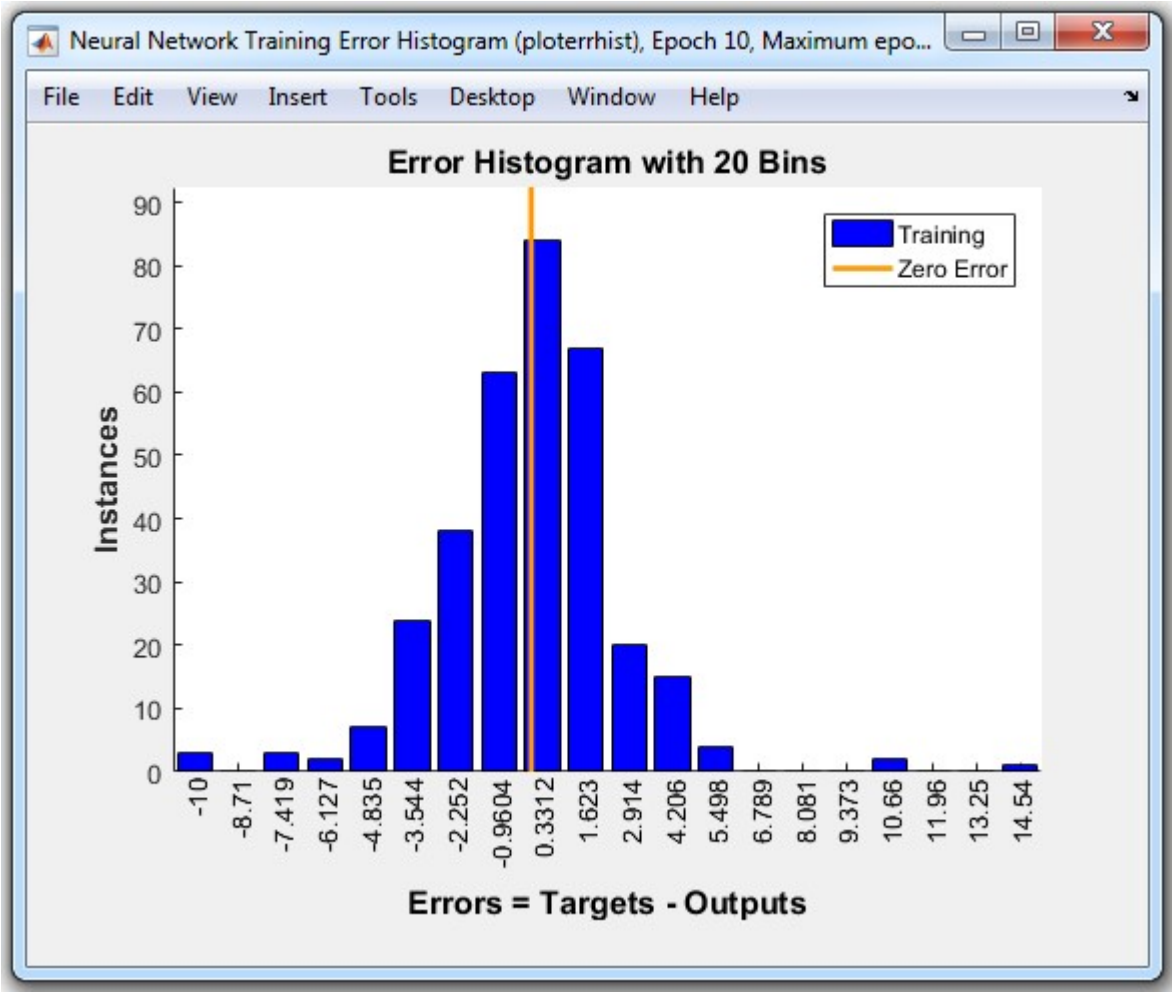


Рис. 5.11. Вікно Neural Network Training Error Histogram

При натисканні на кнопку **Regression** відкривається вікно **Neural Network Training Regression** (рис. 5.12), у якому показані графіки та інформація, які допомагають в оцінці та аналізі продуктивності нейронної мережі під час навчання.

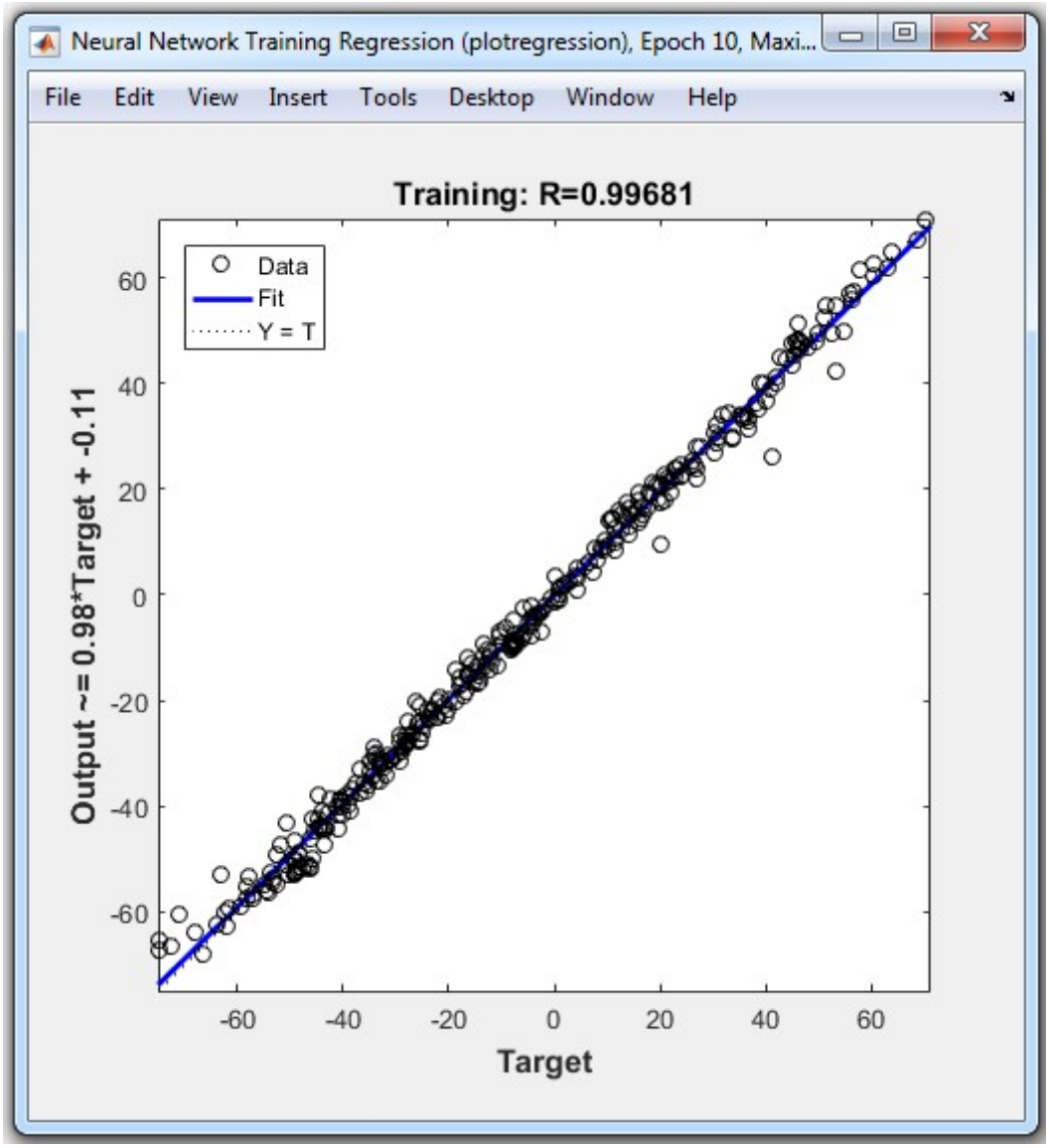


Рис. 5.12. Вікно Neural Network Training Regression

Після того, як навчання закінчене, графіки виходів еталонної моделі і об'єкту управління виводяться на екран у вікні, показаному на рис.5.13.

Якщо точність стеження за еталонною моделлю незадовільна, то можна продовжити навчання регулятора з тим же набором даних, знову скориставшись кнопкою **Train Controller**. Якщо для продовження навчання необхідно використовувати новий набір даних, можна скористатися кнопками **Generate Data** або **Import Data**. Можна продовжити навчання з вибраними вагами, для чого слід зробити відмітку у вікні контролю **Use Current Weight**.

Після закінчення процесу навчання числові значення елементів матриць вагів і зсувів регулятора (тобто першого і другого шарів) вводяться в блок **NN Controller**, а числові значення елементів матриць вагів і зсувів об'єкту (тобто. третього і четвертого шарів) вводяться в блок **NN Plant** системи Simulink (див. рис.5.7). У системі Simulink дані шари представляються у вигляді структурних схем, показаних на рис.5.14 і рис.5.15.

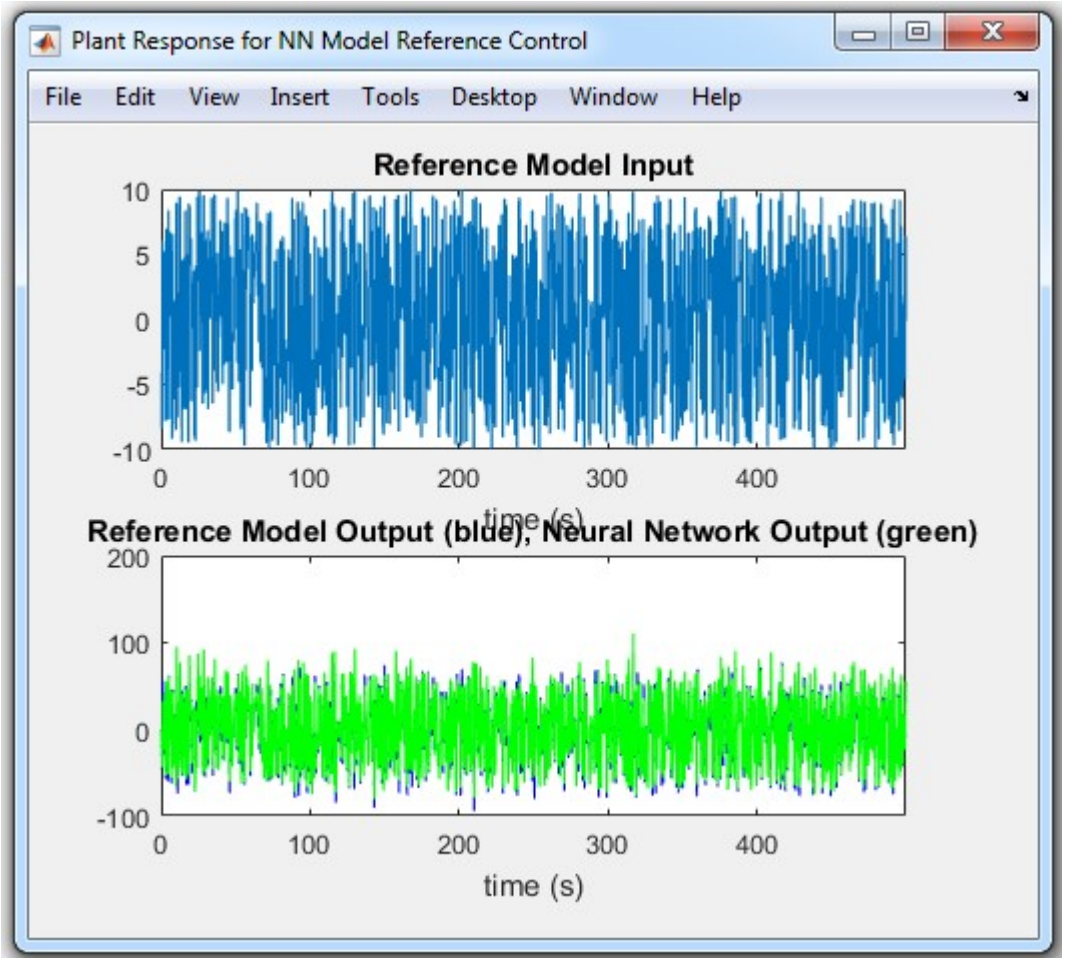


Рис.5.13. Результати тренування мережі регулятора Model Reference Controller

Як випливає з схеми рис.5.15, кількість елементів затримки на вході і виході моделі об'єкту управління може бути рівною тільки 2, інакше з'являється повідомлення про помилку при моделюванні нейромережевої системи. Це вносить істотні обмеження при побудові нейромережевої моделі об'єкту.

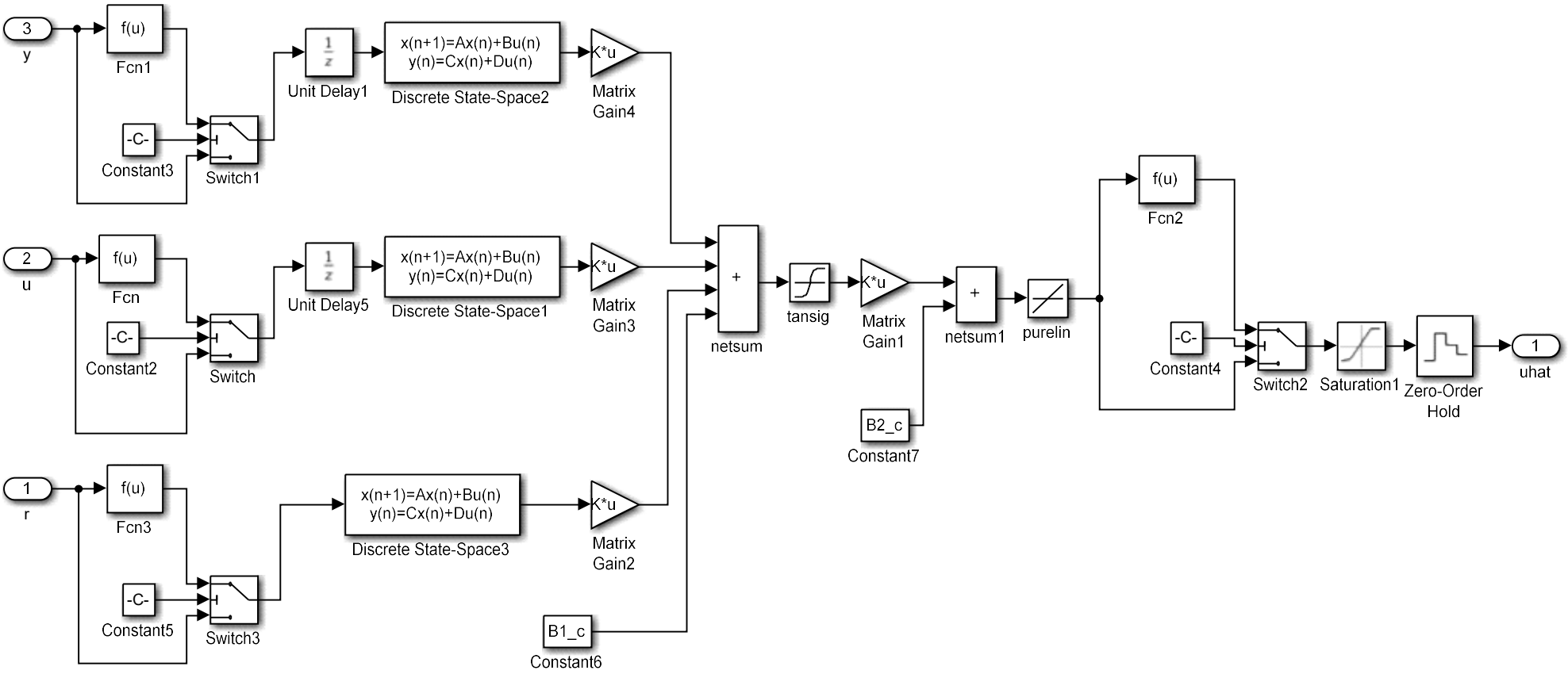


Рис.5.14. Структурна схема блоку **NN Controller** схеми Model Reference Controller

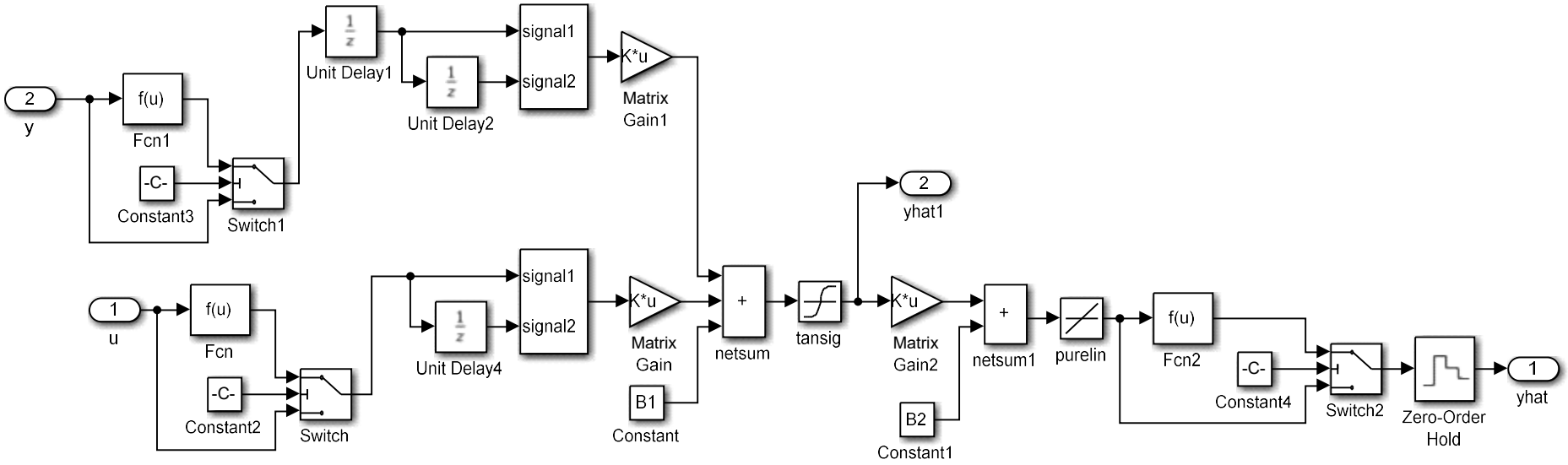


Рис.5.15. Структурна схема блоку **NN Plant** схеми Model Reference Controller

5.4 Вибір параметрів нейрорегулятора Model Reference Controller

При побудові нейронної мережі об'єкту управління і нейронної мережі регулятора кількість нейронів прихованого шару варіювались в широких межі. Оптимальні значення знаходяться в межах $S = 10 \div 15$. Параметри N_B, Δt, $t_{\min}$, $t_{\max}$ приймались такими ж, як і при синтезі NN Predictive Controller.

Кількість елементів запізнювання на вході регулятора N_{rc}, на виході регулятора N_{ic} і на виході моделі об'єкту N_{jc} (при синтезі нейронної мережі регулятора) варіювалося в межах: $N_{rc} = 1 \div 4$, $N_{ic} = 1 \div 5$, $N_{jc} = 1 \div 5$.

Як указувалося вище, при навчанні нейронної мережі регулятора всі навчальні дані розбиваються на n сегментів і з використанням кожного сегменту виконується $N_{\text{ц}}$ циклів навчання. Кількість циклів навчання $N_{\text{ц}}$, після закінчення яких помилка навчання переставала зменшуватися, складало $N_{\text{ц}} = 10 \div 20$ при $n = 30$.

Як зазначалося вище, у якості еталонної моделі приймалася одномасова система. Дослідження показали, що ні при яких параметрах нейрорегулятора Model Reference Controller не вдалося отримати задовільні динамічні характеристики системи.

5.5 Завдання для самостійного виконання

Варіанти завдань для синтезу нейрорегулятора **Model Reference Controller** для двомасових і трьохмасових електромеханічних систем наведено в додатку А.

5.6 Контрольні питання по темі заняття

1. Поясніть принцип дії регулятора з прогнозом NN Predictive Controller, регулятора на основі моделі авторегресії з ковзаючим середнім NARMA-L2 Controller, регулятора на основі еталонної моделі Model Reference Controller.
2. Поясніть структуру нейронних мереж, які використовуються в регуляторі.
3. Наведіть структурніу схему системи управління з нейрорегулятором Model Reference Controller.
4. Перечисліть основні етапи синтезу нейромережевої системи управління, побудованих на основі регулятора моделі Model Reference Controller.
5. Як виконується формування навчальної послідовності?
6. Як відбувається контроль процесу навчання мережі?
7. Як вибираються параметри нейрорегулятора?
8. Які функції виконують нейронні мережі в регуляторі?
9. Скільки нейронних мереж містить нейрорегулятор Model Reference Controller?

СПИСОК ЛІТЕРАТУРИ

1. Технології нейронних мереж і нечіткого модел ювання в системах управління : підруч. для здобувачів вищої освіти спец. 151 Автоматизація та комп'ютерно-інтегровані технології / Г.І. Канюк, Б.І. Кузнецов, Т.Ю. Василець, А.Ю. Мезеря, О.О. Варфоломієв. – Харків : Друкарня Мадрид, 2020. – 306 с.
2. Нейромережеві технології в системах управління: Підручник для вузів./ Б. І. Кузнецов, Т.Ю. Василець, Т.Б. Нікітіна, В. В. Коломиєць, О.О. Варфоломієв; Укр. інж.-пед. акад.. - Харків: УІПА, 2014. - 232 с.
3. Кирик В. В. Математичний апарат штучного інтелекту в електроенергетичних системах:підручник..– Київ: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2019. –224 с.
4. Штучні нейронні мережі: навчальний посібник / С. В. Ткаліченко. – Кривий Ріг, 2023. –150 с.
5. Штучні нейронні мережі: навчальний посібник для студентів вищих навчальних закладів / О.Г. Руденко, Є.В. Бодянський. – К: Компанія СМІТ, 2006, 404 с.
6. Желдак Т.А. Нечіткі множини в системах управління та прийняття рішень: навч. посіб. / Т.А. Желдак, Л.С. Коряшкіна, С.А. Ус, за редакцією С.А. Ус ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2020. – 387 с.
7. Кирик В. В. К43 Математичний апарат штучного інтелекту в електроенергетичних системах: підручник / В. В. Кирик.– Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка» 2019.– 224с.
8. Антоненко В. М., Мамченко С.Д., Рогушина Ю.В. Сучасні інформаційні системи і технології: управління знаннями: навчальний посібник. – Ірпінь : Національний університет ДПС України, 2016. – 212 с.
9. Глибовець М.М., Отецький О.В. Штучний інтелект. – К.: Вид. дім «КМ Академія», 2002. – 366 с.
10. Ямпольський Л. С. Системи штучного інтелекту в плануванні, моделюванні та управлінні : підруч. для студ. вищ. навч. закл. / Л. С. Ямпольський, Б. П. Ткач, О. І. Лісовиченко. - К. : ДП «Вид. дім «Персонал», 2011. - 544 с.
11. Методи та системи штучного інтелекту: Теорія і практика: Навчальний посібник / О.С. Булгаков, В.В. Зосімов, В.О. Поздєєв. – Одеса. : Олді плюс, 2020. – 356с.
12. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с
13. Тимощук П.В. Штучні нейронні мережі; Навч. посібн. - Львів: Львівська політехніка, 2011. -444 с.
14. Кононюк, А. Ю. Нейронні мережі і генетичні алгоритми. - Київ: Корнійчук, 2008. - 468 с.
15. Методи та системи штучного інтелекту: Навчальний посібник./ Уклад. : А.С. Савченко, О. О. Синельніков. – К. : НАУ, 2017. – 190 с.

16. Методи та системи штучного інтелекту: Навчальний посібник / Уклад.: І.М. Удовик, Г.М. Коротенко, Л.М. Коротенко, В.О. Трусов, А.Т. Харь. – Д.: Державний ВНЗ «Національний гірничий університет», 2017. – 105 с.
17. Методи та системи штучного інтелекту: навч. посіб. / укл.Д.В. Лубко, С.В. Шаров. – Мелітополь: ФОП Однорог Т.В., 2019. – 264 с.
18. Паламар М. І. Комп'ютерні технології штучного інтелекту для прецизійного управління у мехатронних системах : навч. посіб. / Михайло Паламар, Михайло Стрембіцький ; Тернопіл. нац. техн. ун-т ім. Івана Пулюя. Тернопіль : Тернопіл. нац. техн. ун-т ім. Івана Пулюя, 2018. 127 с.
19. Литвин В.В. Інтелектуальні системи : підручник / В.В. Литвин, В.В. Пасічник, Ю.В. Яцишен. – Львів: Новий світ, 2009. – 405 с.
20. Троцько В.В., Методи штучного інтелекту: навчально-методичний і практичний посібник. / Троцько В.В. – Київ: Університет економіки та права «КРОК», 2020. – 86 с.
21. Куклін В. М. К Подання знань і операції над ними; навчальний посібник. / В. М. Куклін. Харків: ХНУ імені В. Н. Каразіна, 2019 – 164 с.
22. Снитюк В. Є. Прогнозування. Моделі, методи, алгоритми / В. Є. Снитюк. – Київ : Маклаут, 2008. – 364 с.
23. Івахів, Орест Васильович. Основи побудови систем керування з нечіткою логікою : навчальний посібник /О. Івахів, М. Наконечний.– Львів :Растр-7,2017. – 129 с.
24. Коротка, Лариса Іванівна. Обчислювальний інтелект : теорія нечітких множин: навчальний посібник / Коротка Л.І., Зеленцов Д.Г., Науменко Н.Ю.,Ляшенко О.А., Солодка Н.О.– Дніпро :ДВНЗ УДХТУ,2020. – 161 с.
25. Шушура О.М. Методологічні основи побудови інформаційних технологій для автоматизації управління складними системами на принципах нечіткої логіки : дис. … доктора технічних наук : 05.13.06 / Шушура Олексій Миколайович. − К., 2018. − 332 с.
26. Beale M. H., Carson E., Demuth H. B. *Deep Learning Toolbox: MATLAB R2021a*. The MathWorks, 2021. https://www.mathworks.com/help/deeplearning/

ДОДАТОК А

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Варіант 1

Файл вихідних даних

```
clear;clc;echo on;

% Система управління електроприводом
% механізму підйому шахтної пїдїймальної установки
% Система ТП-Д, двомасова

% Вихідні дані
ktp=44;
Tm=0.005;
Tms=2*Tm;
Re=0.0335;
Te=0.06325;
CF=118;
kdc=0.032;
kdv=4.72;
In=3720;
Mn=In*CF;
Jd=194892;
Jm=43508;
Js=Jd+Jm;
w0=3;
c12=(w0^2)*(Jd*Jm)/(Jd+Jm);
b12=0
Uz=24;

kic=Re/(2*Tm*kdc*ktp);
kpc=kic*Te;
kiv=(kdc*Js)/(32*Tm^2*CF*kdv);
kpv=8*Tm*kiv;

% Матриці
A=[-b12/Jm 1/Jm b12/Jm zeros(1,4);
-c12 0 c12 zeros(1,4);
b12/Jd -1/Jd -b12/Jd CF/Jd zeros(1,3);
0 0 -CF/(Re*Te) -1/Te 1/(Re*Te) 0 0;
0 0 (-kpv*kpc*ktp*kdv)/Tm -(kdc*kpc*ktp)/Tm -1/Tm ktp/Tm ktp*kpc/Tm;
0 0 -kdv*kpv*kic -kic*kdc 0 0 kic;
0 0 -kdv*kiv zeros(1,4)]
B=[0 -1/Jm;
zeros(3,2);
(kpv*kpc*ktp)/Tm 0;
kpv*kic 0;
kiv 0]
C=zeros(6,7);
C(1,1)=1;C(2,2)=1;C(3,3)=1;C(4,4)=1;C(5,6)=1;C(6,7)=1;
D=zeros(6,2);
```

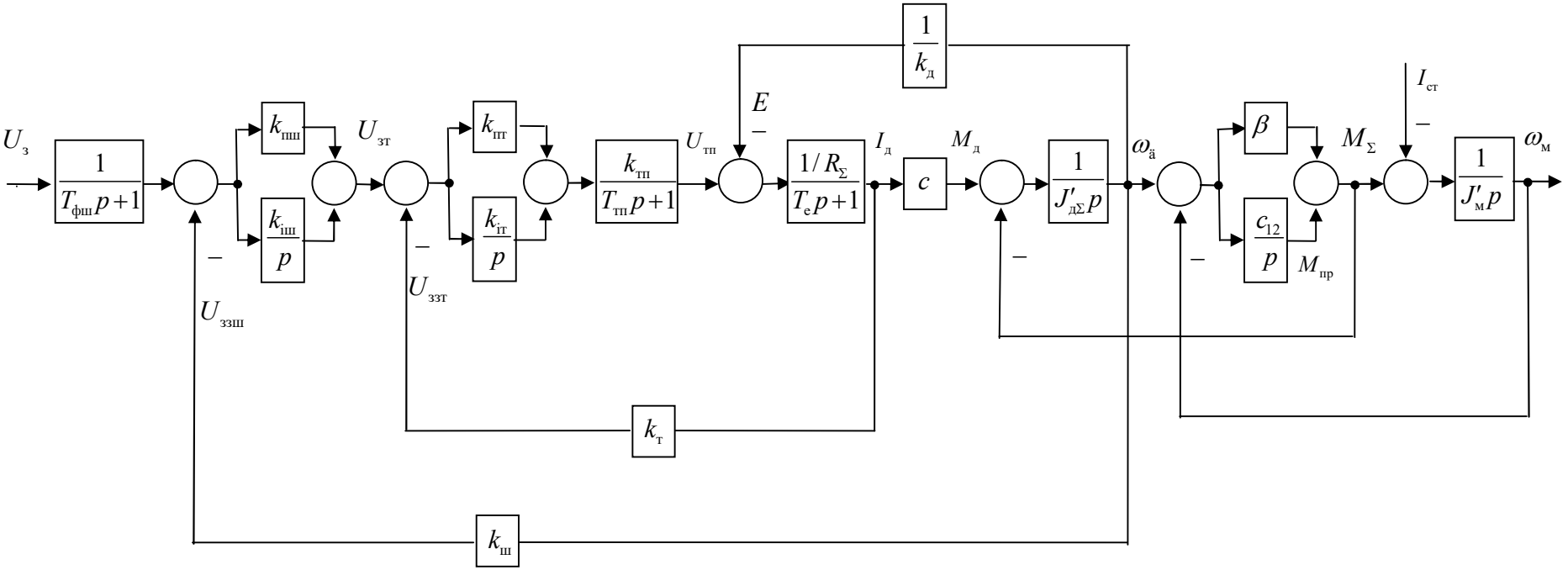


Рис. А.1. Алгоритмічна схема двомасової системи управління електроприводом ТП-Д механізму підйому шахтної пїдіймальної установки

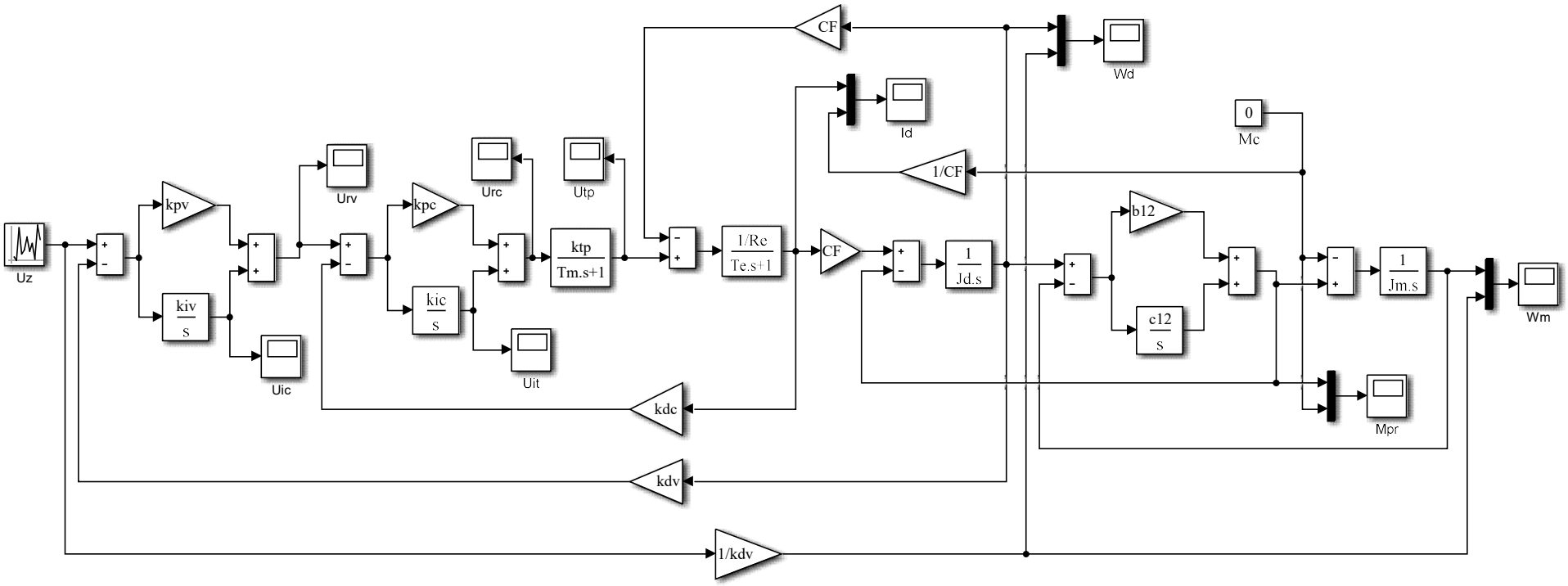


Рис. А.2. Схема моделі двомасової системи управління електроприводом ТП-Д механізму підйому шахтної пїдїймальної установки, розроблена в Simulink ситеми MATLAB

Варіант 2

Файл вихідних даних

```
clear;clc;echo on;

% Система управління електроприводом
% механізму підйому шахтної пїдїймальної установки
% Система ТП-Д, трьохохмасова

% Вихідні дані
ktp=66;
Tm=0.005;
Tms=2*Tm;
Re=0.0228;
Te=0.069;
CF=165;
kdc=0.0038;
kdv=6.7;
In=3180;;
Mn=In*CF;
Jd=532864;
Jk=51097;
Jm=145990;
Js=Jd+Jm+Jk;
Js=729823;
w0=2;
c=(w0^2)*((Jd+Jk/2)*(Jm+Jk/2))/((Jd+Jk/2)+(Jm+Jk/2));
c1=2*c;
c2=2*c1;
b1=0.01*c1;
b2=0.01*c2;

kic=Re/(2*Tm*kdc*ktp);
kpc=kic*Te;
kiv=(kdc*Js)/(32*Tm^2*CF*kdv);
kpv=8*Tm*kiv;
```

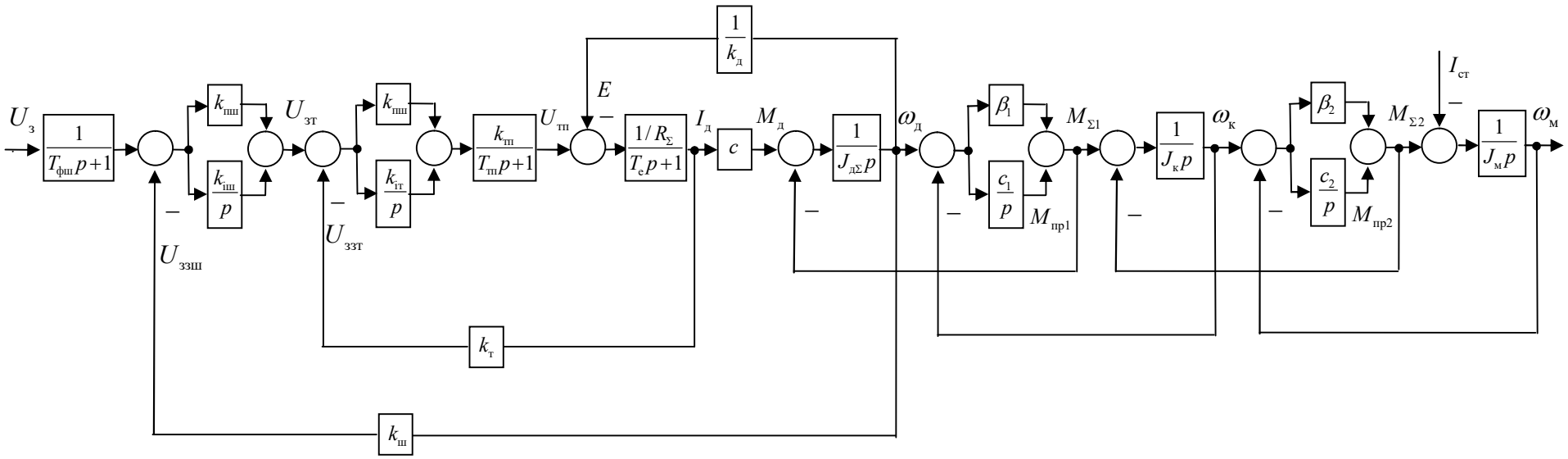


Рис. А.3. Алгоритмічна схема трьохмасової системи управління електроприводом ТП-Д механізму підйому шахтної пїдїймальної установки

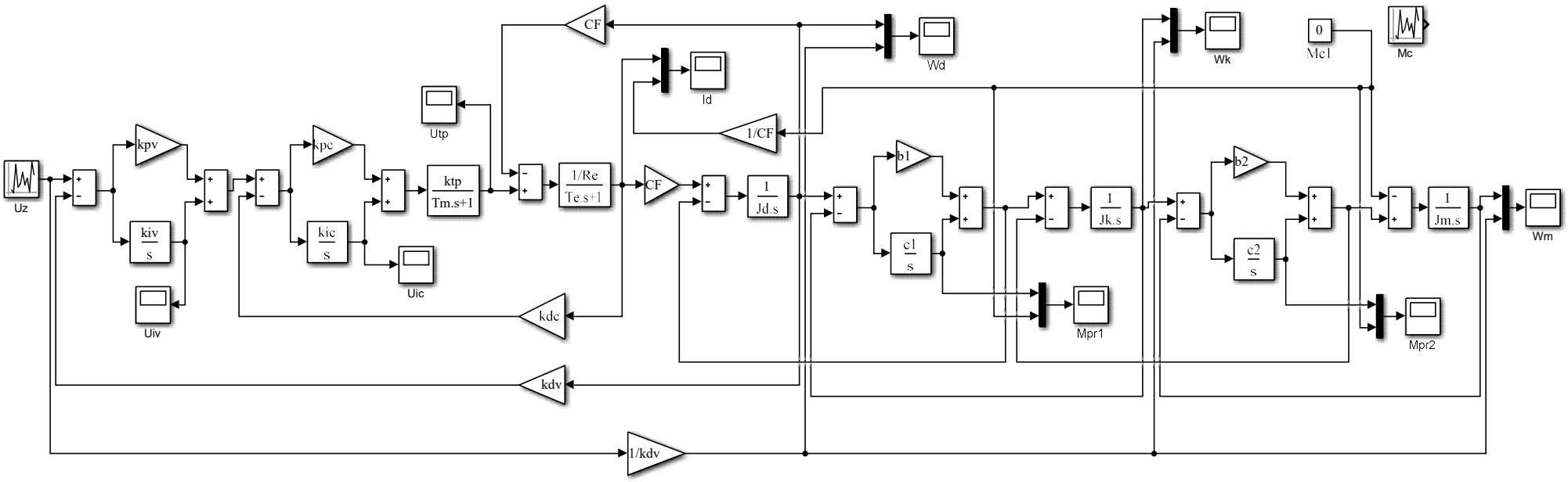


Рис. А.4. Схема моделі трьохасової системи управління електроприводом ТП-Д механізму підйому шахтної піддіймальної установки, розроблена в Simulink ситеми MATLAB

Варіант 3

Файл вихідних даних

```
clear;clc;echo on;

% Система управління електроприводом
% механізму підйому шахтної підіймальної установки
% Система Г-Д, двомасова

% Вихідні дані
kg=3.34;
Tg=4.08;
ktp=22;
Tm=5e-3;
Tms=0.02;
Re=0.0254;
Te=0.036;
CF=170;
In=2624;
Mn=In*CF;
kdn=0.035;
kdc=3.66e-3;
kdv=5.44;
Jd=172098;
Jm=59764;
Js=Jd+Jm;
w0=1.5;
c12=(w0^2)*(Jd*Jm)/(Jd+Jm);
b12=0.005*c12;
kin=1/(2*Tm*ktp*kg*kdn);
kpn=Tg*kin;
kic=kdn*Re/(4*Tm*kdc);
kpc=Te*kic;
kiv=kdc*Js/(128*Tm^2*CF*kdv);
kpv=16*Tm*kiv;
Uz=24;
```

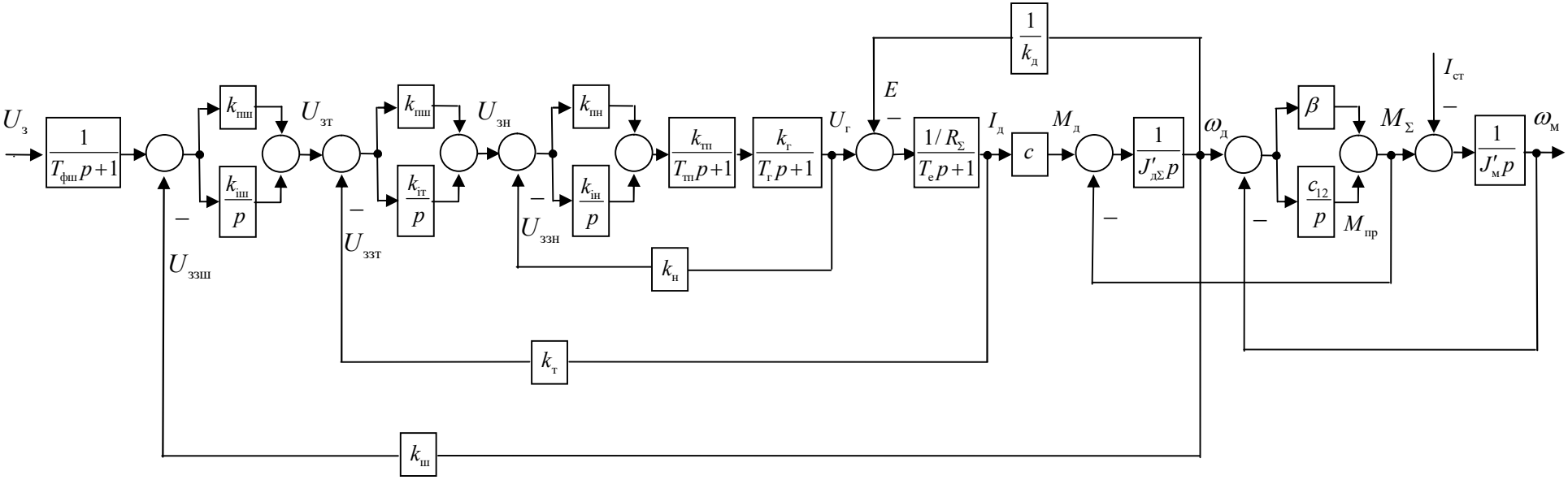


Рис. А.5.Алгоритмічна схема двомасової системи управління електроприводом Г-Д механізму підйому шахтної пїдїймальної установки

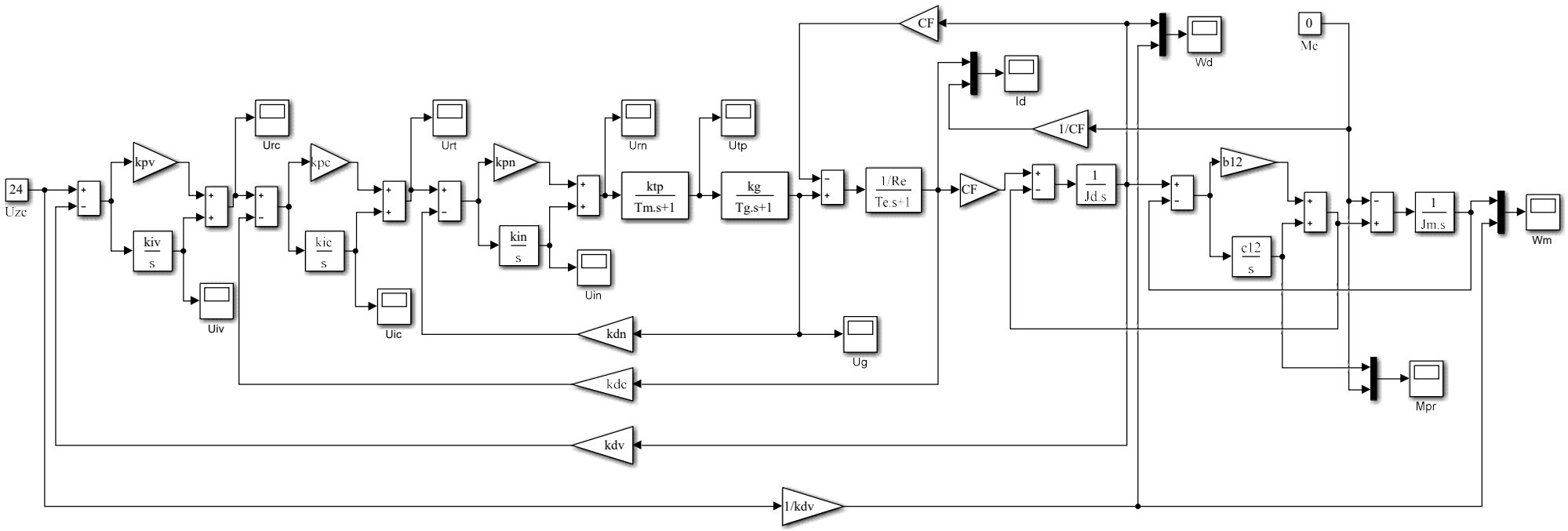


Рис. А.6. Схема моделі двоасової системи управління електроприводом Г-Д механізму підйому шахтної пїдїймальної установки, розроблена в Simulink ситеми MATLAB

Варіант 4

Файл вихідних даних

```
clear;clc;echo on;

% Система управління електроприводом
% механізму підйому шахтної підіймальної установки
% Система Г-Д, трьохмасова

% Вихідні дані
kg=2.845;
Tg=2.9;
ktp=22;
Tm=5e-3;
Tms=0.02;
Re=0.0269;
Te=0.056;
CF=118;
In=3720;
Mn=In*CF;
kdn=0.04;
kdt=2.58e-3;
kds=4.72;
Jd=163917;
Jk=15718;
Jm=54400;
Js=Jd+Jm+Jk;
w0=3;
c=(w0^2)*((Jd+Jk/2)*(Jm+Jk/2))/(Jd+Jm+Jk);
c1=c*2;
c2=c1;
b1=0.005*c1;
b2=0.005*c2;
kin=1/(2*Tm*ktp*kg*kdn);
kpn=Tg*kin;
kit=kdn*Re/(4*Tm*kdt);
kpt=Te*kit;
kis=kdt*Js/(128*Tm^2*CF*kds);
kps=16*Tm*kis;
```

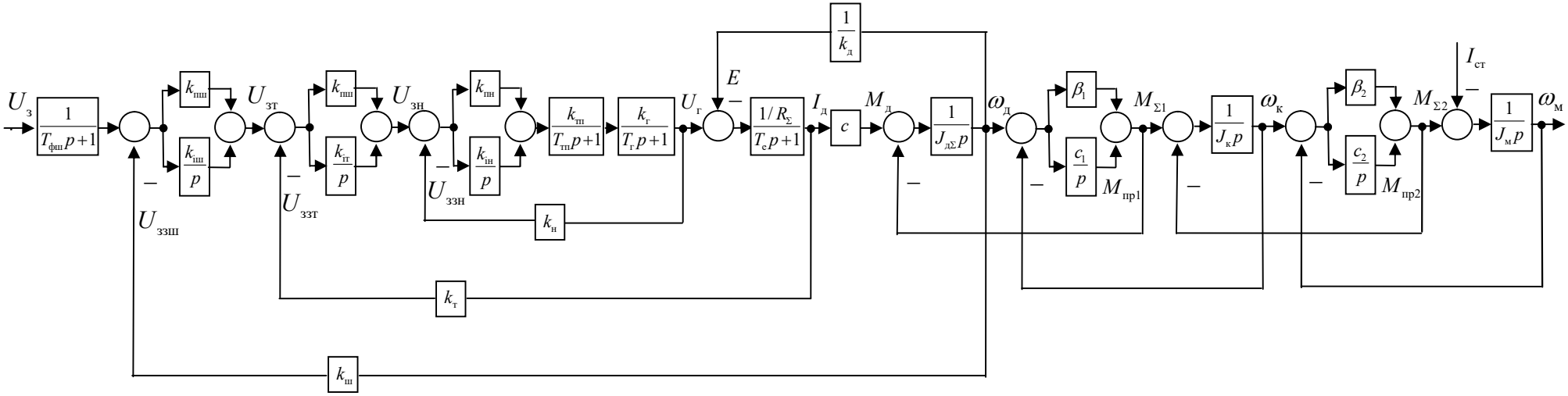


Рис. А.7.Алгоритмічна схема трьохмасової системи управління електроприводом Г-Д механізму підйому шахтної пїдїймальної установки

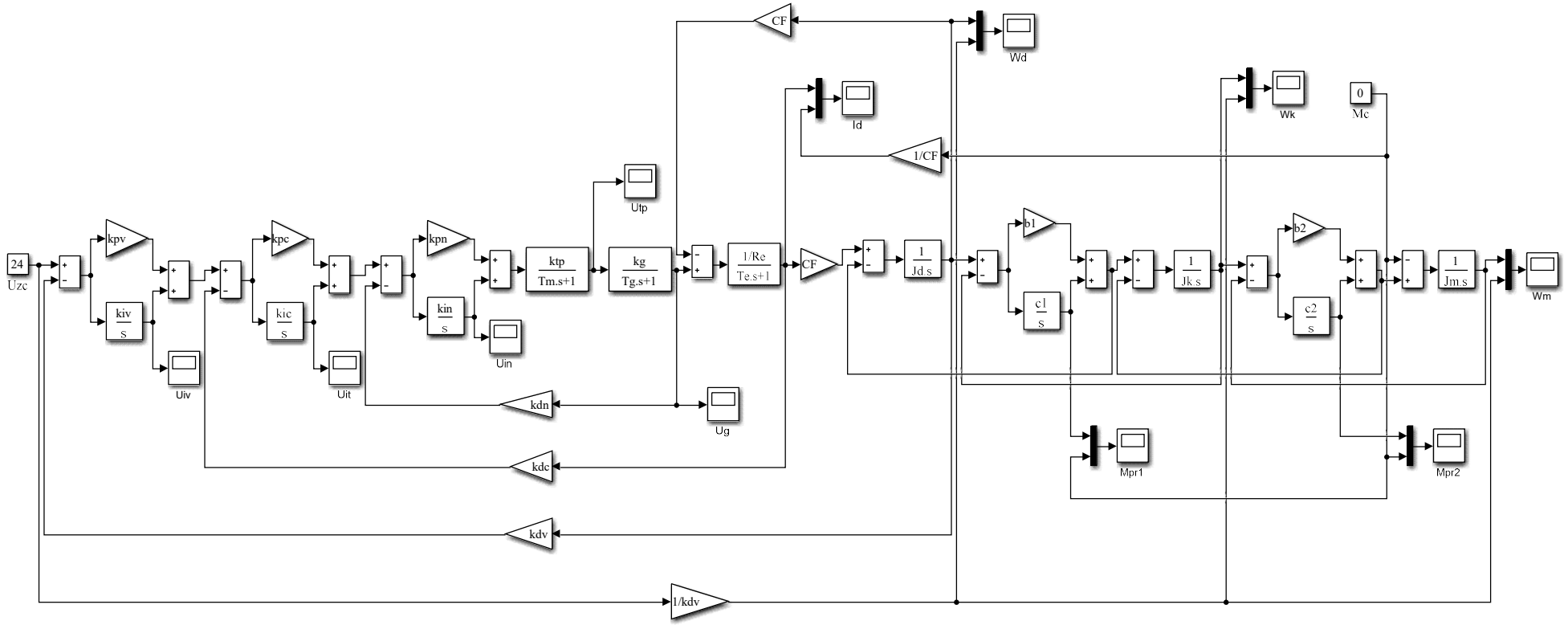


Рис. А.8. Схема моделі трьохмасової системи управління електроприводом Г-Д механізму підйому шахтної пїдїймальної установки, розроблена в Simulink ситеми MATLAB

Варіант 4

Файл вихідних даних

```
% Система управління електроприводом
% повороту стріли роторного екскаватора

% Вихідні дані
kzv=1.84;
ks=4.9;
Tg=0.5;
Tzc=0.01;
Te=0.054;
Re=0.11;
CF=2*3.58;
In=338;
Mn=CF*In;
Jd=2*8.25;
Jm=41.5;
b12=0.02;
b12=0;
c12=50;
Js=Jd+Jm;
Uz=202.3;
ksx=ks/(CF+ks*kzv);
```

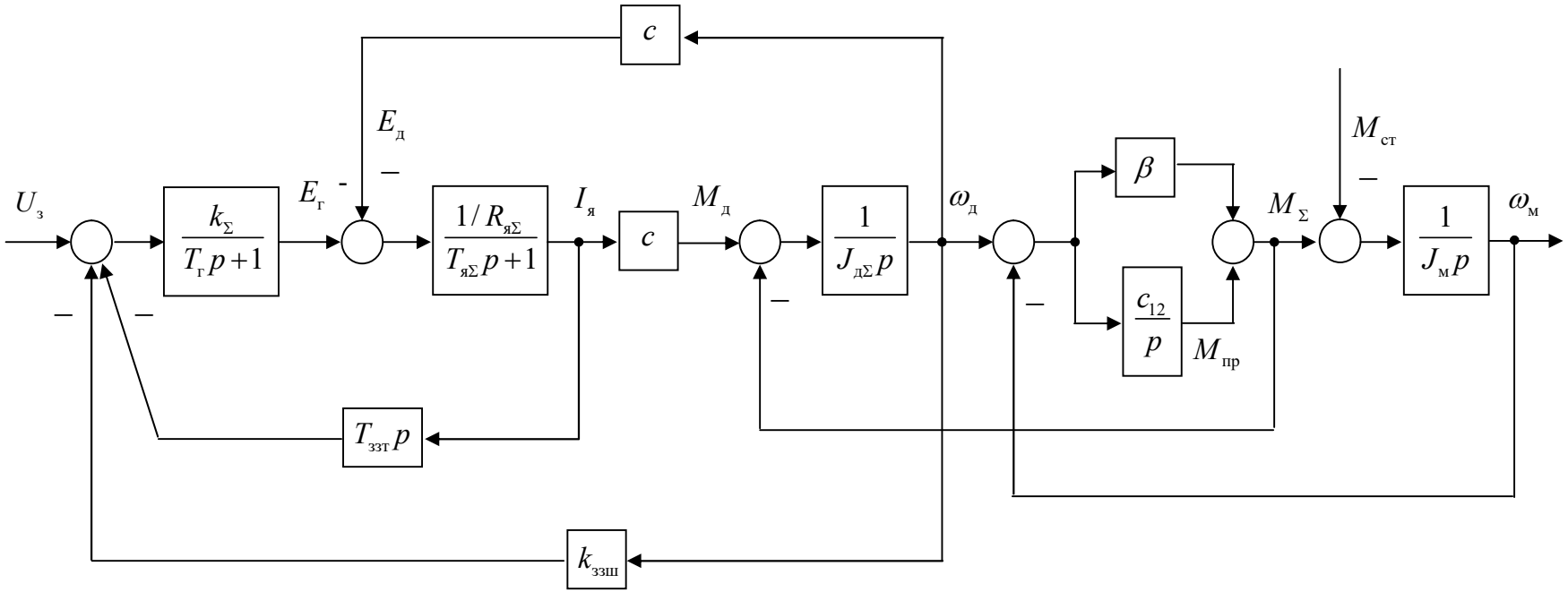


Рис. А.9. Алгоритмічна схема двомасової системи управління електроприводом Г-Д механізму повороту стріли роторного екскаватора

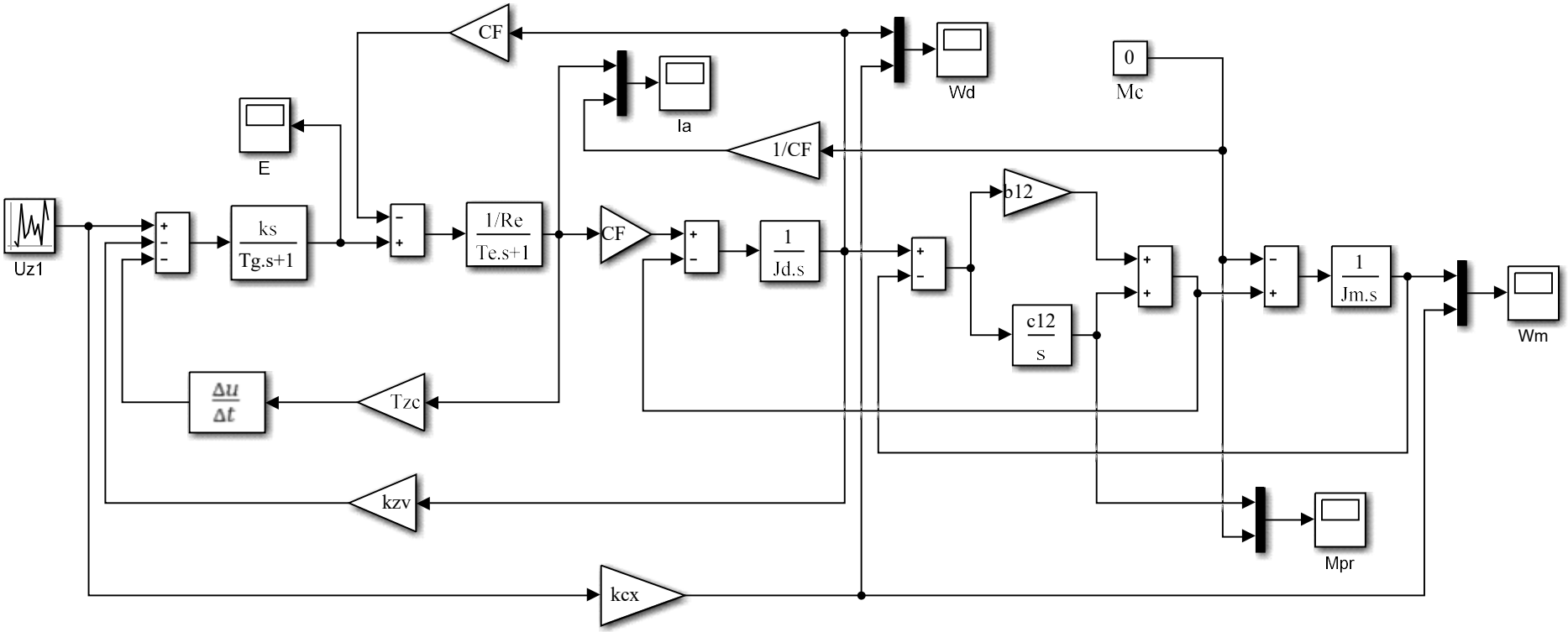


Рис. А10. Схема моделі двомасової системи управління електроприводом Г-Д механізму повороту стріли роторного екскаватора, розроблена в Simulink ситеми MATLAB

Варіант 5

Файл вихідних даних

```
clear;clc;echo on;

% Система управління електроприводом
% розвороту роторного екскаватора

% Вихідні дані
ks=3.43;
Re=0.307;
Rd=2*0.073;
Rcp=0;
Rs=Rd+Rcp;
Ld=2*0.003;
Lcp=0;
Ls=Ld+Lcp;
kzn=0.71;
CF=2*3.42;
Tzc=kzn*Ls;
Ttp=0.005;
Tg=Ttp;
Te=0.03;
kzv=kzn*CF;
In=192;
Mn=CF*In;
Jd=2*2;
Jm=39.4;
b12=0;
c12=25;
Js=Jd+Jm;
Uz=440;
ksx=ks/(CF*(1+ks*kzn));
k1=1.0;
k2=1.6;
k3=3.0;
```

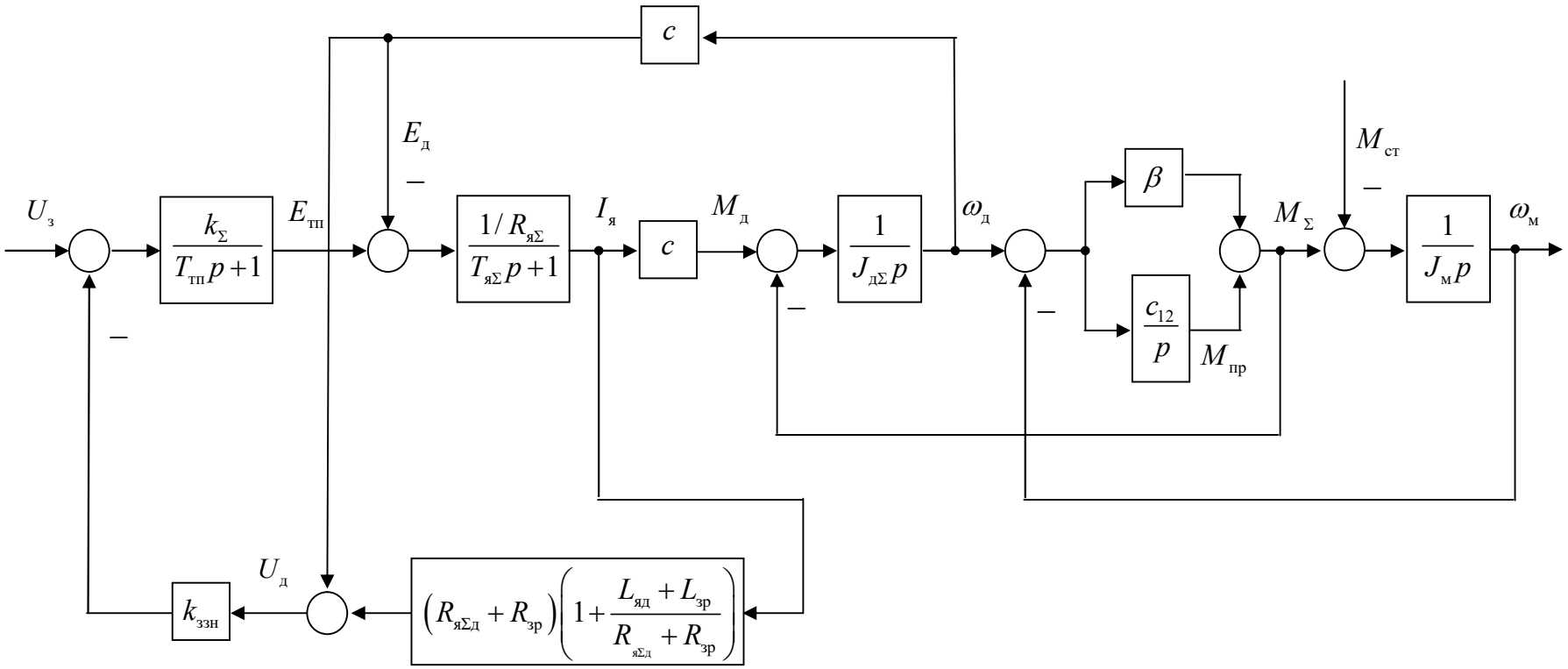


Рис. А.11. Алгоритмічна схема двомасової системи управління електроприводом ТП-Д механізму розвороту роторного екскаватора

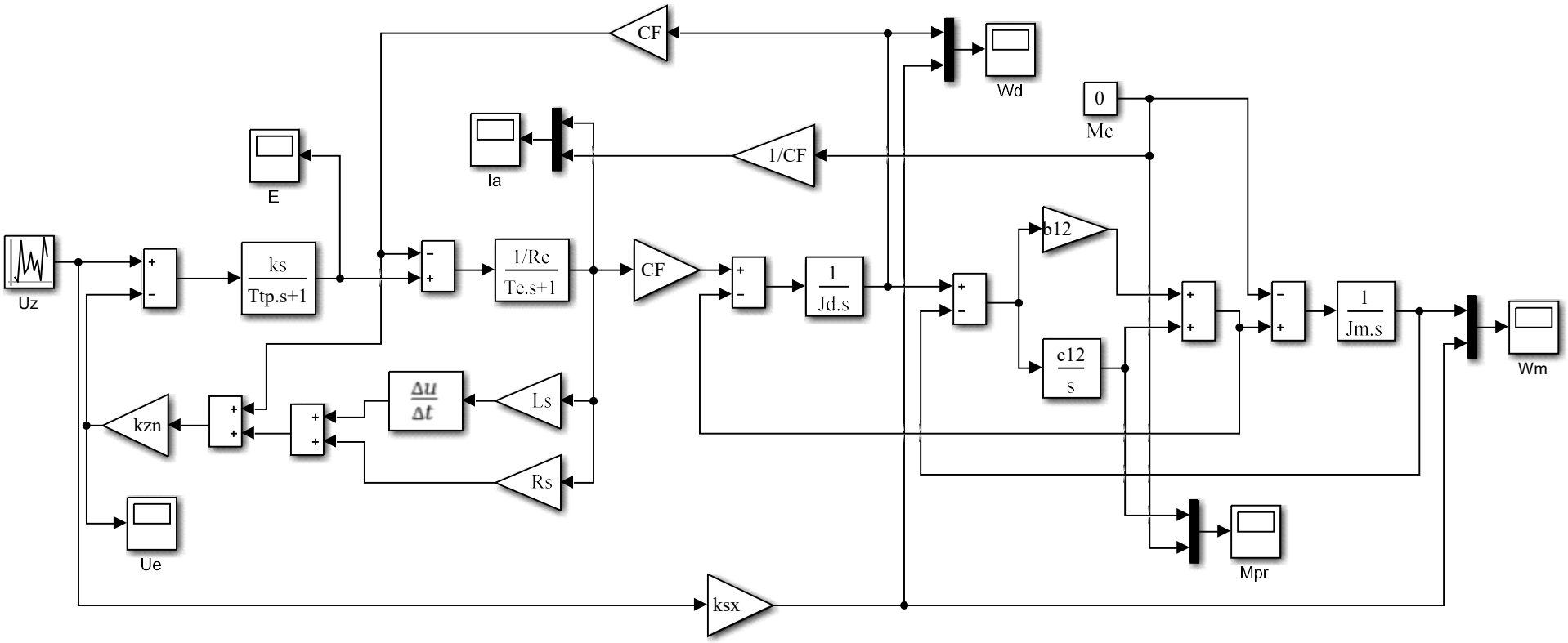


Рис. А.12. Схема моделі двомасової системи управління електроприводом ТП-Д механізму розвороту роторного екскаватора, розроблена в Simulink ситеми MATLAB

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному мережному режимі

Канюк Геннадій Іванович
Василець Тетяна Юхимівна

ТЕХНОЛОГІЇ НЕЙРОННИХ МЕРЕЖ І НЕЧІТКОГО МОДЕЛЮВАННЯ В СИСТЕМАХ УПРАВЛІННЯ

Методичні вказівки
до проведення практичних занять для здобувачів вищої освіти другого (магістерського) рівня за спеціальністю 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

У двох частинах

Частина 1

В авторській редакції

Підписано до розміщення 25.06.2025. Гарнітура Times New Roman.
Ум. друк. арк.5,94. Обсяг. 7,420. Мб. Зам. № 297/25.

Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009
Видавництво ХНУ імені В. Н. Каразіна