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

ТЕХНОЛОГІЇ НЕЙРОННИХ МЕРЕЖ І НЕЧІТКОГО МОДЕЛЮВАННЯ В СИСТЕМАХ УПРАВЛІННЯ

У двох частинах

Частина 1

Навчально-методичний посібник
для здобувачів вищої освіти другого (магістерського) рівня
за спеціальністю 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Електронний ресурс

Харків – 2025

УДК 658.5.011.56 : 658.589 (075.8)
Т 38

Рецензенти:

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

Литвин О. О. – 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 № 8 від 28 березня 2025 року)

Технології нейронних мереж і нечіткого моделювання в системах управління. У 2-х частинах. Ч. 1 : навчально-методичний посібник для здобувачів вищої освіти другого (магістерського) рівня за спеціальністю 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка [Електронний ресурс] / уклад. Г. І. Канюк, Т. Ю. Василець О. О. Варфоломієв. – Харків : ХНУ імені В. Н. Каразіна, 2025. – (PDF 173 с.)

Т 38

У першій частині навчально-методичного посібника викладені питання теорії і методи синтезу систем управління нелінійними динамічними об'єктами на основі багатошарових нейронних мереж. Значна увага приділена основним властивостям нелінійних багатошарових нейронних мереж і алгоритмам їх навчання.

Наведені структури нейромережевих систем управління і ідентифікації стану. Детально розглянуті методи ідентифікації і синтезу нейромережевих систем управління з застосуванням системи MATLAB. Наведені приклади, що ілюструють ефективність застосування багатошарових нейронних мереж в якості моделей нелінійних динамічних об'єктів і нелінійних регуляторів систем управління.

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 (075.8)

ЗМІСТ

ВСТУП

Сучасна класична теорія управління включає теорію адаптивного і оптимального управління, за допомогою якої можна розробляти оптимальні системи, тобто системи, при функціонуванні яких мінімізується або максимізується деякий вибраний наперед критерій якості.

Разом з методами синтезу управляючих пристроїв і систем сучасна теорія управління включає методи ідентифікації і оцінювання станів процесів. Різні алгоритми ідентифікації процесів дозволяють визначити структуру моделі об'єкту управління і відновити параметри цієї моделі як безпосередньо в контурі управління, так і поза ним. Ці алгоритми використовуються при синтезі не тільки звичайних, але і адаптивних систем управління.

Основою класичної теорії управління є добре розвинена теорія лінійних систем. Часто елементи цієї теорії використовуються при дослідженні і синтезі систем управління нелінійними об'єктами, проте, задовільні результати можна отримати лише в тих випадках, коли нелінійність об'єкту є неістотною або, коли об'єкт управління характеризується великими постійними часу і стійкістю в розімкненому стані. Не дивлячись на величезну кількість робіт, обмеження, що накладаються різноманітними видами нелінійностей, динамічними властивостями об'єктів, їх нестаціонарністю, а також наявність зовнішніх збурень і помилок вимірювань, не дозволяє створити єдиний підхід при побудові систем управління. Крім того, як класична, так і сучасна теорія управління грунтуються на ідеї лінеаризації систем. Проте при рішенні практичних задач такий підхід може не забезпечити необхідний критерій якості управління, оскільки модель, побудована на припущенні про лінійність системи, може не відображати її дійсних властивостей.

Останнім часом вельми багатообіцяючою альтернативою класичним методам побудови систем управління нелінійними об'єктами є штучні нейронні мережі (ШНМ). Найбільш чудовою властивістю ШНМ є їх здібність до навчання, що дозволяє отримати простіші рішення для складних задач управління. Крім того, наявність в структурі ШНМ нейронів з нелінійними функціями активації дозволяє використовувати їх для вирішення завдань управління нелінійними об'єктами, тоді як традиційні методи не забезпечують рішення подібних задач [1-16].

Необхідною умовою застосування традиційних методів оптимального і адаптивного управління є наявність великого об'єму апріорної інформації. Завдяки здібності ШНМ до самонавчання, для систем управління, побудованих на основі нейрорегуляторів, такий об'єм інформації не потрібний. Це дозволяє використовувати нейрорегулятори в умовах істотних невизначеностей.

Побудова системи управління на основі нейрорегулятора зводиться до рішення послідовно завдання ідентифікації, а потім до побудови алгоритму управління відповідно до вибраних цілей управління і критерію якості управління. Процес ідентифікації об'єкту зводиться до побудови його нейронної моделі, що реалізовується в

ході навчання мережі, на основі пред'явлення навчальних пар, якими служать вимірювані значення вхідних і відповідних вихідних змінних.

Крім нейронних мереж в задачах управління складними динамічними об’єктами в даний час застосовуються методи нової сучасної технології – нечіткого моделювання. Актуальність цієї нової технології і її перевага перед відомими класичними концепціями моделювання і управління обумовлена наступними факторами.

Перш за все – це тенденція збільшення складності математичних і формальних моделей реальних систем і процесів управління, пов'язана з бажанням підвищити їх адекватність і врахувати все більше число різних факторів, що впливають на процеси ухвалення рішень.

З одного боку, традиційні методи побудови моделей не приводять до задовільних результатів, коли початковий опис проблеми, що підлягає вирішенню, свідомо є неточним або неповним. З іншого боку, прагнення отримати всю вичерпну інформацію для побудови точної математичної моделі скільки-небудь складної реальної ситуації може привести до втрати часу і засобів, оскільки це може бути в принципі неможливо.

У подібних випадках найдоцільніше скористатися такими методами, які спеціально орієнтовані на побудову моделей, що враховують неповноту і неточність початкових даних. Саме у таких ситуаціях технологія нечіткого моделювання виявляється найбільш конструктивною, оскільки за останнє десятиліття на її основі були вирішені сотні практичних завдань управління і ухвалення рішень.

Теоретично, системи з нечіткою логікою і штучні нейронні мережі еквівалентні один одному, проте на практиці у них є свої власні достоїнства і недоліки. Дане міркування лягло в основу апарату гібридних мереж, в яких висновки робляться на основі апарату нечіткої логіки, але відповідні функції приналежності підстроюються з використанням алгоритмів навчання нейронних мереж, наприклад, алгоритму зворотного розповсюдження помилки. Такі системи не тільки використовують апріорну інформацію, але можуть набувати нових знань і для користувача є логічно прозорими.

У навчально-методичному посібнику розглянуті теоретичні аспекти складових подібних мереж, а саме, основи теорії штучних нейронних мереж, апарат нечіткої логіки і гібридних мереж стосовно завдань управління.

Найважливішою особливістю життєздатності тієї або іншої теоретичної концепції є її реалізація і підтримка у відповідних програмних інструментах. Поява і успішний розвиток програмних засобів, які спеціально орієнтовані на рішення задач, пов’язаних з застосуванням нейронних мереж і нечіткого моделювання об'єктивно свідчать на користь того, що нейромережеві технології, теорія нечітких множин і нечітка логіка можуть і повинні бути ефективно використані для вирішення практичних завдань управління. При цьому одним з найбільш ефективних засобів, в якому реалізовані розглянуті технології, є система MATLAB. Тому в підручнику особлива увага приділена програмній реалізації моделей вказаних підходів інструментальними засобами MATLAB.

ОСНОВНІ ПОНЯТТЯ І СТРУКТУРНІ СХЕМИ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

1.1 Області застосування штучних нейронних мереж

Штучні нейронні мережі (ШНМ) здобули популярність завдяки своїй здатності ефективно вирішувати складні завдання в різних сферах, де традиційні алгоритми не дають задовільних результатів. Завдяки можливості навчатися на великому обсязі даних та знаходити складні закономірності, нейронні мережі застосовуються у багатьох галузях науки та техніки. Далі наведено кілька ключових областей застосування ШНМ:

1. Комп'ютерний зір

Комп'ютерний зір - це галузь, що займається автоматичним аналізом і розпізнаванням зображень та відео. ШНМ, зокрема конволюційні нейронні мережі (CNN), є основою для багатьох сучасних систем комп'ютерного зору.

- *Розпізнавання образів:* ШНМ можуть використовуватися для класифікації об'єктів на зображеннях, таких як розпізнавання осіб, автомобілів, тварин.
- *Обробка зображень:* Покращення якості зображень, зменшення шуму, виявлення країв та інших характеристик.
- *Виявлення об'єктів*: Використовуються для локалізації об'єктів на зображеннях або відео, наприклад, для виявлення пішоходів у системах безпеки.

2. Обробка природної мови (NLP)

ШНМ використовуються для вирішення різноманітних задач в обробці тексту та мови, зокрема для:

- *Переклад тексту:* Моделі на основі рекурентних нейронних мереж (RNN) та трансформерів (наприклад, GPT, BERT) активно використовуються для автоматичного перекладу між мовами.
- *Аналіз тональності тексту*: Визначення емоційного забарвлення тексту (позитивний чи негативний відгук).
- *Чат-боти та системи підтримки користувачів*: Використання нейронних мереж для автоматичних відповідей на запитання користувачів у реальному часі.
- Сумаризація тексту: Автоматичне створення коротких резюме великих текстів або статей.

3. Медицина

ШНМ мають важливе застосування в медичній діагностиці та обробці медичних зображень:

- *Розпізнавання зображень для діагностики*: Використання CNN для автоматичної обробки медичних зображень, таких як рентгенівські знімки або МРТ, для виявлення захворювань (наприклад, раку).

- *Передбачення захворювань:* Моделі для прогнозування розвитку хвороб на основі медичних даних, таких як анамнез пацієнта, лабораторні аналізи тощо.
- Індивідуалізовані схеми лікування: Використання даних пацієнта для визначення найбільш ефективних методів лікування.

4. Фінансовий сектор

Нейронні мережі активно використовуються для автоматизації та оптимізації різних процесів у фінансовому секторі:

- *Прогнозування ринкових тенденцій:* ШНМ використовуються для аналізу фінансових даних та прогнозування цін на акції, валютні курси, ціни на товари.
- *Кредитний рейтинг:* Оцінка кредитоспроможності клієнтів на основі історії фінансових операцій та інших характеристик.
- Виявлення шахрайства: Використання нейронних мереж для виявлення аномальних транзакцій та виявлення шахрайських схем у фінансових операціях.

5. Автономні системи

ШНМ мають важливу роль у розвитку **автономних транспортних засобів**:

- *Автономні автомобілі*: Використання нейронних мереж для розпізнавання об'єктів на дорозі, навігації та ухвалення рішень в реальному часі.
- *Безпілотні літальні апарати (дрони):* Нейронні мережі використовуються для обробки зображень та даних від сенсорів, що дозволяє дронам виконувати різні завдання без участі людини.

6. Інтернет речей (IoT)

ШНМ використовуються для обробки великих масивів даних, що генеруються пристроями в Інтернеті речей:

- *Аналіз даних з сенсорів:* Використання нейронних мереж для обробки та аналізу даних, що надходять від сенсорів (температура, вологість, рух) у розумних будинках або промислових об'єктах.
- *Прогнозування технічного стану:* Моделі для прогнозування виходу з ладу обладнання на основі даних, зібраних від сенсорів.

7. Робототехніка

Нейронні мережі допомагають робити роботи більш гнучкими та адаптивними до різних умов:

- *Навчання безпосередньо з досвіду:* За допомогою глибокого навчання роботи можуть навчатися виконувати завдання, аналізуючи відео або сенсорні дані, замість того щоб слідувати жорстко запрограмованим алгоритмам.
- *Розпізнавання об'єктів і маніпуляції з ними:* Використання нейронних мереж для навчання роботів маніпулювати об'єктами, визначати їх форму, орієнтацію та положення.

8. Маркетинг та реклама

ШНМ широко використовуються для покращення маркетингових стратегій:

- *Рекомендаційні системи:* Алгоритми, засновані на нейронних мережах, використовуються для персоналізації контенту та рекомендацій (наприклад, рекомендації товарів на основі попередніх покупок).

- *Аналіз споживчого попиту:* Прогнозування потреб клієнтів на основі їх поведінки та взаємодії з товарами та послугами.

9. Енергетика

ШНМ застосовуються для оптимізації енергетичних систем і зменшення споживання енергії:

- *Прогнозування споживання енергії:* Моделі для передбачення споживання енергії в різних регіонах або в окремих будинках для оптимізації розподілу енергетичних ресурсів.
- *Управління відновлювальними джерелами енергії:* Використання ШНМ для прогнозування та управління потоками енергії з сонячних панелей, вітряків тощо.

Таким чином, штучні нейронні мережі знайшли широке застосування в різноманітних галузях, від медицини до фінансів та робототехніки. Завдяки здатності навчатися на великих обсягах даних, нейронні мережі здатні вирішувати складні завдання, що зумовлює їхній величезний потенціал у майбутньому. З розвитком технологій та зростанням кількості даних нейронні мережі стають все більш важливими інструментами для автоматизації та покращення різних процесів у сучасному світі.

1.2 Застосування нейронних мереж в системах управління

Нейронні мережі набули популярності в системах управління завдяки своїй здатності до навчання та адаптації в реальному часі, а також здатності вирішувати складні задачі, які традиційно важко формалізувати або моделювати за допомогою класичних методів. У сучасних системах управління нейронні мережі використовуються для *регулювання процесів, оптимізації керування* та *прогнозування динамічних змін* в об'єктах управління.

1. Адаптивні системи управління

Нейронні мережі активно використовуються для створення адаптивних систем управління, які здатні автоматично налаштовувати параметри керування на основі змінюваних умов навколишнього середовища або параметрів об'єкта управління. Це особливо важливо для систем, де точне математичне моделювання є складним або навіть неможливим.

- *Адаптація до змін:* Нейронні мережі дозволяють системам автоматично адаптувати свої стратегії управління, враховуючи зміни в умовах або характеристиках об'єкта.
- *Корекція управління в реальному часі:* Використовуючи глибинне навчання, мережі можуть коригувати управлінські сигнали в реальному часі на основі отриманих даних.

2. Контроль складних і нелінійних систем

Багато реальних систем мають нелінійні властивості, що робить їх важкими для контролю за допомогою традиційних методів, таких як класичні регулятори. Нейронні мережі, зокрема многошарові перцептрони (MLP), ефективно справляють-

ся з управлінням такими складними системами.

- *Нелінійний контроль:* Нейронні мережі здатні моделювати складні нелінійні взаємозв'язки між входами та виходами системи, що дозволяє здійснювати точний контроль навіть за нелінійних і нестабільних умов.
- *Моделювання та прогнозування:* Нейронні мережі можуть використовуватись для створення моделей системи, що дозволяє прогнозувати її поведінку та своєчасно коригувати управляючі впливи.

3. Нейроконтролери

Одним з важливих напрямків є використання нейронних мереж для створення нейроконтролерів. Нейроконтролер використовує нейронні мережі для прийняття рішень щодо управлінських впливів на об'єкт.

- *Система управління на основі нейронних мереж:* Використовує вхідні сигнали з об'єкта управління та відповідні виходи для регулювання процесу. Це дозволяє автоматично налаштовувати параметри керування без необхідності детального програмування.
- *Без необхідності математичної моделі:* На відміну від традиційних методів, нейронні мережі можуть працювати без детальних моделей об'єкта управління. Мережа може навчатися на основі прикладів і вдосконалювати стратегію управління.

4. Оптимізація процесів управління

Нейронні мережі використовуються для оптимізації процесів управління, таких як оптимальне розподілення ресурсів, пошук найкращих стратегій управління в умовах обмежених ресурсів або при високому рівні невизначеності.

- *Оптимізація керування в реальному часі*: Наприклад, в системах енергозабезпечення нейронні мережі можуть допомогти оптимізувати розподіл електричної енергії між споживачами.
- *Інтелектуальне управління в логістиці:* В управлінні транспортними потоками, складуванням, роботами на виробництві нейронні мережі можуть знаходити оптимальні маршрути, планувати завдання та навіть передбачати майбутні зміни для запобігання збоям.

5. Моделювання та імітація процесів управління

Нейронні мережі використовуються для створення моделей об'єктів управління, що дозволяє точніше прогнозувати їхню поведінку в різних умовах.

- *Моделювання фізичних, хімічних та біологічних процесів:* Нейронні мережі допомагають моделювати складні системи, де математичні моделі можуть бути важкими або недоступними, наприклад, в хімічних реакторах, біотехнологічних процесах або екологічних системах.
- *Імітація складних виробничих процесів:* Використання ШНМ для створення моделей складних виробничих систем дозволяє проводити симуляції і оптимізувати процеси без втручання в реальні системи.

6. Системи управління роботами

Нейронні мережі є важливим інструментом для автономного управління робо-

тами. Вони дозволяють роботам взаємодіяти з навколишнім середовищем та виконувати складні завдання, такі як маніпуляція об'єктами, орієнтація в просторі, а також адаптація до змін у середовищі.

- *Навчання в реальному часі:* Роботи можуть навчатися на основі даних, отриманих від сенсорів, та вдосконалювати свої стратегії управління в процесі взаємодії з оточенням.
- *Розпізнавання об'єктів і маніпуляції:* Нейронні мережі допомагають роботам точно визначати положення та орієнтацію об'єктів, а також оптимально маніпулювати ними.

7. Управління в складних умовах

Управління в умовах високої невизначеності, шуму або обмеженого доступу до даних є одним із завдань, де ШНМ демонструють свої переваги. Нейронні мережі використовуються для:

- *Прогнозування та планування в умовах невизначеності*, де традиційні методи мають обмеження.
- *Навчання за допомогою зворотного зв'язку*, що дозволяє системі адаптуватися до нових умов, отримуючи інформацію про поточний стан.

З вищесказаного можна зробити висновок, що нейронні мережі в системах управління дозволяють вирішувати складні задачі, пов'язані з адаптацією до змінюваних умов, оптимізацією процесів, управлінням складними нелінійними системами та прогнозуванням. Вони відкривають нові можливості для адаптивного управління, безпілотних систем і розумних технологій, забезпечуючи ефективність, гнучкість і надійність сучасних автоматизованих систем.

Багато завдань, пов’язаних з нейронними мережами в даний час успішно вирішуються засобами пакету прикладних програм (ППП) Deep Learning Toolbox системи MATLAB, який забезпечує досить потужний інструментальні можливості по створенню нейронних мереж і їх застосуванн. До версії MATLAB R2017a цей пакет мав назву Neural Network Toolbox [1,2, 17-19].

1.3 Нейромережевий підхід до управління

Більшість реальних систем характеризуються нелінійними залежностями, складними динамічними властивостями, наявністю неконтрольованих шумів і завад, що перешкоджають реалізації традиційних стратегій управління, оскільки, як сучасна (зокрема теорія адаптивного і оптимального управління), так і класична теорія управління в значній мірі базуються на ідеї лінеаризації систем.

Для практичного застосування нейромережевого підходу необхідна, перш за все, розробка математичної моделі системи. Проте математичне моделювання, що реалізовується на основі припущення про лінійність системи, може не відображати її дійсних фізичних властивостей. Навіть якщо вдається побудувати складні математичні моделі, що точно відображають фізичні співвідношення між входом і виходом системи, вони можуть виявитися даремними для цілей управління. Практично прийнятними можуть бути тільки моделі з низькою чутливістю по параметрах, що для нелінійних систем украй складно забезпечити.

Використання статистичних моделей дозволяє виконати оцінку параметрів процесу за допомогою аналізу частотних характеристик методами класичної теорії управління. Проте ефективність застосування даного методу може не забезпечити достатньої точності для цілей управління. Причина цього недоліку полягає в спробі пристосувати лінійні моделі до складних нелінійних систем. Крім того, дані методи засновані на моделях «чорного ящика» і їх параметри, часто, не мають фізичного сенсу. З цих причин дані методи не знаходять широкого застосування в практиці управління.

Використання адаптивних методів управління припускає наявність математичної моделі, заснованої на фізичних явищах, і методів оцінки параметрів цієї моделі. Потім будується закон управління, направлений на досягнення мети управління, а математична модель розглядається як реальна система. Даний підхід також ґрунтується на теорії лінійних систем. При виникненні змін в об'єкті управління або в зовнішніх умовах необхідно перебудовувати параметри моделі і задавати новий закон управління. Це вимагає втручання оператора для перевірки адекватності моделі і реальної системи.

Таким чином, можна зробити висновок: для практичного застосування алгоритмів управління необхідно, щоб вони були адаптивними, стійкими, нелінійними, а також простими для реалізації і розуміння. Саме з цих причин в даний час широке застосування в завданнях управління отримали штучні нейронні мережі , що володіють вказаними вище властивостями.

Нейромережеве управління засноване на застосуванні повністю визначених ШНМ для вироблення необхідних сигналів управління. Високий інтерес до даного напряму в теорії управління визначається наступними чинниками.

1. Адаптивність

Нейронні мережі здатні адаптувати свої параметри (ваги) на основі вхідних даних. У системах управління це дозволяє мережам автоматично налаштовувати

стратегії управління в умовах змінних або непередбачуваних факторів без потреби в явному математичному моделюванні об'єкта управління.

- Приклад: У адаптивних системах управління нейронні мережі можуть коригувати параметри керування в реальному часі на основі зміни зовнішніх умов або характеристики об'єкта управління.

2. Нелінійність

Багато реальних систем управління мають нелінійні характеристики, які складно описати класичними лінійними моделями. Нейронні мережі, завдяки своїм нелінійним функціям активації (як-от ReLU, sigmoid або tanh), можуть ефективно моделювати і вирішувати задачі, що включають нелінійні взаємозв'язки між вхідними та вихідними величинами.

- Приклад: Нейронні мережі можуть бути використані для управління складними динамічними системами, такими як робототехнічні маніпулятори або електричні двигуни, де динаміка системи є нелінійною.

3. Можливість обробки складних, великих обсягів даних

Нейронні мережі мають здатність ефективно обробляти великі масиви даних, що робить їх ідеальними для використання в системах управління, де необхідно обробляти численні сигнали, параметри або сенсорні дані.

- Приклад: У системах моніторингу та діагностики нейронні мережі можуть обробляти великий обсяг даних від різних датчиків, таких як температурні, тискові, або швидкісні параметри, і приймати рішення для управління в реальному часі.

4. Імплементація в реальному часі

Нейронні мережі можуть працювати з вхідними даними в реальному часі та адаптувати свої стратегії управління або прогнозування на основі нової інформації. Це є ключовою властивістю для їх застосування в системах, де потрібно оперативно реагувати на зміни в середовищі.

- Приклад: У системах автопілотів для транспортних засобів нейронні мережі можуть обробляти інформацію з камер, лідарів і інших сенсорів в реальному часі, здійснюючи корекцію траєкторії руху або прийняття рішень щодо безпеки.

5. Паралельність обчислень

Нейронні мережі здатні виконувати обчислення паралельно, що дозволяє значно зменшити час обробки даних. У системах управління, де швидкість обчислень є важливою, це є великою перевагою.

- Приклад: В системах реального часу для контролю роботизованих систем або автоматизованих виробничих ліній нейронні мережі можуть обробляти дані від численних сенсорів та швидко реагувати на зміни у процесі.

6. Робота в умовах невизначеності та шуму

Нейронні мережі можуть бути ефективними в умовах неповних даних, шуму або невизначеності, що є характерним для багатьох реальних систем управління.

Вони здатні навчатися на основі неідеальних даних і все одно приймати обґрунтовані рішення.

- Приклад: У системах управління в умовах нестабільності (наприклад, управління в складних кліматичних умовах або під час неочікуваних змін у зовнішньому середовищі) нейронні мережі можуть допомогти забезпечити надійне управління, навіть якщо дані є зашумленими або неповними.

7. Моделювання складних залежностей і взаємозв'язків

Нейронні мережі здатні вивчати і моделювати **складні взаємозв'язки** між різними змінними в системах, що робить їх корисними для задач прогнозування та оптимізації в системах управління.

- Приклад: У системах енергозабезпечення нейронні мережі можуть використовуватися для прогнозування навантаження на мережу, оптимізації розподілу енергії та зменшення втрат, враховуючи складні залежності між споживанням енергії, погодними умовами та іншими змінними.

8. Виявлення аномалій і діагностика

Нейронні мережі можуть використовуватися для виявлення аномалій в системах управління, що дозволяє здійснювати діагностику та швидке виявлення можливих збоїв або несправностей.

- Приклад: У системах контролю якості на виробництві нейронні мережі можуть виявляти дефекти продукції на основі аналізу вхідних даних від сенсорів або камер, що дозволяє своєчасно коригувати процеси та запобігати дефектам.

9. Оптимізація та планування

Нейронні мережі можуть бути використані для оптимізації процесів управління в умовах обмежених ресурсів або необхідності прийняття рішень на основі великих обсягів даних, що дозволяє досягати ефективності в управлінні складними системами.

- Приклад: У логістичних системах нейронні мережі можуть оптимізувати маршрути доставки, планувати розподіл товарів або ресурсів, знижуючи витрати та підвищуючи ефективність роботи.

З вищесказаного можна зробити висновок, що нейронні мережі є потужним інструментом для систем управління завдяки їх здатності адаптуватися до змін, працювати з великими обсягами даних, ефективно моделювати нелінійні взаємозв'язки та адаптувати свої стратегії в реальному часі. Ці властивості роблять їх надзвичайно корисними в складних і динамічних системах управління, де потрібна швидка реакція на зміни умов і точність прийняття рішень.

Історія становлення штучних нейронних мереж як наукового напряму починається з робіт Мак-Каллока, Пітерса [20] і Розенблата [21], що породили свого часу великі надії у кібернетиків, але незабаром незаслужено забутих. Перші спроби використовувати штучні нейронні мережі для управління динамічними об'єктами в «біонічних системах» були зроблені ще на початку 70-х років XX-го століття, наприклад [22,23]. Учені прийшли до висновку, що нейронні структури в біологічних управля-

ючих системах працюють в n-мірному фазовому просторі, розбитому на деяке число областей по критерію вигляду

$$Q = \min\{\max F(\Psi_i(t), \Psi_i^{(1)}(t), \Psi_i^{(2)}(t), ..., \Psi_i^{(m)}(t))\}, \quad i = \overline{1,n},$$

де $\Psi_i(t) = x_i(t) - g_i(t)$ – динамічні помилки регулювання;

$\Psi_i^{(j)}(t)$ – j-а похідна від динамічної помилки;

$x_i(t)$ – поточне значення регульованої величини;

$g_i(t)$ – її задане значення;

$F(\cdot)$ – вибираний функціонал, що характеризує вимоги до системи регулювання.

Нейронна регулююча система виконує функцію навчаного або самонавчаного класифікатора в цьому просторі. Термін «класифікатор» тут означає, що мережа виконує оцінювання змінних фазового простору системи щодо околиці особливих точок (або областей) цього простору. Фіксуємо попадання зображуючої точки системи в одну з пронумерованих областей фазового простору викликає на виході класифікатора коректуючі сигнали $\xi_j \Psi_i^{(j)}(t)$, де ξ_j – змінні, залежно від положення зображаючої точки усередині цієї області, параметри. Таким чином, нейронна регулююча структура розглядається як нейронний класифікатор n-мірного фазового простору, а сам процес регулювання зводиться до необхідної зміни вектора стану в каналах зворотного зв'язку. Подібна ідея «біонічного» управління самими ученими називалася гіпотетичною, але з «...достньо великим коефіцієнтом правдоподібності». Як приклад біологічних систем управління, що вписуються в запропоновану концепцію побудови нейромережевої системи з класифікатором фазового простору приводилася окодвигунна система тварин. Відмітимо, що у той час ще не були відомі методи навчання багатошарових нейронних мереж.

Термін **«нейроуправління»** вперше з'явився в роботах Вербоса в 1974 році [24]. Проте тільки в кінці 80-х років почалося реальне застосування навчаних багатошарових нейронних мереж в системах управління. Відродження інтересу до нейронних мереж в ці роки багато в чому пов'язане з ім'ям Д.Румельхарда [25, 26], перевідкрившого алгоритм навчання нейронних мереж, відомий як алгоритм зворотного розповсюдження помилки. Вирішальну роль у впровадженні штучних нейронних мереж в сферу управлінських завдань зіграли роботи Наренди із співавторами, зокрема [27,28].

Нейронні мережі широко використовуються для управління динамічними об'єктами. У роботі [29].визначені проблеми синтезу нейромережевих систем управління динамічними об'єктами. Ці проблеми торкаються:

- синтезу структур нейромережевих систем управління;
- обмежень на швидкість настройки параметрів мережі;

- модифікації алгоритмів настройки, що забезпечують малі траекторні помилки при обмеженнях на значення вагових коефіцієнтів синаптичних зв'язків нейронів;
- модифікації управління, що гарантує грубість в умовах неконтрольованих збурень.

Підходи до вирішення перерахованих проблем відображені в літературі по застосуванню нейронних мереж в завданнях управління. У 1992 р. [30], була розроблена процедура синтезу системи управління нелінійними об'єктами із застосуванням RBF-мережі і достатньо детально розглянуті питання синтезу її архітектури і алгоритму настройки. У ряді інших робіт і в пізніший час були запропоновані методи синтезу нейромережевих систем управління складними технічними об'єктами [31-38]. В роботах [39-50] розглядаються теоретичні аспекти застосування нейронних мереж у сучасних системах автоматичного керування, алгоритми навчання та практичні застосування нейронних мереж для вирішення задач регулювання та управління.

Таким чином, нейронні мережі в системах управління дозволяють вирішувати складні задачі, пов'язані з адаптацією до змінюваних умов, оптимізацією процесів, управлінням складними нелінійними системами та прогнозуванням. Вони відкривають нові можливості для адаптивного управління, безпілотних систем і розумних технологій, забезпечуючи ефективність, гнучкість і надійність сучасних автоматизованих систем.

1.4 Біологічні основи функціонування нейрона

Штучні нейронні мережі виникли на основі знань про функціонування нервової системи живих істот. Вони є спробою використання процесів, що відбуваються в нервових системах, для вироблення нових технологічних рішень.

Нервова клітина, скорочено звана нейроном, є основним елементом нервової системи. Вивчення механізмів функціонування окремих нейронів і їх взаємодії принципово важливе для пізнання процесів пошуку, передачі і обробки інформації, що протікають в нервовій системі. З цієї точки зору представляється необхідним побудувати і вивчити модель біологічного нейрона.

Як і у будь-якої іншої клітки, у нейрона є тіло із стандартним набором органелл, зване сомою, усередині якого розташовується ядро. З соми нейрона виходять численні відростки, що грають ключову роль в його взаємодії з іншими нервовими клітинами. Можна виділити два типи відростків: численні тонкі, густі дендрити, що гілкуються, і товстий аксон, що розщеплюється на кінці (рис.1.1).

Вхідні сигнали поступають в клітку через синапси, тоді як вихідний сигнал відводиться аксоном через його численні нервові закінчення, звані колатералами. Колатерали контактують з сомою і дендритами інших нейронів, утворюючи чергові синапси. Очевидно, що синапси, що підключають до клітки виходи інших нейронів, можуть знаходитися як на дендритах, так і безпосередньо на тілі клітки.

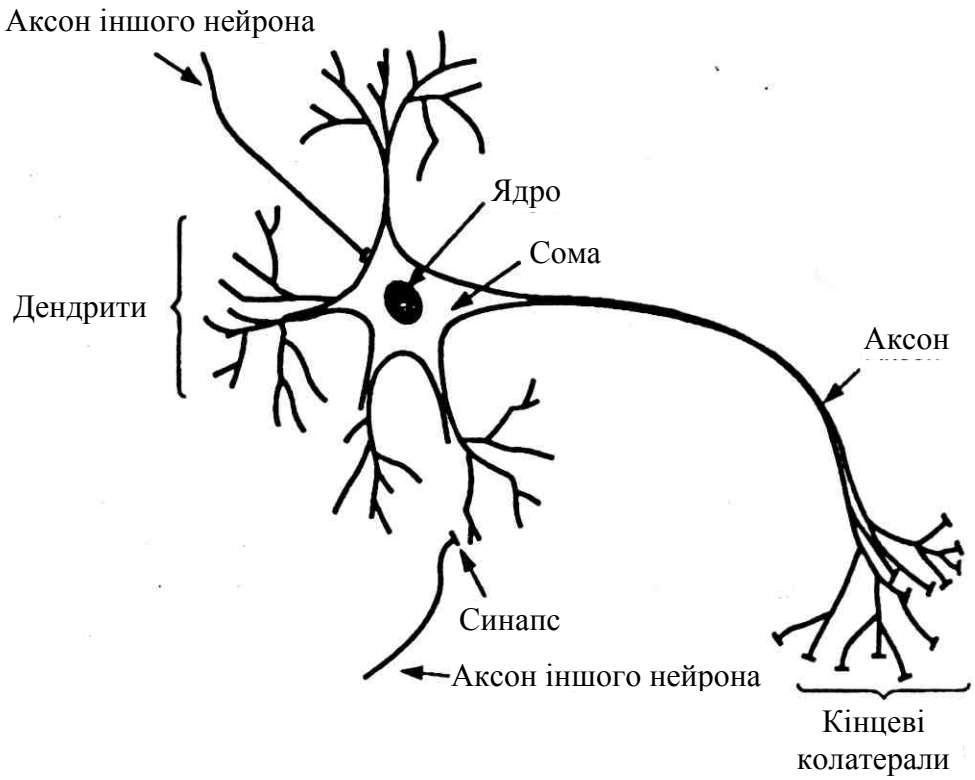


Рис.1.1. Спрощена структура біологічної нервової клітини

Передача сигналів усередині нервової системи – це дуже складний електрохімічний процес. З великим спрощенням можна вважати, що передача нервового імпульсу між двома клітками заснована на виділенні особливих хімічних субстанцій, званих нейромедиаторами, які формуються під впливом подразників, що поступають від синапсів. Ці субстанції впливають на клітинну мембрану, викликаючи зміну її енергетичного потенціалу, причому величина цієї зміни пропорційна кількості нейромедиатора, що потрапляє на мембрану.

Синапси відрізняються один від одного розмірами і можливостями концентрації нейромедиатора поблизу своєї оболонки. З цієї причини імпульси однакової величини, що поступають на входи нервової клітини через різні синапси, можуть збуджувати її різною мірою. Мірою збудження клітки вважається рівень поляризації її мембрани, залежний від сумарної кількості нейромедиатора, виділеного на всіх синапсах.

Із сказаного виходить, що кожному входу клітки можна зіставити чисельні коефіцієнти (ваги), пропорційні кількості нейромедиатора, що одноразово виділяється на відповідному синапсі. У математичній моделі нейрона вхідні сигнали повинні умножатися на ці коефіцієнти для того, щоб коректно враховувати вплив кожного сигналу на стан нервової клітини. Синапсичні ваги повинні бути натуральними числами, що приймають як позитивні, так і негативні значення. У першому випадку синапс надає збудливу, а в другому – гальмуючу дію, що перешкоджає збудженню клітки іншими сигналами. Таким чином, дія збудливого синапсу може моделювати-

ся позитивним значенням синаптичної ваги, а дія гальмуючого синапсу – негативним значенням.

В результаті надходження вхідних імпульсів на конкретні синапси і вивільнення відповідних кількостей нейромедиатора відбувається певне електричне збудження нервової клітини. Якщо відхилення від стану електричної рівноваги невелике або якщо баланс збуджень і гальмувань є негативним, клітка самостійно повертається в початковий стан, і на її виході які-небудь зміни не реєструються. В цьому випадку вважається, що рівень збудження клітки був нижчий за поріг її спрацьовування. Якщо ж сума збуджень і гальмувань перевищила поріг активації клітки, значення вихідного сигналу починає лавиноподібно наростати, приймаючи характерний вид нервового імпульсу (рис.1.2), що пересилається аксоном на інші нейрони, підключені до даної клітки. Величина цього сигналу не залежить від ступеня перевищення порогу. Клітка діє за принципом "все або нічого". Після виконання своєї функції нейромедиатор видаляється. Механізм видалення полягає або у всмоктуванні цієї субстанції кліткою, або в її розкладанні, або у видаленні за межі синапсу.

Одночасно з генерацією нервового імпульсу в клітці запускається процес рефракції. Він проявляється як стрімке зростання порогу активації клітки до значення "плюс нескінченність", внаслідок чого відразу після генерації імпульсу нейрон втрачає здатність виробляти черговий сигнал навіть при сильному збудженні. Такий стан зберігається протягом часу Δt_r, званого періодом абсолютної рефракції. Після закінчення цього терміну наступає період відносної рефракції Δt_w, за який поріг спрацьовування повертається до первинного значення. В цей час клітку можна активувати, але тільки з прикладенням більш сильних збуджень. У природних процесах виконується відношення $\Delta t_w >> \Delta t_r$.

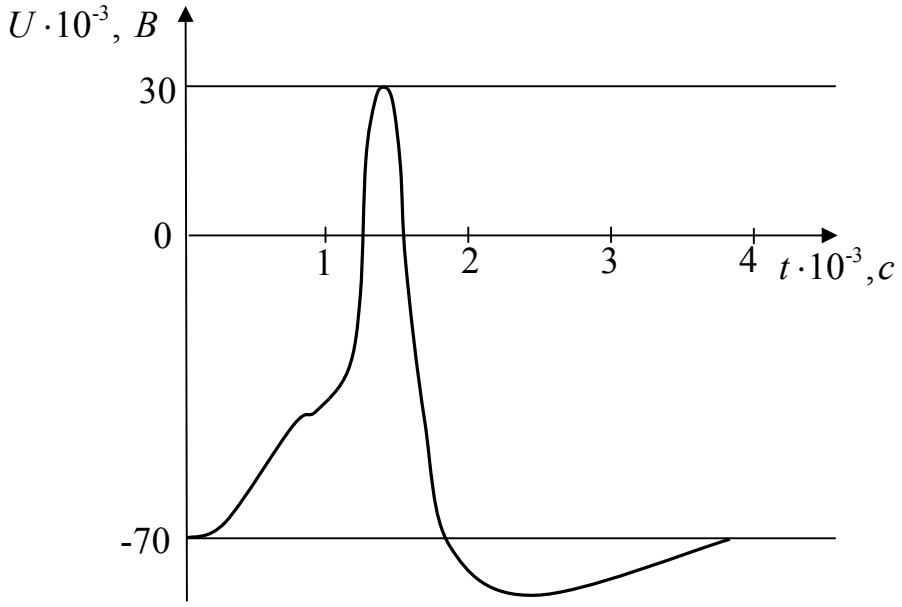


Рис.1.2. Типова форма нервового імпульсу

Кількість нервових клітин, що взаємодіють одна з одною, надзвичайно велика. Вважається, що людський мозок містить біля 10^{11} нейронів, кожний з яких виконує відносно примітивні функції підсумовування вагових коефіцієнтів вхідних сигналів і порівняння отриманої суми з пороговим значенням. Кожен нейрон має свої ваги і своє порогове значення. Вони визначаються місцезнаходженням нейрона і вирішуваною ним задачею і можуть інтерпретуватися аналогічно вмісту локальної пам'яті процесора.

Величезна кількість нейронів і міжнейронних зв'язків (до 1000 входів в кожен нейрон) призводить до того, що помилка в спрацьовуванні окремого нейрона залишається непомітною в загальній масі взаємодіючих кліток. Нейронна мережа проявляє високу стійкість до завад – це "стабільна" мережа, в якій окремі збої не роблять істотного впливу на результати її функціонування. Така головна відмінність нейронних систем від звичайних електронних систем, створених людиною. Слід підкреслити, що жодна сучасна технологія не дозволяє побудувати штучну нейронну мережу, близьку по масштабах до нейронної мережі мозку. Проте вивчення і копіювання біологічних нервових систем дозволяють сподіватися на створення нового покоління електронних пристроїв, що мають аналогічні характеристики.

Інша важлива особливість нервових систем – висока швидкість їх функціонування, не дивлячись на відносно тривалий цикл спрацьовування кожної окремої клітки, вимірюваний в мілісекундах і показаний на рис.1.2. Вона досягається завдяки паралельній обробці інформації в мозку величезною кількістю нейронів, сполучених численними міжнейронними зв'язками. Такі операції, як розпізнавання образів і звуків або ухвалення рішень, виконуються людським мозком за проміжки часу, вимірювані мілісекундами. Досягнення такого результату при використанні напівпровідникової технології все ще виходить за межі сучасних технічних можливостей, хоча цикл спрацьовування окремих виконавчих елементів є достатньо коротким і має порядок $10^{-8} c$. Якщо вдасться, узявши за зразок нервову систему, створити пристрій з високим ступенем паралельності виконання незалежних операцій, то швидкість його функціонування може бути істотно збільшена і наближена до рівня, спостережуваного в процесах обробки інформації біологічними об'єктами.

1.5 Структурна схема штучного нейрона

1.5.1 Нейрон зі скалярним входом

З приведених вище міркувань виходить, що кожен нейрон можна вважати своєрідним процесором: він підсумовує з відповідними вагами сигнали, що приходять від інших нейронів, виконує нелінійну перетворювальну функцію і передає результуюче значення пов'язаним з ним нейронам. Структура нейрона з одним скалярним входом показана на рис.1.3а.

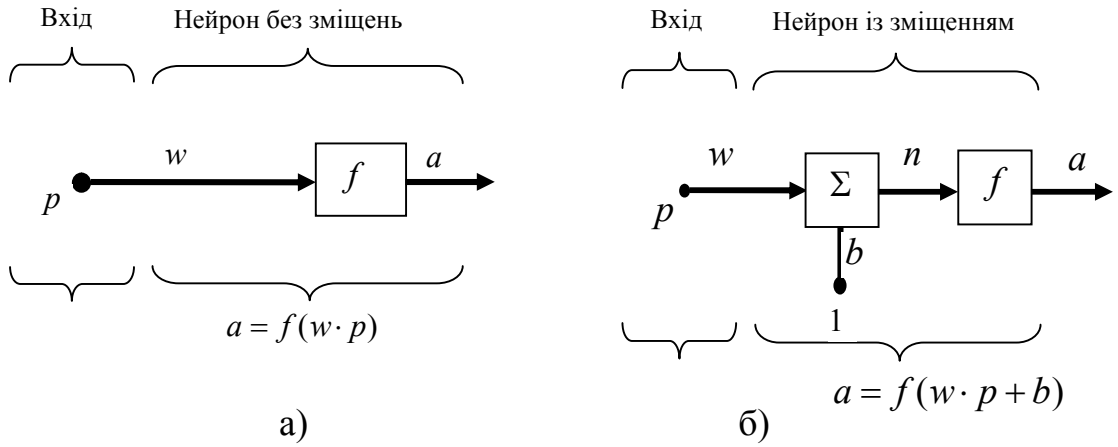


Рис.1.3. Структура простого нейрона:
а) – нейрон з одним скалярним входом; б) – нейрон з скалярним зсувом

Скалярний вхідний сигнал p умножається на скалярний ваговий коефіцієнт w і результуючий зважений вхід $w \cdot p$ є аргументом функції активації нейрона, яка породжує скалярний вихід a.

Нейрон, показаний на рис.1.3б, доповнений скалярним зсувом b. Зсув підсумовується із зваженим входом $w \cdot p$ і приводить до зсуву аргументу функції f на величину b. Дію зсуву можна звести до схеми зважування, якщо уявити, що нейрон має другий вхідний сигнал із значенням, рівним 1. Вхід n функції активації нейрона, як і раніше, залишається скалярним і рівним сумі зваженого входу і зсуву b. Ця сума є аргументом функції активації f; виходом функції активації є сигнал a. Константи w і b є скалярними параметрами нейрона. Основний принцип роботи нейронної мережі полягає в настройці параметрів нейрона так, щоб поведінка мережі відповідала деякій бажаній поведінці. Регулюючи ваги або параметри зсуву, можна навчити мережу виконувати конкретну роботу; можливо також, що мережа сама коректуватиме свої параметри, щоб досягти необхідного результату.

Рівняння нейрона із зсувом має вигляд:

$$a = f(w \cdot p + b \cdot 1) \ .$$

Як вже наголошувалося, зсув b – скалярний параметр нейрона, який не є входом, що настроюється, а константа 1, яка управляє зсувом, розглядається як вхід і може бути врахована у вигляді лінійної комбінації векторів входу

$$a = \begin{bmatrix} w & p \end{bmatrix} \begin{bmatrix} b \\ 1 \end{bmatrix}.$$

1.5.2 Функції активації нейрона

Функції активації (передавальні функції) нейрона можуть мати самий різний вигляд. Приклади активаційних функцій, що набули найбільшого застосування, представлені в табл. 1.1.

Таблиця 1.1

Перелік функцій активації нейронів

Назва	Функція	Похідна
Лінійна	$f(n)=n$	$f'(n)=1$
Ступінчаста (порогова)	$f(n)=\begin{cases}0, & n<0\\1, & n\geq 0\end{cases}$	$f'(n)=0$
Сигмоїдальна (логістична) sigmoid	$f(n)=\frac{1}{1+e^{-n}}$	$f'(n)=f(n)(1-f(n))$
Гіперболічний тангенс tang	$f(n)=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$	$f'(n)=1-f^{2}(n)$
ReLU	$f(n)=\begin{cases}0, & n<0\\n, & n\geq 0\end{cases}$	$f(n)=\begin{cases}0, & n<0\\1, & n\geq 0\end{cases}$
SoftPlus	$f(n)=\log(1+e^{n})$	$f'(n)=\frac{1}{1+e^{-n}}$
Leaky ReLU, Parameterized ReLU	$f(n)=\begin{cases}an, & n<0\\n, & n\geq 0\end{cases}$	$f'(n)=\begin{cases}a, & n<0\\1, & n\geq 0\end{cases}$
ELU	$f(n)=\begin{cases}a(e^{n}-1), & n<0\\n, & n\geq 0\end{cases}$	$f'(n)=\begin{cases}f(n)+a, & n<0\\1, & n\geq 0\end{cases}$

Розглянемо більш детально функції активації, що наведені в табл.1.1

Лінійна функція активації $f(n)=n$ показана на рис.1.4.-Це дуже проста функція активації, яка є базовим прикладом прямої залежності між входом і виходом нейрона. Вона є особливим випадком лінійної функції активації, де коефіцієнт нахилу (вага) дорівнює 1, а зсув (bias) дорівнює 0. Тобто функція $f(n)=n$ передає вхідне значення на вихід без будь-якої модифікації.

Властивості функції активації $f(n)=n$***:***

1. *Лінійність:* Ця функція є лінійною, тобто вихід пропорційний входу з коефіцієнтом нахилу 1 та без зміщення. Лінійна функція не додає жодної складності або нелінійності в мережу.
2. *Простота*: Функція $f(n)=n$ є надзвичайно простою для обчислення. Вона не вимагає складних математичних операцій або трансформацій вхідного значення, що може бути корисно в деяких задачах.
3. *Без додаткової нелінійності:* Оскільки це лінійна функція, вона не дозволяє нейронним мережам вивчати складні, нелінійні залежності. Це обмежує її здатність моделювати більш складні зв'язки в даних.

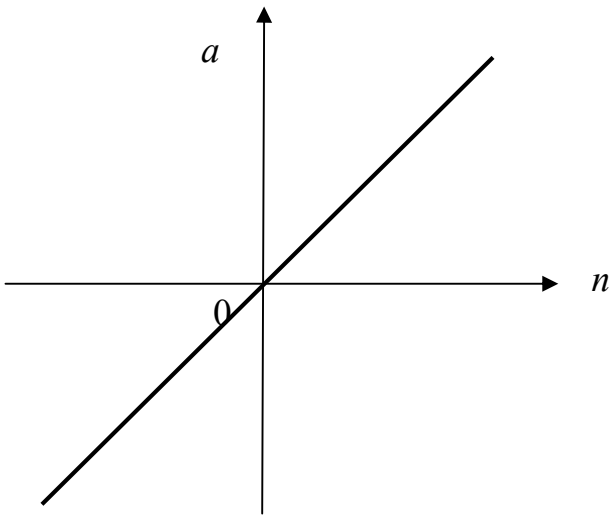


Рис.1.5. Лінійна функція активації

Переваги функції активації $f(n) = n$***:***

1. *Простота та ефективність для лінійних задач*: У задачах, де вхід і вихід мають лінійний зв'язок (наприклад, у простих задачах регресії), ця функція може бути дуже ефективною.
2. *Швидкість обчислень*: Завдяки своїй простоті, ця функція є швидкою для обчислень, що може бути корисно в реальному часі або для дуже простих моделей.

Примітка. ***Задачі регресії*** - це тип задач в машинному навчанні, де метою є передбачити безперервну (числову) змінну на основі вхідних даних. На відміну від задач класифікації, де потрібно передбачити категорії або класи, регресія фокусується на числових результатах.

Недоліки функції активації $f(n) = n$***:***

1. *Невисока здатність до моделювання складних залежностей:* Оскільки функція є лінійною, вона не може моделювати нелінійні зв'язки між вхідними та вихідними даними. Це обмежує її застосування в більш складних задачах, де важливо обробляти нелінійні залежності.
2. *Відсутність здатності до навчання складних моделей:* Вона не додає достатньої гнучкості для складних нейронних мереж, особливо в контексті глибокого навчання, де нелінійні функції активації є важливими для моделювання складних структур і патернів у даних.
3. *Проблеми при використанні в багатошарових мережах*: Якщо в мережі використовується лише лінійна функція активації, то вся нейронна мережа буде лінійною незалежно від кількості шарів. Це означає, що навіть глибока мережа не зможе навчити складні нелінійні функції, і її ефективність буде обмежена.

Застосування функції активації $f(n) = n$***:***

1. *Прості моделі регресії:* Функція $f(n) = n$ може бути використана в *простих моделях регресії,* де потрібна лінійна залежність між вхідними та вихідними

значеннями. В таких випадках модель просто відображає вхідне значення на вихід.

2. *Лінійні системи управління*: Для лінійних систем управління, де змінні вхідні та вихідні величини мають лінійні взаємозв'язки, можна використовувати цю функцію як елемент простого контролера.

Функція активації $f(n) = n$ може використовуватись в ***вихідному шарі*** нейронних мереж, зокрема у двошарових мережах, де на прихованому шарі застосовується нелінійна функція активації, така як логістична функція (sigmoid) або гіперболічний тангенс (tanh).

Використання $f(n) = n$ ***у вихідному шарі:***

У таких моделях, як двошарові нейронні мережі, де на першому (прихованому) шарі застосовуються нелінійні функції активації, лінійна функція активації $f(n) = n$ часто використовується в вихідному шарі, особливо для задач регресії.

Причини, що обумовлюють використання лінійноъ функціь активації на виході.

1. *Лінійний зв'язок між вхідними та вихідними даними:* У задачах регресії важливо мати лінійний зв'язок між вхідними ознаками і результатом, тому в таких випадках на виході мережі використовується лінійна функція активації, щоб прогнозувати безперервні значення.
2. *Нелінійність на прихованих шарах:* Нелінійні функції активації на прихованих шарах дозволяють моделювати складні залежності між вхідними та вихідними даними, в той час як лінійна функція на виході забезпечує коректне передбачення значень.
3. *Задачі регресії:* Лінійна функція активації на виході дозволяє мережі генерувати вихід, який є безперервним значенням, що підходить для задач, де результатом є числове значення, а не категорія.

Підсумок: Лінійна функція активації $f(n) = n$ зазвичай використовується на вихідному шарі нейронних мереж, коли потрібно прогнозувати безперервне значення (наприклад, для задач регресії), а нелінійні функції активації використовуються на прихованих шарах для моделювання складних зв'язків між вхідними даними. Це дозволяє нейронній мережі поєднувати сильні нелінійні властивості (для обробки складних залежностей) з лінійним результатом на виході для отримання точних числових прогнозів.

Ступінчаста (порогова) функція активації (рис.1.5) - одна з простих нелінійних функцій активації, яка визначає вихід нейрона в залежності від певного порогу (на рис.1.5 показана ступінчаста функція активації, у якої порог активації дорівнює 0). Вона є класичним прикладом активації в ранніх моделях нейронних мереж, зокрема в **перцептроні**, і використовується для задач, де потрібно здійснити чітке розмежування між класами або ситуаціями.

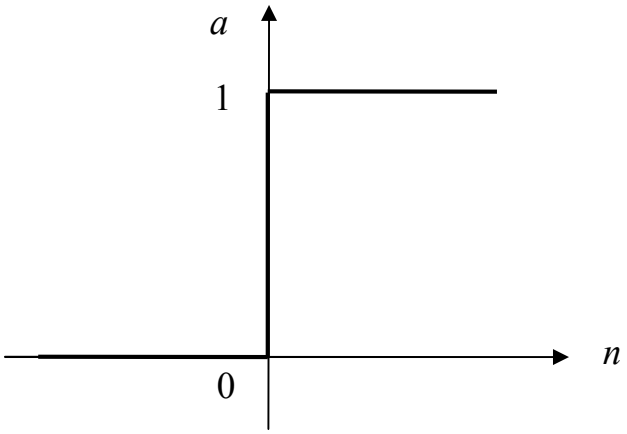


Рис.1.4. Ступінчаста (порогова) функція активації

Ступінчаста функція активації має два можливих виходи: вихід 1, якщо вхідне значення перевищує поріг; вихід 0, якщо вхідне значення менше порогу.

Переваги ступінчастої функції активації:

1. *Простота*: Ступінчаста функція активації є дуже простою в обчисленні, оскільки вона просто порівнює вхід із заданим порогом і видає результат.
2. Чітка класифікація: Вона добре підходить для задач, де потрібно приймати рішення "так/ні" або розділяти вхідні дані на два чітких класи (наприклад, "активний/неактивний" або "1/0").

Недоліки ступінчастої функції активації:

1. *Непреривність і недиференційованість:* Ступінчаста функція є неперервною, але не диференційованою в точці порогу, що ускладнює використання методу зворотного поширення помилки (backpropagation) для навчання нейронної мережі. Зокрема, при наявності ступінчастої функції активації, градієнти не будуть існувати для багатьох значень входу, що ускладнює оптимізацію.
2. *Втрата інформації:* Оскільки вихід функції приймає лише два значення (0 або 1), вона втрачає інформацію про величину вхідного значення. Це означає, що в процесі навчання нейронна мережа не здатна вчитися на основі поступових змін значень вхідних даних.
3. *Проблеми з тренуванням*: Ступінчаста функція активації часто не забезпечує ефективного навчання в глибоких нейронних мережах, оскільки при зворотному поширенні помилки **градієнт** в багатьох точках буде нульовим. Це призводить до того, що мережа не може коректно коригувати ваги і навчатися.

Застосування ступінчастої функції активації:

- *Перцептрон:* Ступінчаста функція активації була основною в ранніх моделях нейронних мереж, зокрема в класичному перцептроні, який використовується для бінарної класифікації.
- *Логічні операції:* Вона добре підходить для моделювання логічних операцій, таких як AND, OR, де вхідні дані повинні бути чітко розподілені на дві категорії.

Підсумок: Ступінчаста функція активації є простою і чіткою, але через свої обмеження (недоцільність в нелінійних задачах, недиференційованість) вона поступово була замінена іншими більш складними функціями активації, такими як ReLU, sigmoid, tanh, що забезпечують більш ефективне навчання нейронних мереж і здатність до обробки складних зв'язків між даними.

Логістична функція активації (або **сигмоїдна функція**) (рис. 1.6) є однією з найпоширеніших нелінійних функцій активації, яка використовується в нейронних мережах, особливо у багатошарових перцептронах (MLP) **та** рекурентних нейронних мережах (RNN)**.** Логістична функція широко використовується для задач класифікації, де результат повинен бути ймовірністю належності до певного класу.

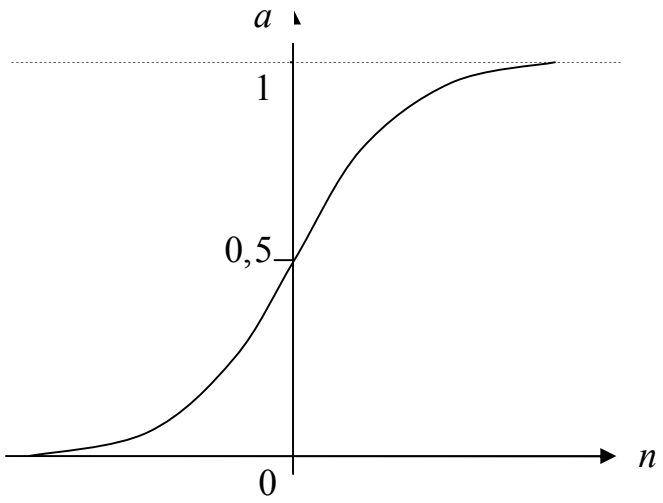


Рис.1.6. Логістична функція активації

Властивості логістичної функції активації:

1. *Вихідне значення*: Логістична функція дає значення на виході в діапазоні від 0 до 1, що дозволяє інтерпретувати результат як ймовірність. Це робить її дуже корисною для задач класифікації, де потрібно вивести ймовірність належності до певного класу.
    - Коли n дуже велике та позитивне, $f(n)$ наближається до 1.
    - Коли n дуже велике та від'ємне, $f(n) = n$ наближається до 0.
2. *Нелінійність*: Логістична функція є нелінійною, що дозволяє нейронній мережі моделювати складні зв'язки між вхідними та вихідними значеннями.

3. *Плавна і безперервна*: Логістична функція є гладкою та неперервною, що дозволяє ефективно використовувати метод градієнтного спуску для навчання нейронних мереж.
4. **Симетричність**: Логістична функція є симетричною відносно точки $x = 0$, тобто $f(-x) = 1 - f(x)$.

Переваги логістичної функції активації:

1. *Інтерпретованість як ймовірність*: Оскільки вихід логістичної функції завжди знаходиться в межах від 0 до 1, це дозволяє інтерпретувати її значення як ймовірність належності до певного класу, що особливо корисно в задачах класифікації.
2. *Нелінійність*: Логістична функція дозволяє моделювати складні залежності між входами та виходами, що важливо для навчання нейронних мереж.
3. Гладкість і безперервність: Вона має похідну в будь-якій точці, що дозволяє ефективно застосовувати алгоритми оптимізації, такі як градієнтний спуск.

Недоліки логістичної функції активації:

1. *Виникнення проблеми зникнення градієнта*: Логістична функція може призводити до зникнення градієнта під час зворотного поширення помилки, особливо для дуже великих або дуже малих значень ххх. Це ускладнює навчання в глибоких мережах, де градієнт може ставати надзвичайно малим, і ваги нейронів перестають оновлюватися.
2. *Обмежений діапазон*: Оскільки логістична функція обмежена значенням від 0 до 1, вона може не бути оптимальною для задач, де потрібні більш широкі або інші діапазони значень.

Застосування логістичної функції активації:

1. *Бінарна класифікація*: Логістична функція використовується в нейронних мережах для задач *бінарної класифікації*, де потрібно передбачити, чи належить вхід до одного з двох класів (наприклад, "так" або "ні").
2. *Ймовірності в моделюванні*: Вона також може бути використана в системах, де передбачення має інтерпретуватися як ймовірність, наприклад, в моделях машинного навчання, таких як логістична регресія.
3. *Останній шар нейронної мережі*: Логістичну функцію часто використовують в останньому шарі нейронних мереж для задач класифікації, де вихід є ймовірністю належності до певного класу.

Підсумок: Логістична функція активації є потужним інструментом для задач бінарної класифікації та моделювання ймовірностей. Однак вона має деякі обмеження, такі як проблема зникнення градієнта в глибоких нейронних мережах. Незважаючи на це, вона залишається одним з основних інструментів в машинному навчанні та нейронних мережах для вирішення завдань, де потрібно класифікувати дані у два класи або оцінити ймовірність.

Функція активації гіперболічного тангенса (tanh) (рис.1.7) є однією з найпоширеніших нелінійних функцій активації в нейронних мережах, яка застосовується для перетворення вхідного сигналу нейрона в значення, що знаходяться в межах від -1 до 1. Вона широко використовується в таких моделях, як рекурентні нейронні мережі (RNN) та багатошарові перцептрони (MLP).

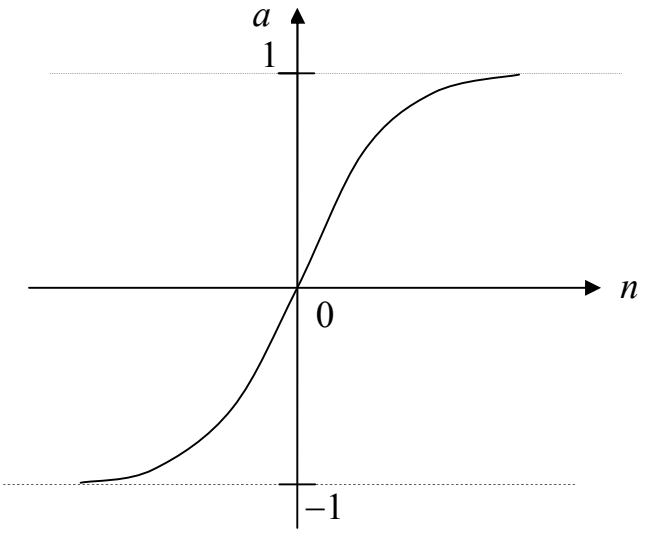


Рис.1.7. Функція активації типу гіперболічного тангенса

Властивості функції активації гіперболічного тангенса:

1. *Вихідний діапазон*: Функція перетворює вхідне значення в діапазон від -1 до 1. Тобто:
   - $f(n) \to 1$ при великих позитивних значеннях n,
   - $f(n) \to -1$ при великих від'ємних значеннях n,
   - $f(n) = 0$ при $n = 0$.

   Така властивість робить її придатною для задач, де потрібно мати вихідні значення, які є *симетричними* відносно нуля.
2. *Нелінійність*: Як і логістична функція активації, гіперболічний тангенс є нелінійною функцією, що дозволяє моделювати складні залежності між входом та виходом нейрона.
3. *Гладкість та диференційованість*: Функція є гладкою (тобто має похідну в будь-якій точці) і безперервною, що робить її зручною для використання в алгоритмах оптимізації, зокрема в методі градієнтного спуску.
4. *Симетричність*: Відмінною рисою tanh є її симетричність відносно нуля. Це означає, що для будь-якого від'ємного значення n, значення $f(n)$ буде від'ємним, а для позитивних значень n - позитивним. Це дозволяє моделі краще обробляти дані з негативними значеннями і покращує навчання мережі.

Переваги функції гіперболічного тангенса:

1. *Симетрія*: На відміну від логістичної функції активації (sigmoid), яка має вихід в діапазоні від 0 до 1, функція tanh має симетричний діапазон [1,1]. Це

дозволяє зменшити зміщення даних під час передачі сигналу через нейронні шари і може пришвидшити навчання.

2. *Плавне відновлення сигналу*: Завдяки своєму гладкому графіку, функція добре підходить для задач, де важливі малі зміни входів. Вона дозволяє навчити мережу на складних даних і має плавний перехід між вихідними значеннями.
3. *Широке застосування*: tanh є однією з основних функцій активації, особливо для рекурентних нейронних мереж (RNN), де потрібно працювати з довгими послідовностями, оскільки вона здатна зберігати інформацію з минулого через більший діапазон значень.

Недоліки функції гіперболічного тангенса:

1. *Проблема зникнення градієнта*: Подібно до логістичної функції активації, tanh страждає від проблеми зникнення градієнта при великих значеннях n. Для дуже великих або дуже малих значень n градієнт функції наближається до нуля, що може призвести до того, що параметри мережі перестануть оновлюватися під час навчання.
2. *Великий час навчання в глибоких мережах*: Через зникнення градієнта ця функція може призвести до довшого навчання в глибоких нейронних мережах. Ця проблема часто вирішується використанням інших функцій активації, таких як ReLU.

Застосування функції активації гіперболічного тангенса:

1. *Рекурентні нейронні мережі (RNN):* tanh є однією з основних функцій активації для рекурентних мереж, де вона використовується для передавання інформації між нейронами на кожному кроці часу.
2. *Глибокі нейронні мережі*: Використовується в багатошарових перцептронах (MLP) для задач, де важливо мати обробку симетричних входів і виходів, особливо коли є потреба в точному передбаченні складних залежностей.
3. *Фінансові моделі та прогнози:* Завдяки своїй здатності до моделювання нелінійних залежностей, ця функція може бути корисною в задачах прогнозування та фінансових моделях, де враховуються складні взаємозв'язки між різними змінними.
4. *Нейромережеві регулятори:* функція гіперболічного тангенса (tanh) використовуються в прихованих шарах нейронних мереж в нейромережевих регуляторах, тому що вони здатні моделювати складні нелінійні залежності між входами та виходами. Вони допомагають досягти ефективного навчання і адаптації нейронної мережі до змінних умов, що є важливим для задач автоматичного управління та регулювання складних систем

Підсумок: Функція гіперболічного тангенса (tanh) є важливою нелінійною функцією активації, яка допомагає моделювати складні залежності в нейронних мережах. Вона має перевагу симетрії та плавного відновлення сигналу, але також страждає від проблеми зникнення градієнта, що може обмежувати її ефективність у глибоких мережах. Вона широко використовується в задачах регресії, класифікації та обробці послідовностей, таких як у рекурентних нейронних мережах.

Функція активації ReLU (Rectified Linear Unit) (рис.1.8) є однією з найпопулярніших і часто використовуваних у сучасних нейронних мережах, особливо для глибоких мереж. Вона є простим і ефективним способом додавання нелінійності до моделі, що дозволяє нейронній мережі вчитися складним залежностям у даних. Основною перевагою ReLU є її здатність подолати проблему *градієнтного зникнення,* що часто виникає при використанні інших класичних функцій активації, таких як Sigmoid або Tanh.

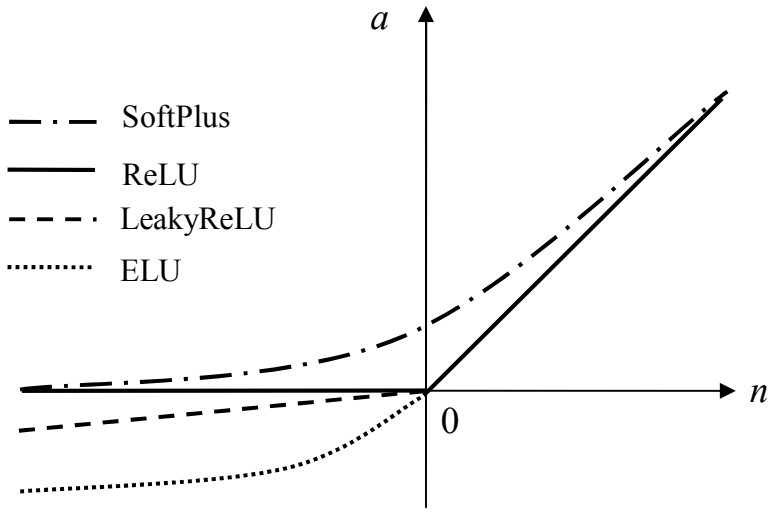


Рис.1.8. Функція активації типу гіперболічного тангенса

Математичне визначення:

Функція ReLU визначається як $f(n) = \max(0, x)$, тобто, для будь-якого вхідного значення n:

- якщо $n > 0$, то функція повертає n;
- якщо $n \le 0$, то функція повертає 0.

Особливості та переваги:

1. *Простота*: ReLU є надзвичайно простою функцією для обчислення, оскільки вона лише порівнює вхідне значення з нулем, що значно зменшує обчислювальну складність у порівнянні з іншими функціями активації.
2. *Уникнення проблеми градієнтного зникнення*: Оскільки функція ReLU не має обмежень по верхній межі (на відміну від Sigmoid чи Tanh), градієнт не стає надто малим для великих значень xxx, що дозволяє більш ефективно навчати глибокі мережі.
3. *Швидка збіжність*: Завдяки простоті та ефективності, ReLU допомагає мережам швидше сходитися під час навчання.
4. *Спарсність активації*: Оскільки ReLU повертає нуль для всіх негативних значень, частина нейронів в мережі може бути неактивною (мають нульовий вихід) для певних входів. Це може бути корисним для зменшення перенавчання, оскільки мережа не залежить від кожного нейрону.

Недоліки:

1. «*Мертві нейрони*»: Одним з основних недоліків ReLU є проблема *мертвих нейронів*. Якщо на будь-якому етапі навчання вага нейрону стає такою, що він отримує постійно негативні значення на вхід, нейрон буде завжди "вимкнений" (отримувати 0 на виході). Це може призвести до того, що нейрон не буде навчатися і не буде вносити свій вклад в мережу.
2. *Вибір функцій активації*: Іноді для специфічних задач замість чистої ReLU використовуються модифікації, такі як Leaky ReLU, Parametric ReLU або ELU (Exponential Linear Unit), щоб мінімізувати проблему мертвих нейронів, дозволяючи маленький позитивний градієнт для негативних значень.

Варіанти ReLU:

- **Leaky ReLU**: У цій версії функції для негативних значень вхідних даних не повертається рівно 0, а дається мале значення, наприклад, $f(n) = \alpha n$ для $n < 0$, де α - мале додатне число (зазвичай 0,01).
- **Parametric ReLU (PReLU)**: Це розширення Leaky ReLU, де параметр α для негативних значень є навчальним параметром, тобто він може адаптуватися під час тренування.
- **ELU (Exponential Linear Unit)**: Виправляє деякі недоліки ReLU, оскільки для негативних значень застосовується експоненціальна функція, що дозволяє покращити якості навчання.

Функція **ReLU** є основою для багатьох сучасних нейронних мереж, і її використання дає значні переваги в задачах, де швидкість і ефективність навчання є критичними.

SoftPlus (див. рис. 1.8) - це ще одна нелінійна функція активації, яка є гладким наближенням до функції ReLU**.** Вона є популярною альтернативою для активаційних функцій, таких як ReLU та sigmoid, завдяки своїй м'якості та безперервності. Softplus особливо корисна в задачах, де потрібна плавна активація з меншою кількістю проблем, пов'язаних з перенавчанням чи зникненням градієнта**.**

Властивості функції Softplus:

1. *Плавна активація*: SoftPlus — це гладка та неперервна функція. Вона не має різких стрибків або порогових значень, на відміну від функції ReLU, що дозволяє їй бути більш стабільною при навчанні нейронних мереж. Оскільки функція гладка, вона допомагає уникнути різких змін у значеннях, що може бути корисним у деяких задачах.
2. *Нелінійність*: Як і інші функції активації, softplus є нелінійною функцією, що дозволяє нейронним мережам моделювати складні залежності між вхідними та вихідними значеннями.
3. *Вихід у діапазоні* $(0,\infty)$. Значення softplus завжди позитивне. Функція наближається до нуля при великих від'ємних значеннях n і зростає, наближаючись до n при великих позитивних значеннях n. Це відрізняє її від ReLU, яка дає нулі для всіх від'ємних значень і є дискретною.

4. *Похідна:* Похідна функції softplus є функцією sigmoid, яка має значення від 0 до 1. Це означає, що softplus володіє корисною властивістю - її похідна не стає нульовою навіть для великих значень n, як це може бути в інших активаційних функціях (наприклад, у sigmoid).

Переваги функції Softplus:

1. *Гладкіст***ь**: Завдяки плавній природі softplus дозволяє уникати різких стрибків у вихідних значеннях і є стабільнішою порівняно з ReLU.
2. *Немає мертвих нейронів*: На відміну від ReLU, softplus не має проблеми мертвих нейронів, оскільки не дає нульових значень для всіх від'ємних входів.
3. *Плавний перехід*: Функція забезпечує плавний перехід між нулем і позитивними значеннями, що корисно для задач, де потрібна м'яка активація.
4. *Зручність при оптимізації*: Softplus допомагає уникнути деяких проблем, які можуть виникати в інших функціях активації, наприклад, із занадто різкими переходами.

Недоліки функції Softplus:

1. *Більш обчислювально витратна:* Оскільки SoftPlus включає обчислення експоненційної функції і логарифма, це може бути більш витратним з точки зору обчислювальних ресурсів порівняно з простими функціями активації, такими як ReLU.
2. *Не обмежує вихід*: Вихід SoftPlus не обмежений, і він може ставати дуже великим при великих значеннях n. Це може бути проблемою в певних контекстах, де потрібно обмежити вихід.

Застосування функції SoftPlus:

1. *Глибокі нейронні мережі:* SoftPlus може використовуватись в глибоких нейронних мережах, де потрібна м'яка і стабільна активація для обробки складних даних.
2. *Регресія*: В задачах, де потрібен плавний вихід (наприклад, для передбачення числових значень), softplus може бути корисною, оскільки вона не призводить до різких змін у вихідному значенні.
3. *Автокодери та рекурентні мережі*: Завдяки своїй гладкості, SoftPlus також може бути корисною в автокодерах або рекурентних нейронних мережах, де потрібно обробляти послідовності з плавними змінами.

Підсумок: Функція активації SoftSlus є м'яким і плавним варіантом для використання в нейронних мережах, зокрема в задачах, де потрібна стабільність і поступове зростання вихідних значень. Вона може бути корисною для уникнення проблем з мертвими нейронами (як у ReLU) та для задач регресії чи обробки складних даних. Однак її обчислювальна витратність може бути вищою порівняно з іншими функціями активації, такими як ReLU.

У ППП Deep Learning Toolbox включені і інші функції активації. Використовуючи мову MATLAB, користувач може створювати і свої власні унікальні функції.

1.5.3 Нейрон з векторним входом

Нейрон з одним вектором входу **p** з R елементами $p_1, p_2, ..., p_R$ показаний на рис.1.9. Тут кожен елемент входу умножається на ваги $w_{11}, w_{12}, ..., w_{1R}$ відповідно, і зважені значення передаються на суматор. Їх сума рівна скалярному добутку вектора-рядка **W** на вектор входу **p**.

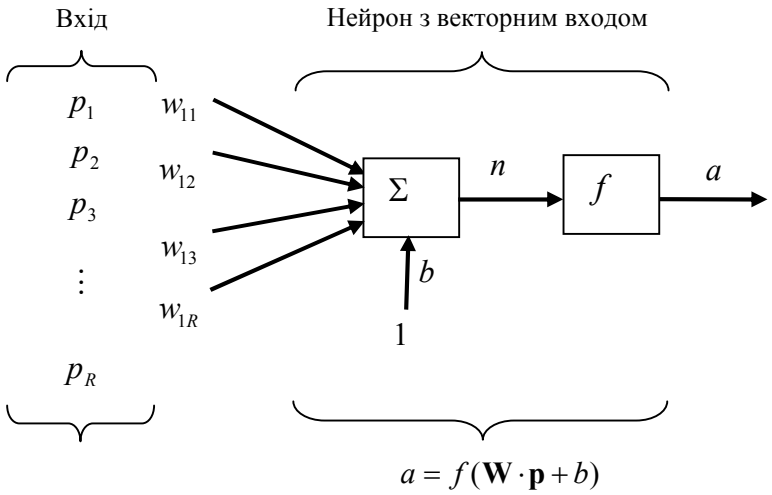


Рис.19. Схема нейрона з векторним входом

Нейрон має зсув b, який підсумовується із зваженою сумою входів. Результуюча сума n рівна

$$n = w_{11} \cdot p_1 + w_{12} \cdot p_2 + ... + w_{1R} \cdot p_R + b$$

і служить аргументом функції активації f. У нотації мови MATLAB цей вираз записується так:

$$n = \mathbf{W} \cdot \mathbf{p} + b\,.$$

Структура нейрона, показана вище, містить багато зайвих деталей. При розгляді мереж, що складаються з великого числа нейронів, в ППП Neural Network Toolbox використовується укрупнена структурна схема нейрона (рис.1.10).

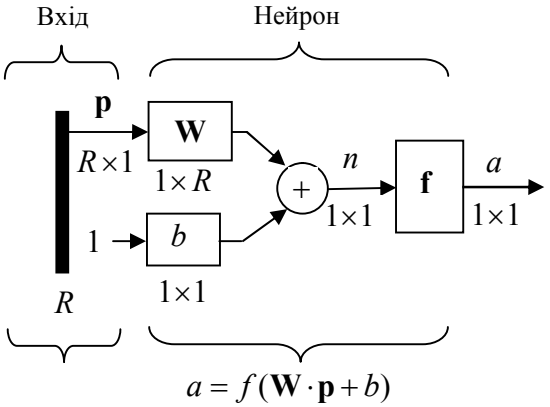


Рис.1.10. Укрупнена структурна схема нейрона

Вхід нейрона зображується у вигляді темної вертикальної лінії, під якою указується кількість елементів входу R. Розмір вектора входу **p** указується нижче за символ **p** і рівний R. Вектор входу умножається на вектор-рядок **W** довжини R. Як і раніше, константа 1 розглядається як вхід, який умножається на скалярний зсув b. Входом n функції активації нейрона служить сума зсуву b і добутки $\mathbf{W} \cdot \mathbf{p}$. Ця сума перетворюється функцією активації f, на виході якої отримується вихід нейрона a, який в даному випадку є скалярною величиною. Структурна схема, приведена на рис.1.9, називається **шаром мережі**. Шар характеризується матрицею вагів $\mathbf{W}$, зсувом b, операціями множення $\mathbf{W} \cdot \mathbf{p}$, підсумовування і функцією активації f. Вектор входів **p** зазвичай не включається в характеристики шару.

1.6 Класичні нейронні мережі і їх властивості

Штучна нейронна мережа – це набір нейронів, сполучених між собою. Як правило, передавальні (активаційні) функції всіх нейронів в мережі фіксовані, а ваги є параметрами мережі і можуть змінюватися. Деякі входи нейронів помічені як зовнішні входи мережі, а деякі виходи – як зовнішні виходи мережі. Подаючи деякі числа на входи мережі, отримується якийсь набір чисел на виходах мережі. Таким чином, робота нейромережі полягає в перетворенні вхідного вектора $\mathbf{X}$ у вихідний вектор $\mathbf{Y}$, причому це перетворення задається вагами мережі.

Практично будь-яке завдання можна звести до завдання, вирішуваного нейромережею.

Питання про методику побудови нейронної мережі вирішується в два етапи:

1. вибір типу (архітектури) мережі;
2. підбір вагів (навчання) мережі.

На першому етапі слід вибрати наступне:

1. які нейрони слід використовувати (число входів, передавальні функції);
2. яким чином слід з'єднати їх між собою;
3. що взяти за входи і виходи мережі.

Це завдання на перший погляд здається неозорим, але, на щастя, не обов'язково придумувати нейромережу «з нуля» – існує велика кылькысть нейромережевих архітектур, причому ефективність багатьох з них доведена математично.

На другому етапі слід «навчити» вибрану мережу, тобто підібрати такі значення її вагів, щоб мережа працювала потрібним чином. У використовуваних на практиці нейромережах кількість вагів може бути досить великою, тому навчання – дійсно складний процес. Для багатьох архітектур розроблені спеціальні алгоритми навчання, які дозволяють набудувати ваги мережі певним чином.

Залежно від функцій, що виконуються нейронами в мережі, можна виділити три їх типи:

- вхідні нейрони – на які подається вхідний вектор, що кодує вхідну дію або образ зовнішнього середовища; у них зазвичай не здійснюється обчислю-

вальних процедур, інформація передається з входу на вихід нейрона шляхом зміни його активації;

- вихідні нейрони – вихідні значення яких представляють вихід мережі; перетворення в таких нейронах здійснюються по приведених вище виразах;
- проміжні нейрони – складають основу штучних нейронних мереж, перетворення в них виконуються по цих же виразах.

У більшості нейронних моделей тип нейрона пов'язаний з його розташуванням в мережі. Якщо нейрон має тільки вхідні зв'язки, то це вхідний нейрон, якщо навпаки – вихідний нейрон. Проте може зустрітися випадок, коли вихід внутрішнього нейрона розглядається як частина виходу мережі. В процесі функціонування мережі здійснюється перетворення вхідного вектора у вихідний, тобто деяка переробка інформації. Конкретний вид виконуваного мережею перетворення інформації обумовлюється не тільки характеристиками нейроподібних елементів (функціями активації і т. п.), але і особливостями її архітектури. Зокрема, тією або іншою топологією міжнейронних зв'язків, вибором певних підмножин нейроподібних елементів для введення і виведення інформації, способами навчання мережі, наявністю або відсутністю конкуренції між нейронами, напрямом і способами передачі інформації між нейронами.

Нейронні мережі є потужним інструментом для вирішення різноманітних задач в області машинного навчання та штучного інтелекту. Існує багато різних архітектур нейронних мереж, кожна з яких має свої особливості та застосування. До класичних нейронних мереж можна віднести одношароваі нейронні мережі (Single-Layer Neural Network) та багатошарові мережі прямого розповсюдження (Multilayer Perceptron, MLP),мережі з «бічними з'єднаннями», **м**ережі Кохенена (Self-Organizing Maps, SOM), **м**ережі Хопфілда**,** АРТ-мережі (Adaptive Resonance Theory) і ін. Всі ці мережі мають свої специфічні застосування, залежно від типу задач і характеру даних, що обробляються. З розвитком глибокого навчання, ці архітектури поступово інтегруються в більш складні моделі, але досі залишаються важливими для окремих специфічних задач.

Структури класичних нейронних мереж

Одношарова нейронна мережа (Single-Layer Neural Network) (рис. 1.11 а). Одношарова нейронна мережа має лише один шар нейронів, який здійснює обробку вхідних даних і генерує результати. У більшості випадків цей шар є вихідним.

Основні характеристики:

Структура:

- Мережа складається з одного шару нейронів, який отримує вхідні дані, обробляє їх і генерує вихід.
- Функція активації: На кожен вхід застосовується функція активації (наприклад, сигмоїд, ReLU), щоб отримати вихідний сигнал.
- Моделі: Одношарові нейронні мережі використовуються в простих моделях, таких як перцептрон.

Обмеження:

- Одношарова нейронна мережа може розв'язувати лише лінійно роздільні задачі. Це означає, що вона не здатна обробляти складні, нелінійні залежності між вхідними та вихідними даними.
- Для більш складних задач потрібні багатошарові мережі.

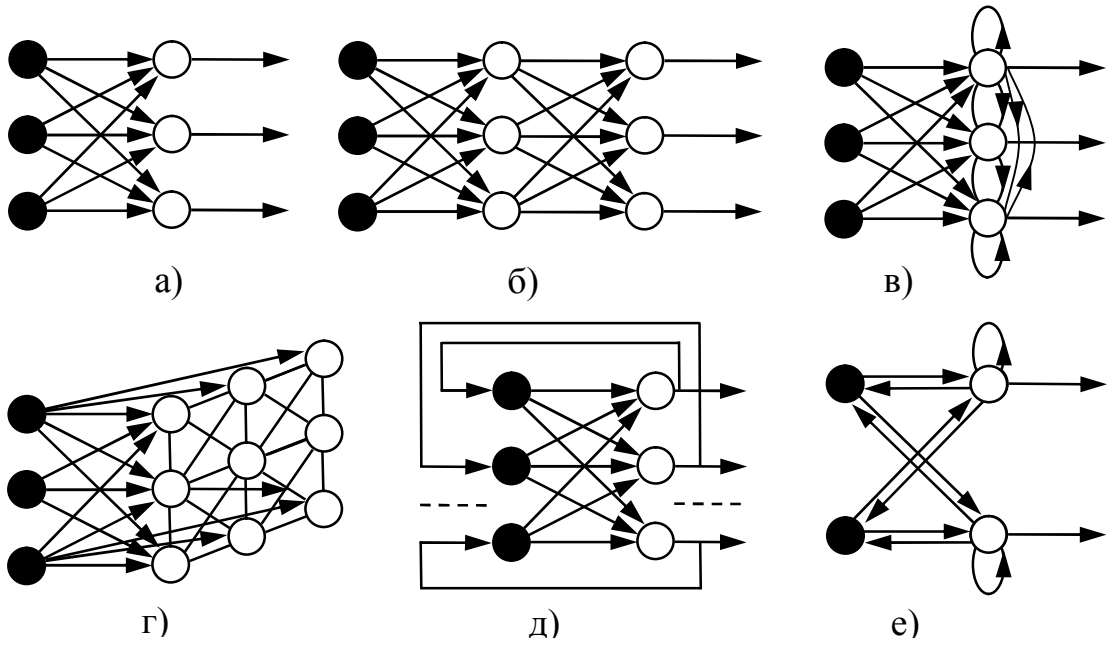


Рис.1.11. Структури класичних штучних нейронних мереж
а) – одношарова мережа; б) – багатошарова мережа (персептрон); в) – мережі з «бічними з'єднаннями»; г) – мережі Кохонена; д) – мережі Хопфілда; е) – АРТ-мережі

Багатошарові мережі прямого розповсюдження (Multilayer Perceptron, MLP) (рис. 1.11 б) є класичними нейронними мережами, які є основою для багатьох сучасних глибоких нейронних мереж, і вони заслуговують на окрему увагу в контексті огляду нейронних мереж.

Багатошарова перцептронна мережа (MLP) - це тип нейронної мережі, яка складається з кількох шарів нейронів: *вхідного шару,* одного або кількох *прихованих шарів* та *вихідного шару*. MLP є основною архітектурою, яка заклала основи для розвитку глибокого навчання.

Структура MLP:

1. Вхідний шар (Input layer) :
    - Приймає вхідні дані.
    - Кількість нейронів у цьому шарі відповідає кількості ознак у наборі даних.
2. Приховані шари (Hidden layers):
    - Можуть бути одним або кількома.
    - Нейрони в прихованому шарі отримують зважені значення з вхідного шару або попереднього прихованого шару.

- Кожен нейрон застосовує функцію активації до вхідних даних для введення нелінійності.
- Завдяки глибоким мережам, вони можуть навчатися складним патернам.

3. Вихідний шар (Output layer):
   - Генерує фінальні передбачення або класифікацію на основі активацій з останнього прихованого шару.

Основні характеристики MLP:

- Пряме поширення (Feedforward): Інформація в мережі рухається лише в одному напрямку — від вхідного шару до вихідного, без зворотних зв'язків.
- Обчислення та навчання: Мережа тренується за допомогою методу зворотного розповсюдження (backpropagation), який використовує градієнтний спуск для коригування ваг в мережі з метою мінімізації функції втрат.

MLP: Класична чи глибока мережа?

- Класична мережа: Якщо MLP містить лише один прихований шар, вона зазвичай вважається **класичною нейронною мережею**.
- Глибока нейронна мережа: Якщо ж MLP має кілька прихованих шарів, вона стає глибокою нейронною мережею (DNN), і цей тип архітектури є основою для сучасних глибоких нейронних мереж.

Переваги та недоліки MLP:

- Переваги:
  - Простота і універсальність для задач класифікації та регресії.
  - Легкість реалізації та інтеграції в різні системи.
  - Може бути дуже ефективною для задач, де дані не мають чіткої структури (як у зображеннях або послідовностях).
- Недоліки:
  - Може страждати від перенавчання при використанні великої кількості параметрів.
  - Може бути не так ефективною для задач з просторовими або часовими залежностями (наприклад, обробка зображень, звуку або тексту), де потрібні спеціалізовані мережі, такі як CNN або RNN.

Висновок: Багатошарові мережі прямого розповсюдження (MLP) є важливою частиною історії розвитку нейронних мереж і залишаються важливими для задач, що не вимагають складних просторових чи часових залежностей. З розвитком глибоких нейронних мереж, MLP з декількома прихованими шарами стали основою для більш складних моделей, таких як глибокі нейронні мережі (DNN), що дозволяють вирішувати складніші задачі.

Мережі з бічними з'єднаннями (**Networks with Skip Connections)** (рис.1.11в) - це нейронні мережі, у яких інформація може передаватися не тільки через звичайні з'єднання між шарами, але й через бічні або додаткові зв'язки між нейронами, що належать до одного шару або між різними шарами. Це дозволяє зменшити етапи пе-

редачі інформації через мережу і може покращити ефективність обробки даних, особливо для складних задач.

Опис та основні принципи:

Мережі з бічними з'єднаннями мають особливу архітектуру, де нейрони можуть не тільки передавати інформацію вперед, як у мережах прямого розповсюдження, але й обмінюватися даними між різними шарами або нейронами. Бічні зв'язки можуть бути додані на різних етапах роботи мережі, що допомагає:

1. Збереження інформації на різних етапах обробки: У мережах прямого розповсюдження інформація зазвичай передається з вхідного шару через приховані шари до вихідного. В мережах з бічними з'єднаннями цей процес значно ускладнюється, оскільки інформація може бути передана назад до попередніх нейронів або шарів. Це дозволяє більш ефективно «запам'ятовувати» важливі аспекти обробленої інформації.
2. Покращення обробки складних зв'язків: Завдяки бічним зв'язкам мережа може краще моделювати нелінійні залежності між даними, оскільки інформація може коригуватися кілька разів на шляху від вхідного шару до вихідного.
3. Скорочення часу навчання: Бічні зв'язки дозволяють зробити процес навчання більш швидким і ефективним, оскільки з'єднання можуть пропускати інформацію без необхідності її обробки через кожен наступний шар.

Типи бічних з'єднань:

1. Між нейронами одного шару (рис. 1.11 в):
    - Бічні зв'язки можуть передавати інформацію між нейронами всередині одного шару, що дозволяє нейронам взаємодіяти між собою, а не тільки через один напрямок.
    - Це допомагає в ситуаціях, коли важливі сигнали можуть виникати в межах одного шару.
2. Між шарами:
    - Бічні зв'язки можуть передавати сигнали безпосередньо від одного шару до іншого. Наприклад, інформація з прихованого шару може передаватися не тільки до наступного шару, а й до попереднього або віддаленого шару.
    - Це особливо корисно в рекурентних мережах (RNN) та глибоких мережах, де важливо, щоб сигнал міг проходити назад для коригування обробки.

4. Мережі Кохенена (Kohonen Networks) (рис.1.11 г).

Мережі Кохенена (також відомі як Самоорганізуючі карти (SOM)) є типом нейронної мережі, що використовується для кластеризації та визначення топології даних.

Структура:

- Мережа складається з двох шарів: вхідного шару та самоорганізуючої топології, де нейрони організовані в двовимірну решітку.

- Кожен нейрон у цій мережі відповідає за кластеризацію певної групи вхідних даних.

***Особливості**:*

- Мережа не потребує наявності міток для тренування (без нагляду).
- Використовується для виявлення патернів у великих наборах даних.

***Переваги**:*

- Застосовується для безнаглядної класифікації даних.
- Легко візуалізувати результати (наприклад, візуалізація груп у вигляді карт).

***Недоліки**:*

- Не підходить для задач, де потрібна висока точність класифікації.
- Обмежена здатність працювати з великими або складними даними.

5. Мережі Хопфілда (Hopfield Networks) (рис.1.11 д).

Мережі Хопфілда — це вид рекурентних нейронних мереж, які використовуються для запам'ятовування та відновлення образів або патернів.

***Структура**:*

- Мережа має односторонні зв'язки між нейронами, і кожен нейрон може зберігати біт інформації.
- Мережі Хопфілда є рекурентними, що означає, що інформація може циркулювати в мережі до досягнення стабільного стану.

***Особливості**:*

- Використовується для відновлення інформації (наприклад, для відновлення пошкоджених або неповних даних).
- Рекурентна мережа, що дозволяє ефективно працювати з пам'яттю.

***Переваги**:*

- Здатність до відновлення інформації навіть при неповних або зашумлених вхідних даних.

***Недоліки**:*

- Обмеження у здатності до навчання великих та складних моделей.
- Мережі Хопфілда зазвичай використовуються в специфічних задачах і не є універсальними.

6. АРТ-мережі (ART, Adaptive Resonance Theory) (рис.1.11 е).

ART-мережі (теорія адаптивного резонансу) використовуються для безнаглядної класифікації і є потужним інструментом для задач, де дані змінюються з часом.

***Структура**:*

- Мережі ART мають дві основні частини: входи, що класифікуються, та механізм для узгодження нових і старих класів.
- Підходять для задач, де є необхідність у додатковому навчанні нових класів без перезапуску моделі.

***Особливості**:*

- Мережі ART адаптуються до нових даних без потреби в перепідготовці всієї моделі.
- Підходить для динамічних задач, де дані змінюються в часі (наприклад, обробка сигналів або даних з сенсорів).

***Переваги**:*

- Мережі ART мають здатність до самоорганізації та адаптації без втрати раніше отриманих знань.

***Недоліки**:*

- Можуть бути складними в налаштуванні.
- Зазвичай використовуються для дуже специфічних задач.

Кожна з цих архітектур має свої сильні та слабкі сторони, і їх вибір залежить від специфіки задачі, на якій ви працюєте.

Хоча глибокі нейронні мережі значно розширили можливості машинного навчання, класичні нейронні мережі, такі як мережі Кохенена, Хопфілда, АРТ-мережі та інші, все ще можуть мати корисні застосування, особливо в специфічних задачах, де потрібна адаптація до нових даних, безнаглядне навчання або ефективне зберігання пам'яті. Вони можуть бути корисними в обмежених ресурсах або для вирішення конкретних проблем, але в загальному випадку сучасні методи, такі як глибокі нейронні мережі, часто є більш потужними і універсальними.

1.7 Глибокі нейронні мережі

Огляд сучасних структур, функцій активації та методів навчання нейронних мереж відображає розвиток технологій глибокого навчання та їх широке застосування в різноманітних областях, таких як комп'ютерний зір, обробка природної мови, робототехніка та автономні системи. Сучасні нейронні мережі часто містять складні багатошарові архітектури, зокрема конволюційні нейронні мережі (CNN)**,** які є ефективними для аналізу зображень завдяки використанню згорткових шарів для автоматичного виділення ознак з вхідних даних. Рекурентні нейронні мережі (RNN) та їх варіанти, як-от LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit)**,** застосовуються для обробки послідовних даних, таких як текст або часові ряди, завдяки своїй здатності зберігати та враховувати попередні стани через рекурентні зв'язки. Генеративні змагальні мережі (GAN) стали важливими для генерації нових даних, таких як зображення, відео чи музика, шляхом змагання двох мереж: генератора та дискримінатора. Що стосується функцій активації, класичні функції, як-от Sigmoid та Tanh, поступово замінюються більш ефективними функціями, такими як ReLU (Rectified Linear Unit)**,** яка дозволяє швидше навчати глибокі мережі завдяки своїй простоті і здатності запобігати проблемам з градієнтним зникненням. Для подолання цього ж обмеження, використовуються варіанти ReLU, такі як Leaky ReLU **і** Parametric ReLU**.** Методи навчання також зазнали значних змін. Класичний алгоритм зворотного розповсюдження помилки (backpropagation) доповнений су-

часними техніками, такими як адаптивні оптимізатори (наприклад, Adam**,** RMSprop**),** які значно покращують швидкість та стабільність навчання, враховуючи адаптивне коригування швидкості навчання для кожного параметра. Крім того, для зменшення перенавчання використовуються методи регуляризації, як-от dropout та batch normalization, що допомагають стабілізувати процес навчання та покращити узагальнення моделі на нових даних. Ці сучасні архітектури і методи значно розширили можливості нейронних мереж та зробили їх більш ефективними в задачах, що вимагають обробки великих і складних наборів даних.

1.7.1 Визначення глибоких нейронних мереж і глибокого навчання

Глибокі нейронні мережі (Deep Neural Networks, DNN) - це тип штучних нейронних мереж, що складаються з кількох шарів нейронів, через які проходять дані, перш ніж досягти виходу. Вони є основою для глибокого навчання (deep learning) і використовуються в задачах, які вимагають складного аналізу даних, таких як обробка зображень, мовлення, текстів, прогнозування тощо.

Глибоке навчання (deep learning) - це підгалузь машинного навчання, що зосереджена на використанні глибоких нейронних мереж для автоматичного навчання та обробки складних даних. Основна ідея глибокого навчання полягає в тому, щоб створити моделі, які можуть вчитися з великих обсягів даних, знаходити в них складні, багаторівневі структури та приймати рішення на основі цього навчання.

У порівнянні з традиційними методами машинного навчання, глибоке навчання дозволяє моделі автоматично витягувати корисні ознаки (фічі) з даних без необхідності вручну визначати їх. Це робиться через кілька рівнів абстракції, де кожен рівень мережі вчиться більш складним і абстрактним представленням даних.

Особливості глибоких нейронних мереж*:*

1. Багатошаровість**:** Глибинні нейронні мережі мають декілька шарів, що дозволяє модель обробляти і витягувати різноманітні рівні абстракцій з вхідних даних. Наприклад, у випадку зображень, перші шари можуть виявляти прості ознаки, такі як краї та текстури, а глибші шари - складніші структури, наприклад, об'єкти чи обличчя.
2. Автоматичне виявлення ознак**:** Однією з ключових переваг глибинних мереж є їх здатність автоматично виявляти ознаки з даних. Це значно зменшує потребу в ручному видобутку ознак, як це робиться в традиційних підходах машинного навчання. Мережі самостійно вчаться виділяти корисні патерни, що робить їх надзвичайно потужними для роботи з великими і складними наборами даних.
3. Підвищення ефективності з великими даними: Глибинні нейронні мережі особливо ефективні, коли йдеться про обробку великих наборів даних. Сучасні алгоритми, такі як масштабування навчання та паралельні обчислення, дозволяють нейронним мережам навчатися на величезних масивах даних, що в свою чергу покращує їхню здатність до узагальнення.

4. Використання нелінійностей**:** Завдяки використанню нелінійних функцій активації (наприклад, ReLU, Sigmoid, Tanh), глибинні мережі можуть моделювати складні функціональні залежності між вхідними та вихідними даними. Це дозволяє їм вирішувати задачі, які важко або неможливо вирішити за допомогою лінійних моделей.

5. Тренування за допомогою алгоритму зворотного розповсюдження помилки (backpropagation), який дозволяє оптимізувати параметри мережі через мінімізацію функції втрат.

6. Використання потужних обчислювальних ресурсів: Для тренування великих моделей глибокого навчання потрібні потужні комп'ютери.

1.7.2 Структури глибоких нейронних мереж

Існує кілька основних типів структур глибоких нейронних мереж, кожен з яких застосовується в різних сферах і задачах. Ось деякі з найпоширеніших:

1. Штучні нейронні мережі (ANN, Artificial Neural Networks)

Це базова структура глибокої нейронної мережі, яка складається з трьох основних типів шарів:

- Вхідний шар: приймає вхідні дані.
- Приховані шари: складаються з нейронів, які обробляють інформацію, передану з попереднього шару.
- Вихідний шар: дає результат або прогноз.

Зазвичай використовуються функції активації (ReLU, Sigmoid, Tanh тощо) для кожного шару, щоб вводити нелінійність в модель.

2. Згорткові (конволюційні) нейронні мережі (CNN, Convolutional Neural Networks)

CNN спеціалізуються на обробці та аналізі зображень та відео, але також можуть використовуватись для аналізу інших типів даних, таких як текст чи аудіо. Основні компоненти CNN:

- Згорткові шари (Convolutional Layers): здійснюють згортку (convolution) для виявлення просторових ознак в зображеннях (наприклад, краї, текстури, форми).
- Шари підвибірки (Pooling Layers): зменшують розмірність даних, зберігаючи важливу інформацію.
- Повнозв'язні шари (Fully Connected Layers): використовуються для фінальної класифікації або регресії.

Ці мережі дуже ефективні для задач розпізнавання образів, відео, а також для аналізу сигналів.

3. Рекурентні нейронні мережі (RNN, Recurrent Neural Networks)

RNN використовуються для обробки послідовних даних, таких як текст, аудіо або часові ряди. Вони мають особливість, що дозволяє передавати інформацію між

етапами обробки через "пам'ять" (внутрішній стан), що дозволяє моделі враховувати контекст попередніх входів:

- Рекурентні шари: нейрони мають зворотні зв'язки, що дозволяє враховувати попередні стани.

Це робить RNN потужними для задач прогнозування та генерації послідовностей, наприклад, для машинного перекладу, генерації текстів, синтезу мовлення.

4. Довгі короткочасні пам'яті (LSTM, Long Short-Term Memory)

LSTM є вдосконаленим варіантом RNN, який покращує здатність моделі запам'ятовувати та утримувати важливу інформацію на довгий період часу. Вони мають спеціальні механізми (вхідні, вихідні та забуваючі гейти), які дозволяють контролювати, яка інформація повинна зберігатися або забуватися в процесі обробки.

LSTM дуже ефективні для задач, де необхідно зберігати інформацію на довший термін, наприклад, при роботі з текстом або часовими рядами.

5. Автокодери (Autoencoders)

Автокодери — це нейронні мережі, які навчаються стискати дані (знижувати їх вимірність) і потім відновлювати їх в початкову форму. Вони складаються з двох основних частин:

- Кодувальний шар (Encoder): знижує розмірність входу, створюючи компактне представлення.
- Декодувальний шар (Decoder): відновлює дані з цього компактного представлення.

Автокодери використовуються для стиснення даних, очищення від шуму, а також для генерації нових даних, схожих на навчальні.

6. Генеративні змагальні мережі (GAN, Generative Adversarial Networks)

GAN складаються з двох мереж:

- Генератор: намагається створити нові, правдоподібні дані.
- Дискримінатор: намагається відрізнити справжні дані від фальшивих, створених генератором.

Ці мережі використовуються для генерації нових зображень, відео, текстів або навіть музики. Вони можуть бути застосовані для створення високоякісних синтетичних даних, таких як фотореалістичні зображення.

7. Трансформери (Transformers)

Трансформери - це архітектура, що зазвичай застосовується для обробки послідовних даних, особливо в задачах обробки природної мови (NLP). Вони використовують механізм уваги (Attention Mechanism), що дозволяє моделі фокусуватися на важливих частинах послідовності, а не обробляти дані послідовно, як в RNN або LSTM.

Трансформери стали основою для багатьох сучасних моделей NLP, таких як GPT, BERT і T5.

8. Сегментаційні мережі (Segmentation Networks)

Ці мережі, як правило, використовуються для задач, пов'язаних з обробкою зображень, де потрібно класифікувати кожен піксель зображення в окрему категорію (наприклад, для медичних зображень або автономного водіння). Одним з прикладів є **U-Net**-мережа, розроблена для сегментації зображень.

Сучасні нейронні мережі постійно еволюціонують, і нові архітектури та підходи дозволяють вирішувати все складніші завдання. Вони займають ключові позиції в найрізноманітніших галузях, від автоматичного перекладу до автономних транспортних засобів, і продовжують змінювати наш підхід до вирішення проблем в обробці даних і штучному інтелекті.

1.7.3 Методи навчання глибинних нейронних мереж

Навчання глибоких нейронних мереж є основним етапом, який дозволяє глибоким нейронним мережам «навчатися» на основі вхідних даних і коригувати свої параметри для досягнення оптимальних результатів.

Основною метою навчання глибоких нейронних мереж є оптимізація їх параметрів - *ваівг* та *зсувів* - таким чином, щоб мережа могла правильно класифікувати, прогнозувати або генерувати дані. Процес навчання включає використання методів *градієнтного спуску*, що дозволяють поступово коригувати параметри мережі на основі помилок, отриманих під час тестування.

Алгоритм зворотного розповсюдження (Backpropagation):

Основний метод навчання нейронних мереж, який полягає в мінімізації функції втрат за допомогою *градієнтного спуску.* Алгоритм зворотного розповсюдження дозволяє обчислювати градієнти помилки по всіх шарах мережі та коригувати ваги для мінімізації цієї помилки.

Адаптивні оптимізатори.

Оскільки градієнтний спуск може бути неефективним у деяких випадках, для оптимізації навчання використовуються методи, які адаптують швидкість навчання в залежності від величини градієнта для кожного параметра. Одним з найбільш популярних оптимізаторів є **Adam**, який поєднує переваги **AdaGrad** і **RMSProp**.

Методи регуляризаці.

Для уникнення перенавчання використовуються різноманітні техніки регуляризації, такі як *dropout, batch normalization* та *L2 регуляризація*, що допомагають зберегти модель простішою та здатною до узагальнення.

Техніки для зменшення обчислювальних витрат:

Для прискорення навчання використовуються такі методи, як *паралельні обчислення* (наприклад, використання графічних процесорів або TPU) та *переднавчання (pretraining)* для зменшення часу навчання складних моделей.

1.7.4 Функції втрат

Функції втрат (Loss Functions) - це математичні функції, які вимірюють різницю між передбаченнями моделі та реальними результатами (мітками). Вони є основним інструментом для оптимізації нейронних мереж, оскільки під час навчання модель намагається мінімізувати функцію втрат, щоб покращити свої передбачення. Вибір функції втрат залежить від типу задачі (класифікація, регресія тощо).

1. Функції втрат для задач класифікації

a) Крос-ентропія (Cross-Entropy Loss)

Крос-ентропія є однією з найпоширеніших функцій втрат для задач класифікації, особливо для бінарної та багатокласової класифікації. Вона вимірює відстань між двома ймовірнісними розподілами - реальним розподілом міток (справжнім класом) і передбаченим розподілом ймовірностей, що генерується моделлю.

- *Бінарна крос-ентропія:* Для задачі бінарної класифікації, де є два класи (наприклад, 0 та 1), бінарна крос-ентропія (також відома як log loss) має вигляд:

$$E = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

де: y_i - реальна мітка (0 або 1); p_i - ймовірність, передбачена моделлю для класу 1.

- *Багатокласова крос-ентропія:* Для задач багатокласової класифікації, де є більше двох класів, крос-ентропія виглядає так:

$$E = -\frac{1}{N}\sum_{i=1}^{N}\ \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}),$$

де: $y_{i,c}$ - індикаторна змінна, що вказує, чи належить приклад i класу c (1 - так, 0 - ні); $p_{i,c}$ — ймовірність, передбачена моделлю для класу c.

Переваги:

- Найпоширеніша функція втрат для задач класифікації.
- Враховує ймовірнісні оцінки, що дозволяє моделі оцінювати свою впевненість у передбаченні.

Недоліки:

- Якщо дані сильно незбалансовані, клас з малою кількістю прикладів може домінувати в підсумкових оцінках.

b) Hinge Loss

Hinge Loss використовується переважно для машин опорних векторів (SVM), але її також можна використовувати для класифікаційних нейронних мереж, особливо для задач бінарної класифікації.

Математичне вираження:

$$E = \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$$

де: y_i - реальна мітка (1 або -1); $f(x_i)$ - передбачене значення (вихід моделі).

Hinge Loss штрафує моделі за помилки, де передбачене значення має протилежний знак до реальної мітки.

Переваги:

- Дуже добре працює для задач з чіткими межами між класами.
- Використовується для SVM, але може бути адаптована для нейронних мереж.

Недоліки:

- Не завжди дає найкращі результати для м'яких задач класифікації, де потрібна ймовірнісна інтерпретація.

2. Функції втрат для задач регресії

a) Середня квадратична помилка (MSE, Mean Squared Error)

MSE є стандартною функцією втрат для задач **регресії**, де модель передбачає безперервні значення (наприклад, ціну будинку або температуру). MSE вимірює середню квадратичну різницю між передбаченими значеннями $\hat{y}$ і реальними значеннями y.

Математична формула:

$$E = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad (1.1)$$

де: y_i - реальне значення; $\hat{y}_i$ - передбачене значення.

Переваги:

- Легко інтерпретується.
- Широко використовується для задач регресії.

Недоліки:

- Може бути дуже чутливою до великих помилок (викидів).

b) Середня абсолютна помилка (MAE, Mean Absolute Error)

MAE є альтернативою MSE і вимірює середню абсолютну різницю між передбаченими і реальними значеннями. На відміну від MSE, MAE не підвищує вагу великих помилок.

Математична формула:

$$E = \frac{1}{N} \sum_{i=1}^{N} | y_i - \hat{y}_i | . \qquad (1.2)$$

Переваги:

- Менше чутливий до викидів, ніж MSE.
- Легко інтерпретується.

Недоліки:

- Не штрафує великі помилки так сильно, як MSE.

c) Huber Loss

Huber Loss поєднує переваги MSE та MAE. Вона використовує квадратичні помилки для малих помилок і абсолютні помилки для великих, що дозволяє знизити чутливість до викидів, зберігаючи при цьому більшу стійкість для малих помилок.

Математична формула:

$$E_\delta = \sum_{i=1}^{N} \frac{0,5(y_i - \hat{y}_i)^2}{\delta \mid y_i - \hat{y}_i \mid -0,5\delta^2} \frac{\text{для} \mid y_i - \hat{y}_i \mid \le \delta,}{\text{для} \mid y_i - \hat{y}_i \mid > \delta.}$$

Переваги:

- Збалансує чутливість до викидів і великої помилки.
- Часто використовується при наявності викидів у даних.

Недоліки:

- Потрібно налаштовувати гіперпараметр δ.

Існують інші функції втрат. Правильний вибір функції втрат залежить від специфіки задачі та типу даних, з якими працює модель.

1.7.5 Критерії якості навчання

Критерії якості використовуються для оцінки того, наскільки добре модель виконує свою задачу. Вибір правильного критерію якості залежить від типу задачі (класифікація, регресія, кластеризація тощо) та специфіки проблеми, яку модель намагається вирішити. Ось деякі з найбільш поширених критеріїв якості для різних типів задач:

1. Критерії якості для задач класифікації

a) Точність (Accuracy)

Точність визначає частку правильних передбачень серед усіх передбачених значень.

Переваги:

- Легко розуміти і використовувати.
- Підходить для збалансованих класів.

Недоліки:

- Може бути неефективною, якщо класи сильно незбалансовані (наприклад, при великій кількості негативних прикладів і кількох позитивних).

b) Точність, Повнота та F1-міра

- *Точність (Precision)* - це частка правильно класифікованих позитивних елементів серед всіх елементів, які були класифіковані як позитивні:

- *Повнота (Recall)* - це частка правильно класифікованих позитивних елементів серед всіх справжніх позитивних елементів:
- *F1-міра* - це середнє гармонійне між точністю і повнотою:
  F1-міра є корисною, коли потрібно збалансувати точність і повноту.
  Переваги:
- Чудово підходить для незбалансованих класів, де одна категорія значно більше представлена.
- F1-міра допомагає знайти баланс між точністю та повнотою.
  Недоліки:
- Може не відображати загальну ефективність моделі, якщо класи дуже незбалансовані.

c) Матриця плутанини (Confusion Matrix)

Матриця плутанини дає детальну інформацію про те, як добре модель класифікує дані, зокрема кількість істинно позитивних (TP), хибно позитивних (FP), хибно негативних (FN) і істинно негативних (TN) передбачень.

Матриця плутанини є корисною для розуміння сильних і слабких сторін класифікації.

d) ROC і AUC (Receiver Operating Characteristic і Area Under Curve)

- *ROC (Receiver Operating Characteristic) Curve* - це графік, що показує взаємозв'язок між чутливістю (повнотою) та специфічністю (1 - частка хибно позитивних).
- *AUC (Area Under Curve)* - це площа під ROC-кривою. AUC дає змогу оцінити загальну здатність моделі відрізняти позитивні класи від негативних (чим вище AUC, тим краща модель).

2. Критерії якості для задач регресії

а) Середня квадратична помилка (MSE, Mean Squared Error)

MSE вимірює середню квадратичну помилку між передбаченими значеннями $\hat{y}$ та реальними значеннями y. Обчислюється по формулі (1.1).

Переваги:

- Широко використовується і легко інтерпретується.
- Знижує вплив великих помилок завдяки квадратичному компоненту.
  Недоліки:
- Чутлива до великих відхилень або викидів.

b) Корінь середньої квадратичної помилки (RMSE, Root Mean Squared Error)

RMSE - це корінь із середньої квадратичної помилки, який дає в результаті значення в одиницях вимірювання, схожих на одиниці вхідних даних: $RMSE = \sqrt{MSE}$.

RMSE є зручним для інтерпретації у реальних одиницях.

c) Середня абсолютна помилка (MAE, Mean Absolute Error)

MAE вимірює середню абсолютну різницю між передбаченими та реальними значеннями. Визначається по формулі (1.2)

Переваги:

- Більш стійка до викидів порівняно з MSE.
- Легко інтерпретується, оскільки дає середню абсолютну різницю.

Недоліки:

- Не штрафує великі помилки так сильно, як MSE.

d) R^2 (Коефіцієнт детермінації)

R^2 показує, яку частину варіації в даних модель може пояснити. Він варіюється від 0 до 1, де 1 означає ідеальне пояснення:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2},$$

де $\bar{y}$ - середнє значення реальних даних.

Переваги:

- Чудово підходить для задач регресії.
- Легко інтерпретується: чим вище R^2, тим краще модель підходить для даних.

Недоліки:

- Не завжди коректно оцінює модель, якщо в даних є багато викидів.

3. Критерії якості для кластеризації

a) Індекс Сілуета (Silhouette Score)

Індекс Сілуета вимірює, наскільки добре кожен об'єкт кластеризується. Відношення між відстанню до найближчого сусіднього кластера і середньою відстанню до об'єктів у своєму кластері. Індекс варіюється від -1 (погана кластеризація) до 1 (добра кластеризація).

b) Коефіцієнт Джардіна (Jaccard Index)

Використовується для оцінки подібності між двома наборами, зокрема при кластеризації.

Таким чином, критерії якості для машинного навчання варіюються в залежності від типу задачі. Вибір відповідного критерію якості залежить від мети, яку необхідно досягти, і особливостей ваших даних.

1.8 Архітектура і властивості нейронних мереж прямого розповсюдження

В нейрорегуляторах, які реалізовані у ППП Deep Learning Toolbox викаористовуються багатошарові нейронні мережі прямого розповсюдження, тому доцільно розглянути їх більш детально.

1.8.1. Одношарові мережі

Реальна нейронна мережа може містити один або більшу кількість шарів і відповідно характеризуватися як одношарова або як багатошарова.

Розгорнена схема мережі з одного шару з R вхідними елементами і S нейронами показана на рис.1.12.

У цій мережі кожен елемент вектора входу сполучений зі всіма входами нейрона і це з'єднання задається матрицею вагів $\mathbf{W}$; при цьому кожен i-й нейрон включає підсумовуючий елемент, який формує скалярний вихід $n(i)$. Сукупність скалярних функцій $n(i)$ об'єднується в S-елементний вектор входу $\mathbf{n}$ функції активації шару. Виходи шару нейронів формують вектор-стовпець $\mathbf{a}$, і, таким чином, опис шару нейронів має вигляд:

$$\mathbf{a} = f(\mathbf{W} \cdot \mathbf{p} + \mathbf{b}).$$

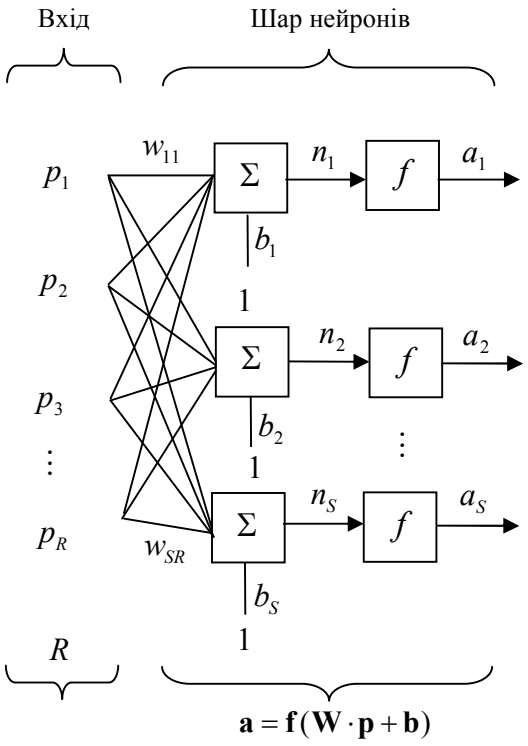


Рис.1.12. Схема одношарової мережі

Кількість входів R в шарі може не співпадати з кількістю нейронів S. У кожному шарі, як правило, використовується одна і та ж функція активації. Проте мож-

на створювати складені шари нейронів з використанням різних функцій активації, сполучаючи мережі, подібні зображеній на рис.1.12, паралельно. Обидві мережі матимуть ті ж самі входи, і кожна мережа генеруватиме певну частину виходів.

Елементи вектора входу передаються в мережу через матрицю вагів $\mathbf{W}$, що має вигляд:

$$\mathbf{W} = \begin{vmatrix} w_{11} & w_{12} & \dots & w_{1R} \\ w_{21} & w_{22} & \dots & w_{2R} \\ \dots & \dots & \dots & \dots \\ w_{S1} & w_{S2} & \dots & w_{SR} \end{vmatrix}.$$

Відмітимо, що індекси рядків матриці $\mathbf{W}$ указують адресатів (пункти призначення) вагів нейронів, а індекси стовпців – яке джерело є входом для цієї ваги. Таким чином, елемент матриці вагів $w_{12} = \mathbf{W}(1,2)$ визначає коефіцієнт, на який умножається другий елемент входу при передачі його на перший нейрон.

Для одношарової мережі з S нейронами укрупнена структурна схема показана рис.1.13.

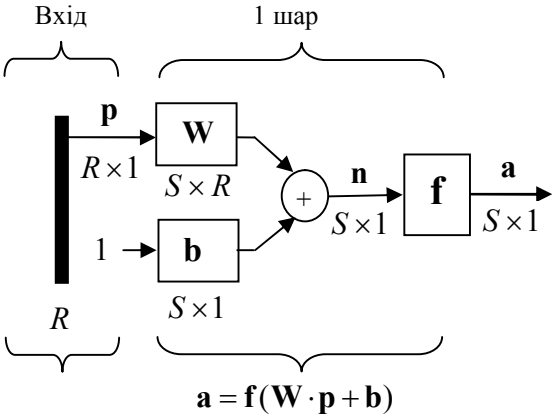


Рис.1.13. Укрупнена структурна схема одношарової мережі

На рисунку позначено: $\mathbf{p}$ – вектор входу розміру $R \times 1$; $\mathbf{W}$ – вагова матриця розміру $S \times R$; $\mathbf{a}$, $\mathbf{b}$, $\mathbf{n}$ – вектори розміру $S \times 1$.

1.8.2. Багатошарові мережі

Розглянемо мережі, що мають декілька шарів. Вагові матриці, сполучені з входами, називаються *вагами вхідного шару*, а вагові матриці для сигналів, витікаючих з шару, називаються *вагами вихідного шару*. Далі,будемо використовувати верхні індекси, щоб вказати джерело і адресат для різних вагів і інших елементів нейронної мережі. Щоб пояснити це, розглянемо спочатку тільки один, перший шар багатошарової мережі (рис.1.14).

Позначимо вагову матрицю, пов'язану з входами, через $\mathbf{IW}^{11}$, верхні індекси якої указують, що джерелом входів є перший шар (другий індекс) і адресатом є також перший шар (перший індекс). Елементи цього шару, такі, як зсув $\mathbf{b}^1$, вхід функції активації $\mathbf{n}^1$ і вихід шару $\mathbf{a}^1$ мають верхній індекс 1, щоб позначити, що вони пов'язані з першим шаром. Надалі для матриць вагів входу і виходу шару будуть використані позначення **IW** (Input Weight) і **LW** (Layer Weight) відповідно.

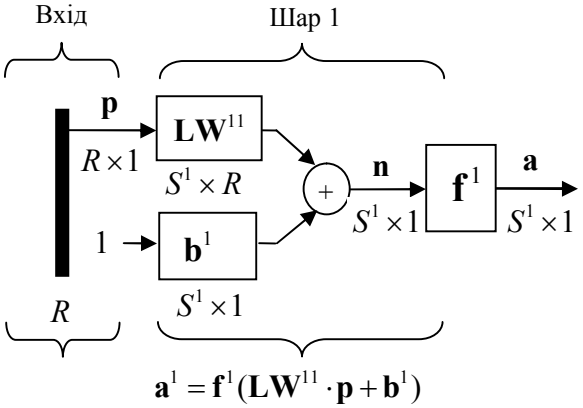


Рис.1.14. Структурна схема першого шару багатошарової мережі

Коли мережа має декілька шарів, то кожен шар має свою матрицю вагів $\mathbf{W}$, вектор зсуву $\mathbf{b}$ і вектор виходу $\mathbf{a}$. Щоб розрізняти вагові матриці, вектори виходу і т.д. для кожного з цих шарів, введемо номер шару як верхній індекс для представляючої інтерес змінної. Використання цієї системи позначень для мережі з трьох шарів можна бачити на показаній нижче структурній схемі і з рівнянь, приведених в нижній частині рис.1.15.

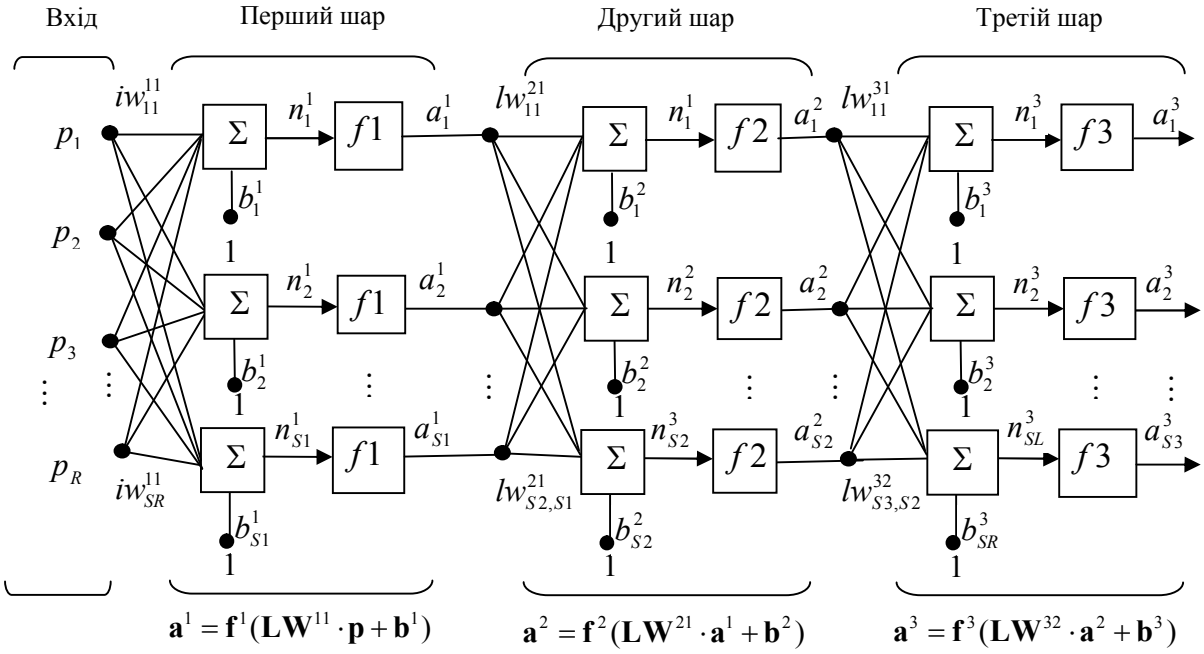


Рис.1.15. Структурна схема багатошарової мережі

Мережа, показана на рис.115, має R входів, S^1 нейронів в першому шарі, S^2 нейронів в другому шарі і т.д. Для спільності вважатимемо, що різні шари мають різне число нейронів. На зсуви для кожного нейрона поданий постійний вхідний сигнал 1. Відмітимо, що виходи кожного проміжного шару служать входами для наступного шару. Таким чином, шар 2 може бути розглянутий як один шар мережі з S^1 входами S^2 нейронами і $S^1 \times S^2$ матрицею вагів $\mathbf{W}_2$. Вхід до шару 2 є 1, а вихід – 2. Тепер, коли позначені всі вектори і матриці шару 2, можна трактувати його як самостійну одношарову мережу. Такий підхід може бути використаний до будь-якого шару мережі.

Шари багатошарової мережі мають різні призначення. Шар, який утворює вихід мережі, називається *шаром виходу*. Всі інші шари називаються *прихованими шарами.* Тришарова мережа, показана вище, має вихідний шар (шар 3) і 2 прихованих шари (шар 1 і шар 2). Ця ж тришарова мережа може бути представлена у вигляді укрупненої структурної схеми (рис.1.16).

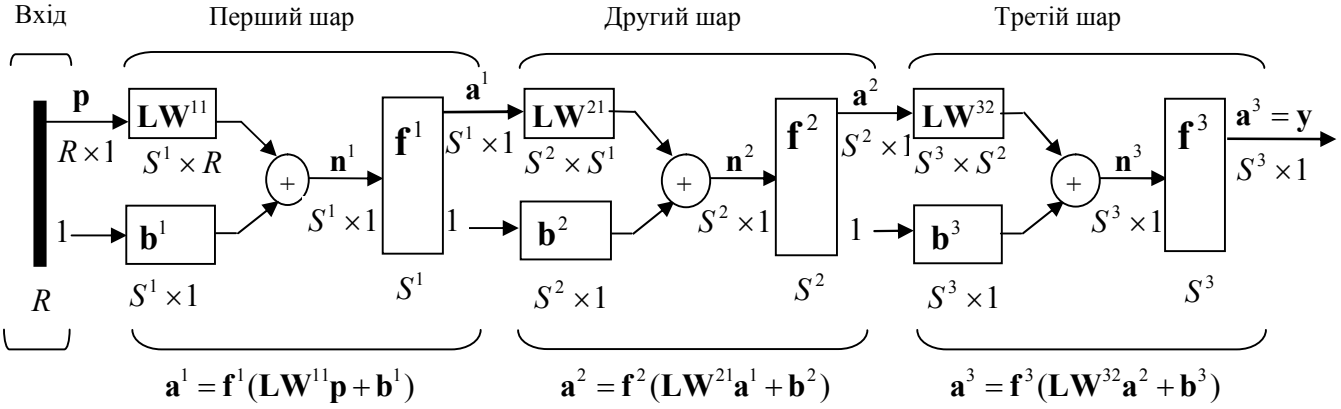


Рис.1.16. Укрупнена структурна схема багатошарової мережі

Відмітимо, що вихід третього шару $\mathbf{a}^3$ позначений через **y**. Ця зроблено для того, щоб підкреслити, що вихід останнього шару є виходом мережі.

Багатошарові мережі володіють вельми потужними можливостями.

1.8.3 Мережі прямого розповсюдження з сигмоїдальними функціями активації

Як зазначалося, широке застосування отримали сигмоїдальні функції активації. Одношарова мережа з S нейронами з функціями активації **logsig**, що має R входів, показана на рис.1.17.

Ця мережа, що не має зворотних зв'язків, називається *мережею з прямою передачею сигналу*. Такі мережі часто мають один або більше прихованих шарів нейронів з сигмоїдальними функціями активації, тоді як вихідний шар містить нейрони з *лінійними* функціями активації. Мережі з такою архітектурою можуть відтворювати вельми складні нелінійні залежності між входом і виходом мережі.

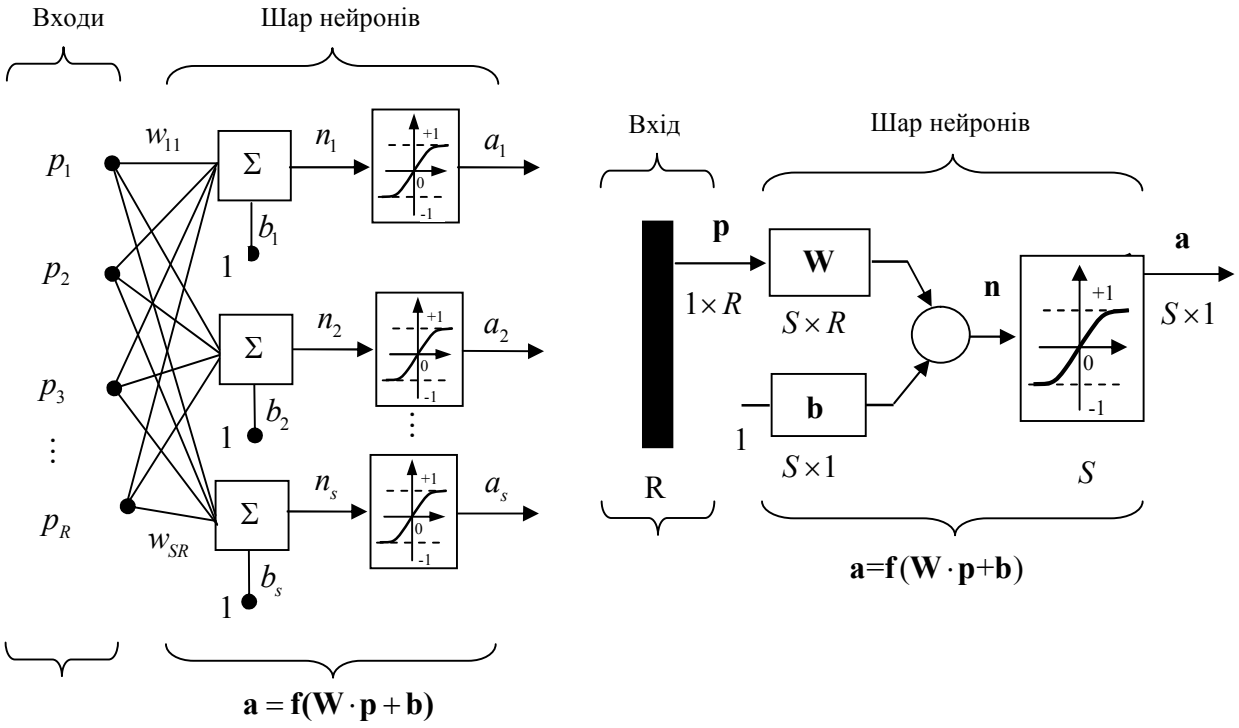


Рис.1.17. Структурна схема одношарової мережі з R входами

Двошарова мережа показана на рис.1.18. Двошарова мережа, в якій перший шар містить сигмоїдальну, а другий шар – лінійну функцію активації, може бути навчена апроксимувати з довільною точністю будь-яку функцію з кінцевим числом точок розриву, якщо задати достатнє число нейронів прихованого шару.

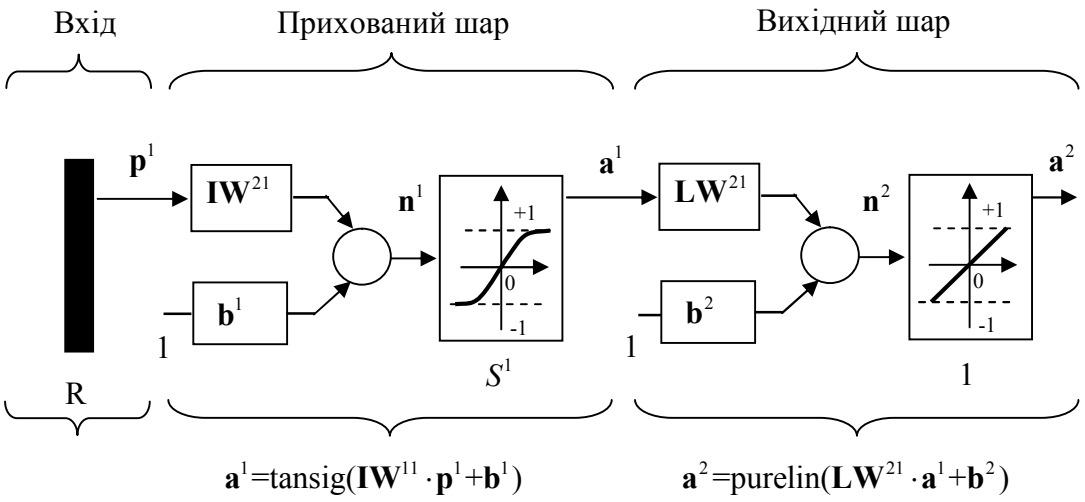


Рис.1.18. Структурна схема двошарової мережі з R входам

На закінчення можна сформулювати наступні висновки.

Вхід функції активації нейрона визначається зсувом і сумою зважених входів. Вихід нейрона залежить як від входів нейрона, так і від виду функції активації. Один нейрон не може вирішувати складні завдання, проте декілька нейронів, об'єднаних в один або декілька шарів, володіють великими можливостями.

Архітектура мережі складається з опису того, скільки шарів має мережа, кількості нейронів в кожному шарі, виду функції активації кожного шару і інформації про з'єднання шарів. Архітектура мережі залежить від того конкретного завдання, яке повинна вирішувати мережа.

Робота мережі полягає в обчисленні виходів мережі на основі відомих входів з метою формування бажаного відображення вхід/вихід. Конкретне завдання визначає число входів і число виходів мережі. Окрім числа нейронів у вихідному шарі мережі, для проектувальника важливе число нейронів в кожному шарі. Більша кількість нейронів в прихованих шарах забезпечує потужнішу мережу. Якщо повинно бути реалізоване лінійне відображення, то слід використовувати нейрони з лінійними функціями активації. При цьому треба пам'ятати, що лінійні нейронні мережі не можуть формувати нелінійні відображення. Використання нелінійних функцій активації дозволяє налаштувати нейронну мережу на реалізацію нелінійних зв'язків між входом і виходом.

Мережі із зсувом дозволяють формувати складніші зв'язки між входами і виходами, ніж мережі без зсуву. Наприклад, нейрон без зсуву, коли всі входи нульові, завжди задаватиме вхід функції активації рівним нулю, проте нейрон із зсувом може бути навчений так, щоб за тих же умов задати вхід функції активації довільної форми.

У багатошарових мережах часто застосовуються нелінійні сигмоїдальні функції активації типа логістичної або гіперболічного тангенса. Якщо останній шар багатошарової мережі використовує такі функції активації, то виходи мережі будуть обмежені. Коли у вихідному шарі використовуються лінійні нейрони, то виходи мережі можуть приймати довільні значення.

Питання для самоконтролю.

1. Що називається нейронною мережею?
2. Які найважливіші особливість та властивості нейронних мереж?
3. Дайте визначення нейромережевої системи управління.
4. Які функції виконує нейронна мережа у складі автоматизованих систем управління?
5. Перерахуйте області застосування штучних нейронних мереж.
6. Які задачі вирішуються за допомогою штучних нейронних мереж?
7. Дайте характеристику основним етапам розвитку нейронних мереж.
8. Які фактори обумовлюють застосування нейромережевих технологій в системах управління?
9. Наведіть структурну схема і запишіть основні рівняння штучного нейрона з скалярним входом (без зсуву і зі зсувом).
10. Які ви знаєте види функцій активації нейронів? Запишіть їх формули та намалюйте графіки.
11. Наведіть структурну схему і запишіть основні рівняння штучного нейрона з векторним входом.
12. Наведіть укрупнену схема нейрона з векторним входом. Дайте пояснення основних елементів схеми.
13. У чому полягає алгоритм функціонування мережі.
14. Назвіть етапи формування нейронної мережі.
15. Які дії слід виконаним на кожному етапі формування нейронної мережі.
16. Назвіть типи нейронів в мережі в залежності від функцій, що вони виконують.
17. Перечисліть і дайте характеристику класичним нейронним мережам.
18. Дайте визначення глибоких нейронних мереж і глибокого навчання.
19. Перечисліть і дайте характеристику глибоких нейронних мереж
20. Охарактеризуйти методи навчання глибинних нейронних мереж
21. Що таке функції втрат і критерії якості навчання? Яка між ними різниця?
22. Зобразіть схему і запишіть рівняння одношарової нейронної мережі.
23. Який вигляд має матриця вагів?
24. Зобразіть укрупнену структурну схему одношарової нейронної мережі.
25. Зобразіть розгорнуту і укрупнену схему багатошарової нейронної мережі.
26. Зобразіть розгорнуту і укрупнену схему багатошарової нейронної мережі з сигмоїдальними функціями активації.

НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

2.1 Сутність процесу навчання

При рішенні за допомогою нейронних мереж прикладних завдань необхідно зібрати точний і представницький об'єм даних для того, щоб навчити нейронну мережу рішенню таких задач. Навчальний набір даних – це набір спостережень, що містять ознаки об'єкту, який вивчається. Перше питання: які ознаки використовувати і скільки і які спостереження треба провести?

Вибір ознак, принаймні первинний, здійснюється евристично на основі наявного досвіду, який може підказати, які ознаки є найбільш важливими. Спочатку слід включити всі ознаки, які, на думку аналітиків або експертів, є істотними, на подальших етапах ця множина буде скорочена.

Нейронні мережі працюють з числовими даними, узятими, як правило, з деякого обмеженого діапазону. Це може створити проблеми, якщо значення спостережень виходять за межі цього діапазону або пропущені.

Питання про те, скільки потрібно мати спостережень для навчання мережі, часто виявляється непростим. Відомий ряд евристичних правил, які встановлюють зв'язок між кількістю необхідних спостережень і розмірами мережі. Просте з них свідчить, що кількість спостережень повинна бути в 10 разів більше числа зв'язків в мережі. Насправді це число залежить від складності того відображення, яке повинна відтворювати нейронна мережа. Із зростанням числа використовуваних ознак кількість спостережень зростає по нелінійному закону, так що вже при досить невеликому числі ознак, скажемо 50, може бути потрібно величезне число спостережень. Ця проблема носить назву "Прокляття розмірності".

Для більшості реальних завдань буває достатнім декількох сотень або тисяч спостережень. Для складних завдань може бути потрібно більша кількість, проте дуже рідко зустрічаються завдання, де потрібно менше 100 спостережень. Якщо даних мало, то мережа не має достатньої інформації для навчання, і краще, що можна в цьому випадку зробити, – це спробувати підігнати до даних деяку лінійну модель.

Після того, як визначена кількість шарів мережі і число нейронів в кожному з них, потрібно призначити значення вагів і зсувів, які мінімізують помилку рішення. Це досягається за допомогою **процедур навчання**. Навчання мережі відбувається відповідно до схеми, показаної на рис.2.1.

Шляхом аналізу тих вхідних і вихідних даних, що є у розпорядженні аналітика, ваги і зсуви мережі автоматично настроюються так, щоб мінімізувати різницю між бажаним сигналом і отриманим на виході мережі в результаті моделювання. Ця різниця носить назву **помилки навчання**. Таким чином, процес навчання – це про-

цес підгонки параметрів тієї моделі процесу або явища, яка реалізується нейронною мережею. Помилка навчання для конкретної конфігурації нейронної мережі визначається шляхом прогону через мережу всіх наявних спостережень і порівняння вихідних значень з бажаними, цільовими значеннями. Ці різниці дозволяють сформувати так звану функцію помилок (критерій якості навчання). Як така функція найчастіше береться сума квадратів помилок. При моделюванні нейронних мереж з лінійними функціями активації нейронів можна побудувати алгоритм, що гарантує досягнення абсолютного мінімум помилки навчання. Для нейронних мереж з нелінійними функціями активації в загальному випадку не можна гарантувати досягнення глобального мінімуму функції помилки.

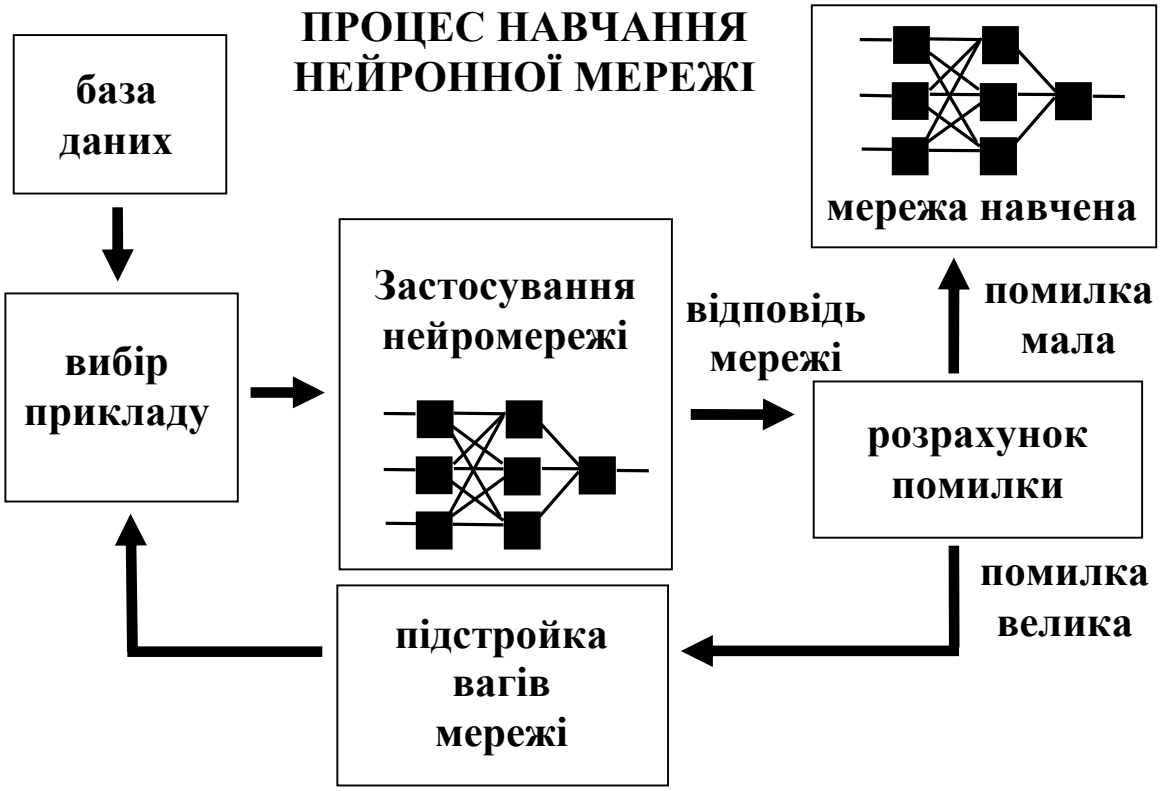


Рис.2.1 Ілюстрація процесу навчання ШНМ

При такому підході до процедури навчання може виявитися корисним геометричний аналіз поверхні функції помилок. Визначимо ваги і зсуви як вільні параметри моделі і їх загальне число позначимо через N; кожному набору таких параметрів поставимо у відповідність одне вимірювання у вигляді помилки мережі. Тоді для всіляких поєднань вагів і зсувів відповідну помилку мережі можна зобразити точкою в $N+1$-вимірному просторі, а всі такі точки утворюють деяку поверхню, звану поверхнею функції помилок. При такому підході мета навчання нейронної мережі полягає в тому, щоб знайти на цій багатовимірній поверхні глобальний мінімум.

У разі лінійної моделі мережі і функції помилок у вигляді суми квадратів така поверхня буде параболоїдом, який має єдиний мінімум і це дозволяє відшукати такий мінімум досить просто.

У разі нелінійної моделі поверхня помилок має набагато складнішу будову і має ряд несприятливих властивостей, зокрема може мати локальні мінімуми, плоскі ділянки, сідлові точки і довгі вузькі яри.

Визначити глобальний мінімум багатовимірної функції аналітично неможливо і тому навчання нейронної мережі, по суті справи, є процедурою вивчення поверхні функції помилок. Відштовхуючись від випадково вибраної точки на поверхні функції помилок, алгоритм навчання поступово відшукує глобальний мінімум. Як правило, для цього обчислюється градієнт (нахил) функції помилок в даній точці, а потім ця інформація використовується для просування вниз по схилу. Врешті-решт алгоритм зупиняється в деякому мінімумі, який може виявитись лише локальним мінімумом, а якщо повезе, то і глобальним.

Таким чином, по суті алгоритми навчання нейронних мереж аналогічні алгоритмам пошуку глобального екстремуму функції багатьох змінних. Серед останніх слід виділити алгоритми зв'язаних градієнтів і Левенберга-Марквардта (Levenberg-Marquardt).

З урахуванням специфіки нейронних мереж для них розроблені спеціальні алгоритми навчання, серед яких слід виділити алгоритм зворотного розповсюдження помилки. При використанні алгоритму зворотного розповсюдження помилки мережа розраховує помилку, що виникає у вихідному шарі, і обчислює вектор градієнта як функцію вагів і зсувів. Цей вектор указує напрям найкоротшого спуску по поверхні для даної точки, тому якщо просунутися в цьому напрямі, то помилка зменшиться. Послідовність таких кроків врешті-решт приведе до мінімуму того або іншого типу. Певну трудність тут викликає вибір величини кроку.

При великій довжині кроку збіжність буде швидшою, але є небезпека перестрибнути через рішення або піти в неправильному напрямі. Класичним прикладом такого явища при навчанні нейронної мережі є ситуація, коли алгоритм дуже поволі просувається по вузькому яру з крутими схилами, перестрибуючи з одного схилу на інший. Навпаки, при малому кроці, ймовірно, буде вибрано вірний напрям, проте при цьому буде потрібно дуже багато ітерацій. На практиці величина кроку вибирається пропорційно крутизні схилу (градієнту функції помилок); такий коефіцієнт пропорційності називається параметром швидкості настройки. Правильний вибір параметра швидкості настройки залежить від конкретного завдання і зазвичай здійснюється дослідним шляхом; цей параметр може також залежати від часу, зменшуючись у міру виконання алгоритму.

Алгоритм діє ітеративно, і його кроки прийнято називати **епохами** або **циклами**. На кожному циклі на вхід мережі послідовно подаються всі навчальні спостереження, вихідні значення порівнюються з цільовими значеннями і обчислюється функція помилки. Значення функції помилки, а також її градієнта використовуються для коректування вагів і зсувів, після чого всі дії повторюються. Початкові значення вагів і зсувів мережі вибираються випадковим чином і процес навчання припиняється або коли реалізована певна кількість циклів, або коли помилка досягне деякого малого значення або перестане зменшуватися.

2.2 Явище перенавчання

Одна з найбільш серйозних труднощів при навчанні мережі полягає в тому, що у ряді випадків ми мінімізуємо не ту помилку, яку насправді потрібно мінімізувати. Потрібно мінімізувати помилку, яка з'являється в мережі, коли на неї подаються абсолютно нові спостереження. Вельми важливо, щоб нейронна мережа володіла здатністю пристосовуватися до цих нових спостережень. Що ж відбувається насправді? Мережа навчається мінімізувати помилку на деякій обмеженій навчальній множині. Це не відповідає вимогам теорії про наявність ідеальної і нескінченно великої навчальної множини. І це не відповідає тій реальній ситуації, коли треба мінімізувати конкретну функцію помилок для наперед невідомої моделі.

Це породжує проблему, яка відома як явище **перенавчання**. Звернемося до завдання апроксимації деякої функції многочленом. Графіки многочленів часто мають вельми хитромудрі форми, і чим вище ступінь многочлена, тим складніше їх форма. Якщо є деякий набір даних, то можна поставити мету підібрати для нього апроксимуючий многочлен і таким чином отримати відповідну математичну модель для цього набору даних. Оскільки початкові дані, як правило, задані з погрішностями, то не можна вважати, що краща модель задається кривою, яка проходить точно через задані точки. Многочлен низького порядку може виявитися достатньо грубим для апроксимації даних, тоді як многочлен високого порядку може точно слідувати даним, приймаючи при цьому вельми хитромудру форму, що не має ніякого відношення до форми дійсної залежності. Остання ситуація і демонструє те, що називається явищем перенавчання.

При роботі з нейронними мережами користувач стикається з тією ж проблемою. Мережі з великою кількістю вагів дозволяють відтворювати дуже складні функції, і в цьому сенсі вони схильні до перенавчання. Мережа ж з невеликою кількістю вагів може опинитися недостатньо гнучкою, щоб змоделювати наявну залежність. Наприклад, одношарова лінійна мережа здатна відтворювати тільки лінійні функції. Якщо використовувати багатошарові лінійні мережі, то помилка завжди буде менша, але може свідчити не про хорошу якість моделі, а про те, що виявляється явище перенавчання.

Для того, щоб виявити ефект перенавчання, використовується механізм контрольної перевірки. Частина навчальних спостережень резервується як контрольні спостереження і не використовується при навчанні мережі. Натомість у міру роботи алгоритму ці спостереження застосовуються для незалежного контролю результату. Спочатку помилка мережі на навчальній і контрольній множинах буде однаковою; якщо вони істотно відрізняються, то, мабуть, це означає, що розбиття спостережень на 2 множини не забезпечило їх однорідність. У міру навчання мережі помилка убуває, і, поки навчання зменшує функцію помилок, помилка на контрольній множині також убуватиме. Якщо ж контрольна помилка перестала убувати або стала зростати, це указує на те, що мережа почала дуже близько слідувати початковим даним і навчання слід зупинити. В цьому випадку слід зменшити кількість нейронів або шарів, бо мережа є дуже потужною для вирішення даного завдання Якщо ж, навпаки,

мережа має недостатню потужність, щоб відтворити наявну залежність, то явище перенавчання швидше за все спостерігатися не буде і обидві помилки – навчання і перевірки – не досягнуть достатньо малого рівня.

Проблеми відшукання глобального мінімуму або вибору розміру мережі, що виникають при роботі з нейронними мережами, призводять до того, що при практичній роботі доводиться експериментувати з великим числом мереж різних конфігурацій, інколи навчаючи кожну з них кілька разів і порівнюючи отримані результати. Головним критерієм вибору в цих випадках є контрольна погрішність. При цьому застосовується правило, згідно якого з двох нейронних мереж з приблизно рівними контрольними погрішностями слід вибирати ту, яка простіша.

Необхідність багатократних експериментів веде до того, що контрольна множина починає грати ключову роль у виборі моделі нейронної мережі, тобто стає частиною процесу навчання. Тим самим її роль як незалежного критерію якості моделі ослабляється, оскільки при великому числі експериментів виникає ризик перенавчання нейронної мережі на контрольній множині. Для того, щоб гарантувати надійність вибираної моделі мережі, резервують ще одну – тестову множину спостережень. Підсумкова модель тестується на даних з цієї множини, щоб переконатися, що результати, досягнуті на навчальній і контрольній множинах реальні. Зрозуміло, для того, щоб добре грати свою роль, тестова множина повинна бути використана тільки 1 раз: якщо її використовувати повторно для коректування процесу навчання, то вона фактично перетвориться на контрольну множину.

2.3 Здібність узагальнення

При описі процедури навчання нейронних мереж неявно використовувалося припущення, що навчальна, контрольна і тестова множини є представницькими для вирішуваної задачі. Зазвичай як навчальні беруться дані, випробувані на ряді прикладів. Якщо обставини змінилися, то закономірності, що мали місце у минулому, можуть більше не діяти.

Крім того, нейронна мережа може навчатися тільки на тих даних, які вона має в своєму розпорядженні. Припустимо, що відома навчальна множина для системи стабілізації літака при польоті в спокійній атмосфері, а потрібно спроектувати систему стабілізації на основі нейронної мережі для умов польоту при сильних збуреннях. Тоді навряд чи можна чекати від мережі правильного рішення в абсолютно новій для неї ситуації.

Класичним прикладом непредставницької моделі нейронної мережі є наступна ситуація. При проектуванні системи машинного зору, призначеної для автоматичного розпізнавання цілей, мережа навчалася на 100 картинках, що містять зображення танків, і на 100 інших картинках, де танків не було. Після навчання мережі був досягнутий стовідсотково "правильний" результат. Але коли на вхід мережі були подані нові дані, вона безнадійно провалилася. У чому ж була причина? З'ясувалося, що фотографії з танками були зроблені в похмурий, дощовий день, а фотографії без та-

нків – в сонячний день. Мережа навчилася уловлювати різницю в загальній освітленості. Щоб мережа могла результативно працювати, її слід було навчати на даних, де б були присутні всі погодні умови і типи освітлення, при яких мережу передбачається використовувати, і це не кажучи ще про рельєф місцевості, кут і дистанцію зйомки і т.д.

Якщо мережа мінімізує загальну погрішність, великого значення набувають пропорції, в яких представлені дані різних типів. Мережа, навчена на 900 "хороших" і 100 "поганих" спостереженнях, спотворюватиме результат на користь хороших спостережень, оскільки це дозволить алгоритму зменшити загальну погрішність. Якщо в реальній ситуації "хороші" і "погані" об'єкти представлені в іншій пропорції, то результати, що видаються мережею, можуть виявитися невірними. Прикладом цього може бути завдання виявлення захворювань. Хай, наприклад, при звичайних обстеженнях в середньому 90% людей виявляються здоровими і мережа, таким чином, навчається на даних, в яких пропорція здорові/хворі рівна 90/10. Потім ця ж мережа застосовується для діагностики пацієнтів з певними скаргами, серед яких співвідношення здорові/хворі вже 50/50. В цьому випадку мережа ставитиме діагноз занадто обережно і не розпізнаватиме захворювання у деяких хворих. Якщо ж, навпаки, мережу навчити на даних "з скаргами", а потім протестувати на "звичайних" даних, то вона видаватиме підвищене число неправильних діагнозів про наявність захворювання. У таких ситуаціях навчальні дані потрібно скоректувати так, щоб були враховані відмінності в розподілі даних (наприклад, можна повторити рідкісні спостереження або видалити ті, що часто зустрічаються). Як правило, краще всього постаратися зробити так, щоб спостереження різних типів були представлені рівномірно, і відповідно цьому інтерпретувати результати, які видає мережа.

Здатність мережі, навченої на деякій множині даних, видавати правильні результати для достатньо широкого класу нових даних, у тому числі і не представлених при навчанні, називається **властивістю узагальнення** нейронної мережі.

Інший підхід до процедури навчання мережі можна сформулювати, якщо розглядати її як процедуру, зворотну моделюванню. В цьому випадку потрібно підібрати такі значення вагів і зсувів, які забезпечували б потрібну відповідність між входами і бажаними значеннями на виході. Така процедура навчання носить назву адаптації і достатньо широко застосовується для настройки параметрів нейронних мереж.

2.4 Критерій якості навчання

Фаза навчання є найбільш проблематичною при використанні нейронних мереж. Для багатошарових мереж процес навчання є багатозмінним нелінійним оптимізаційним завданням. Його аналітичне рішення неможливе, що вимагає використання ітераційних методів оптимізації.

У загальному випадку початковими даними є: структура мережі, тренувальні дані і критерій оцінки якості тренування. Мета оптимізації – визначити параметри

мережі, що забезпечують виконання критерію якості. Таким критерієм якості найчастіше виступає мінімум квадратичного функціонала:

$$J = \frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^L}(t_i^q - a_i^{qL})^2 ,$$

де J – функціонал;

Q – об'єм вибірки;

L – число шарів мережі;

q – номер вибірки;

S^L – число нейронів вихідного шару;

$\mathbf{a}^q = [a_i^{qL}]$ – вектор сигналу на виході мережі;

$\mathbf{t}^q = [t_i^q]$ – вектор бажаних (цільових) значень сигналу на виході мережі для вибірки з номером q.

Потім за допомогою того або іншого методу навчання визначаються значення параметрів (вагів і зсувів), що настроюються, мережі, які забезпечують мінімальне значення функціонала помилки.

Методика тренування нейронних мереж була в основному запозичена з відомих методів теорії оптимізації. Зважаючи на величезне число створених алгоритмів тут можна дати лише їх короткий огляд. Вживані методи навчання грубо можна розділити на наступні основні групи: методи глобальної оптимізації; градієнтні методи першого і другого порядку.

Істотною проблемою, що виникає при використанні градієнтних методів настройки параметрів ШНМ, як зазначалося, є їх локальність. В той же час цільова функція (сумарна помилка на тренувальному наборі шаблонів або кумулятивна оцінка ефективності роботи навчаної системи) не унімодальна. Кількість локальних оптимумів для більшості практичних завдань навчання обчислюється мільйонами при розмірності пошукового простору порядку $10^2 - 10^3$. Внаслідок цього результат навчання залежить від правильності вибору стартової точки, і виникає необхідність багатократного повторення процедури настройки параметрів ШНМ. Перераховані проблеми можуть бути вирішені при використанні методів глобальної оптимізації. Найбільш ефективним з них є генетичний алгоритм (ГА) [51-54]. Розглядаючи ШНМ як єдиний набір параметрів, ГА здатний здійснювати її оптимальну настройку при розмірності пошукового простору достатньою для вирішення більшості практичних завдань. Розроблено багато способів навчання ШНМ за допомогою ГА. Отримані результати доводять великі можливості такого симбіозу. Сумісне використання ШНМ і ГА має і ідеологічну перевагу тому, що вони відносяться до методів еволюційного моделювання і розвиваються в рамках одного підходу, запозичених технікою природних методів і механізмів як найбільш оптимальних.

Слід зазначити, що завдання глобальної оптимізації вирішуються шляхом систематичного перебору всіх значень всіх аргументів, тому із-за експоненціального зростання складності завдання із зростанням її розмірності практичне застосування цих методів для великих мереж дуже утруднено.

У сучасних завданнях використовуються в основному градієнтні методи навчання. Вони відносно швидко знаходять оптимум. Проте при цьому відсутній доказ того, що знайдений абсолютний екстремум. Численний досвід тренування мереж свідчить, що застосування методів локальної оптимізації сумісно з «виштовхуванням» мережі з локального мінімуму і послідовним збільшенням числа нейронів також приводить до успішного навчання мережі.

2.5 Навчання одношарової мережі

Найпростіше градієнт функціонала обчислюється для одношарових нейронних мереж. В цьому випадку $L=1$ і вираз для функціонала приймає вигляд:

$$J=\frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^1}(t_i^q-a_i^q)^2=\frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^1}(t_i^q-f(n_i^q))^2 \quad i=\overline{1,S}\,, \tag{2.1}$$

де $f(n_i^q)$ – функція активації;

$n_i^q=\sum_{j=0}^{R}w_{ij}p_j^q$ – сигнал на вході функції активації для i-го нейрона;

$\mathbf{p}^q=[p_i^q]$ – вектор вхідного сигналу;

R – число елементів вектора входу;

S – число нейронів в шарі;

w_{ij} – вагові коефіцієнти мережі.

Включимо вектор зсуву до складу матриці вагів $\mathbf{W}=[w_{ij}]$, $i=\overline{1,S}$, $j=\overline{1,R}$, а вектор входу доповнимо елементом, рівним 1.

Застосовуючи правило диференціювання складної функції, обчислимо градієнт функціонала помилки, припускаючи при цьому, що функція активації диференцируєма

$$\frac{\partial J}{\partial w_{ij}}=-\sum_{q=1}^{Q}(t_i^q-f(n_i^q))\frac{\partial(f(n_i^q))}{\partial w_{ij}}=-\sum_{q=1}^{Q}(t_i^q-f(n_i^q))f'(n_i^q)p_j^q\,. \tag{2.2}$$

Введемо позначення

$$\Delta_i^q=(t_i^q-f(n_i^q))f'(n_i^q)p_j^q=(t_i^q-a_i^q)f'(n_i^q)p_j^q,\ \ i=\overline{1,S}\,,$$

і перетворимо вираз (2.2) таким чином:

$$\frac{\partial J}{\partial w_{ij}} = -\sum_{q=1}^{Q} \Delta_i^q f'(n_i^q) p_j^q, \quad i = \overline{1,S}.$$

Отримані вирази спрощуються, якщо мережа лінійна. Оскільки для такої мережі виконується співвідношення $a_i^q = n_i^q$, то справедливою є умова $f'(n_i^q) = 1$. В цьому випадку вираз (2.2) приймає вигляд:

$$\frac{\partial J}{\partial w_{ij}} = (t_i^q - a_i^q) p_j^q, \quad i = \overline{1,S}, \; j = \overline{0,R}. \tag{2.3}$$

Вираз (2.3) покладений в основу алгоритму, вживаного для навчання лінійних нейронних мереж.

Лінійні мережі можуть бути навчені і без використання ітераційних методів, а шляхом рішення наступної системи лінійних рівнянь:

$$\sum_{j=0}^{R} w_{ij} p_j^q = t_i^q, \quad i = \overline{1,S}, \quad q = \overline{1,Q}, \tag{2.4}$$

або у векторній формі запису:

$$\mathbf{Wp} = \mathbf{t}, \; \mathbf{W}=[w_{ij}], \; \mathbf{p}=[p_j^q], \; \mathbf{t}=[t_i^q], \; i = \overline{1,S}, \quad j = \overline{0,R}, \quad q = \overline{1,Q}.$$

Якщо число невідомих системи (2.4) дорівнює числу рівнянь, то така система може бути вирішена, наприклад, методом виключення Гауса з вибором головного елементу. Якщо ж число рівнянь перевищує число невідомих, то рішення шукається з використанням методу найменших квадратів.

2.6 Навчання багатошарової мережі

Архітектура багатошарової мережі істотно залежить від вирішуваної задачі. Для лінійних нейронних мереж може бути встановлений зв'язок між сумарною кількістю вагів і зсувів з довжиною навчальної послідовності. Для інших типів мереж число шарів і нейронів в шарі часто визначається досвідом, інтуїцією проектувальника і евристичними правилами.

Навчання мережі включає декілька кроків:

- вибір початкової конфігурації мережі з використанням, наприклад, наступного евристичного правила: кількість нейронів проміжного шару визначається половиною сумарної кількості входів і виходів;
- проведення ряду експериментів з різними конфігураціями мережі і вибір тієї, яка дає мінімальне значення функціонала помилки;
- якщо якість навчання недостатня, слід збільшити число нейронів шару або кількість шарів;

- якщо спостерігається явище перенавчання, слід зменшити число нейронів в шарі або видалити один або декілька шарів.

Нейронні мережі, призначені для вирішення практичних завдань, можуть містити до декількох тисяч параметрів, що настроюються, тому обчислення градієнта може вимагати вельми великих витрат обчислювальних ресурсів. З урахуванням специфіки багатошарових нейронних мереж для них розроблені спеціальні методи розрахунку градієнта, серед яких слід виділити метод зворотного розповсюдження помилки, який був розроблений в 1986г [55]. Потім з'явилися численні роботи по прискоренню збіжності алгоритму. Метод зворотного розповсюдження помилки є першим теоретично доведеним алгоритмом для настройки вагів при навчанні багатошарового персептрона. В даний час метод зворотного розповсюдження помилки може розглядатися як еталонний тест, з яким порівнюються всі інші методи навчання.

2.7 Метод зворотного розповсюдження помилки

Термін "зворотне розповсюдження" відноситься до процесу, за допомогою якого можуть бути обчислені похідні функціонала помилки по параметрах мережі. Цей процес може використовуватися у поєднанні з різними стратегіями оптимізації. Існує багато варіантів і самого алгоритму зворотного розповсюдження. Звернемося до одного з них.

Розглянемо вираз для градієнта критерію якості по вагових коефіцієнтах для вихідного шару L:

$$\frac{\partial J}{\partial w_{ij}^L} = \frac{\partial}{\partial w_{ij}^L}\left(\frac{1}{2}\sum_{q=1}^{Q}\sum_{k=1}^{S^L}(t_k^q - a_k^{qL})^2\right) = -\sum_{q=1}^{Q}\sum_{k=1}^{S^L}(t_k^q - a_k^{qL})\frac{\partial a_k^{qL}}{\partial w_{ij}^L}\ ;\quad i=\overline{1,S^L};\quad j=\overline{0,S^{L-1}},\quad (2.5)$$

де S^L – число нейронів в шарі;

a_k^{qL} – k-й елемент вектора виходу шару L для елементу вибірки з номером q.

Правило функціонування шару L записується так:

$$a_k^{qL} = f_L\left(\sum_{l=0}^{S^{L-1}} w_{ij}^L a_l^{q(L-1)}\right),\quad k=\overline{1,S^L}\quad . \qquad (2.6)$$

З рівняння (2.6) виходить

$$\frac{\partial a_k^{qL}}{\partial w_{ij}^L} = \begin{cases} 0, & k \neq i \\ f_L'(n_i^{qL})a_j^{q(L-1)}, & k = i \end{cases} \qquad (2.7)$$

$$i=\overline{1,S^L};\quad j=\overline{0,S^{L-1}}\ .$$

Після підстановки (2.7) в (2.5) маємо:

$$\frac{\partial J}{\partial w_{ij}^{L}} = -\sum_{q=1}^{Q}(t_i^q - a_i^{qL}) f_L'(n_i^{qL}) a_j^{q(L-1)} .$$

Якщо позначити

$$\Delta_i^{qL} = (t_i^q - a_i^{qL}) f_L'(n_i^{qL}), \quad i = 1,...,S^L ,$$

то отримаємо

$$\frac{dJ}{dw_{ij}^{L}} = -\sum_{q=1}^{Q}\Delta_i^{qL} a_j^{q(L-1)} ; \quad i = \overline{1, S^L} ; \quad j = \overline{0, S^{L-1}} .$$

Перейдемо до виведення співвідношень для настройки вагів w_{ij}^{L-1} шару $L-1$:

$$\frac{\partial J}{\partial w_{ij}^{L-1}} = -\sum_{q=1}^{Q}\sum_{k=1}^{S^L}(t_k^q - a_k^{qL})\frac{\partial a_k^{qL}}{\partial n_k^{qL}}\frac{\partial n_k^{qL}}{da_i^{q(L-1)}}\frac{\partial a_i^{q(L-1)}}{\partial n_i^{q(L-1)}}\frac{\partial n_i^{q(L-1)}}{\partial w_{ij}^{L-1}} =$$

$$= -\sum_{q=1}^{Q}\sum_{k=1}^{S^L}(t_k^q - a_k^{qL}) f_L'(n_k^{qM}) w_{ki}^{L} f_{L-1}'(n_i^{q(L-1)})\ a_j^{q(L-2)} =$$

$$= -\sum_{q=1}^{Q}\Delta_i^{q(L-1)} a_j^{q(L-2)} ,$$

де $\Delta_i^{q(L-1)}$ локальна помилка шару $L-1$

$$\Delta_i^{q(L-1)} = \sum_{k=1}^{S^L}(t_k^q - a_k^{qL}) f_L'(n_k^{qL}) w_{ki}^{L} f_{L-1}'(n_i^{q(L-1)}) = f_{L-1}'(n_i^{q(L-1)})\sum_{k=1}^{S^L}\Delta_k^{qL} w_{ki}^{L}, \quad i = \overline{1, S^{L-1}} .$$

Для шарів $L-2$, $L-3$,..., 1 обчислення приватних похідних критерію J по елементах матриць вагових коефіцієнтів виконується аналогічно. У результаті отримуємо наступну загальну формулу:

$$\frac{\partial J}{\partial w_{ij}^{r}} = \sum_{q=1}^{Q}\Delta_i^{qr} a_j^{q(r-1)}) , \quad r = \overline{1, L} , \quad i = \overline{1, S^r}, \quad j = \overline{0, S^{r-1}} , \qquad (2.8)$$

де r – номер шару;

$$\Delta_i^{qr} = f_r'(n_i^{qr})\sum_{k=1}^{S^{r+1}}\Delta_k^{q(r+1)} w_{ki}^{r+1}, \quad r = \overline{1, L-1} ;$$

$$\Delta_i^{qL} = (t_i^q - a_i^{qL}) f_L'(n_i^{qL}) , \quad i = \overline{1, S^L} .$$

На рис.2.2 представлена схема обчислень, відповідна виразу (2.8). На схемі символом * позначена операція поелементного множення векторів, а символом ** –

множення вектора Δ на $\mathbf{a}^{\mathrm{T}}$; символ, що позначає номер елементу вибірки, скорочено опущений.

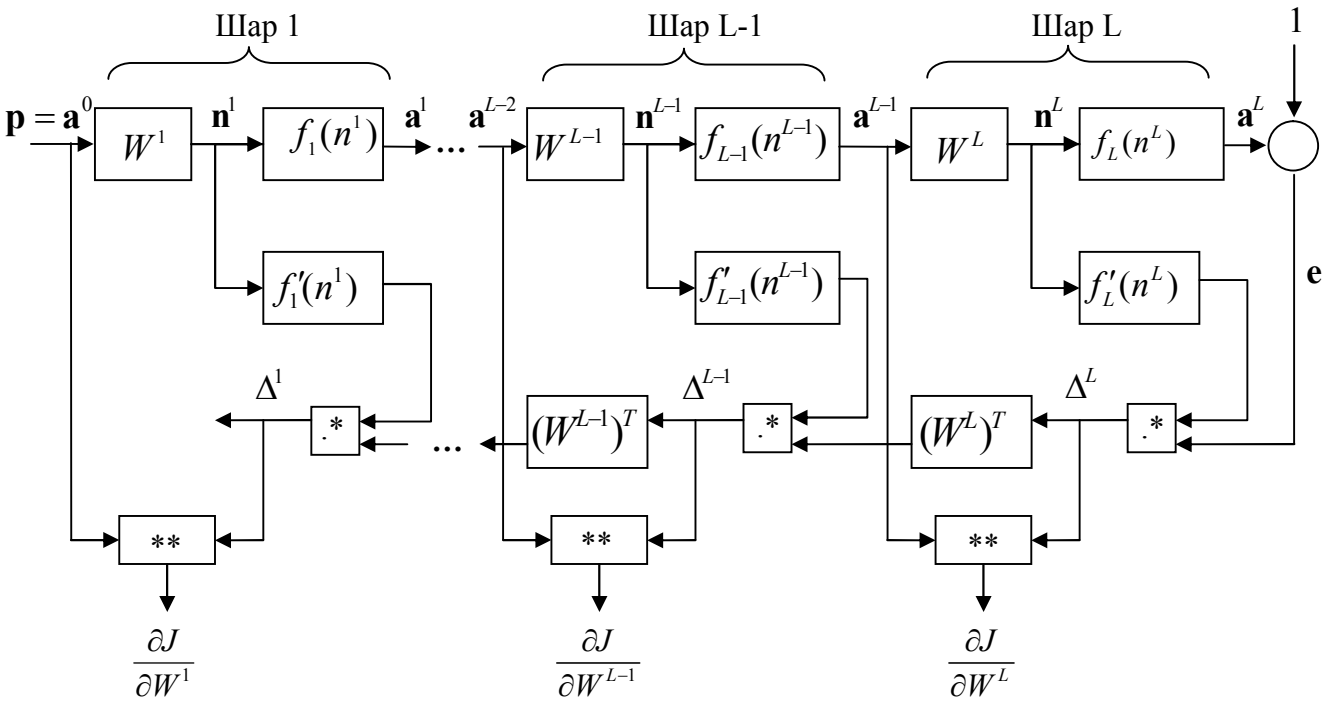


Рис.2.2. Схема обчислень приватних похідних критерію J
по елементах матриць вагових коефіцієнтів

2.8 Настройка вагових коефіцієнтів мережі

Як зазначалося, методи, використовувані при навчанні нейронних мереж, багато в чому аналогічні методам визначення екстремуму функції декілька змінних. Вони діляться на дві категорії – методи першого і другого порядку.

У методах *першого* порядку використовується градієнт функціонала по параметрах, що настроюються, і на кожній ітерації k уточнення вектора вагів $\mathbf{w}_k$ проводиться по формулі

$$\mathbf{w}_{k+1} = \mathbf{w}_k - a_k \mathbf{g}_k \ , \tag{2.9}$$

де a_k – параметр швидкості навчання;

$\mathbf{g}_k$ – градієнт функціонала, відповідний ітерації з номером k.

Вектор в напрямі, протилежному градієнту, указує напрям найкоротшого спуску по поверхні функціонала помилки. Якщо реалізується рух в цьому напрямі, то помилка зменшуватиметься. Послідовність таких кроків врешті-решт приведе до значень настоюваних параметрів, що забезпечують мінімум функціонала. Вище зазначалось, що певну трудність тут викликає вибір параметра швидкості навчання a_k, оскільки при великому значенні параметра a_k збіжність буде швидкою, але існує небезпека пропустити рішення або піти в неправильному напрямі. Залежно від при-

йнятого алгоритму параметр швидкості навчання може бути постійним або змінним. Правильний вибір цього параметра залежить від конкретного завдання. У разі змінного параметра його значення зменшується у міру наближення до мінімуму функціонала. Згідно (2.9) уточнюються значення вагів $\mathbf{w}_k$ в методі зворотного розповсюдження в його стандартній реалізації.

У алгоритмах зв'язаного градієнта пошук мінімуму виконується уздовж зв'язаних напрямів, що забезпечує зазвичай швидшу збіжність, ніж при найшвидшому спуску. Всі алгоритми зв'язаних градієнтів на першій ітерації починають рух у напрямі антиградієнта

$$\mathbf{p}_0 = -\mathbf{g}_0 .$$

Тоді напрям наступного руху визначається так, щоб він був зв'язаний з попереднім. Відповідний вираз для нового напряму руху є комбінацією нового напряму найшвидшого спуску і попереднього напряму:

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \cdot \mathbf{p}_{k-1} ,$$

де $\mathbf{p}_k$. – напрям руху;

$\mathbf{g}_k$ – градієнт функціонала помилки;

β_k – коефіцієнт, що відповідає ітерації з номером k.

Коли напрям спуску визначений, то нове значення вектора параметрів $\mathbf{w}_{k+1}$, що настроюються, обчислюється за формулою

$$\mathbf{w}_{k+1} = \mathbf{w}_k - a_k \cdot \mathbf{p}_k .$$

Градієнтні методи першого порядку характеризуються малою швидкістю збіжності. Це породило помилкову думку про те, що всі нейронні мережі вимагають для тренування великих витрат часу.

Потужніші градієнтні методи *другого* порядку базуються на методі локальної оптимізації Ньютона, який при пошуку оптимуму додатково враховує ще і вигин поверхні функціонала якості (другі похідні).

Основний крок методу Ньютона визначається по формулі:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{H}_k^{-1}\mathbf{g}_k ,$$

де $\mathbf{H}$ – матриця Гессе других похідних функціонала якості по вагових коефіцієнтах мережі.

У багатьох випадках метод Ньютона сходиться швидше, ніж методи зв'язаного градієнта, але вимагає великих витрат із-за обчислення гессиана. Для того, щоб уникнути обчислення матриці Гессе, пропонуються різні способи її заміни наближеними виразами, що породжує так звані *квазіньютонові алгоритми*, такі як алгоритм, запропонований Бройлером, Флетчером, Гольдфарбом і Шанно [56], однокроковий алгоритм методу січних площин OSS, описаний в роботі Батіті [57], алгоритм LM Левенберга-Марквардта [58] і ін.

Ефективним є алгоритм Левенберга-Марквардта. Даний алгоритм реалізує наступну стратегію для оцінки матриці Гессе. Оскільки функціонал якості визначається як сума квадратів помилок, то гессиан може бути приблизно обчислений як

$$\mathbf{H} \cong \mathbf{J}^T\mathbf{J} \ , \qquad (2.10)$$

а градієнт розрахований по формулі:

$$\mathbf{g} \cong \mathbf{J}^T\mathbf{e},$$

де $\mathbf{J} = \dfrac{\partial J}{\partial \mathbf{w}}$ – матриця Якобі похідних функціонала помилки по параметрах, що настроюються;

$\mathbf{e}$ – вектор помилки мережі.

Матриця Якобі може бути обчислена на основі стандартного методу зворотного розповсюдження помилки, що істотно простіше за матрицю Гессе.

У алгоритмі Левенберга-Марквардта нове значення вектора параметрів $\mathbf{w}_{k+1}$, що настроюються, обчислюється за формулою:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - (\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}_k \ .$$

Коли коефіцієнт μ рівний 0, отримуємо метод Ньютона з наближенням гессиана у формі (2.10), коли значення μ велике, отримуємо метод градієнтного спуску з маленьким кроком. Оскільки метод Ньютона має велику точність і швидкість збіжності поблизу мінімуму, завдання полягає в тому, щоб в процесі мінімізації щонайшвидше перейти до методу Ньютона. З цією метою параметр μ зменшують після кожної успішної ітерації і збільшують тільки тоді, коли пробний крок показує, що функціонал помилки зростає. Така стратегія забезпечує зменшення помилки після кожної ітерації алгоритму.

Питання для самоконтролю.

1. У чому полягає сутність процесу навчання нейронних мереж?
2. Покажіть зв'язок алгоритмів навчання нейронних мереж і алгоритмів пошуку глобального екстремуму функції багатьох змінних.
3. Що таке «явище перенавчання» ? Як виявити ефект перенавчання?
4. У чому полягає здібність узагальнення нейронних мереж?
5. Запишіть квадратичний функціонал якості навчання.
6. Які проблеми виникають при використанні градієнтних методів настройки параметрів ШНМ?
7. Які ви знаєте методи навчання нейронних мереж?
8. Запишіть основні етапи навчання одношарової мережі.
9. Перерахуйте кроки навчання нейронної мережі.
10. Запишіть основні етапи навчання багатошарової мережі методом зворотного розповсюдження помилки.
11. Перечисліть і запишіть основні співвідношення методів настройки вагових коефіцієнтів мережі.

ІДЕНТИФІКАЦІЯ ДИНАМІЧНИХ СИСТЕМ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

3.1 Моделі систем, використовувані при нейромережевій ідентифікації

Основним питанням в нейрорегулюванні є отримання адекватних моделей динамічних систем. Питання про існування дискретних моделей, або іншими словами про спостережуваність нелінійних систем, є ключовим. У загальному випадку завдання побудови математичної моделі об'єкту полягає у виборі структури і оцінці її параметрів так, щоб при використанні критерію мінімуму деякої функції різниці розрахункових і експериментальних даних дотримувалася умова близькості моделі досліджуваному процесу. Застосування нейронних мереж як моделі може служити альтернативою класичним методам математичного опису систем, оскільки в цьому випадку точне знання внутрішніх процесів не є необхідною умовою моделювання.

При нейронній ідентифікації нелінійних систем найчастіше знаходять застосування наступні основні нелінійні дискретні моделі.

У найбільш загальному вигляді, прийнятому в класичній теорії управління, опис стохастичних нелінійних систем в просторі станів розглядався в роботі Наренди К.С. і Партасараті К. [27].

$$z(k+1) = \Psi[z(k),\ u(k),\ \varphi(k)]\ ;$$

$$y(k) = \Phi[z(k),\ u(k),\ \phi(k)],$$

де $z(k) = [z_1(k),\ z_2(k),...,\ z_{k_z}(k)]^{\mathrm{T}}$ – вектор стану системи;

$u(k) = [u_1(k),\ u_2(k),...,\ u_{k_u}(k)]^{\mathrm{T}}$ – вектор вхідних сигналів;

$y(k) = [y_1(k),\ y_2(k),...,\ y_{k_y}(k)]^{\mathrm{T}}$ – вектор вихідних сигналів;

$\Psi[\cdot],\ \Phi[\cdot]$ – деякі статичні нелінійні функції;

$\varphi(k),\ (k)$ – шум процесу і шум вимірювань відповідно.

В роботі [59] Леонтарітіс І. і Біллінгз С. запропонували для опису нелінійних систем використовувати так звану нелінійну модель авторегресії ковзаючого середнього з додатковими вхідними сигналами (NARMAX модель), що одержала подальший розвиток в [60-62]:

$$y(k) = f[y(k-1),...,y(k-n_y),\ u(k-1),...,u(k-n_u),\ e(k-1),...,e(k-n_e)] + e(k)\,, \quad (3.1)$$

де $y(k) \in \mathbf{R}^{M\times 1}$ – вихідний сигнал об'єкту;

$u(k) \in \mathbf{R}^{N\times 1}$ – вхідний сигнал об'єкту;

$e(k) \in \mathbf{R}^{M\times 1}$ – різниця вихідних сигналів об'єкту і моделі;

$f[\cdot]$ – нелінійна функція перетворення f: $\mathbf{R}^{(k_y M+k_u N+k_e M)\times 1} \to \mathbf{R}^{M\times 1}$;

n_y, n_u, n_e – порядки запізнювання по вихідному, вхідному сигналам об'єкту і помилці вимірювань відповідно.

Скалярна форма рівняння (3.1) має вигляд:

$$y_i(k) = f_1[y_1(k-1),...,y_1(k-n_y),y_2(k-1),...,y_2(k-n_y),y_M(k-1),...,y_M(k-n_y),$$

$$u_1(k-1),...,u_1(k-n_u),u_2(k-1),...,u_2(k-n_u),u_N(k-1),...,u_N(k-n_u),$$

$$e_1(k-1),...,e_1(k-n_e),e_2(k-1),...,e_2(k-n_e),e_M(k-1),...,e_M(k-n_e)] + e_i(k). \qquad (3.2)$$

Нелінійні системи можуть бути представлені нелінійною NARX-моделлю:

$$y(k) = f[y(k-1),...,y(k-n_y),\ u(k-1),...,u(k-n_u)] + e(k), \qquad (3.3)$$

або в скалярному вигляді

$$y_i(k) = f_1[y_1(k-1),...,y_1(k-k_y),y_2(k-1),...,y_2(k-k_y),y_M(k-1),...,y_M(k-k_y),$$

$$u_1(k-1),...,u_1(k-k_u),u_2(k-1),...,u_2(k-k_u),u_N(k-1),...,u_N(k-k_u)] + e_i(k). \qquad (8.4)$$

Представлення об'єктів у вигляді NARMAX або NARX моделей грають фундаментальну роль при дослідженні нелінійних об'єктів за допомогою штучних нейронних мереж. При цьому важливо розглянути узагальнення моделей (3.1) – (3.4), що враховує різні види нелінійностей по вхідних і вихідних змінних. Наренда К.С. [27] виділив чотири такі моделі:

– модель 1 (рис.3.1):

$$y(k+1) = \sum_{i=1}^{k_y-1} a_i y(k-i) + g[\ u(k),\ u(k-1),...,u(k-n_u+1)]; \qquad (3.5)$$

– модель 2 (рис.3.2):

$$y(k+1) = f[y(k),\ y(k-1),...,y(k-n_y+1)] + \sum_{i=1}^{k_u-1} b_i u(k-i); \qquad (3.6)$$

– модель 3 (рис.3.3):

$$y(k+1) = f[y(k),\ y(k-1),...,y(k-n_y+1)] + g[\ u(k),\ u(k-1),...,u(k-n_u+1)]; \quad (3.7)$$

– модель 4 (рис.3.4):

$$y(k+1) = f[y(k),\ y(k-1),...,y(k-n_y+1),\ u(k),\ u(k-1),...,u(k-n_u+1)]. \qquad (3.8)$$

Модель (3.8) є найбільш загальною зі всіх моделей вхід/вихід, описаних вище. При цьому модель 3.8 є аналітично вирішуваною і найбільш зручною для практичних застосувань. Внаслідок цього вона зазвичай використовується при ідентифікації нелінійних об'єктів за допомогою штучних нейронних мереж.

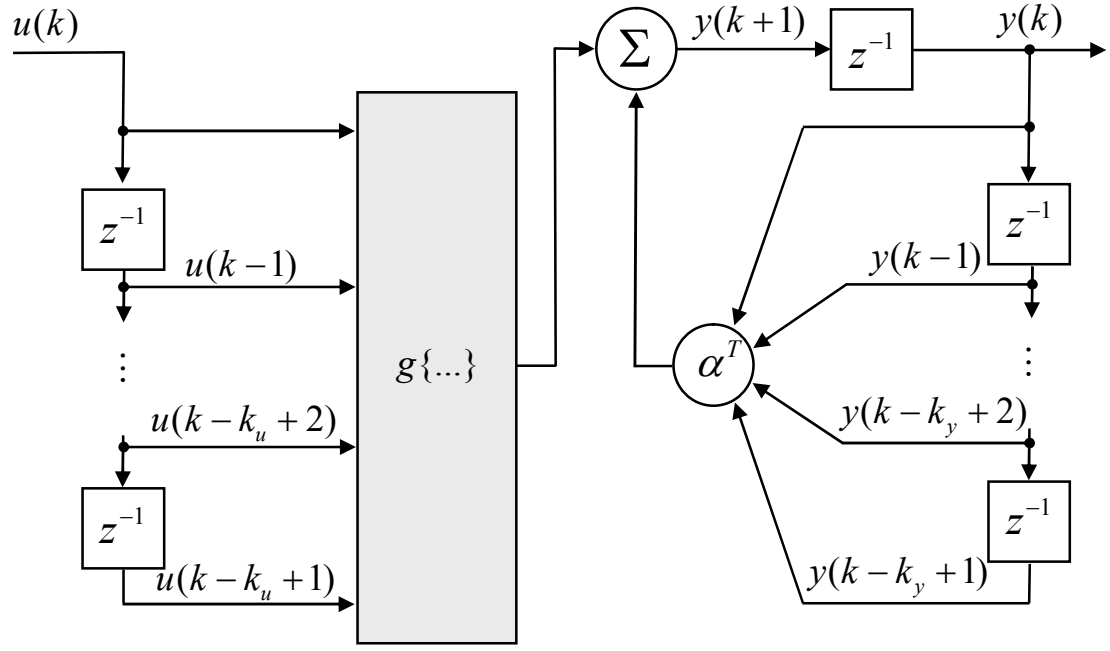


Рис.3.1. Модель 1. Вихід моделі нелінійно залежить від вхідних сигналів і лінійно від вихідних

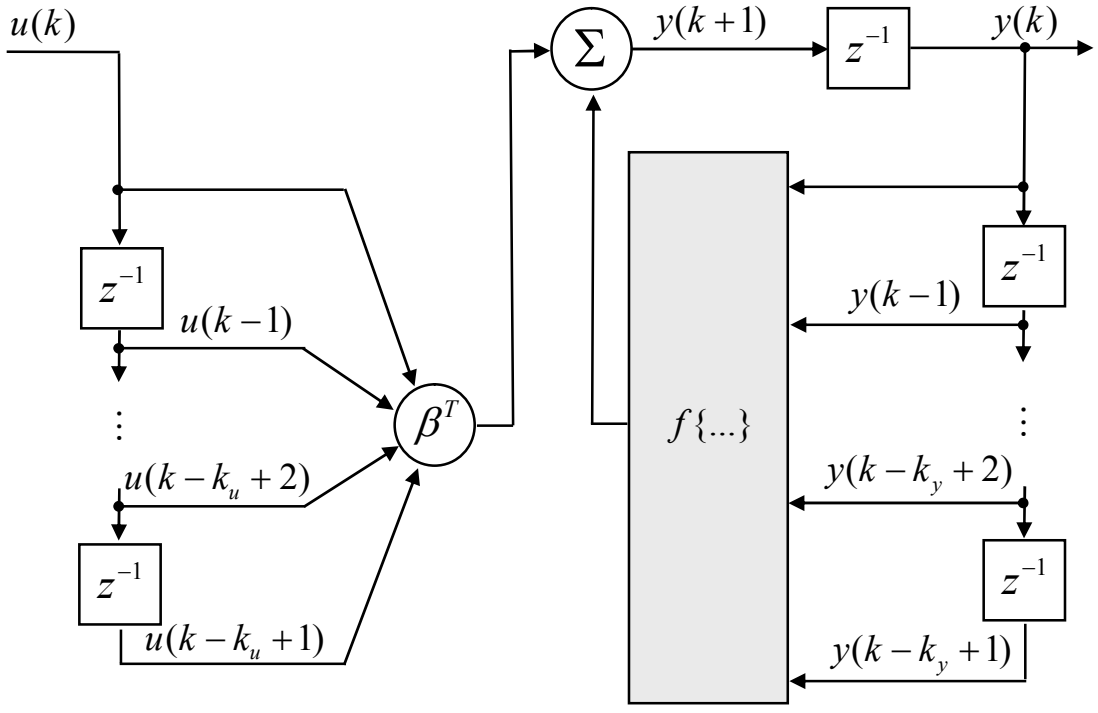


Рис.3.2. Модель 2. Вихід моделі лінійно залежить від вхідних сигналів і нелінійно від вихідних

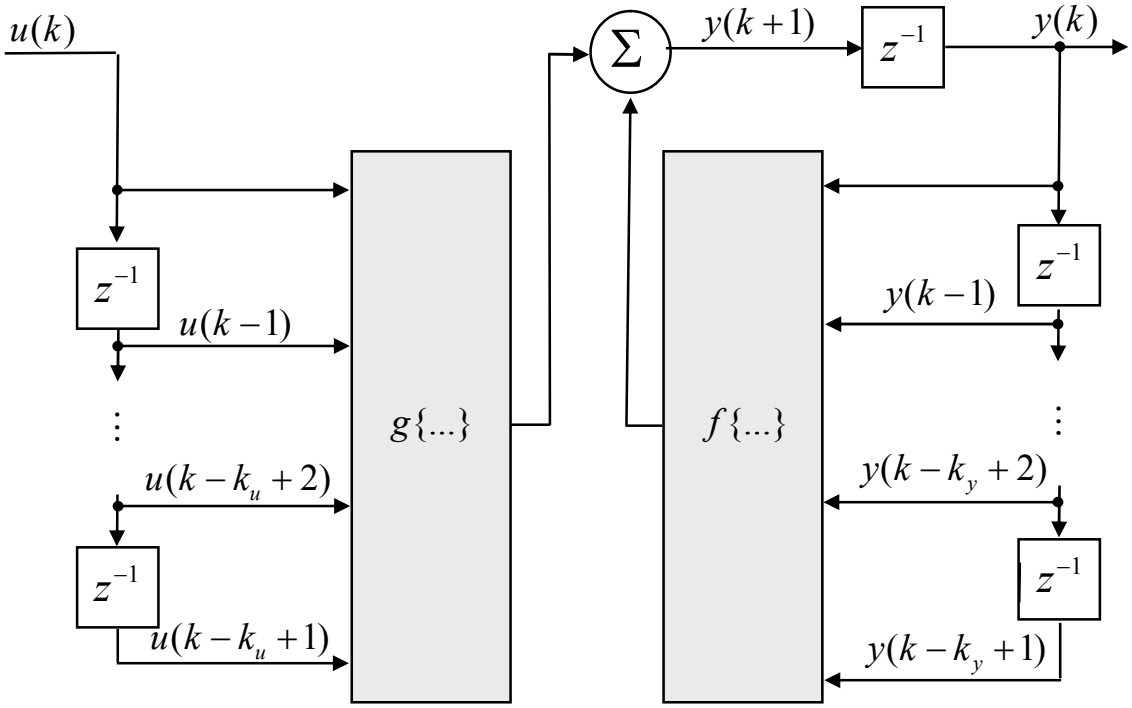


Рис.3.3. Модель 3. Результуючий сигнал є адитивною функцією нелінійних перетворень вхідних і вихідних сигналів

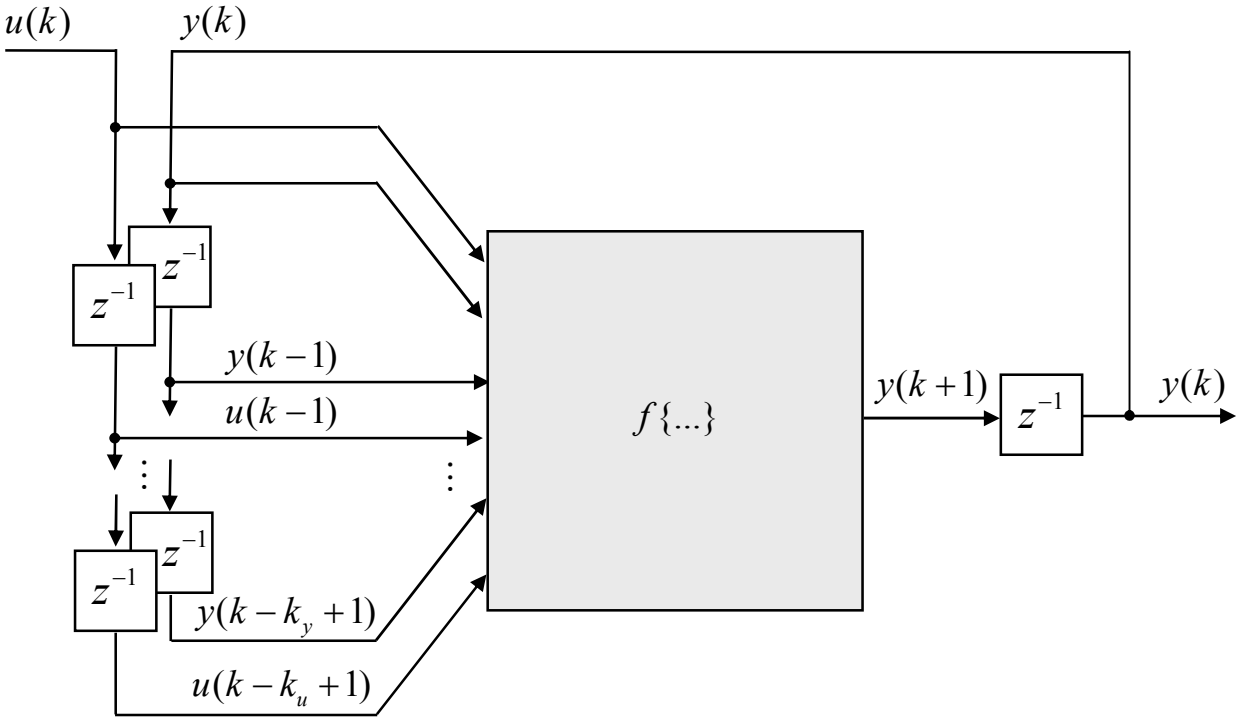


Рис.3.4. Модель 4. Вихід моделі нелінійно залежить як від вхідних, так і від вихідних сигналів

3.2. Нейронні мережі як універсальні моделі

Оскільки тип і взаємозв'язок нелінійностей в об'єкті, як правило, не відомі, для моделювання слід використовувати якомога більш універсальні структури моделей. Нейронні мережі є саме такими структурами.

Нелінійні моделі в просторі станів можуть бути реалізовані за допомогою рекурсивних нейронних мереж з внутрішніми зворотними зв'язками. Ідентифікація параметрів цих мереж вимагає спеціальної і досить трудомісткої форми алгоритму навчання. Крім того, доказ стійкості таких нелінійних динамічних моделей практично неможливо зробити. Тому використовуються прямонаправлені мережі, що характеризуються наявністю зв'язків між нейронами тільки в прямому напрямі без зворотних зв'язків усередині мережі – багатошаровий персептрон. За допомогою БП можна апроксимувати з бажаною точністю не тільки будь-які статичні функції. Попередні значення вхідних/вихідних координат у вхідному векторі дозволяють додати прямонаправленным мережам динамічні властивості.

Володіючи схожими з радіально-базисними мережами властивостями, БП забезпечує компактніше представлення нелінійного об'єкту, оскільки число параметрів в мережі РБМ експоненціально зростає із збільшенням розмірності вхідного простору, тоді як для БП ця залежність лінійна. Важливою є наочність, що виражається в тому, що персептронне представлення моделі еквівалентне традиційному. Це дозволяє використовувати для визначення параметрів штучних нейронних мереж добре розвинені методи оцінювання, використовувані при традиційному підході.

Модель системи в просторі станів може бути успішно реалізована за допомогою прямонаправлених мереж, якщо змінні стану сформувати на виході мережі і потім завести їх на вхід мережі із затримкою. Але для цього потрібно, щоб всі змінні стану були вимірювані, оскільки тільки в цьому випадку можна забезпечити тренування мережі.

NARX мережеві моделі параметризуються лінійно і тому використовують стандартні методи тренування мережі. Внаслідок цього саме даний вид моделей найчастіше використовується при нейронній ідентифікації систем. Для простоти викладу надалі розглядаємо системи з одним входом і одним виходом SISO (singl input singl output), хоча всі висновки можуть бути поширені і на MIMO (multi input multi output) системи. Прямій нейронній NARX моделі нелінійної динамічної системи відповідає наступний вираз:

$$\hat{y}(k) = f\left(y(k-1)...y(k-n_y), u(k-n_k)...u(k-n_u-n_k+1), \mathbf{w}\right), \qquad (3.9)$$

а нейронній інверсній NARX моделі:

$$\hat{u}(k) = g\left(y(k+n_k)...y(k+n_k-n_y), u(k-1)...u(k-n_u+1), \mathbf{w}\right) \qquad (3.10)$$

де f, g – функції, що реалізовуються нейронними мережами;

u, y – вхідна і вихідна координати відповідно;

$\hat{u}$, $\hat{y}$ – значення координат на виході мережі;

$\mathbf{w}$ – матриця лінійних параметрів мережі;

n_k – запізнювання в системі, найчастіше $n_k = 1$.

Вирази (3.9) і (3.10) можна представити графічно у вигляді структурних схем, показаних на рис.3.5 і рис.3.6. NARMAX моделі характеризуються вищою якістю ідентифікації при зашумлених тренувальних даних. У [63] пропонується спосіб модифікації критерію якості алгоритму тренування. Результати ідентифікації за допомогою NARX моделей наближаються в цьому випадку до результатів, отриманих за допомогою NARMAX моделей. Дані NARX нейромоделі можуть бути успішно застосовані для вирішення поставлених завдань.

3.3 Побудова прямої і інверсної нейронної моделі об'єкту

Як зазначалося вище, завдання теорії регулювання, що займається пошуком моделей динамічних систем на основі апріорної інформації і вимірюваних вхідних/вихідних сигналів системи, носить назву ідентифікація. Відомі класичні методи ідентифікації нелінійних систем застосовні тільки у поєднанні з певним типом моделі (наприклад, ряди Вальтера, моделі Хаммерштейна і Вінера), що описує певний клас нелінійних динамічних систем. В подальших розділах під термінами «ідентифікація системи», «тренування» і «навчання» розуміється процес підстроювання вагових коефіцієнтів мережі за допомогою градієнтних методів. Метою цього процесу є по можливості точне наближення поведінки моделі до спостережуваної поведінки динамічної системи.

На рис.3.7 представлені схеми тренування прямої і інверсної нейронних моделей об'єкту. Для спрощення на рис.3.7 не зображені ланки затримки за часом, що формують попередні значення координат на вході нейронних мереж.

Вхідний сигнал нейронної моделі об'єкту $\mathbf{M}$ на рис. 3.7а ідентичний вхідному сигналу об'єкту $\mathbf{S}$. Для інверсної нейромоделі $\mathbf{N}$ як вхідний сигнал використовується вихідний сигнал об'єкту $\mathbf{S}$, тоді як нейромодель повинна відновити вхідний сигнал об'єкту (див. рис.3.7б). Поки присутня помилка між виходом нейромоделі і її зразковим сигналом, ця помилка використовується для тренування мережі. У обох випадках ідентифікації обчислюється градієнт функціонала якості (див. таблицю 3.1). Функціонал якості найчастіше при цьому квадратичний.

Оскільки $\frac{\partial \hat{y}}{\partial \mathbf{w}}$ і $\frac{\partial \hat{u}}{\partial \mathbf{w}}$ обчислювані, то це дозволяє вести підстроювання вагових коефіцієнтів мережі згідно розглянутим градієнтним методам. Результатом цього буде мінімізація функціонала якості. Оскільки набір тренувальних даних однозначно відображає лише поведінку модельованого об'єкту $\mathbf{S}$ у робочому діапазоні, то після завершення тренування маємо $\mathbf{M} \approx \mathbf{S}$ і $\mathbf{N} \approx \mathbf{S}^{-1}$.

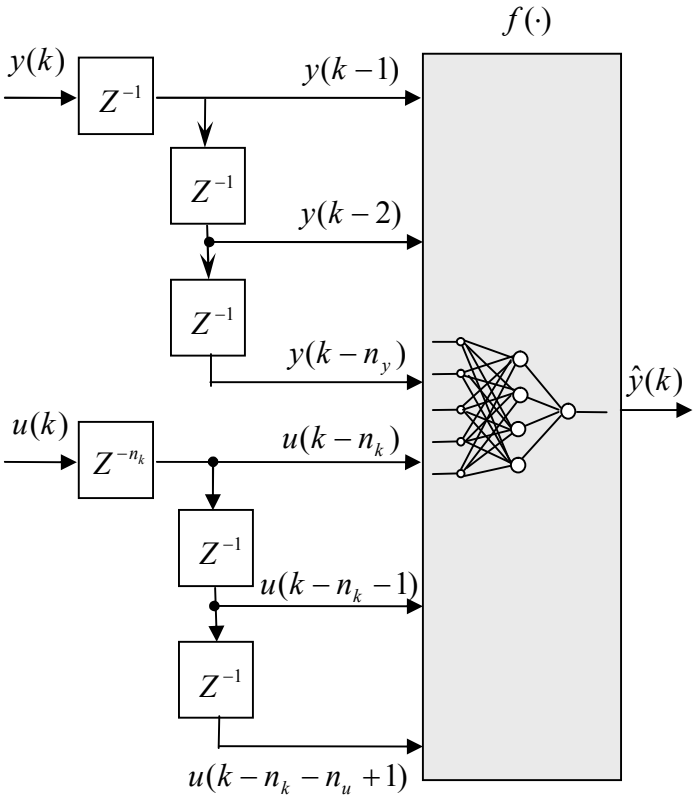


Рис.3.5. Пряма нейромережева NARX модель

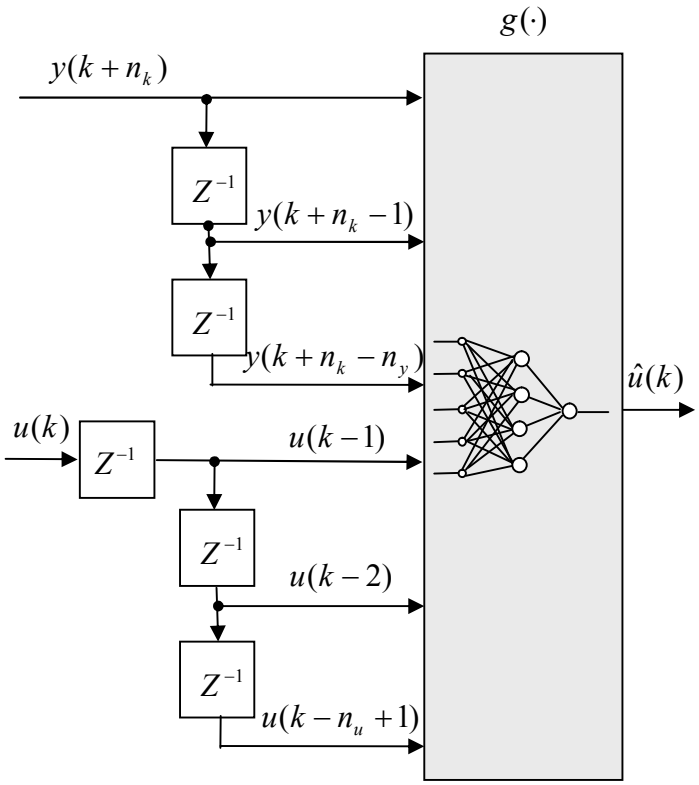


Рис.3.6. Інверсна нейромережева NARX модель

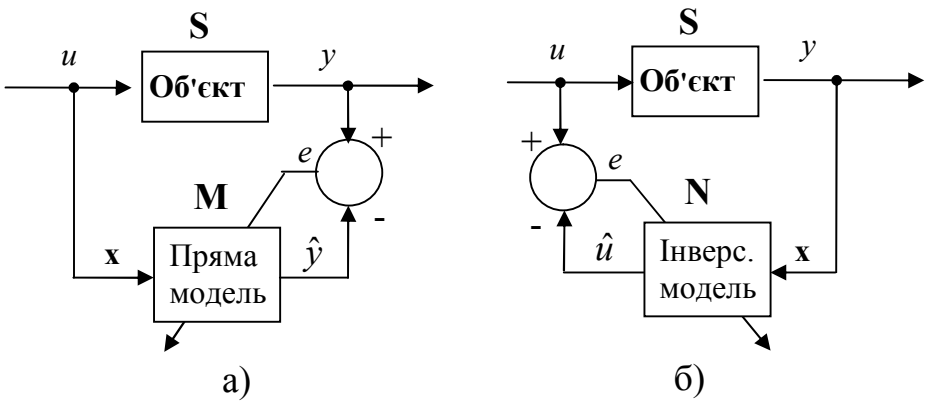


Рис.3.7. Схеми тренування прямої (а) і інверсної (б) моделей

Таблиця 3.1

Обчислення градієнта при тренуванні нейромоделей

	Пряма модель	Інверсна модель
Бажаний вихідний сигнал	y	u
Вихідний сигнал нейромоделі	$\hat{y} = \mathbf{M}(\mathbf{x}, \mathbf{w})$	$\hat{u} = \mathbf{N}(\mathbf{x}, \mathbf{w})$
Помилка навчання	$e = y - \hat{y}$	$e = u - \hat{u}$
Градієнт функціонала якості	$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \mathbf{w}} = -e\frac{\partial \hat{y}}{\partial \mathbf{w}}$	$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \mathbf{w}} = -e\frac{\partial \hat{u}}{\partial \mathbf{w}}$

Типова послідовність дій, що приводить до формування нейромоделі (ідентифікації) динамічної системи, відображена в таблиці 3.2.

Таблиця 3.2

Ідентифікація за допомогою нейронних мереж

Шаг	Дія
1	Знімання тренувальних даних
2	Вибір структури мережі
3	Вибір типу моделі
4	Тренування мережі
5	Тестування нейромоделі
6	При незадовільному результаті тестування, перейти до пункту 3 або до пункту 2

Оскільки тренувальні дані є носієм інформації про систему, що ідентифікується, то наявності, кількості і якості цих даних слід приділити особливу увагу. Тренувальних дані повинні мати наступні характеристики [64]: якісна дискретизація, рівномірність дисперсії і величини варіації, рівномірність амплітуд всіх сигналів і їх середньоквадратичних значень.

При отриманні позитивного результату апроксимації, слід також перевірити здібність нейромоделі до узагальнення, тобто її здатність повторювати істотні характеристики об'єкту. Для цього тесту необхідний ще один набір даних, що не брав участь при тренуванні мережі. Цей тест дає остаточний висновок про адекватність нейромоделі.

Схема ідентифікації нелінійного об'єкту на основі ШНМ з зображенням ланок затримки за часом, що формують попередні значення координат на вході нейронних мереж, приведена на рис.3.8. Схема складена з урахуванням завад вимірювання вихідного сигналу ξ.

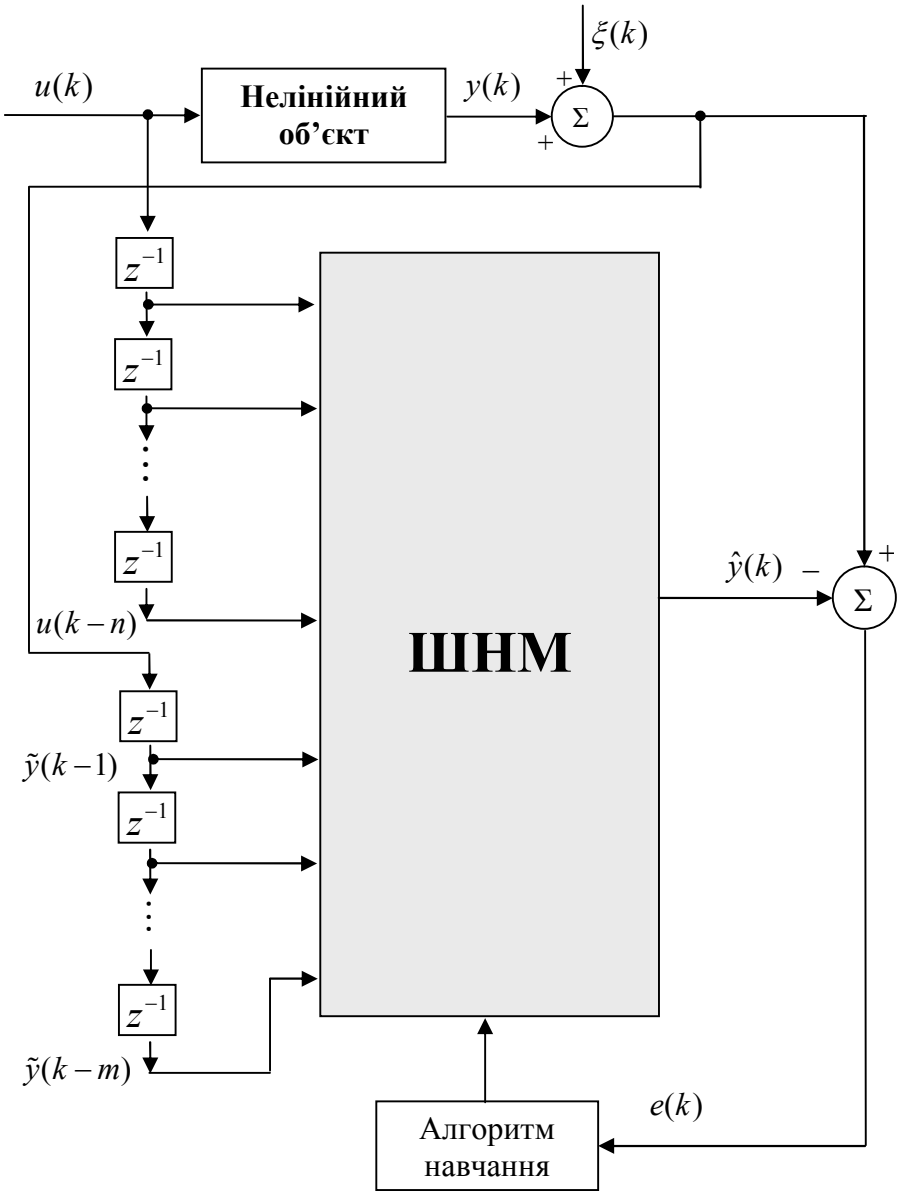


Рис.3.8. Схема ідентифікації нелінійного об’єкту управління

Представлені теоретичні положення по нейронній ідентифікації, а також характеристики прямонаправлених мереж є основою успішного регулювання нелінійних динамічних систем.

Питання для самоконтролю.

1. Які моделі систем використовуються при нейромережевій ідентифікації?
2. Запишіть рівняння узагальнених NARMAX або NARX моделей, що враховують різні види нелінійностей по вхідних і вихідних змінних.
3. Зобразіть структурні схеми узагальнених NARMAX або NARX моделей, що враховують різні види нелінійностей по вхідних і вихідних змінних.
4. Запишіть рівняння прямої і інверсної нейронних NARX моделей нелінійної динамічної системи.
5. Зобразіть структурні схеми прямої і інверсної нейронних NARX моделей.
6. Зобразіть структурні схеми тренування прямої і інверсної нейронних моделей.
7. Запишіть методику обчислення градієнта при тренуванні нейромоделей.
8. Запишіть типову послідовність дій, що приводить до формування нейромоделі (ідентифікації) динамічної системи.
9. Зобразіть схему ідентифікації нелінійного об’єкту управління за допомогою штучної нейронної мережі.
10. Зобразіть схему ідентифікації нелінійного об'єкту за допомогою багатошарового персептрона.
11. Які Ви знаєте програмні засоби для побудови нейромережевих моделей динамічних об’єктів і систем?

СИНТЕЗ НЕЙРОМЕРЕЖЕВИХ СИСТЕМ УПРАВЛІННЯ З ВИКОРИСТАННЯМ СИСТЕМИ MATLAB

4.1 Математична модель трьохмасової системи

Методику синтезу нейромережевих систем управління розглянемо на прикладі побудови системи управління механізмом підйому шахтної підіймальної установки з урахуванням пружних властивостей підіймального канату.

При розгляді каната як пружного елементу з розподіленою масою можна ігнорувати ряд незначних факторів. Наприклад, канат можна вважати консервативною ланкою, оскільки демпфування, що здійснюється двигуном і тертям підйомної судини об ті, значно перевищує сили внутрішнього тертя в канаті. Також не суттєво нерівномірне розподілення маси по довжині каната, яке викликане його деформацією від власної ваги.

В урахуванні даних допущень основне значення мають подовжні коливання каната, тоді як поперечні і крутильні коливання є другорядними. Найбільший вплив на процеси в системі має навантажена частина каната, коли судно знаходиться в крайньому нижньому положенні, а тривалість перехідного режиму під час розгону незначна, тому зміна довжини каната через його навивку на барабан є малою у відношенні до загальної довжини каната. Тому можна припустити, що довжина пружного елемента і його жорсткість залишаються незмінними в період розгону.

З урахуванням зазначених припущень розрахункову схему механічної підйомної установки можна зобразити як дві зосереджені маси m_1 і m_2, сполучені між собою ідеально пружним стрижнем (рис. 4.1).

У технічній літературі приводиться методика розрахунку вказаного завдання. Проте для практичних розрахунків ця методика достатньо складна. В цілях спрощення можна розглядати механічну частину підйомної установки як трьохмасову систему, в якій підйомний канат зображається зосередженою масою, пов'язаною з двома іншими масами еквівалентними невагомими пружними зв'язками, а також двомасової системи, в якій канат приведений до зосереджених мас (див. рис. 3.1).

Зосереджена ведена m_1, провідна m_2 маси і маса каната $m_к$ рівні:

$$m_1 = \mu_1 m \qquad m_2 = \mu_2 m \qquad m_к = \mu_к m,$$

де μ_1, μ_2, $\mu_к$ відносні приведені маси; $m = m_1 + m_2 + m_{1к} \cdot H$ - сумарна приведена маса; $m_{1к}$ - маса одиниці довжини каната.

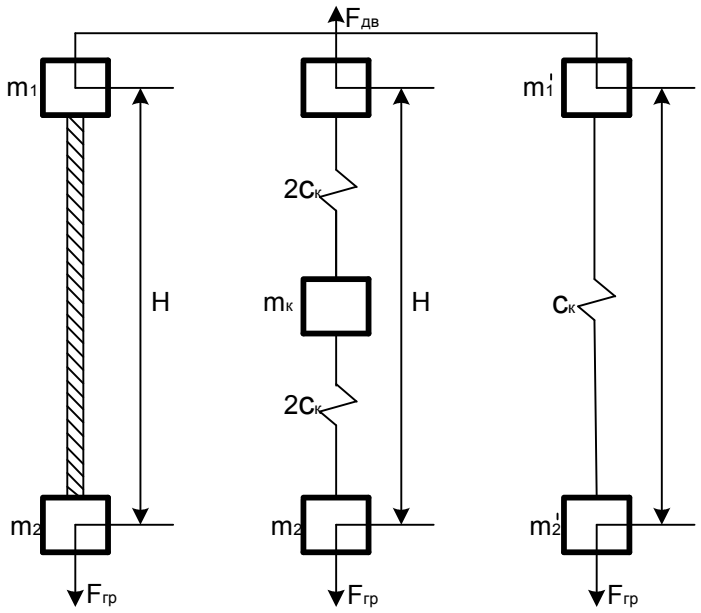


Рис. 4.1. Розрахункові схеми механічної частоти підйомної установки

У інженерних розрахунках розглядають підйомний канат, що володіє розподіленою масою, пов'язаною з масами m_1 і m_2 еквівалентними невагомими пружними зв'язками з жорсткістю c_1 і c_2, тобто розглядають трьохмасову систему. В цілях спрощення досліджень трьохмасова система може бути замінена двомасовою з приведеними масами $m_1^{/} = m_1 + m_к / 2$ і $m_2^{/} = m_2 + m_к / 2$, сполученими невагомим канатом з жорсткістю c_{12}.

Динамічні процеси в механічній частині підйомної установки, представленою трьохмасовою системою, описуються рівняннями

$$\begin{cases} J_{д\Sigma}\dfrac{d\omega_д}{dt} = k\Phi I_д - M_{\Sigma 1}; \\ J_к\dfrac{d\omega_к}{dt} = M_{\Sigma 1} - M_{\Sigma 2}; \\ J_м\dfrac{d\omega_м}{dt} = M_{\Sigma 2} - M_{ст}; \\ \dfrac{dM_{пр1}}{dt} = c_1\left(\omega_д - \omega_к\right); \\ \dfrac{dM_{пр2}}{dt} = c_2\left(\omega_к - \omega_м\right), \end{cases} \qquad (4.1)$$

де $M_{\Sigma 1}$, $M_{\Sigma 2}$ - сумарні моменти, що передаються пружними передачами

$$M_{\Sigma 1} = M_{пр1} + M_{вт1}, \qquad M_{\Sigma 2} = M_{пр2} + M_{вт2}, \qquad (4.2)$$

$M_{\text{пр1}}$, $M_{\text{пр2}}$- моменти пружної взаємодії між масами трьохмасової системи; $M_{\text{вт1}}$, $M_{\text{вт2}}$ - моменти в'язкого тертя

$$M_{\text{вт1}} = \beta_1(\omega_{\text{д}} - \omega_{\text{к}}); \qquad M_{\text{вт2}} = \beta_2(\omega_{\text{к}} - \omega_{\text{м}}), \tag{4.3}$$

β_1, β_2 - коефіцієнти внутрішнього в'язкого тертя ділянок каната між масами; $J_{\text{дΣ}}$ - приведений момент інерції провідної маси: $J_{\text{дΣ}} = m_1\rho^2$; $J_{\text{к}} = m_{к1}H\rho^2$ - приведений момент інерції каната (другої маси); $m_{к1}$ - маса одного погонного метра каната; H - висота підйому; $J_{\text{м}}$ - приведений момент інерції навантаженого скипа: $J_{\text{м}} = m_2\rho^2$; $\rho = \dfrac{V_{\text{п}}}{\omega_{\text{д}}} = \dfrac{D_{\text{б}}}{2}$ - радіус приведення поступальної швидкості до обертальної; c_1, c_2 - приведені жорсткості ділянок між скипами визначаються таким чином: $c_1 = c_2 = 2c_{\text{к}}\rho^2$; $c_{\text{к}}$ - жорсткість каната: $c_{\text{к}} = c_{1\text{к}}/H$; $c_{1\text{к}}$ - жорсткість одного метра підйомного каната;

Для двомасової системи рівняння руху мають вид:

$$\begin{cases} J'_{\text{м}}\dfrac{d\omega_{\text{м}}}{dt} = M_{\Sigma} - M_{\text{ст}}; \\ \dfrac{dM_{\text{пр}}}{dt} = c_{12}(\omega_{\text{д}} - \omega_{\text{м}}); \\ J'_{\text{дΣ}}\dfrac{d\omega_{\text{д}}}{dt} = k\Phi_{\text{н}}I_{\text{д}} - M_{\Sigma}, \end{cases}$$

де M_{Σ} сумарний момент, що передається пружною передачею, який рівний сумі пружного моменту $M_{\text{пр}}$ і моменту в'язкого тертя $M_{\text{вт}} = \beta(\omega_{\text{д}} - \omega_{\text{м}})$;

$$M_{\Sigma} = M_{\text{пр}} + \beta(\omega_{\text{д}} - \omega_{\text{м}}),$$

β – коефіцієнт внутрішнього в'язкого тертя каната між масами;

$$J'_{\text{дΣ}} = J_{\text{дΣ}} + \frac{J_{\text{к}}}{2}; \quad J'_{\text{м}} = J_{\text{м}} + \frac{J_k}{2}; \quad c_{12} = c_{\text{к}}\rho^2.$$

Розглянемо трьохмасову систему підлеглого регулювання швидкості двигуна з регулятором швидкості, настроєним по симетричному критерію з урахуванням контура струму, налаштованим на модульний оптимум.

Запишемо систему (4.1) з урахуванням (4.2) і (4.3) у формі Коші

$$
\begin{cases}
\dfrac{d\omega_{\text{м}}}{dt} = \dfrac{1}{J_{\text{м}}} M_{\text{пр2}} + \dfrac{\beta_2}{J_{\text{м}}}\omega_{\text{к}} - \dfrac{\beta_2}{J_{\text{м}}}\omega_{\text{м}} - \dfrac{1}{J_{\text{м}}} M_{\text{ст}}; \\
\dfrac{dM_{\text{пр1}}}{dt} = c_1\omega_{\text{д}} - c_1\omega_{\text{к}}; \\
\dfrac{d\omega_{\text{к}}}{dt} = \dfrac{1}{J_{\text{к}}} M_{\text{пр1}} - \dfrac{\beta_1}{J_{\text{к}}}\omega_{\text{д}} + \dfrac{\beta_1}{J_{\text{к}}}\omega_{\text{к}} - \dfrac{1}{J_{\text{к}}} M_{\text{пр2}} - \dfrac{\beta_2}{J_{\text{к}}}\omega_{\text{к}} + \dfrac{\beta_2}{J_{\text{к}}}\omega_{\text{м}}; \\
\dfrac{dM_{\text{пр2}}}{dt} = c_2\omega_{\text{к}} - c_2\omega_{\text{м}}; \\
\dfrac{d\omega_{\text{д}}}{dt} = \dfrac{k\Phi_{\text{н}}}{J_{\text{дΣ}}} I_{\text{д}} - \dfrac{1}{J_{\text{дΣ}}} M_{\text{пр1}} - \dfrac{\beta_1}{J_{\text{дΣ}}}\omega_{\text{д}} + \dfrac{\beta_1}{J_{\text{дΣ}}}\omega_{\text{к}}.
\end{cases}
\tag{4.4}
$$

Розглянемо алгоритмічну схему одномасової системи запишемо рівняння стану системи регулювання без урахування пружних властивостей елементів механічної частини. Далі, додавши рівняння трьохмасової системи, отримаємо диференціальні рівняння стану системи управління з урахуванням пружного зв'язку.

Електропривод механізму підйому виконано за системою тиристорний перетворювач – двигун постійного струму (ТП-Д). Система управління має зворотні зв’язки за струмом і за швидкістю. У якості регуляторів струму і швидкості застосовані пропорційно-інтегральні регулятори. Параметри регуляторів вибрані таким чином, що контур струму оптимізовано за модульним критерієм, а контур швидкості – за симетричним критерієм.

Рівняння стану одномасової системи:

$$
\begin{cases}
\dfrac{d\omega_{\text{д}}}{dt} = \dfrac{k\Phi_{\text{н}}}{J_{\text{дΣ}}} I_{\text{д}} - \dfrac{1}{J_{\text{дΣ}}} M_{\text{ст}}; \\
\dfrac{dI_{\text{д}}}{dt} = -\dfrac{k\Phi_{\text{н}}}{R_\Sigma T_{\text{е}}}\omega_{\text{д}} - \dfrac{1}{T_{\text{е}}} I_{\text{д}} + \dfrac{1}{R_\Sigma T_{\text{е}}} U_{\text{тп}}; \\
\dfrac{dU_{\text{тп}}}{dt} = -\dfrac{k_{\text{пш}} k_{\text{пт}} k_{\text{тп}} k_{\text{ш}}}{T_{\text{µт}}}\omega_{\text{д}} - \dfrac{k_{\text{т}} k_{\text{пт}} k_{\text{тп}}}{T_{\text{µт}}} I_{\text{д}} - \dfrac{1}{T_{\text{µт}}} U_{\text{тп}} - \\
\qquad - \dfrac{k_{\text{тп}}}{T_{\text{µт}}} U_{\text{рт}} - \dfrac{k_{\text{тп}} k_{\text{пт}}}{T_{\text{µт}}} U_{\text{рш}} + \dfrac{k_{\text{пш}} k_{\text{пт}} k_{\text{тп}}}{T_{\text{µт}}} U_{\text{з}}; \\
\dfrac{dU_{\text{рт}}}{dt} = -k_{\text{ш}} k_{\text{пш}} k_{\text{іт}}\omega_{\text{д}} - k_{\text{іт}} k_{\text{т}} I_{\text{д}} + k_{\text{іт}} U_{\text{рш}} + k_{\text{пш}} k_{\text{іт}} U_{\text{з}}; \\
\dfrac{dU_{\text{рш}}}{dt} = -k_{\text{ш}} k_{\text{іш}}\omega_{\text{д}} + k_{\text{іш}} U_{\text{з}}.
\end{cases}
\tag{4.5}
$$

При записі системи (4.5) прийняті наступні позначення: $I_{д}$ – струм якірного кола двигуна; k – коефіцієнт, що залежить від конструктивних даних двигуна; $\Phi_{н}$ – номінальний магнітний потік двигуна; $U_{тп}$ – напруга тиристорного перетворювача; $U_{рт}$, $U_{рш}$ – напруги на виході інтегральної частини ПІ-регуляторів струму і швидкості відповідно; $U_{з}$ – напруга завдання; $T_{е}$ – електромагнітна постійна часу електроприводу; R_{Σ} – сумарний активний опір якірного кола; $k_{тп}$ – коефіцієнт посилення тиристорного перетворювача; $k_{д}$ – коефіцієнт посилення двигуна; $k_{ш}$, $k_{т}$ – коефіцієнт посилення зворотного зв'язку за швидкістю і струмом; $T_{μт}$ – мала, що не компенсується, постійна часу контуру струму; $k_{пш}$, $k_{іш}$, $k_{пт}$, $k_{іт}$ – коефіцієнти підсилення пропорційної і інтегральної частин регулятора швидкості і струму відповідно; значення указаних параметрів можуть бути отримані з виразів передатних функцій регуляторів:

$$W_{рш}(p)=\frac{T_{м}k_{т}}{8T_{μш}^{2}k_{ш}k_{д}R_{\Sigma}}\frac{1+4T_{μш}p}{p}=\frac{4T_{μш}T_{м}k_{т}}{8T_{μш}^{2}k_{ш}k_{д}R_{\Sigma}}+\frac{T_{м}k_{т}}{8T_{μш}^{2}k_{ш}k_{д}R_{\Sigma}}\frac{1}{p}=k_{пш}+\frac{k_{іш}}{p},$$

$$k_{пш}=\frac{4T_{μш}T_{м}k_{т}}{8T_{μш}^{2}k_{ш}k_{д}R_{\Sigma}},\ k_{іш}=\frac{T_{м}k_{т}}{8T_{μш}^{2}k_{ш}k_{д}R_{\Sigma}},$$

$$W_{рт}(p)=\frac{R_{\Sigma}}{2T_{μт}k_{тп}k_{т}}\frac{1+T_{е}p}{p}=\frac{R_{\Sigma}T_{е}}{2T_{μт}k_{тп}k_{т}}+\frac{R_{\Sigma}}{2T_{μт}k_{тп}k_{т}}\frac{1}{p}=k_{пт}+\frac{k_{іт}}{p},$$

$$k_{пт}=\frac{R_{\Sigma}T_{е}}{2T_{μт}k_{тп}k_{т}},\ \ k_{іт}=\frac{R_{\Sigma}}{2T_{μт}k_{тп}k_{т}},$$

При записі системи (4.5) також замість $T_{м}$ і $k_{д}$ записані відповідні вирази

$$T_{м}=J_{\Sigma}\frac{R_{\Sigma}}{\left(k\Phi_{н}\right)^{2}};\ \ k_{д}=\frac{1}{k\Phi_{н}}.$$

Постійними часу датчиків струму $T_{дт}$ і швидкості $T_{дш}$ нехтуємо.

В результаті об'єднання систем (4.4) і (4.5), отримано систему диференціальних рівнянь трьохмасової системи:

$$
\begin{cases}
\dfrac{d\omega_{\text{м}}}{dt} = \dfrac{1}{J_{\text{м}}} M_{\text{пр2}} + \dfrac{\beta_2}{J_{\text{м}}} \omega_{\text{к}} - \dfrac{\beta_2}{J_{\text{м}}} \omega_{\text{м}} - \dfrac{1}{J_{\text{м}}} M_{\text{ст}}; \\
\dfrac{dM_{\text{пр1}}}{dt} = c_1 \omega_{\text{д}} - c_1 \omega_{\text{к}}; \\
\dfrac{d\omega_{\text{к}}}{dt} = \dfrac{1}{J_{\text{к}}} M_{\text{пр1}} - \dfrac{\beta_1}{J_{\text{к}}} \omega_{\text{д}} + \dfrac{\beta_1}{J_{\text{к}}} \omega_{\text{к}} - \dfrac{1}{J_{\text{к}}} M_{\text{пр2}} - \dfrac{\beta_2}{J_{\text{к}}} \omega_{\text{к}} + \dfrac{\beta_2}{J_{\text{к}}} \omega_{\text{м}}; \\
\dfrac{dM_{\text{пр2}}}{dt} = c_2 \omega_{\text{к}} - c_2 \omega_{\text{м}}; \\
\dfrac{d\omega_{\text{д}}}{dt} = \dfrac{k\Phi_{\text{н}}}{J_{\text{д}\Sigma}} I_{\text{д}} - \dfrac{1}{J_{\text{д}\Sigma}} M_{\text{пр1}} - \dfrac{\beta_1}{J_{\text{д}\Sigma}} \omega_{\text{д}} + \dfrac{\beta_1}{J_{\text{д}\Sigma}} \omega_{\text{к}}; \\
\dfrac{dI_{\text{д}}}{dt} = -\dfrac{k\Phi_{\text{н}}}{R_\Sigma T_{\text{е}}} \omega_{\text{д}} - \dfrac{1}{T_{\text{е}}} I_{\text{д}} + \dfrac{1}{R_\Sigma T_{\text{е}}} U_{\text{тп}}; \\
\dfrac{dU_{\text{тп}}}{dt} = -\dfrac{k_{\text{пш}} k_{\text{пт}} k_{\text{тп}} k_{\text{ш}}}{T_{\mu\text{т}}} \omega_{\text{д}} - \dfrac{k_{\text{т}} k_{\text{пт}} k_{\text{тп}}}{T_{\mu\text{т}}} I_{\text{д}} - \dfrac{1}{T_{\mu\text{т}}} U_{\text{тп}} - \dfrac{k_{\text{тп}}}{T_{\mu\text{т}}} U_{\text{рт}} - \dfrac{k_{\text{тп}} k_{\text{пт}}}{T_{\mu\text{т}}} U_{\text{рш}} + \dfrac{k_{\text{пш}} k_{\text{пт}} k_{\text{тп}}}{T_{\mu\text{т}}} U_{\text{з}}; \\
\dfrac{dU_{\text{рт}}}{dt} = -k_{\text{ш}} k_{\text{пш}} k_{\text{іт}} \omega_{\text{д}} - k_{\text{іт}} k_{\text{т}} I_{\text{д}} + k_{\text{іт}} U_{\text{рш}} + k_{\text{пш}} k_{\text{іт}} U_{\text{з}}; \\
\dfrac{dU_{\text{рш}}}{dt} = -k_{\text{ш}} k_{\text{іш}} \omega_{\text{д}} + k_{\text{іш}} U_{\text{з}}.
\end{cases}
$$

Алгоритмічна схема трьохмасової системи показана на рис. 4.2. На схемі замість $M_{\text{ст}}$ показаний струм статичного навантаження $I_{\text{ст}}=M_{\text{ст}} / (k\Phi_{\text{н}})$.

Для дослідження динамічних характеристик трьохмасової системи використано пакет прикладних програм MATLAB. Розрахунок виконано з використанням рівнянь стану системи, а також моделі Simulink системи, зображеної на рис.4.3. Графіки перехідних процесів перших шести змінних стану системи по задаючій і збурюючій діях приведені на рис.4.4 і рис.4.5. Для формування вхідного сигналу $U_{\text{з}}$ і сигналу збурення $M_{\text{ст}}$ 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 швидкості другої і третьої маси, а також моментів пружності мають характер слабозатухаючих коливань. По збурюючій дії перехідні процеси всіх змінних стану мають коливальний характер, що недопустимо за умовами експлуатації підіймальної установки

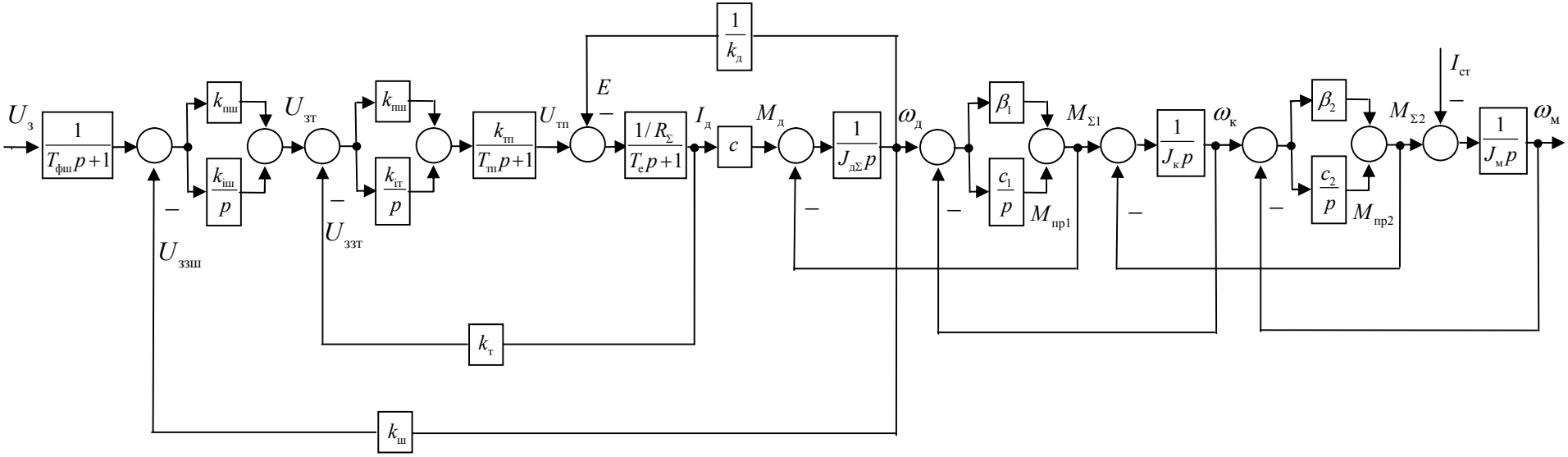


Рис.4.2. Алгоритмічна схема трьохмасової системи

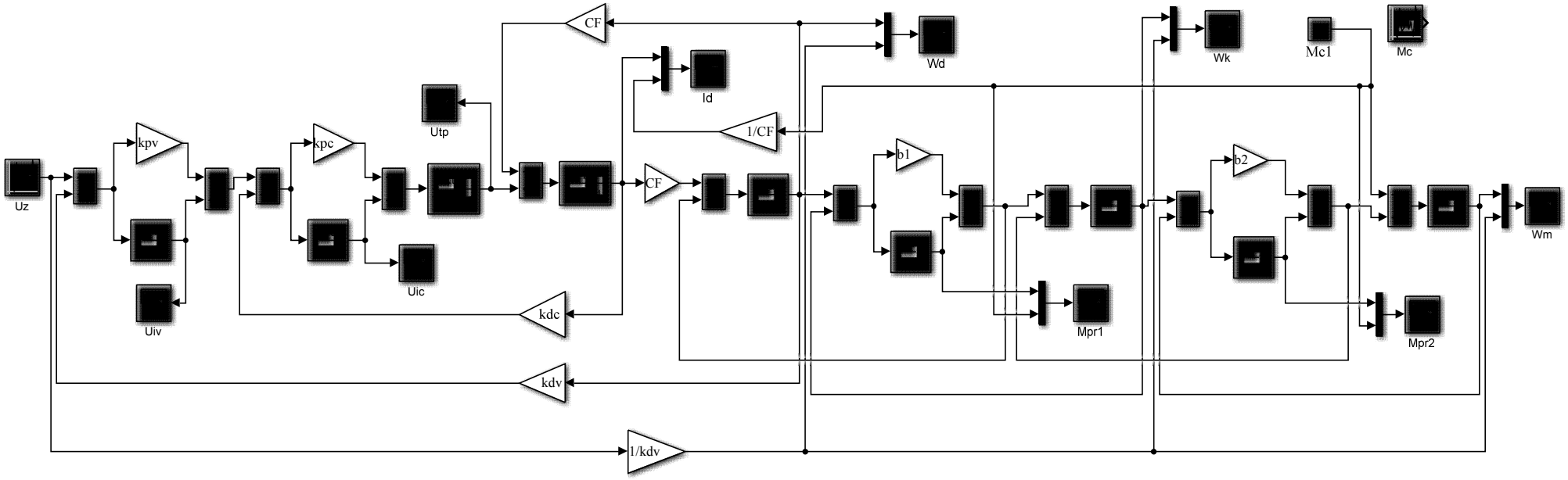
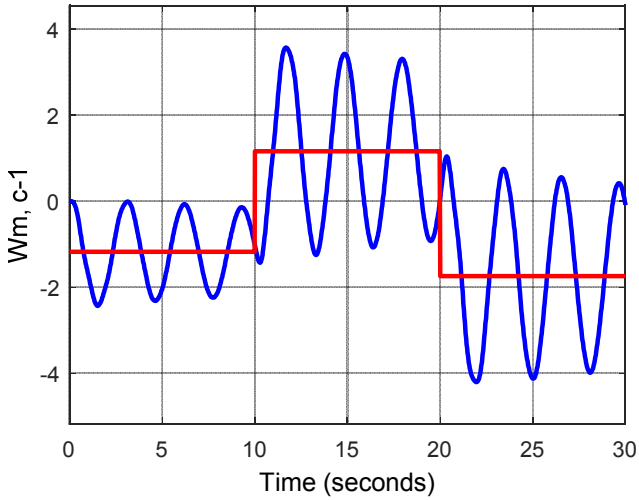
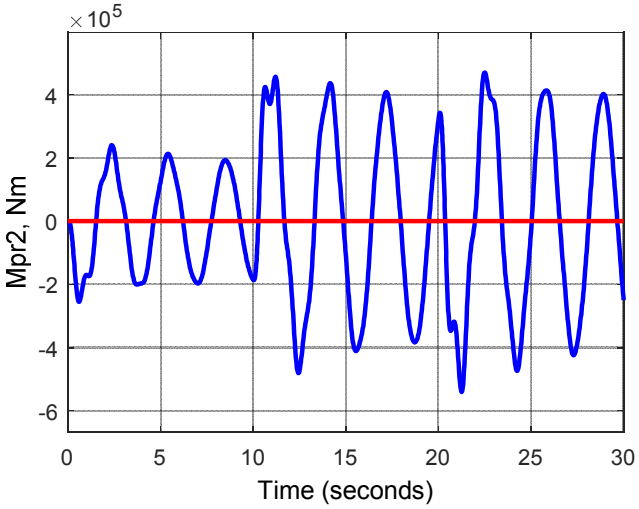


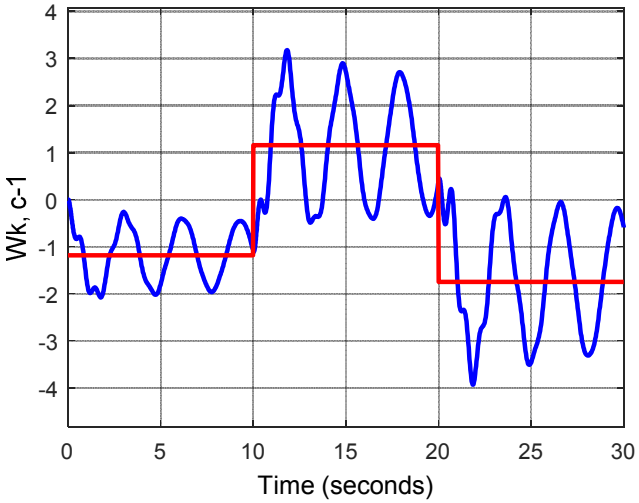
Рис. 4.3. Схема моделі трьохмасової системи, розроблена в Simulink системи MATLAB



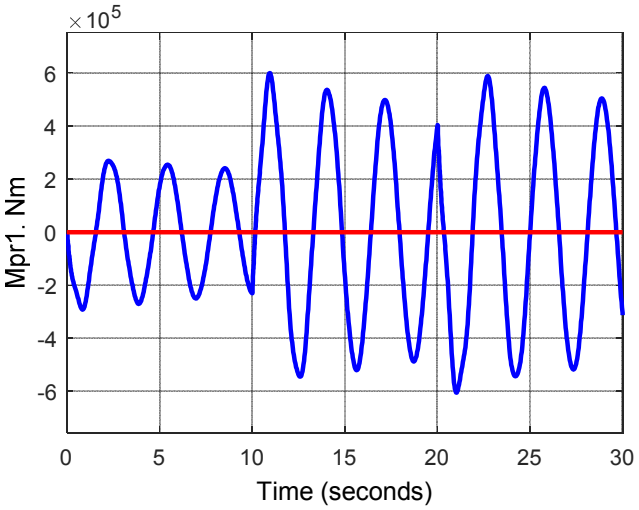
швидкість механізму по задаючій дії



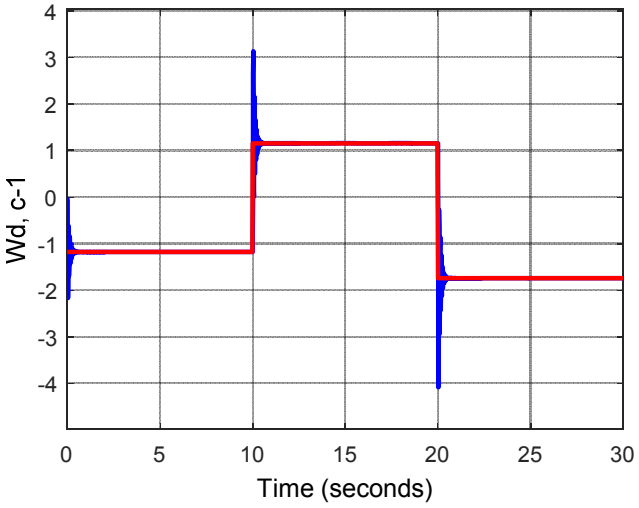
другий момент пружності по задаючій дії



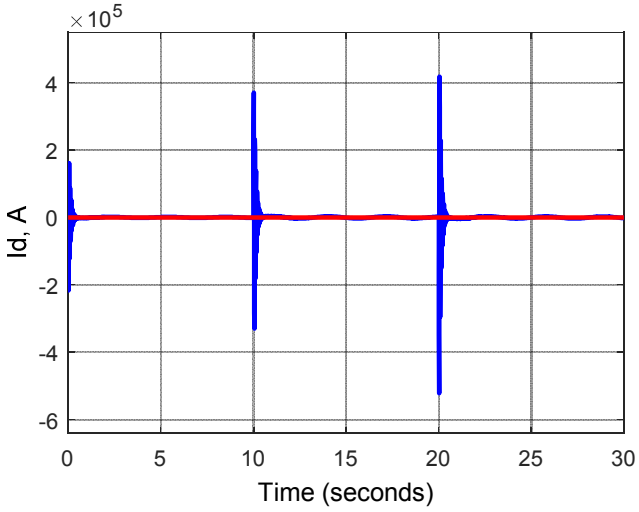
швидкість канату по задаючій дії



перший момент пружності по задаючій дії

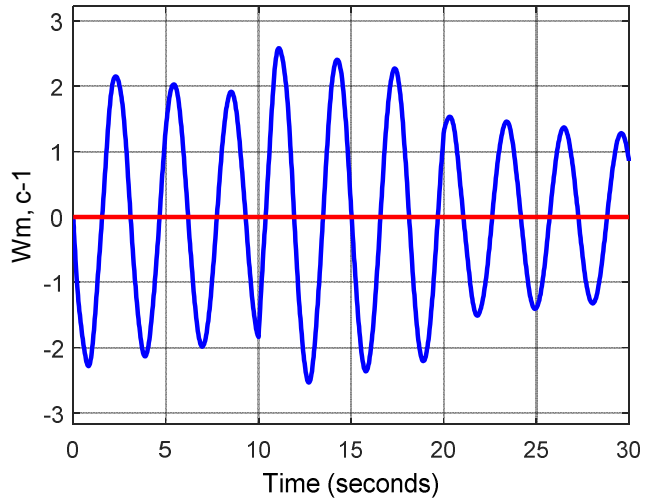


швидкість двигуна по задаючій дії

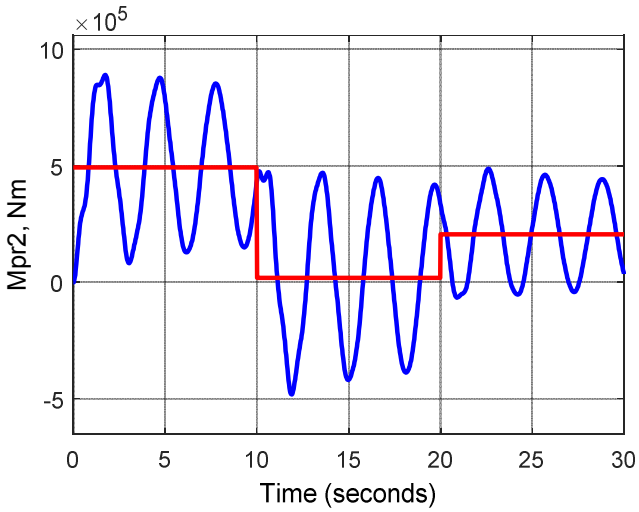


струм двигуна по задаючій дії

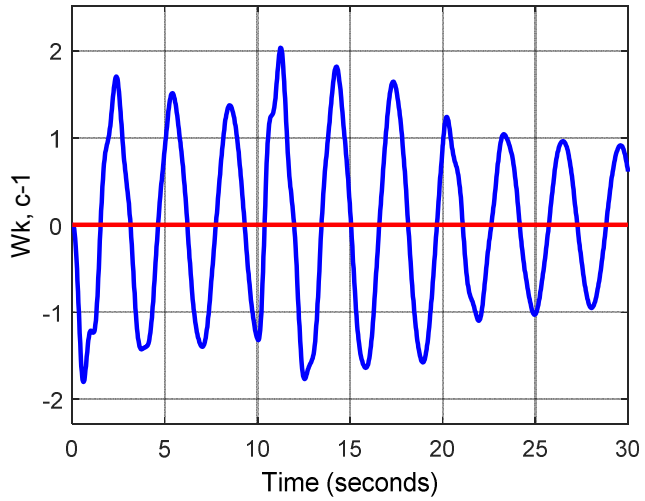
Рис.4.4. Графіки перехідних процесів трьохмасової системи по задаючій дії



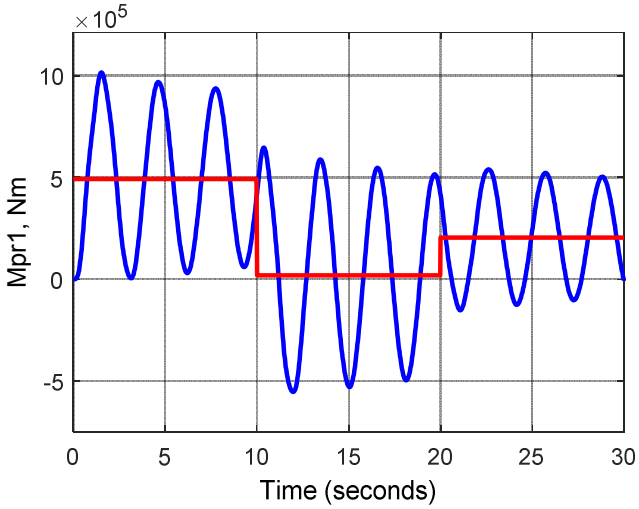
швидкість механізму по збурюючій дії



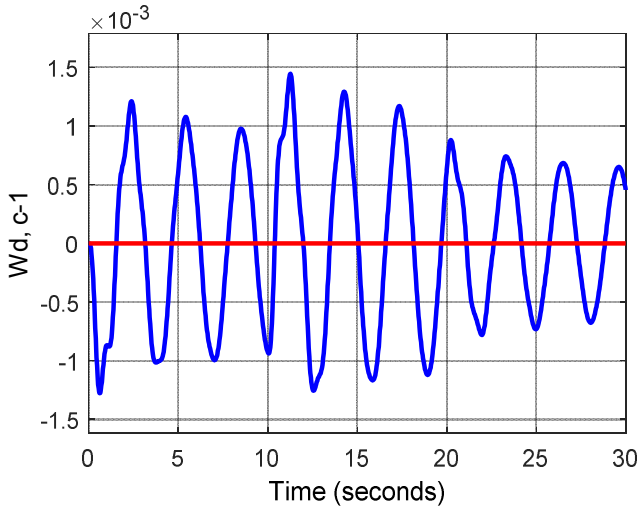
другий момент пружності по збурюючій дії



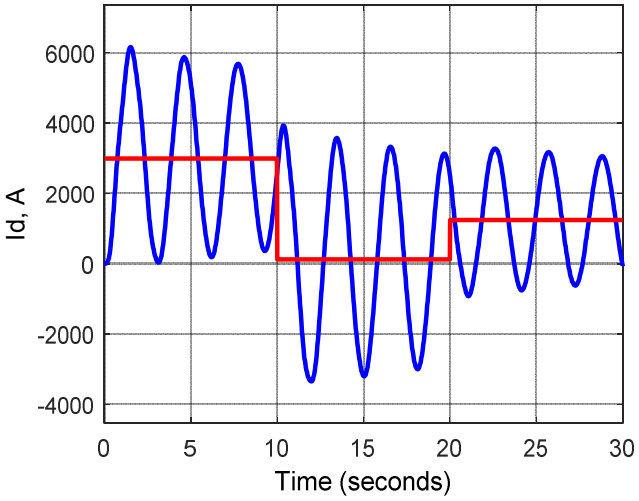
швидкість канату по збурюючій дії



перший момент пружності по збурюючій дії



швидкість двигуна по збурюючій дії



струм двигуна по збурюючій дії

Рис.4.5. Графіки перехідних процесів трьохмасової системи по збурюючій дії

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

Синтез системи управління проводився за допомогою **Neural Network Toolbox** в системі MATLAB (версія MATLAB 2016a). Далі коротко описується цей інструментарій, процес синтезу нейрорегулятора та висвітлені результати моделювання системи управління.

В Neural Network Toolbox MATLAB реалізовано три регулятори:

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

Використання нейронних мереж у задачах управління може бути розкладено на два етапи проектування:

- етап ідентифікації об'єкта управління;
- етап синтезу правила управління.

На етапі ідентифікації створюється модель керованого об'єкту у вигляді нейронної мережі, яка потім використовується на етапі синтезу для розробки регулятора. Для кожної з трьох архітектур використовується той самий процес ідентифікації, проте етапи синтезу суттєво відрізняються.

При управлінні з прогнозом, модель керованого об'єкту використовується для прогнозування його майбутньої поведінки, а алгоритм оптимізації застосовується для обчислення управління, яке мінімізує різницю між бажаними та фактичними змінами виходу моделі.

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

При управлінні на основі еталонної моделі, регулятор - це навчена нейронна мережа, яка управляє об'єктом так, щоб він відтворював поведінку еталонної моделі. У цьому випадку модель керованого об'єкту активно використовується для налаштування параметрів самого регулятора.

Динамічні моделі систем управління з нейромережевими регуляторами розміщені в спеціальному розділі **Control Systems Neural Network Toolbox** (рис.4.6).

Регулятор з прогнозом використовує модель керованого процесу у вигляді нейронної мережі для передбачення майбутньої реакції процесу на випадкові управляючі сигнали. Алгоритм оптимізації визначає керуючі сигнали, які мінімізують відмінність між очікуваними та фактичними змінами виходу моделі, тим самим оптимізуючи управління процесом. Побудова моделі керованого процесу автономно виконується за допомогою нейронної мережі, яка навчається в груповому режимі за одним з алгоритмів навчання. Цей тип регулятора потребує значних обчислювальних ресурсів, оскільки оптимізація відбувається на кожному такті управління.

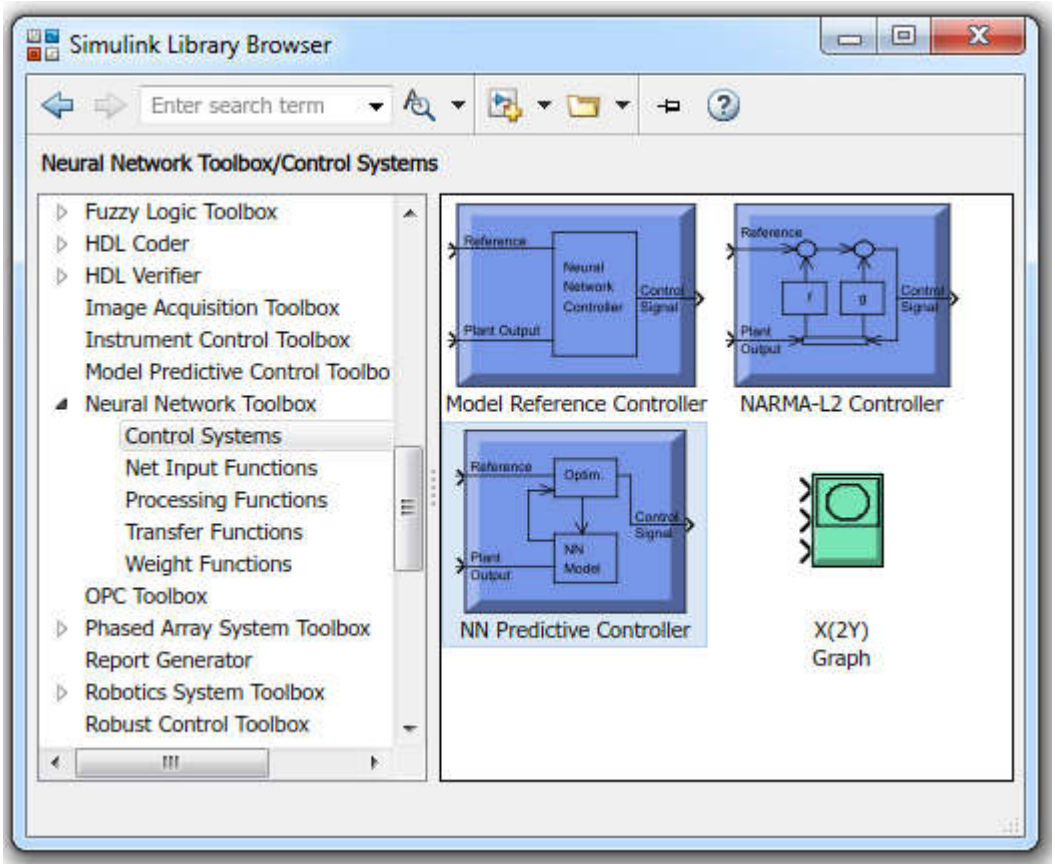


Рис.4.6. Блоки **Neural Network Blocksets** розділу **Control Systems** системи SIMULINK

Регулятор на основі моделі авторегресії з ковзаючим середнім вимагає найменшої кількості обчислень серед всіх типів регуляторів. Цей регулятор є простою реконструкцією нейромережевої моделі керованого об'єкту, яка була отримана під час автономної ідентифікації. Його обмеження полягає у тому, що модель об'єкту повинна бути визначена у канонічній формі простору стану, що відповідає певній матриці, що може призводити до обчислювальних похибок.

Регулятор на основі еталонної моделі вимагає обчислювальних ресурсів, що порівнюються з попереднім типом. Однак архітектура цього регулятора потребує навчання нейронної мережі як для об'єкту управління, так і для регулятора. Навчання регулятора виявляється досить складним, оскільки воно ґрунтується на динамічному варіанті методу зворотного розповсюдження помилок. Перевагою регуляторів на основі еталонної моделі є їхнє застосування до різних класів об'єктів управління.

4.3 Принцип побудови регулятора з прогнозом
NN Predictive Controller

Регулятор **NN Predictive Controller,** що реалізований у пакеті **Neural Network Toolbox**, використовує нейронну мережу для моделювання нелінійної динаміки керованого об'єкту з метою прогнозування його майбутньої поведінки. Більше того,

цей регулятор розраховує сигнал управління, що оптимізує реакцію об'єкту протягом визначеного інтервалу часу.

4.3.1 Ідентифікація об'єкту управління

Схема підсистеми ідентифікації, яка показана на рис. 4.7, включає в себе модель об'єкту управління у формі нейронної мережі, яка повинна пройти автономне навчання для мінімізації різниці між реакціями об'єкту та моделлю $e = y - y_n$ на послідовність тестових сигналів u.

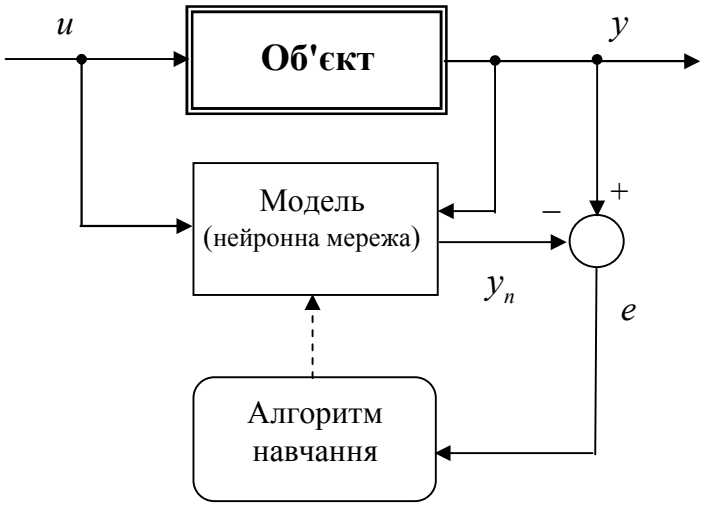


Рис.4.7. Схема підсистеми ідентифікації

Нейронна мережа, що використовується у регуляторі, показана на рис.4.8.Вона має два шари нейронів та використовує лінії затримки (ЛЗ) для збереження попередніх значень входів та виходів об'єкту з метою передбачення майбутніх значень виходу.

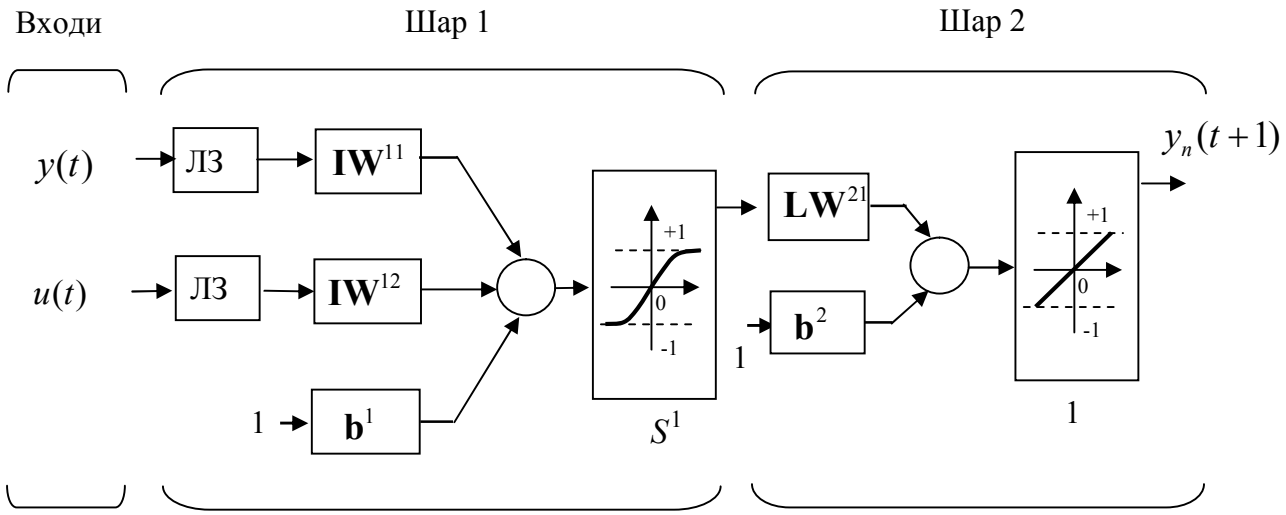


Рис.4.8. Нейронна мережа регулятора керованого об'єкту

Настройка параметрів цієї мережі виконується автономно за допомогою методу групового навчання, використовуючи дані, накопичені під час експериментів з реальним об'єктом. Для навчання мережі може бути використано будь-який із доступних алгоритмів навчання для нейронних мереж. У даній роботі був використаний алгоритм Левенберга-Марквардта, реалізований у вигляді М-функції **trainlm** у пакеті **Neural Network Toolbox**.

4.3.2 Схема управління з прогнозом

Управління з прогнозом використовує принцип передбачення реакції керованого об'єкту на певний майбутній інтервал часу за допомогою нейромережевої моделі. Прогноз використовується у чисельній оптимізації для розрахунку сигналу управління, який мінімізує критерій якості, що визначений таким чином:

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

де константи N_1, N_2 і N_4 визначають межі, в яких обчислюються помилки стеження та потужність управляючого сигналу. Перемінна u' сигнал завдання, що подається не нейромережеву модель, y_r – очікувана, а y_m – фактична реакція моделі системи. Змінна ρ визначає внесок потужності управління у критерій якості. На рис 4.9 показано структурну схему процесу управління з прогнозом, де регулятор складається з нейромережевої моделі керованого об'єкту і блоку оптимізації. Блок оптимізації визначає значення управління, яке мінімізує критерій якості управління, що управляє процесом.

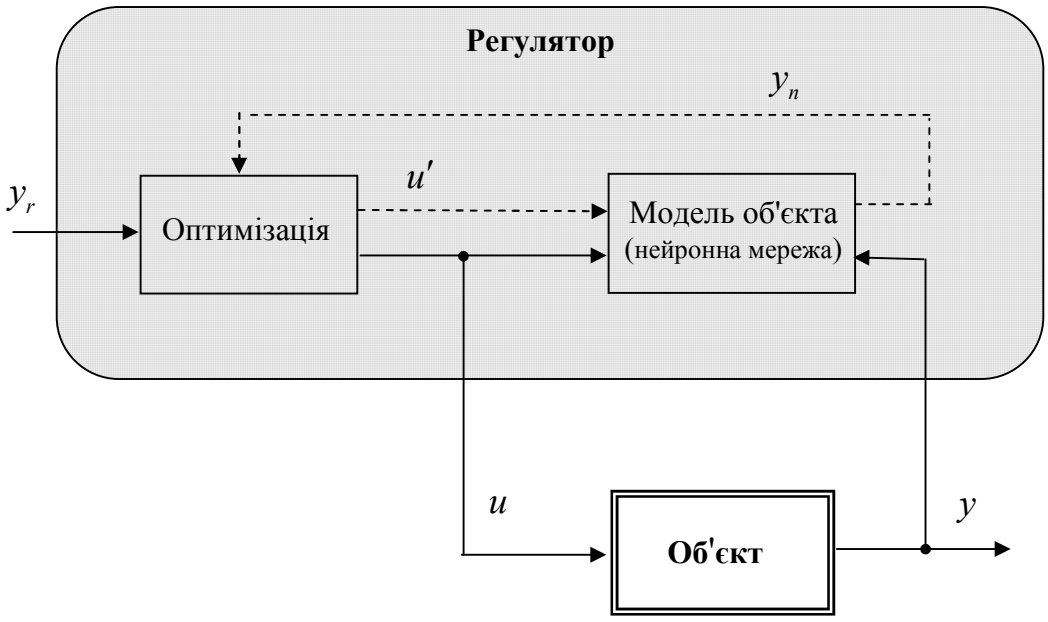


Рис.4.9. Структурна схема системи з регулятором,
що використовує принцип прогнозу

4.4 Синтез нейрорегулятора з прогнозом NN Predictive Controller з використанням системи MATLAB

4.4.1 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.

У вікні системи Simulink формується схема системи із структурою, показаною на рис. 4.10.

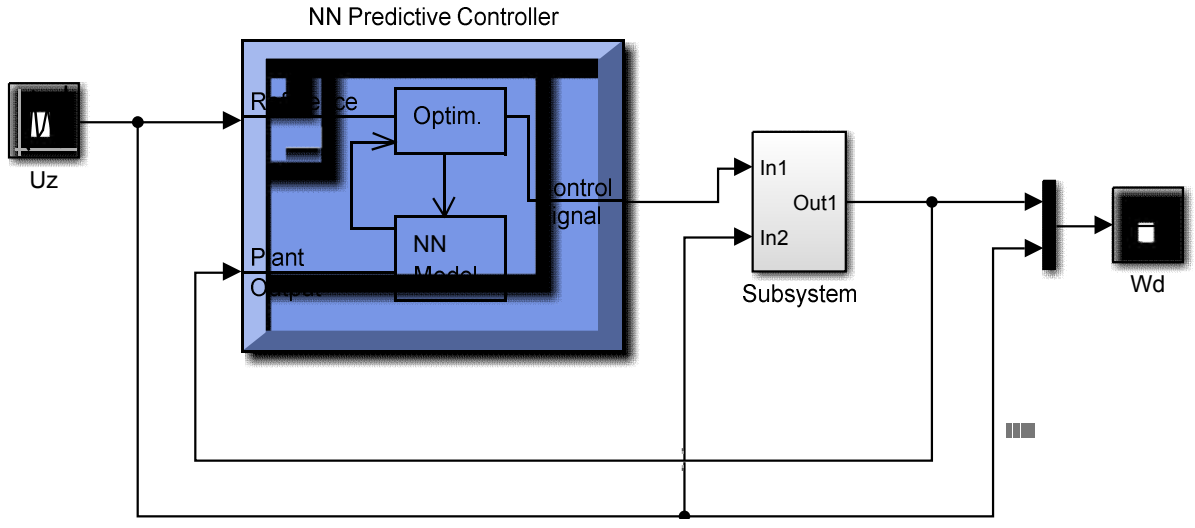


Рис. 4.10. Схема системи управління з нейрорегулятором NN Predictive Controller

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.4.11. Ця схема відображається у відокремленому вікні, яке вибирається через пункт меню **Look Under Mask** блоку NN Predictive Controller.

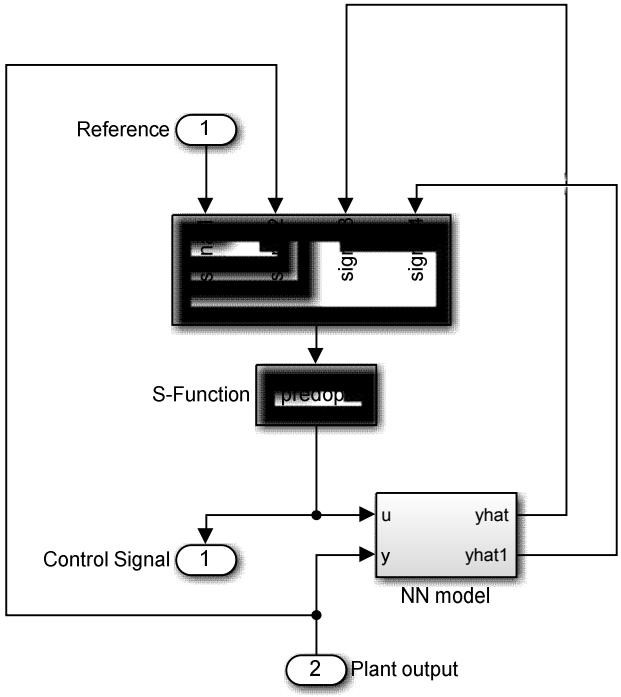


Рис. 4.11.Структурна схеманейрорегулятора

Процес створення нейрорегулятора починається з активації блоку **NN Predictive Controller**. З'являється вікно, показане на рис. 4.12. Це вікно виконує функції графічного інтерфейсу користувача.

У рамці вікна рис. 4.12 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**.

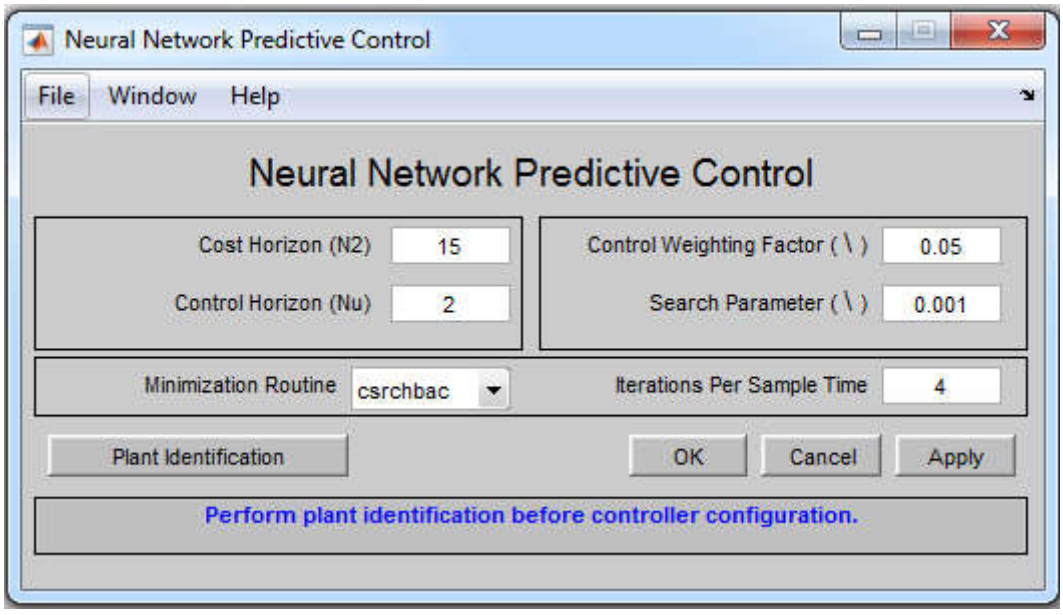


Рис. 4.12. Вікно завдання параметрів регулятора

Вікно **Plant Identification** зображено на рис.4.13. 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 трьхова система.

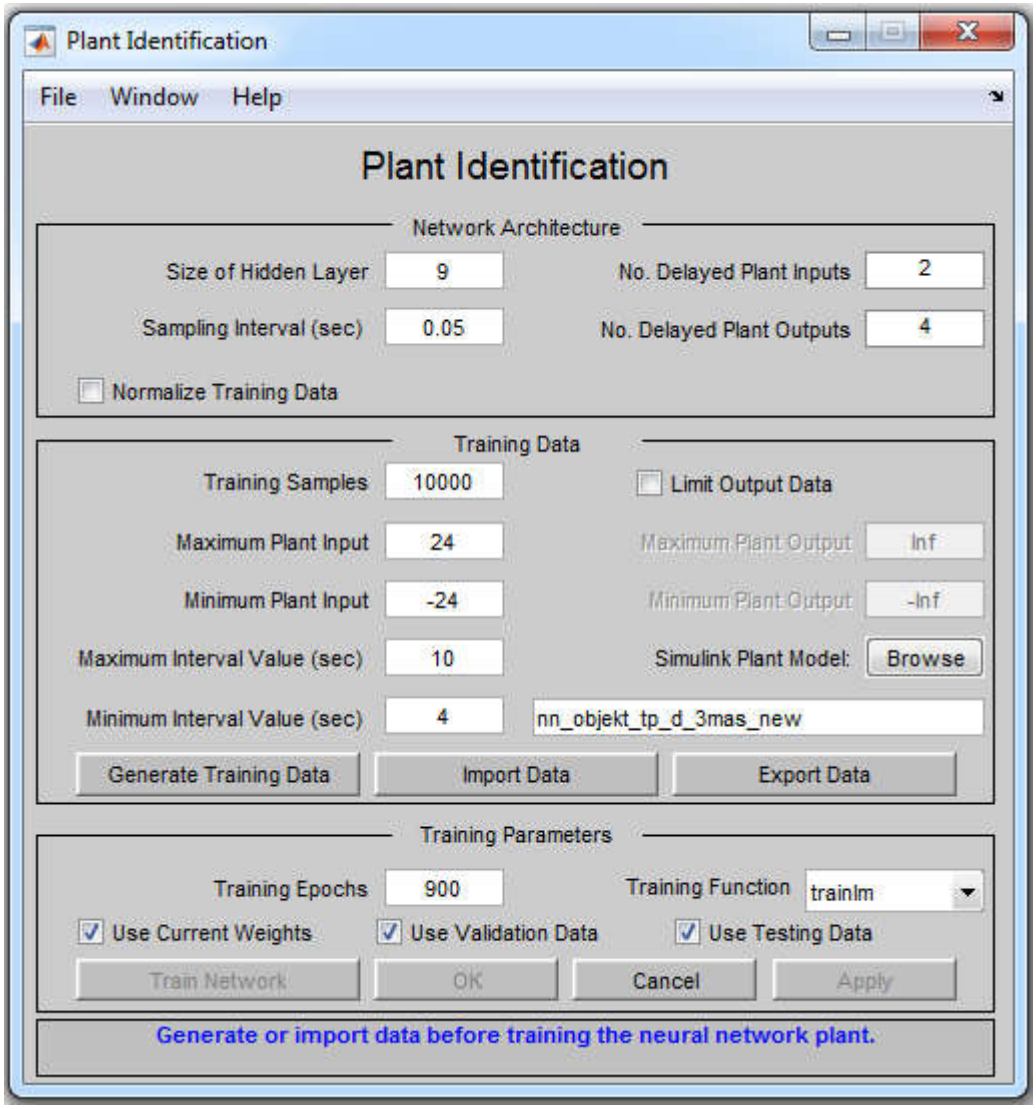


Рис. 4.13.Вікно ідентифікації керованої системи

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. Вибір розміру прихованого шару є важливим параметром при проектуванні нейронної мережі, оскільки він впливає на потужність і здатність мережі моделювати складність взаємозв'язків у ваших даних. Зазвичай, збільшення розміру прихованого шару може допомогти нейронній мережі навчатися більш складним залежностям у даних, але воно також може призвести до перенавчання (overfitting), тобто погіршення загальної здатності мережі генералізувати результати на нові дані.

Розмір прихованого шару часто є гіперпараметром, який може бути експериментально налаштований для досягнення найкращих результатів у конкретній задачі.

Sampling Interval (sec) (інтервал вибірки в секудах). Параметр Sampling Interval (sec) визначає інтервал між двома послідовними моментами знімання даних. Параметр важливий при зборі даних для моделювання або ідентифікації системи, оскільки впливає на точність та роздільну здатність зібраних даних. Для точної роботи моделі ідентифікації важливо визначити правильний інтервал вибірки, який враховує динаміку системи і вимоги до точності результатів ідентифікації.

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. Запізнені входи враховують затримки у вхідних сигналах системи і допомагають створити більш точну модель системи.

Запізнені входи особливо важливі в ідентифікації систем, де існують затримки між вхідними сигналами та відповідними виходами. Наприклад, в автоматичному керуванні часто є затримки в активних процесах, і ідентифікація таких систем може вимагати врахування цих затримок.

Кількість No. Delayed Plant Inputs вказує, скільки вхідних сигналів системи вважаються запізненими і будуть використовуватися при створенні моделі системи. Від правильного визначення цього параметра залежить точність ідентифікації системи, особливо в умовах існування затримок між сигналами.

Цей параметр важливий для налаштування ідентифікаційного процесу та допомагає створити більш адекватну модель системи, яка може враховувати реальні умови та характеристики системи керування.

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. Запізнені виходи враховують затримки у виходних сигналах системи і допомагають створити більш точну модель системи.

Цей параметр дозволяє визначити, скільки затриманих виходів системи вважаються при ідентифікації і впливає на точність та адекватність створеної моделі. Затримані виходи дозволяють враховувати затримки в виходних сигналах, які можуть бути важливими для ефективної ідентифікації та моделювання системи.

Вибір кількості No. Delayed Plant Outputs залежить від конкретних характеристик досліджуваної системи та вимог до точності ідентифікації.

Normalize Training Data (нормалізація даних навчання). Вікна контролю нормування навчальних даних. Normalize Training Data вказує на необхідність нормалізації даних під час навчання моделі системи. Нормалізація даних - це процес масштабування вхідних та вихідних даних системи до певного діапазону або стандартної форми перед подачею їх на вхід моделі для навчання.

Якщо вікно увімкнено, то вхідні та вихідні дані будуть нормалізовані перед початком процесу навчання моделі. Нормалізація даних може включати в себе масштабування даних до діапазону від 0 до 1, використання стандартного відхилення та середнього значення для центрування даних, або інші методи нормалізації.

Якщо вікно не увімкнено, то нормалізація даних не буде застосована, і дані будуть використовуватися без змін.

Мета нормалізації даних під час навчання полягає в тому, щоб забезпечити стабільність та швидкість навчання моделі, особливо коли вхідні дані мають різний масштаб або діапазон значень. Нормалізація може полегшити збіжність навчання та поліпшити точність ідентифікації системи.

Вибір варіанта Normalize Training Data повинен здійснюватися відповідно до характеру та потреб конкретної задачі ідентифікації 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.

Основна мета Training Data полягає в навчанні моделі системи таким чином, щоб вона могла адекватно відтворювати динаміку реальної системи. Дані включають в себе виміряні значення входів та виходів системи протягом певного періоду часу або при певних умовах.

Зазвичай Training Data містять дані з різних режимів роботи системи, щоб модель була здатна відтворювати різноманітні аспекти системи. Важливо, щоб ці дані були репрезентативними та відображали різноманітні умови функціонування системи.

На основі цих даних проводиться процес ідентифікації, під час якого створюється модель, яка намагається знайти математичні зв'язки між вхідними та вихідними сигналами системи. Точність та адекватність цієї моделі залежать від якості та репрезентативності Training Data.

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. Довжина навчальної вибірки

грає важливу роль у процедурі ідентифікації об'єкта і створенні адекватної 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.

Максимальне значення вхідного сигналу важливо вказати для того, щоб навчальний процес був обмежений реалістичними значеннями вхідного сигналу. Вказане значення може відображати фізичні обмеження системи або діапазони значень, які найчастіше використовуються в реальних умовах.

Важливо правильно налаштувати цей параметр, оскільки надмірно велике значення може призвести до надмірного навчання моделі на діапазонах вхідних сигналів, які надто віддалені від реальних умов. Це може призвести до некоректних прогнозів та погіршення якості моделі системи.

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.

Максимальне і мінімальне значення вхідного сигналу важливо вказати для того, щоб навчальний процес був обмежений реалістичними значеннями вхідного сигналу. Вказане значення може відображати фізичні обмеження системи або діапазони значень, які найчастіше використовуються в реальних умовах. Неправильне встановлення цих параметрів може призвести до некоректних прогнозів та погіршення якості моделі системи.

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.

Вказане максимальне значення в секундах служить для обмеження часового діапазону аналізу інформації. Це корисно в ситуаціях, коли вхідні та вихідні дані мають значення протягом певного часового інтервалу, і деякий період часу може бути менш важливим для ідентифікації системи.

Зазвичай, під час встановлення параметру Maximum Interval Value, враховуються характеристики конкретної ідентифікаційної задачі. Навчальний набір даних може включати в себе декілька циклів або періодів роботи системи, і цей параметр допомагає обмежити аналіз до конкретного періоду часу.

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 ди-

наміки системи, тобто мінімальний інтервал ідентифікації. Цей параметр дозволяє обмежити часовий період, на який поширюється навчальний набір даних, використовуваний під час ідентифікації.

Мінімальне значення вказує на мінімальний період, який вважається допустимим для аналізу даних. Налаштування Minimum Interval Value (sec) допомагає контролювати, які частини часового ряду вважати допустимими для використання під час ідентифікації, і може допомогти враховувати тільки ті інтервали, які мають певну мінімальну тривалість або важливість.

Цей параметр важливий для налаштування ідентифікаційного процесу відповідно до конкретних вимог і особливостей досліджуваної системи.

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.

Параметри Maximum Plant Output і Minimum Plant Output корисні, коли потрібно провести аналіз лише на певному діапазоні вихідних даних, або коли дослідника цікавить певний аспект системи, що відбувається в певний період часу.

Параметри Maximum Plant Output і Minimum Plant Output дозволяють зменшити обсяг даних, які використовуються для ідентифікації, і спростити процес 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. 4.14.

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.

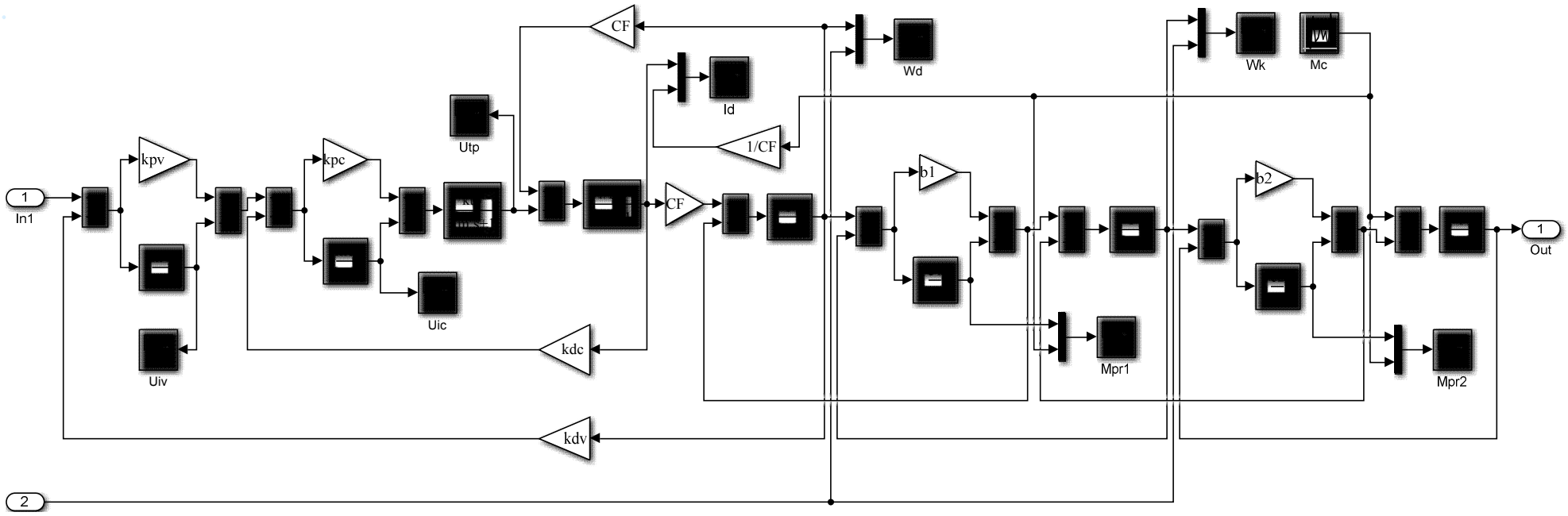


Рис. 4.14. Модель об'єкту управління нейрорегулятора

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.

Кількість Training Epochs визначає, скільки разів модель або алгоритм пройде через навчальні дані для покращення своєї здатності до відтворення динаміки системи. Зазвичай навчання включає багато епох, і після кожної епохи модель оцінюється на тестових даних, щоб визначити, наскільки добре вона справляється з завданням ідентифікації.

Вибір кількості Training Epochs може бути важливим параметром при налаштуванні процесу навчання. Занадто мала кількість епох може призвести до недоосвіченої моделі, яка не здатна адекватно відтворити динаміку системи, тоді як занадто велика кількість епох може призвести до перенавчання (overfitting) моделі на навчальних даних, що погіршує її здатність до узагальнення на нові дані. Точна кількість епох може варіюватися в залежності від конкретної задачі ідентифікації.

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.

Вибір правильної Training Function важливий для успішного навчання моделі. У MATLAB і Neural Network Toolbox, доступні різні Training Functions для нейронних мереж та інших моделей ідентифікації, наприклад: traingd - градієнтний спуск; traingdm - градієнтний спуск з моментом; trainlm - метод найменших квадратів Левенберга-Марквардта; trainbr - байєсівський регулятор.

Кожна з цих функцій має свої особливості та параметри, які можна налаштовувати для досягнення найкращих результатів в конкретній задачі ідентифікації. Training Function визначає, як модель або алгоритм оновлює внутрішні параметри під час навчання та яким чином він збільшує точність та здатність моделі до адекватного відтворення динаміки системи.

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.

Якщо Use Current Weights встановлено в «Так» (або включено), то попередні ваги (параметри моделі) вважаються початковими значеннями для нового процесу навчання. Це означає, що процес навчання розпочнеться з вагами, які вже були

встановлені або навчені під час попередніх ітерацій.

Якщо Use Current Weights встановлено в «Ні» (або виключено), то навчання розпочнеться зі звичайними початковими значеннями, які зазвичай є випадковими або ініціалізованими за допомогою певних методів.

Використання поточних ваг при навчанні може бути корисним, якщо необхідно покращити або доповнити існуючу модель, або якщо відомо, що поточні ваги вже набули певного рівня адекватності. Вважливо розуміти, як цей параметр впливає на процес навчання та на результати ідентифікації системи.

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.

Якщо Use Validation Data встановлено в «Так» (або включено), то під час навчання моделі буде використовуватися окремий набір даних, який не брав участь в процесі навчання. Ці дані використовуються для оцінки того, наскільки добре модель узгоджується з даними, які вона раніше не бачила. Використання валідаційних даних допомагає виявити ознаки перенавчання моделі та визначити, наскільки добре модель узгоджується з новими вхідними даними.

Якщо Use Validation Data встановлено в «Ні» (або виключено), то валідація не буде використовуватися, і модель буде оцінюватися лише на основі навчальних даних. Використання валідаційних даних рекомендується для кращої оцінки якості моделі та попередження перенавчання, але варіант без валідації може бути використаний, коли недоступні додаткові дані для цієї цілі.

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.

Якщо Use Testing Data встановлено в «Так» (або включено), то після завершення навчання і валідації моделі буде використовуватися окремий набір тестових даних для оцінки якості моделі. Ці дані допомагають визначити, наскільки добре модель здатна передбачати реальну динаміку системи, якщо у неї немає попередніх знань про ці дані.

Якщо Use Testing Data встановлено в «Ні» (або виключено), то тестові дані не використовуються, і модель оцінюється лише на основі навчальних і, можливо, валідаційних даних.

Використання тестових даних важливе для об'єктивної оцінки якості моделі та підтвердження її ефективності на нових, незалежних вибірках даних. Тестові дані допомагають уникнути перенавчання та підтвердити, наскільки добре модель може працювати на нових вхідних даних.

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 керова-

ного об'єкту. Графіки входового та вихідного сигналів об'єкту управління відображаються на екрані (див. рис. 4.15). Після завершення процесу створення навчальної послідовності користувач має можливість прийняти (**Accept Data**) або відхилити (**Reject Data**) згенеровані дані.

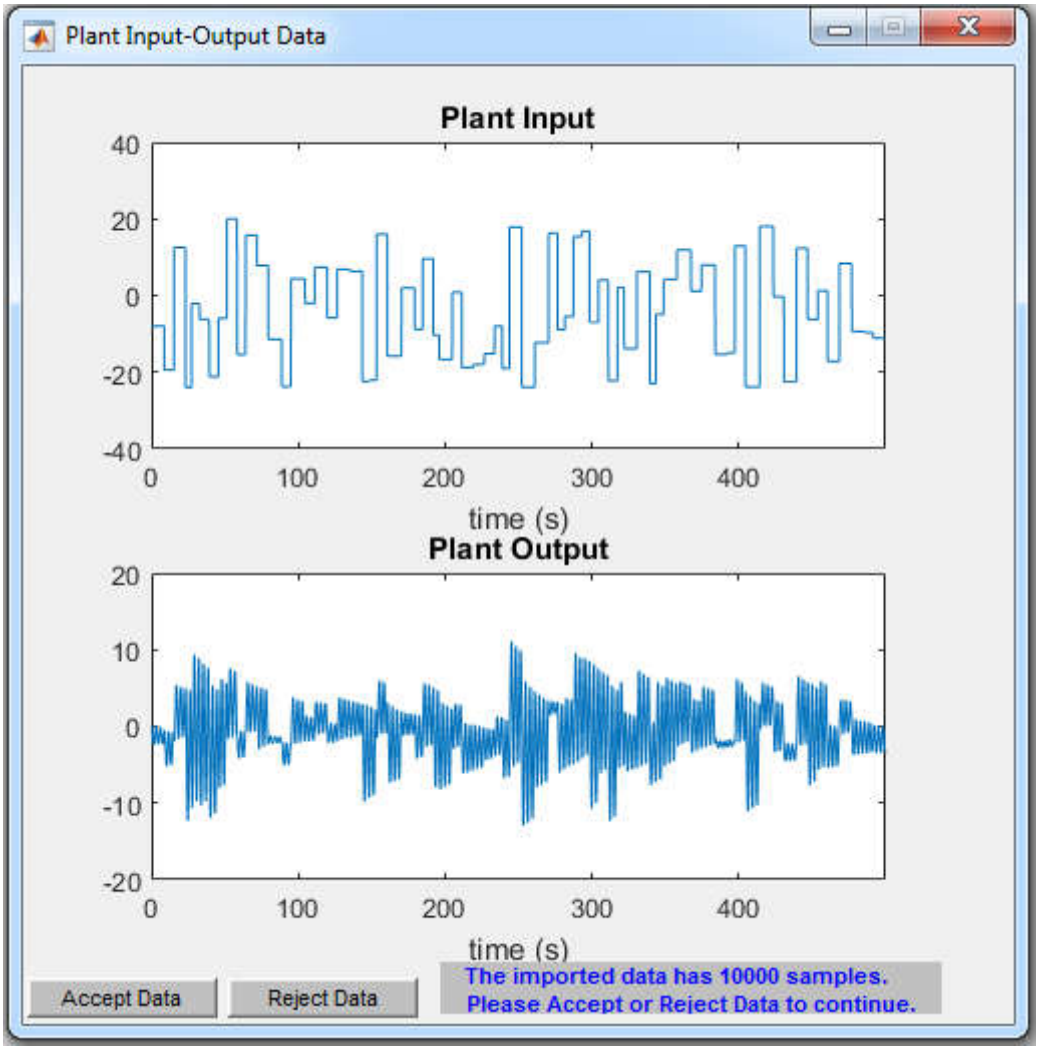


Рис. 4.15. Графіки вхідного і вихідного сигналів при генерації навчальної послідовності

Якщо дані прийняті, програма відкриє модифіковане вікно **Plant Identification** (див. рис.4.16). У цьому вікні деякі елементи стають недоступними, а кнопка Generate Training Data замінюється кнопкою Erase Generate Data, яка дає можливість видалити згенеровані дані.

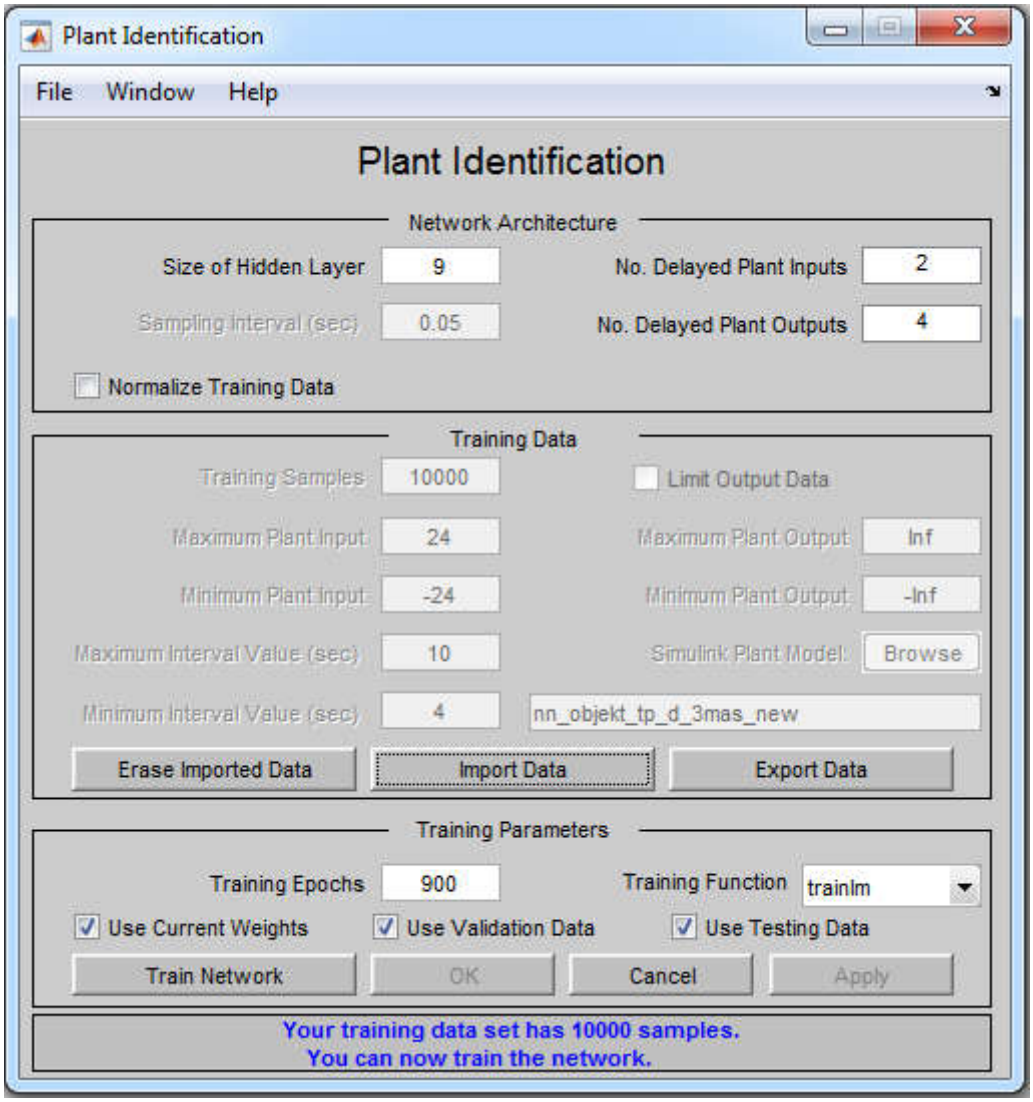


Рис.4.16. Вид вікна Plant Identification після генерації навчальної послідовності

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. 4.17).

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. 4.18.

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.

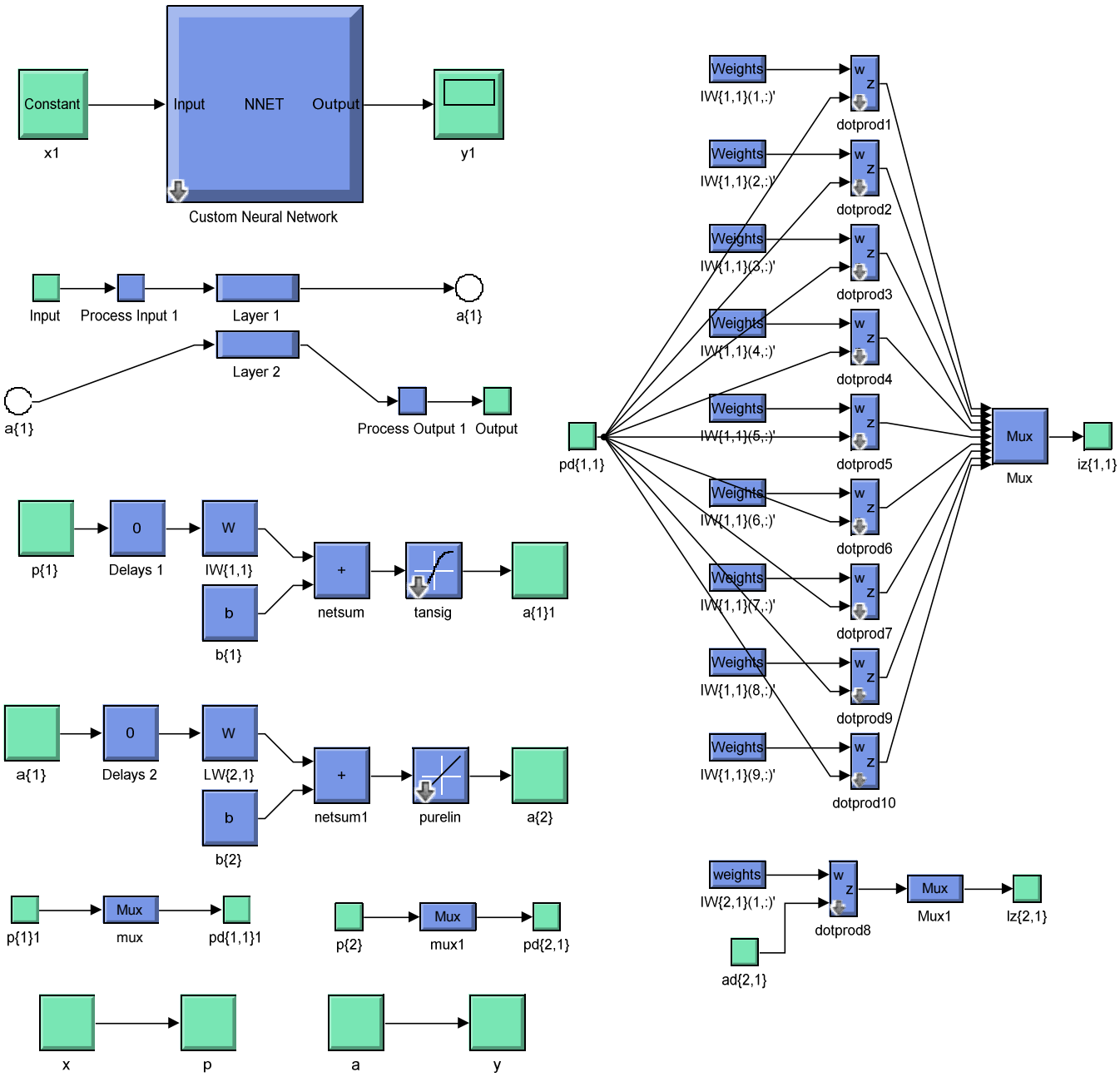


Рис. 4.17. Моделі елементів мережі з прямою передачею сигналу, реалізовані в Simulink

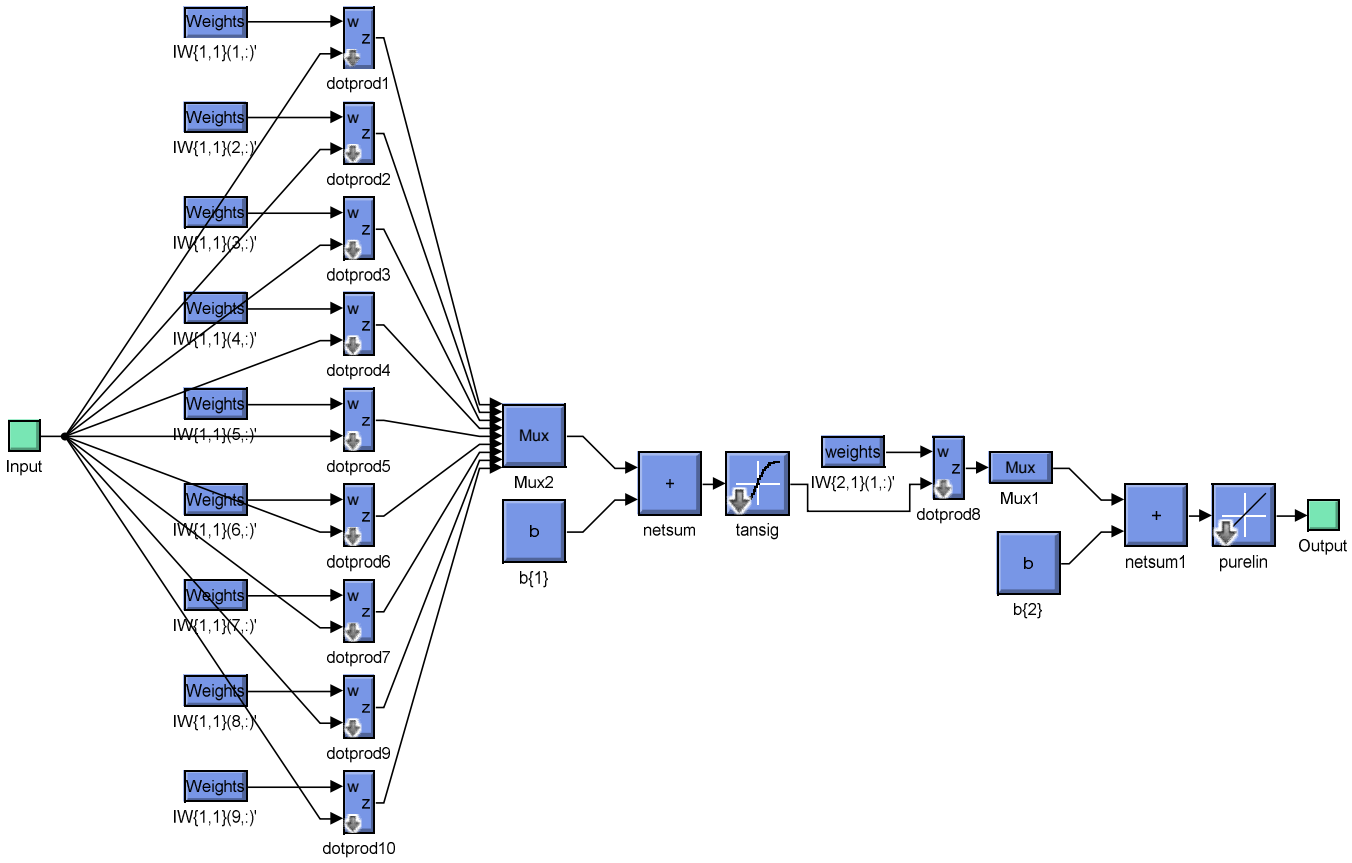


Рис. 4.18. Модель статичної мережі з прямою передачею сигналу, побудована в Simulink

Після створення мережі починається процес її навчанні. Процес та результати навчання наочно демонструються за допомогою діалогового вікна **Neural Network Training (nntraintool)**, показаного на рис..4.19. 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.

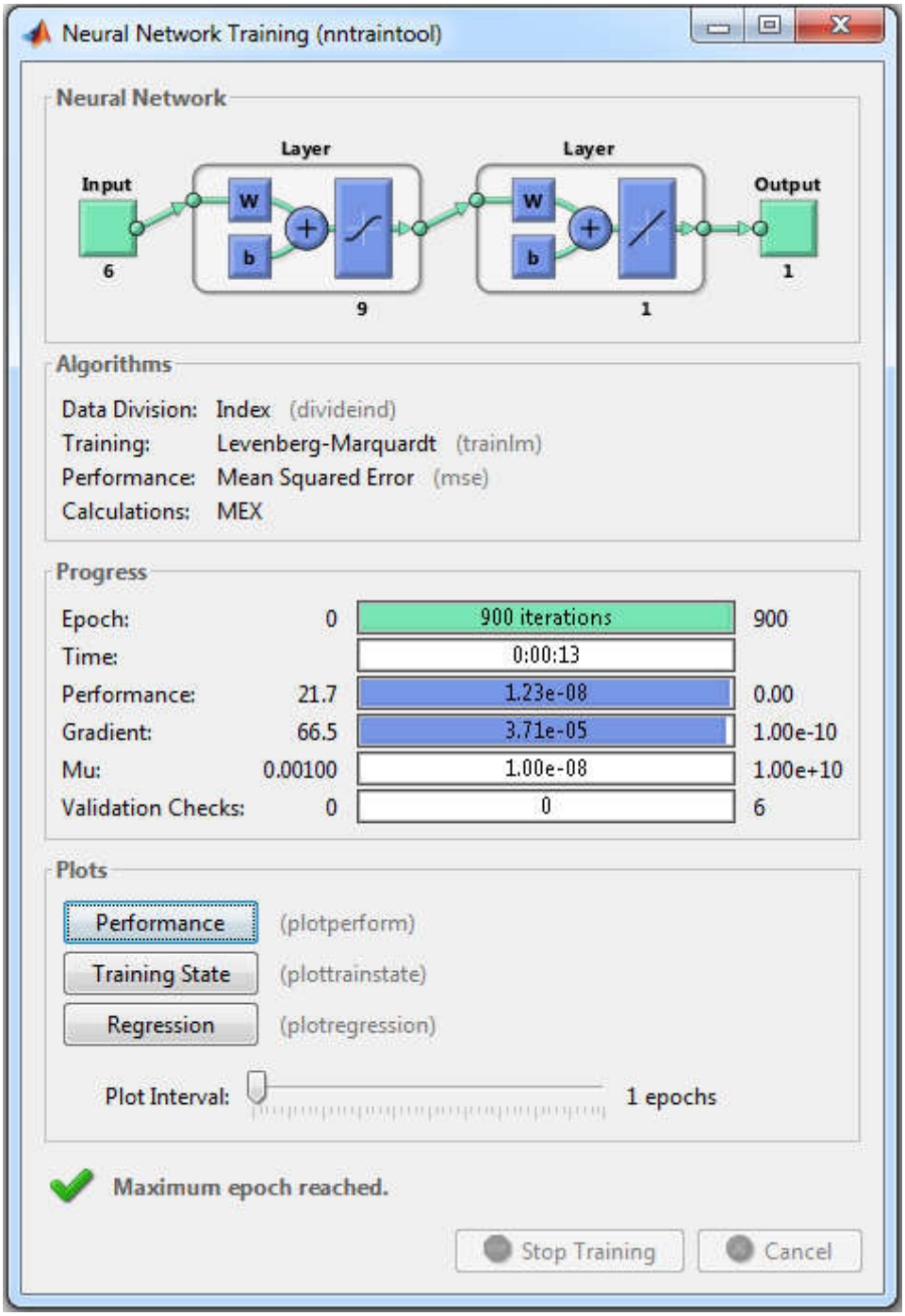


Рис. 4.19. Головне вікно, що відображає хід навчання
та результати навчання нейронної мережі

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. Формула MSE виглядає наступним чином:

$$MSE = \frac{\sum_{i=1}^{N} (y_{actual_i} - y_{predicted_i})^2}{N},$$

де N - кількість прикладів даних в наборі; y_{actual_i} - фактичні значення; $y_{predicted_i}$ - прогнозовані значення, отримані від нейронної мережі.

MSE вимірює середню квадратичну відстань між фактичними і прогнозованими значеннями. Чим менше значення MSE, тим краще модель адаптується до даних. Під час навчання нейронної мережі, оптимізаційний алгоритм намагається мінімізувати MSE, тобто підганяти прогнозовані значення, щоб вони були ближчі до фактичних.

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.

В контексті навчання нейронних мереж Calculations: MEX вказує, чи використовувати MEX-функції для підвищення швидкості обчислень під час тренування. MEX-функції можуть бути більш швидкими за інтерпретовані мови, такі як MATLAB, і зазвичай вони використовуються для виконання обчислень на рівні мови програмування C або C++.

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. Тау-параметр (Mu) використовується в контексті адаптивних методів оптимізації, таких як метод зворотного розповсюдження і метод Бройдена-Флетчера-Гольдфарба-Шанно (BFGS), для регулювання швидкості навчання. Він дозволяє алгоритму оптимізації адаптуватися до поточної динаміки процесу навчання та забезпечити збіжність навчання навіть при складних функціях втрат або підвищених градієнтах.

Зазвичай значення Mu обчислюється алгоритмом оптимізації під час навчання і може змінюватися на кожній ітерації. Цей параметр є однією з ключових частин процесу оптимізації нейронної мережі та допомагає покращити збіжність та стабільність навчання.

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.

Основна ідея полягає в тому, що під час навчання нейронної мережі може виникнути перенавчання (overfitting), коли модель вивчає «на пам'ять» навчальний набір даних і втрачає здатність узагальнювати на нові дані. Для запобігання перенавчанню, використовуються перевірки якості моделі на окремому наборі даних, який не використовується під час навчання, а називається набором даних для перевірки (validation dataset).

Validation Checks вказують, як часто виконувати перевірки якості моделі під час навчання. Наприклад, якщо встановлено 6 перевірок, то під час навчання нейронної мережі програма буде регулярно перевіряти, наскільки добре модель працює на наборі даних для перевірки після кожних 6 епох навчання (де епоха - це один прохід через весь навчальний набір даних). Якщо якість моделі на наборі даних для перевірки перестає покращуватися або погіршується, то це може бути індикатором перенавчання, і навчання може бути припинене.

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. 4.20.

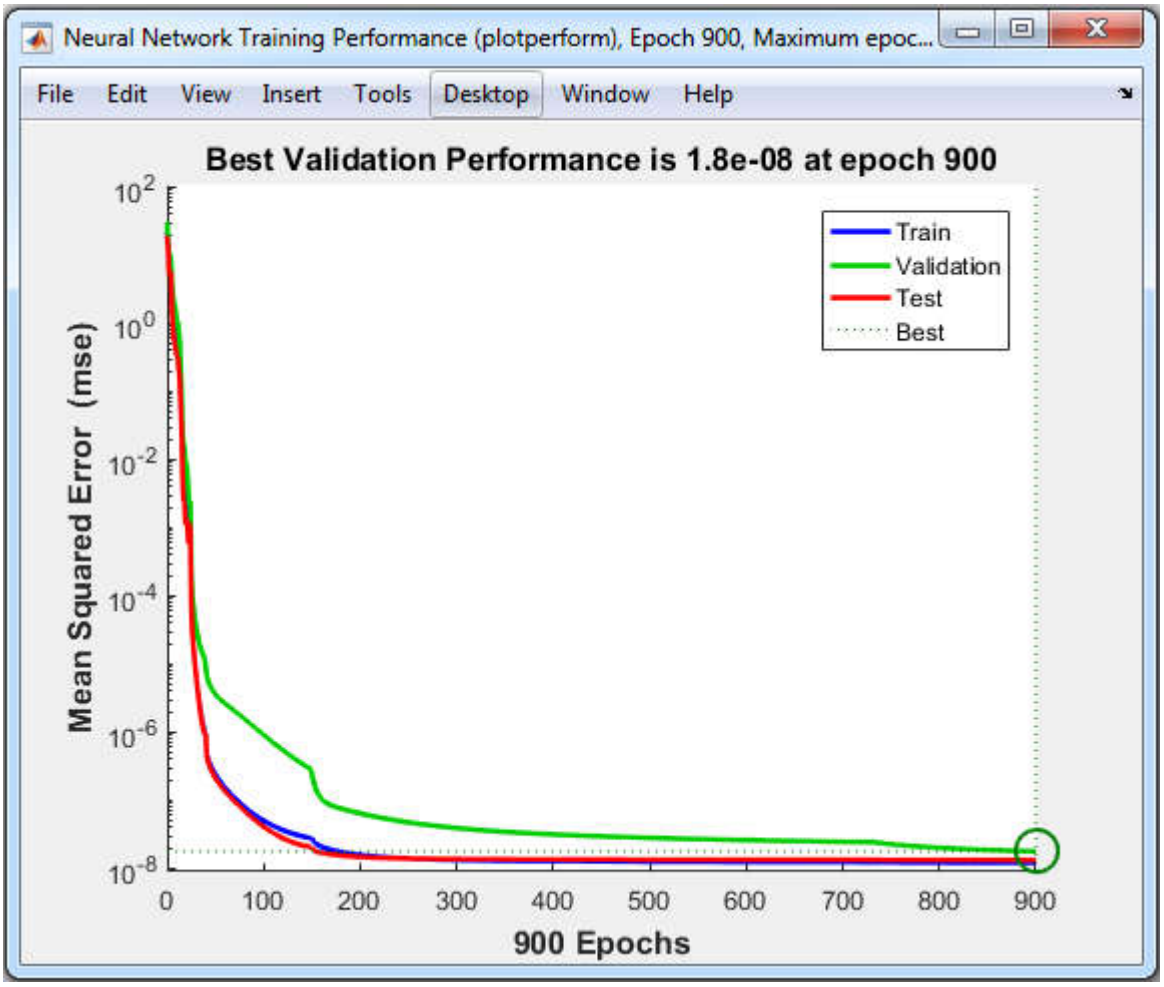


Рис. 4.20. Вікно контролю процесу навчання

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. 4.21**)** 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.

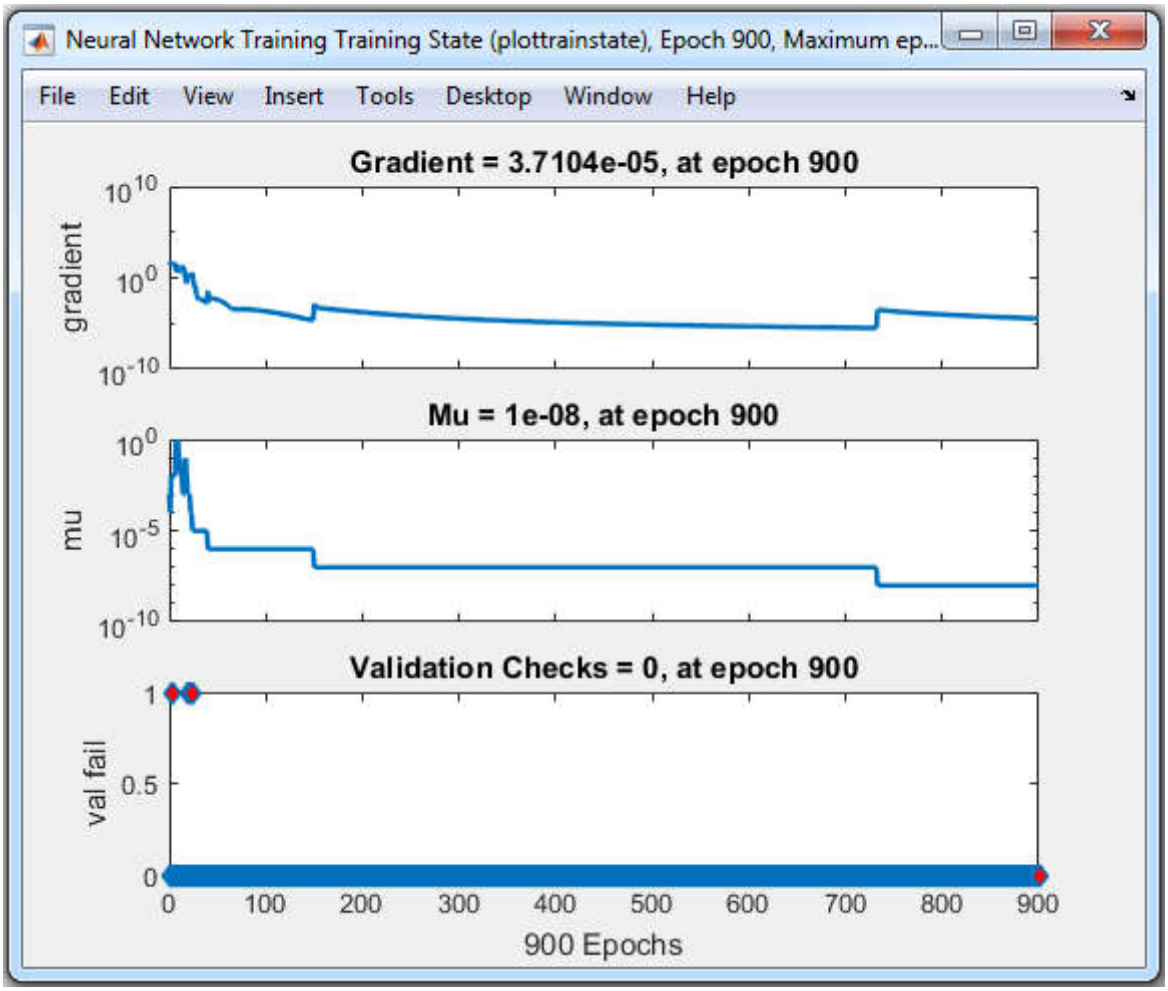


Рис.4.21. Вікно Neural Network Training State

Основна ідея використання градієнта полягає в тому, що шляхом зміни вагових коефіцієнтів мережі в напрямку, протилежному до градієнта, можна найбільше зменшити значення функції втрати. Градієнт допомагає визначити, в якому напрямку і наскільки потрібно вносити зміни в вагові коефіцієнти для покращення продуктивності мережі.

На графіку Gradient відображається, як змінюється величина градієнта під час процесу навчання. Зазвичай, мережа спочатку навчається з великими градієнтами, і з часом ці градієнти зменшуються, коли мережа наближається до оптимальних вагових коефіцієнтів.

Моніторинг графіка Gradient може бути корисним для визначення, чи існують проблеми з градієнтним зникненням або вибуванням (gradient vanishing/exploding), а також для визначення, чи виникають проблеми зі збіжністю навчання. Великі або надто малі градієнти можуть вказувати на необхідність оптимізації параметрів навчання.

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.

Зміна mu показує, як модифікується швидкість навчання під час процесу навчання. Зазвичай, mu може збільшуватися або зменшуватися в залежності від реакції нейронної мережі на дані. Наприклад, велике значення "mu" може вказувати на те, що швидкість навчання збільшилася для прискорення збіжності, але при цьому може виникнути ризик нестабільності. Зменшення mu може вказувати на те, що навчання сталося занадто швидко або навіть зупинилося.

Моніторинг графіка mu може бути корисним для контролю за швидкістю навчання та стабільністю процесу навчання. Великий вміст інформації в графіку mu може вказувати на те, що вибрані параметри навчання потребують подальшої налаштування.

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. Зазвичай, мінімізація цього показника є однією з цілей тренування мережі. Якщо кількість помилок на валідації росте, це може бути ознакою перенавчання, і може бути необхідно прийняти заходи для покращення продуктивності мережі, такі як зупинка навчання раніше або зміна гіперпараметрів.

Продовження опису вікна Neural Network Training (nntraintool)

Rнопка **Regression (регресія)** використовується для вибору та налаштування параметрів навчання нейронної мережі для завдань регресії. Як відомо. завдання регресії - це один з типів завдань в машинному навчанні, де основною метою є передбачення числових значень або кількісних величин на основі вхідних даних. У регресійних завданнях модель намагається знайти функціональну залежність між вхідними ознаками (параметрами) і вихідними числовими значеннями, які зазвичай називаються цільовими або відгуком. .

При натисканні на кнопку Regression відкривається вікно **Neural Network Training Regression** (рис. 4.22), 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.

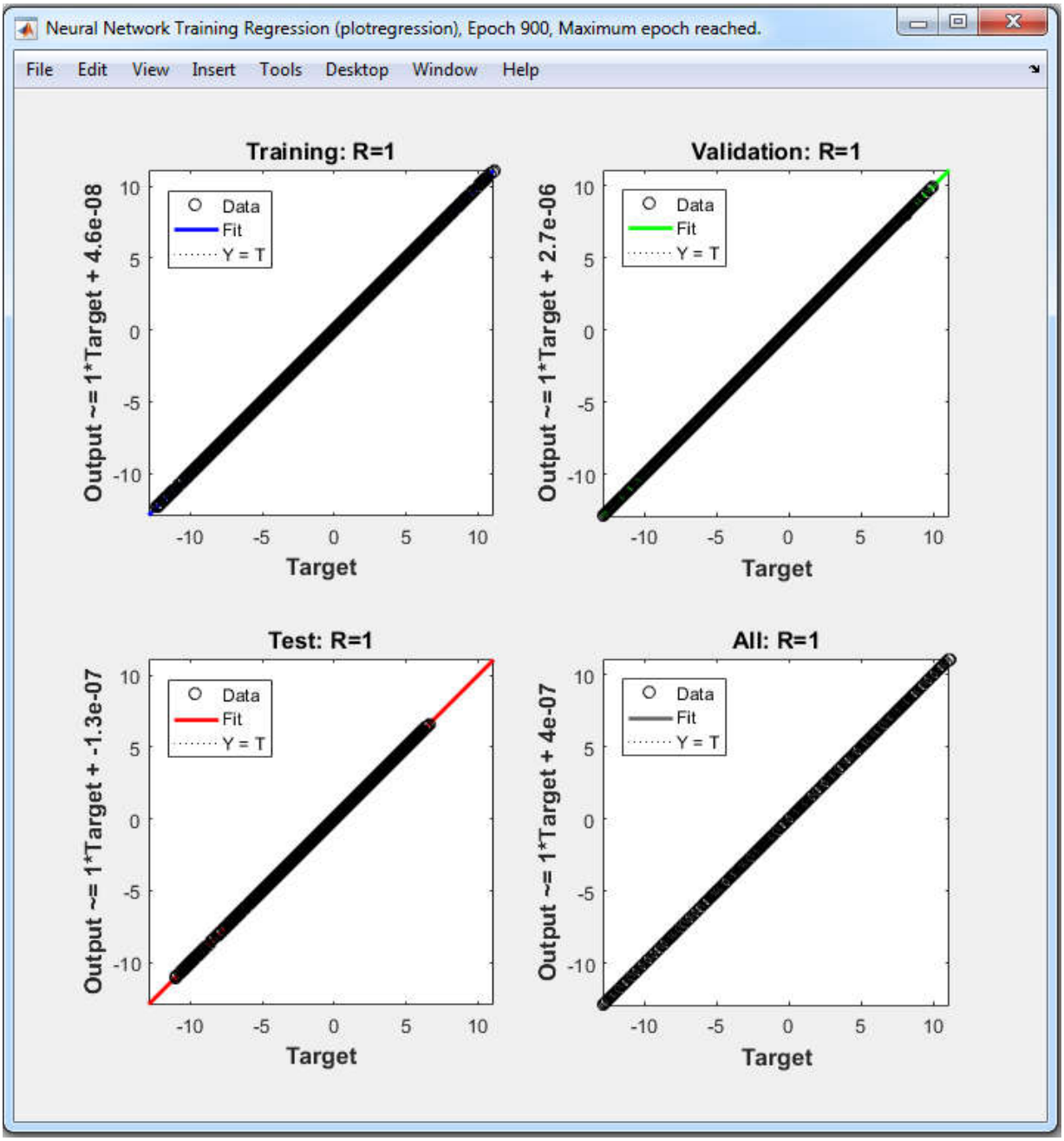


Рис. 4.22. Вікно Neural Network Training Regression

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.

Графіки допомагають візуально оцінити точність моделі на трьох різних наборах даних і визначити, наскільки добре модель може узгоджуватися з фактичними даними. Вони можуть бути корисними для виявлення аномалій, оцінки загальної продуктивності моделі і визначення необхідності подальшої оптимізації.

Графіки **Fit (прогноз)** – це графіки лінійної регресії, які відображають прогнозовані значення, які надає нейронна мережа на основі вхідних даних. Графіки Fit демонструють відповідність прогнозованих значень, отриманих від моделі, фактичним значенням з наборів даних. Вони допомагають візуально оцінити, наскільки точно модель регресії передбачає числові значення на основі вхідних даних. Якщо прогнозована лінія добре відповідає фактичним даним і проходить через багато точок, це свідчить про високу точність моделі. Напротив, якщо лінія регресії відхиляється від фактичних значень, це може вказувати на неадекватність моделі і необхідність налаштування параметрів нейронної мережі.

Вирази для лінійфної регресії, а також значення коефіцієнта кореляції R наведені у вікні Neural Network Training Regression для кожного набору даних: Training, Validation, Test та All. Якщо графік Fit добре узгоджується з графіком Y=T, то це свідчить про високу точність моделі регресії.

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.

Зазвичай, коефіцієнт кореляції R може приймати значення в діапазоні від -1 до 1. Його інтерпретація наступна:

- Коефіцієнт кореляції R дорівнює 1: Це означає, що прогнозовані значення і фактичні значення абсолютно лінійно залежні одне від одного, і модель регресії ідеально передбачає фактичні значення.
- Коефіцієнт кореляції R дорівнює 0: Це вказує на жодну лінійну залежність між прогнозованими і фактичними значеннями, і модель не може передбачити дані з жодною точністю.
- Коефіцієнт кореляції R дорівнює -1: Це означає, що прогнозовані значення і фактичні значення є лінійно залежними, але зворотним способом, і модель регресії передбачає їх ідеально, але з зміненим напрямком.

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.

Зазвичай вибір Plot Interval допомагає контролювати кількість інформації, яка відображається під час навчання, особливо якщо навчання триває багато епох і графіки можуть бути дуже завантажені. Встановлення меншого значення Plot Interval означатиме, що графіки оновлюватимуться частіше, що може бути корисним для більш детального аналізу, але збільшить обсяг відображеної інформації. Навпаки, встановлення більшого значення Plot Interval зменшить частоту оновлень графіків і зменшить обсяг відображеної інформації.

Зазвичай, налаштування Plot Interval є важливою опцією під час навчання моделі для того, щоб зручно аналізувати процес навчання та вплив варіацій параметрів моделі на результати.

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.4.23.

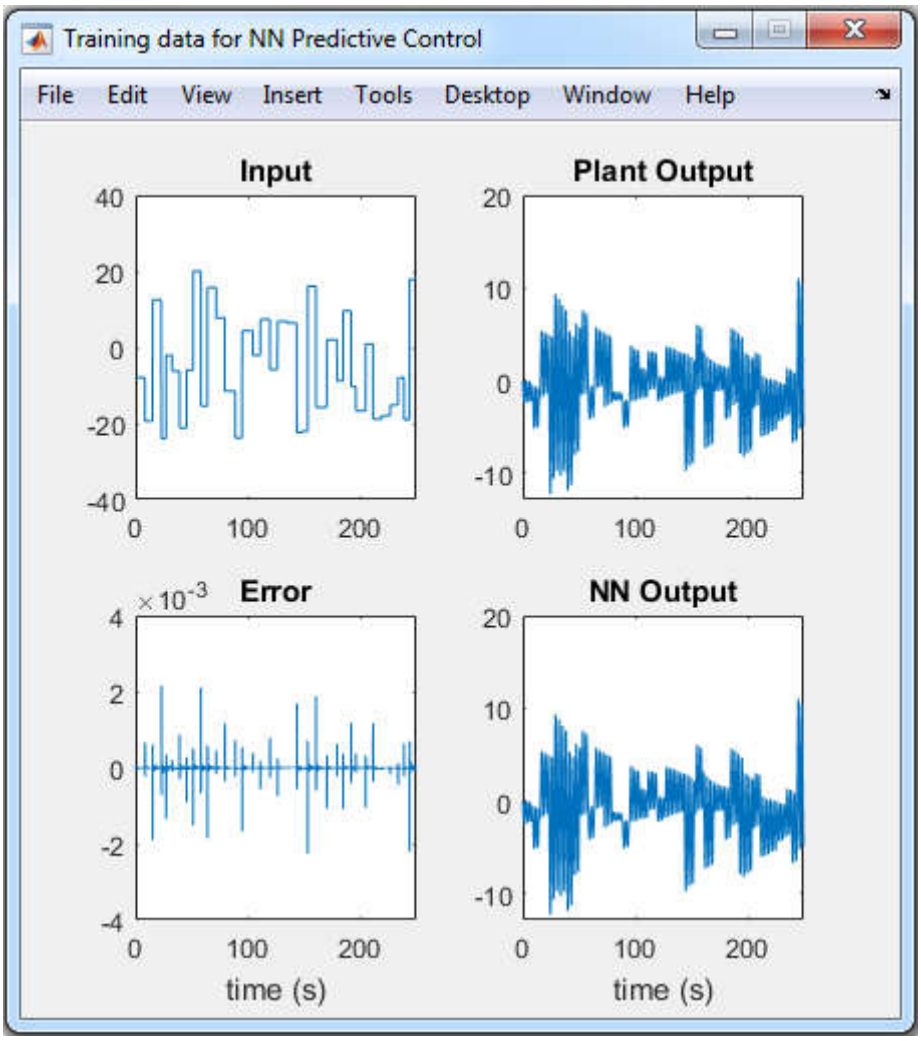


Рис.4.23. Результати тренування мережі

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.4.24 і рис.4.25 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.

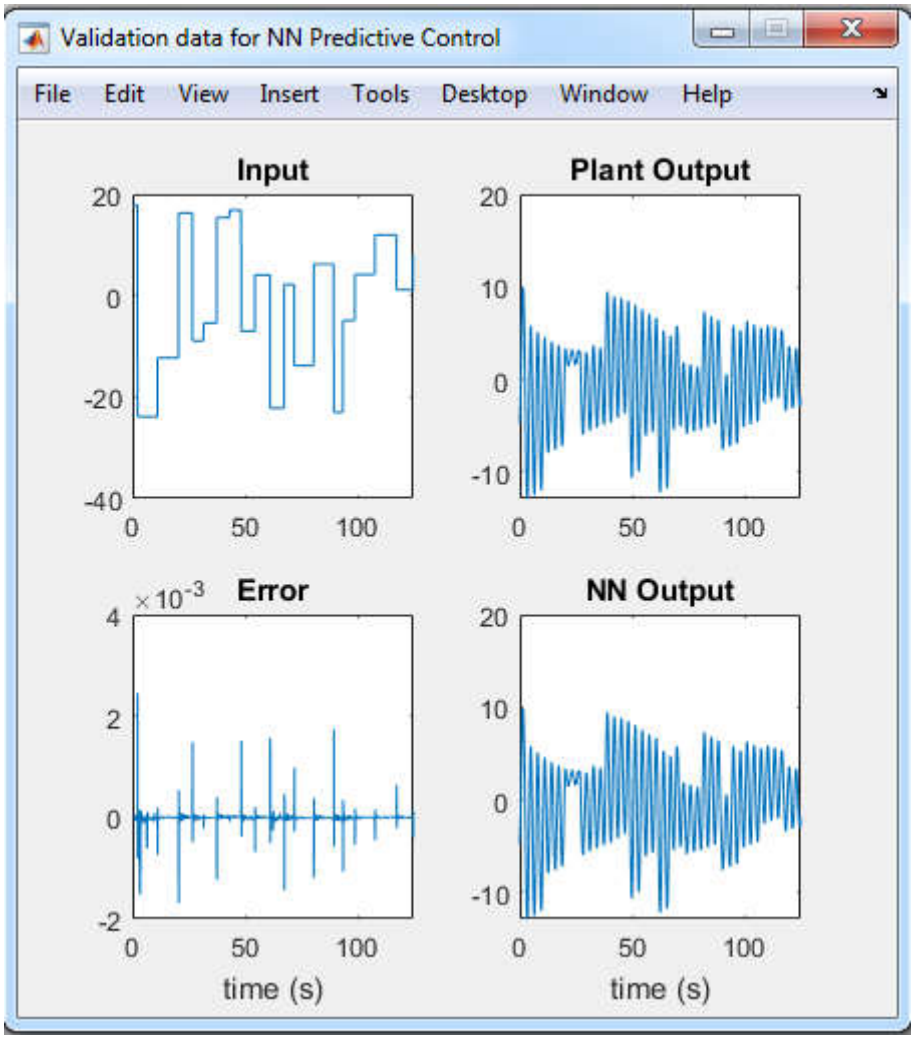


Рис.4.24. Результати перевірки мережі на контрольній множині

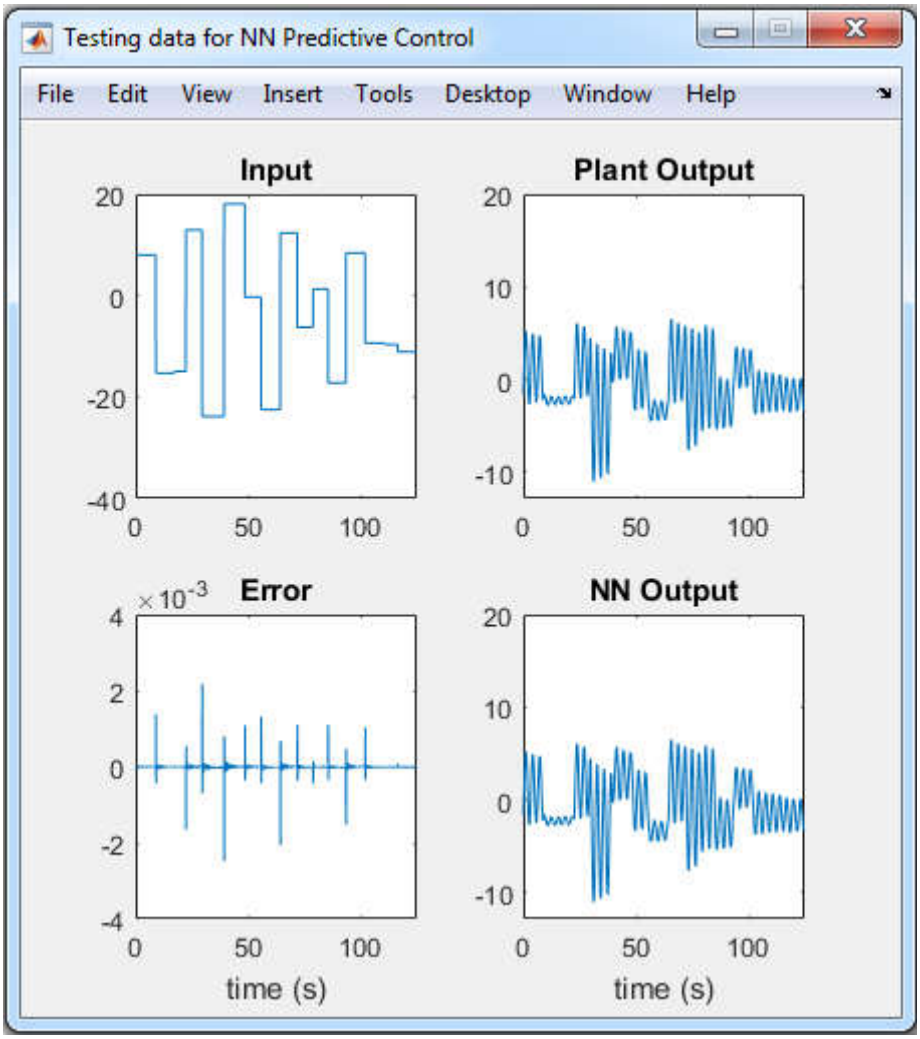


Рис.4.25.Результати перевірки мережі на тестовій множині

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.4.26.

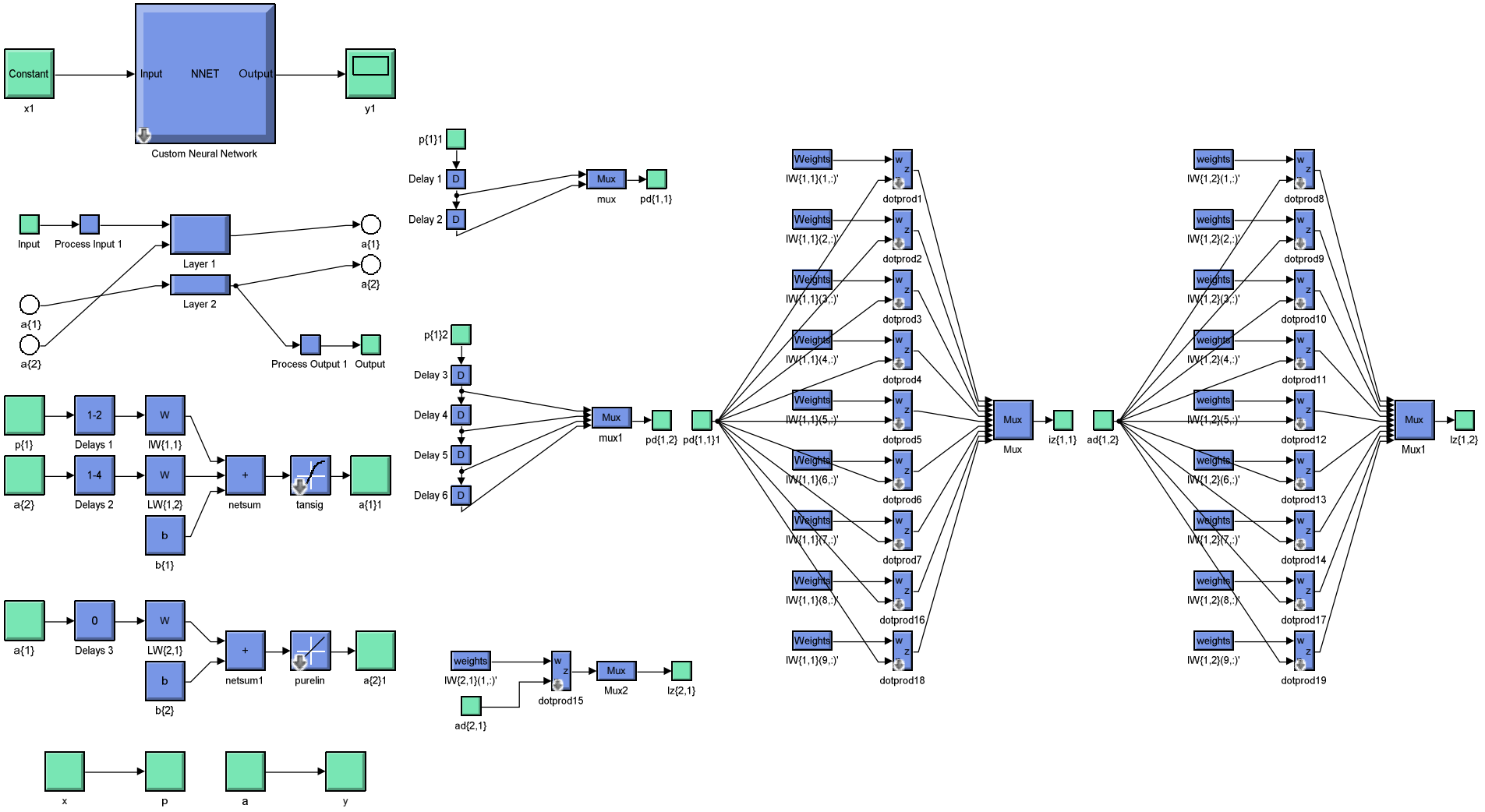


Рис.4.26. Моделі елементів динамічної мережі з елементами затримок, реалізовані в Simulink

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.4.27.

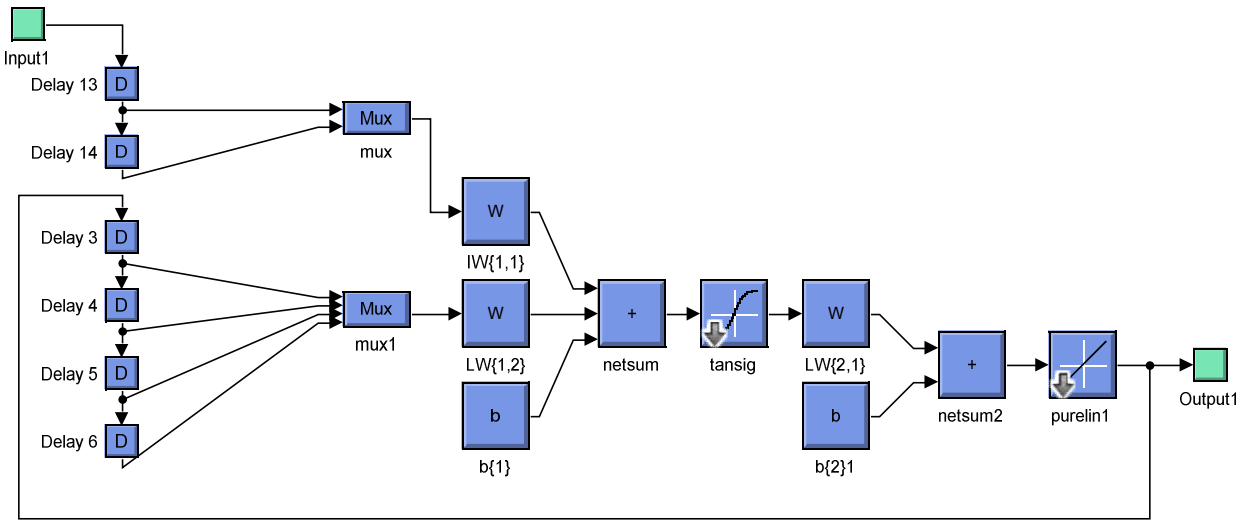


Рис.4.27. Модель динамічної мережі з елементами затримок, побудованої в Simulink

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.4.28.

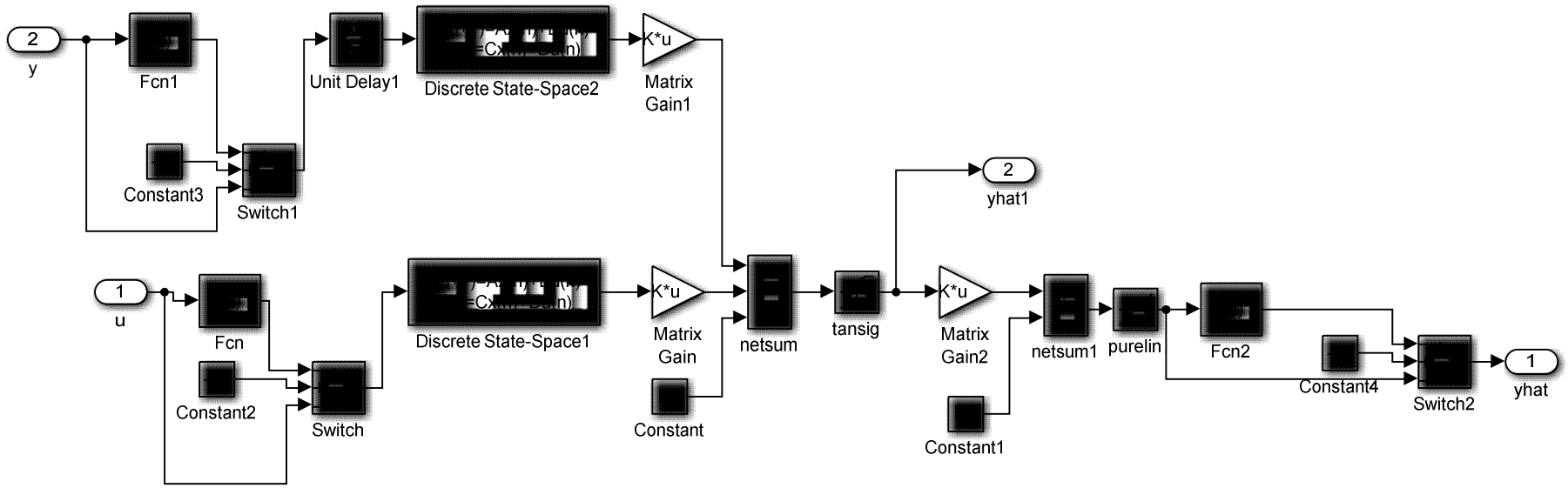


Рис. 4.28. Структурна схема нейромережевої моделі об'єкту регулювання

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

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

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

Для випадку $N_i = 2$ і $N_j = 5$ чисельні значення матриць $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ і $\mathbf{D}$ вказаних блоків наступні

Discrete State Space 1

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

Discrete State Space 2

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

У системі Simulink формується також схема **ptest3sim2**, показана на рис.4.29. Після активізації блоку Subsystem відкривається вікно з схемою рис.4.30. 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.4.11 і рис.4.28 показана на рис 4.31.

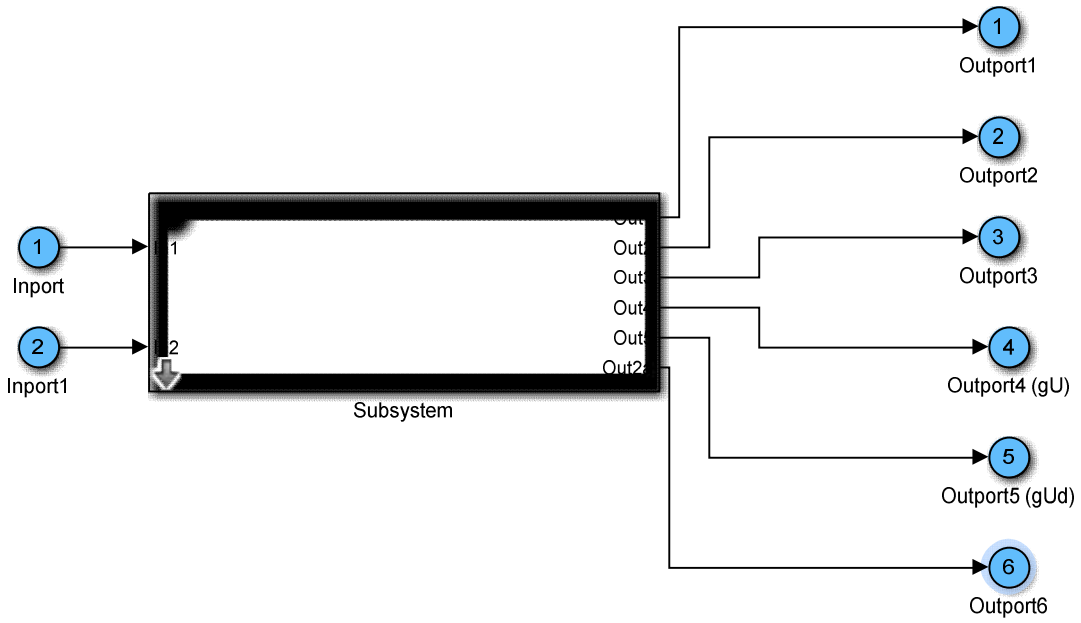


Рис. 4.29.Схема ptest3sim2, використовувана M-функцією predopt
для прогнозу процесу в майбутньому

Після створення нейромережевої моделі керованого об'єкта здійснюється повернення до вікна **Neural Network Predictive Controller** (рис. 4.12), 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. Вибір значення Const Horizon (N2) важливий, оскільки він впливає на те, наскільки точно і передбачувано система буде керована за допомогою алгоритму NNPC.

Зазвичай, збільшення значення Const Horizon (N2) дозволяє отримати більший горизонт прогнозу, що означає більш далекі випереджальні прогнози, але водночас може збільшити обчислювальну складність і затримки у системі керування. Обране значення "Const Horizon" повинно бути ретельно підібрано з урахуванням конкретних вимог і характеристик вашого керувального завдання і системи.

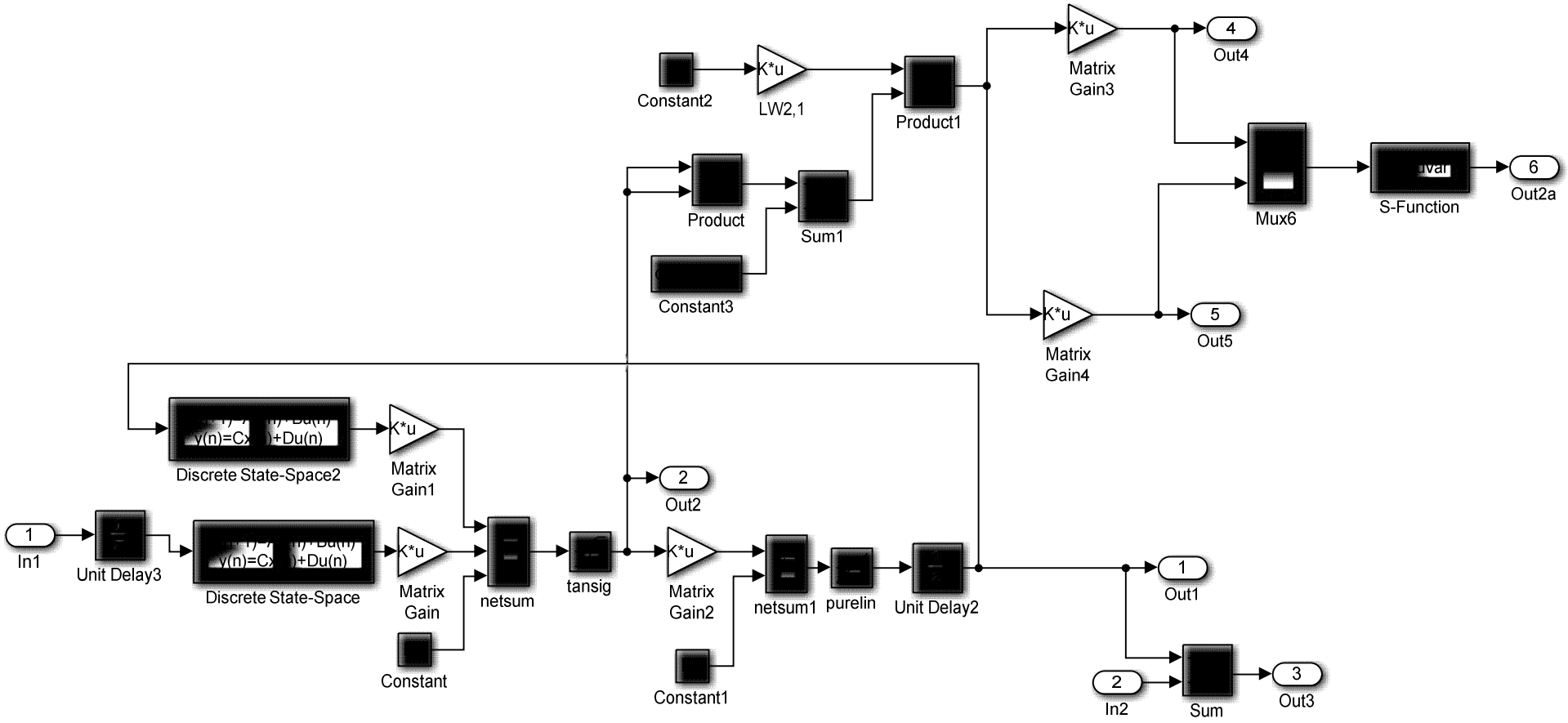


Рис. 30. Схема блоку Subsystem системи ptest3sim2

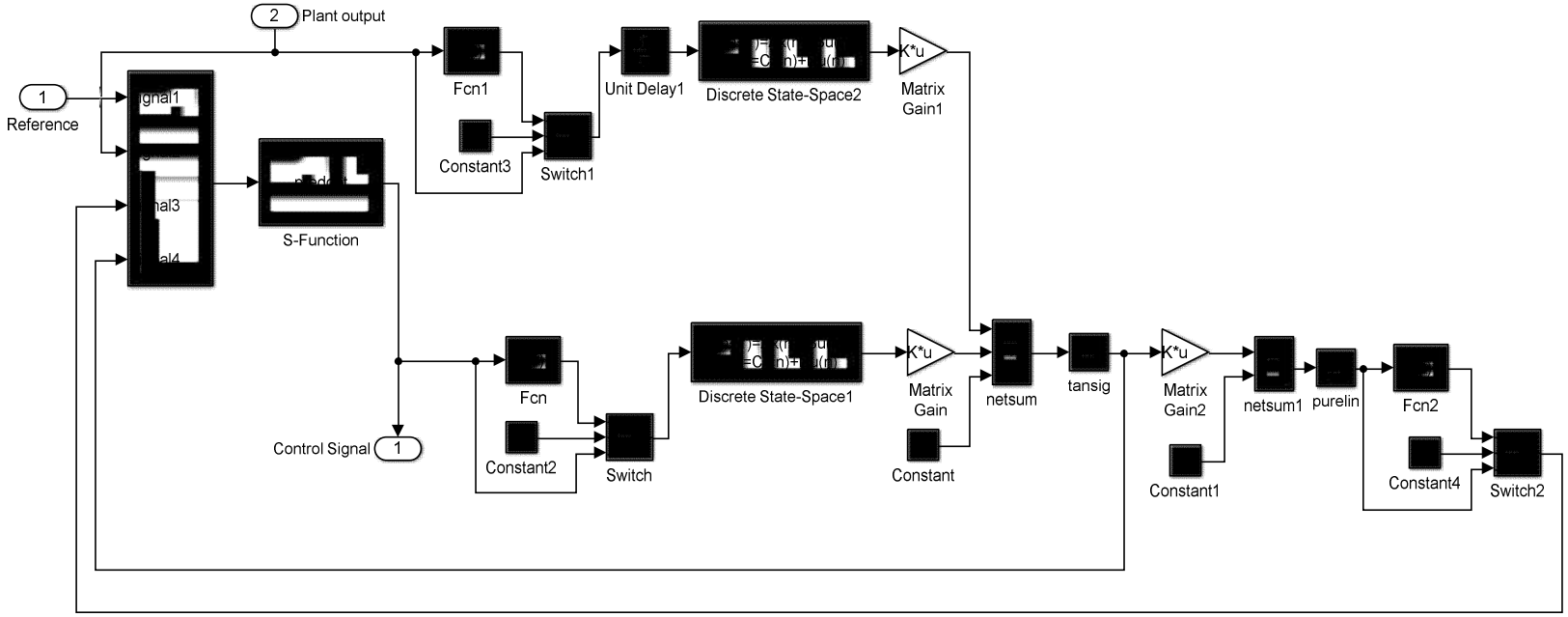


Рис.4.31. Перетворена структурна схема нейрорегулятора

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.

Основна ідея цього параметру полягає в тому, що велике значення Control Horizon (Nu) дозволяє оптимізаційному алгоритму керування ураховувати більшу кількість кроків у майбутньому при плануванні керування. Це може бути корисним для досягнення бажаного стану системи та забезпечення стабільності, а також враховувати обмеження на керування.

Однак велика кількість кроків в Control Horizon (Nu) також може призвести до складніших обчислювальних задач і збільшити вимоги до ресурсів, тому вибір значення параметра Control Horizon (Nu) повинен бути обґрунтованим і базуватися на конкретних вимогах та характеристиках системи керування.

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

Контролер має оптимізувати вихідні сигнали системи, забезпечуючи бажаний режим роботи. Однак інколи деякі сигнали керування можуть бути дорогою або необхідною величиною. Наприклад, в деяких ситуаціях, зниження витрат на керування може бути важливим завданням.

Control Weighting Factor визначає, як обчислювати вартість або функцію витрат управління під час оптимізації. Якщо цей фактор великий, то контролер надає велику вагу мінімізації витрат на керування, і, отже, старається робити керуючі сигнали якнайбільш економічними. Зменшення фактора зробить керуючі сигнали менш важливими у функції витрат, і контролер буде акцентуватися на досягненні бажаного режиму роботи системи.

Користувачі можуть регулювати Control Weighting Factor в залежності від своїх конкретних вимог та цілей управління. Цей параметр дозволяє балансувати між досягненням бажаного результату і вартістю управління в контексті нейронного мережевого прогнозного управління.

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.

Одновимірний пошук (або одновимірна оптимізація) - це метод оптимізації, де здійснюється пошук оптимального значення одного параметра, тримаючи інші параметри незмінними. В контексті контролера з використанням нейронної мережі це може бути важливим для налаштування певних параметрів контрольної стратегії, які впливають на якість та ефективність керування.

Поріг зменшення показника якості вказує на те, наскільки мінімальне зменшення величини, яку слід оптимізувати, вважати прийнятним для завершення процесу оптимізації. Якщо зменшення показника якості стає меншим за встановлений

поріг, то оптимізація завершується, і визначені параметри вважаються оптимальними.

Цей параметр дозволяє контролеру зупинити оптимізацію, коли досягнуто задовільного рівня покращення показника якості, що може бути корисним для економлення обчислювальних ресурсів та забезпечення швидкої збіжності оптимізації.

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. Вибір конкретного "Minimization Routine" залежить від характеристик завдання керування, обмежень, доступності обчислювальних ресурсів та інших факторів.

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.

Важливо збалансувати кількість ітерацій Iterations Per Sample Time так, щоб контролер зберігав ефективність та швидкість реакції на зміни у системі керування. Занадто багато ітерацій може призвести до затримок в реакції контролера, тоді як занадто мало ітерацій може призвести до недостатньої точності оптимізації.

Загальною практикою є експериментальне налаштування цього параметра в залежності від конкретних вимог та обмежень системи керування для досягнення оптимального балансу між точністю та швидкістю оптимізації.

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

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

Як навчальну функцію використано **trainlm**.

4.5 Розрахунок динамічних характеристик систем з нейрорегулятором NN Predictive Controller

Для побудови графіків перехідних процесів трьохмасової системи з нейрорегулятором NN Predictive Controller використовуємо моделі Simulink, приведені на рис. 4.10 або 4.32.

На рис. 4.33 і рис. 4.34 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 трьохмасовою електромеханічною системою, що розглядається у наведеному прикладі

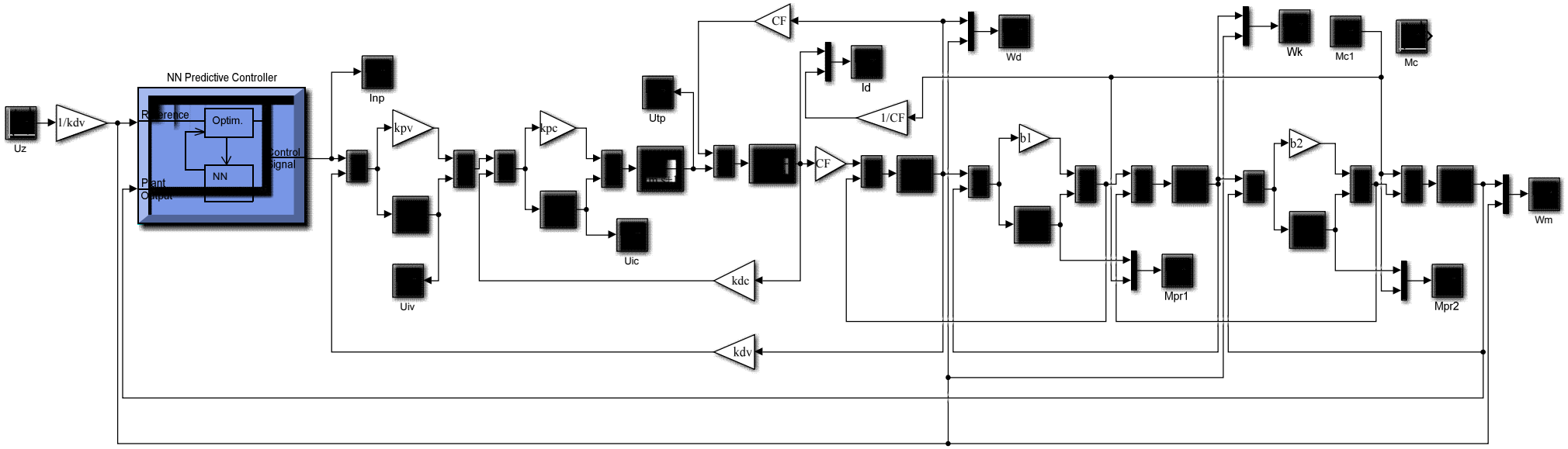
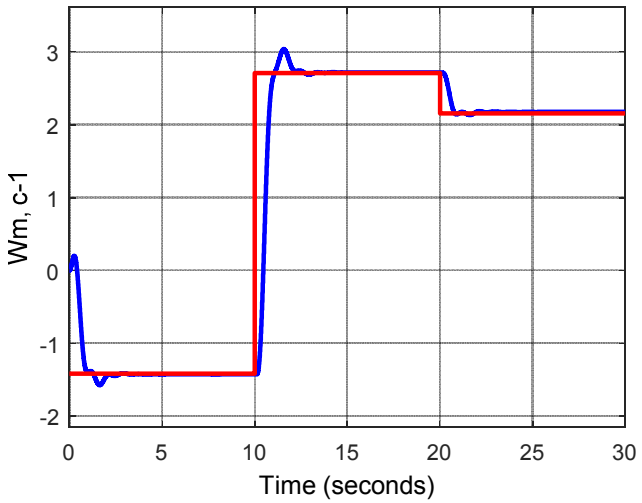
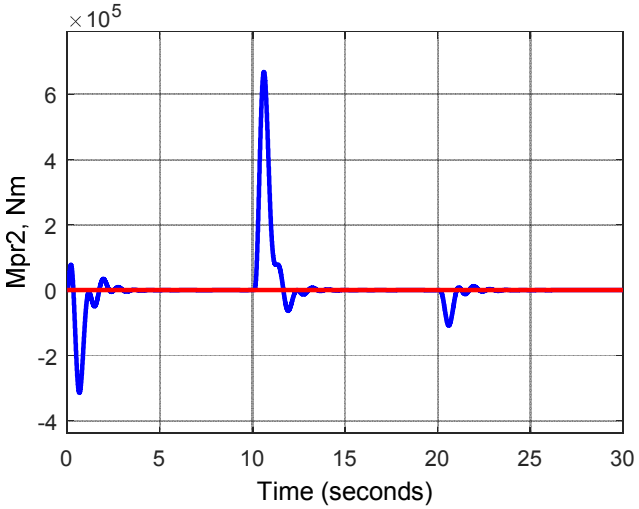


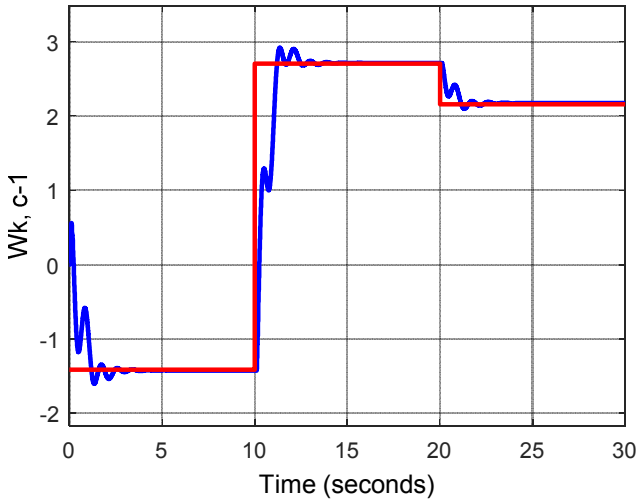
Рис. 4.32. Схема моделі трьохмасової системи з нейрорегулятором



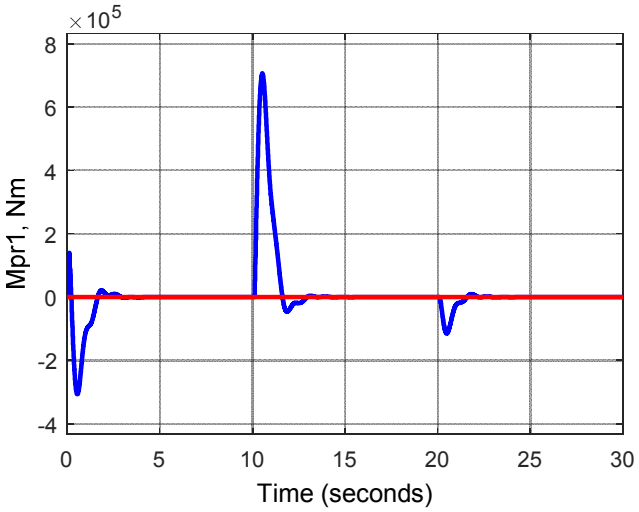
швидкість механізму по задаючій дії



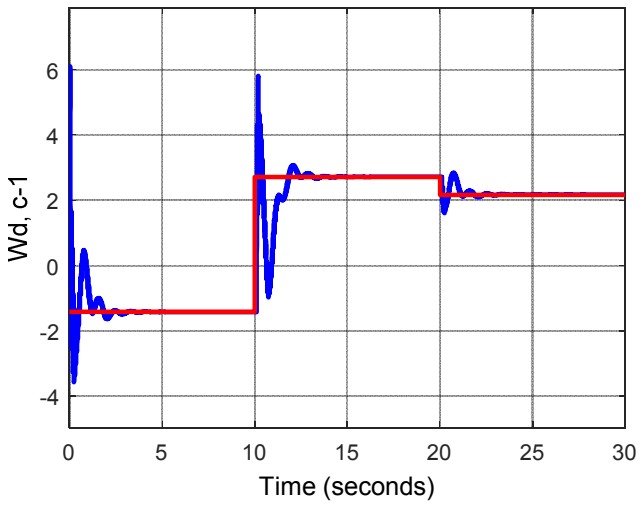
другий момент пружності по задаючій дії



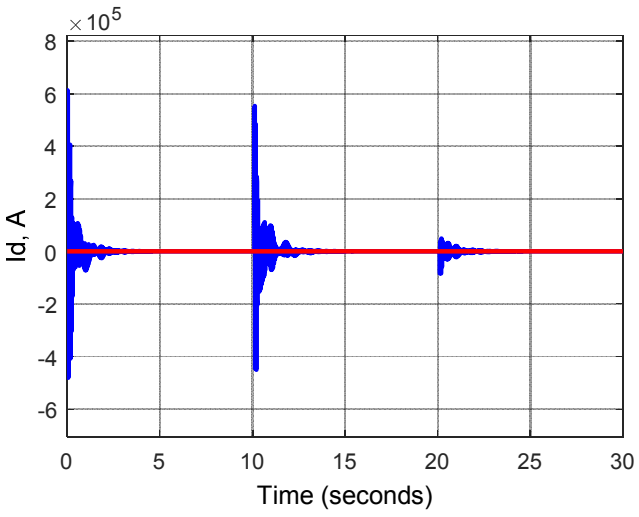
швидкість канату по задаючій дії



перший момент пружності по задаючій дії

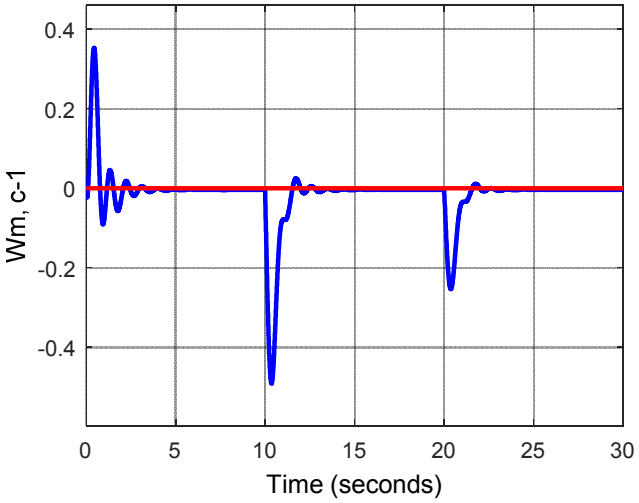


швидкість двигуна по задаючій дії

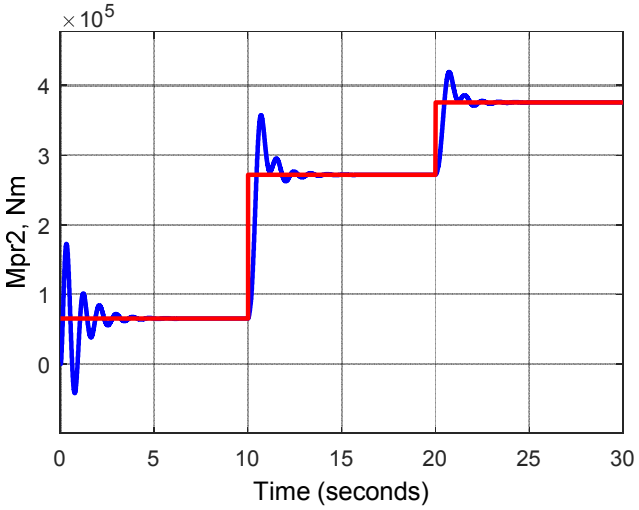


струм двигуна по задаючій дії

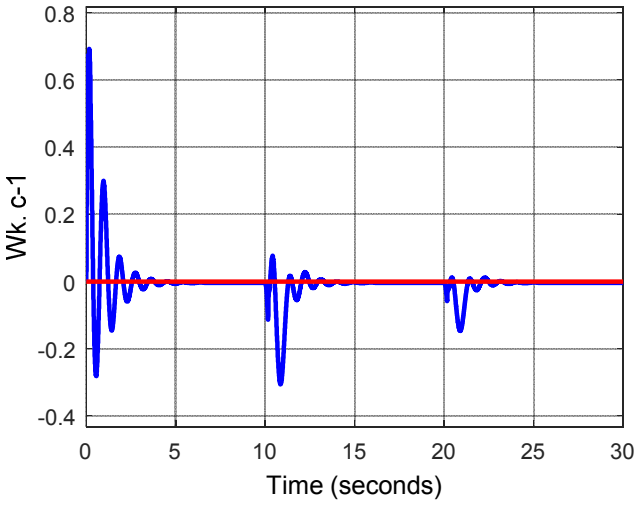
Рис.4.33. Графіки перехідних процесів трьохмасової системи з нейрорегулятором NN Predictive Controller по задаючій дії



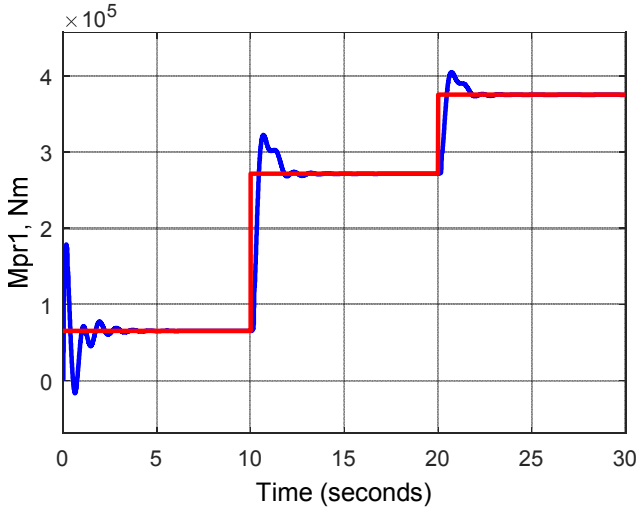
швидкість механізму по збурюючій дії



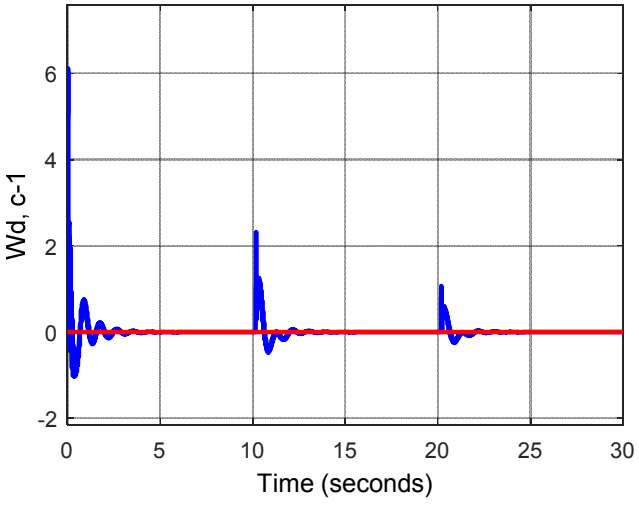
другий момент пружності по збурюючій дії



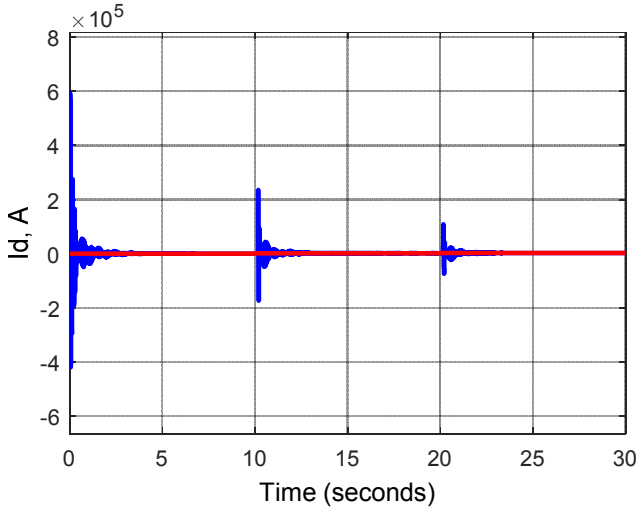
швидкість канату по збурюючій дії



перший момент пружності по збурюючій дії



швидкість двигуна по збурюючій дії



струм двигуна по збурюючій дії

Рис.4.34. Графіки перехідних процесів трьохмасової системи системи з нейрорегулятором NN Predictive Controller по збурюючій дії

4.6 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

$$y(k+d) = N[y(k), y(k-1),..., y(k-n+1),\ u(k),\ u(k-1),...,u(k-m+1)]\ , \qquad (4.4)$$

де $y(k)$ – вихід моделі;

d – число тактів прогнозу;

$u(k)$ – вхід моделі.

На етапі ідентифікації будується нейронна мережа для NARMA-моделі вигляду (4.4).

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 розповсюдження 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 $\{y(k), y(k-1),..., y(k-n+1)$:

$$u(k)=\frac{y_r(k+d)-f[y(k),y(k-1),...,y(k-n+1),\ u(k-1),...,u(k-m+1)]}{g[y(k),y(k-1),...,y(k-n+1),\ u(k-1),...,u(k-m+1)]}. \quad (4.5)$$

Безпосереднє застосування цього співвідношення для реалізації регулятора утруднене, оскільки управління залежить від поточного значення виходу, тому рівняння (4.5) модифікується таким чином:

$$u(k+1)=\frac{y_r(k+d)-f[y(k),y(k-1),...,y(k-n+1),\ u(k-1),...,u(k-m+1)]}{g[y(k),y(k-1),...,y(k-n+1),\ u(k-1),...,u(k-m+1)]}, \quad (4.6)$$

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

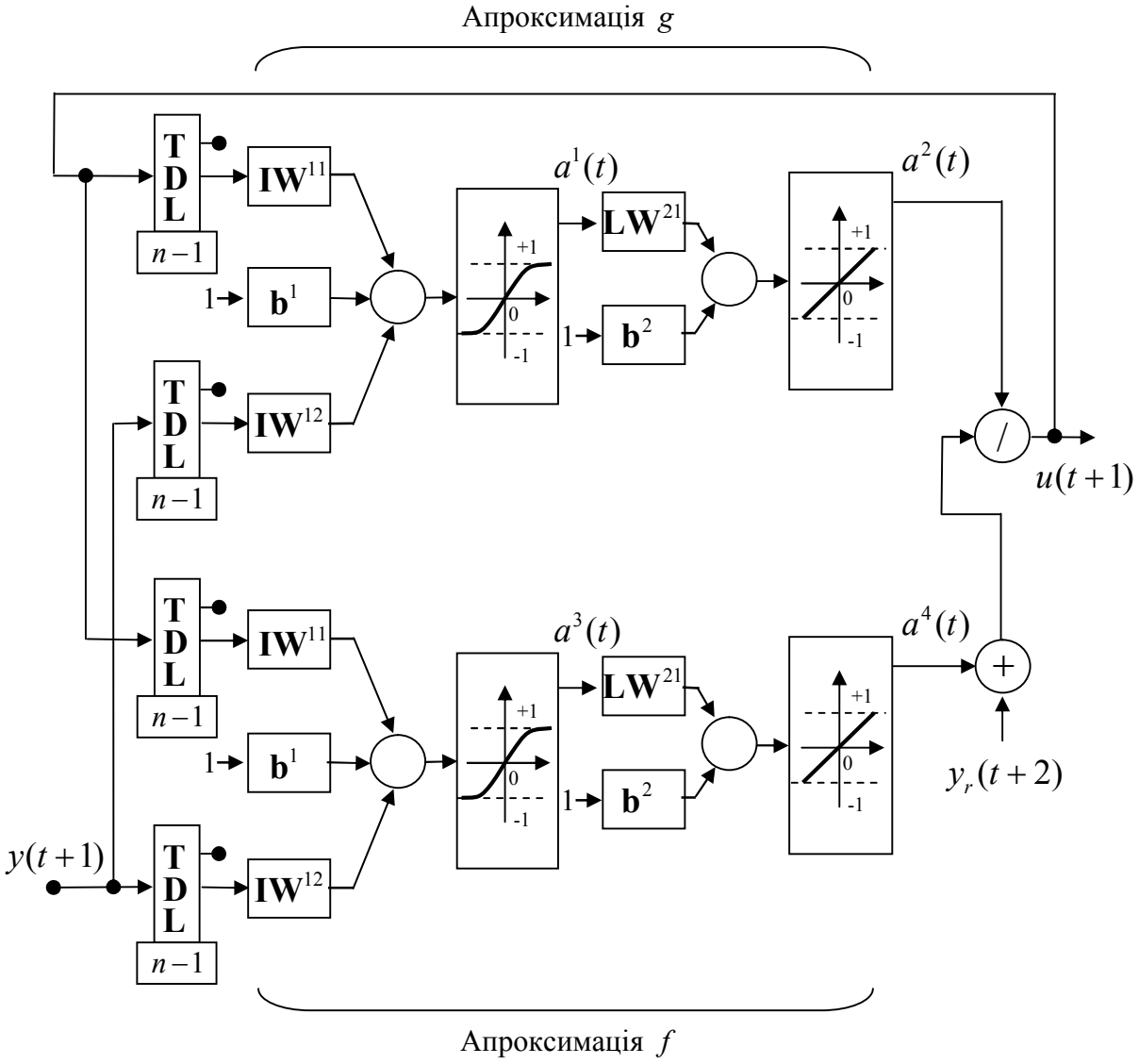


Рис.4.35. Структура NARMA-L2 регулятора у вигляді нейронної мережі

На рис.4.35 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 (4.6).

Загальна структурна схема системи з регулятором NARMA-L2 показана на рис.4.36. На схемі явним чином виділена еталонна модель, яка задає бажану траєкторію для виходу керованого об'єкту.

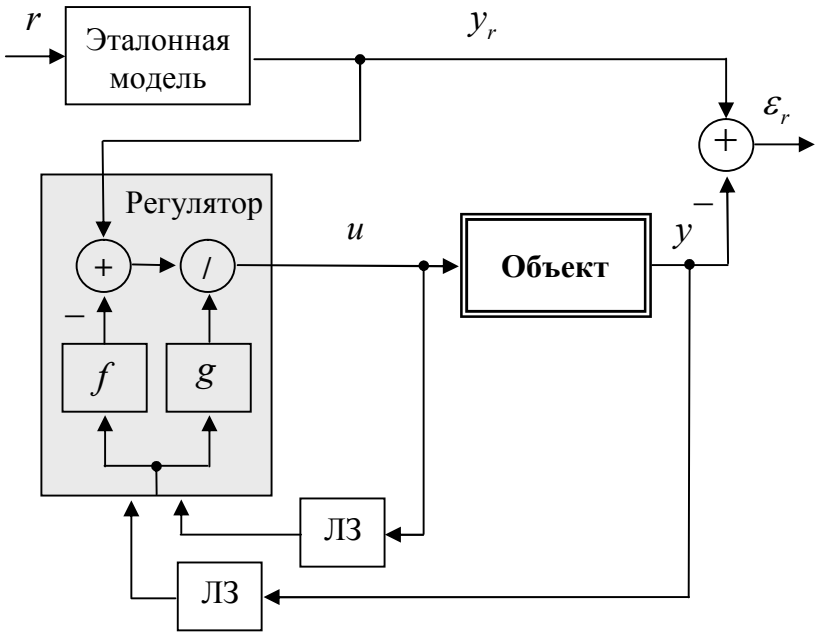


Рис.4.36. Структурна схема системи з регулятором NARMA-L2

4.7 Синтез нейрорегулятора NARMA-L2 CONTROLLER з використанням системи MATLAB

4.7.1 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.4.36. Ця структура аналогічна схемі рис. 4.9, проте замість блоку нейрорегулятора NN Prediction Controller використовується блок NARMA-L2 Controller. Схема блоку Subsystem відповідає рис. 4.13.

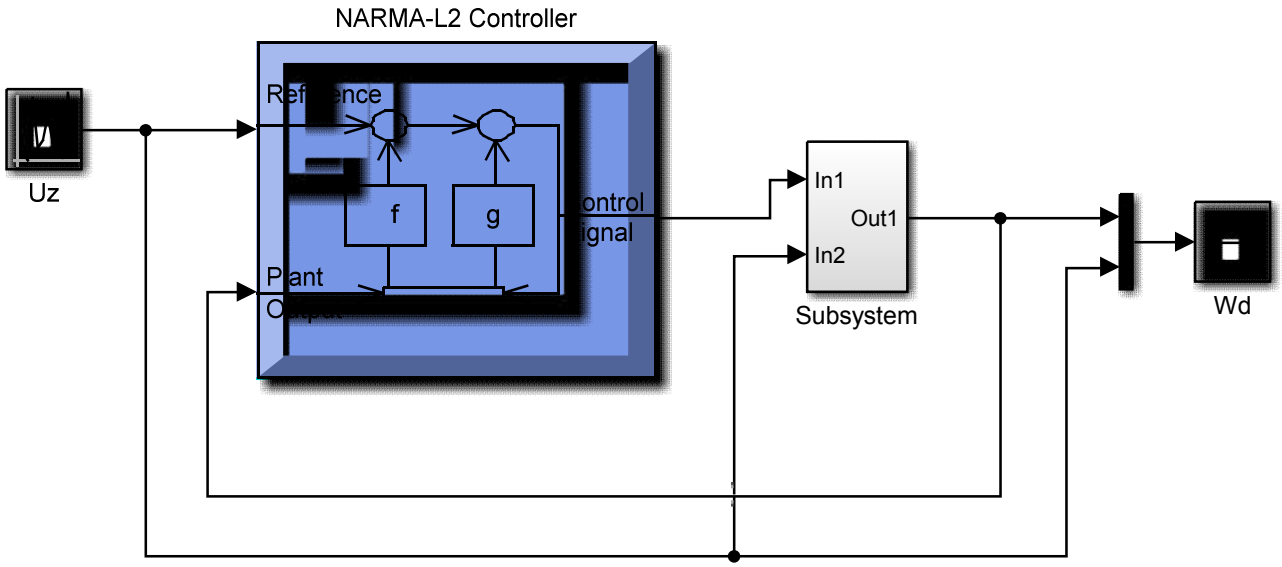


Рис.4.36. Схема системи управління з нейрорегулятором NARMA-L2 Controller

Процес синтезу нейрорегулятора починається шляхом активізації блоку **NARMA-L2 Controller**. З'являється вікно **Plant Identification NARMA-L2**, показане на рис.4.37.

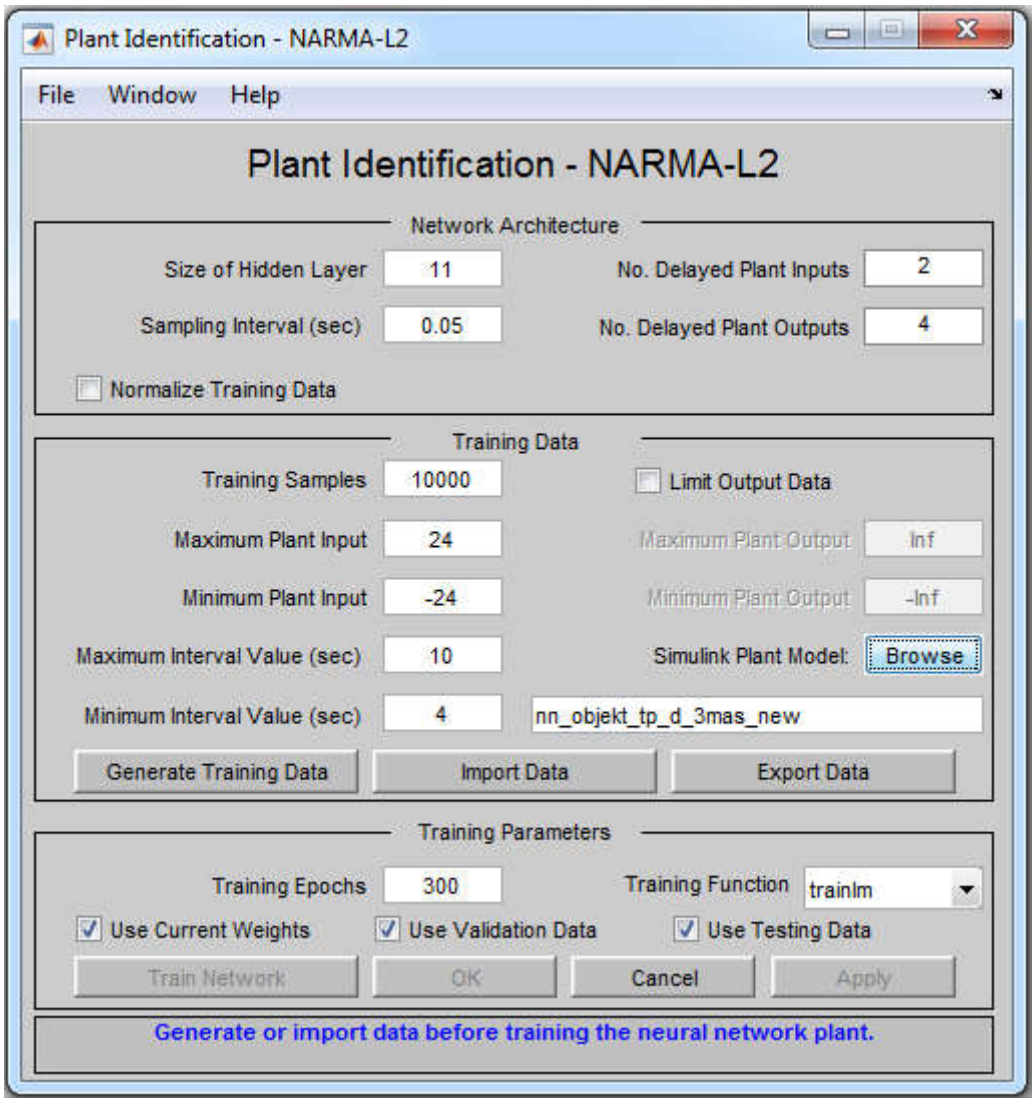


Рис.4.37. Вікно ідентифікації керованого об'єкту Plant Identification NARMA-L2

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 описані в розділі 4.3.

Вибір процедури **Generate Training Data** призведе до запуску програми для створення навчальної послідовності. Графіки входового та вихідного сигналів об'єкту управління відображаються на екрані (див. рис. 4.38). Після завершення процесу створення навчальної послідовності користувач має можливість прийняти (**Accept Data**) або відхилити (**Reject Data**) згенеровані дані.

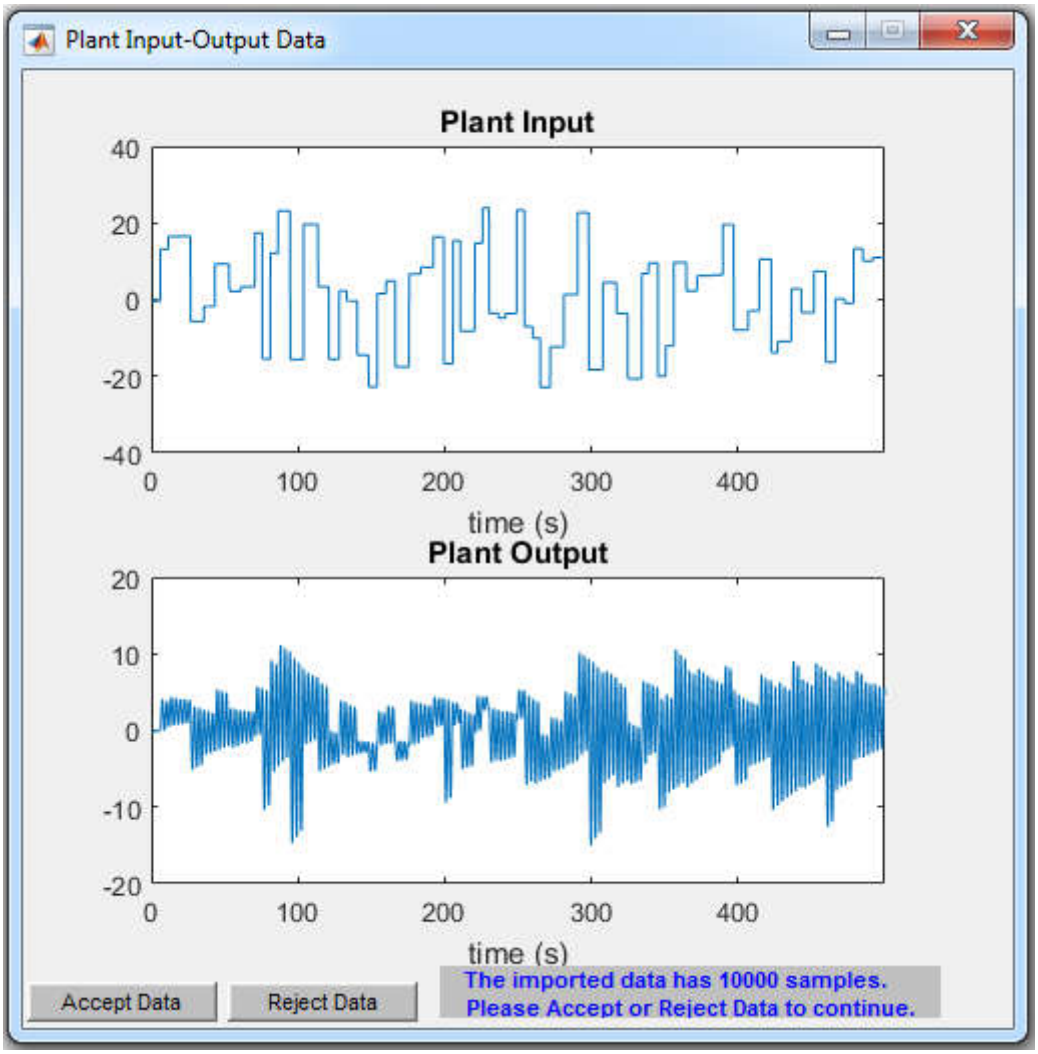


Рис. 4.38. Графіки вхідного і вихідного сигналів при генерації навчальної послідовності

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.4.39 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.4.40.

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.4.41 Схема перетвореної мережі, побудована з використанням елементів, приведених на рис.4.41, показана на рис.4.42.

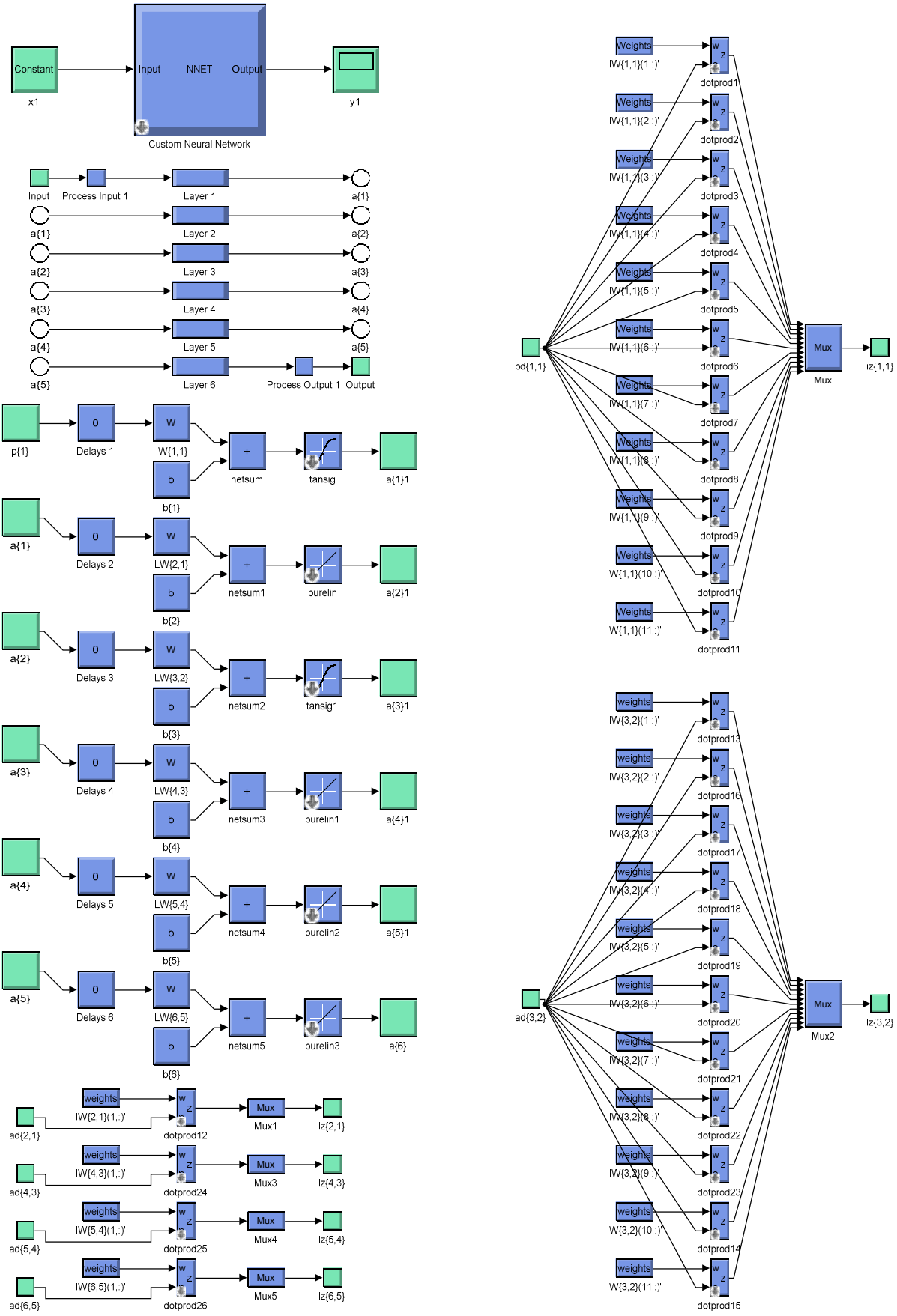


Рис.4.39. Моделі елементів мережі **netn** з прямою передачею сигналу нейрорегулятора NARMA-L2 Controller

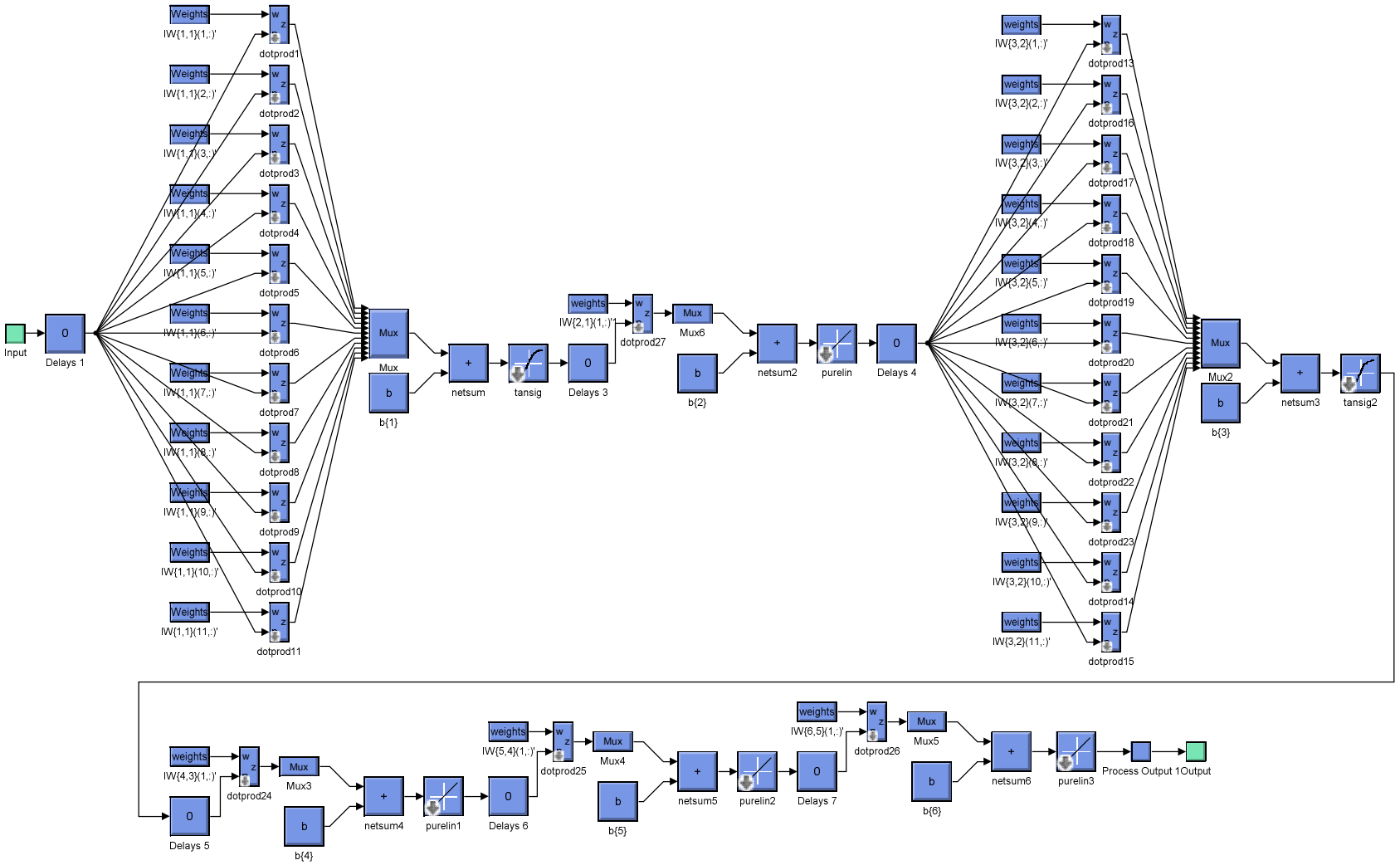


Рис.4.40. Модель статичної мережі **netn** з прямою передачею сигналу нейрорегулятора NARMA-L2

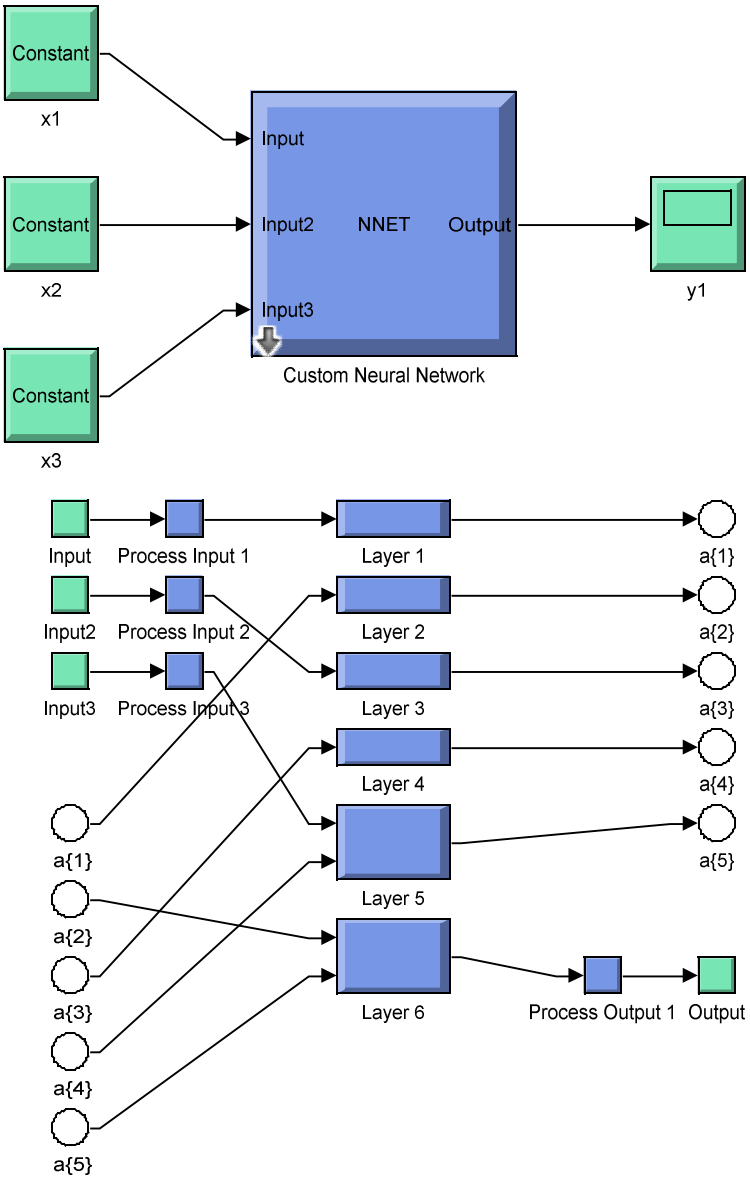


Рис.4.41. Моделі елементів перетвореної мережі **netn** регулятора NARMA-L2 Controller

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.

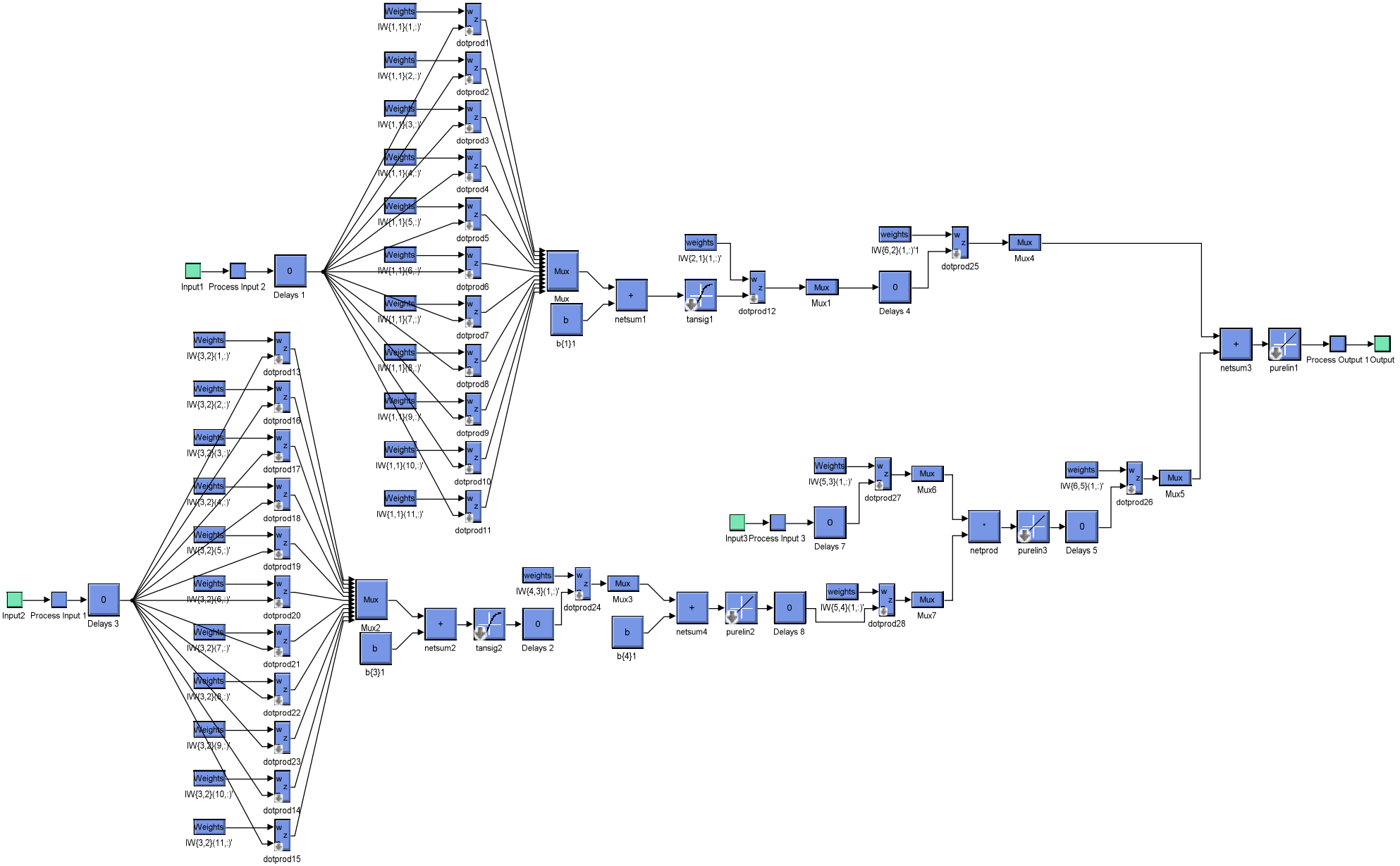


Рис.4.42. Модель перетвореної мережі **netn** регулятора NARMA-L2 Controller з трьома векторами входу

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..4.43.

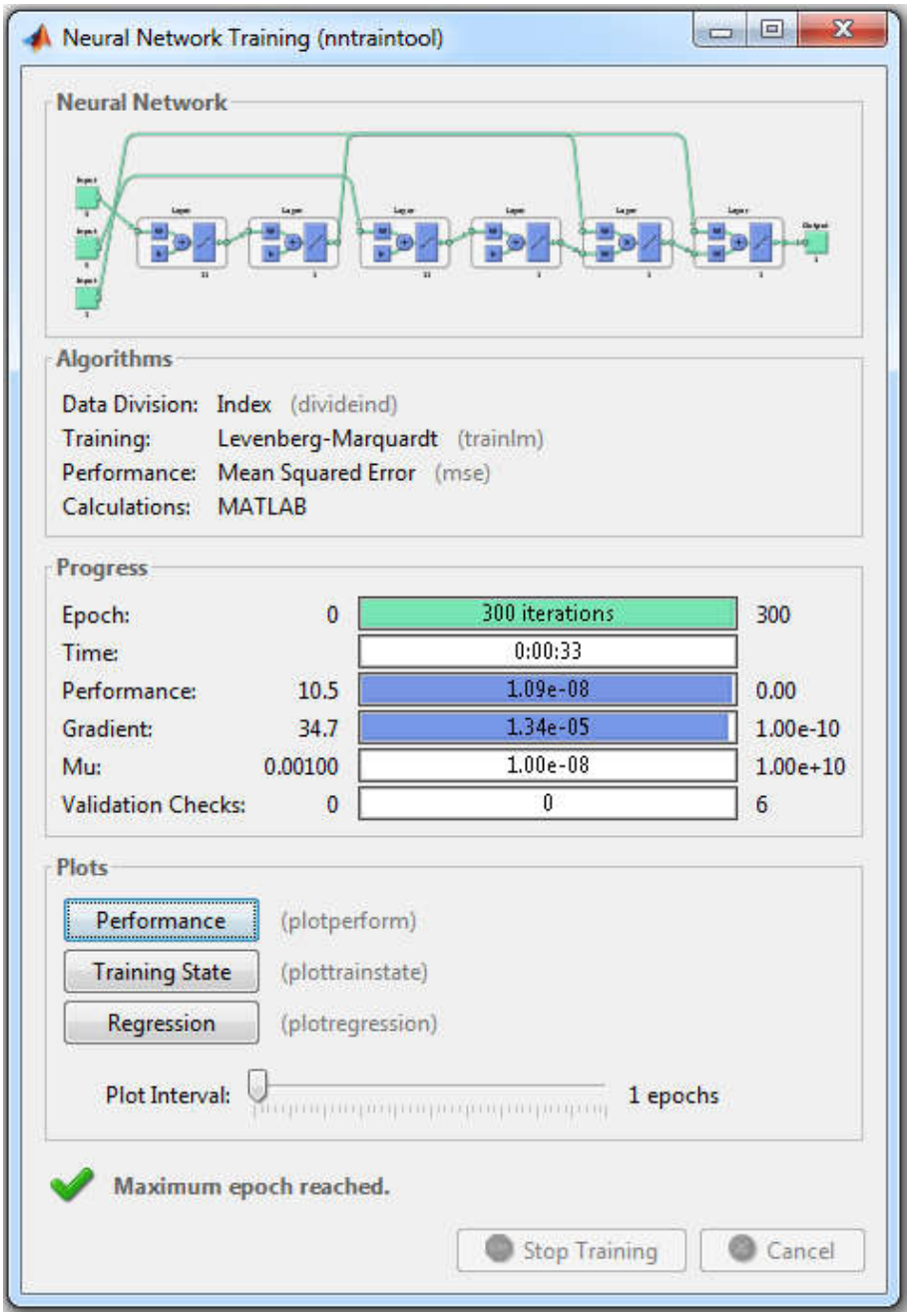


Рис. 4.43. Ввікно Neural Network Training , що відображає хід навчання та результати навчання нейронної мережі

Динаміка зміни помилки навчання, а також перевірки на контрольній і текстовій множинах відбивається у вікні,зображеному на рис.4.44.

Результати навчання, а також результати тестування на контрольній і тестовій множині відображаються на графіках, представлених на рис.4.45 – рис.4.47.

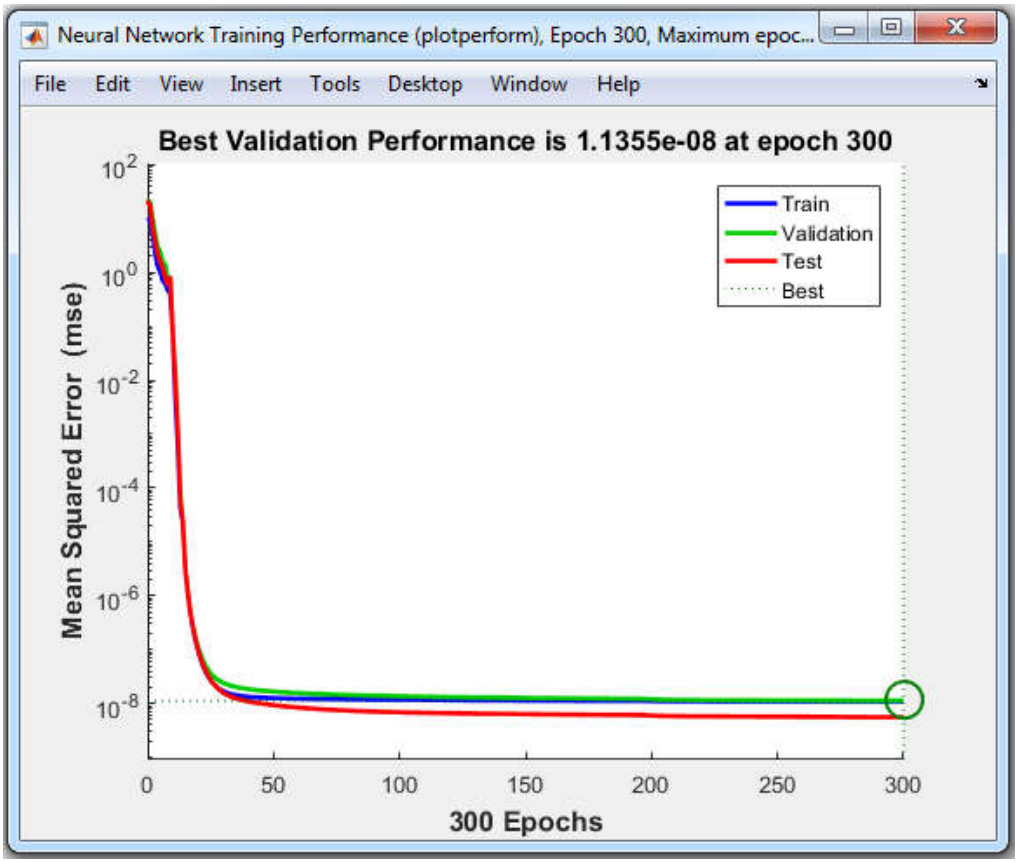


Рис.4.44. Вікно контролю процесу навчання нейронної мережі нейрорегулятора NARMA-L2 Controller

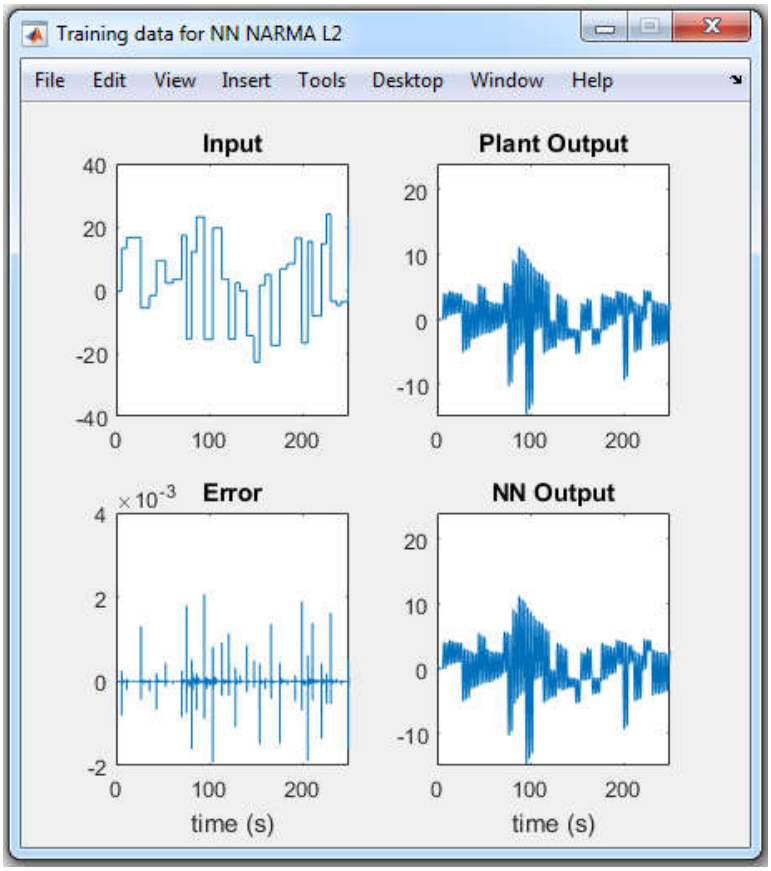


Рис.4.45. Результати тренування мережі нейрорегулятора NARMA-L2 Controller

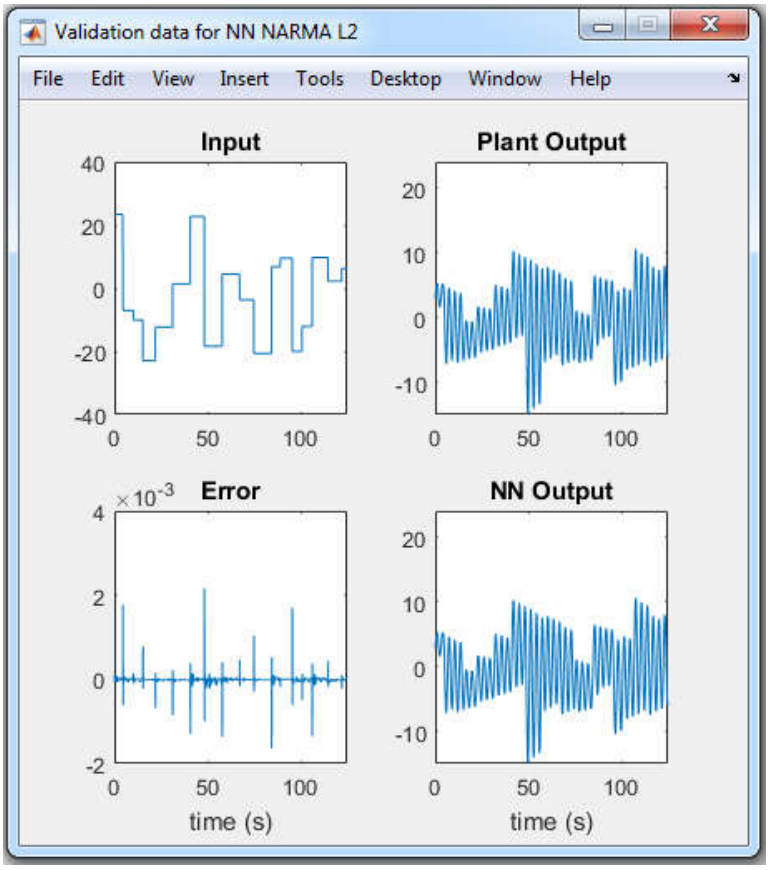


Рис.4.46. Результати тестування мережі нейрорегулятора NARMA-L2 Controller на контрольній множині

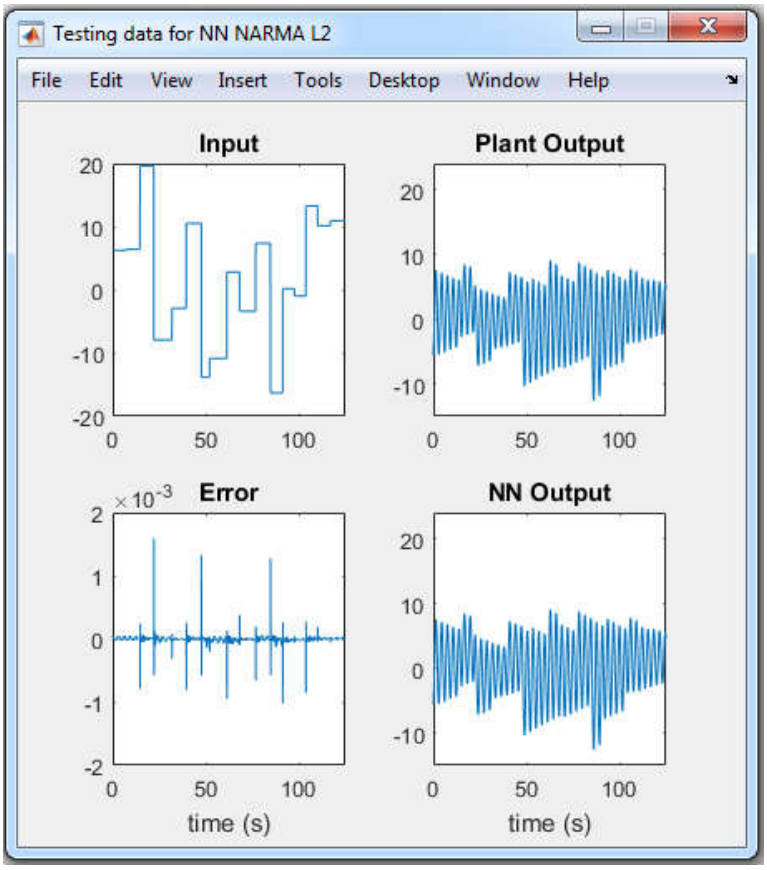


Рис.4.47. Результати тестування мережі нейрорегулятора NARMA-L2 Controller на тестовій множині

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.4.48. 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.

4.7.2 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 $N_i = 1 \div 4$, $N_j = 2 \div 5$.

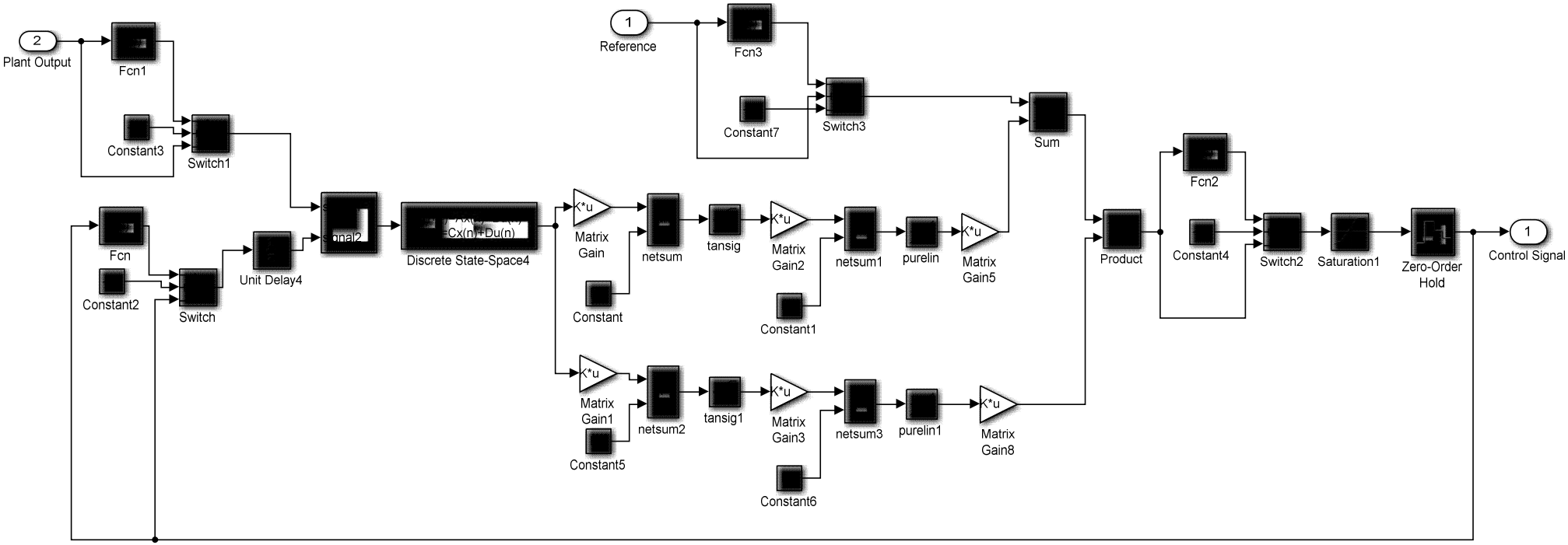


Рис.4.48. Структурна схема нейронної мережі нейрорегулятора NARMA-L2 Controller в системі Simulink

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.4.44 – рис.4.47, 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.

4.8 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.4.49.

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.

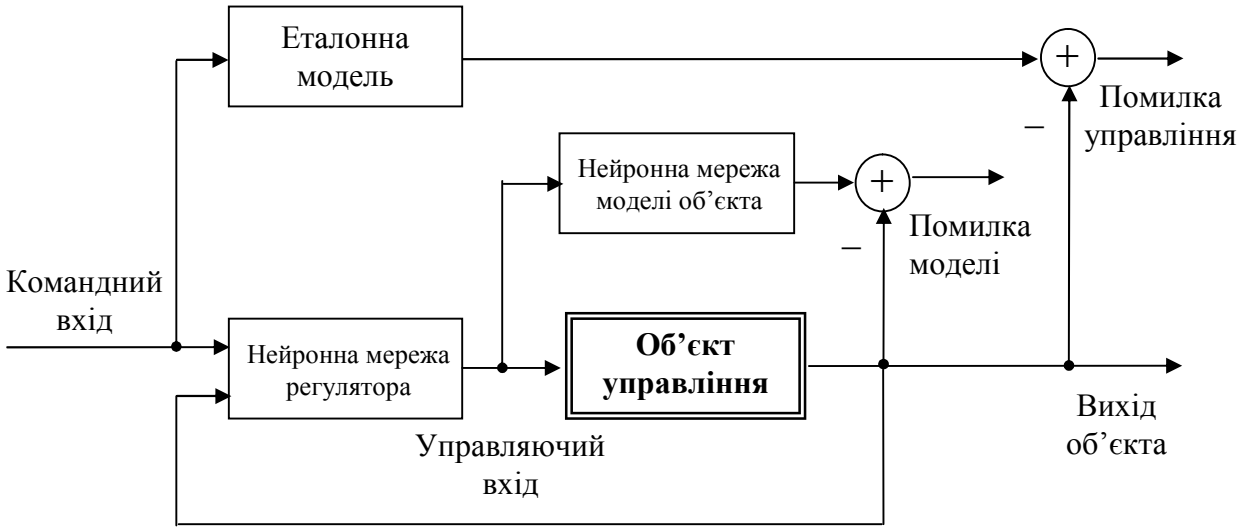


Рис.4.49. Структурна схема системи управління з еталонною моделлю

Архітектура нейронних мереж регулятора і об'єкту управління надані на рис.4.50. Мережі мають два шари нейронів і містять лінії затримок, що використовуються для формування входів нейронних мереж.

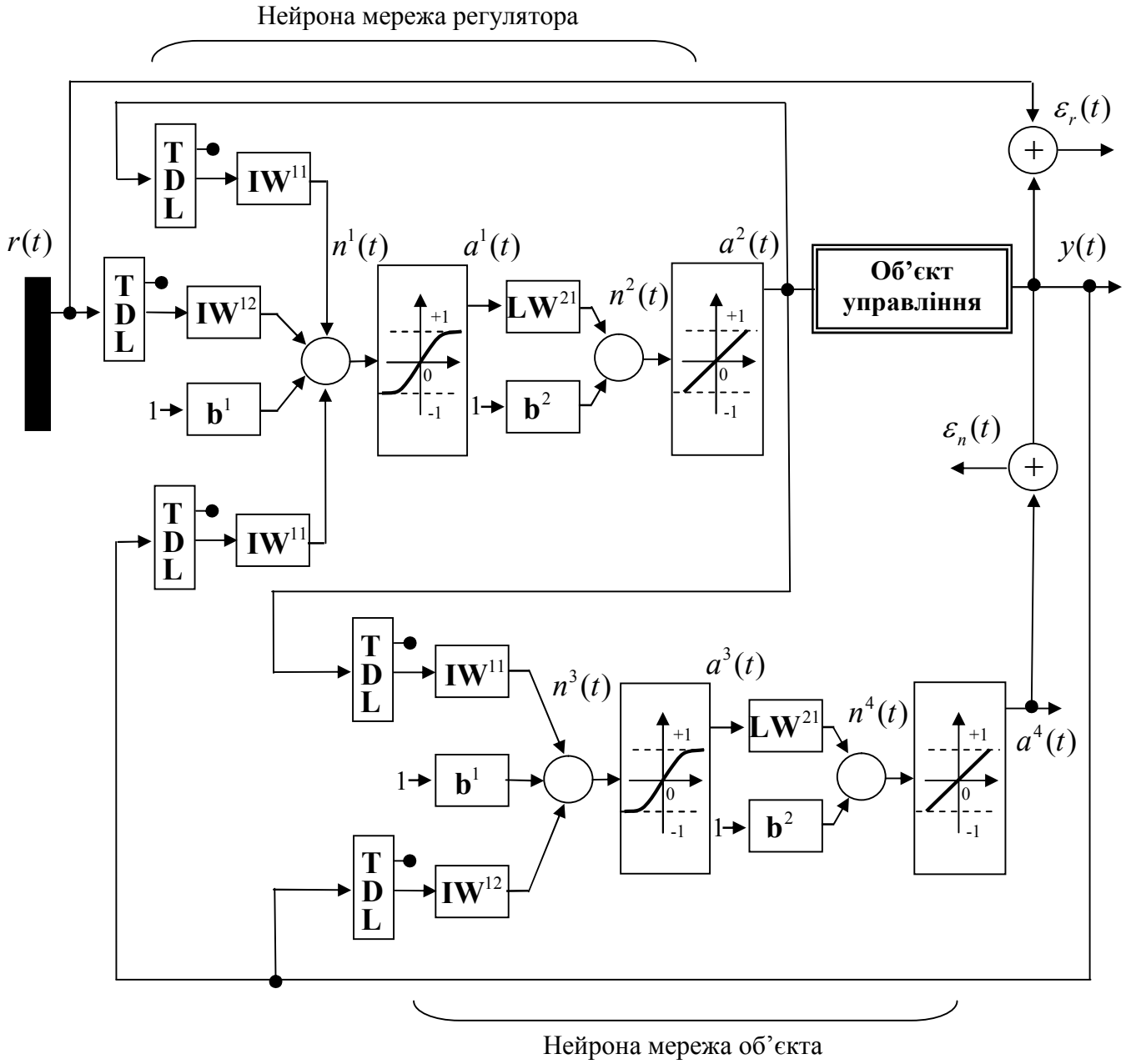


Рис.4.50. Схема включення нейронних мереж регулятора і об'єкту управління в системі управління з еталонною моделлю

4.9 Синтез нейрорегулятора Model Reference Controller з використанням системи MATLAB

4.9.1 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.4.51. Ця структура аналогічна схемі рис. 4.9, проте замість блоку нейрорегулятора NN Prediction Controller використовується блок Model Reference Controller. Схема блоку Subsystem відповідає рис. 4.13.

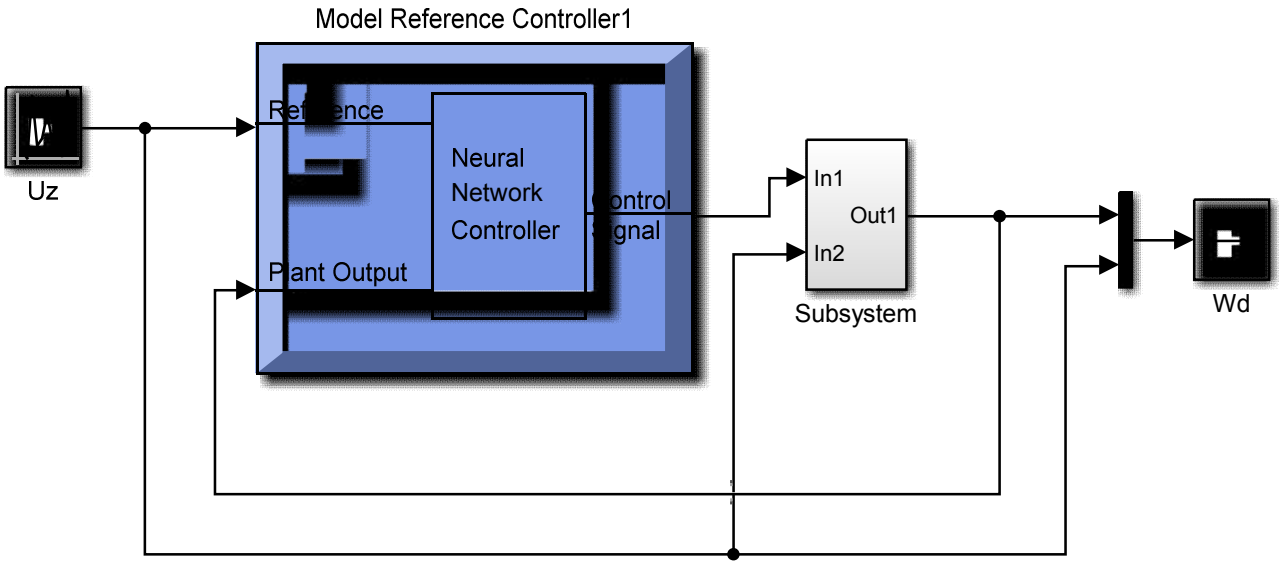


Рис.4.51. Схема системи управління з нейрорегулятором Model Reference Controller

Структурна схема нейрорегулятора Model Reference Controller показана на рис.4.52. Дана схема з'являється в окремому вікні при виборі пункту меню **Look Under Mask** блоку **Model Reference Controller**.

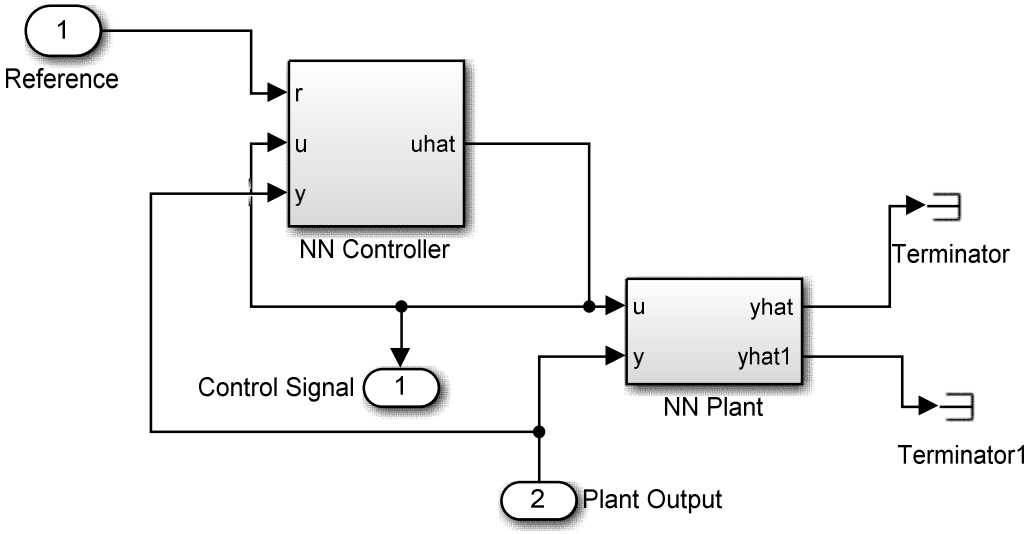


Рис.4.52. Структурна схема нейрорегулятора Model Reference Controller

Процес синтезу нейрорегулятора починається шляхом активізації блоку **Model Reference Controller**. З'являється вікно, показане на рис.4.53.

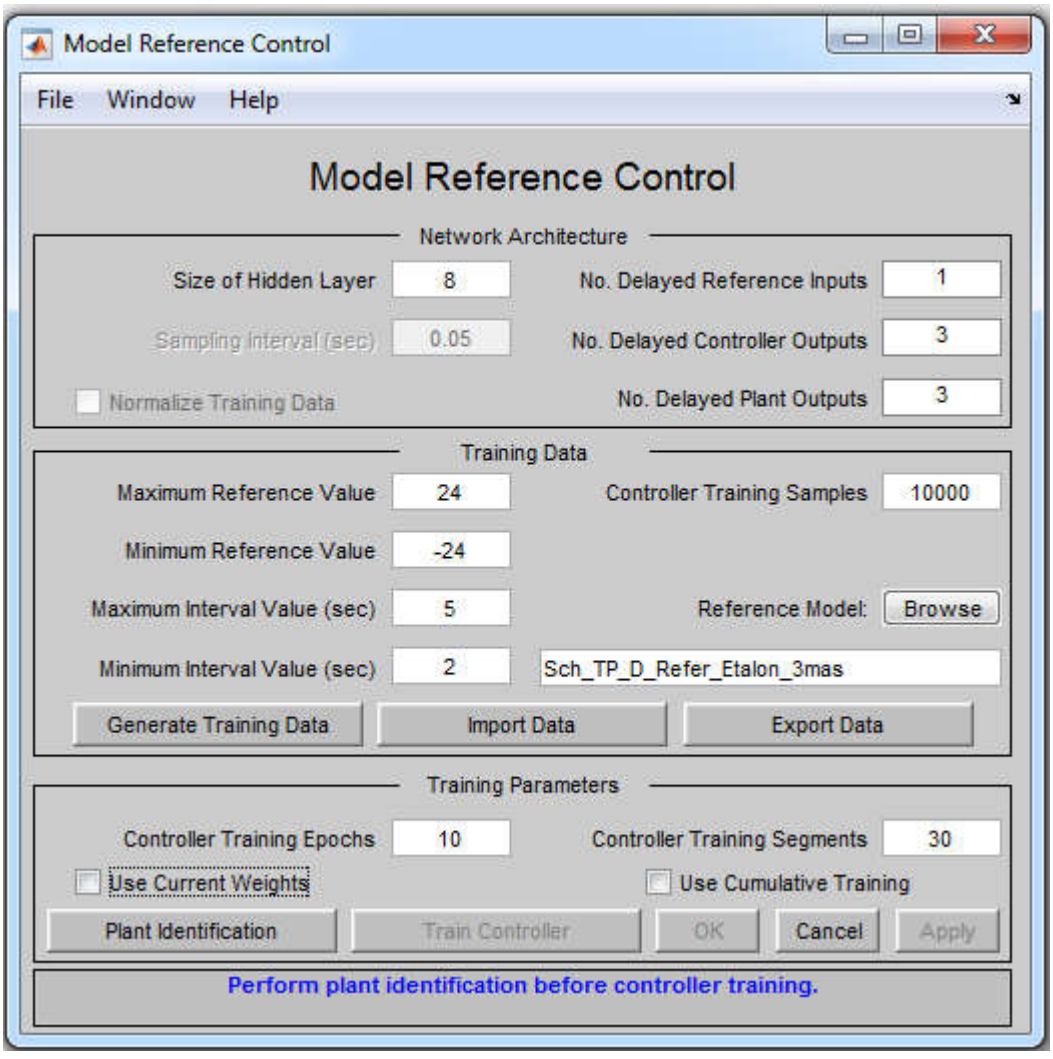


Рис.4.53. Вікно завдання параметрів нейрорегулятора Model Reference Controller

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.4.53 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.4.53).

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.4.54 показані моделі елементів нейронної мережі, по-

будовані за допомогою оператора **gensim(netn).** З даних елементів в системі Simulink побудована схема мережі, показана на рис.4.55.

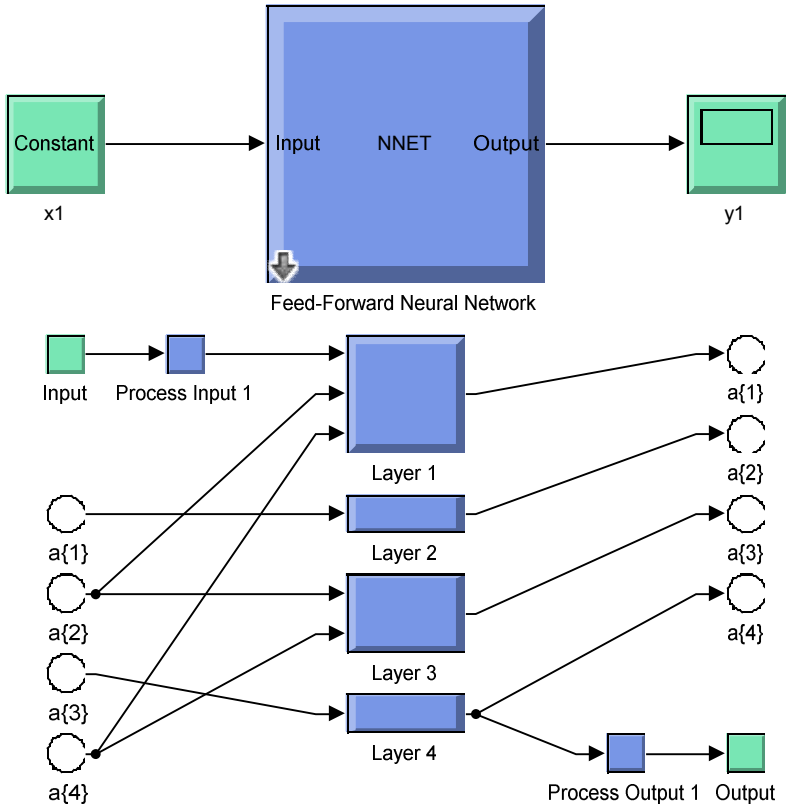


Рис.4.54. Моделі елементів динамічної мережі нейрорегулятора Model Reference Controller

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.4.53). 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.

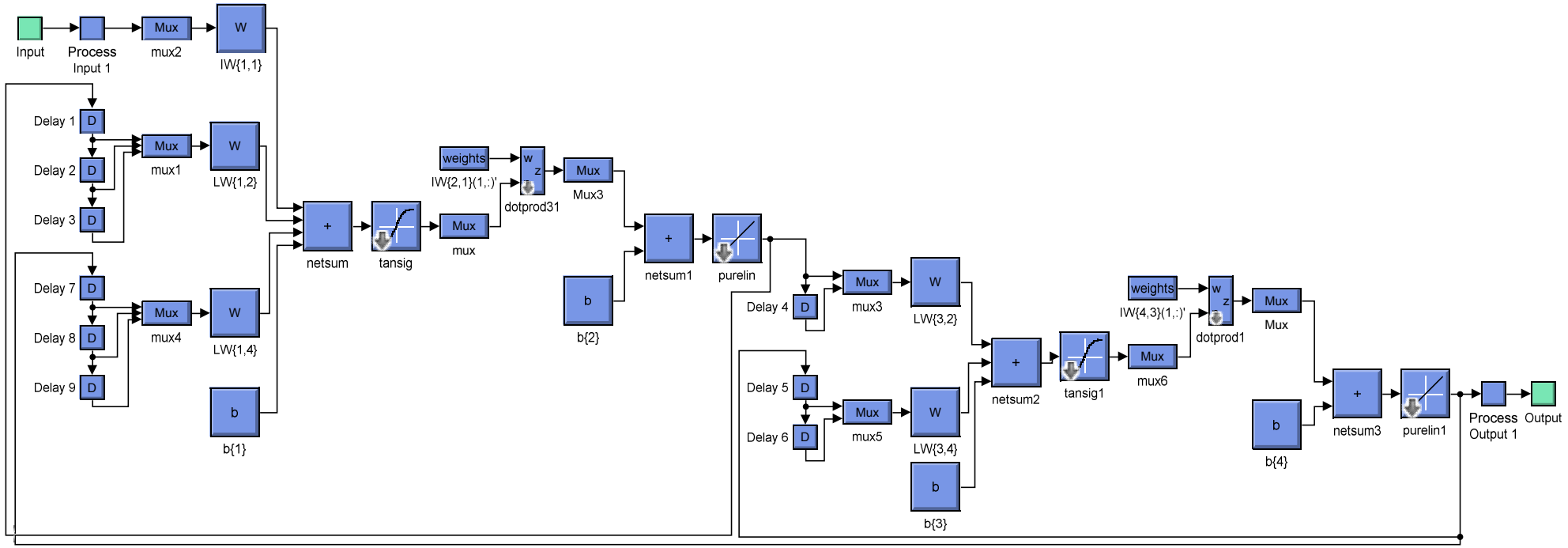


Рис.4.55. Модель динамічної мережі, що формується при синтезі нейрорегулятора Model Reference Controller

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..4.56.

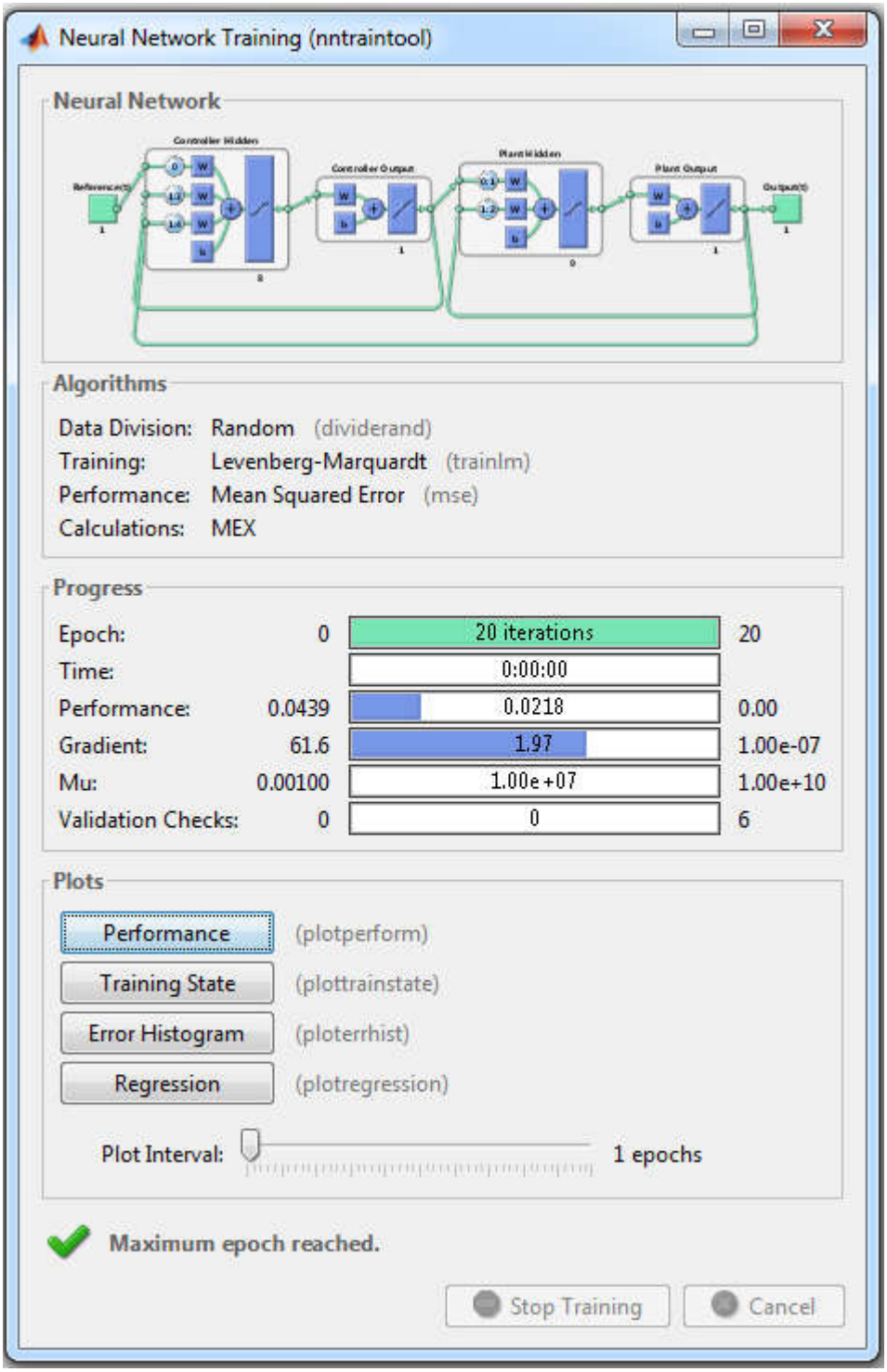


Рис. 4.56. Вікно, що відображає хід навчання та результати навчання нейронної мережі при тренувані нейронної мережі регулятора

При натисканні на кнопку **Performance** відкривається вікно **Neural Network Training Performance**, показане на рис. 4.57. У вікні відображаються графік **Mean Squared Error (mse)**, який показує, як змінюється середньоквадратична помилка навчальння.

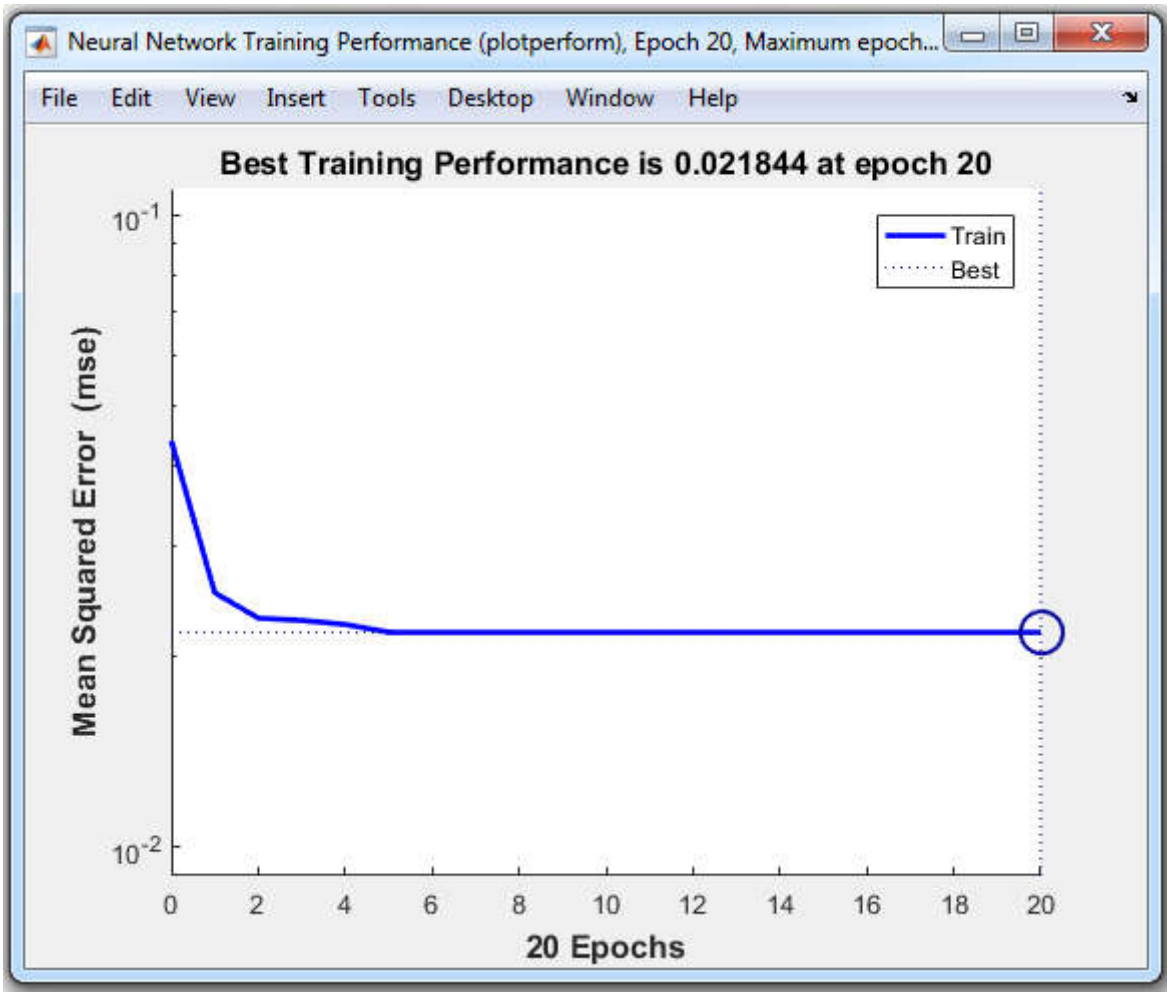


Рис.4.57. Вікно контролю процесу навчання нейронної мережі нейрорегулятора Model Reference Controller

При натисканні на кнопку **Training State** відкривається вікно **Neural Network Training State** (рис. 4.58**)** в якому відображається інформацію про стан навчання нейронної мережі на певний момент часу під час процесу навчання.

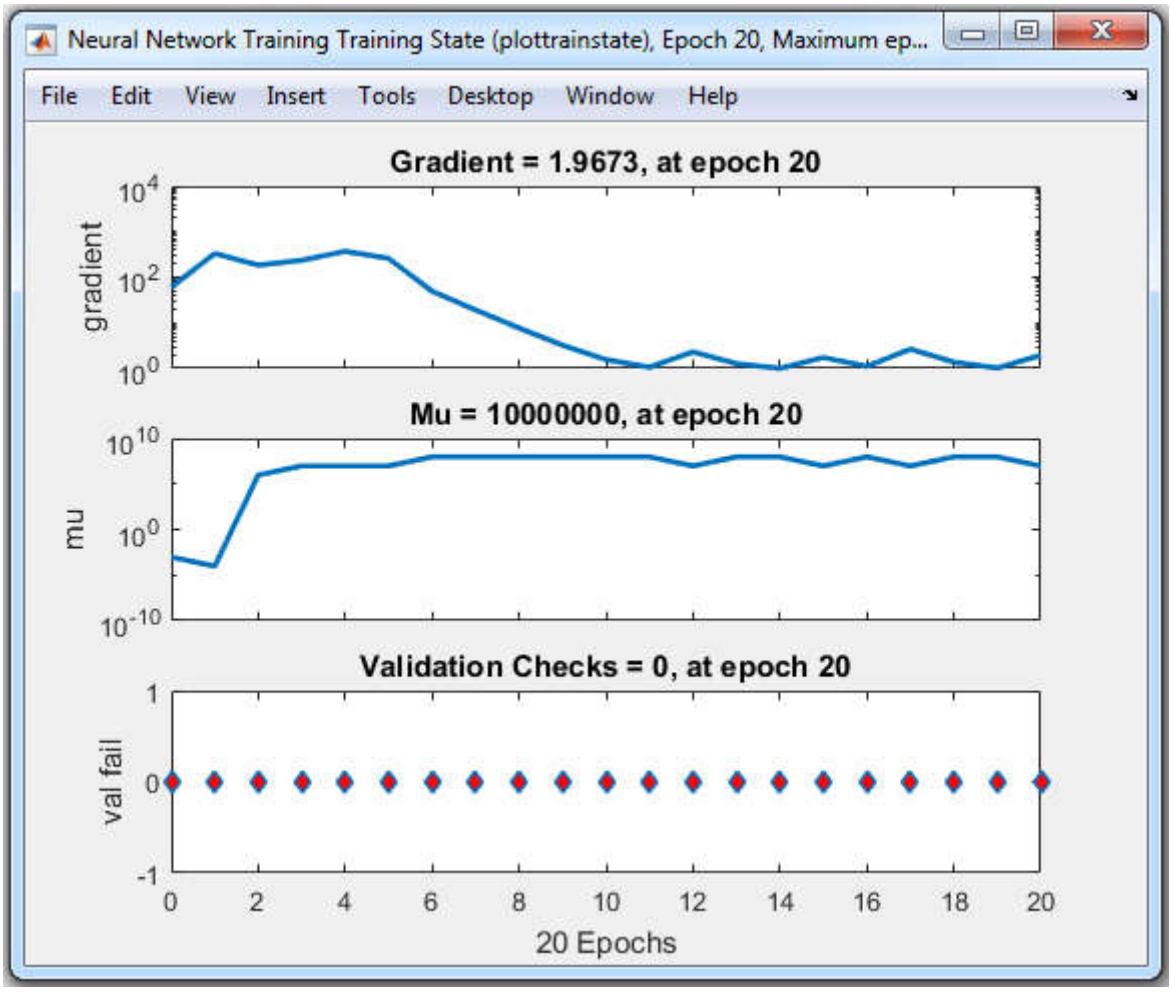


Рис.4.58. Вікно Neural Network Training State

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 візу-

альну інформацію про розподіл помилок та, за необхідності, скорегувати процес навчання чи архітектуру нейронної мережі (рис.4.59).

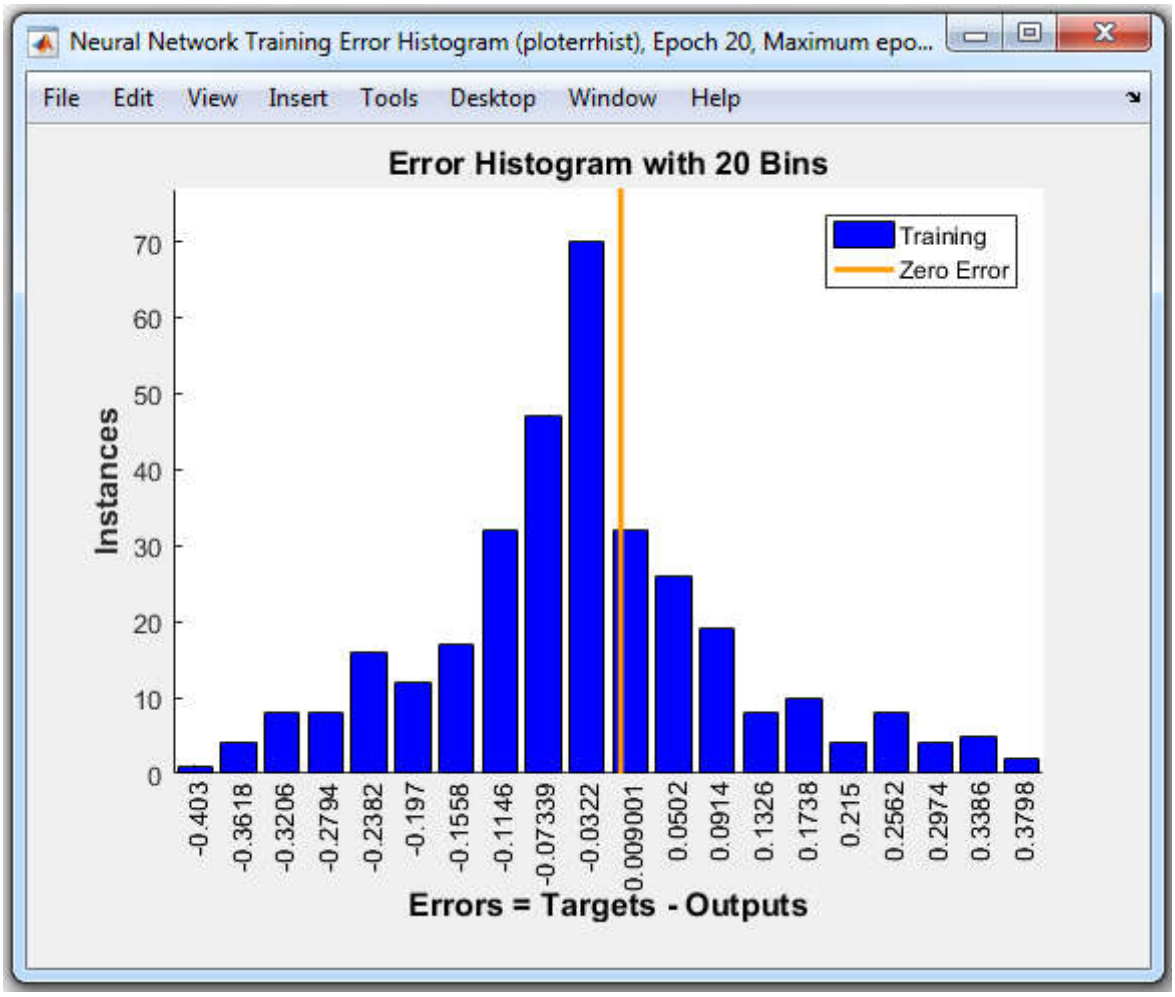


Рис. 4.59. Вікно Neural Network Training Error Histogram

При натисканні на кнопку **Regression** відкривається вікно **Neural Network Training Regression** (рис. 4.60), у якому показані графіки та інформація, які допомагають в оцінці та аналізі продуктивності нейронної мережі під час навчання.

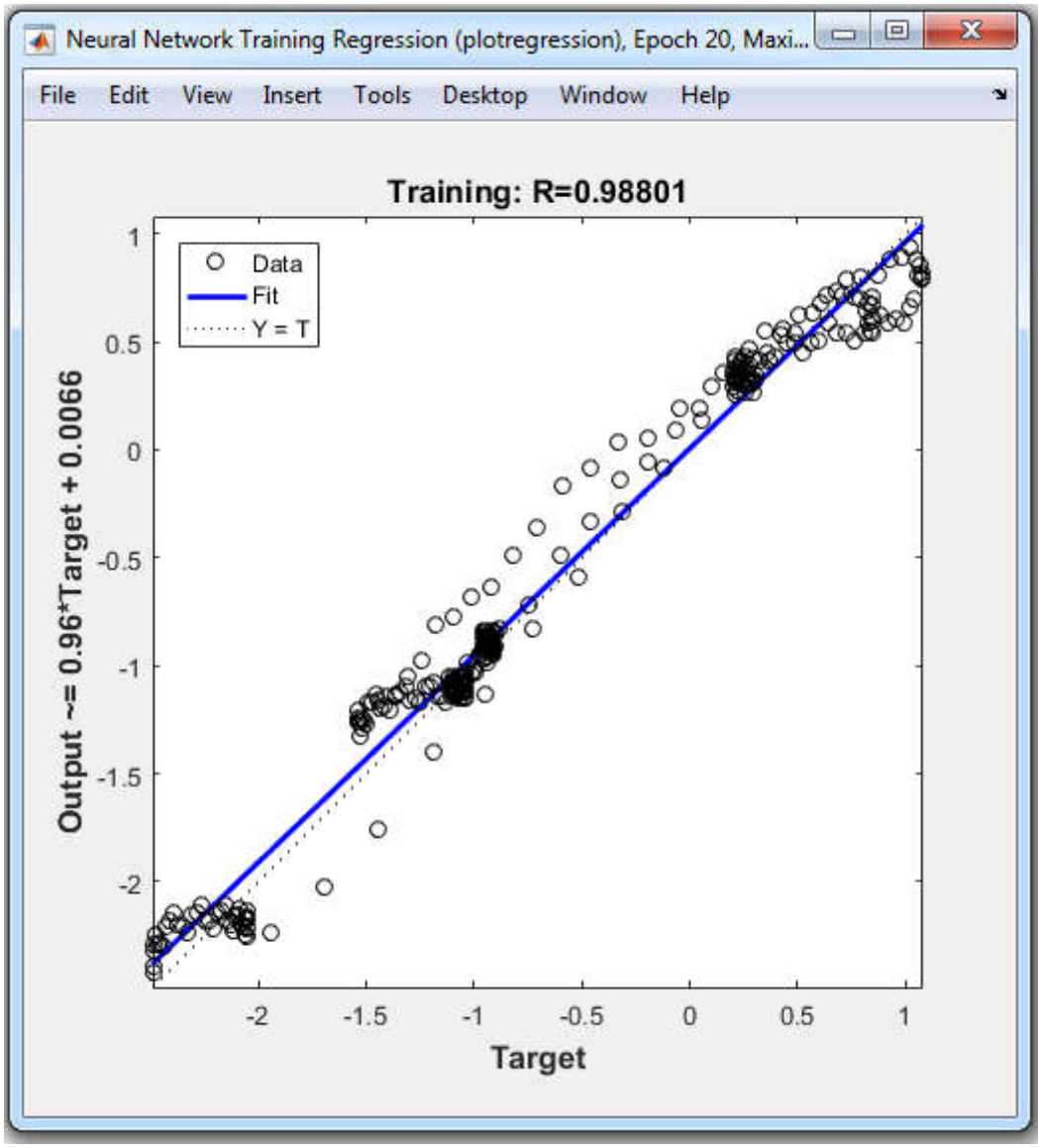


Рис. 4.60. Вікно Neural Network Training Regression

Після того, як навчання закінчене, графіки виходів еталонної моделі і об'єкту управління виводяться на екран у вікні, показаному на рис.4.61.

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.4.55). У системі Simulink дані шари представляються у вигляді структурних схем, показаних на рис.4.62 і рис.4.63.

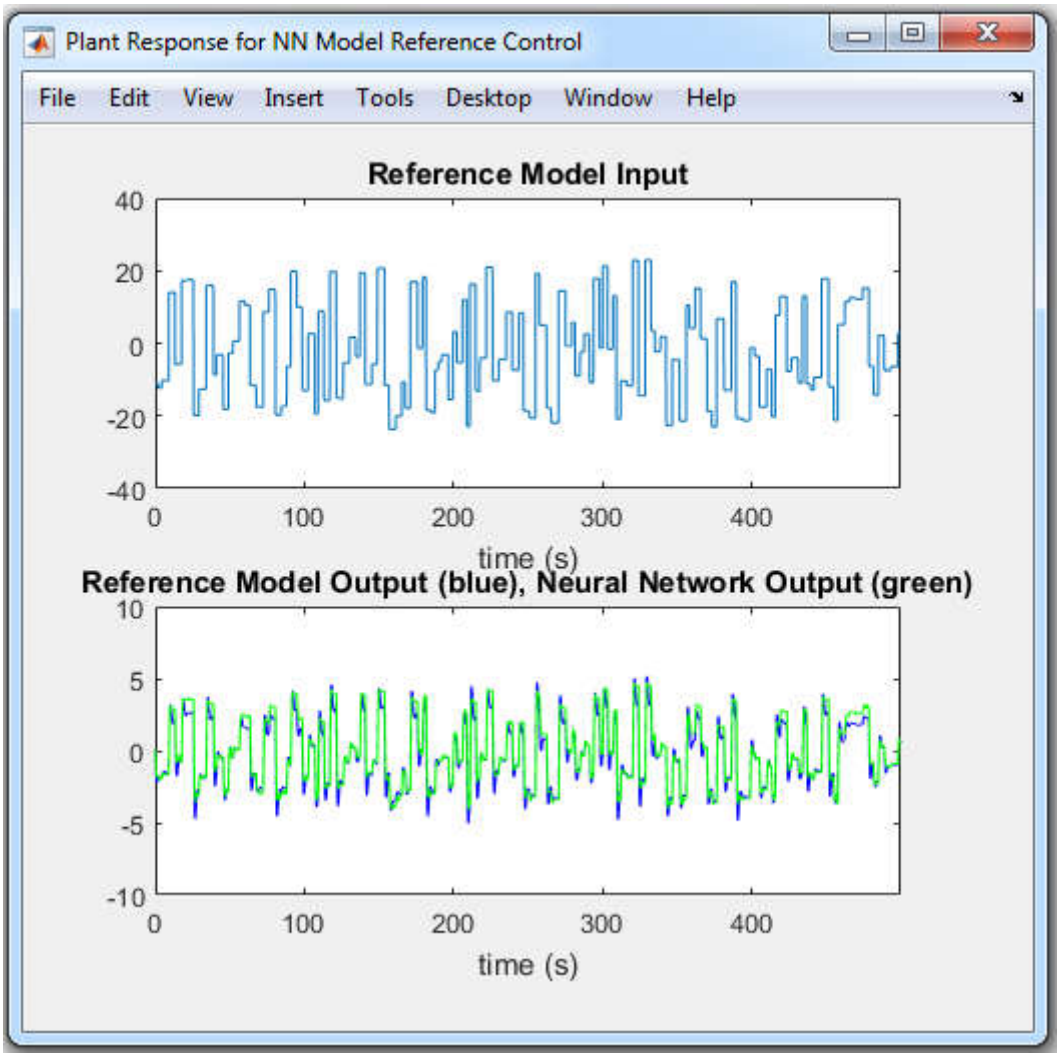


Рис.4.61. Результати тренування мережі регулятора Model Reference Controller

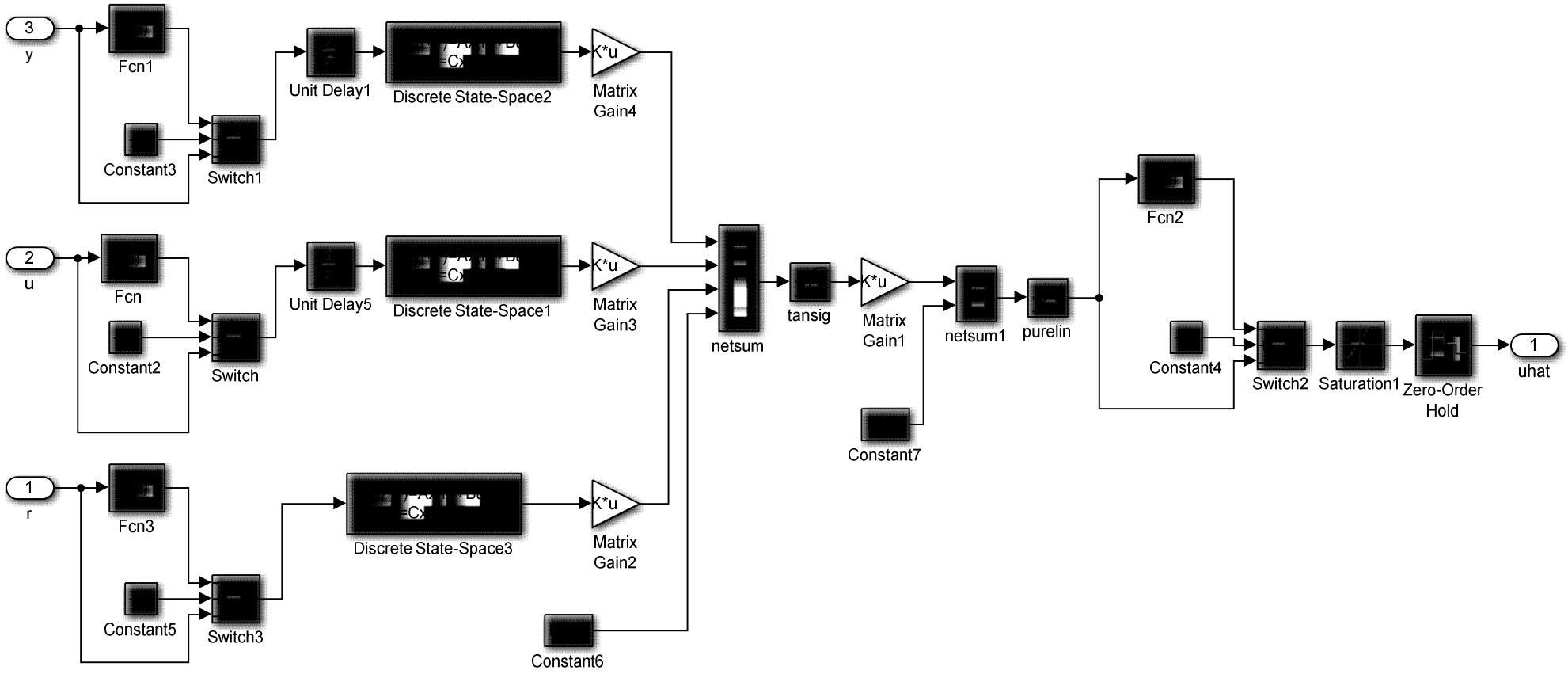


Рис.4.62. Структурна схема блоку **NN Controller** схеми Model Reference Controller

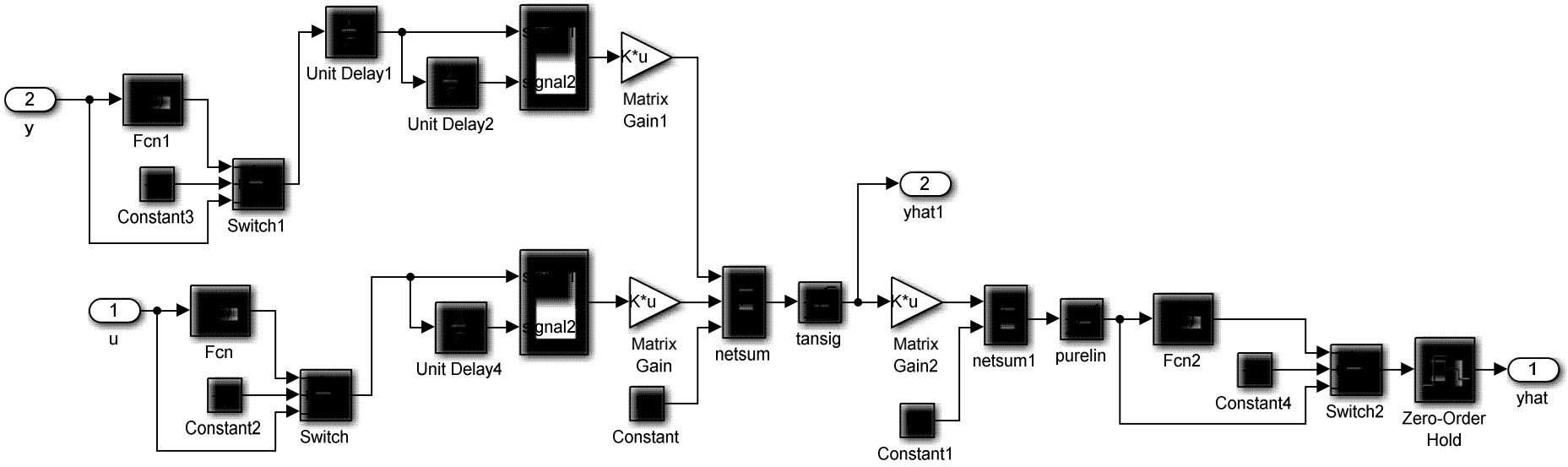


Рис.4.63. Структурна схема блоку **NN Plant** схеми Model Reference Controller

Як випливає з схеми рис.4.63, 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.

4.9.2 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 одно масова ситема. Дослідження показали, що ні при яких параметрах нейрорегулятора Model Reference Controller не вдалося отримати задовільні динамічні характеристики системи

Питання для самоконтролю.

1. Перечисліть регулятори реалізовані в пакеті прикладних програм Neural Network Toolbox системи MATLAB.
2. 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.
3. Поясніть структуру нейронних мереж, які використовуються в кожному регуляторі.
4. Наведіть структурні схеми системи управління з нейрорегулятором NN Predictive Controller, NARMA-L2 Controller і Model Reference Controller.
5. Перечисліть основні етапи синтезу нейромережевих систем управління, побудованих на основі вищеназваних регуляторів.
6. Як виконується формування навчальної послідовності?
7. Як відбувається контроль процесу навчання мережі?
8. Як вибираються параметри нейрорегулятора?
9. Які функції виконують нейронні мережі в вищеназваних регуляторах?
10. Скільки шарів містить нейронна мережа нейрорегулятора NARMA-L2 Controller?
11. Скільки нейронних мереж містить нейрорегулятор Model Reference Controller?

СПИСОК ЛІТЕРАТУРИ

1. Технології нейронних мереж і нечіткого моделювання 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. С.В.Ткаліченко. Штучні нейронні мережі: Навчальний посібник. – Кривий Ріг: Державний університет економіки і технологій, 2023. –150 с.
4. Кирик В. В. Математичний апарат штучного інтелекту в електроенергетичних системах:підручник..– Київ: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2019. –224 с.
5. Зайченко Ю.П. Основи проектування інтелектуальних систем.. – К.: Слово, 2004. –353 с.
6. Штучні нейронні мережі: навчальний посібник для студентів вищих навчальних закладів / О.Г. Руденко, Є.В. Бодянський. – К: Компанія СМІТ, 2006, 404 с.
7. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с
8. Тимощук П.В. Штучні нейронні мережі; Навч. посібн. - Львів: Львівська політехніка, 2011. -444 с.
9. Кононюк, А. Ю. .Нейронні мережі і генетичні алгоритми. - Київ: Корнійчук, 2008. - 468 с.
10. 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.
11. 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.
12. 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.
13. Глибовець М.М., Отецький О.В. Штучний інтелект. – К.: Вид. дім «КМ Академія», 2002. – 366 с.
14. 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. Лісови-ченко. — К. : ДП «Вид. дім «Персонал», 2011. — 544 с.

15. Методи та системи штучного інтелекту: Навчальний посібник для студентів напряму підготовки 6.050101 «Комп'ютерні науки» / Уклад. : А.С. Савченко, О. О. Синельніков. – К. : НАУ, 2017. – 190 с.
16. Інтелектуальні системи автоматизації : монографія / Аврунін О. Г., Владов С. І., Петченко М. В., Семенець В. В., Татарінов В. В., Тельнова Г. В., Філатов В. О., Шмельов Ю. М., Шушляпіна Н. О. – Кременчук : Видавництво «НОВА-БУК», 2021. – 322 с
17. Beale M. H., Carson E., Demuth H. B. *Deep Learning Toolbox: MATLAB R2021a*. The MathWorks, 2021. https://www.mathworks.com/help/deeplearning/ (стор. 1-400).
18. Гурко О.Г. Аналіз та синтез систем автоматичного управління у MATLAB: Навчальний посібник / О.Г. Гурко, І.Ф. Єрьоменко. Харків, ХНАДУ, 2012. – 284 с..
19. Моделювання систем управління в SIMULINK : навч. посібник / В. О. Богомолов, О. Г. Гурко, В. І. Клименко, Д. М. Леонтьєв, О. М. Красюк. Харків ХНАДУ, 2018. – 220 с.
20. McCulloch W. S., Pitts W. A logical calculus of ideas imminent in nervous activity // Bulletin Mathematical Biophysics. – 1943. – № 5. – P.115-133 (Имеется перевод: Дж. Маккаллок, У. Питтс. Логическое исчисление идей, относящихся к нервной деятельности. В кн.: Автоматы. – М.: ИЛ, 1956).
21. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain // Psychological Review. – 1958. – № 65.– P.386-407 (Имеется перевод: Розенблатт Ф. Принципы нейродинамики. – М.: Мир, 1966. – 480 с.).
22. Рябинин А. Д., Шквар А. М. Некоторые принципы функционального построения инвариантных бионических систем управления // Тр. 4-го Всесоюз. совещания. "Теория инвариантности и теория чувствительности автоматических систем". Ч. II. 26-30 апреля 1971.– Киев, 1971.– С 54-64.
23. Шквар А. М. Функциональные и структурные аспекты построения некоторых нейронных структур управления // Тр. 4-го Всесоюз. совещания "Теория инвариантности и теория чувствительности автоматических систем". Ч. II. 26-30 апреля 1971. – Киев, 1971. – С. 78-89.
24. Werbos P. J. Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD Thesis, Harvard University, Cambridge, MA. – 1974.– P. 124-137.
25. Rumelhart D. E., Hinton G E., Williams R. J. Learning internal representation by error propagation // In: D.E. Rumelhart, J.L. McClelland (Eds.) Parallel Distributed Processing, Vol. I Foundations. – Cambridge, MA: MIT Press, 1986. – P. 318-362.
26. Rumelhart D. E., Hinton G. E., Williams R. J. Learning representation by back-propagating errors // Nature. – 1986. – Vol. 323. – P. 533-536.
27. Narendra K.S., Parthasarathy K. Identification and control of dynamical system using neural networks // IEEE Trans. Neural Networks. – 1990. – Vol.1. – №1. – P. 4-27.

28. Narendra K.S., Parthasarathy K. Gradient methods for the optimization of dynamical systems containing neural networks // IEEE Trans. Neural Networks. – 1991. – Vol.2. – № 2. – P.252-262.
29. Sunner R. M., Slotine J. E. Gaussian Networks for Direct Adaptive Control // IEEE Trans. Neural Networks. – 1992. – Vol. 3. № 6. – P. 837-863.
30. Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches / Ed. by David A. White & Donald A. Sofge. N. – Y. Van Nostrand Rcinhold. – 1992. – 568 p.
31. Neural networks for control systems: A survey / K. J. Hunt, D. Sbarbaro, R. Zbikowski, P. J. Gawthrop // Automatica. – 1992. – Vol. 28. – № 6. – P. 1083–1112.
32. Suykens Johan A. K., Vandewall Joos P. L., De Moor Bart L. R. Artificial Neural Networks for Modelling and Control of Non Linear Systems / A. K.Suykens Johan, P. L. Vandewall Joos, L. R. De Moor Bart. – Kluwer Academic Publishers. Boston, Dordrecht, London, 1997. – 235 p.
33. Kosko B. Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence / B. Kosko. – New Jersey: Prentice-Hall Englewood Cliffs, 1991. – 375 p.
34. Vas P. Arlificial-Intelligence-Based Electrical Machines and Drives. Application of Fuzzy, Neural, Fuzzy-Neural and Genetic Algorithms / P. Vas. – Oxford University Press, Monographs in Electrical and Electronic Engineering, ISBN 0-19-859397-X. – 1999. – 626 p.
35. Efe M.O., Fiskiran A.M., Kaynak O. Derivation of a Parameter Stabilizing Training Criterion for Adaptive Neuro-Fuzzy Inference Systems in Motion Control / M. O. Efe, A. M. Fiskiran, O. Kaynak // International Journal of Systems Science. – 2001. – Vol.32. – №:4. – P. 513–521.
36. Werbos P.J. Neurocontrol and related techniques / P. J. Werbos // In Handbook of Neural Computing Applications. Hrsg. Maren A., Harston C., Pap R. Academic Press. San Diego, 1990. – P. 345-380.
37. Werbos P. J. Backpropagation and neurocontrol: A review and prospectus / P. J. Werbos // Proc. of International Joint Conf. On Neural Networks. Washington, DC. – 1989.–Vol. 1. – P.209–216.
38. Holland J. H. Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence / J. H. Holland. – London : Bradford book edition, 1994. – 211 p.
39. Taylor Stephen J. C. Neural Networks for Control. – Cambridge: MIT Press, 1999. – 400 c.
40. Lim Chee Peng, Loo Patrick P. S., Lee Keith M. C. Neural Network Control of Robot Manipulators and Nonlinear Systems. – Singapore: World Scientific Publishing, 1996. – 416 c.
41. Hsieh M. A. Neural Networks for Control. – Boston: Birkhäuser, 1993. – 298 c.
42. Chakrabarti Vladimir K. Neural Networks in Control Systems: Theory and Applications. – London: Springer, 1994. – 560 c.

43. Jack R. L. K. Neural Networks for Control. – Berlin: Springer, 2004. – 288 c.
44. Lim C. P. Artificial Neural Networks for Engineering Applications. – CRC Press, 2000. – 272 c.
45. Aguiar R. B. R., Pereira A. D. S. Artificial Neural Networks in Control Systems. – Springer, 2006. – 532 c.
46. Lim C. P., Loo P. P. S., Lee K. M. C. Neural Network Control of Robot Manipulators and Nonlinear Systems. – World Scientific Publishing, 2003. – 416 c.
47. Xie L., Zhang L. Neural Networks for Control and System Identification. – Springer, 2011. – 512 c.
48. Aggarwal C. Neural Networks and Deep Learning: A Textbook. – Springer, 2018. – 472 c.
49. Wei D. S. L., Ng T. S. Intelligent Control Systems with MATLAB. – CRC Press, 2013. – 320 c.
50. Doan D. D. Artificial Neural Networks in Control and Automation: A Practical Approach. – London: Springer, 2009. – 330 c
51. Goldberg D. E., Deb K., Dark J. H. Genetic algorithms: noise, and the sizing of populations // Complex Systems. – 1992. – Vol. 6. – P. 333-362.
52. Grefenstette J. J. Optimization of control parameters for genetic algorithm // IEEE Trans. Sys., Man and Cybern. – 1986. – Vol. 16. – № 1. – P. 122-128.
53. Hesser J., Manner R. Towards an optimal mutation probability in genetic algorithms // In Parallel Problem Solving from Nature, Vol. 496 of Lecture Notes in Computer Science, Springer, Berlin. – 1991. – P.23-32.
54. Krietinsson K., Dumont G. A. System identification and control using genetic algorithms // IEEE Trans. Sys., Man and Cybern. – 1992. – Vol. 22. – № 5. – P. 1033-1046.
55. Rumelhart D. E., Hinton G. E., Williams R.J. Learning internal representations by error propagation // Parallel Distributed Processing. – 1986. – Vol.1.– № 8. – P. 318- 362.
56. Liu D. C., Nocedal J. On the limited memory BFGS method for large scale optimization // Math. Prog. – 1989. – Vol. 45. – P. 503-528.
57. Battiti R. First and second order methods for learning: Between steepest descent and Newton's method // Neural Computation. – 1992. – Vol.4. – №2. – P. 141-166.
58. Hagan M. T., Mcnhaj M. Training feedforward networks with the Marquardt algorithm // IEEE Transactions on Neural Networks. – 1994. – Vol. 5. – № 6. – P. 989-993.
59. Leontaritis I. J., Billings S. A. Input-output parametric model for nonlinear systems. Part 1: Deterministic non-linear systems. Part 2: Stochastic non-linear systems // Int. J. Control. – 1985. – Vol. 41. – №2. –P. 303-344.
60. Billings S. A., Chen S., Korenberg M. J. Identification of MIMO non-linear systems using forward-regression orthogonal estimator // Int. J. Control. – 1989. – Vol. 49. – №9. – P. 2157-2189.

61. Chen S., Billings S. A., Cowan C. F., Grant P. M. Practical identification of NARMAX model using radial basis function // Bit.J. Control. – 1990. – Vol.52.– №6. – P. 1327-1350.
62. Peyton J. C., Billings S. A. Mean levels in nonlinear analysis and identification //Int. J. Control. – 1993. –Vol.58. – №5. – P.1033-1052.
63. Bunke J. Kunstliche neuronale Netze zur Systemidentifikation aus gestorten Mefiwerten. Forschr. Ber. VDI, Reihe 8, № 667. – Dusseldorf: VDI Verlag. – 1997. – P. 37-51.
64. Nergaard M. Neural networks for modelling and control of dynamic systems: a practitioner's handbook. London: Springer. – 2000. – 139 p.

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному мережному режимі

Канюк Геннадій Іванович
Василець Тетяна Юхимівна
Варфоломієв Олексій Олексійович

ТЕХНОЛОГІЇ НЕЙРОННИХ МЕРЕЖ І НЕЧІТКОГО МОДЕЛЮВАННЯ В СИСТЕМАХ УПРАВЛІННЯ

У двох частинах

Частина 1

Навчально-методичний посібник
для здобувачів вищої освіти другого (магістерського) рівня
за спеціальністю 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

В авторській редакції

Підписано до розміщення 28.03.2025. Гарнітура Times New Roman.
Ум. друк. арк. 13,1. Обсяг 4,676 Кб. Зам. № 112/25.

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