Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Системи штучного інтелекту

Методи штучного інтелекту

Комп’ютерний практикум

Видання друге, перероблене і доповнене

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра
за освітньою програмою «Технічні та програмні засоби автоматизації»
спеціальності 174 Автоматизація, комп'ютерно- інтегровані технології та робототехніка

Укладачі: Л. Д. Ярощук, Є. О. Тюріна, Р. О. Путятін

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2025

УДК 004.8:681.5
С 34

Укладачі: *Ярощук Людмила Дем'янівна*, канд. техн. наук, доц.
Тюріна Євгенія Олександрівна, *PhD*, асист.
Путятін Редріх Олегович

Рецензент *Корнієнко, Б. М.*, д-р техн. наук, проф.,
каф. інформаційних систем та технологій факультету інформатики та обчислювальної техніки КПІ ім. Ігоря Сікорського

Відповідальний редактор *Цапар В. С.*, канд. техн. наук, завідувач каф. технічних та програмних засобів автоматизації інженерно-хімічного факультету КПІ ім. Ігоря Сікорського

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 6 від 10.04.2025 р.)
за поданням вченої ради інженерно хімічного факультету
(протокол № 3 від 24.03.2025 р.)

XXX **Системи штучного інтелекту.** Методи штучного інтелекту [Електронний ресурс] : комп. практикум : навч. посіб. для здобувачів ступеня магістра за освіт. програмою «Технічні та програмні засоби автоматизації» спец. 174 Автоматизація, комп'ютерно- інтегровані технології та робототехніка / КПІ ім. Ігоря Сікорського ; уклад.: Л. Д. Ярощук, Є. О. Тюріна, Р. О. Путятін. – 2-ге вид., перероб. і доп. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2025. – 108 с.

Посібник містить теоретичний матеріал та завдання для комп'ютерного дослідження методів штучного інтелекту в задачах автоматизації технологічних об'єктів. Приділено увагу експертним системам, керуванню на основі нечітких множин і *fuzzy logic*, нейромережевим моделям та генетичним алгоритмам в задачах оптимізації. Передбачено використання математичних процесорів *Matlab*.
Посібник є другим, доповненим та переробленим виданням навчального посібника «Інтелектуальні системи управління: Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: Л. Д. Ярощук, В. І. Бородін. – Електронні текстові дані (1 файл: 1,91 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 81 с.
Призначений для здобувачів ступеня магістра за спеціальністю «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» усіх форм навчання.

УДК 004.8:681.5

Реєстр. № НП 24/25-421. Обсяг 6,2 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Різноманіття сучасних складних багато-параметричних об’єктів викликає зростання зацікавленості розробників як математичного, так і технічного забезпечення систем керування у методах моделювання та керування, які застосовують спрощений гнучкий математичний апарат й емпіричну логіку штучного інтелекту (ШІ). Метою дисципліни «Методи штучного інтелекту», яка є першою частиною дисципліни «Системи штучного інтелекту», є вироблення у студентів компетентностей, які дозволяють реалізувати методи ШІ в системах керування.

Найбільш поширеними при автоматизації об’єктів різної природи наразі є такі напрямки реалізації ШІ, як експертні системи, нечіткі множини та логічний висновок, нейронні мережі, генетичні алгоритми.

Головною особливістю методів ШІ є потреба отримувати та використовувати емпіричні знання фахівців предметних областей.

Навчальний посібник містить завдання, пов’язані з усіма згаданими вище напрямками. Важливою відмінністю цього комп’ютерного практикуму від інших є спрямованість досліджень на технологічні об’єкти різноманітних виробництв. Реальні умови таких об’єктів спонукають студентів заглиблюватись в особливості методів та їхні можливості. Удосконалення математичного забезпечення таких систем є актуальним напрямком розвитку систем автоматизації.

Існують спеціалізовані математичні пакети, а також окремі системи програмування, які дозволяють моделювати та досліджувати методи ШІ в системах керування. При виконанні завдань практикумів передбачено використання математичних процесорів *MS Excel* та *MATLAB*. Останній має спеціальні бібліотеки для реалізації методів ШІ, тому в посібнику суттєва увага приділена ознайомленню студентів з цими бібліотеками.

Організація кожного практикуму, що використовує *MATLAB,* передбачає спочатку покрокове виконання блоку завдань на прикладі одного для всіх студенів технологічного об’єкту, а потім виконання цього ж блоку завдань на іншому об’єкті, особливості якого студент знає досконально (за вибором студента).

Студентам рекомендовано вибирати для досліджень типові технологічні об’єкти, описи яких є у літературних джерелах відкритого доступу, об’єкти дослідження бакалаврського дипломного проекту чи магістерської дисертації. Це дозволяє сприймати студента в ролі достатньо кваліфікованого експерта.

Посібник є другим, доповненим та переробленим виданням навчального посібника «Інтелектуальні системи управління: Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: Л. Д. Ярощук, В. І. Бородін. – Електронні текстові дані (1 файл: 1,91 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 81 с.

У порівнянні з попереднім виданням більш детально сформульовано рекомендації по виконанню завдань, у більшості практикумів уведено додаткові завдання для індивідуальних досліджень студентів. У новому виданні відсутня одна з робіт, натомість уведено три нових за додатковими темами, додані описи виконання робіт для сучасних версій *MATLAB*, оновлено список літератури.

ПРАКТИКУМ 1

ЗАСТОСУВАННЯ ТА ДОСЛІДЖЕННЯ СИСТЕМ ЕКСПЕРТНОГО ОЦІНЮВАННЯ

Мета роботи – навчитися отримувати, обробляти та аналізувати знання експертів при автоматизації технологічних процесів, дослідити особливості систем експертного оцінювання.

Стислі теоретичні відомості

При автоматизації складних технологічних систем виникають наступні задачі:

- оцінювання тісноти зв’язків між технологічними змінними, як вимірюваними, так і не вимірюваними;
- визначення факторів для включення у математичну модель;
- визначення пріоритетності встановлення контрольно – вимірювальної апаратури;
- визначення керувальних змінних;
- визначення причин аварійних ситуацій тощо.

Ці задачі можуть бути успішно розв'язані лише з урахуванням досвіду фахівців. Отже, класифікацію ситуацій й усунення тих з них, які можуть призвести до руйнування самої системи, доцільно здійснювати на основі систем експертних знань. Алгоритми і програмне забезпечення, які дають змогу отримати і обробити таку експертну інформацію, називаються ***системами експертного оцінювання*** (СЕО). При їх використанні експерти надають факторам, які впливають на певну систему, бали (так звані ***ранги***).

Правила виставляння рангів залежать від вибраного способу ранжування. В цій лабораторній роботі будуть досліджуватись два алгоритми: ***одночасного та попарного ранжування.*** На рис. 1.1 наведено загальну для обох способів схему алгоритму експертного оцінювання.

При ***одночасному ранжуванні*** експерт отримує для порівняння одразу весь перелік факторів і проставляє кожному з них відповідний ранг.

Алгоритм ***попарного ранжування*** базується на тому, що експерт порівнює фактори попарно.

Алгоритм, наведений на рис.1.1, має в цілому лінійну структуру, але його використання передбачає декілька циклів.

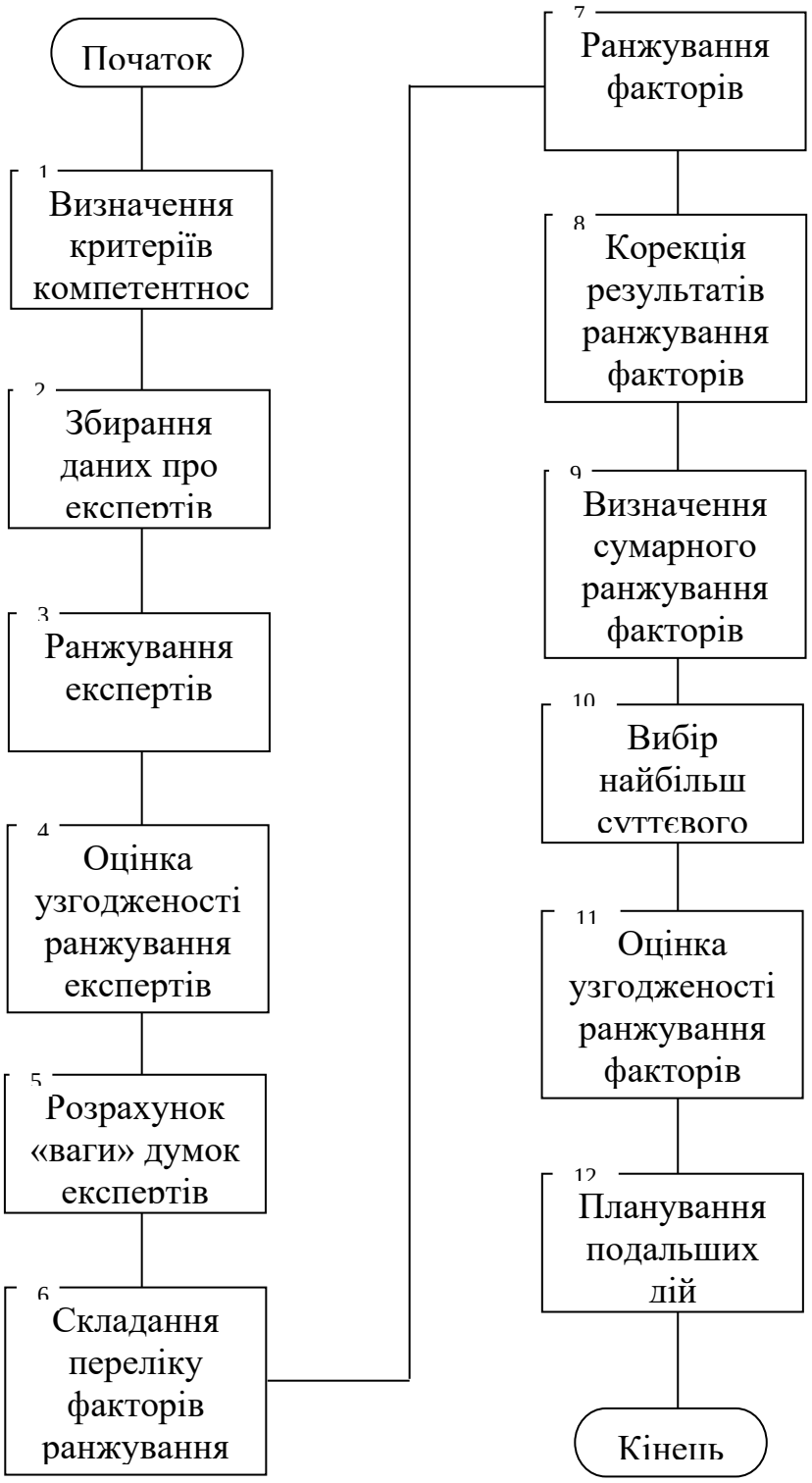


Рис. 1.1. Схема алгоритму роботи системи експертного оцінювання

Цикли в алгоритмі викликані, по-перше, опитуванням декількох фахівців а, по-друге, неодноразовим їх опитуванням. Опишемо вищенаведений алгоритм спочатку на прикладі одночасного ранжування.

Блок 1. *Визначення критеріїв компетентності експертів*. При виборі експертів керуються предметом експертизи, її метою і вимогами до однорідності експертів.

У прикладі, який наведено у лабораторній роботі, критеріями компетентності стали стаж роботи на даному технологічному обладнанні та виробничий розряд (кількість критеріїв *KR*=2).

Блок 2. *Збирання даних про експертів*. У цій частині алгоритму до ЕОМ заводять дані про кожного експерта відповідно до вибраних критеріїв компетентності (кількість експертів *NE*=10). Приклад до блоку 2 наведено у табл. 1.1 [1].

Таблиця 1.1. **Дані про компетентність експертів**

Критерій компетентності	Номер експерта									
	1	2	3	4	5	6	7	8	9	10
Стаж роботи (роки)	0,5	5	3,5	0,5	1,5	1	0,4	8	3	6
Розряд	5	6	6	4	5	4	3	5	6	5

Блок 3. *Ранжування експертів*. У цьому блоці експертам надають ранги у відповідності до числових значень критеріїв їхньої компетентності (чим більші стаж та розряд, тим менші відповідні ранги). Ранжування виконують за кожним з критеріїв окремо. Якщо є однакові ранги, то переходять до нормальної матриці ранжування, а потім розраховують суму рангів за всіма ранжуваннями. Результат такої обробки даних поданий у табл. 1.2.

Таблиця 1.2. **Нормальна матриця ранжування експертів**

Номер ранжування	Номер експерта										
	1	2	3	4	5	6	7	8	9	10	$p_{j,u}$
1(за стажем)	8,5	3	4	8,5	6	7	10	1	5	2	2
2 (за розрядом)	5,5	2	2	8,5	5,5	8,5	10	5,5	2	5,5	3+4+2
$\sum_{j-1}^{KR} y_{ij}$	14	5	6	17	11,5	15,5	20	6,5	7	7,5	-
Результуючий ранг	7	1	2	9	6	8	10	3	4	5	-

Блок 4. *Оцінка узгодженості ранжування експертів*. У табл. 1.2 величина $p_{j,u}$ – це кількість повторень u-о рангу у j-у рядку; $\sum_{j-1}^{KR} y_{ij}$ – сума рангів, отриманих i-м експертом у KR ранжуваннях.

Для з'ясування узгодженості ранжування експертів визначають коефіцієнт конкордації і оцінюють його статистичну значущість.

Приклад розрахунку коефіцієнта конкордації ранжування експертів [1]:

а) розрахунок середнього рангу нормальної матриці рангів експертів:

$$B = \frac{1}{2} KR(NE - 1) = 11;$$

б) обчислення суми квадратів різниць між членами сумарного ранжування і членами ряду, який складено із середніх значень:

$$S_e = \sum_{i=1}^{NE} \left(\sum_{j=1}^{KR} y_{ij} - B\right)^2 = 256;$$

в) визначення коефіцієнта конкордації ранжування експертів:

$$W_e = \frac{12 S_e}{KR^2 (NE^3 - NE) - KR \sum_{j=1}^{L} P_j}, \qquad (1.1)$$

де L – кількість рядків у табл. 1.2, які містять однакові («зв'язані») ранги, L = 2.

Величину P_j розраховують за формулою

$$P_j = \sum_{u=1}^{U} (p_{j,u}^3 - p_{j,u}),$$

де U – кількість типів «зв’язаних» рангів у j-му рядку табл. 1.2 (для j=1 приймаємо U = 1; для j = 2 приймаємо U = 3).

Статистичну значущість коефіцієнта конкордації перевіряють за χ^2-критерієм (критерієм Пірсона). Розрахункове значення визначають за виразом

$$\chi_e^2 = KR(NE - 1) W_e = 14{,}76.$$

При рівні значущості α = 1 та ступеню вільності $NU = NE - 1 = 9$, табличне значення $\chi_{e,tabl}^2$ критерію дорівнює 14.7.

Отже, виконується умова

$$\chi_e^2 > \chi_{e,\text{tabl}}^2 . \qquad (1.2)$$

Таким чином, можна вважати узгодженим ранжування експертів при $\alpha = 0{,}1$.

Блок 5. *Розрахунок «ваги» думок експертів*. Найдосвідченішим експертом вважається той, у якого сума рангів буде найменшою (у розглянутому прикладі це другий експерт). Його думці надається «вага» $\delta_2 = 2$. Думка найменш досвідченого експерта (7-го) має «вагу» $\delta_7 = 1$.

Для визначення «ваги» думок інших експертів використовують рівняння

$$\delta_i = a + c\sum_{j=1}^{KR} y_{ij} .$$

Для визначення коефіцієнтів *a* та *b* запишемо систему

$$\begin{cases} 2 = a + c \cdot 5; \\ 1 = a + c \cdot 20, \end{cases}$$

звідки $a = 7/3, b = -1/15$.

Для прикладу визначимо «вагу» думки 1-о експерта:

$$\delta_3 = \frac{7}{3} - \frac{1}{15}14 = 1{,}4 .$$

Блок 6. *Складання переліку факторів ранжування*. У список для ранжування включають фактори, які впливають на хід технологічного процесу.

Блок 7. *Ранжування факторів*. У цій частині алгоритму виконують власне ранжування як таке. Згідно з методом одночасного ранжування. експерти ставлять у відповідність кожному фактору ранг – число з натурального ряду $\overline{1,K}$, де *K* – кількість факторів у списку. Правило ранжування таке – чим менше фактор впливає на технологію, тим більший ранг йому треба надати.

Значущість кількох факторів може бути однаковою, і їм надають однакові ранги. Приклад ранжування наведений у табл. 1.3.

Таблиця 1.3. **Результати ранжування факторів**

Номер експерта ($j=\overline{1,NE}$)	Номер фактору ($i=\overline{1,K}$)					
	1	2	3	4	5	6
1	2	2	2	2	1	1
2	3	3	3	2	1	1
3	2	4	2	3	1	1
...	-	-	-	-	-	-

У цьому ж блоці виконується перехід до нормального виду матриці рангів (див. табл. 1.4), тобто враховують той факт, що декілька факторів набули одного і того ж рангу.

Таблиця 1.4. **Нормальна матриця рангів**

Номер експерта ($j=\overline{1,NE}$)	Номер фактору ($i=\overline{1,K}$)					
	1	2	3	4	5	6
1	4,5	4,5	4,5	4,5	1,5	1,5
2	5	5	5	3	1,5	1,5
3	3,5	6	3,5	5	1,5	1,5
...	-	-	-	-	-	-

Блок 8. *Корекція результатів ранжування факторів*. Елементи нормальної матриці коригують з урахуванням «ваги» думок фахівців, які розраховують у блоці 3.

Основними показниками сили впливу фактору на процес є «зважені» суми рангів:

$$\sum_{j=1}^{NE} a_{ij}\delta_j,$$

де a_{ij} – ранг, що був наданий i-у фактору j-м експертом без урахування його компетентності.

Результати експертного опитування, в яких врахована «вага» думок експертів, наведені у табл. 1.5.

Таблиця 1.5. **Таблиця «зважених» рангів**

Номер експерта ($j=\overline{1,NE}$)	Номер фактору ($i=\overline{1,K}$)						t_ν
	1	2	3	4	5	6	
1	6,30	6,30	6,30	6,30	2,10	2,10	4; 2
2	10,00	10,00	10,00	6,00	3,00	3,00	3; 2
3	6,76	11,56	6,76	9,65	2,90	2,90	2; 2
...	-	-	-	-	-	-	
$\sum_{j=1}^{NE} a_{ij}\delta_j$	52,72	69,37	69,87	65,88	36,89	41,63	

У табл. 1.5 параметр $t_{j,\nu}$ – кількість однакових рангів різних типів у рядку. Так, у першому рядку два різних типи: 6,30 та 2,10. Перших значень 4, інших 2; тому $t_{1,1}=4$; $t_{1,2}=2$. Якщо немає однакових значень у рядку, то $t_\nu=0$.

Блок 9. *Визначення сумарного ранжування факторів*. Виконують підсумовування «зважених» рангів по всіх експертах (див. $\sum_{j=1}^{NE} a_{ij}\delta_j$ табл. 1.5).

Блок 10. *Вибір найбільш суттєвого фактору*. Умова вибору - найменше значення суми рангів (у прикладі це 5-й фактор).

Блок. 11. *Оцінка узгодженості ранжування факторів*. Обчислюють коефіцієнт конкордації за формулою, аналогічною (1.1):

$$W_f = \frac{12S_f}{\left[NE^2(K^3-K) - \sum_{j=1}^{NE} T_j\right]\left(\frac{1}{NE}\sum_{j=1}^{NE}\delta_j^2\right)},$$

де

$$S_f = \sum_{i=1}^{K}\left(\sum_{j=1}^{NE} a_{ij}\delta_j - \frac{\sum_{i=1}^{K}\sum_{j=1}^{NE} a_{ij}\delta_j}{K}\right) = 1045{,}36;$$

$$T_j = \sum_{\nu=1}^{P} (t_{\nu j}^3 - t_{\nu j}),$$

де P – кількість типів «зв'язаних» рангів в j-у рядку.

Після цього розраховують значення критерію Пірсона за формулою

$$\chi_f^2 = NE(K-1)W_f = 11{,}9.$$

Табличне значення критерію Пірсона визначають за рівнем значущості α і ступенем вільності $NF = K - 1 = 5$. Так, для α=0,05 і NF=5 отримаємо $\chi_{f,tabl}^2$=11,1. У цьому разі виконується умова

$$\chi_f^2 > \chi_{f,tabl}^2, \qquad (1.3)$$

і можна казати про наявність узгодженості у думках експертів при ранжуванні факторів.

Блок 12. *Планування подальших дій.* Якщо умова (1.3) виконується, то задачу можна вважати розв’язаною. У тому випадку, коли умова (1.3) не виконується, то потрібно звернути увагу на правильність формулювання задачі дослідження, уточнити склад групи експертів і повернутися до початку розв'язання задачі (від блоку 1).

Розглянемо застосування алгоритму для метода попарного ранжування. Відмінності є для блоків 7-11, далі позначатимемо ці блоки індексом «п».

Блок 7п. Експерт згідно з цим методом оцінює більш значущий фактор одиницею (1), а інший нулем (0) [1]. Результат такого ранжування може бути подано таким, наприклад, як у табл. 1.6.

Таблиця 1.6. **Результати ранжування при парному порівнянні**

Номери факторів	1	2	3	...	*K*
1	-	1	0	-	-
2	0	-	1	-	-
3	1	0	-	-	-
...	-	-	-	-	-
K	-	-	-	-	-

Одиницю записуємо у комірку ij (i – номер рядка, j – номер стовпчика), якщо фактору i віддаємо перевагу перед фактором j. Таку таблицю заповнює кожний з NE експертів.

Блок 8п. Якщо треба врахувати компетентність експерта, то кожний елемент таблиці перемножують на «вагу» думки відповідного експерта.

Блок 9п. Для визначення результату ранжування створюють нову таблицю, у кожній її комірці вказують суму чисел, які знаходяться у відповідних комірках всіх NE таблиць.

Блок 10п. Підраховують суму чисел у всіх комірках для кожного рядка $i(i=\overline{1,K})$. Чим більша ця сума, тим більше значення має i-й фактор.

Блок 11п. Для визначення узгодженості ранжування використовують лише частину таблиці, розташовану або над головною діагоналлю, або під нею.

Коефіцієнт конкордації обчислюють за формулою

$$W=\frac{4Q}{Q_{\max}}=\frac{4\sum\limits_{i=1}^{K}\sum\limits_{j=1}C_2^{r_0}}{NE(NE-1)K(K-1)},$$

де $C_2^{r_0}$ – біноміальний коефіцієнт; r_{ij} – число, що розміщено у комірці з координатами $ij(i\neq j)$.

Значення Q може бути отримане таким чином:

$$Q=\sum_{\substack{i=1\\j=1}}^{K}r_{ij}^2-NE\sum_{\substack{i=1\\j=1}}^{K}r_{ij}+C_2^{NE}C_2^K.$$

Для оцінювання значущості коефіцієнта узгодженості W використовують критерій χ^2.

Розраховують значення цього критерію так:

$$\chi_f^2=\frac{4}{NE-2}\left(Q-\frac{1}{2}C_2^KC_2^{NE}\frac{NE-3}{NE-2}\right),$$

Далі порівнюють χ_f^2 з табличним значенням $\chi_{f,tabl}^2$ для рівня значущості α і NF ступенів вільності.

Параметр *NF* обчислюють за формулою

$$NF = \frac{C_2^K \, NE(NE-1)}{(NE-1)^2}$$

і округлюють до найближчого цілого.

Порядок виконання роботи

1. Сформулювати задачу дослідження. Визначити список факторів, які треба ранжувати за ступенем впливу на певний критерій.

2. Провести експертні опитування з врахуванням компетентності експертів та без нього. Роботу з програмами виконувати за зразками, наведеними в лабораторній роботі (див. рис. 1.2 – 1.8 для одночасного ранжування та рис. 1.9 – 1.11 для попарного).

3. Розрахувати показники експертизи за обома способами ранжування. Визначити узгодженість ранжування факторів та експертів за заданими критеріями компетентності. Розрахунки виконати у середовищі *MS Excel*.

Вміст звіту

Опис предметної області, задача дослідження, початкові та нормальні матриці ранжування факторів та експертів, результати ранжування у числовому та графічному видах. Розрахунки, виконані у *MS Excel*.

За результатами лабораторної роботи дослідити суперечливість думок експертів. Зробити висновки з наступних питань:

а) про можливість роз’язання поставленої задачі аналітичним чи експериментально-статистичним методами;

б) про узгодженість думок експертів;

в) про узгодженість вибраних критеріїв компетентності;

г) про вплив критеріїв компетентності експертів на результати ранжування факторів;

д) про наявність суперечностей між результатами двох методів ранжування.

Контрольні запитання

1. Для чого використовують системи експертного оцінювання?

2. Які є СЕО? Як виконується ранжування в кожній з них?

3. Як визначити узгодженість думок експертів у кожній із систем?

4. Як визначити «вагу» думки експерта?

5. Як використати дані про компетентність експертів?

5. Які можуть бути подальші дії при неузгодженості роботи експертів?

6. Як подати результати експертизи графічно?

7. Як можна оцінити результативність експертизи за видом графічного зображення її результатів?

ПРАКТИКУМ 2

ДОСЛІДЖЕННЯ ПОРУШЕНЬ ПРОЦЕСУ ВИПАЛЮВАННЯ ЦЕГЛИ ЗА ДОПОМОГОЮ ЕКСПЕРТНОЇ СИСТЕМИ

Мета роботи – дослідити структуру бази знань та спосіб роботи механізму виведення експертної системи (ЕС) прогнозувального та діагностувального типів для предметної області «Керування процесом випалювання керамічної цегли».

Стислі теоретичні відомості

Задача запобігання порушенням нормального перебігу технологічних процесів виникає при розробці будь-якої системи керування. Особливо актуальне розв’язання цієї задачі для виробництв, яким притаманні такі риси, як велика потужність, енергоємність, наявність вибухо - та пожежонебезпечних речовин. Таким, зокрема, є виробництво керамічної цегли. Найбільш важливим його етапом є випалювання цегли в тунельній печі.

Технологічна схема цього процесу представлена на рис. 2.1 [2].

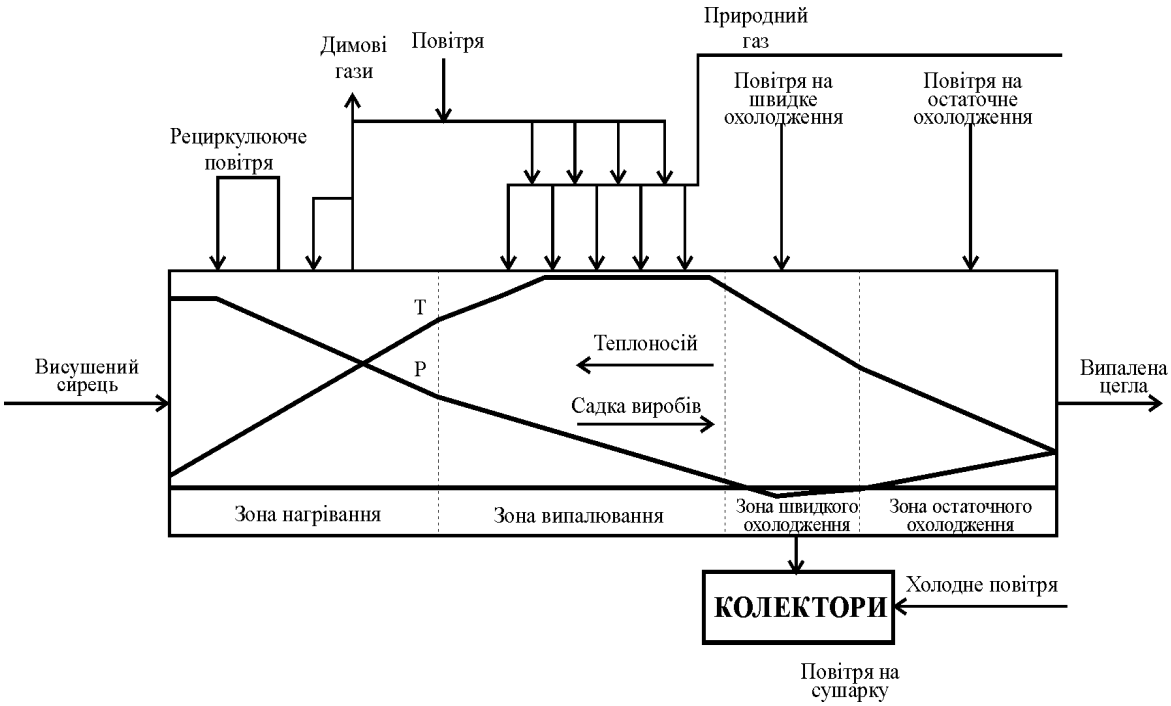


Рис. 2.1. Технологічна схема процесу випалювання керамічної цегли

Тунельна піч є складним багатофакторним об’єктом керування із значною кількістю матеріальних потоків, властивості якого розподілені за всіма координатами, але найбільш суттєво – по довжині. Керування окремими процесами, які мають місце у печі (нагрівання, випалювання, охолодження матеріалу, згоряння палива тощо), складно здійснювати через відсутність інформації про стан виробів впродовж усього часу, що цеглини знаходяться в середині корпусу печі, а це більше двох діб.

Через ці причини кожне порушення, яке може статися в печі, призводить до тривалого пошуку і усунення недоліків і суттєвих матеріальних збитків.

Запропонована в лабораторній роботі експертна система розрахована на функціонування в структурі системи автоматизації для виконання наступних задач:

1) діагностування причин аварійних ситуацій;
2) прогнозування можливих порушень нормального режиму роботи.

Головною складовою ЕС є база знань (БЗ). Інформація, яку закладено у БЗ, є переліком фактів та правил використання цих фактів (правил продукції). У розглянутій БЗ фактами є можливі порушення процесу випалювання, викликані різноманітними причинами.

Для кращого сприйняття цієї інформації створено семантичну мережу порушень, фрагмент якої зображено на рис. 2.2 [2].

Пересування по мережі згори в низ дозволяє виконувати пошук причин порушень технологічного процесу, тобто ***діагностування порушень.***

Пересування у протилежному напрямку – знизу на гору, дає можливість з’ясувати наслідки кожного порушення, тобто виконує ***прогнозування***.

У вузлах мережі знаходяться факти – можливі порушення. Дуги мережі мають смисл «може бути викликане» для діагностування причин порушень та «викликає» – при прогнозуванні.

Біля кожної дуги мережі розташована певна інформація. При назвах контрольованих змінних у квадратних дужках наведено їх менші та більші значення.

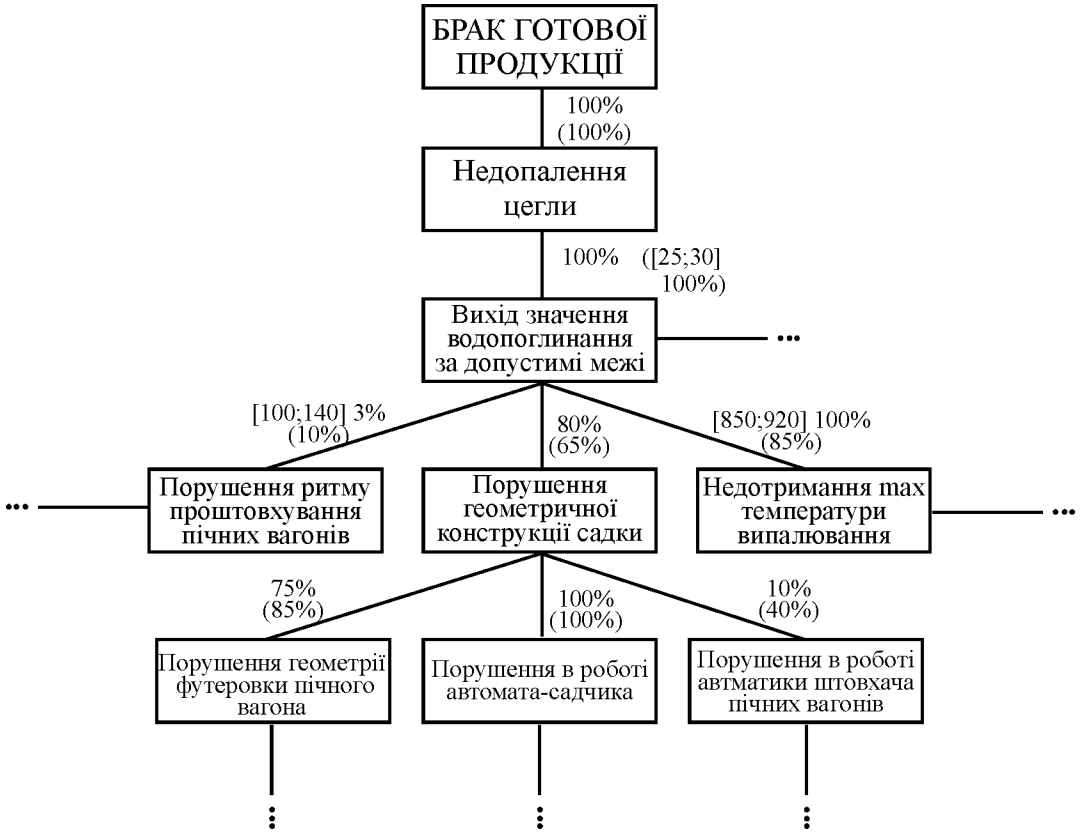


Рис. 2.2. Фрагмент семантичної мережі подання знань про процес випалювання цегли для факту «Недопалення цегли»

Якщо контрольована змінна набуває значення з цього інтервалу, то ситуація, зазначена у верхньому вузлі може викликати ситуацію з нижнього вузла. Відсотки означають імовірність виникнення порушень: вказані без дужок належать режиму діагностування, у дужках – прогнозуванню.

Приклад трактування мережі для діагностування причин порушення ***Брак продукції***.

Брак продукції з імовірністю 100 % має місце тоді, коли визнано факт ***Недопалення цегли***. А цей факт з імовірністю 100 % може бути викликаний порушенням ***Вихід водопоглинання за допустимі межі***. Слід зазначити, що ***Недопалення цегли*** викликають ще й інші порушення, які будуть досліджуватися в ході виконання лабораторної роботи. Для прикладу розглядаємо тільки мережу, зображену на рис. 2.2.

Вихід водопоглинання за допустимі межі може бути викликаний наступними подіями:

1) ***Порушенням ритму проштовхування пічних вагонів*** з імовірністю 3 %;

2) ***Порушенням геометричної конструкції садки*** з імовірністю 80 %. Ця подія, у свою чергу може бути викликана наступним:

   a) *Порушенням геометрії футеровки пічного вагона* (75 %);

   b) *Порушення в роботі автомата-садчика* (100 %);

   c) *Порушення в роботі автоматики штовхача пічних вагонів* (10 %).

3) ***Недотримання максимальної*** **(max)** ***температури випалювання*** з імовірністю 100 %.

Приклад трактування графу для прогнозування. Почнемо з нижніх вузлів мережі. ***Порушення геометрії футеровки пічного вагона*** з 85 % імовірністю викликає ***Порушення геометричної конструкції садки***.

Ця подія, в свою чергу, вплине на ***Вихід значення водопоглинання за допустимі межі*** з імовірністю 65 %, а це з імовірністю 100 % є причиною ***Недопалення цегли***.

Для взаємодії ЕС та користувача розроблений механізм виведення. Він складається з низки вікон, зокрема ***Назва та версія*** (заставка) (рис. 2.3), ***Вибір режиму*** (рис. 2.4), ***Перелік аварійних ситуацій*** найвищого рівня (рис.2.5), типових вікон ***діагностування*** (рис. 2.6) та ***прогнозування*** (рис. 2.7) нижніх рівнів.

Наведемо приклади цих вікон [2].

Рис. 2.3. Інформаційне вікно ЕС – *ExpertSys* версії 5.1.2

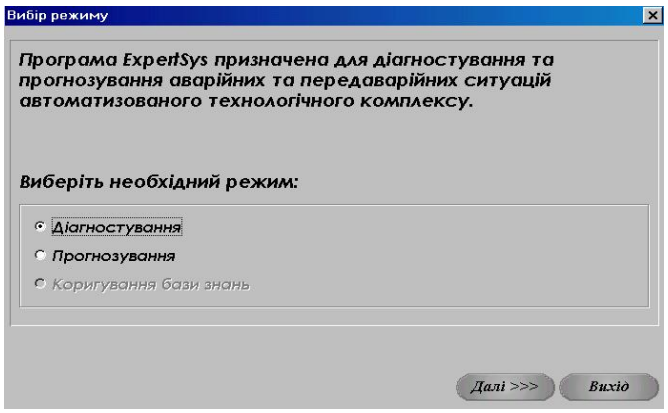


Рис. 2.4. Вікно з поясненням призначення ЕС *ExpertSys* та переліком режимів її робот

При діагностуванні причин аварії серед наданого переліку можливих причин (див. рис. 2.5) треба вибрати ту, яка дійсно відбулася, і могла спровокувати появу порушення вищого рівня. На практиці, якщо невідомо, яка саме відбулася подія з наданого списку, можна керуватися наведеними ймовірностями (див. рис. 2.6).

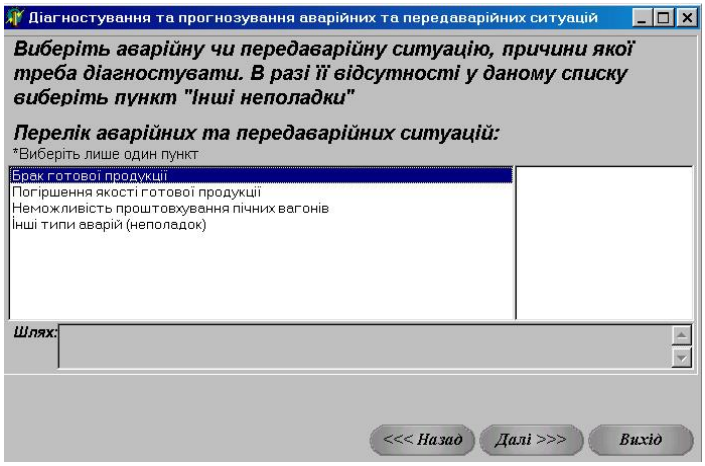


Рис. 2.5. Вікно діагностування з переліком аварійних ситуацій найвищого рівня

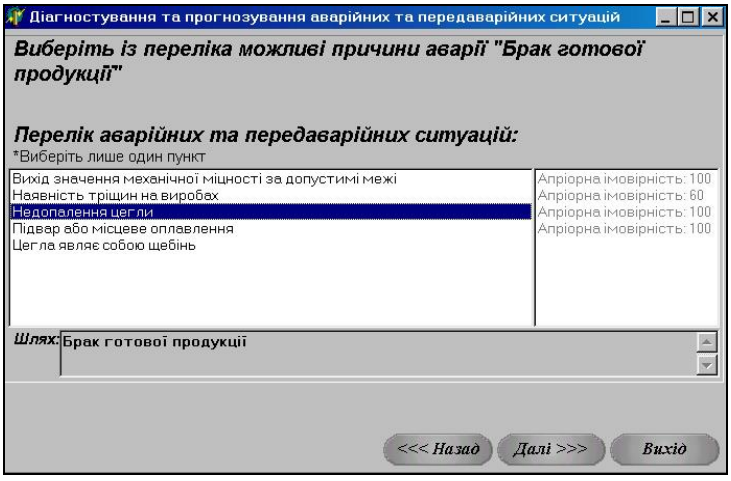


Рис. 2.6. Вікно діагностування з переліком аварійних ситуацій першого рівня

Режим прогнозування дозволяє визначити, до яких наслідків може призвести певне порушення.

Для того, щоб пришвидшити пошук актуального порушення серед значної кількості їх у БЗ, порушення умовно розділено на чотири групи (див. рис. 2.7):

- Контрольовані технологічні змінні;
- Технологічне обладнання;
- Система контролю чи керування;
- Людський фактор.

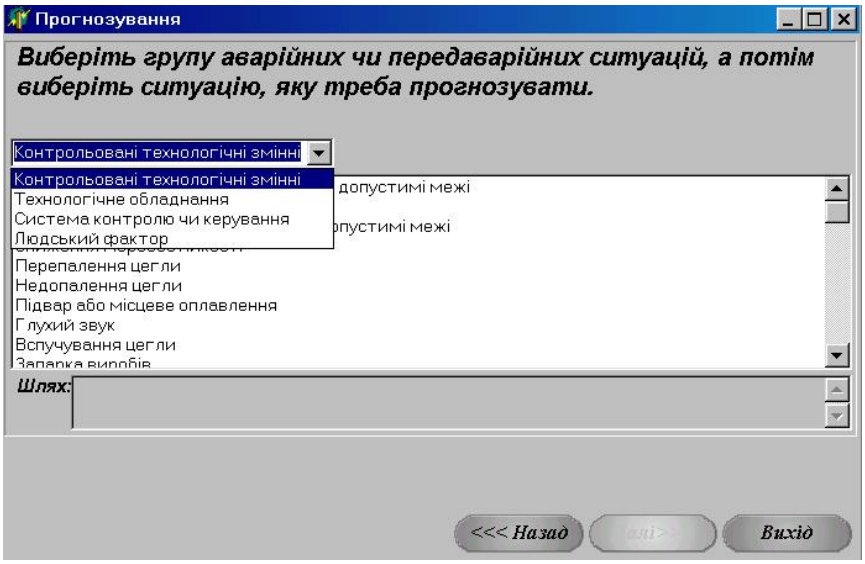


Рис. 2.7. Вікно прогнозування для вибору порушень

Усі події, які входять до мережі аварійних ситуацій є сумісними. Тому для визначення ймовірності події верхнього рівня, потрібно використовувати формулу імовірності для суми N сумісних подій нижнього рівня:

$$P(A_1 + A_2 + ... + A_N) = P(A_1) + P(A_2) + ... + P(A_n) - P(A_1A_2) - ... - P(A_{N-1}A_N) + \\ + P(A_1A_2A_3) + ... + P(A_{N-2}A_{N-1}A_N) + ... + (-1)^N P(A_1A_2...A_N),$$

де $P(A_1)$, $P(A_2)$,..., $P(A_N)$ – імовірності виникнення подій A_1, A_2, …,A_N; $P(A_1A_2)$ – імовірність сумісного виникнення двох подій A_1 та A_2; $P(A_1A_2…A_N)$ – імовірність сумісного виникнення N подій.

Ілюстрацією мережі фактів і правил, розглянутих у практикумі, може стати семантична мережа, зображена на рис. 2.8.

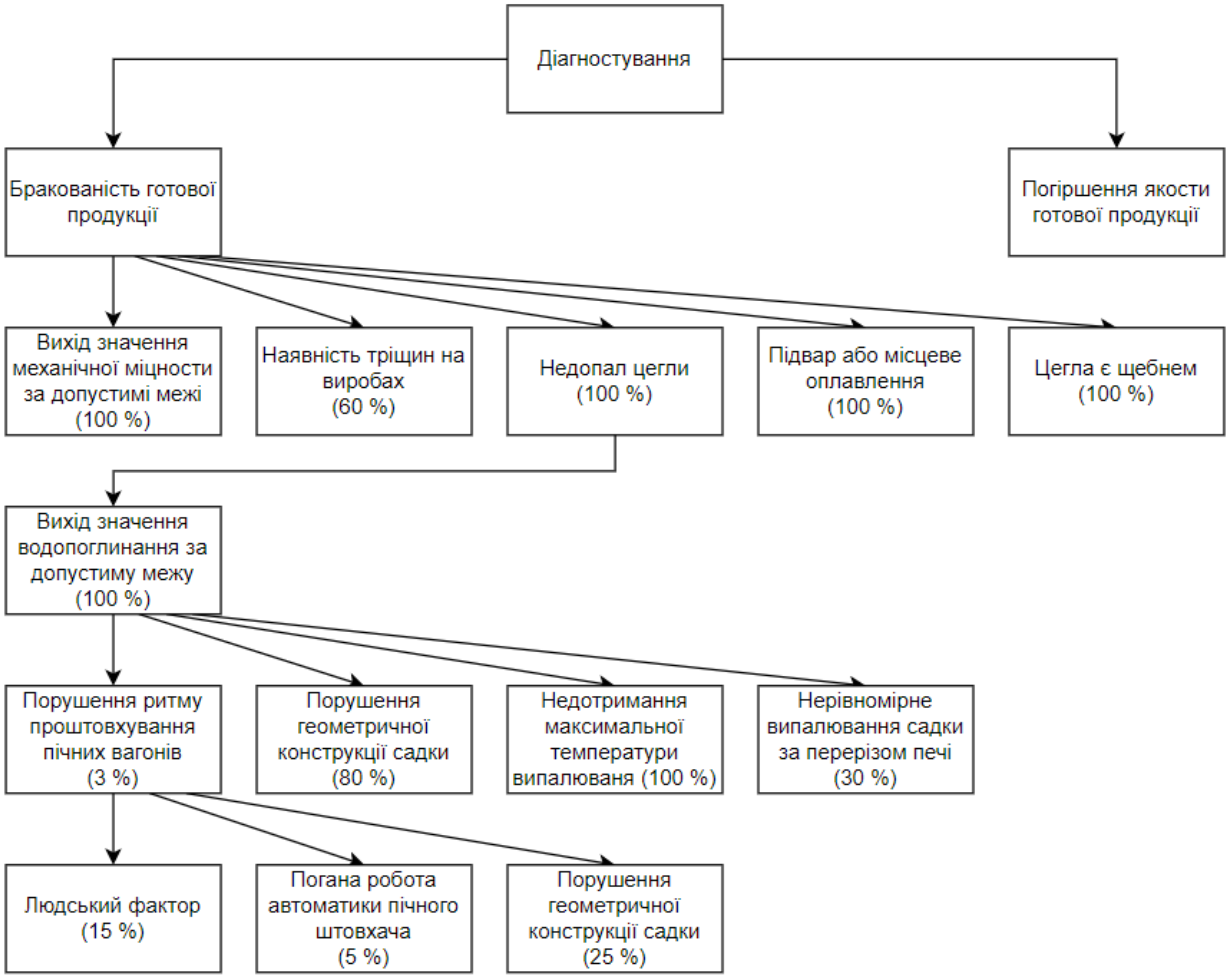


Рис. 2.8. Семантична мережа аварійних ситуацій ЕС, розглянутих у практикумі

Порядок виконання роботи

1. **Дослідити порушення процесу випалювання цегли**.

1.1. Активізувати роботу ЕС за допомогою програми, вказаної викладачем.

1.2. У вікні ***Привітання*** (рис. 2.3) натиснути кнопку **Далі>>>**.

1.3. У вікні ***Вибору режиму*** (рис. 2.4) вибрати опцію ***Діагностування***. Натиснути кнопку ***Далі***>>>.

1.4. З переліку аварійних та передаварійних ситуацій (рис. 2.5) вибрати ***Брак готової продукції***. Натиснути кнопку ***Далі***>>>.

1.5. З переліку ситуацій, що визначають брак продукції вибрати **Недопалення цегли**. Знайти всі можливі причини появи цього порушення.

1.6. Повернутися до вікна (рис. 2.5). Вибрати ***Брак готової продукції***. З переліку ситуацій, що визначають брак продукції вибрати ***Підвар або місцеве оплавлення***. Знайти всі можливі причини появи цього порушення.

1.7. Повернутись до ***Вікна вибору режиму*** (рис. 2.4) шляхом натискання кнопки <<<***Назад***.

1.8. Вибрати опцію ***Прогнозування***. Натиснути кнопку **Далі>>.**

1.9. У вікні прогнозування (рис. 2.7) розкрити меню, що випадає, а у ньому пункт ***Контрольовані технологічні змінні***. Знайти наслідки ситуацій:

- підсоси повітря в каналі печі;
- порушення аеродинамічного режиму;
- недотримання максимальної температури випалювання.

1.10. У меню, що випадає, вибрати пункт ***Система контролю чи керування***. Знайти наслідки ситуацій:

- погана робота автоматики пічного штовхача;
- незадовільна робота контуру співвідношення газ/повітря;
- незадовільна робота контуру керування температурою.

2. **Сформувати дерево аварійних ситуацій для фрагменту індивідуальної технологічної системи** (за прикладом рис. 2.8)**.**

Вміст звіту

Зображення мереж аварійних ситуацій ***Брак готової продукції*** та ***Погіршення якості готової продукції***. Приклад розрахунку ймовірності порушення верхнього рівня за результатами порушень нижнього рівня.

Фреймова структура, яку можна використати для бази знань досліджуваної експертної системи.

Контрольні запитання

1. Що таке експертна система?
2. Які функції виконує досліджена ЕС?
3. Назвіть загальні складові ЕС?
4. Які функції виконує інженер знань при створенні цієї ЕС?
5. Наведіть приклад спілкування інженера знань з експертом предметної області для отримання БЗ цієї ЕС.
6. На прикладі цієї системи поясніть роботу механізму виведення.
7. Як розрахувати ймовірність заданої викладачем події верхнього рівня за ймовірностями подій нижнього рівня?

ПРАКТИКУМ 3

СТВОРЕННЯ НЕЧІТКОЇ МОДЕЛІ ТЕХНОЛОГІЧНОГО ОБ'ЄКТА ЗАСОБАМИ *MATLAB*

Мета роботи – ознайомитись із способом створенням математичних моделей об’єкта управління на базі нечіткої логіки засобами *MATLAB*, навчитися досліджувати такі моделі.

Стислі теоретичні відомості

Нечітка логіка (*Fuzzy Logic*) **–** це надмножина класичної булевої логіки. Вона розширює можливості останньої, дозволяючи застосовувати концепцію невизначеності в логічних висновках. Апарат нечіткої логіки теж строгий і точний, як і класичний, але разом із протилежними значеннями «хибність» та «істина» він дозволяє використовувати значення, що знаходяться між ними [3 – 5].

Нечітка множина (*Fuzzy Set*) є сукупністю елементів довільної природи, щодо яких не можна з повною визначеністю стверджувати, чи належить той або інший елемент сукупності певній множині чи ні. Іншими словами, не можна чітко визначити, що даний вислів є «хибністю» або «істиною» («чорним» або «білим»), оскільки множина «чорно-біле» містить усі відтінки від чорного до білого.

Функція належності (*Membership Functions*) є апаратом для символьного (математичного) опису нечіткої множини. Вона показує цей перехід від «істини» до «хибності». Жодних обмежень на вибір конкретної функції належності немає.

Проте, на практиці зручно використовувати ті з них, які допускають аналітичне подання у вигляді простої математичної функції. Необхідність типізації окремих функцій належності обумовлена реалізацією відповідних функцій в інструментальних засобах, наприклад, у *MATLAB.*

Для реалізації процесу нечіткого моделювання в середовищі *MATLAB* призначений спеціальний пакет розширення *Fuzzy Logic Toolbox*. У рамках цього пакету користувач може виконувати необхідні дії по розробці і використанню нечітких моделей.

Застосуємо метод нечіткого моделювання для процесів сушіння та грануляції мінеральних добрив, які відбуваються в апараті «барабанний гранулятор – сушарка» (БГС). Для створення нечіткої моделі ознайомимося з технологічними особливостями об'єкта керування [2].

Упарена пульпа збирається у збірнику з мішалкою і обігрівом, звідки прямує в апарат БГС на грануляцію і сушіння.

Пульпу напилюють пневматичною форсункою на завісу сухого матеріалу (так званий ретур). Гранули укрупнюються, обкочуються і висушуються. Отримані гранули добре обкатані і містять 0,5-1 % вологи. Сушіння здійснюється газовим теплоносієм, який подають прямотечією до диспергованих часток при температурі на вході 450 – 500 С, температура газів на виході 90-100 °С.

Висушений продукт розділяють на три фракції: велику (∅ > 3,2 мм), товарну і дрібну (∅ <1 мм). Гранули великої фракції подрібнюються в лотковій дробарці, подрібнений матеріал іде знову на розсіювання разом з основним потоком матеріалу. Гранули +3,2 мм з верхнього сита надходять на валкову дробарку, а потім – знову на розсівання.

Готовий продукт з нижнього сита йде на охолоджування, а потім на склад: фракції діаметром меншим за 1 мм повертають в БГС як ретур.

Задача полягає в тому, щоб розробити систему керування процесом гранулоутворення, яка б ґрунтувалася на нечіткому моделюванні, враховуючи досвід спеціалістів з автоматизації БГС.

На перебіг процесів сушіння та гранулоутворення крім властивостей пульпи та теплоносія впливає щільність завіси ретуру перед форсункою, а також концентрація дрібної фракції на виході апарату БГС, вплив останньої буде відчутно через певний час. Кваліфіковані експерти співставляють ці дві змінні для нанесення керувального впливу.

Керувальною змінною у системі керування визначено тиск повітря у пневматичній форсунці.

З огляду на це факторами моделі визначимо 2 нечіткі **лінгвістичні змінні**: «*Концентрація дрібної фракції на виході апарату*» (далі «*Концентрація дрібної фракції*») та «*Завіса*», а за вихідну змінну приймемо нечітку **лінгвістичну змінну** «*Тиск повітря пневмофорсунки*» (далі «*Тиск пневмофорсунки*»).

За **терм-множину** першої лінгвістичної змінної «*Концентрація дрібної фракції*» візьмемо множину Т1={«*Занадто мало*», «*Мало*», «*Нормально*», «*Багато*», «*Занадто багато*»}, а за терм - множину другої лінгвістичної змінної «*Завіса*» використаємо множину Т2={«*рідка*», «*нормальна*», «*щільна*»}. За терм - множину вихідної лінгвістичної змінної «*Тиск пневмофорсунки*» візьмемо множину Т3={«*Дуже малий*», «*Малий*», *«Зменшений», «Нормальний», «Збільшений», «Великий»*, «*Дуже великий*»}. При цьому терм-множину вхідної змінної (*Концентрація дрібної фракції*) оцінюватимемо за 20 %-й шкалою, при якій значенню 0 % відповідає найменша концентрація дрібної фракції на виході БГС, а цифрі 20 %-й – найбільша. Терм-множину вхідної змінної («*Завіса*») запропоновано оцінювати за 10-бальною шкалою, при якій значенню «0» відповідає найменша щільність завіси в БГС, а значенню «10» – найбільша. Що стосується терм-множині вихідної змінної «*Тиск пневмофорсунки*», то його шкала буде проградуйована від 0,3 МПа до 0,6 МПа.

Послідовність створення моделі

Процес розробки системи і моделі нечіткого висновку в інтерактивному режимі для розглянутого вище прикладу «*Барабанний гранулятор – сушарка*» полягає у виконанні наступної послідовності дій.

1. Викликати редактор системи нечіткого висновку *FIS*, для чого у вікні команд набрати ім'я відповідної функції *fuzzy*. Після виконання цієї команди на екрані з'явиться графічний інтерфейс редактора *FIS* з ім'ям системи нечіткого висновку *Untitled* і типом системи нечіткого висновку (Мамдані), запропонованим за умовчанням.

2. Оскільки в прикладі з БГС розглянуто систему нечіткого висновку з двома входами, то необхідно додати в систему *FIS*, що розробляється, ще одну вхідну змінну. Для цього слід виконати команду меню *Edit>Add Variable... >Input.* У результаті виконання цієї команди на діаграмі системи нечіткого висновку з'явиться новий жовтий прямокутник з ім'ям другої вхідної змінної: *input2* (див рис. 3.1).

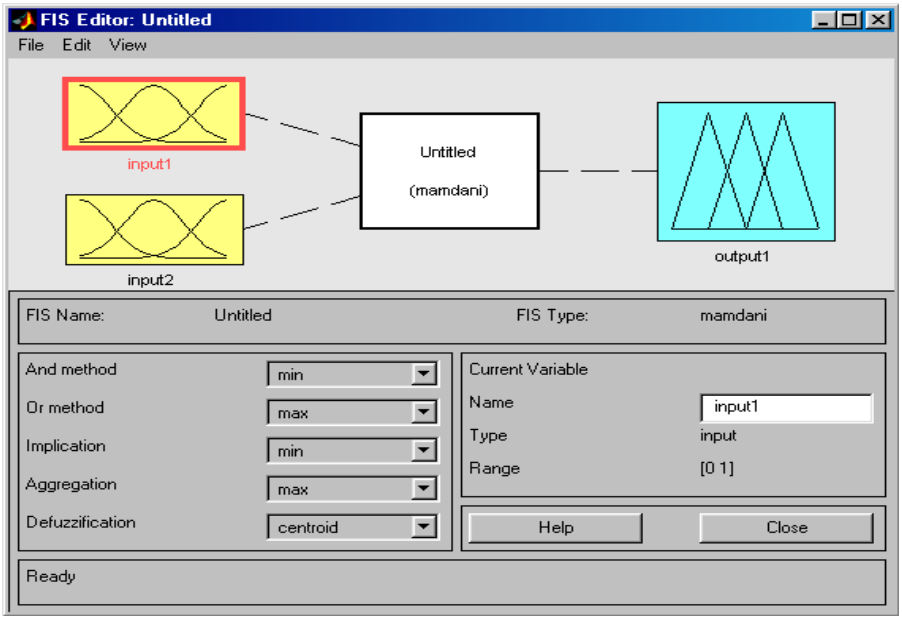


Рис. 3.1. Вид редактора *FIS* після додавання другої вхідної змінної

3. Змінимо імена вхідних і вихідних змінних, запропонованих системою *MATLAB* за умовчанням. Для цього необхідно виділити прямокутник з ім'ям відповідної змінної, виконавши клацання на його зображенні на діаграмі (сторони виділеного прямокутника мають червоний колір). Після цього слід набрати нове ім'я змінної в полі введення *Name* у правій частині редактора *FIS*. Результат надання нових імен змінним системи нечіткого висновку зображений на рис. 3.2.

Рекомендація. Назви змінних і термів доцільно робити короткими, сформованими з латинських літер. Це важливо для використання нечітких моделей в інших застосунках.

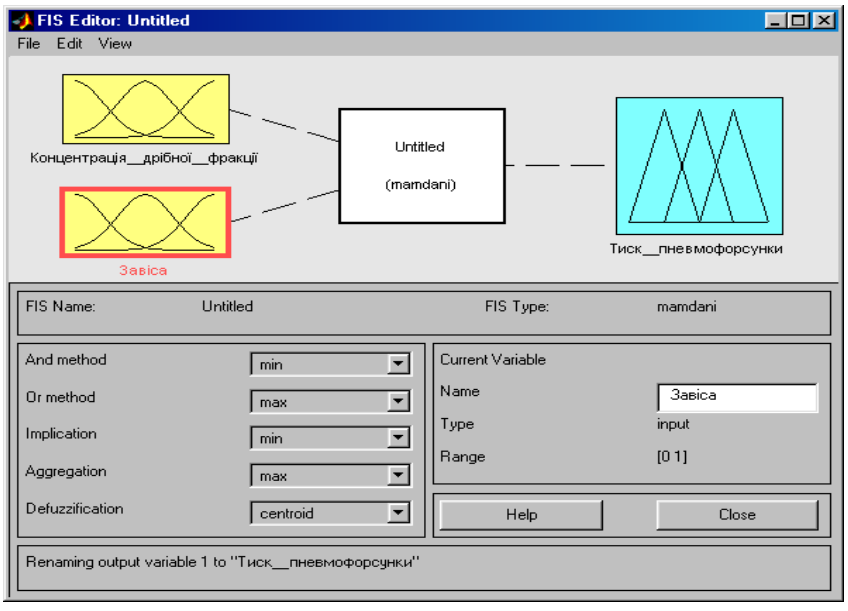


Рис. 3.2. Вид редактора *FIS* після надання нових імен змінним

Змінимо ім’я системи нечіткого висновку (*Untitled*), запропоноване за умовчанням. Для цього збережемо створену структуру *FIS* в зовнішньому файлі з ім'ям *mytip.fis*, виконавши команду меню *File>Export>To Disk...*. При цьому буде викликано стандартне діалогове вікно збереження файлу, в якому користувачу пропонують ввести ім'я відповідного файлу (розширення файлу приписується автоматично). **Студентам треба назвати файл не “*mytip.fis*”, а за шаблоном «Прізвище_Група»** (наприклад, Труш_ЛК-91).

Залишимо без зміни запропоновані системою *MATLAB* за умовчанням: метод нечіткого логічного **І** (*And method*) – значення «*min*», метод нечіткого логічного **АБО** (*Or method*) – значення «*max*», метод імплікації (*Implication*) – значення «*min*», метод агрегації (*Aggregation*) – значення «*max*» і метод дефаззифікації (*Defuzzification*) – значення «*centroid*». Очевидно, що ці значення можуть бути змінені користувачем.

4. Тепер необхідно визначити терми і їх функції належності для вхідних і вихідних змінних нашої системи нечіткого висновку. Для цієї мети слід скористатися редактором функцій належності, який може бути викликаний одним з наступних способів:

– подвійним клацанням ЛКМ на значку прямокутника з ім'ям відповідної змінної;

– командою меню *Edit>Membership Functions...* (заздалегідь повинен бути виділений прямокутник з ім'ям відповідної змінної).

Після виклику редактора функцій належності кожної із змінних за умовчанням пропонується 3 терми з трикутними функції належності (рис.3.3).

Спочатку змінимо діапазон значень вхідних змінних, для чого в полях введення *Range* і *Display Range* змінимо верхнє значення з 1 на 20 %. Аналогічно виконуються зміни відповідних діапазонів для вхідної змінної «*Завіса*» [0 – 10] і для вихідної змінної «*Тиск пневмофорсунки*» [0.3 0.6]. Зміни підтверджуються натисканням на клавішу *Enter* на клавіатурі.

Оскільки для вибраної задачі необхідно 5 термів, а система *MATLAB* пропонує за умовчанням тільки 3. Командою меню *Edit>Remove All MFs* видаляємо запропоновані терми. Командою меню *Edit>Add MFs…* створюємо 5 нових термів. (рис. 3.4).

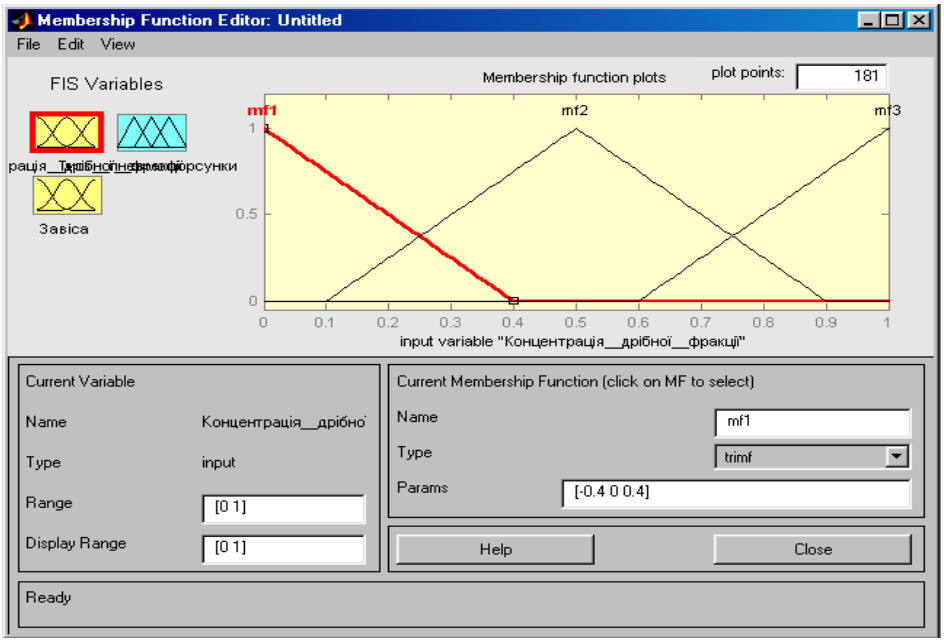


Рис. 3.3. Вікно редактора функцій належності з функціями належності для термів змінної «*Концентрація дрібної фракції*», запропонованих системою *MATLAB* за умовчанням

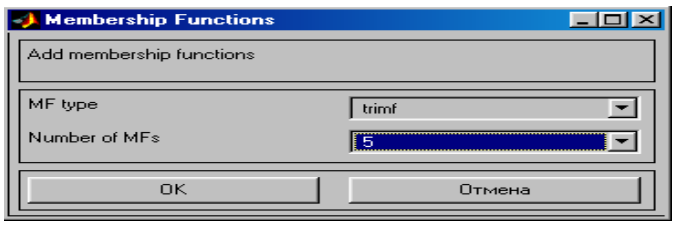


Рис.3.4. Вікно створення необхідної кількості термів

Далі слід змінити назви термів першої вхідної змінної «*Концентрація дрібної фракції*» з *mf*1, *mf*2, *mf*3, *mf*4, *mf*5 на «*Занадто мало*», «*Мало*», «*Нормально*», «*Багато*», «*Занадто багато*» відповідно. Після цього змінимо типи крайніх функцій належності, запропонованих за умовчанням, на трапецієвидні функції (*trapmf*), вибравши відповідний пункт в полі *Type*.

Параметри функцій задамо таким чином: для терма «Занадто мало» задамо параметри [0 0 2 6], для терма «Мало» – [2 6 10], для терма «Нормально» – [6 10 14], для терма «Багато» – [10 14 18], для терма «Занадто багато» – [14 18 20 20]. Вигляд редактора функцій належності після внесених змін зображений на рис.3.5.

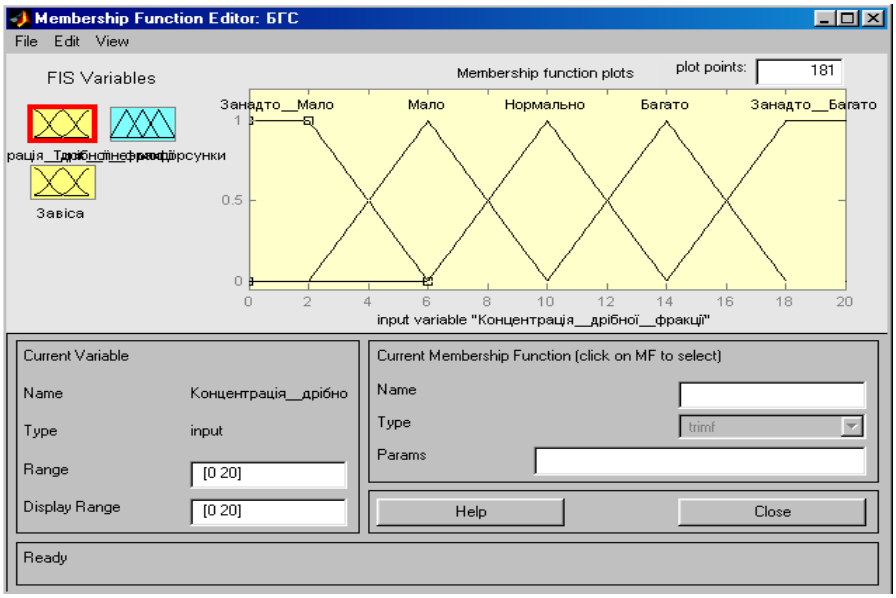


Рис.3.5. Вигляд редактора функцій належності після зміни назви термів і типу функцій належності для вхідної змінної «*Концентрація дрібної фракції*»

Аналогічним чином змінимо назви термів другої вхідної змінної «*Завіса*». Після чого змінимо, запропонований за умовчанням, тип функцій належності на функції типу Гаусса (*gaussmf*), вибравши відповідний пункт в полі *Type*. Параметри функцій належності залишимо без змін. За терм-множину другої лінгвістичної змінної «*Завіса*» використано «*Рідка*», «*Нормальна*», «*Щільна*». Вигляд редактора функцій належності після внесених змін зображений на рис. 3.6.

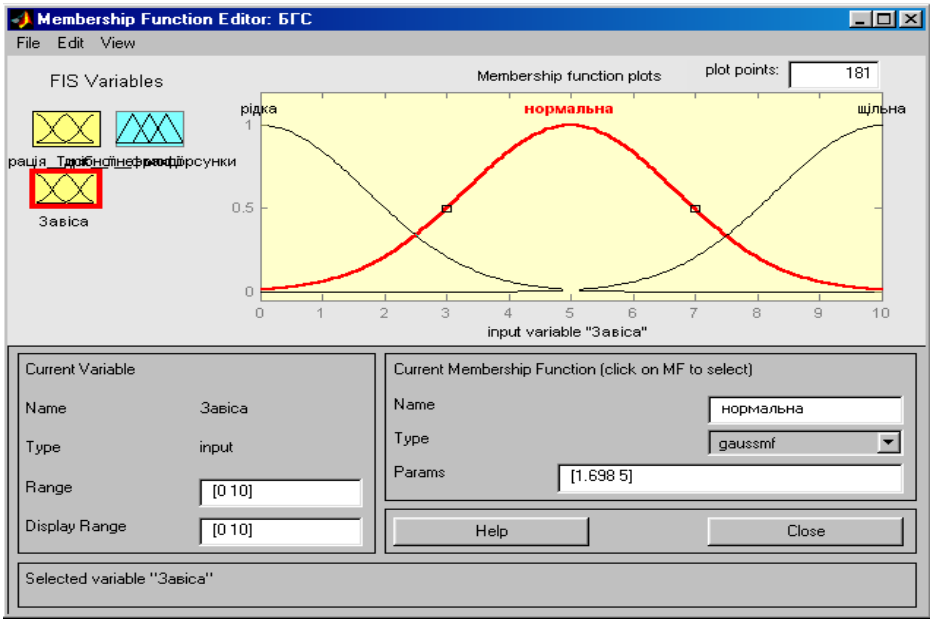


Рис.3.6. Вигляд редактора функцій належності після зміни назви термів і типу їх функцій належності для вхідної змінної «*Завіса*»

Нарешті, змінимо назви термів і параметри функцій належності для вихідної змінної «*Тиск пневмофорсунки*». Нам необхідні 7 термів, а система *MATLAB* пропонує за умовчанням тільки 3. Командою меню *Edit>Remove All MFs* видаляємо запропоновані терми. Командою меню *Edit>Add MFs*_створюємо 7 нових термів. Далі змінимо назви термів вихідної змінної «*Тиск пневмофорсунки*» з *mf*1, *mf*2, *mf*3, *mf*4, *mf*5, *mf*6, *mf*7 на «*Дуже малий*», «*Малий*», «*Зменшений*», «*Нормальний*», «*Збільшений*», «*Великий*», «*Дуже великий*» відповідно. Після чого змінимо типи крайніх функцій належності, запропонованих за умовчанням, на трапецієвидні функції (*trapmf*), вибравши відповідний пункт в полі *Type*.

Параметри функцій задамо таким чином: для терма «*Дуже малий*» задамо параметри [0.3 0.3 0.3283 0.3667], для терма «*Малий*» – [0.3283 0.3667 0.4083], для терма «*Зменшений*» – [0.3667 0.4083 0.45], для терма «*Нормальний*» – [0.4083 0.45 0.4917], для терма «*Збільшений*» – [0.45 0.4917 0.5333], для терма «*Великий*» – [0.4917 0.5333 0.5716], для терма «*Дуже великий*» – [0.5333 0.5716 0.6 0.6]. Вид редактора функцій належності після внесених змін зображений на рис. 3.7.

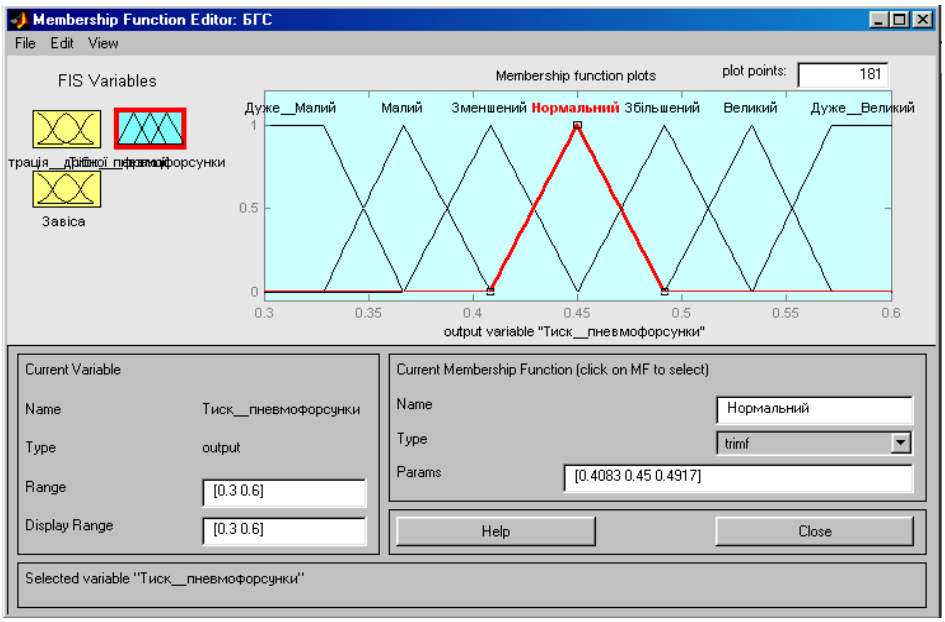


Рис. 3.7. Редактора функцій належності після зміни назви термів і типу функцій належності для вихідної змінної «*Тиск пневмофорсунки*»

Порядок виконання роботи

1. Створити модель процесу гранулоутворення мінеральних добрив у БГС за методикою, описаною в роботі.

2. Розробити математичну модель на базі нечіткого моделювання для власного технологічного об’єкта керування.

3. Дослідити типи функцій належності, які надає пакет, і способи їх математичного подання.

Вміст звіту

Зображення вікон з функціями належностей лінгвістичних змінних для моделі БГС (рис. 3.5 – 3.7 з ідентифікацією студента за допомогою **Прізвище_Група**).

Структурно-параметрична схема для власного об’єкта керування з назвами вхідних та вихідної змінних моделі. Опис лінгвістичних змінних, які використані у власній моделі процесу, вікна редактора *MATLAB* із зображеннями та характеристиками функцій належності цих змінних.

Контрольні запитання

1. Як описати лінгвістичну змінну?
2. Як описати нечітку змінну?
3. Які існують види функцій належності?
4. Як додати вхідну змінну в редакторі пакету *Fuzzy Logic*?
5. Як надати лінгвістичній змінній певне ім’я?
6. Як задати терми і функції належності лінгвістичної змінної?
7. Як змінити кількість термів?
8. Як змінити тип функції належності?
9. Як задати потрібний діапазон значень лінгвістичної змінної?

ПРАКТИКУМ 4

СТВОРЕННЯ ТА ДОСЛІДЖЕННЯ АВТОМАТИЧНОЇ СИСТЕМИ КЕРУВАННЯ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ

Мета роботи – навчитися створювати системи керування на основі нечітких множин і нечіткої логіки (нечіткі автоматичні системи керування, НчАСК), реалізовувати і досліджувати їх засобами *MATLAB.*

Стислі теоретичні відомості

База правил систем нечіткого висновку призначена для формального представлення емпіричних знань або знань експертів у тій або іншій проблемній області. У системах нечіткого висновку використовують правила нечітких продукцій, в яких умови і висновки сформульовані в термінах нечітких лінгвістичних висловів. Сукупність таких правил далі називатимемо базами правил нечітких продукцій [3 – 5].

База правил нечітких продукцій є кінцевою множиною правил нечітких продукцій, узгоджених відносно лінгвістичних змінних, що використовуються в них. Найчастіше базу правил подають у наступній формі структурованого тексту:

***ПРАВИЛО*_1**: **ЯКЩО** «Умова_1» **ТО** «Висновок_1» (F_1)
***ПРАВИЛО*_2**: **ЯКЩО** «Умова_2» **ТО** «Висновок _2» (F_2)
...
ПРАВИЛО_*n*: **ЯКЩО** «Умова_*n*» **ТО** «Висновок _*n*» (F_n)
або в еквівалентній формі у *MATLAB*:
RULE_1: ***IF*** *Condition*_1 ***THEN*** *Conclusion*_ 1 (F_1)
RULE_2: ***IF*** *Condition*_2 ***THEN*** *Conclusion*_ 2 (F_2)
...
RULE_*n*: ***IF*** *Condition_n* ***THEN*** *Conclusion_ n* (F_n)

Тут через F_i ($i \in \{1, 2 ..., n\}$) позначені *коефіцієнти визначеності* або вагові коефіцієнти відповідних правил. Ці коефіцієнти можуть приймати значення з інтервалу [0,1]. У випадку, якщо ці вагові коефіцієнти відсутні, зручно прийняти, що їх значення дорівнюють 1.

У системах нечіткого висновку лінгвістичні змінні, які використовуються в нечітких висловах підумов правил нечітких продукцій, часто називають *вхідними лінгвістичними змінними*, а змінні, які використовуються в нечітких висловах підвисновків правил нечітких продукцій, часто називають *вихідними лінгвістичними змінними*.

Таким чином, при формуванні бази правил необхідно визначити множину правил нечітких продукцій: $P=\{R_1, R_2, ..., R_n\}$, множину вхідних лінгвістичних змінних: $V=\{\beta_1\ \beta_2\ .\ \beta_n\}$, множину вихідних лінгвістичних змінних: $W=\{\omega_1, \omega_2, \ldots, \omega_n\}$. Тим самим база правил нечітких продукцій вважається заданою, якщо задані множини *P, V, W*.

Нагадаємо, що вхідна $\beta_i \in V$ або вихідна $\omega_i \in W$ лінгвістична змінна вважається заданою або визначеною, якщо для неї визначена базова терм-множина з відповідними функціями належності кожного терма, а також дві процедури. Найпоширенішим випадком є використання як функції належності термів трикутних або трапецієвидних функцій належності, розглянутих вище. При цьому для зручності запису застосовують спеціальні скорочення для найменувань окремих термів вхідних і вихідних лінгвістичних змінних.

Для формування бази правил в інтерактивному режимі в системі *MATLAB* слід скористатися редактором правил, який може бути викликаний одним з наступних способів:

– подвійним клацанням на значку квадрата в центрі з ім'ям створюваної системи нечіткого висновку (*myfis*);

– командою меню *Edit* > *Rules*.

Оскільки спочатку база правил нечіткого висновку порожня, то після виклику редактора правил центральне багаторядкове поле введення не містить жодних правил. Для їх визначення слід використовувати поля меню і перемикачі в нижній частині графічного інтерфейсу редактора правил. Для задання першого правила слід задати у полі першої вхідної змінної (*Концентрація дрібної фракції*) терм «нормальна», для другої вхідної змінної (*Завіса*) теж терм «нормальна», для вихідної змінної (*Тиск пневмофорсунки*) – «нормальний». Далі слід перемикач *Connection* поставити в положення *and* (логічне **І**) і натискати на кнопку *Add rule*. Після цього перше правило з символами кирилиці відобразиться у верхньому рядку вікна.

Таким же чином задають друге правило, для якого слід виділити імена термів відповідно «мала», «нормальна» і «збільшений», і третє правило з іменами термів «велика», «нормальна» і «збільшений» і так далі як показано на рис. 4.1.

Треба зауважити, що в полі введення *Weight* відображається вага кожного правила, яку можна змінювати в межах інтервалу [0, 1] (залишимо без зміни його значення за умовчанням, яке дорівнює 1 для всіх правил). Ця ж вага правил записується в круглих дужках у вікні правил після кожного з правил нечіткого висновку.

Після завдання правил нечіткого висновку виявляється можливим отримати результат нечіткого висновку (значення вихідної змінної) для конкретних значень вхідних змінних. З цією метою необхідно відкрити програму перегляду правил командою меню *View > Rules*.

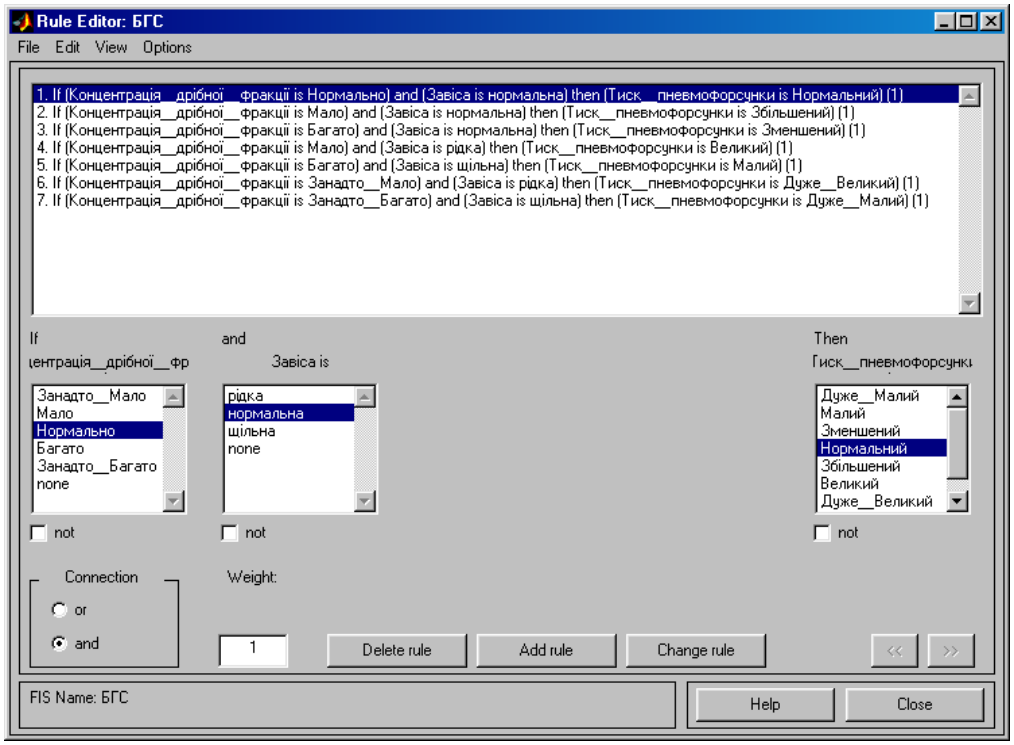


Рис. 4.1. Вікно редактора правил нечіткого висновку після їх визначення

Вигляд редактора правил після їх визначення для експертної системи, що розробляється, зображений на рис. 4.2.

Після виклику програми перегляду правил для нашої системи нечіткого висновку за умовчанням для вхідних змінних запропоновані середні значення з інтервалу їх допустимих значень (значення [10 5] в полі введення *Input*). Це означає, що концентрація дрібної фракції оцінюється експертом як нормальна і щільність завіси теж нормальна.

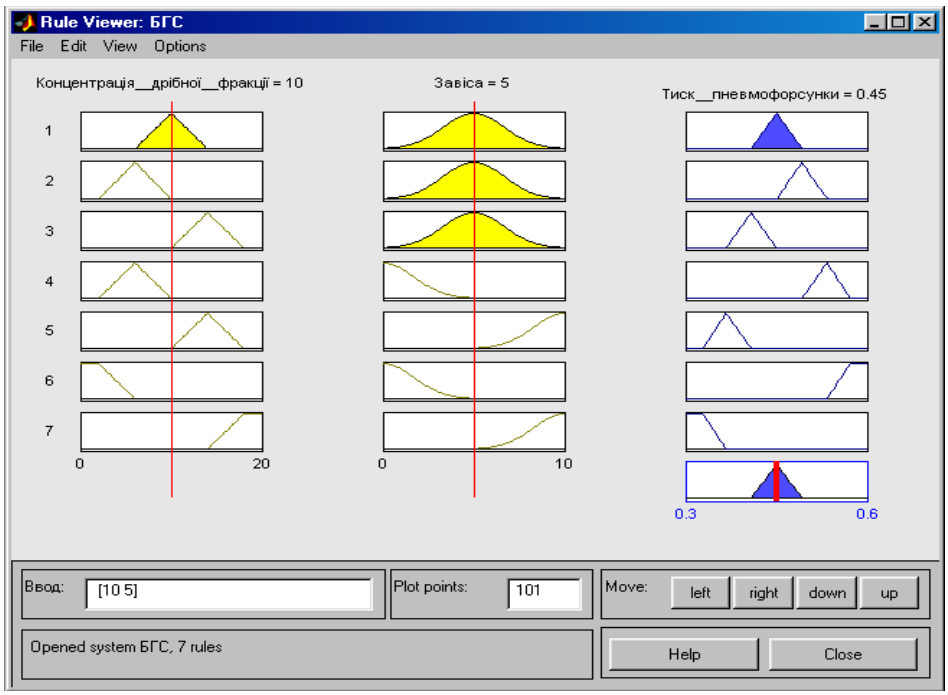


Рис. 4.2. Вікно програми перегляду роботи системи нечіткого висновку для значень вхідних змінних [10 5]

Цим значенням вхідних змінних відповідає значення тиску пневмофорсунки в 0,45 МПа, яке відображається вище за прямокутники правил в правій частині вікна програми перегляду.

Змінимо значення вхідних змінних для іншого випадку, якому відповідає концентрація фракції в 6 балів (мало дрібній фракції) і густину завіси в 5 балів (нормальна).

Для цього курсор миші перемістимо в полі введення *Input* і введемо відповідні значення вхідних змінних: [6 5]. Система *MATLAB* виведе значення тиску пневмофорсунки 0,492 МПа.

Оскільки процес нечіткого моделювання припускає аналіз результатів нечіткого висновку при різних значеннях вхідних змінних з метою встановлення адекватності розробленої нечіткої моделі, розглянемо й інші випадки. Припустимо, що концентрація дрібної фракції оцінюється в 14 балів (багато дрібної фракції), а щільність завіси – в 7 балів (густа, але не дуже). Уведемо відповідні значення змінних. Тепер розроблена система нечіткого висновку рекомендує значення тиску пневмофорсунки, що дорівнює 0,397 МПа.

Якщо ж припустити, що дрібної фракції як і раніше багато (14 балів), а щільність завіси стала нормальною і оцінюється в 5 балів, то величина тиску пневмофорсунки істотно зміниться і дорівнюватиме 0,408 МПа.

Якщо експертам така система видасться неадекватною, то можна змінити існуючі правила або додати нові, а також змінити параметри функцій належності вхідних і вихідний змінних.

Для остаточного аналізу розробленої нечіткої моделі може виявитися корисною програма перегляду поверхні нечіткого висновку, яка може бути викликана командою меню *View* > *Surface* редактора *FIS*.

Графічний інтерфейс програми перегляду поверхні нечіткого висновку для розробленої нечіткої моделі зображений на рис. 4.3.

Ця програма використовується для загального аналізу адекватності нечіткої моделі, дозволяючи оцінити вплив зміни значень вхідних нечітких змінних на значення однієї з вихідних нечітких змінних. У разі потреби можна отримати графік залежності вихідної змінної від однієї з вхідних змінних. Для цього необхідно вибрати потрібну змінну в списку, що розкривається, *X* (*input*), а в списку, що розкривається, *Y* (*input*) вибрати значення *-none-*.

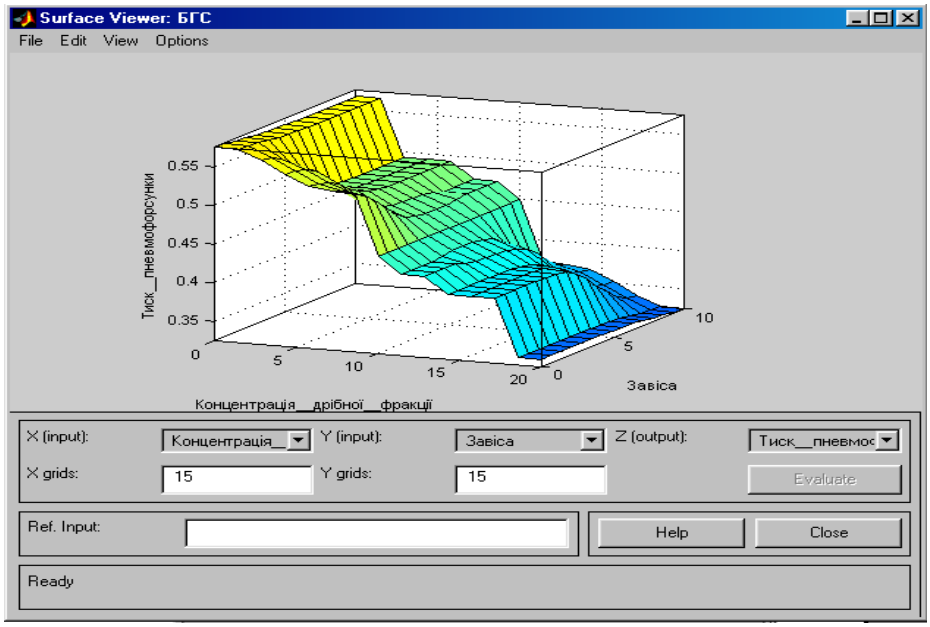


Рис. 4.3. Вікно перегляду поверхні нечіткого висновку для розробленої нечіткої моделі

На формування бази правил систем нечіткого висновку часто впливають деякі додаткові чинники, які визначаються специфікою вирішуваної задача або алгоритму нечіткого висновку, що використовується. В основному ж для формування бази правил систем нечіткого висновку необхідно заздалегідь визначити вхідні і вихідні лінгвістичні змінні об'єкту керування, а також їх зв'язки і ступені впливу вхідних змінних на вихідні.

Порядок виконання роботи

1. Виконати проєктування НчАСК процесом гранулоутворення мінеральних добрив у БГС за методикою, описаною в роботі. Передбачити ідентифікацію студента за допомогою «**Прізвище_Група»** у назвах вікон чи їх елементів.

2. Виконати проєктування НчАСК для власного технологічного об’єкта керування. Передбачити ідентифікацію студента за допомогою «**Прізвище_Група»** у назвах вікон чи їх елементів.

3. Дослідити вплив кількості термів вхідної лінгвістичної змінної, типу та параметрів середнього терму цієї змінної на значення розрахованої керувальної змінної.

Вміст звіту

Перелік правил продукції та вікно *MATLAB* з внесеними правилами для автоматизації БГС. Вікна *MATLAB*, які відображають роботу нечіткої системи (рис. 4.2 та 4.3) для НчАСК процесом гранулоутворення мінеральних добрив у БГС.

Схема власного контуру керування. Вікна з функціями належностей лінгвістичних змінних. Перелік правил продукції та вікно *MATLAB* з внесеними правилами. Вікна *MATLAB*, які відображають роботу нечіткої системи (аналоги рис. 4.2 та 4.3). Вікна з результатами дослідження впливу кількості та характеристик термів на рекомендоване значення керувальної змінної. Висновки по дослідженню п.5 «Порядку виконання роботи».

Контрольні запитання та завдання

1. Як задати правила нечіткого висновку у системі *MATLAB*?
2. Як переглянути задані правила?
3. Пояснити зображення поверхні нечіткого висновку для розробленої нечіткої моделі та бази правил.
4. Як визначити значення керувального впливу при заданих значеннях вхідних змінних?

ПРАКТИКУМ 5

СТВОРЕННЯ ТА ДОСЛІДЖЕННЯ НЕЧІТКОЇ СИСТЕМИ КЕРУВАННЯ ЗАСОБАМИ *MATLAB + SIMULINK*

Мета роботи – навчитися створювати та досліджувати нечіткі системи керування технологічними об’єктами засобами *MATLAB + SIMULINK*.

Стислі теоретичні відомості

Пакет *Simulink* є ядром інтерактивного програмного комплексу, призначеного для математичного моделювання систем та пристроїв, поданих своєю функціональною блок – схемою (так звана *S - модель* або просто *модель).*

Програмний засіб *Simulink* належить до візуально – орієнтованих мов програмування. Це означає, що на всіх етапах моделювання користувач практично не виконує традиційного програмування. Моделювання в *Simulink* полягає у складанні схем з функціональних блоків. При цьому автоматично генерується програма в кодах в залежності від складу вибраних блоків, їх з’єднань і параметрів.

Для формування моделі в середовищі *Simulink* треба відкрити вікно моделі. Це виконують командами *File→New→Model* (Файл→Новий→Модель), за допомогою піктограми на панелі інструментів *New Model* (Нова модель) чи «гарячими» клавішами <*Ctrl + N*>. На рис. 5.1 наведено вікно моделі *Simulink.*

Вікно моделі з’являється також при завантаженні існуючої моделі *File→Open…* (Файл→Відкрити…), за допомогою піктограми на панелі

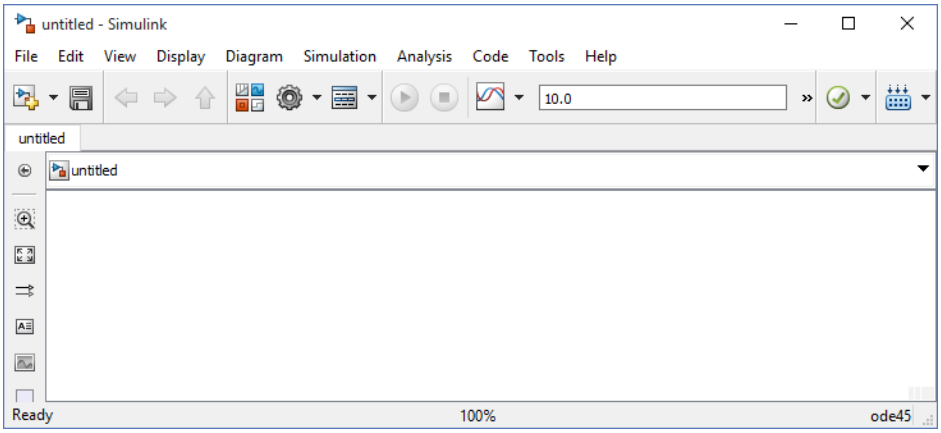


Рис. 5.1. Вікно моделі *Simulink*

Для розташування блоків у вікні моделі треба відкрити відповідний розділ бібліотеки – *Fuzzy Logic Toolbox* (рис. 5.2).

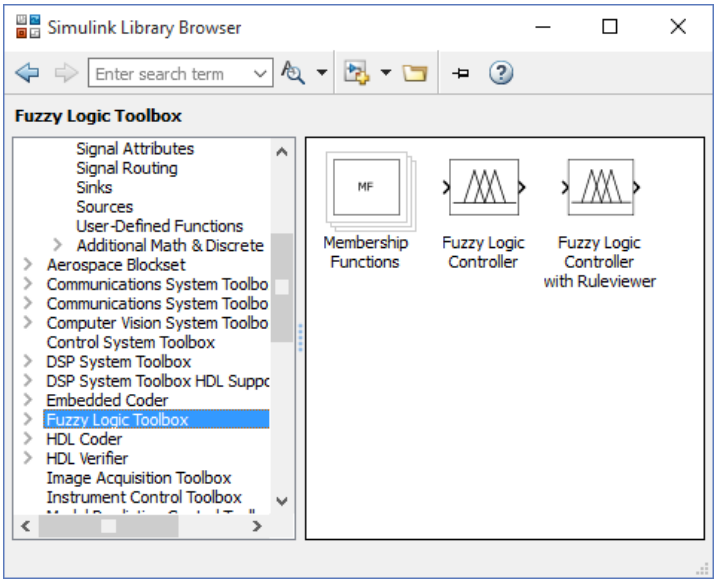


Рис. 5.2. Вікно розділу *Fuzzy Logic Toolbox* з блоками

Крім блоків цього розділу для створення моделі системи керування і її дослідження треба використати наступні розділи: *Continuous* – для моделювання каналів впливу (рис. 5.3), *Sinks* – для моделювання вимірювальних пристроїв (рис. 5.4), *Sources* – для моделювання вхідних сигналів (рис. 5.5).

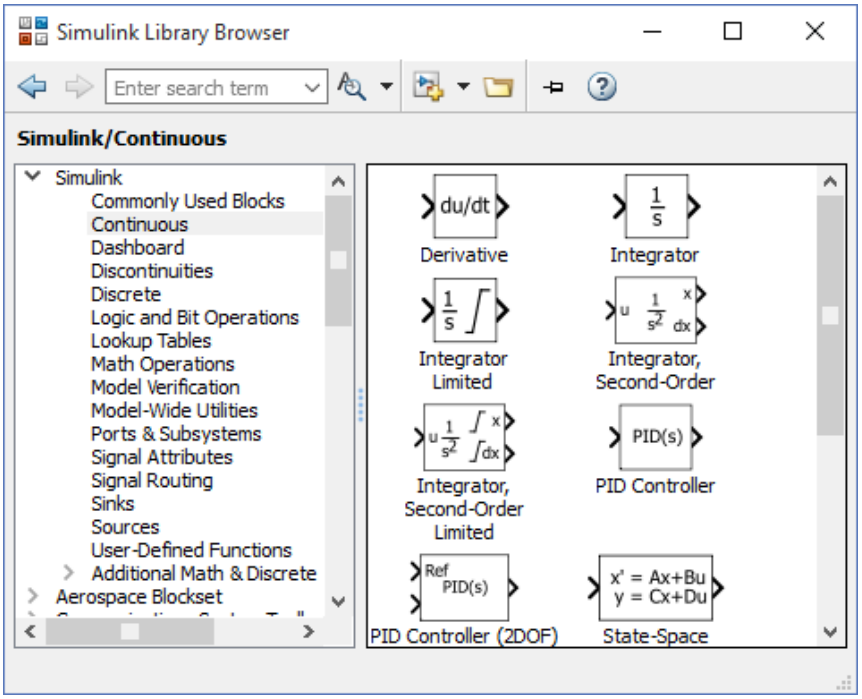


Рис. 5.3. Вікно розділу *Continuous* з блоками

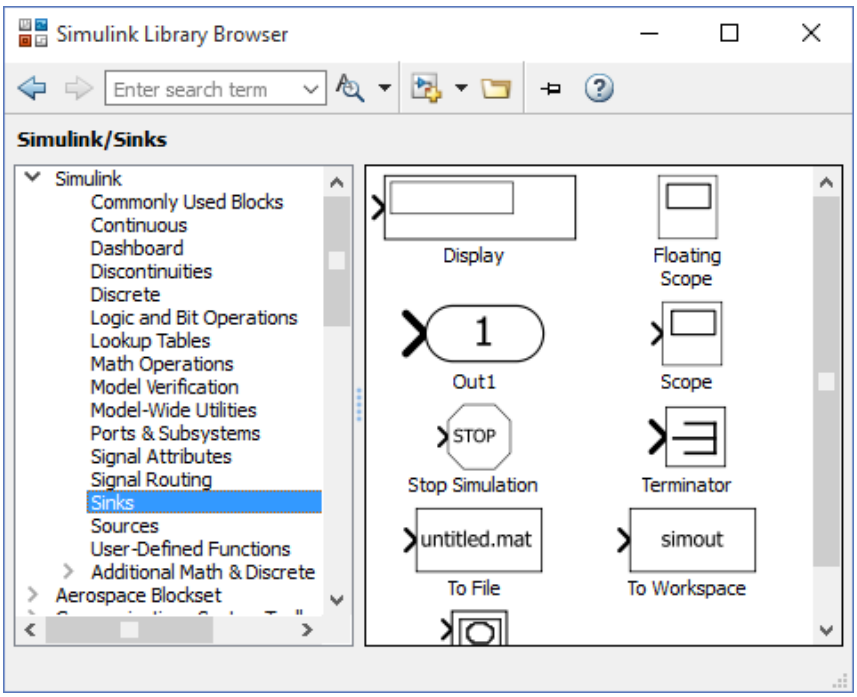


Рис. 5.4. Вікно розділу *Sinks* з блоками

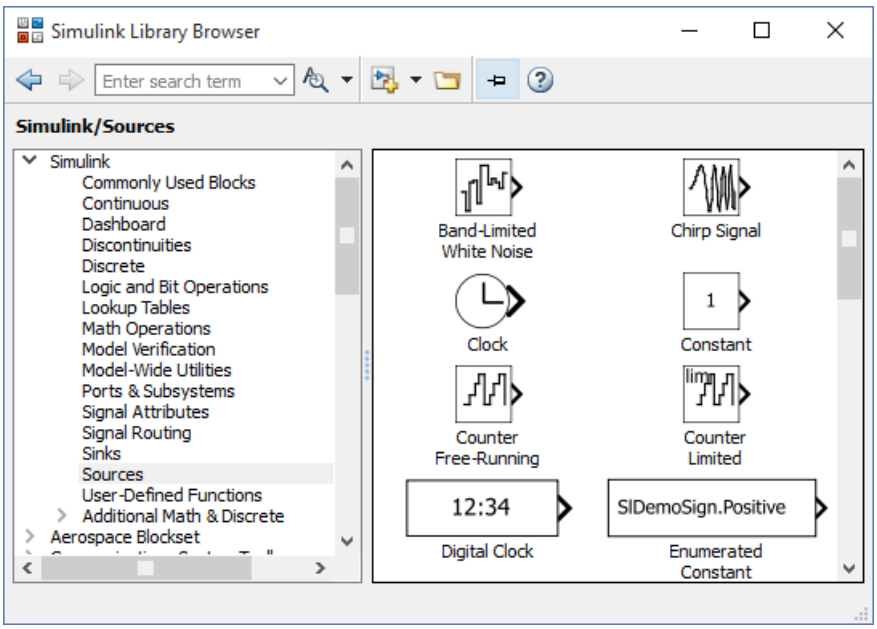


Рис. 5.5. Вікно розділу *Sources* з блоками

Позначивши потрібний блок курсором та натиснувши на ліву кнопку «миші», перетягують блок у вікно моделі. На рис. 5.6 зображено вікно моделі з трьома блоками.

Для встановлення потрібних параметрів блока (кожний з блоків завжди має певні значення параметрів, встановлені системою) потрібно двічі клацнути ЛКМ на його зображенні.

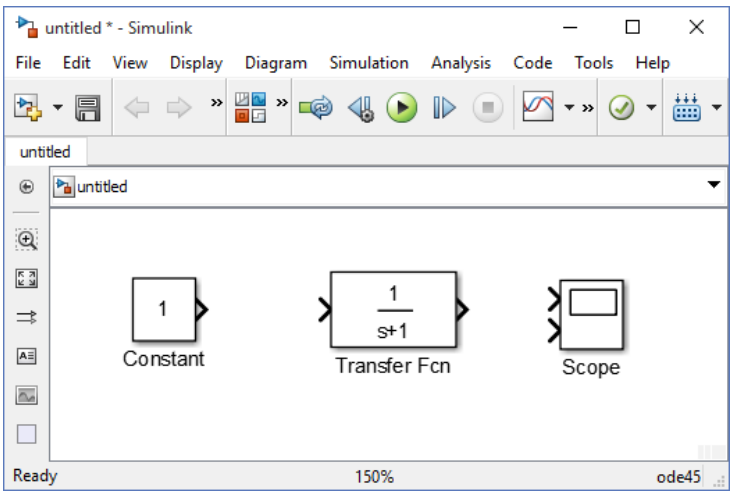


Рис.5.6. Вікно моделі з блоками

Для встановлення потрібних параметрів блока (кожний з них завжди має певні значення параметрів, встановлені системою) потрібно двічі клацнути ЛКМ на його зображенні.

У результаті з'являється вікно редагування параметрів (розділовим знаком для дробових величин є крапка). На рис. 5.7 подано приклад редагування параметрів передавальної функції відповідного блока.

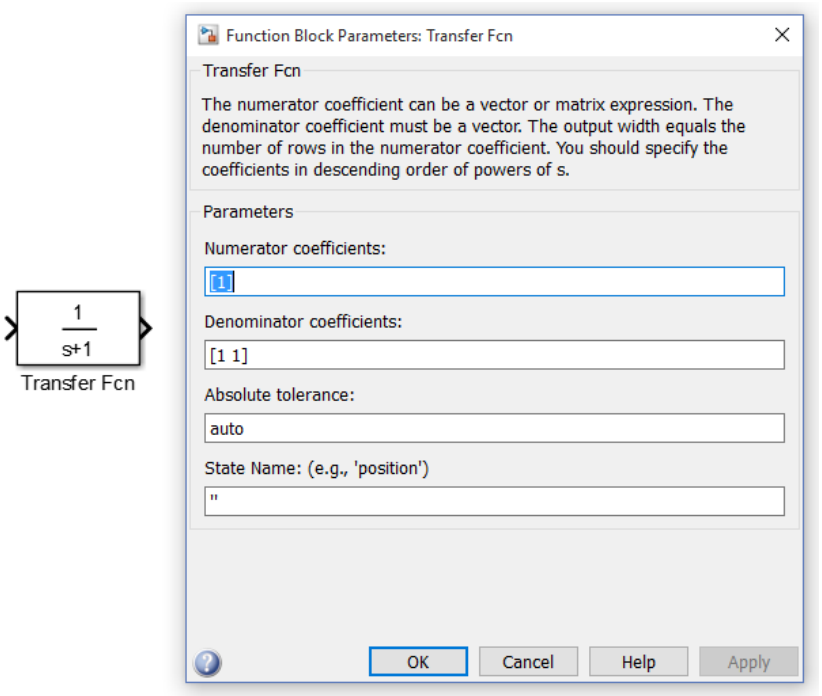


Рис. 5.7. Вікно редагування параметрів передавальної функції

Для створення нечіткої системи керування у робочому полі *Simulink* складаємо схему зображену на рис. 5.8.

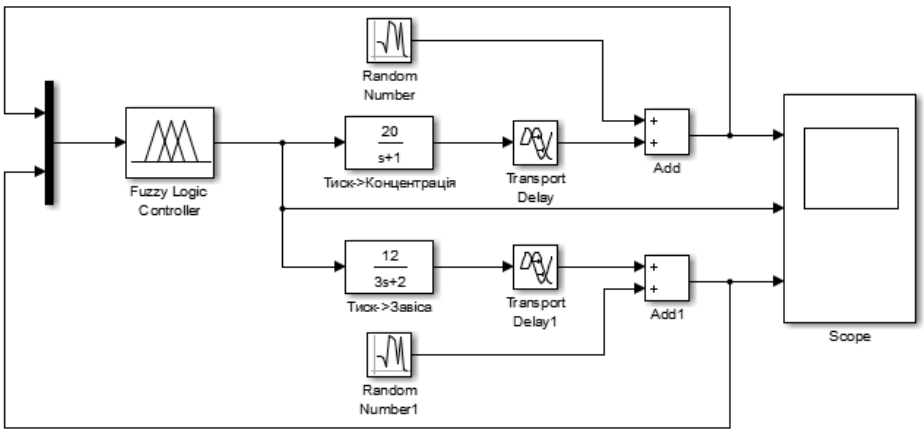


Рис. 5.8. *S*-модель нечіткої системи керування

Якщо система нечіткого логічного висновку має декілька входів, тоді в *Simulink*-моделі ці входи необхідно мультиплексувати разом до вводу в нечіткий контролер. Аналогічно, якщо система нечіткого логічного висновку має декілька виходів, то вихідні сигнали блоку будуть представлені однією мультиплексною лінією. Транспортні запізнювання можна встановити однаковими (наприклад, 1).

Задання параметру нечіткого регулятора (НР). ***Перед активацією редагування параметрів НР*** необхідно імпортувати створені в лабораторних роботах 3 та 4 моделі та правила нечіткого логічного висновку (тобто НчАСК) в *Workspace*. Виконати це можна різними методами. Один з них передбачає виклик файлу, в якому міститься НчАСК (у нас це *mytip.fis*), з диску у робоче середовище *Fuzzy Logic*. Після цього файл треба записати у *Workspace*. Послідовність дій наведена на рис. 5.9 та 5.10.

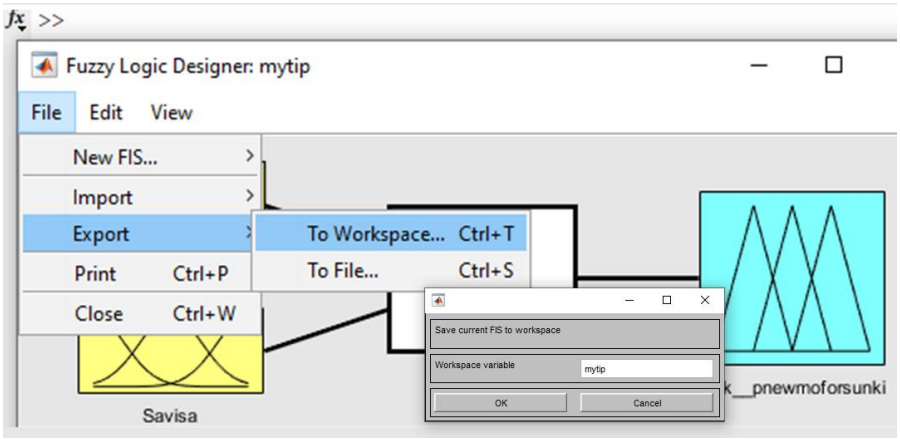


Рис. 5.9. Вікно з кроками занесення нечіткої системи у *Workspace*

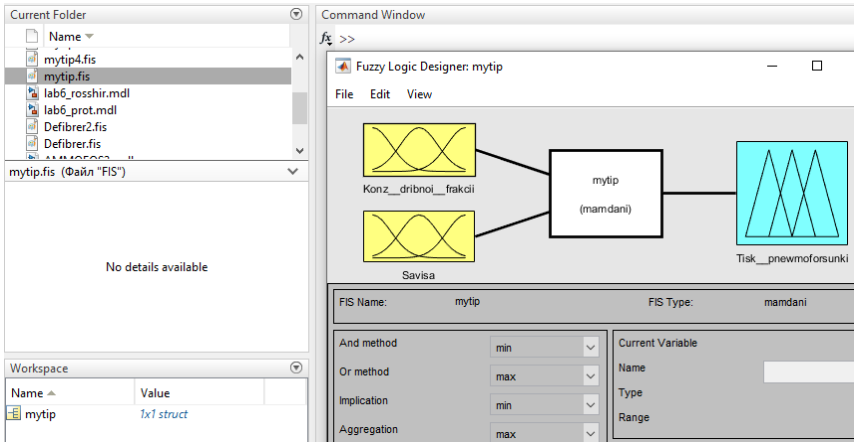


Рис. 5.10. Вікно з зображенням файлової системи для виклику *mytipe.fis*, з редактором нечіткої системи та зони *Workspace* з описом нечіткої системи

Інший спосіб передбачає активацію файлу з НЧАСК (*mytipe.fis*) без виклику редактору *FIS*. Активація відбувається викликом файлу *mytipe.fis* із вибраної папки (рис. 5.11).

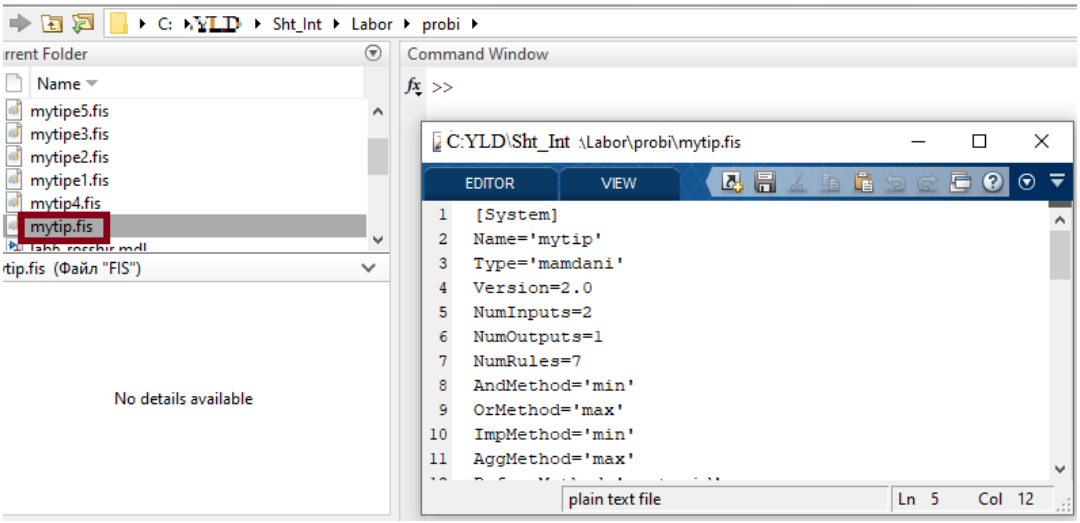


Рис. 5.11. Зображення вікон при виклику файлу *mytipe.fis* з диску

Після цього блок *Fuzzy Logic Controller* (на рис. 5.8) міститиме моделі та правила продукції, створені у лабораторних роботах № 3 та № 4, тобто його параметри можна налаштовувати (вікно зображене на рис. 5.12.). Для занесення у блок *Fuzzy Logic Controller* бази знань НчАСК необхідно в полі *FIS name* записати повну назву раніше створеного файлу(в нашому випадку вводимо *mytip.fis*. Студент повинен увести назву файлу ***Прізвище_група***), що був створений ним у попередніх роботах.

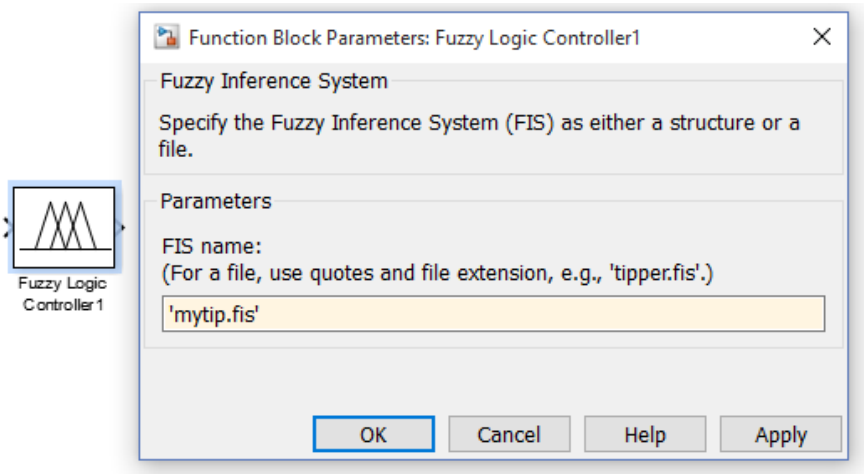


Рис. 5.12. Вікно налаштування блока *Fuzzy Logic Controller*

Після створення та налаштування моделі НчАСК треба виконати її випробування. Для цього необхідно запустити симуляцію, по її закінченню треба двічі клацнути ЛКМ по позначці *Scope*. Графіки сигналів вікна *Scope*, отримані при дослідженні НчАСК зі збуреннями, показані на рис. 5.13.

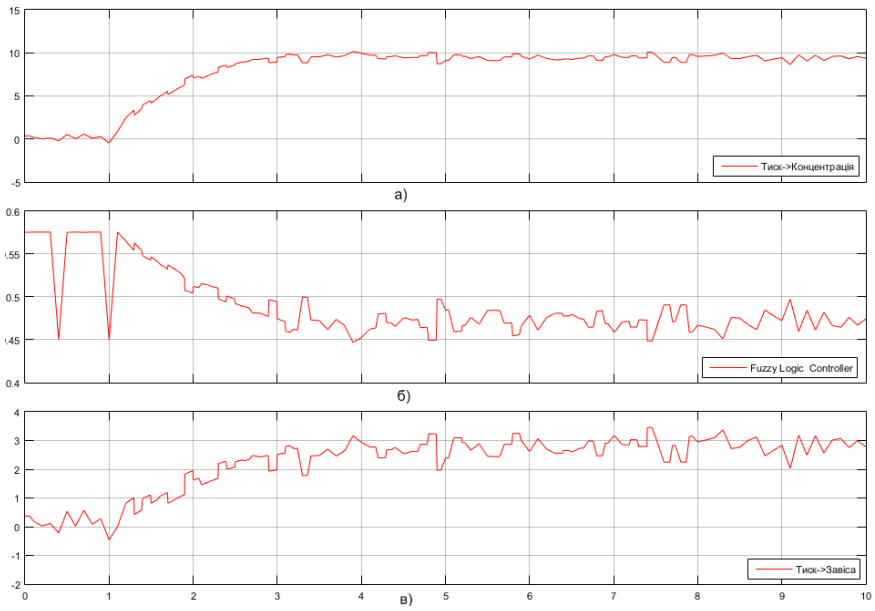


Рис. 5.13. Графіки сигналів НчАСК за наявності збурень:
а) перехідна характеристика за каналом «Тиск→Концентрація дрібної фракції»;
б) керувальний сигнал контролера; в) перехідна характеристика за каналом «Тиск→Завіса»

У *Simulink* для того, щоб при дослідженні НчАСК переглядати в реальному часі виконання правил, використовують блок *Fuzzy Logic Controller with Ruleviewer*. Параметри налаштування цього регулятора вказано на рис. 5.14. Вікно з переглядом правил, яке виникає під час роботи системи, – на рис. 5.15.

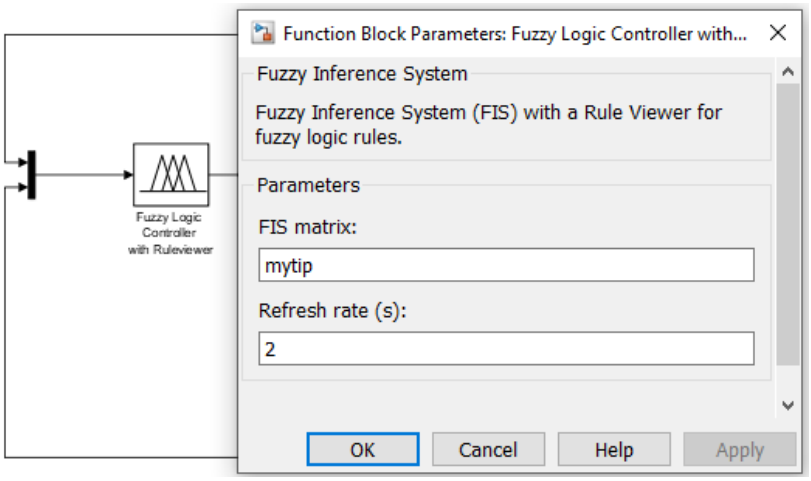


Рис. 5.14. Параметри налаштування нечіткого регулятора з виведенням правил

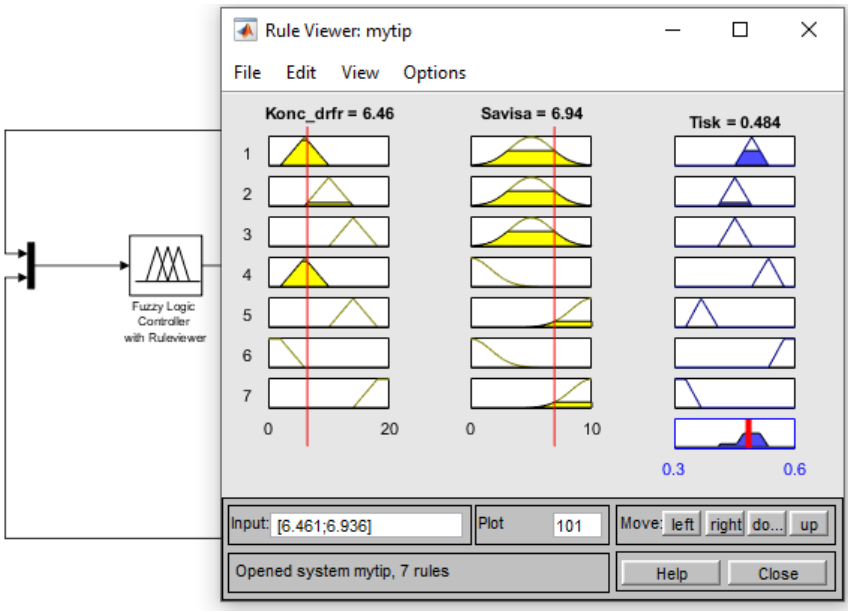


Рис. 5.15. Елементи вікон *Simulink*-моделі НчАСК з *Fuzzy Logic Controller with Ruleviewer* та виведення правил нечіткого висновку

Порядок виконання роботи

1. Скласти модель системи керування процесами сушіння та гранулювання у *Simulink* згідно до рис. 5.8.

2. Дослідити роботу моделі та зафіксувати графіки значень контрольованих величин.

3. Увести у модель рис. 5.8 канали збурення по вологості пульпи і дослідити їх вплив на роботу системи керування.

4. Використати у моделі блок *Fuzzy Logic Controller with Ruleviewer* (рис. 5.13) та проілюструвати графіками роботу НчАСК.

Вміст звіту

Моделі нечіткої системи керування у *Simulink* без каналів збурення по вологості пульпи та з ними. Графіки роботи нечіткої системи керування при різних типах нечітких регуляторів.

Контрольні запитання і завдання

1. Як увести у модель *Simulink* базу знань нечіткої системи керування, створену за допомогою функції *fuzzy*?

2. У яких розділах бібліотеки знаходиться кожний блок моделі нечіткої системи?

3. Пояснити результати досліджень, проведених у лабораторній роботі.

ПРАКТИКУМ 6

ІДЕНТИФІКАЦІЯ ОБ'ЄКТА КЕРУВАННЯ НЕЙРОННОЮ МЕРЕЖЕЮ

Мета роботи: Навчитися створювати нейронні мережі засобами *MATLAB,* які відтворюють поведінку об'єктів керування у статичних режимах. Вихідними даними можуть бути результати експериментів або функціональні залежності.

Стислі теоретичні відомості та задача ідентифікації

Нейронні мережі (*NN – Neural Networks*) використовують для виконання різноманітних завдань. Серед галузей їх застосування – обробка аналогових і цифрових сигналів, синтез та ідентифікація об'єктів і систем. Математичний процесор *MATLAB* має спеціалізовані засоби для створення нейронних мереж.

У лабораторній роботі розглянуті питання, пов'язані з мережами прямого поширення сигналу, тобто без зворотних зв'язків. Головною позитивною властивістю таких мереж є їхня стійкість.

Вибір структури *NN* і типів нейронів – самостійна достатньо складна задача, у цій роботі вона не розглядається. Після того як структура *NN* обрана, повинні бути визначені її параметри. Це досягають шляхом розв'язування певної задачі оптимізації, таку процедуру в теорії *NN* називають навчанням мережі.

У роботі будуть розглянуті наступні питання, пов'язані використанням засобів *MATLAB* для створення нейромережевих моделей об'єктів керування:

1) інтерфейс діалогових вікон;
2) підготовка даних;
3) створення нейронної мережі;
4) навчання мережі.

Під ідентифікацією об'єкта керування розуміють визначення математичної моделі, яка встановлює з відомою точністю зв'язок між вхідними і вихідними змінними об'єкта. Її виконують на основі результатів, отриманих при експериментальних дослідженнях об'єкта.

Задача ідентифікації. Розглянемо створення нейронної мережі для ділянки виробництва азотної кислоти. Згідно з технологією азотну кислоту перед відвантаженням споживачам очищують від оксидів азоту. Для цього у

спеціальному апараті крізь неї пропускають повітря. Повітря захоплює розчинені в кислоті оксиди азоту, тим самим зменшуючи їх концентрацію в кислоті (C, %). Відомо, що на процес очищення впливають такі фактори, як витрата повітря (V, м^3/с), витрата кислоти (F, м^3/с) та рівень кислоти в апараті (L, м).

Інформаційну схему об’єкту зображено на рис. 6.1.

Рис. 6.1. Інформаційна схема об’єкта ідентифікації

Отримана нейронна мережа стане імітаційною статичною моделлю процесу очищення кислоти для поточного розрахунку частки оксидів азоту в кислоті за виміряними значеннями зазначених трьох змінних:

$$C = f(V, L, F). \qquad (6.1)$$

Експериментальні дані, отримані в результаті досліджень процесу очищення азотної кислоти, наведені в табл. 6.1.

Таблиця 6.1. **Експериментальні дані дослідження процесу очищення кислоти**

№	V	L	F, 10^3	C	№	V	L	F, 10^3	C
1	0,861	0,650	1,111	0,091	16	0,694	0,550	1,944	0,140
2	0,889	0,650	1,111	0,088	17	1,000	0,600	1,250	0,070
3	1,139	0,600	1,806	0,078	18	0,917	0,650	1,944	0,130
4	0,861	0,700	2,083	0,152	19	0,806	0,600	1,528	0,116
5	1,128	0,700	1,250	0,051	20	0,806	0,750	0,833	0,107
6	1,389	0,750	1,944	0,079	21	0,972	0,650	0,972	0,074
7	1,194	0,650	1,806	0,081	22	0,750	0,750	1,806	0,164
8	0,917	0,550	0,833	0,065	23	1,111	0,650	1,528	0,073
9	1,111	0,500	1,667	0,064	24	1,028	0,750	0,972	0,072
10	0,750	0,650	1,389	0,125	25	1,194	0,550	1,250	0,047
11	1,333	0,700	2,222	0,082	26	1,194	0,700	1,667	0,085
12	1,306	0,600	0,833	0,038	27	1,139	0,550	1,111	0,048
13	1,278	0,500	1,528	0,046	28	0,694	0,700	1,944	0,169
14	0,861	0,550	1,389	0,087	29	0,722	0,700	1,111	0,121
15	1,111	0,500	1,528	0,060	30	0,917	0,500	2,222	0,107

Розглянемо послідовність створення нейронної мережі засобами *MATLAB* для формування залежності (6.1).

Формування масивів експериментальних даних. Перед створенням мережі необхідно підготувати набір вхідних та вихідних (цільових) навчальних даних. Такими даними в нашому випадку є експериментальні дані (табл. 6.1). Ці дані були відібрані в статичних режимах.

Вхідні навчальні дані задають матрицею розміром $n \times m$, де n – кількість рядків, яка відповідає кількості входів мережі, а m – кількість стовбців, що дорівнює кількості елементів в навчальній вибірці. Цільові дані задають матрицею розміром $l \times m$, де l – кількість виходів мережі.

Щоб задати матрицю входів (вхідну матрицю), створимо в робочому просторі *MATLAB* матрицю з трьома рядками: перший рядок відповідає витраті повітря *V*; другий – рівню кислоти в апараті *L*; третій – витраті кислоти *F*. Приклад заповнення матриць наведено на рис. 6.2. Для спрощення у прикладі використано результати перших 10 з 30 дослідів. Видно, що робочому просторі (*Workspace*) з'явилися матриці **A** та **Targ**.

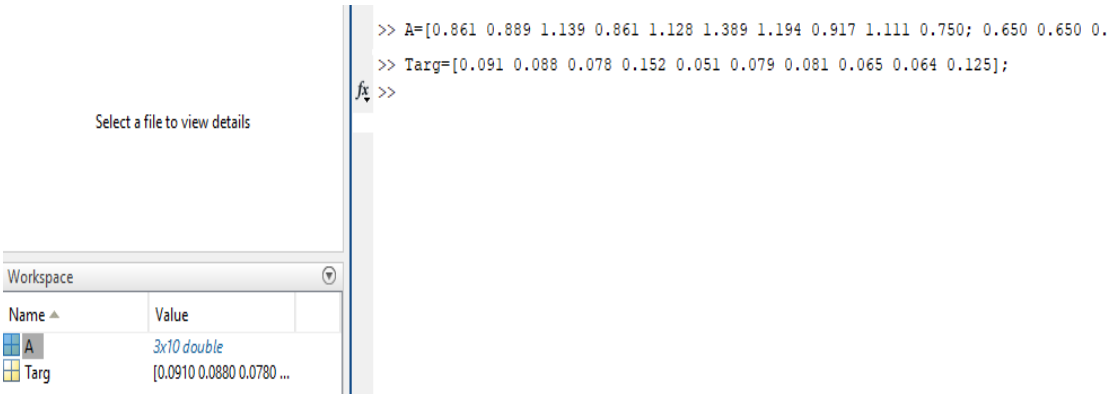


Рис. 6.2 – Приклад заповнення масивів вхідних та вихідної змінних моделі у командному вікні

Матриці **A** та **Targ** можна створити спочатку в *MS Excel*, а потім імпортувати в *MATLAB*.

Існують дві версії програмної реалізації створення нейромережевих моделей: ***NNTool*** та ***NFTOOL***. Перша може працювати у версіях *MATLAB* до 2023 р.

6.1. Використання *NNTool*

Для завантаження *NNTool*, необхідно виконати однойменну команду в командному вікні *MATLAB*:

>> nntool

Після цього з'явиться головне вікно *NNTool* (рис. 6.3): «Вікно керування мережами і даними» (*Network / Data Manager*). У нижньому рядку меню цього вікна розташовані вкладки з наступними призначеннями:

1) імпорт (*Import ...*) – імпорт даних з робочого простору *MATLAB* в простір змінних;

2) нові дані (*New ...*) – виклик вікна, що дозволяє створювати нейронні мережі та нові набори даних;

3) відкриття (*Open …*) – відкриття вікна перегляду та редагування наборів даних або нейронних мереж;

4) експорт (*Export ...*) – експорт даних з простору змінних *NNTool* в робочий простір *MATLAB* або файл;

5) видалення (*Delete*) – видалення обраного об'єкта;

6) допомога (*Help*) – короткий опис керуючих елементів цього вікна;

7) закриття (*Close*) – закриття вікна *NNTool*.

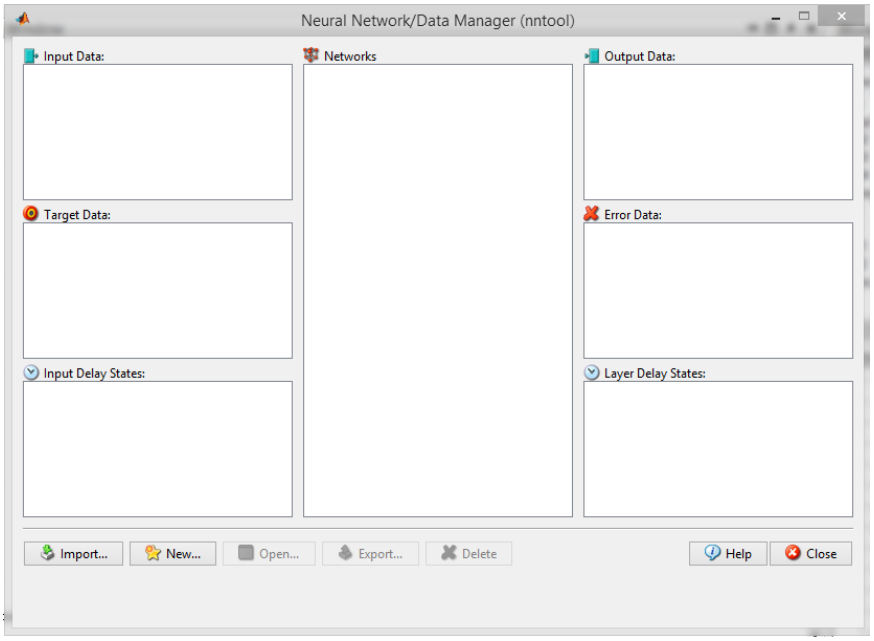


Рис. 6.3. Головне вікно *NNTool*

Для переходу до вікна керування даними (*Data Manager*), зображеного на рис. 6.4, треба скористатися кнопкою *Import*. З переліку доступних змінних треба вибрати матрицю **A**, визначити її як вхідні дані (*Input Data)* і натиснути на кнопку *Import*.

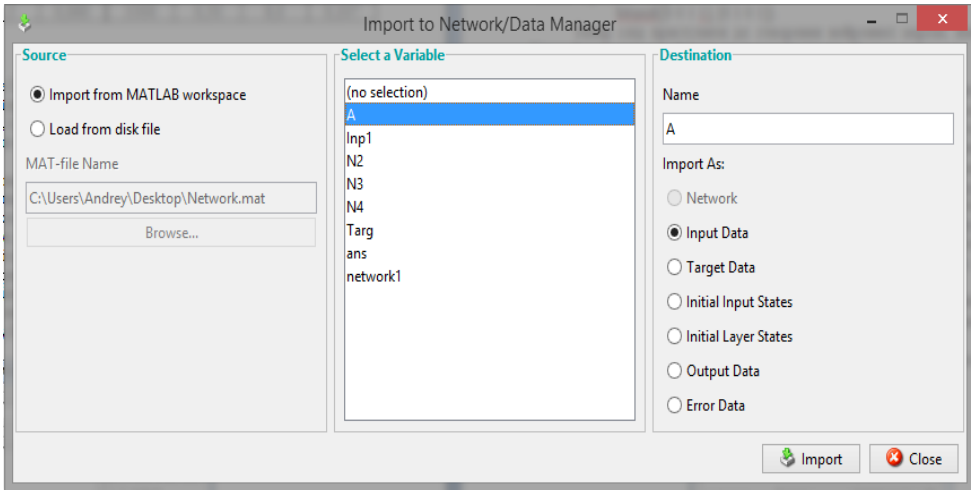


Рис. 6.4. Вікно визначення вхідних матриць і векторів

За цим виникає вікно для підтвердження виконаної дії (рис. 6.5).

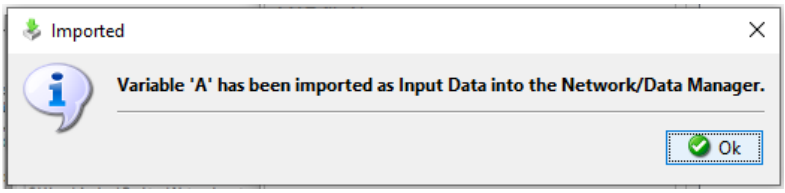


Рис. 6.5. Вікно підтвердження подання даних матриці **A** як вхідних змінних нейромережі

Вектор вихідної змінної (**Targ**) задають схожим чином (рис. 6.6). Для повернення в головне меню *NNTool* (рис. 7.3) треба натиснути кнопку *Close*.

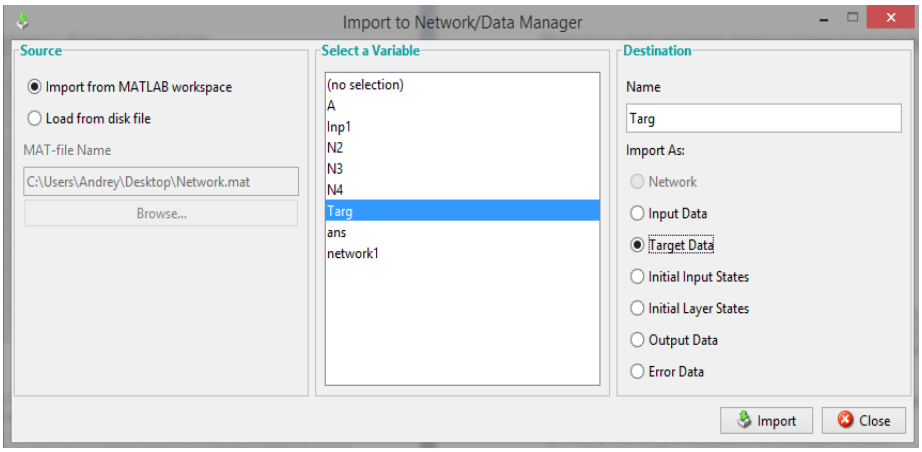


Рис. 6.6. Задання вектору **Targ**, як вихідної змінної нейромережі

Для того, щоб задати входи та вихід нейромережі треба натиснути на кнопку *New* на вкладці *Data* головного меню *NNTool* та увести потім дані в вікно *Value*.

Створення мережі. Треба вибрати мережу, яка має 3 входи, 1 прихований шар з 2 нейронів та один вихід.

У головному меню *NNTool* натискаємо на кнопку *New* і заповнюємо форму створення мережі (*Create Network of Data*) так, як показано на рис. 6.7. Після внесення потрібних налаштувань активуємо команду *Creat*.

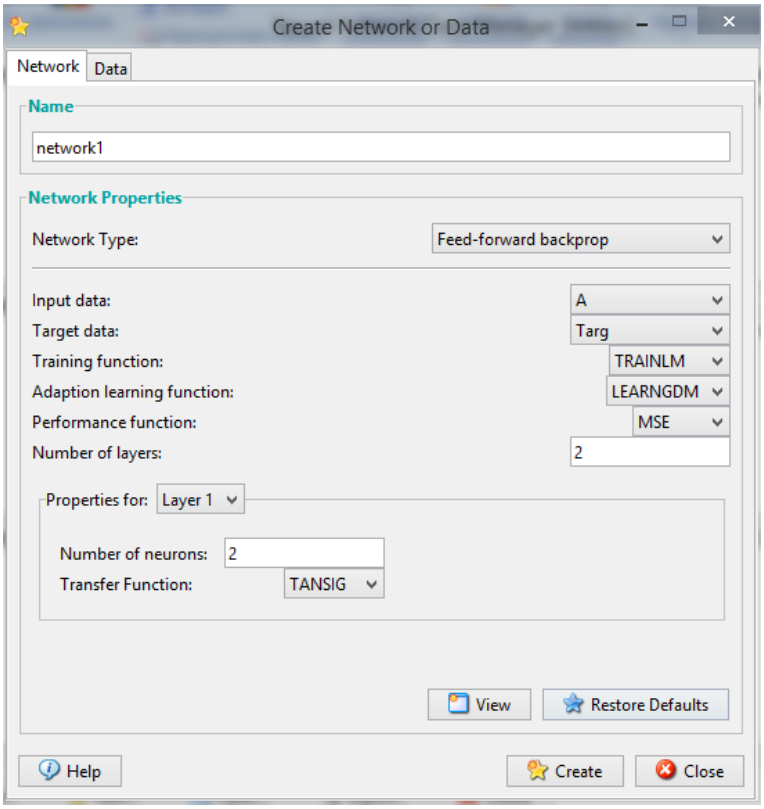


Рис. 6.7. Вікно «Створення мережі»

Поля цього вікна мають наступні смислові навантаження:

- ім'я мережі (*Name*) – задання імені створюваної мережі.
- тип мережі (*Network Type*) – визначення типу мережі і в контексті обраного типу подає для введення різні параметри в частині вікна, розташованій нижче цього пункту (тобто, ***для різних типів мереж вікно змінює свій зміст***).

На рис. 6.7 наведено вид вікна для типу ***прямого поширення*** (*feedforward neural network)* та з методом навчання мережі шляхом ***зворотного поширення помилки*** (*backpropagation*).

Опишемо елементи цього вікна:

- вхідні дані (*Input data*) – вибір вхідних даних;
- виходи мережі (*Target data*) – вибір цільових даних;
- кількість шарів (*Number of layers*) – задання кількості прихованих шарів нейронної мережі, враховуючи вихідний;
- кількість нейронів (*Number of neurons*) – задання кількості нейронів у шарі;
- передавальна функція (*Transfer function*) – вибір функції активації нейронів;
- функція навчання (*Training function*) – вибір функції, що відповідає за оновлення ваг і зсувів мережі в процесі навчання;
- кількість шарів (*Number of layers*) – задання кількості прихованих шарів нейронної мережі, враховуючи вихідний.

За допомогою кнопки «Вид» (*View*) можна побачити архітектуру створеної мережі (рис. 6.8). Це дозволяє пересвідчитися, чи всі дії були проведені вірно.

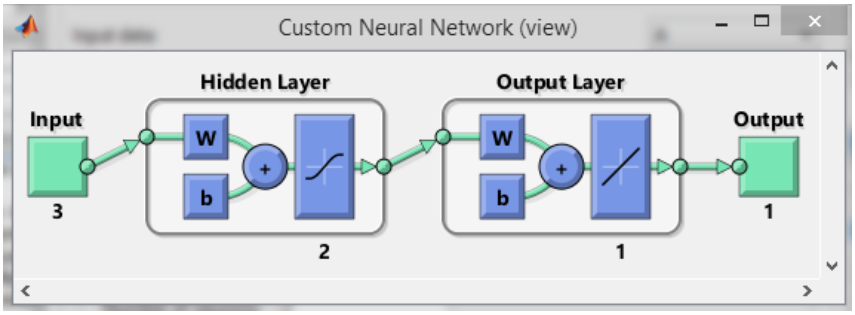


Рис. 6.8. Вікно попереднього перегляду створеної мережі

Якщо структура мережі відповідає завданню, то можна закрити вікно попереднього перегляду і підтвердити намір про створення мережі. Це виконують натисканням кнопки *Create* у вікні створення мережі (рис. 6.7).

Виникає також вікно для підтвердження створення мережі (рис. 6.9).

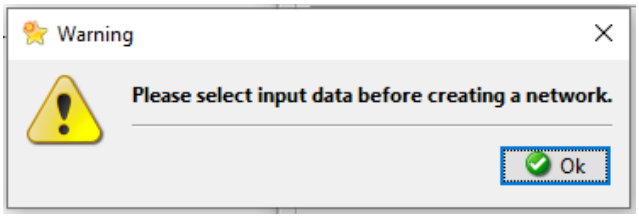


Рис. 6.9. Вікно з запитом на підтвердження мережі

У результаті виконаних операцій в розділі «Мережі» (*Networks*) головного меню *NNTool* (рис. 6.3) з'явиться об'єкт з назвою *network1* (рис. 6.10).

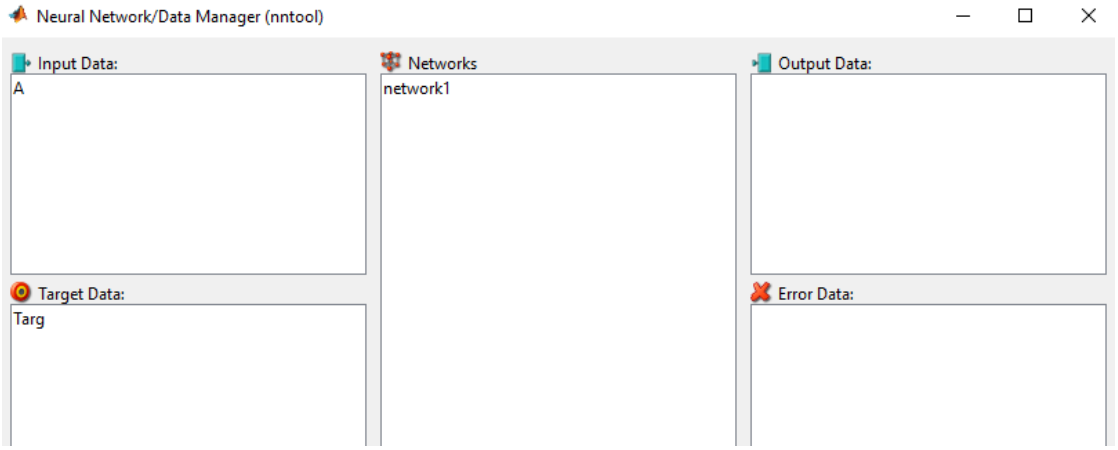


Рис. 6.10. Фрагмент головного меню *NNTool* з інформацією про вхідні та вихідні дані мережі *network1*

Навчання мережі. Відомо, що зразу після формування поведінка нейронної мережі не буде відповідати поведінці досліджуваного об’єкту. Для отримання адекватної моделі об’єкта мережу необхідно належним чином навчити, тобто підібрати такі значення її параметрів, які б забезпечили задані критерії точності мережі як моделі.

Для цього слід повернутися до головного вікна *NNTool* (рис. 6.10). Позначивши покажчиком миші об'єкт мережі *network1* і натиснувши кнопку *Open*, можна відкрити вікно керування мережею з переліком вкладок (рис. 6.11) з функціями керування навчанням. Активуємо вкладку *Train*. У результаті побачимо наступне діалогове вікно, на якому активуємо «Інформація навчання» (*Training info*) та заповнимо необхідні поля: потрібно вказати масив навчальних даних у полі «Входи» (*Inputs*) і вектор вихідних даних у полі «Цілі» (*Targets*). Поля «Виходи» (*Outputs*) і «Помилки» (*Errors*) *NNTool* заповнюються автоматично. При цьому результати навчання, до яких належать виходи і помилки, будуть зберігатися в змінних із зазначеними іменами.

Завершити процес навчання можна, керуючись різними критеріями. Можливі ситуації, коли доречно зупинити навчання, вважаючи достатнім деякий інтервал часу. Об'єктивним критерієм є також значення помилки моделювання.

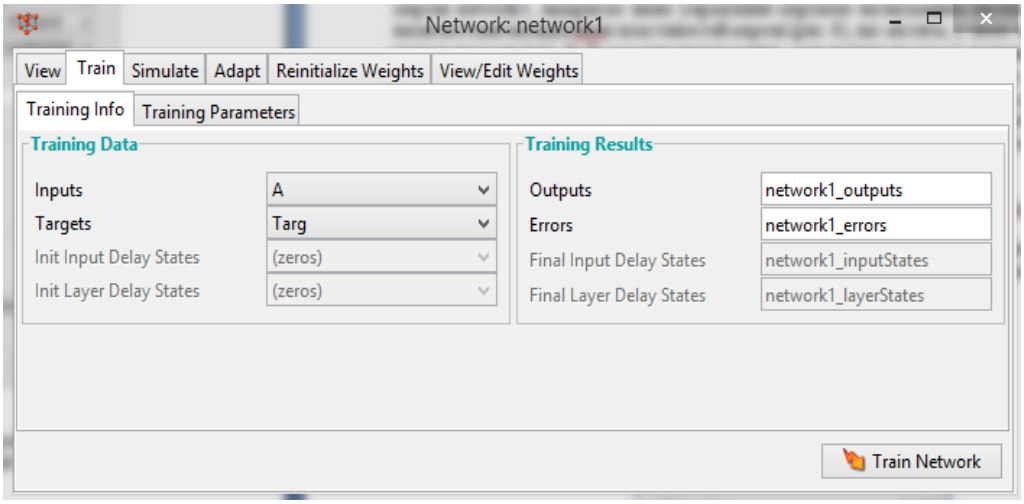


Рис. 6.11. Вікно параметрів *Training Info*, що відкрите на вкладці «Навчання» (*Train*)

У діалоговому вікні вкладки «Параметри навчання» (*Training parameters*) для вибраної мережі (рис. 6.8) можна встановити наступні параметри (рис. 6.12):

- кількість епох (*epochs*) – визначає кількість епох, по закінченні яких навчання буде припинено (епоха – одноразове подання всіх навчальних вхідних даних на входи мережі);
- досягнення мети або попадання (*goal*) – тут задається абсолютна величина функції помилки, при якій мета буде вважатися досягнутою;
- період оновлення (*show*) – період оновлення графіка кривої навчання, виражений числом епох;
- час навчання (*time*) – після закінчення цього часового інтервалу, поданого в секундах, навчання припиняється.

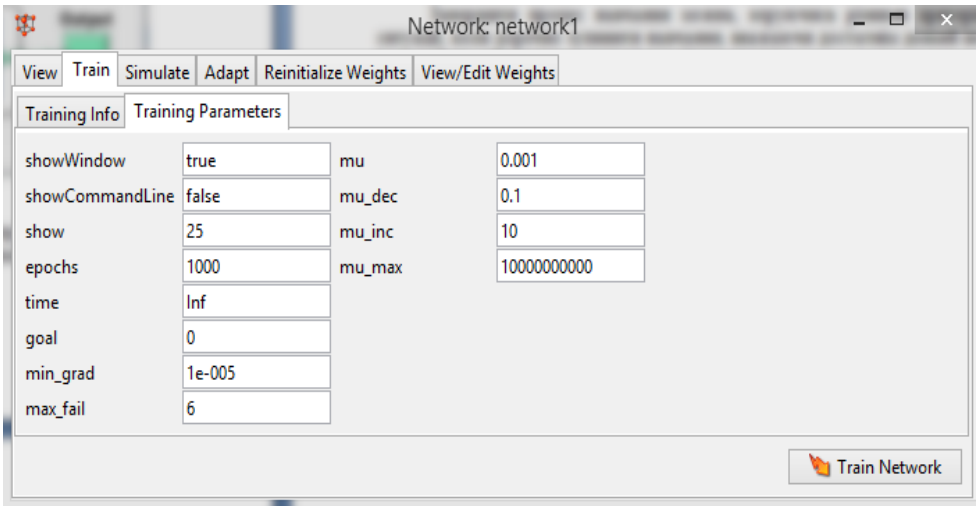


Рис. 6.12. Вікно вкладки визначення параметрів навчання (*Training parameters*)

Значення параметрів залишимо за умовчанням. Зауважимо тільки, що поле часу навчання містить запис *Inf*, який визначає нескінченний інтервал часу (від англійського *Infinite* – нескінченний).

Для того, щоб почати навчання, потрібно натиснути на кнопку «Навчити мережу» (*Train Network*).

Після цього з’явиться вікно (рис. 6.13) в якому відображатиметься стан навчання мережі. Кнопкою «Зупинити навчання» (*Stop Training*) можна припинити процес навчання. З рисунку видно, що навчання було зупинене, оскільки градієнт досяг мінімального значення.

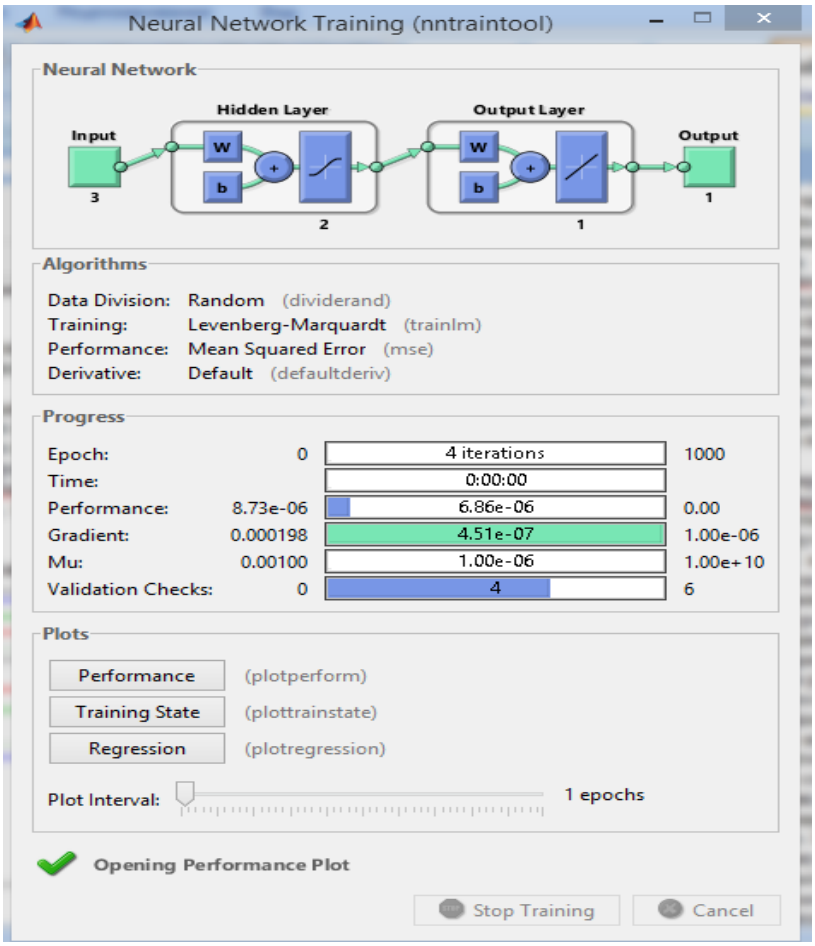


Рис. 6.13. Вікно інформації про стан навчання мережі

Після закінчення навчання можна переглянути графік, що ілюструє динаміку цільової функції – криву навчання. У нашому випадку графік може виглядати так, як показано на рис. 6.14. Отже, алгоритм навчання знайшов розв’язок задачі.

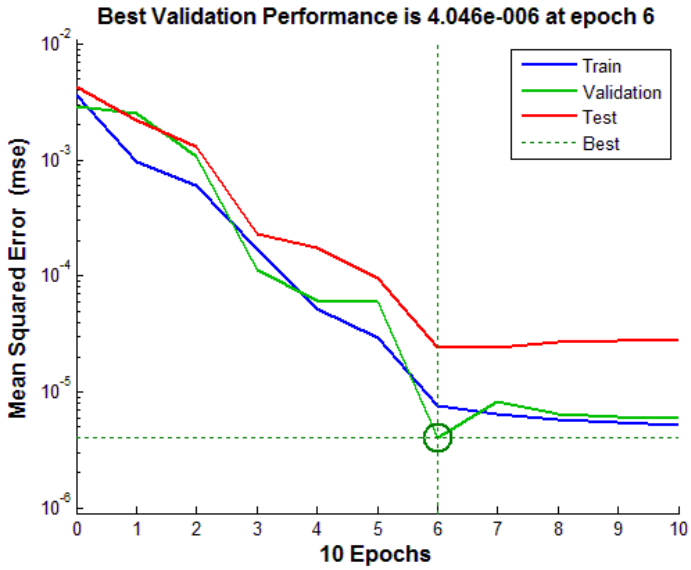


Рис. 6.14. Криві навчання нейромережі

Продемонструємо роботу нейронної мережі. Для цього експортуємо мережу в робочий простір *MATLAB* і виконаємо наступний скрипт:

```
[X, Y]=meshgrid(0,833*10^-3:10^-4:2,222*10^-3, 0.2:0.05:1);
k=size(X);
for j=1:k(1,1),
for i=1:k(1,2),
        Z1(j,i)=network1([0,694;Y(j,i);X(j,i)]);
        Z2(j,i)=network1([1,389;Y(j,i);X(j,i)]);
end
end
surf(X, Y, Z1);
hold on;
surf(X, Y, Z2);
```

Функція [*X*, *Y*] = *meshgrid*(*x*, *y*) формує масиви *X* та *Y*, які визначають координати вузлів прямокутника, який задають векторами *x* та *y*. Цей прямокутник відповідає області визначення функції двох змінних, яку можна побудувати у вигляді 3*D*-поверхні.

Команда *surf*(*X*, *Y*, *Z*) виводить на екран сітчасту поверхню для значень масиву *Z*, визначених на множині значень масивів *X* і *Y*.

Команда *hold on* включає режим зберігання поточного графіку і параметрів об’єкту *axes*, так що наступні команди призведуть до додавання нових графіків в графічному вікні.

Команда *size*(*X*) визначає розмірність масиву *X*.

У результаті описаних дій отримаємо дві поверхні в просторових координатах (рис. 6.15), які відображають вміст оксидів азоту в кислоті в залежності від рівня кислоти в апараті та її витрати. Вісь *X* відповідає витраті кислоти *F*, вісь *Y* – рівню кислоти в апараті, вісь *Z* – вмісту оксидів азоту. Нижня площина розрахована при витраті повітря V = 0,694 м^3/с, верхня – при V = 1,389 м^3/с.

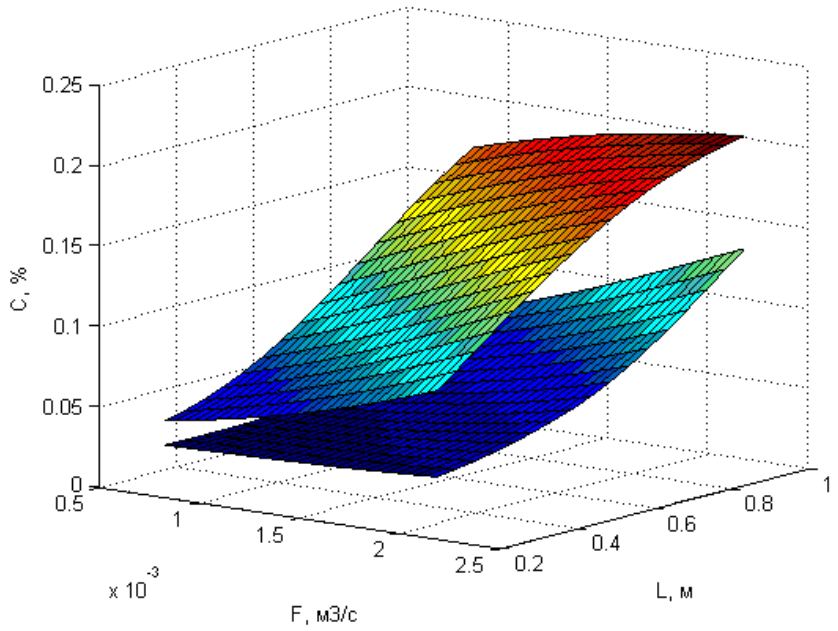


Рис. 6.15. Залежність оксидів азоту від рівня кислоти та її витрати

Створена нейронна мережа дозволить прогнозувати вміст оксидів азоту в кислоті. Слід зауважити, що мережа створюється ініціалізованою, тобто, із заданими значеннями ваг і зсувів. Початкові умови цих параметрів, зазвичай, оновлюють перед кожним наступним циклом навчання. Для цього на вкладці «Ініціалізація» (*Initialize*) передбачена функція ініціалізації. Якщо потрібно провести декілька незалежних циклів навчання, ініціалізацію ваг і зсувів перед кожним з них здійснюють натисканням кнопки «Ініціалізувати ваги» (*Initialize Weights*).

Для попередження перенавчання застосовують наступну методику. Дані розділяють на дві множини (вибірки): навчальна (*Training Data*) та контрольна (*Validation Data*). Контрольну множину в навчанні не використовують. На початку роботи припустимі помилки мережі на навчальній та контрольній множинах задають однаковими. У міру того, як мережа навчається, помилка навчання зменшується. Доки навчання зменшує дійсну функцію помилки, помилка на контрольній множині також буде спадати. Якщо ж контрольна помилка припинила зменшуватися або навіть стала зростати, то навчання слід закінчити. Зупинка на цьому етапі називається *ранньої зупинкою* (*Early stopping*).

Таким чином, необхідно провести серію експериментів з різними мережами перш ніж буде отримана необхідна. Для того, щоб не бути введеним в оману локальними мінімумами функції помилки, слід кілька разів навчати кожну мережу.

Якщо в результаті послідовних кроків навчання і контролю помилка залишається неприпустимо великою, доцільно змінити модель нейронної мережі (наприклад, ускладнити мережу, збільшивши кількість нейронів, або використати мережу іншого виду). У такій ситуації рекомендують застосовувати ще одну множину – тестову множину спостережень (*Test Data*), яка є незалежною вибіркою з вхідних даних. Нейронну мережу, яка пройшла навчання, тестують на цій множині. Така процедура дає додаткову можливість переконатися в точності ідентифікації об’єкта. Тестова множина повинна бути використана тільки один раз. Якщо її використати і для коригування мережі, то вона фактично перетвориться на контрольну множину.

6.2. Використання *NFTOOL*

Перейти до цієї бібліотеки можна або за командою ***NSSTART***, або за командою ***NFTOOL.*** У першому випадку спочатку виникне вікно вибору задач (рис. 6.16), у другому – зразу перейдемо до вікна ідентифікації (рис. 6.17).

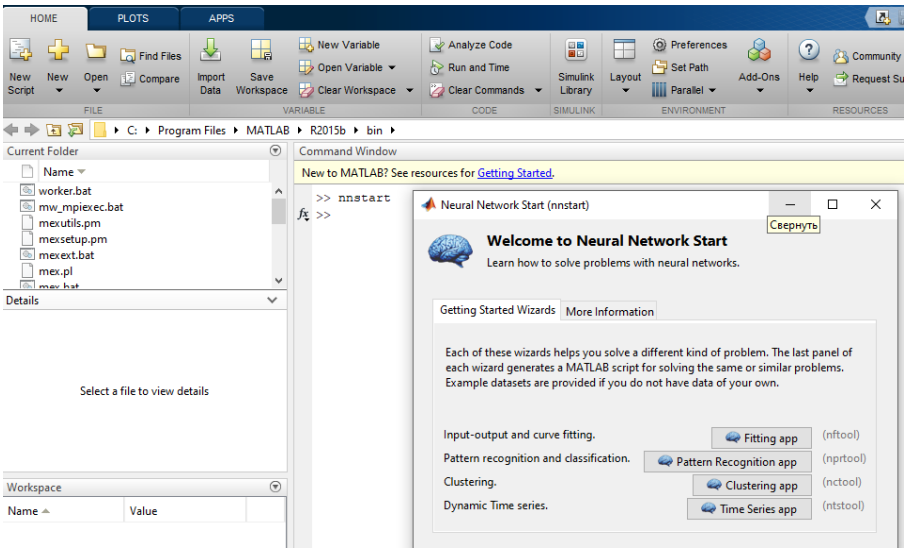


Рис. 6.16. Вікно вибору задач бібліотеки *NN*

У переліку задач треба вибрати пункт *Fitting app* (у дужках – команда *nftool*). Після вибору з’явиться перше вікно загальної інформації для проєктування нейромережі (рис. 6.17). У ньому треба вибрати *Next* і з’явиться вікно визначення джерел даних (рис. 6.18).

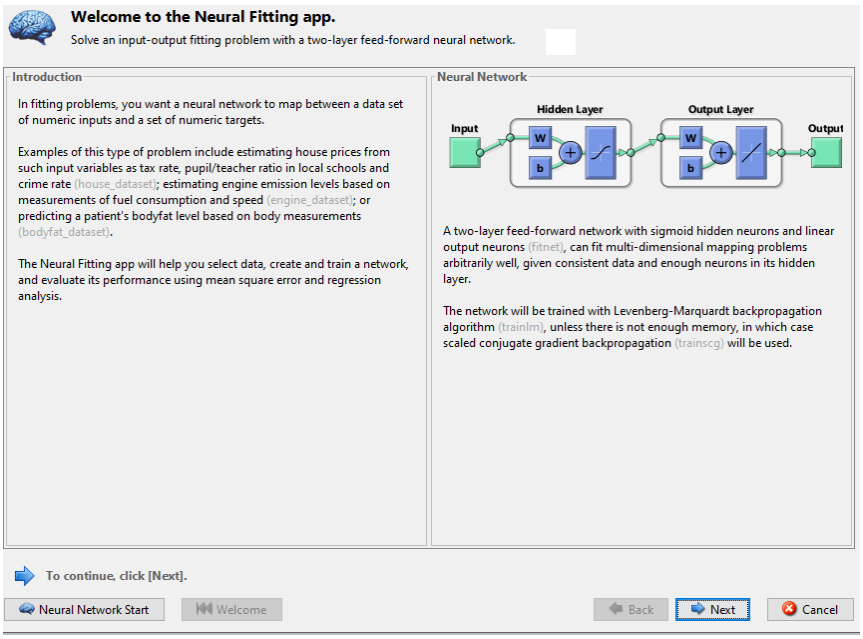


Рис. 6.17. Вікно загальної інформації для проєктування нейромережі

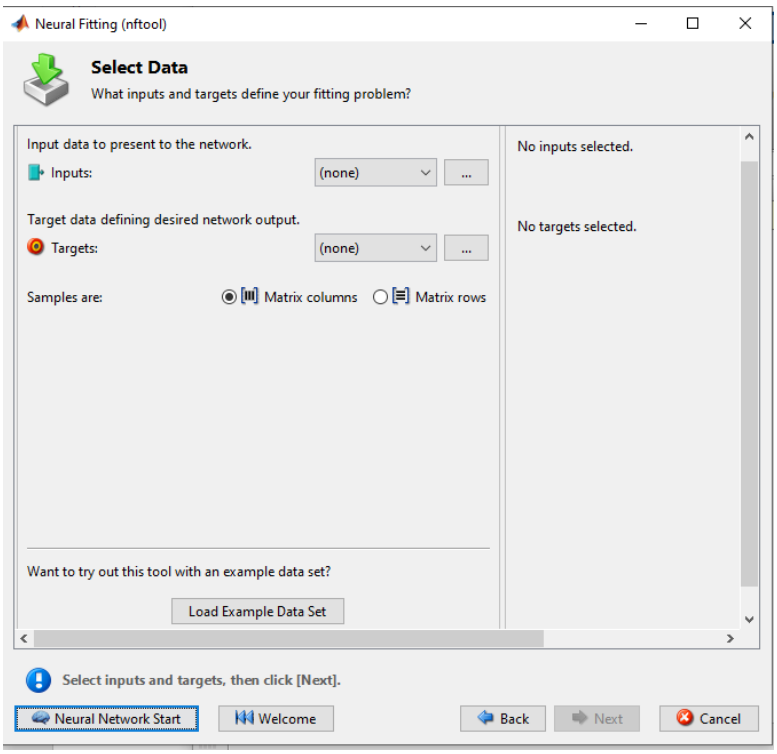


Рис. 6.18. Вікно визначення вхідних матриць і векторів у *NFTOOL*

Звернімося до п. 2 **Порядку виконання роботи**. Згідно з вибраною функцією п. 2 спочатку треба сформувати два масиви даних (**X** та **Y**). Для цього доцільно написати програму (код) розрахунку функції, вибраної з переліку наданих структур. Приклад – на рис. 6.19 (*Workspace* містить відповідні матриці).

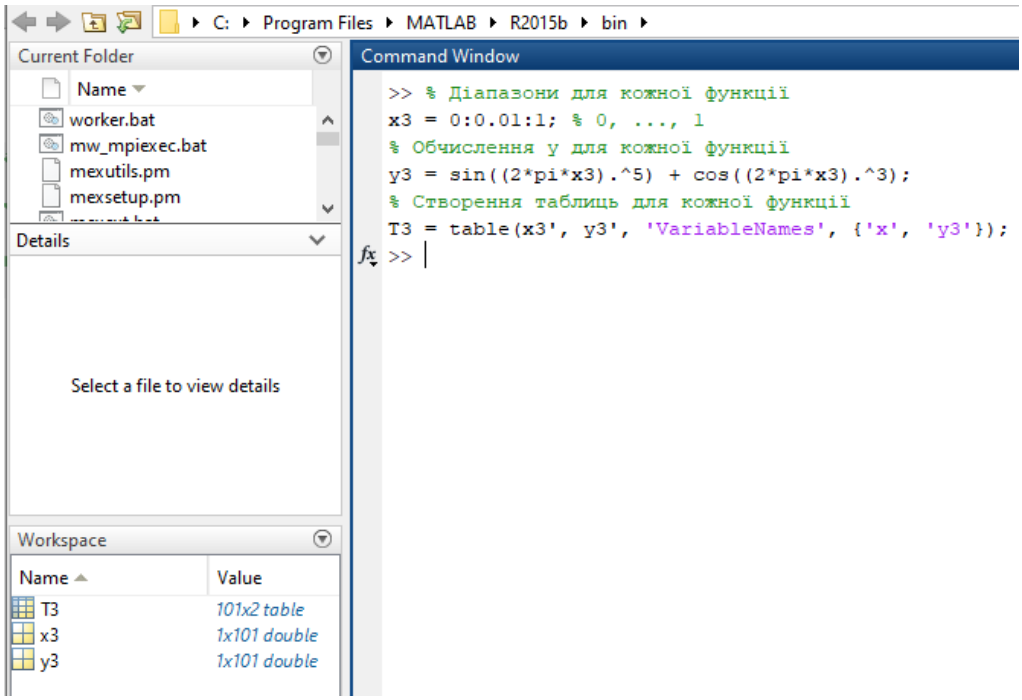


Рис. 6.19. Вікно з розрахунком функції $y = f(x)$

Далі треба увести масиви *x*3 та *y*3 (**X** та **Y)** у списки зон *Inputs* і *Targets* відповідно (див. рис. 6.18). Тоді можна натиснути кнопку *Next* для визначення розподілу даних для етапів навчання, контролю та тестування (рис. 6.20).

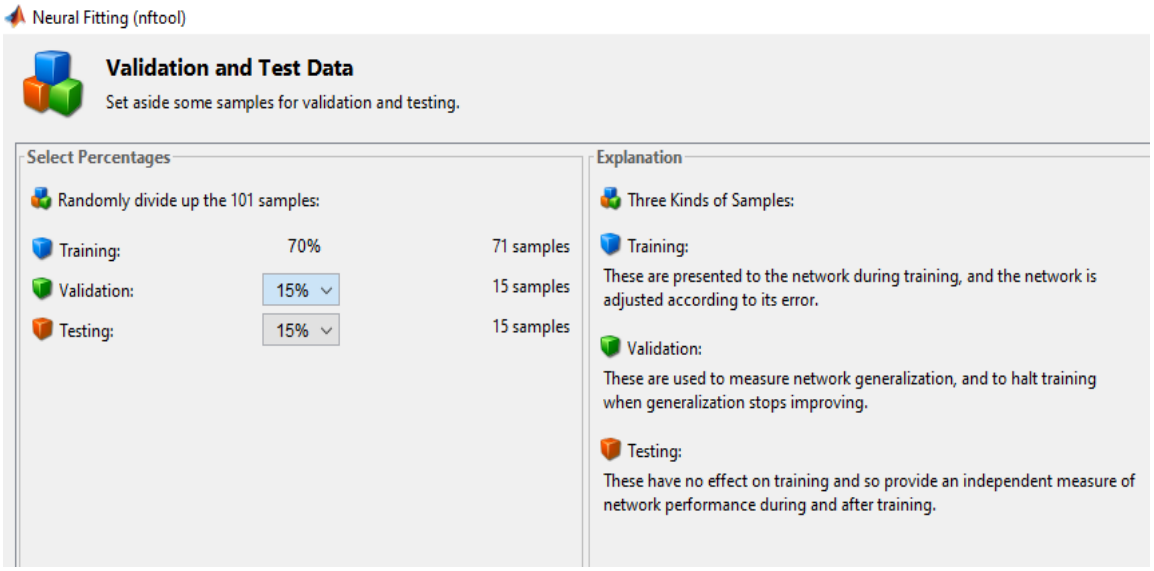


Рис. 6.20. Вікно розподілу даних для етапів навчання, контролю та тестування

Після натискання кнопки *Next* цього вікна з’являється вікно визначення структури мережі (рис. 6.21).

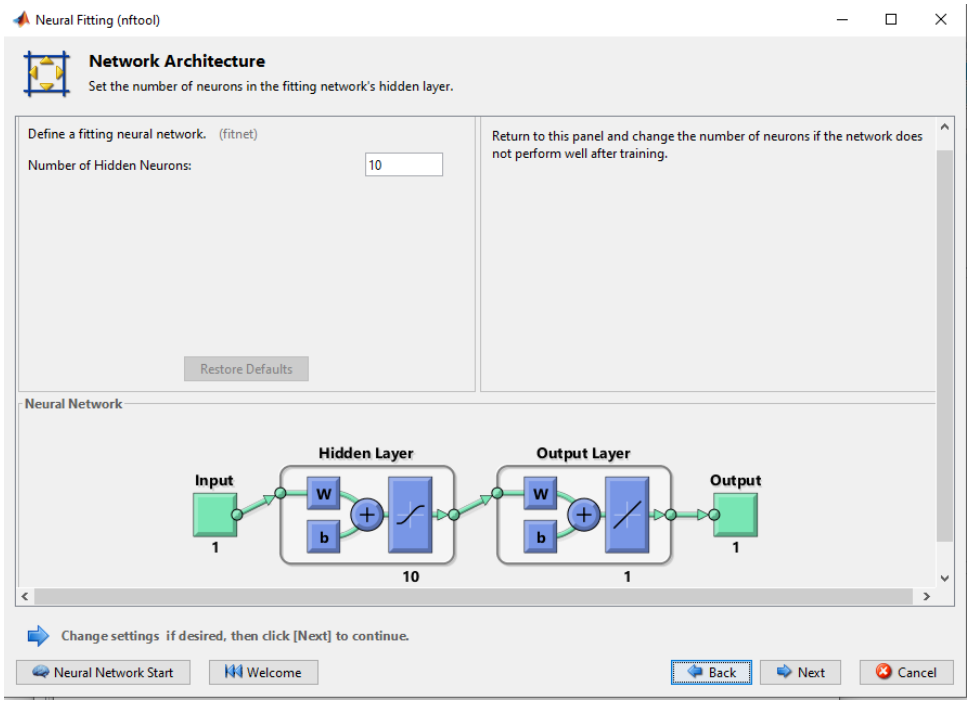


Рис. 6.21. Вікно визначення структури мережі

Після натискання кнопки *Next* цього вікна з’являється вікно визначення методу навчання мережі (рис. 6.22).

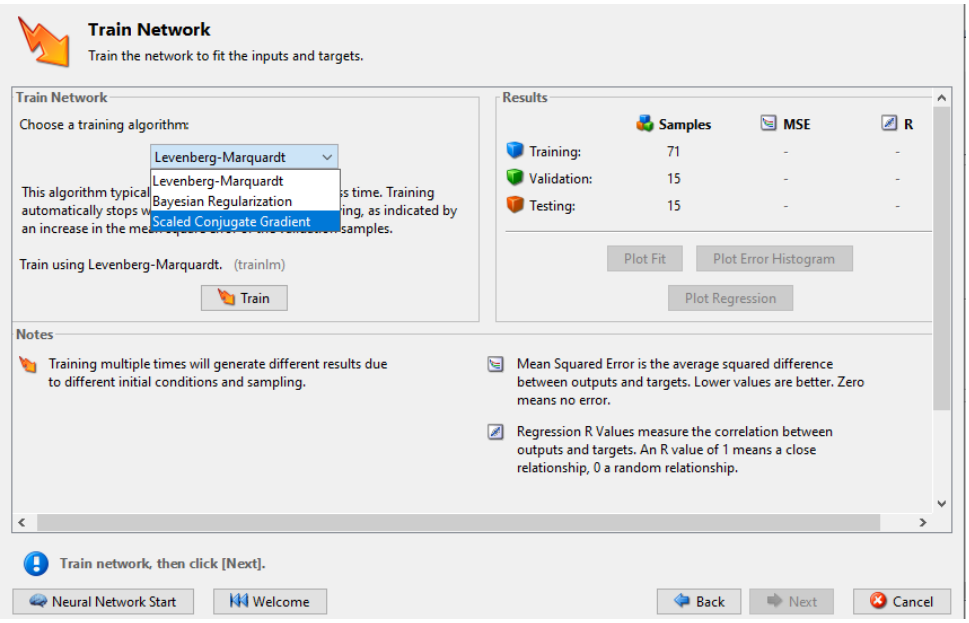


Рис. 6.22. Вікно визначення методу навчання мережі

Кнопкою *Train* можна розпочати процес навчання. Результати з'являються у вікнах виду 6.23 з числовими даними (середньоквадратична похибка та коефіцієнт кореляцію між вихідними даними об'єкта та розрахунком за моделлю) та з меню графічних результатів, а також виду 6.24 з більш широким спектром числових характеристик процесу навчання.

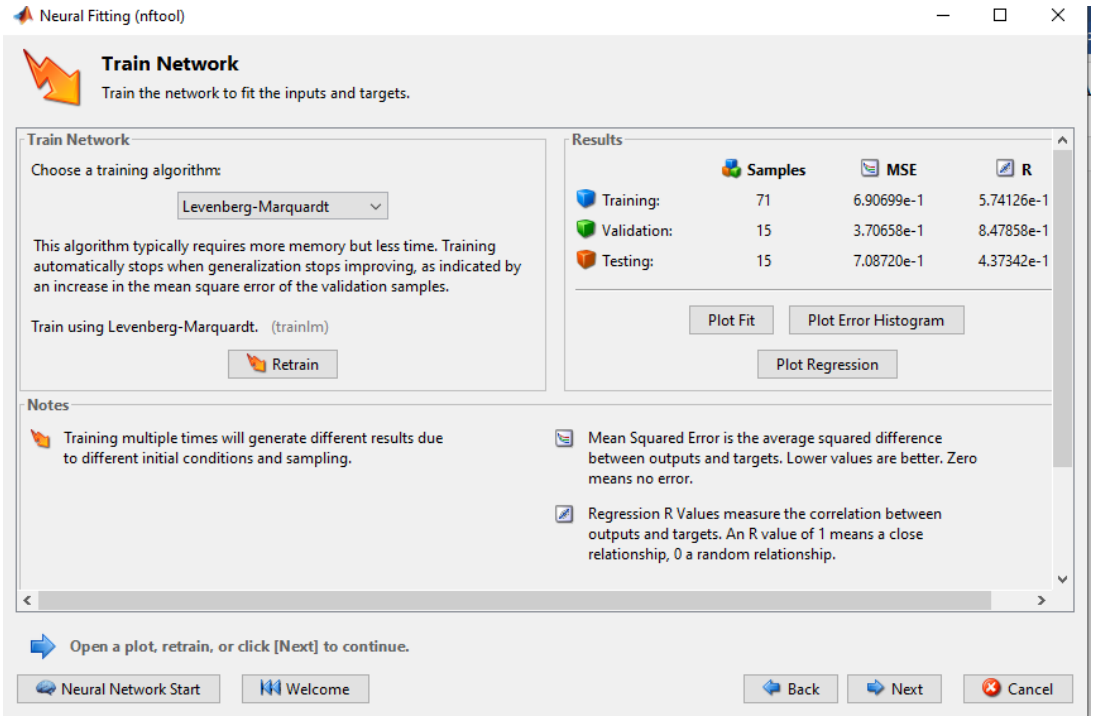


Рис. 6.23. Вікно з результатами навчання та меню графічних результатів

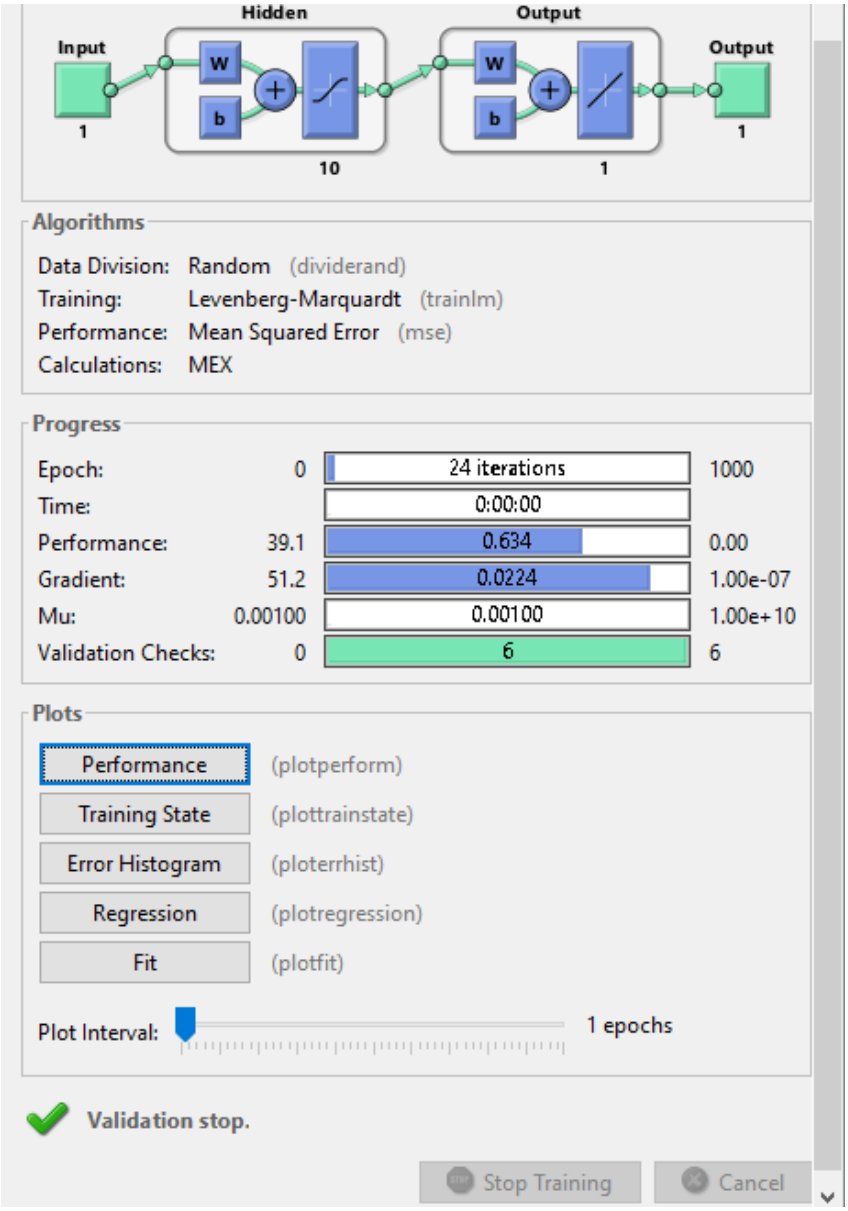


Рис. 6.24. Вікно числових характеристик процесу навчання

Почнемо з кнопки *Plot Fit* (рис. 6.23). У вікні рис. 6.25 зображено експериментальні дані (крапки, з'єднані прямими лініями) та результати моделювання ШНМ (зверху) та графік залишків (*Error*) моделі (знизу). Вісь абсцис – значення вхідної величини, *x*.

При натисканні кнопки *Plot Error Histogram* з'являється зображення стовпчастої діаграми, по осі абсцис якої відкладають помилки ШНМ (різниці між значеннями вихідної величини, *y* та результатами розрахунків за моделлю). По осі ординат відкладають частоти (кількості) появи відповідних різниць (рис. 6.26).

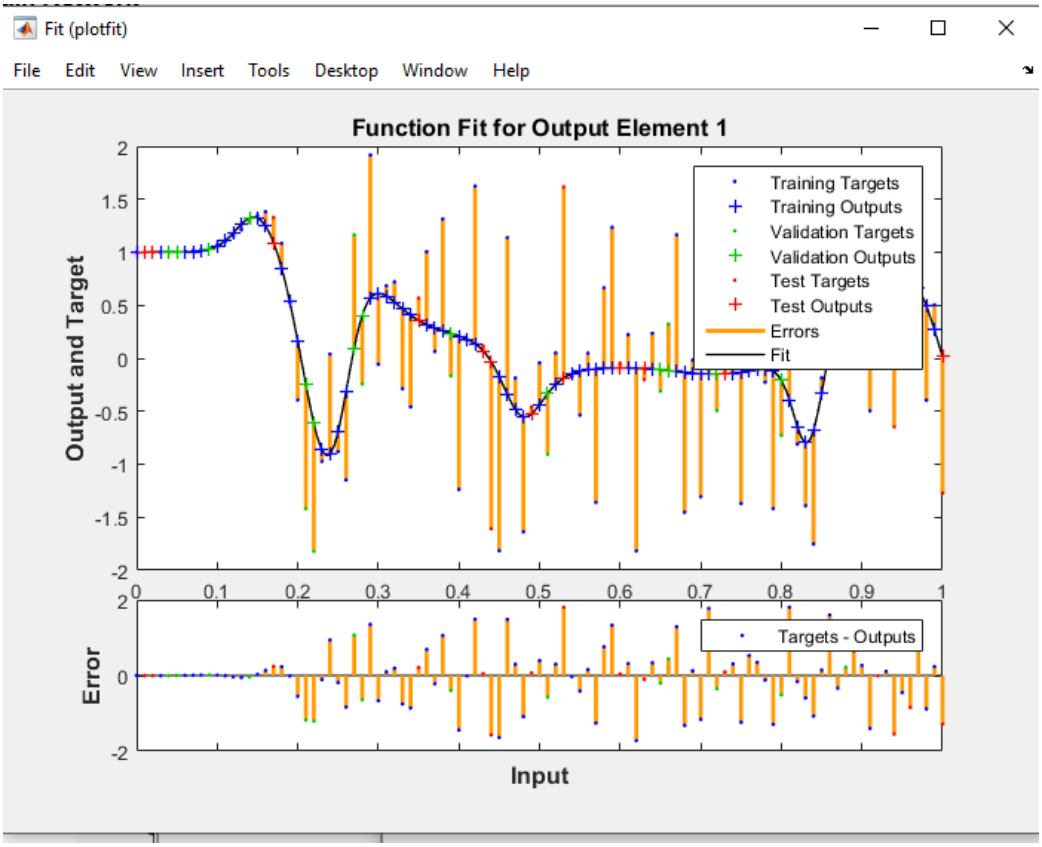


Рис. 6.25. Вікно *Plot Fit* з графічними результатами дослідження моделі

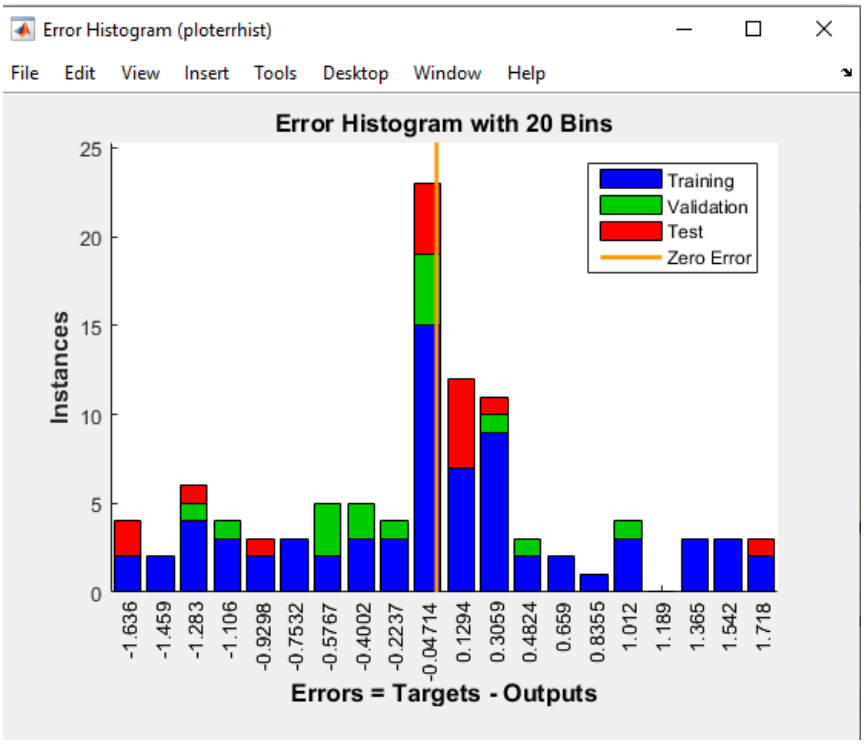


Рис. 6.26. Вікно *Plot Error Histogram* з графічними результатами дослідження моделі

Натискання на кнопку *Plot Regression* призводить до появи вікна (рис. 6.27) з кореляційними полями між значеннями вихідної змінної, *y* та результатами розрахунків за моделлю. Діапазон значень коефіцієнта парної кореляції між ними 0..1. Кореляційні поля наведені для кожної з вибірок і загального масиву даних.

Графіки показують, що модель недостатньо точна і процес навчання треба продовжити.

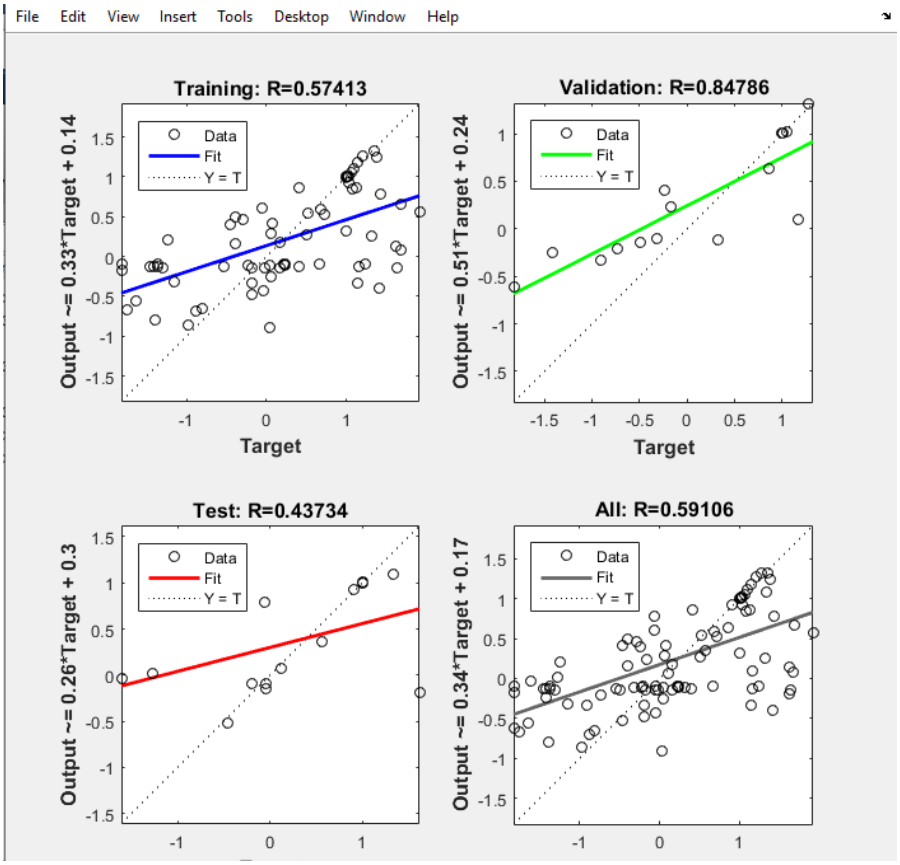


Рис. 6.27. Вікно *Plot Regression* з графічними результатами дослідження моделі

На рис. 6.28 наведено приклади зображень кореляційних полів, побудованих для добре навченої (адекватної) моделі.

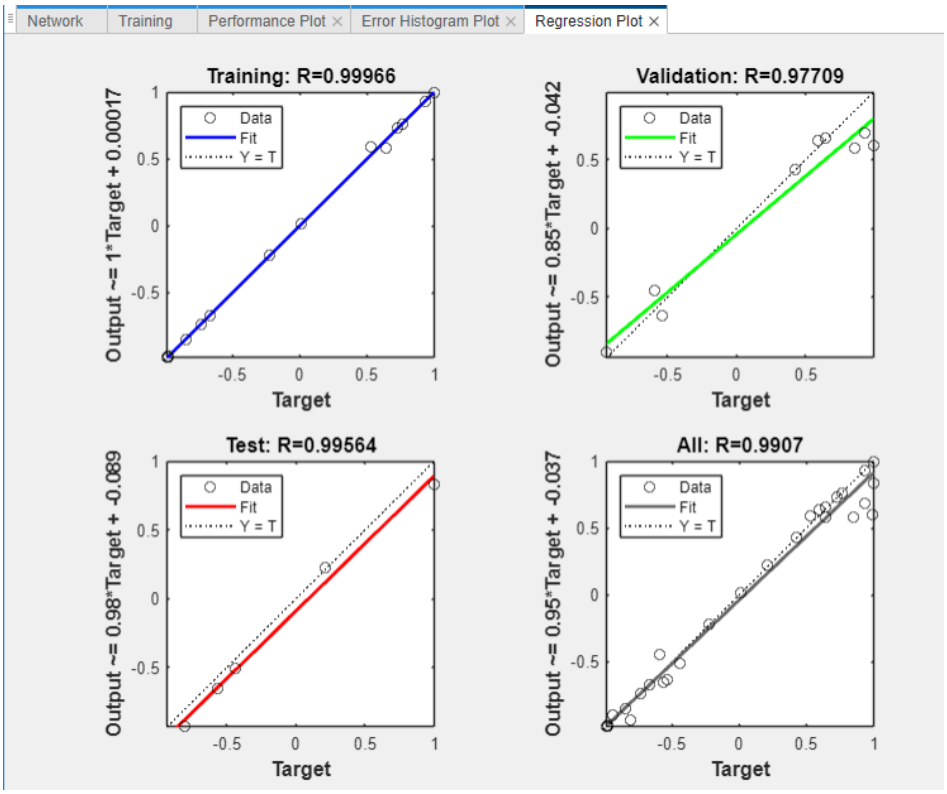


Рис. 6.28. Вікно *Plot Regression* з графічними результатами дослідження моделі

Для подальшого використання створеної нейронної мережі її доцільно експортувати у *Workspace MATLAB*.

Порядок виконання роботи

1. Створити нейромережеву модель залежності частки оксидів азоту в HNO_3 від значень трьох змінних $C = f(V, L, F)$ за методикою, описаною в посібнику.

Студентам треба назвати нейромережу не “*network*1”, а за шаблоном Прізвище_Назва групи (наприклад, Труш_ЛК91).

2. Створити нейромережеву модель для функціональної залежності, вибраної зі списку:

1) $y(x) = 0{,}0001 \cdot x^4 - 0{,}001 \cdot x^3 - 1{,}2 \cdot x^2 + 50 \cdot x + 20, \quad x = -120 \ldots 100;$

2) $y(x) = \exp(-7 \cdot x^2 + 7 \cdot x^{0{,}5}) \cdot \sin(10 \cdot \pi \cdot x), \quad x = 0 \ldots 1:$

3) $y(x) = \sin(2 \cdot \pi \cdot x)^5 + \cos(5 \cdot \pi \cdot x)^3, \; x = 0 \ldots 1;$

4) $y(x) = 1 + 2 \cdot x \cdot \sin(2/x) - 2 \cdot \cos(2/x), \; x = 0{,}1 \ldots 1.$

Номер моделі визначити з таблиці:

Номер моделі	Номер варіанту студента
1	1, 5, 9, 13, 17, 21, 25, 29
2	2, 6, 10, 14, 18, 22, 26, 30
3	3, 7, 11, 15, 19, 23, 27
4	4, 8, 12, 16, 20, 24, 28

Виконати наступні дії:

- підібрати для вибраної функції структуру мережі типу *Feed-forward backprop* та навчити її так, щоб досягти заданої точності апроксимації;

- вивести на екран в одних координатних осях графік функції, що підлягала апроксимації, та значень від створеної нейромережевої моделі.

3. Зробити висновок про необхідні інформаційні ресурси для виконання нейромережевої ідентифікації та її точність.

Вміст звіту:

Для кожного з пп. 1 та 2 Порядку виконання роботи: формулювання завдань; зображення вікон з архітектурами створених нейромереж; графіки навчання нейромереж; графіки еталонної функції та апроксимованої нейромережами. Висновки до п.3.

Контрольні запитання та завдання

1. В якому вигляді необхідно подавати вхідні і вихідні дані в бібліотеках *NNTool* і *NFTool*?

2. Як впливає надмірне узагальнення нейронної мережі на її роботу?

3. Як впливає перенавчання нейронної мережі на її роботу?

4. Як уникнути перенавчання мережі?

5. Як уникнути локальних мінімумів при навчанні мережі?

6. Які функції активації нейронів ви знаєте? Напишіть формули та наведіть графіки.

7. Зобразіть персептрон з 4 входами, 2 виходами, який містить 2 прихованих шари по 3 нейрони в кожному.

ПРАКТИКУМ 7

СТВОРЕННЯ МАСИВІВ ДАНИХ ДЛЯ НЕЙРОМЕРЕЖЕВОЇ ІДЕНТИФІКАЦІЇ ЧАСОВИХ РЯДІВ

Мета роботи: навчитися створювати часові ряди на основі математичних моделей об’єктів керування.

Стислі теоретичні відомості

За допомогою штучних нейронних мереж можна ідентифікувати не лише статичні залежності, але й часові ряди. Обсяг навчальних даних для такої мережі часто більший, ніж отримують дослідним шляхом. У такому разі вдаються до *розширювання*, або *доповнювання*, *даних* (*data augmentation*) [6].

Як приклад розглянемо дослідні дані про осаджування води в нафті після оброблення її електричним полем, створеним між електродами особливого приладу – зневоднювача. Досліди проводили за 4-х різних значень напруги: 0 кВ, 2 кВ, 4 кВ, 6 кВ. Графіки перехідних процесів за каналом «Напруга – Залишкова концентрація води» наведено на Рис. 7.1 [7].

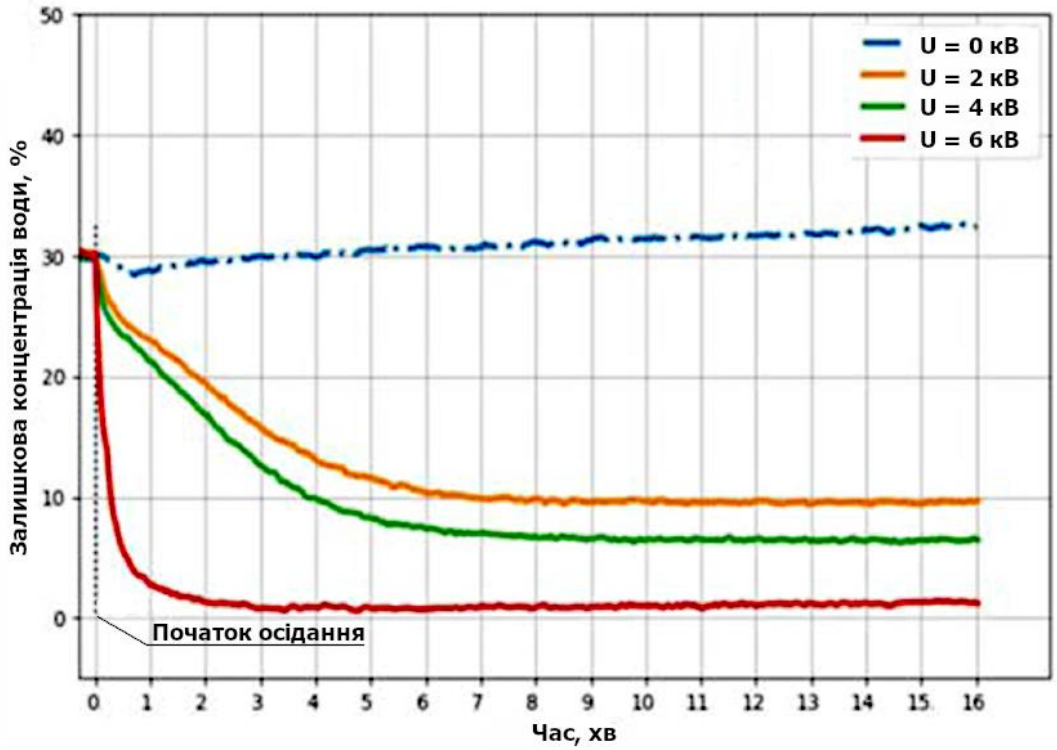


Рис. 7.1. Перехідні процеси для концентрації води в нафті після оброблення електричним полем

Щоб доповнити ці дані, ми скористаємося процесами за напруг 2, 4 й 6 кВ; процес без увімкнення електричного поля (за 0 кВ) ми не користатимемо. Об'єкт за каналом «Напруга – Залишкова концентрація води» поводиться подібно до об'єкта першого порядку, хоча його підсилення та швидкодія, вочевидь, залежать від напруги.

Систему з такими особливостями можна описати різними способами: нелінійним диференційним рівнянням, нелінійною моделлю в просторі станів або ж передавальною функцією першого порядку зі змінними коефіцієнтами. Вдамося до останнього способу, тобто скористаємося передавальною функцією

$$W(s,u) = \frac{K(u)}{T(u)\,s + 1}, \tag{7.1}$$

де s – комплексна змінна, u – вхідна величина об'єкта – напруга між електродами зневоднювача, а $K(u)$ і $T(u)$ – коефіцієнт підсилення та стала часу, відповідно. За допомогою методу найменших квадратів для змінних параметрів $K(u)$ і $T(u)$ отримано значення, наведені в *Таблиця. 7.1*. Ми користатимемо такі самі одиниці вимірювання, що й у [8]. Зрештою, нейронна мережа працюватиме з часом лиш опосередковано, через часові кроки, розмір яких для її роботи значення не матиме.

Таблиця. 7.1. **Значення параметрів моделі за різних значень напруги**

u, кВ	$K(u)$, (% / кВ)	$T(u)$, хв
2	−10,36	2,46
4	−5,94	2,29
6	−4,82	0,26

З навчальною метою ми також вважатимемо, що об'єкт працює не порційно (як це відбувається насправді), а безперервно. У такому разі його властивості з *Таблиця. 7.1* можна використати, щоб створити єдиний процес, у якому час від часу змінюватиметься напруга, й ми відстежуватимемо відгук об'єкта на ці зміни. Для цього скористаємося середовищем *Simulink* і системою, наведеною на Рис. 7.2 і Рис. 7.3.

Зверніть увагу, що схема на Рис. 7.3 містить підсилювач із коефіцієнтом «−1». Крім того, ми не зсували значень вихідної величини на 30 % (початкове значення), тож по суті ми перетворити вихідну величину як

$$y(t) \mapsto 30 - y(t).$$

Перехідний процес починатиметься від нуля і зростатиме в додатний бік. В такому вигляді його зручно сприймати, а також такий вигляд відповідає традиціям. Надалі ми вживатимемо умовну назву «концентрація води».

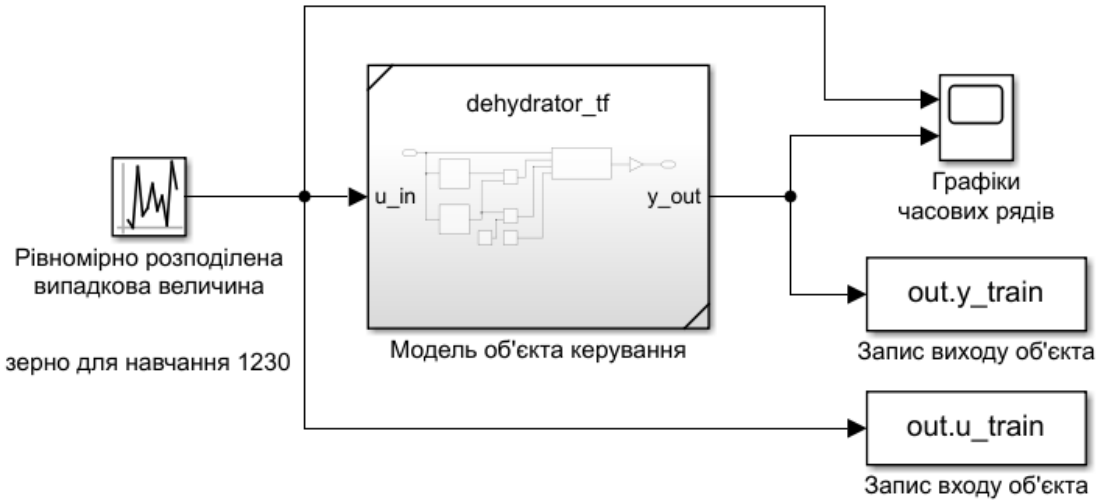


Рис. 7.2. Схема системи для генерування нової вибірки в *Simulink*

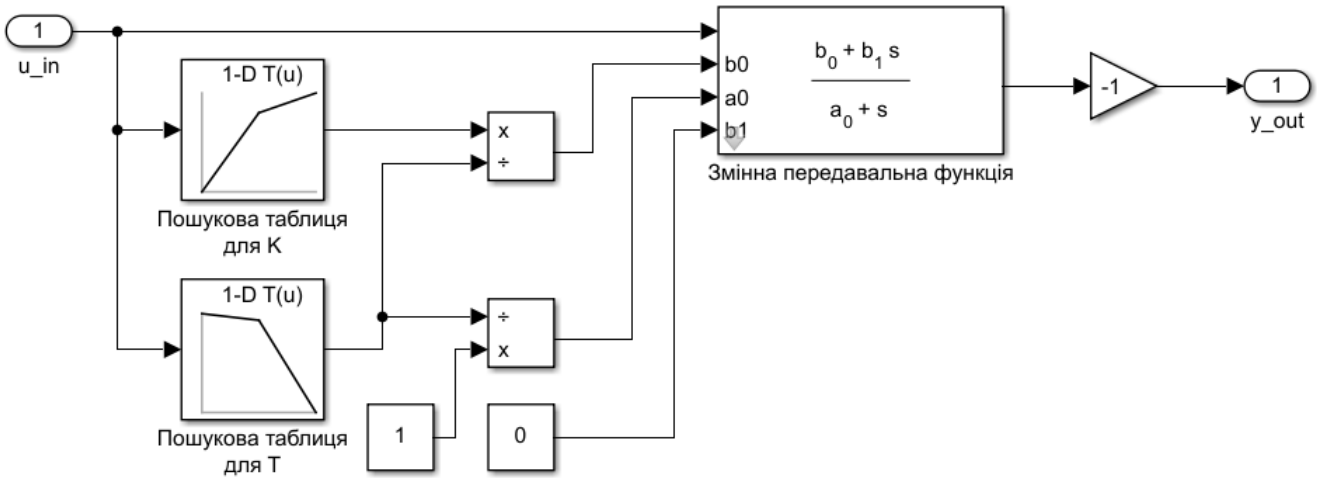


Рис. 7.3. Модель об’єкта керування

Генеруючи часові ряди, що залежать від зовнішніх впливів, з метою ідентифікувати ці ряди, доцільно на вхід системи подавати псевдовипадкові сигнали. Зручний спосіб отримати такий сигнал у *Simulink* – скористатися блоком *Рівномірно розподілене випадкове число* (*Uniform random number*, Рис. 7.2). На Рис. 7.4 наведено вікно з його налаштунками.

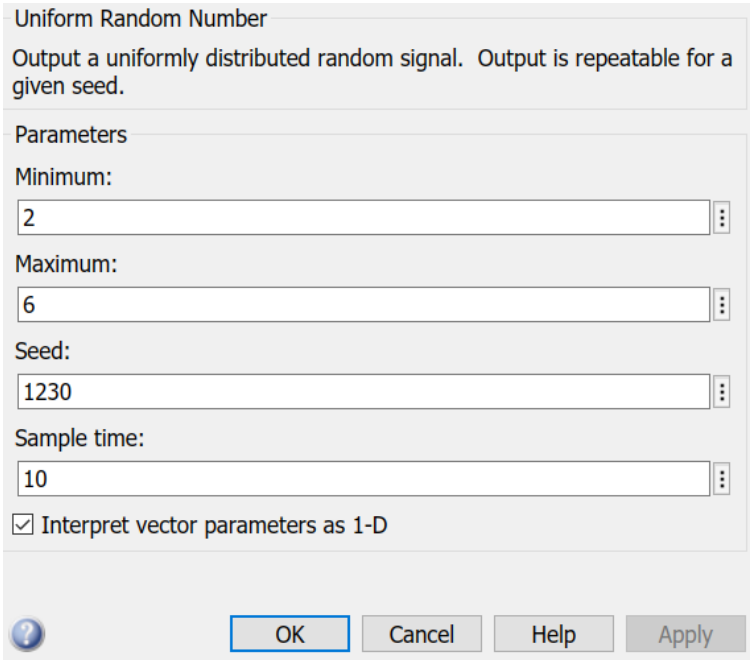
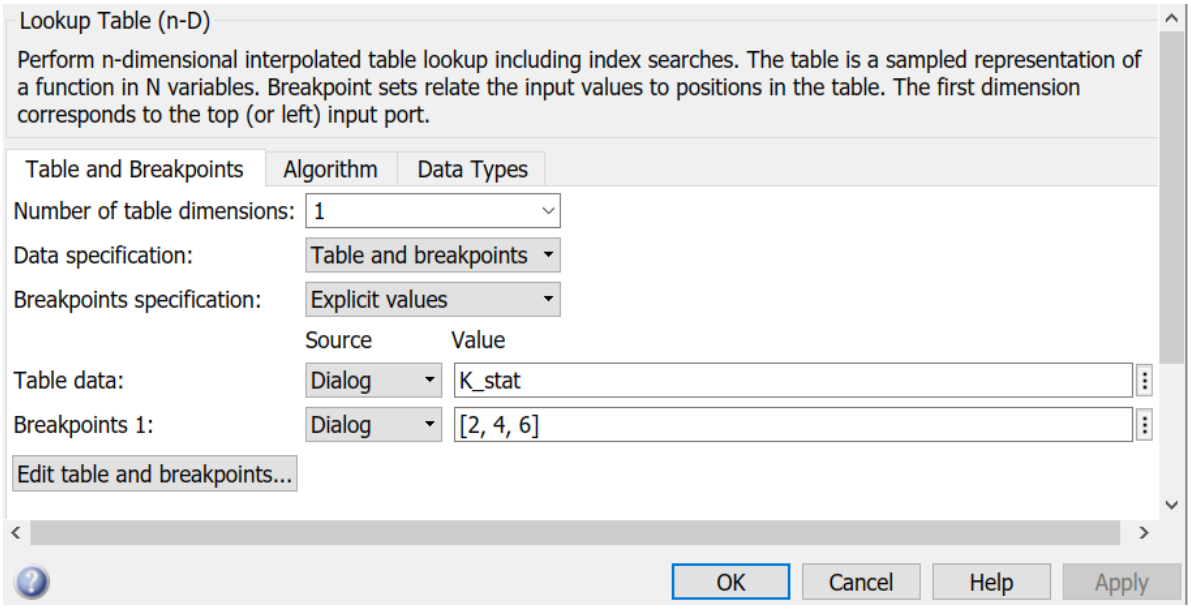


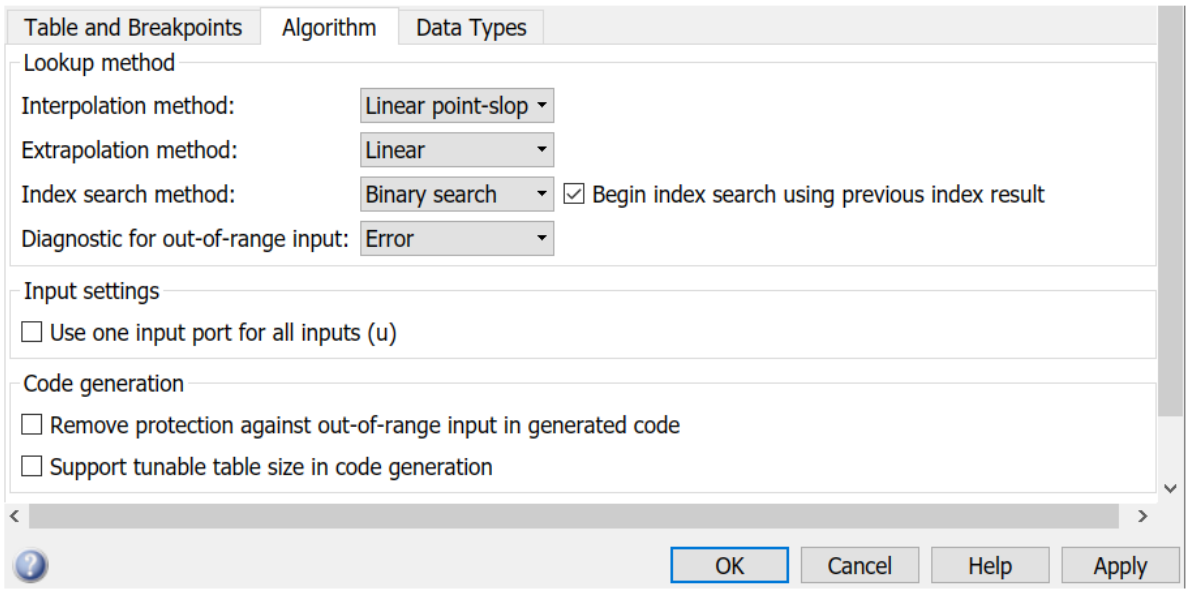
Рис. 7.4. Вікно з налаштунками блоку *Рівномірно розподілене випадкове число*

Верхня й *нижня ме́жі* (*Minimum* і *Maximum*) – це межі проміжку, на якому чи́сла розподілено. *Зерно* (*Seed*) – це число, яке генератори псевдовипадкових чисел «беруть за основу» для послідовності. Його задають насамперед задля відтворності вислідів, тож його можна вибирати майже довільно; єдина вимога – число має бути невід'ємним цілим. *Період квантування* (*Sample time*) – це проміжок часу, впродовж якого вихід блоку буде сталий. Зверніть увагу, що векторні параметри потрібно трактувати як одновимірні («прапорець» унизу вікна).

Блок *Пошукова таблиця* (*Lookup table*, Рис. 7.3) дає змогу розрахувати параметри передавальної функції (7.1) за будь-якого значення входу в досліджуваному проміжку. Для цього блоку ми задали спосіб інтерполювання *Лінійний* (*Linear point-slope*), інші налаштунки залишили без змін. На Рис. 7.5 зображено відповідні вікна. Графіки інтерполювальних функцій зображено на самих блоках (Рис. 7.3). У змінну `K_stat` записано другий стовпець із *Таблиця. 7.1*. Тут і далі `цей шрифт` користатимемо для уривків коду.



а



б

Рис. 7.5. Вікна з налаштунками блоку *Пошукова таблиця* для коефіцієнта підсилення об'єкта K:

а – введення даних; б – алгоритм інтерполювання

Передавальну функцію об'єкта (7.1) втілює блок *Змінна передавальна функція* (*Varying transfer function*, Рис. 7.3). Він дає змогу в дійсному часі врахувати залежність параметрів моделі від вхідного сигналу. Цей блок приймає не лише вхідний сигнал, але й значення параметрів передавальної функції, розраховані на основі інтерполювання.

Нарешті, щоб записати згенеровані процеси у змінні середовища *MATLAB* (тут ця змінна називається `out`), у схемі використано блоки *До робочого простору* (*To workspace*). На Рис. 7.6 наведено вікно з його налаштунками для вхідного сигналу об'єкта, $u(t)$. У разі вихідного сигналу, $y(t)$, іншою буде лише *назва*

змінної (*Variable name*). Зберігати дані доцільно у звичайному *масиві* (*Save format – Array*). Зверніть увагу, що час квантування має бути на два-три порядки менший за час квантування в блоці *Рівномірно розподілене випадкове число*.

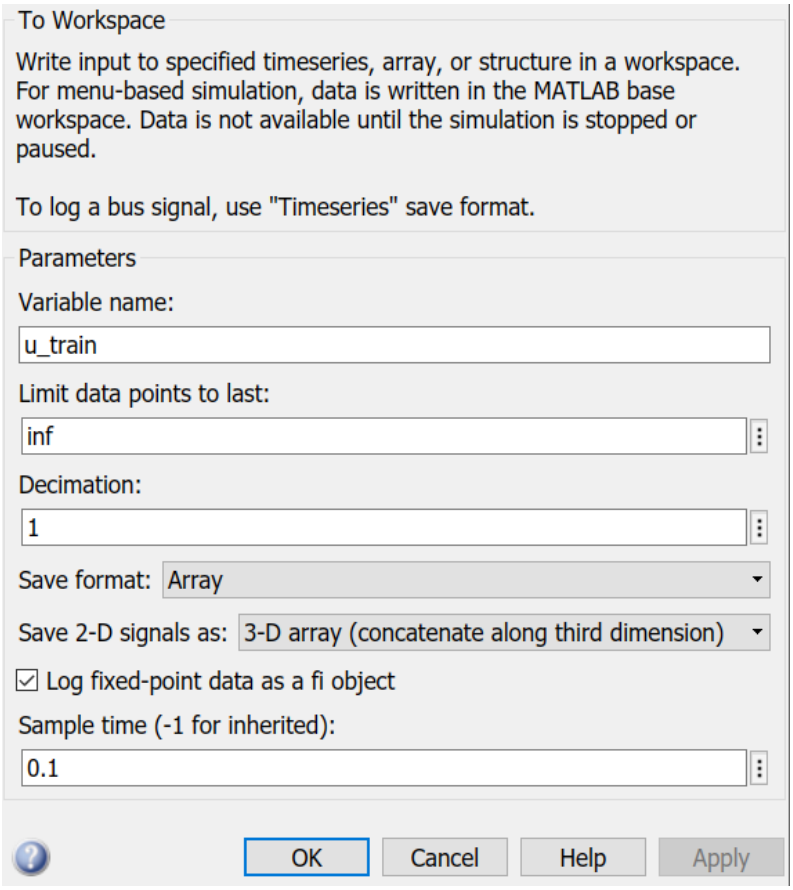


Рис. 7.6. Вікно з налаштунками блоку *До робочого простору* для вхідного сигналу об'єкта, $u(t)$

Попри те, що моделювати систему можна й у вікні *Simulink*, ми радимо робити це за допомогою такого коду, записаного у звичайному *скрипті* (*Script*) або *«живому» скрипті* (*Live script*) *MATLAB*:

```
out = sim("data_augmenter", 300);
u_train = out.u_train;
y_train = out.y_train;
```

Функція `sim` запускає модель *Simulink*. Перший її аргумент – назва файлу, другий – тривалість моделювання у секундах (далі ми трактуватимемо ці одиниці як хвилини). Тривалості 300 хвилин, на нашу думку, достатньо, щоб навчити мережу.

На Рис. 7.7 наведено графіки згенерованих даних (позаяк це не перетворені дослідні дані, а геть нові процеси, то ми їх називаємо саме *згенерованими*, а *не розширеними*). Вхідна змінна – напруга – змінюється східчасто щодесять хвилин.

За цей час вихідна змінна – концентрація води – встигає достатньо наблизитись до відповідного усталеного значення. Крок за часом становить 0,1 *хвилини* (6 секунд).

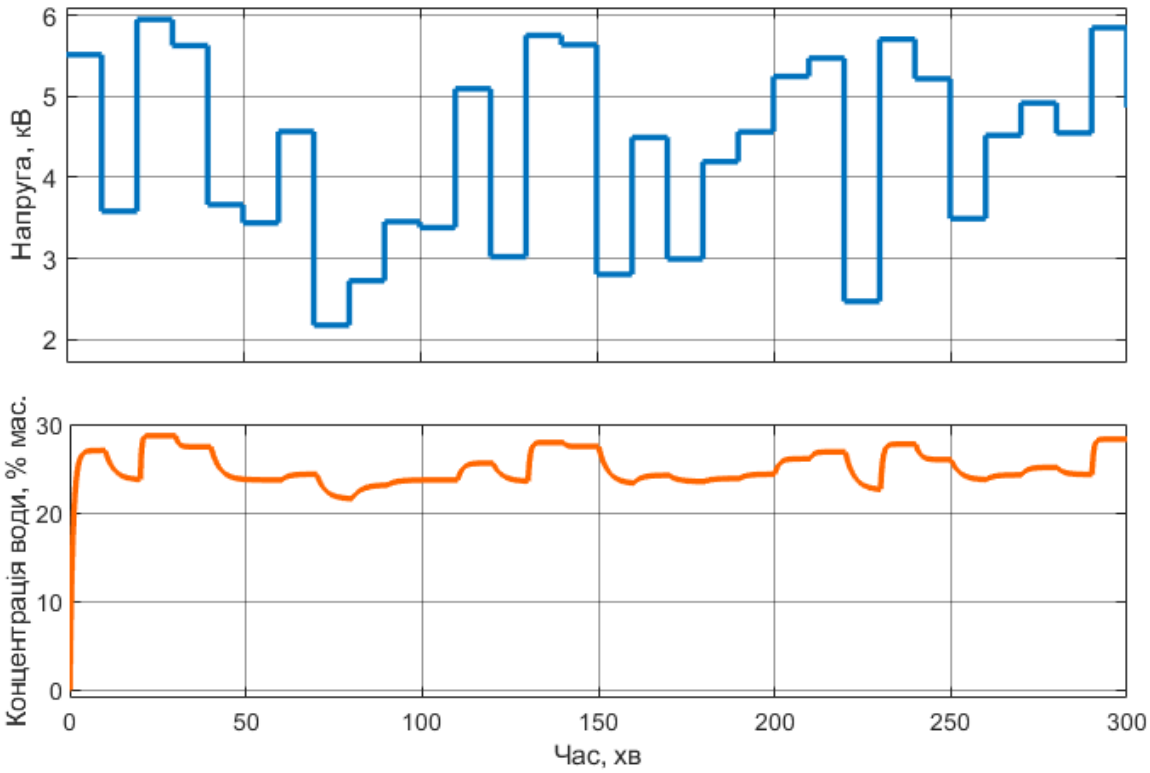


Рис. 7.7. Навчальні часові ряди: для вхідного сигналу (вгорі) та для вихідного сигналу (внизу)

Щоб отримані дані можна було використати будь-коли після закриття *MATLAB*, не обчислюючи їх заново, їх слід зберегти у файл. Для цього потрібно виконати рядок коду

```
save('filename.mat', 'u_train', 'y_train');
```

де *filename* – назва файлу, після якої через кому можна вказати довільну кількість назв змінних із робочого простору, котрі ми хочемо зберегти. Якщо їх не вказувати, *MATLAB* збереже *всі* наявні змінні. Тут ми зберігаємо лише один процес.

Передавальну функцію, хай і зі змінними параметрами, можна отримати не завжди. Тож розгляньмо тепер, як у *Simulink* можна розв'язувати нелінійні диференційні рівняння. Для прикладу візьмемо таке рівняння:

$$10\,y''(t) + 2\,y'(t) + (y(t) + 0{,}5)^2 = 20\,u(t). \qquad (7.2)$$

Щоб зібрати відповідну схему в *Simulink*, потрібно спершу розв'язати це рівняння відносно старшої похідної. У такому разі воно набуде вигляду

$$y''(t) = 2\,u(t) - 0{,}1\,(y(t) + 0{,}5)^2 - 0{,}2\,y'(t).$$

Зверніть увагу, що коли рівняння відносно старшої похідної розв'язати неможливо, то цей підхід не працюватиме.

Тепер зберемо в *Simulink* схему, зображену на Рис. 7.8. Її вхідні сигнали – вхід об'єкта, $u(t)$, а також його вихід, $y(t)$, і його перша похідна, $y'(t)$, а вихідний – друга похідна виходу, $y''(t)$. Щоб піднести сигнал до квадрату, потрібно використати блок *Математична функція* (*Math function*) і в його налаштунках вибрати відповідну функцію – *квадрат* (*square*).

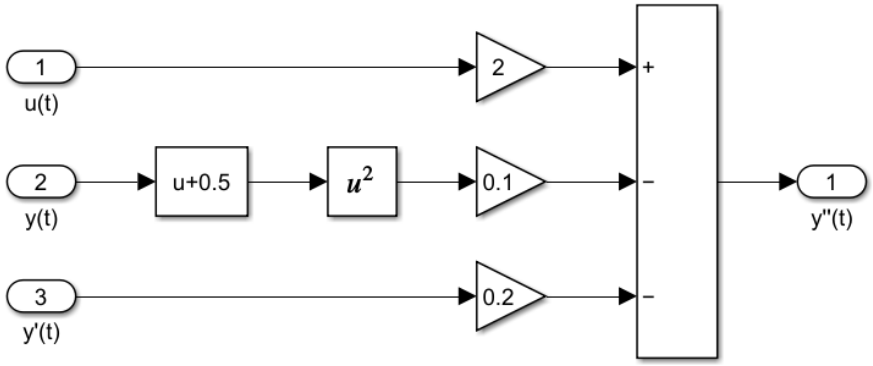


Рис. 7.8. Підсистема для обчислювання $y''(t)$

Тепер потрібно виділити всю створену систему та в підказці, що з'явиться, вибрати *Створити підсистему* (*Create subsystem*). Всередині підсистеми схема залишиться така сама, як на Рис. 7.8. Ззовні ж видалимо всі елементи, крім самої підсистеми, та зберемо схему, зображену на Рис. 7.9.

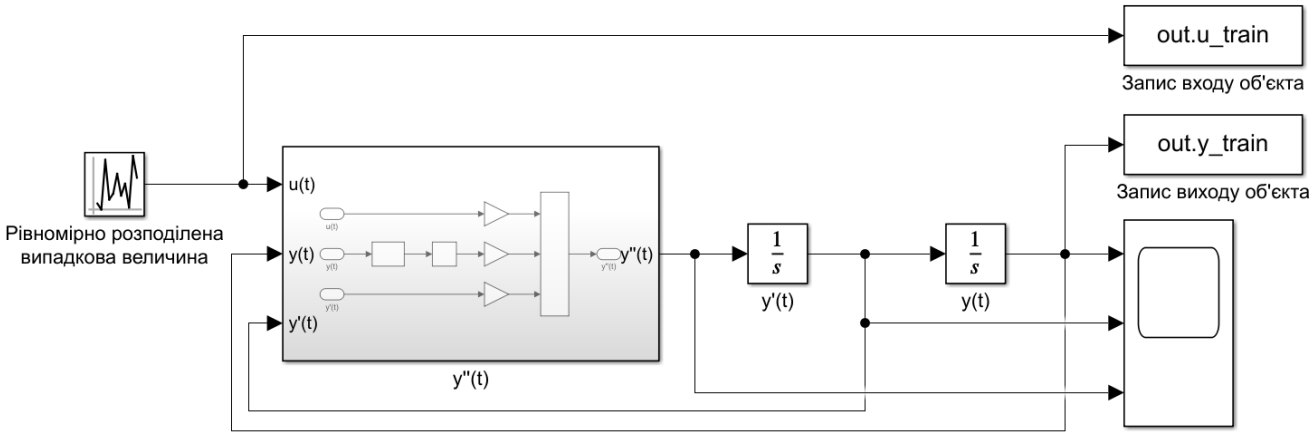


Рис. 7.9. Система для розв'язування диференційного рівняння

Щоб із другої похідної на виході підсистеми отримати сам сигнал, $y(t)$, ми додали два блоки інтегрування. Зворотний зв'язок від них надходить у

підсистему. Також, задля наочності, ми будуємо графіки не лише для самого сигналу, але й для двох його похідних. Налаштунки блоків *Рівномірно розподілене випадкове число* і *До робочого простору* такі самі, що й раніше. Щоб обмежити період квантування, що його неявно користає *Simulink* для неперервних систем, ми задали цей період у блоці вхідного сигналу, а саме 0,01 секунди.

Додамо до рівняння (7.2) початкові умови:

$$\begin{cases} y(0) = 2 \\ y'(0) = -1 \end{cases}$$

Виходи підсистеми, що на Рис. 7.8, будуть спершу нульовими. Щоб урахувати початкові умови, в налаштунках блоків інтегрування потрібно задати їх явно, в рядку *Початкова умова* (*Initial condition*, Рис. 7.10).

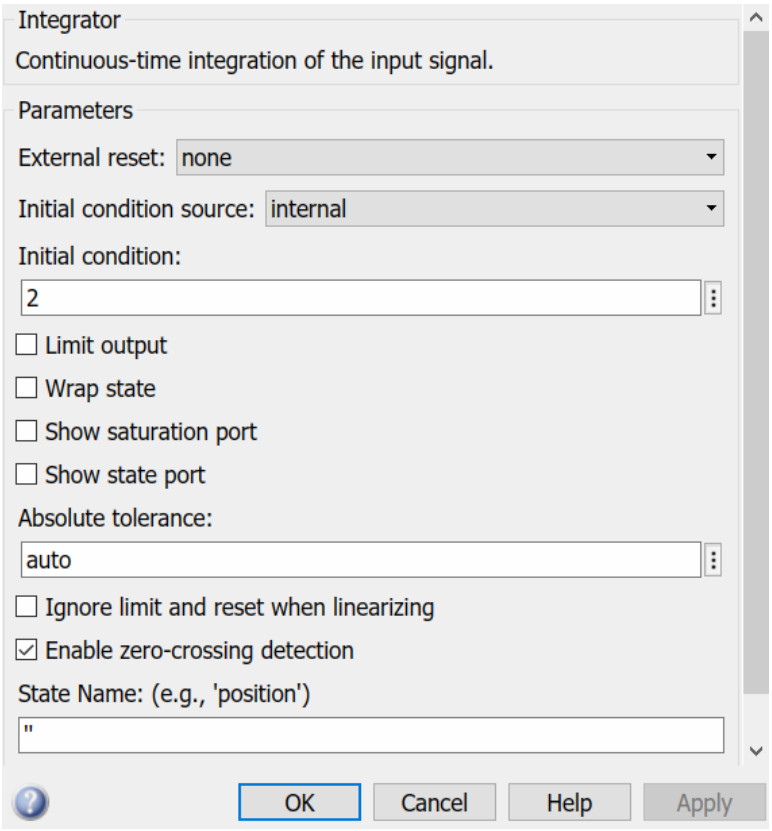


Рис. 7.10. Вікно з налаштунками блоку *Інтегрування* для сигналу $y(t)$

На Рис. 7.11 зображено перехідні процеси в системі: вихідний сигнал та дві його похідні.

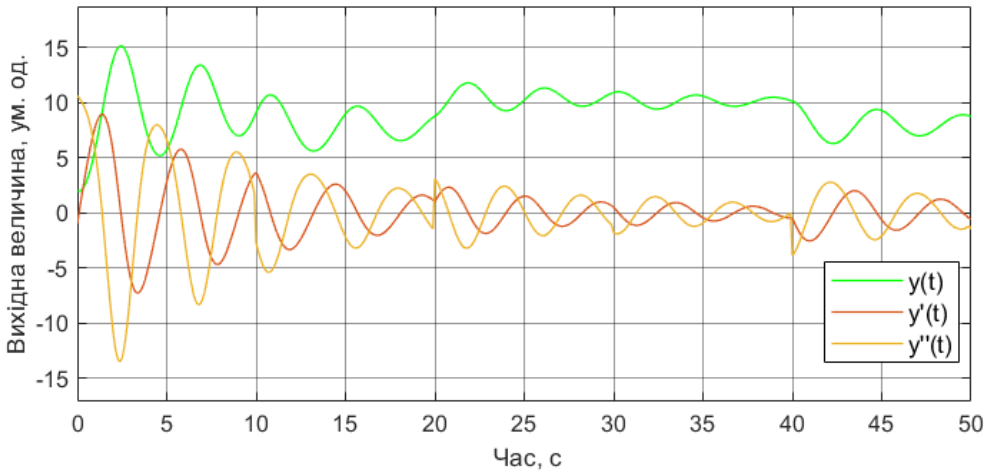


Рис. 7.11. Перехідні процеси в системі

З рисунку видно, що вихідний сигнал системи не встигає вийти на усталене значення за такого періоду квантування блоку *Рівномірно розподілене випадкове число*, тож прийнятне значення, за якого вихід встигатиме, для цієї системи ще слід буде визначити.

Насамкінець зауважимо, що в *Simulink* блок *Передавальна функція* та похідні від нього завжди мають нульові початкові умови. Більшу гнучкість мають моделі в просторі станів, представлені, зокрема, блоками *Простір станів* (*State space*) та *Змінний простір станів* (*Variable state space*). У цих та інших подібних блоках можна задати початкові умови для кожного стану. По докладні відомості радимо звернутись до офіційної документації на вашу версію *MATLAB*.

Порядок виконання роботи

1. Записати математичну модель свого об’єкта керування як диференційне рівняння, передавальну функцію або модель у просторі станів. Початкові умови й межі змінювання вхідного сигналу визначити на свій розсуд. Якщо моделі нема, тоді відповідно до варіанту вибрати одне з наведених нижче диференційних рівнянь та умови до нього:

а)
$$\begin{cases} 10\,y'(t) + 0{,}9\sqrt{y(t) - 0{,}2} + 0{,}5\sqrt{y(t) - 0{,}5} = 2\,u(t) - 0{,}1, \\ y(0) = 1, \\ 0{,}5 \le u(t) \le 2. \end{cases}$$

б)
$$\begin{cases} \big(20 - 8\,u(t)\big)\,y'(t) + y(t) = \big(10 - 3\,u(t)\big)\,u(t), \\ y(0) = 0, \\ 0 \le u(t) \le 2. \end{cases}$$

в)
$$\begin{cases} 2000\,y''(t) + (440\,u(t) + 25000)\,y'(t) + (6\,u(t) + 20)\,y(t) = \\ = 200\,u(t) + 500, \\ y(0) = 35, \\ 10 \le u(t) \le 100. \end{cases}$$

г)
$$\begin{cases} 0{,}005\,y'(t) + \left(0{,}75\,\mathrm{acos}\big(2\,u(t)\big) - 3\,u(t)\sqrt{0{,}25 - u(t)^2}\right)\,y(t) = 2000, \\ y(0) = 500, \\ -0{,}45 \le u(t) \le 0{,}45. \end{cases}$$

2. Зібрати в *Simulink* схему системи, за допомогою якої будете розраховувати перехідні процеси.
3. Самостійно визначити прийнятні *тривалість моделювання* (*Stop time*; її задають угорі вікна моделі), період квантування *вхідного* сигналу (*Sample time*), за якого вихід об’єкта щоразу встигатиме досягнути усталеного рівня, та період квантування *вихідного* сигналу (період записування до робочого простору), який дасть змогу достатньо точно відтворити перехідні процеси. Тривалість моделювання та період квантування вхідного сигналу мають співвідноситися так, щоб отриманий процес містив від 10 до 20 змін вхідної величини (Рис. 7.7).

4. Задати довільне зерно у блоці *Рівномірно розподілене випадкове число*. Розрахувати перехідний процес. Вхідний і вихідний сигнали записати у змінні `u_train` та `y_train`, відповідно.
5. Задати інше зерно й розрахувати нові перехідні процеси. Записати отримані значення у змінні `u_test` та `y_test`.
6. Переконатися, що всі отримані часові ряди записано як вектори-стовпці. Якщо ні, то подати їх у такому вигляді.
7. Зберегти масиви `u_train`, `y_train` (перший процес) та `u_test`, `y_test` (другий процес) до файлу з назвою `Прізвище_група.mat`. Пересвідчитися, що всі змінні збереглися, відкривши цей самий файл із середовища *MATLAB*.

Вміст звіту

- Математична модель об’єкта, початкові умови, $y(0)$, та межі змінювання вхідного сигналу, $u(t)$.
- Схему системи в *Simulink*, за допомогою якої розрахували перехідні процеси.
- Зе́рна для генерування псевдовипадкових сигналів у обох процесах.
- Графіки входу й виходу об’єкта в обох процесах.

Контрольні запитання та завдання

1. Чому може виникнути потреба згенерувати нові (штучні) дані на основі дослідних даних?
2. Для чого в роботі потрібен блок Рівномірно розподілене випадкове число? Які параметри в ньому можна налаштувати?
3. Що таке зерно для генерування псевдовипадкових числових послідовностей? Для чого його користають?
4. Який загальний спосіб розв’язування диференційних рівнянь у *Simulink*? Яка необхідна умова його застосовності?
5. Як урахувати ненульові початкові умови в моделі *Simulink*?

ПРАКТИКУМ 8

НЕЙРОМЕРЕЖЕВА ІДЕНТИФІКАЦІЯ ЧАСОВИХ РЯДІВ

Мета роботи: навчитися створювати нейронні мережі в застосунку *NTSTool* і користати їх для ідентифікації та передбачування часових рядів.

Стислі теоретичні відомості

У попередній роботі ми створили набори даних для навчання й випробування штучної нейромережі. Тепер потрібно, власне, створити мережу та навчити її.

Для роботи користатимемо середовище *MATLAB* (*2021a*) і застосунок *NTSTool* (*Neural network Time Series Tool*, нейромережеве знаряддя для часових рядів). Значок застосунку можна знайти на вкладці *Застосунки* (*Apps*), у розділі *Машинне та глибинне навчання* (*Machine learning and Deep Learning*). Щоб запустити застосунок із командного рядка, потрібно виконати команду `ntstool`. Початкове вікно наведено на Рис. 8.1.

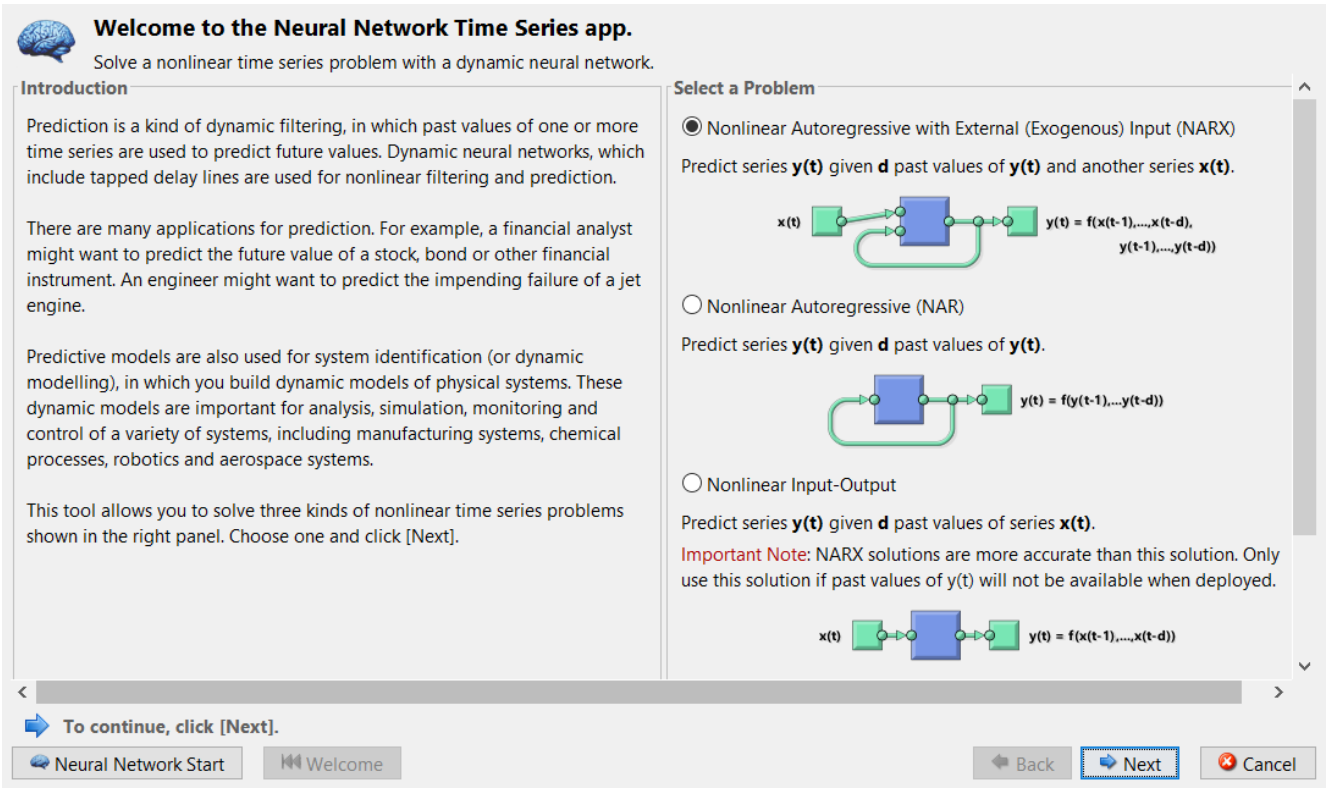


Рис. 8.1. Початкове вікно *NTSTool*

Спершу потрібно вибрати вид моделі. Найзагальніша – *нелінійна саморегресійна із зовнішнім входом* (*nonlinear autoregressive exogeneous*, *NARX*). Щоб обчислити значення виходу об'єкта в наступну мить, вона користає поточне й попередні значення виходу об'єкта та його входу. Також доступні окремі випадки: *нелінійна саморегресійна* (*nonlinear autoregressive*, *NAR*) і *нелінійна вхід-вихід* (*nonlinear input-output*; інша назва – *модель зі скінченним відгуком*, *finite input response*, *FIR*). Навчальні дані, що ми використаємо, відповідають моделі *NARX*, тож обираємо її. Вираз для передбачення наступного значення виходу системи буде

$$y(t) = f\big(y(t-1), \dots, y(t-d), u(t-1), \dots, u(t-d)\big),$$

де t – номер кроку, d – порядок дискретної моделі (кількість попередніх кроків, які буде використано для передбачування). Натискаємо на кнопку *Далі* (*Next*).

На Рис. 8.2 зображено вікно для завантажування навчальних даних. Відкривши випадні списки *Входи* (*Inputs*) й *Мети* (*Targets*), можна вибрати масиви з робочого простору *MATLAB*. Натиснувши на кнопки з трьома крапками, можна завантажити дані зі збереженого завчасно файлу.

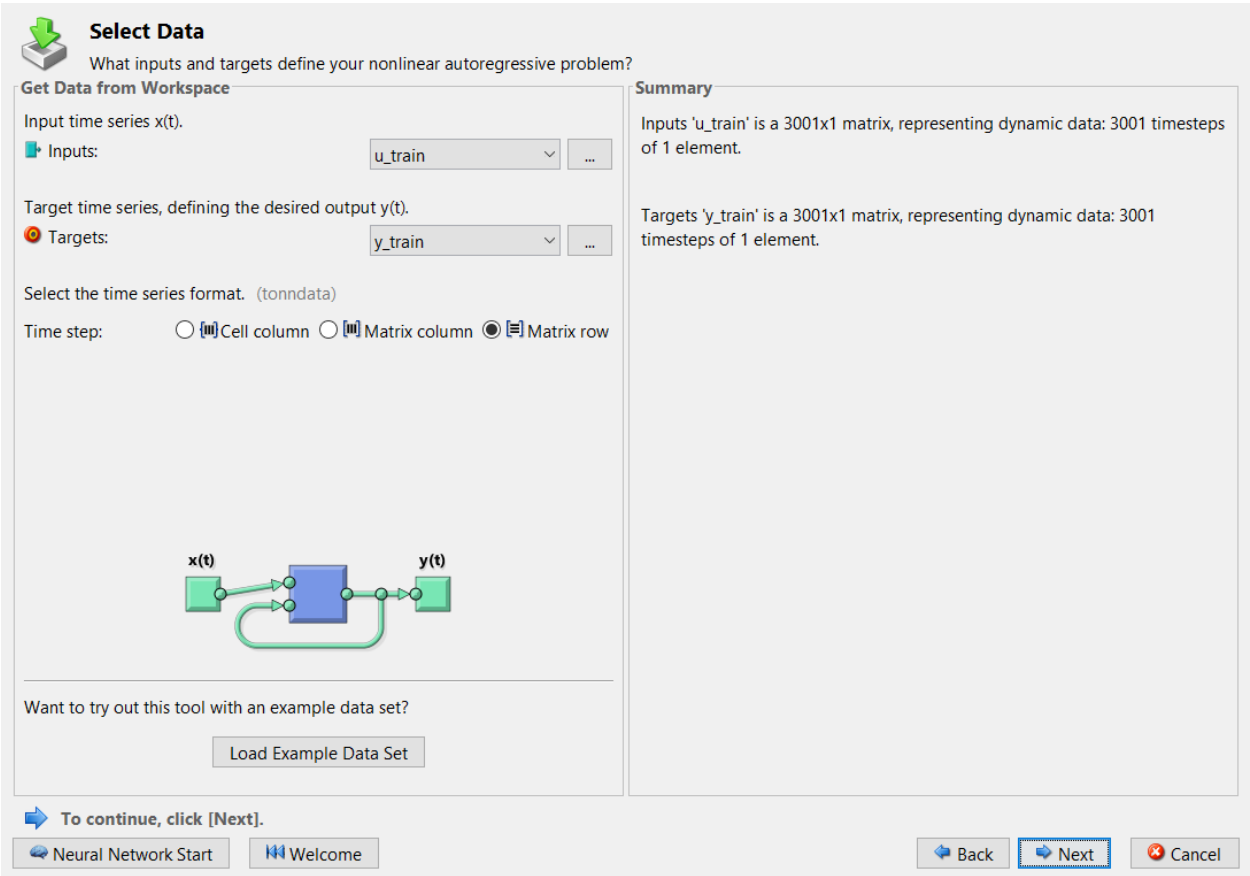


Рис. 8.2. Вікно для завантажування навчальних даних

Щоби скористатися змінними, збереженими у файлі, доцільно їх спершу завантажити до робочого простору *MATLAB*. Для цього слід виконати команду

```
load('filename.mat')
```

із відповідною назвою файлу. Після цього у випадних списках вікна (Рис. 8.2) вибираємо для входів і мет змінні `u_train` та `y_train`, відповідно.

Останній важливий крок – вказати, як дані впорядковані. В нашому разі сукупності вхідних і вихідних змінних – вектори-стовпці, а окремі значення у часі (*часокроки*, *timesteps*) – елементи цих векторів-стовпців, тобто їхні рядки. Вибираємо варіант *Рядок матриці* (*Matrix row*). Щоб перевірити, як система трактує дані, можна прочитати розгорнутий опис праворуч. Маємо такі описи: «3001 часокрок для 1 входу» та «3001 часокрок для 1 виходу» (в описі англійською *входи* й *виходи* об'єкта названо *елементами*). Подання правильне. Натискаємо *Далі*.

Наступне вікно (Рис. 8.3) призначене для розділювання даних на три підвибірки: *навчальну* (або *тренувальну*, *training*), *перевірну* (або *валідаційну*, *validation*) й *випробну* (або *випробувальну*, або *тестову*, *test*). Початковий розподіл – 70 %, 15 % і 15 %, відповідно. Його можна не змінювати. Натискаємо *Далі*.

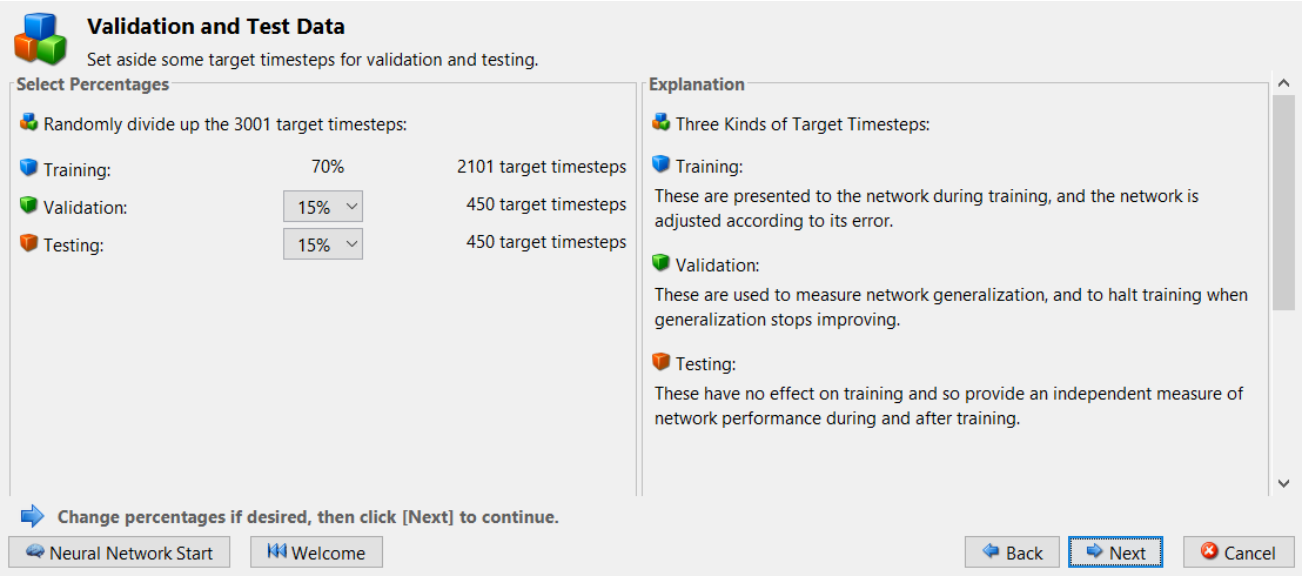


Рис. 8.3. Вікно для розділювання даних на навчальну, перевірну й випробну підвибірки

Тепер потрібно визначити *надпараметри*, або *архітектуру*, мережі (*Network Architecture*, Рис. 8.4): кількість нейронів у прихованому шарі й порядок моделі *d*. Кількість нейронів залишаємо без змін, а для *d* задамо значення 1, що відповідає приблизній моделі, за допомогою якої ми генерували ці дані [8]. Натискаємо *Далі*.

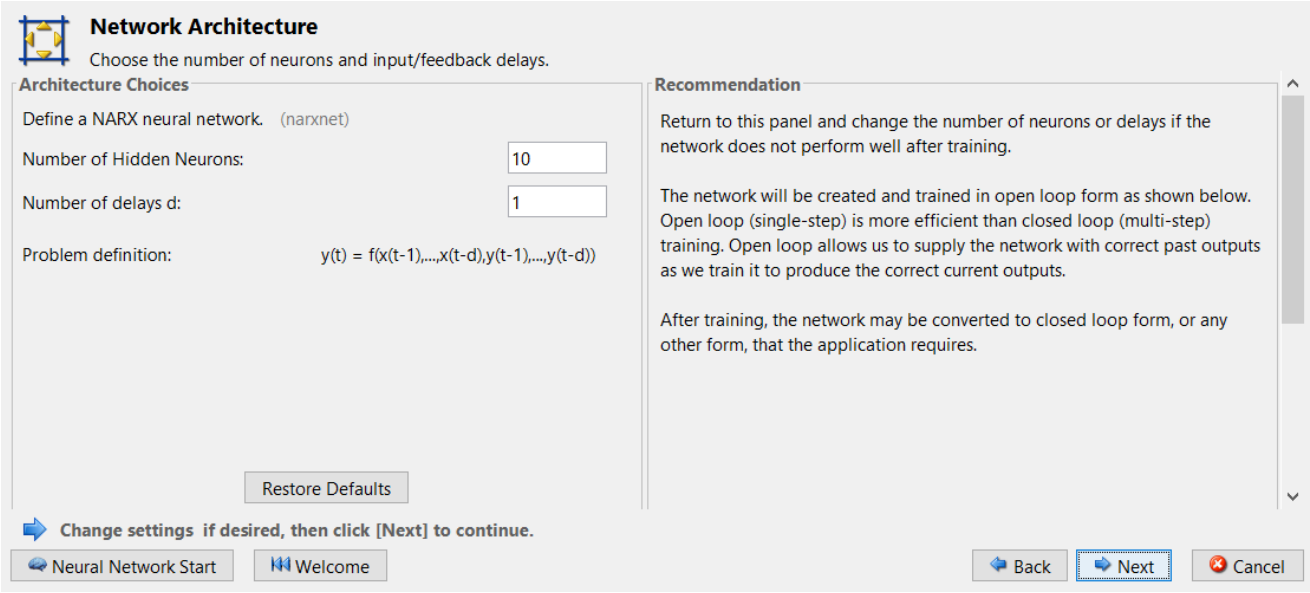


Рис. 8.4. Вікно з вибором надпараметрів мережі

Наступне вікно (Рис. 8.5) присвячене вибору методу навчання мережі. Застосунок пропонує три методи: Левенберґа-Марквардта, Баєсової регуляризації, масштабованого спряженого градієнта.

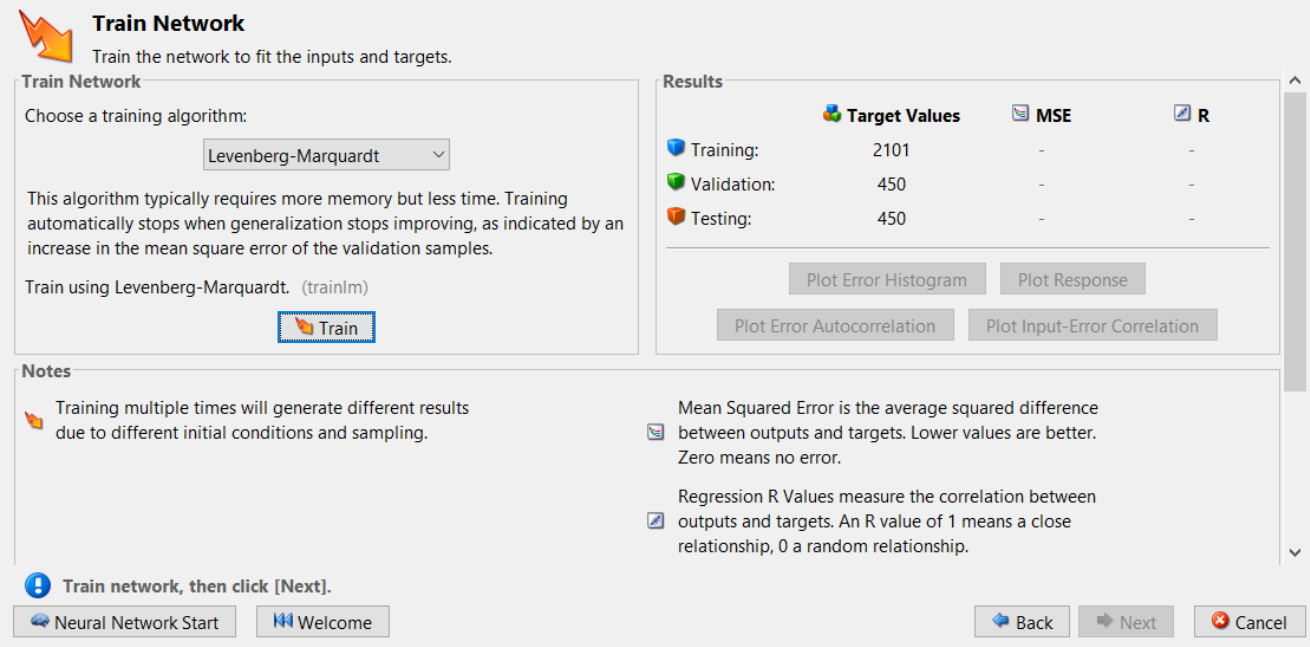


Рис. 8.5. Вікно з вибором способу навчання

Після вибору кожного з методів буде наведено його стислий опис та пояснення особливостей роботи. Вибираємо, до прикладу, метод Баєсової регуляризації й натискаємо *Навчити* (*Train*).

З’явиться вікно з відомостями про перебіг процесу навчання (Рис. 8.6). У вікні в дійсному часі висвітлюватимуться подробиці, як-от: пройдена кількість

епох, тривалість навчання, значення функції втрати, величина градієнту тощо. У верхній частині вікна зображено архітектуру мережі. Зауважимо, що збільшити кількість епох навчання в застосунку неможливо, зате можна передчасно *зупинити навчання*, натиснувши на відповідну кнопку (*Stop Training*), розміщену внизу вікна.

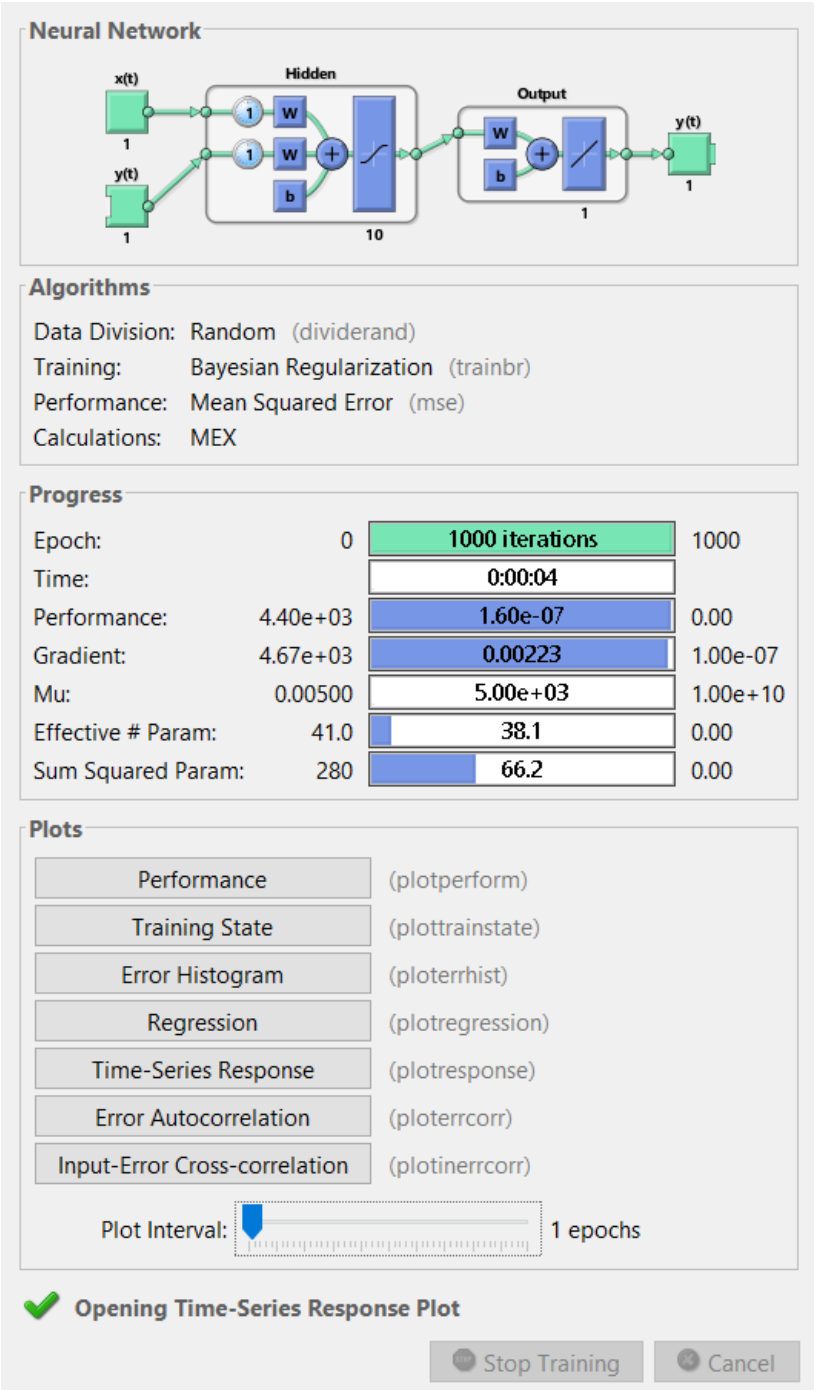


Рис. 8.6. Вікно з відомостями про навчання мережі

Також у нижній частині розміщено кнопки, що дають змогу унаочнити показники навчання, а поруч із ними в дужках указано назви відповідних

функцій, які для цього можна використати в командному рядку. Виведемо перехідний процес і похибку передбачення (Рис. 8.7), натиснувши на кнопку *Передбачення для часового ряду* (*Time-Series Response*).

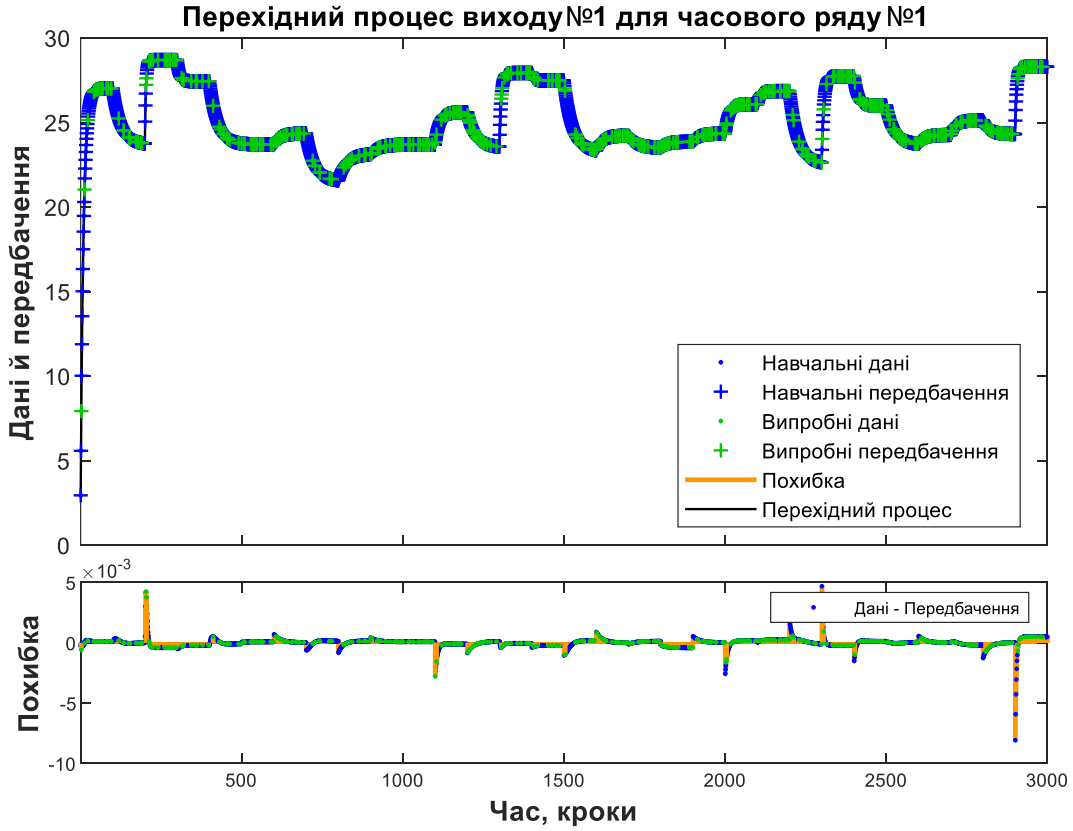


Рис. 8.7. Передбачення мережі на навчальній і випробній вибірках: навчальні дані та передбачення мережі (вгорі); похибки передбачення мережі (внизу)

З першого графіка на Рис. 8.7 бачимо, що дані й передбачення практично збігаються. Це підтверджує другий графік, із якого видно, що похибка хоч і має помітні стрибки на початку зміни вихідної величини, залишається дуже малою (меншою за 0,01 % мас.). Зверніть увагу, що якщо користати інші два навчальні алгоритми, то графіки міститимуть і перевірні дані. Вважаючи, що мережа навчилася достатньо добре, повертаємося до основного вікна (Рис. 8.5) та натискаємо *Далі*.

У наступному вікні (Рис. 8.8) запропоновано різні напрями подальшого вдосконалювання мережі: навчити заново (кнопка *Train Again*), змінити надпараметри (кнопка *Adjust Network Size*), завантажити більшу вибірку (кнопка *Import Larger Dataset*), випробувати мережу на інших даних (кнопка *Test Network*). Натискаємо *Далі*.

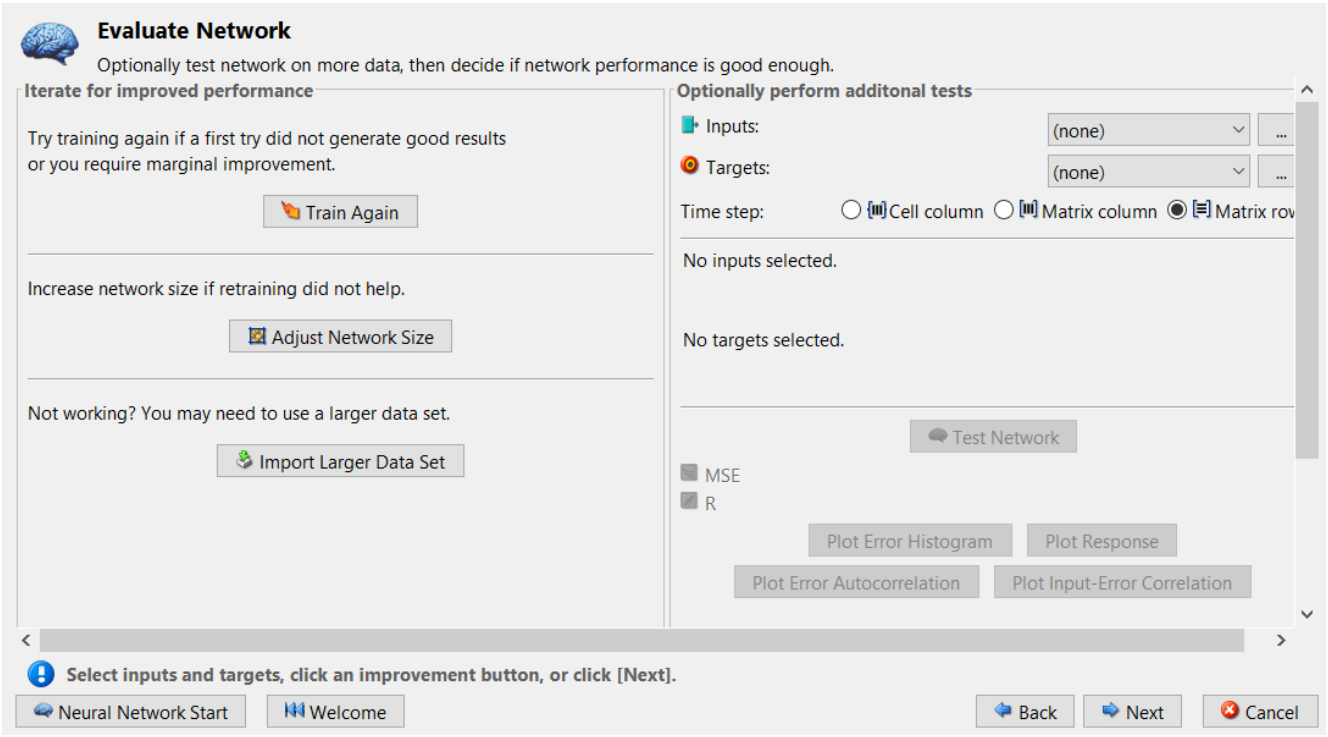


Рис. 8.8. Вікно для доопрацювання мережі

Наступне вікно (Рис. 8.9) пропонує розгорнути навчену нейромережу, щоб і надалі користуватися нею за межами застосунку *NTSTool*. Вибираємо варіант *Функція MATLAB* (*MATLAB Function*). Це створить файл із готовим програмним кодом, що втілює навчену мережу, та поясненнями до нього. Зберігаємо його як `NN_bayes.m`. Щойно створену функцію можна викликáти, подаючи в неї дані про вхідну й вихідну величини модельованої системи, за якими вона передбачить подальші значення виходу.

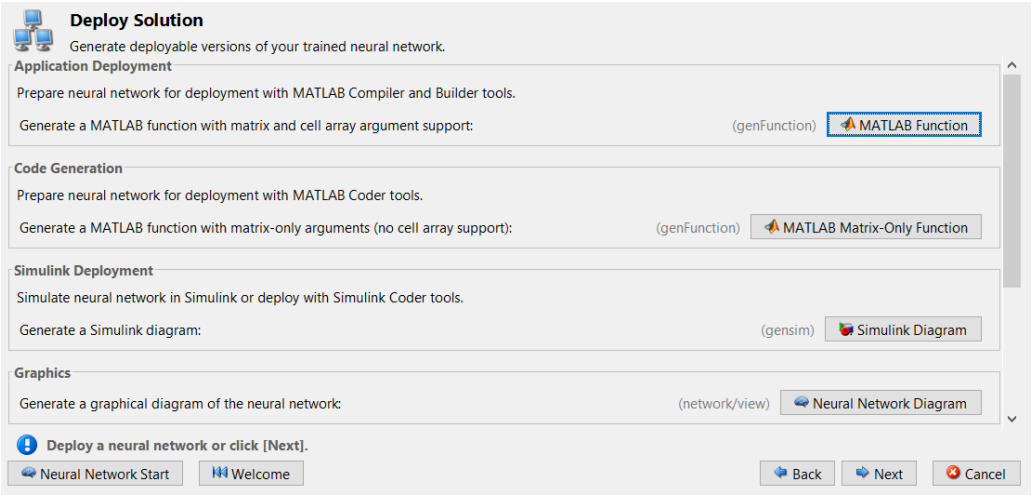


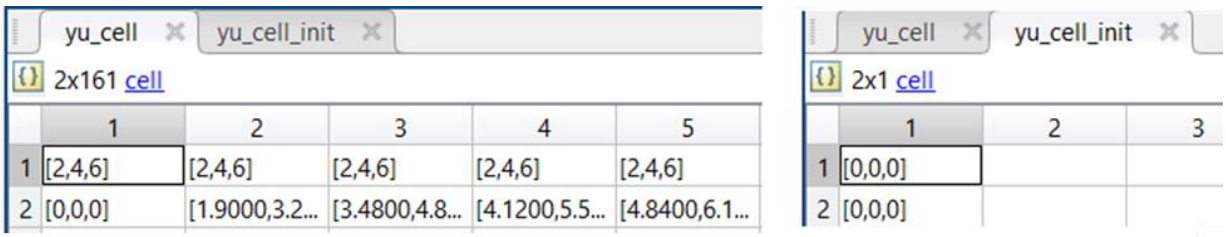
Рис. 8.9. Вікно для розгортання навченої мережі

Автоматично створена документація (Рис. 8.10) містить відомості про аргументи функції та значення, що вона повертає. Зверніть увагу, що всі ці величини – не звичайні масиви, а коміркові.

```
% [Y,Xf,Af] = myNeuralNetworkFunction(X,Xi,~) takes these arguments:
%
%   X = 2xTS cell, 2 inputs over TS timesteps
%   Each X{1,ts} = 1xQ matrix, input #1 at timestep ts.
%   Each X{2,ts} = 1xQ matrix, input #2 at timestep ts.
%
%   Xi = 2x1 cell 2, initial 1 input delay states.
%   Each Xi{1,ts} = 1xQ matrix, initial states for input #1.
%   Each Xi{2,ts} = 1xQ matrix, initial states for input #2.
%
%   Ai = 2x0 cell 2, initial 1 layer delay states.
%   Each Ai{1,ts} = 10xQ matrix, initial states for layer #1.
%   Each Ai{2,ts} = 1xQ matrix, initial states for layer #2.
%
```

Рис. 8.10. Уривок документації до функції, що втілює навчену нейронну мережу

Випробуємо мережу на дослідних даних (безпосередньо на них мережу ми не навчали). Для цього дані потрібно записати в комірковий масив завбільшки 2×161 комірку, а кожна комірка містить масив завбільшки 1×3 (Рис. 8.11). Тут 161 – кількість часокроків у процесах, а 2 – сукупна кількість входів і виходів об'єкта. Перший рядок *коміркового* масиву містить значення входу об'єкта, а другий рядок – його виходу. Кожне з чисел звичайного масиву, що міститься в комірці, відповідає одному з трьох процесів: за напруги 2 кВ, 4 кВ і 6 кВ, відповідно. У вас кожна комірка масиву `yu_cell` міститиме лише одне число.



а б

Рис. 8.11. Приклад коміркового масиву з даними для нейромережі:
повні відомості про процес (а) і початкові умови (б)

Другий аргумент функції – комірковий масив, що містить початкові умови. Устрій масиву такий самий, а кількість його стовпців залежить від порядку різницевих рівнянь, використаних у нейромережі. Ми використали рівняння першого порядку, тож і початкові умови мають містити лише одне число для

входу та для виходу. Загалом же кількість стовпців цього коміркового масиву має дорівнювати порядку рівняння *d*. Ми радимо вам за початкові умови взяти потрібну кількість значень із випробних даних. Зверніть увагу, що в такому разі випробні дані вже не повинні містити цих значень.

Щоб із *одновимірних* масивів `u_test` та `y_test` (ваші випробні дані) отримати потрібні коміркові масиви, слід виконати такий код (*d* – порядок моделі):

```
yu_cell = cell([2, size(u_test, 1) - d]);
for i = (1 + d): size(y_test, 1)
    yu_cell{1, i - d} = u_test(i);
    yu_cell{2, i - d} = y_test(i);
end
yu_cell_init = cell([2, d]);
for i = 1: d
yu_cell_init{1, i} = u_test(i);
yu_cell_init{2, i} = y_test(i);
end
```

Розрахуємо передбачення мережі для процесу, виконавши такий рядок (на́зви функції та змінних у вас будуть інші):

```
[y_pred_cell, ~, ~] = NN_bayes(yu_cell, yu_cell_init);
```

Тут нам потрібні лише передбачення мережі, тож інші два виходи ми просто відкидаємо (знак «~» замість назви змінної – це особливість синтаксису мови *MATLAB* для таких випадків). Крім того, ми не передаємо у функцію необов’язковий третій аргумент.

Отримані передбачення можна використати як завгодно. До прикладу, побудуємо суміщені графіки даних і передбачень та похибок. Для цього виконаємо такий код (вам у циклі потрібно вказувати лише індекси у фігурних дужках):

```
yu_cell_4 = cell(1, size(yu_cell, 2));
y_pred_cell_4 = cell(1, size(y_pred_cell, 2));
for i = 1: size(yu_cell, 2)
yu_cell_4{i} = yu_cell{2, i}(2);
y_pred_cell_4{i} = y_pred_cell{i}(2);
end
plotresponse(yu_cell_4, y_pred_cell_4);
```

де `yu_cell_4` – комірковий масив з дійсними значеннями виходу системи за напруги 4 *кВ*, `y_pred_cell_4` – комірковий масив із відповідними передбаченими значеннями. Кожна комірка цих масивів містить лише одне число. Графіки наведено на Рис. 8.12. Бачимо, що на початку процесу похибки доволі великі, та надалі мережа «виправляється» й починає передбачати точніше. Отже, в цьому разі мережа, навчена на згенерованому процесі, може доволі непогано передбачати процес, отриманий дослідно. Наголошуємо, що тут *мережа передбачає лише на один крок наперед*.

Функція `plotresponse` може опрацьовувати лише одну пару процесів, а дані обов'язково подавати в коміркових масивах (інакше функція працюватиме неправильно). Працювати з цією та іншими функціями через програмний код можна певною мірою гнучкіше, ніж через застосунок, хоч це й вимагає глибших знань.

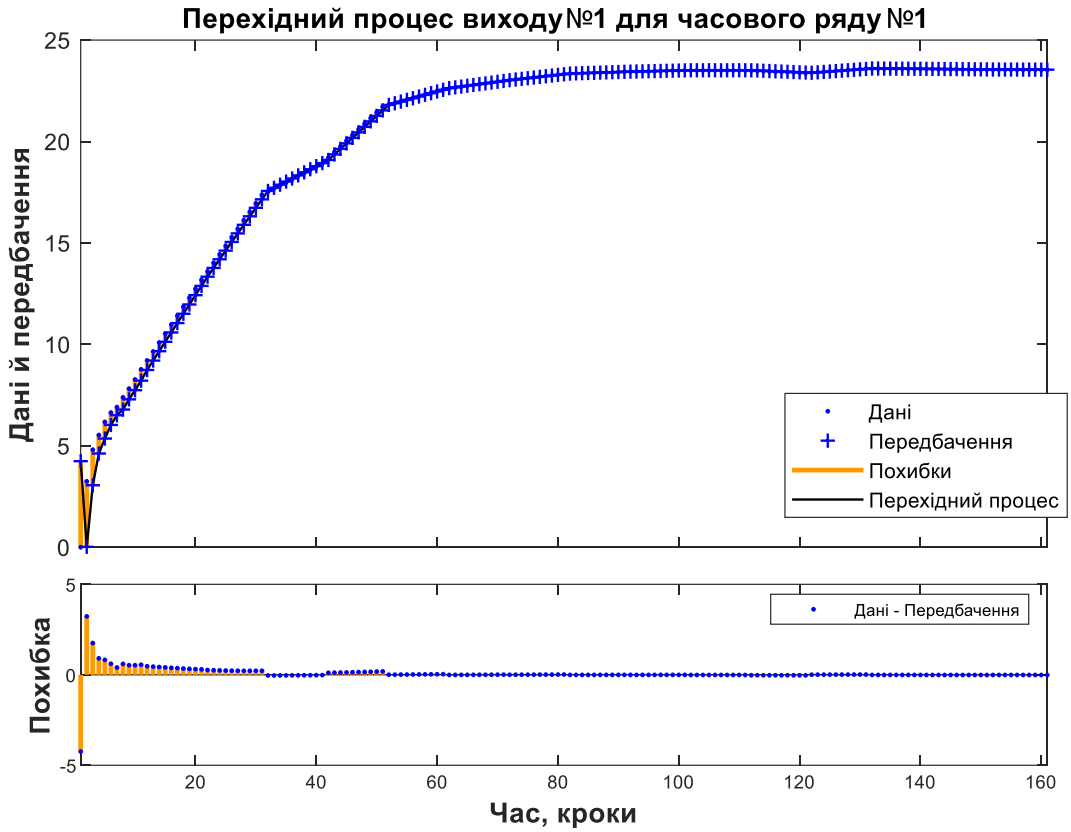


Рис. 8.12. Передбачення мережі на дослідних даних за напруги 4 кВ: дослідні дані та передбачення мережі (вгорі); похибки передбачення мережі (внизу)

Повернімося до застосунку. Якщо у вікні для розгортання мережі (Рис. 8.9) натиснути *Далі*, то побачим останнє вікно застосунку – збереження (Рис. 8.13). Воно дає змогу зберегти всю послідовність дій (або її частину), виконаних у

застосунку за поточну сесію, у скрипт *MATLAB*. Крім того, тут можна зберегти навчену нейромережу як об'єкт, а також зберегти у вигляді матриць і структур додаткові дані про навчання та роботу мережі. Докладно на цьому не зупинятимемося.

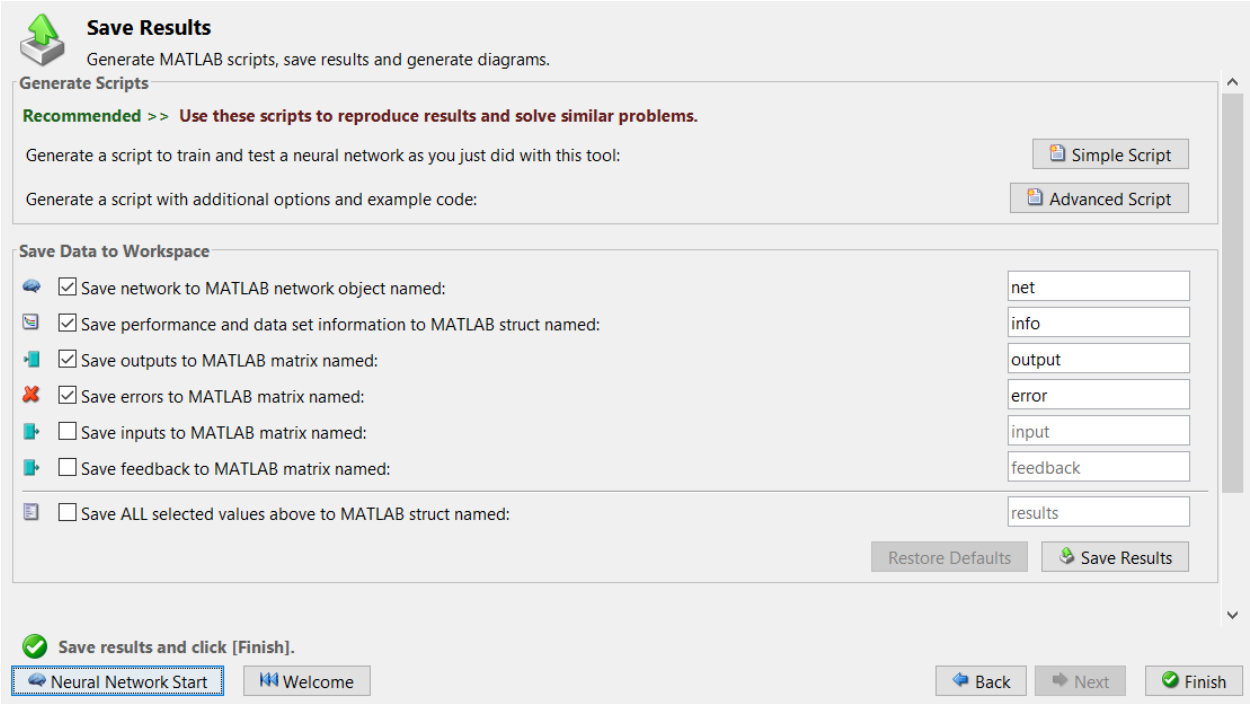


Рис. 8.13. Вікно збереження послідовності створення нейромережі та вислідів роботи з нею

Порядок виконання роботи

1. Завантажити навчальні й випробні дані з файлу `Прізвище_група.mat`, збереженого в попередній роботі.
2. Відкрити застосунок *NTSTool*. Вибрати архітектуру *NARX*.
3. Завантажити в застосунок навчальні дані (змінні `u_train` та `y_train`). Вибрати подання *Рядок матриці* (*Matrix row*). Розділити дані на навчальну, перевірну й випробну підвибірки, по 70 %, 15 % і 15 %, відповідно.
4. Задати параметри мережі. Кількість нейронів обрати довільно, а порядок моделі такий, що відповідає порядку диференційного рівняння.
5. Навчити мережу будь-яким із запропонованих способів.
6. Вивести графіки навчання (*Performance*) та передбачень (*Time-Series Response*).
7. Якщо найбільше значення відносної похибки навченої мережі перевищує 5 %, повторити кроки 5–7. Також можна спробувати змінити порядок

моделі (параметр *d*). Відносну похибку можете оцінити за графіками передбачень.

8. Зберегти навчену мережу як функцію *MATLAB* із назвою *`Прізвище_група.m`*.
9. Користуючись отриманою функцією, розрахувати передбачення на випробних даних (змінні *`u_test`* та *`y_test`*).
10. Вивести графіки передбаченого перехідного процесу та похибок за допомогою функції *`plotresponse`*.

Вміст звіту

- Остаточна архітектура мережі.
- Алгоритм навчання, кількість епох, кінцеві значення функції втрати на кожній із підвибірок (у разі Баєсової регуляризації – крім перевірної). Для отримання значень функції втрати у вікні графіка *Навчання* (*Performance*) натиснути *Інструменти – Трасування графіка* (*Tools – Data Tips*).
- Вікно з відомостями про навчання мережі (дивіться Рис. 8.6).
- Графік навчання.
- Графіки передбачень на навчальних і випробних даних (дивіться Рис. 8.7 і Рис. 8.12).

Контрольні запитання та завдання

1. За яким виразом розраховують вихід кожного нейрона прихованого шару мережі *NARX*?
2. Які надпараметри мережі *NARX* можна налаштувати в застосунку? Поясніть суть цих параметрів.
3. Яку функцію втрати використано для навчання мережі в цій роботі? Який математичний вираз для цієї функції?
4. Які ще нейромережеві архітектури для наближення часових рядів ви знаєте? Чим вони відрізняються від архітектури *NARX*?
5. Опишіть устрій коміркового масиву, в який потрібно записувати вхідні дані для функції, що втілює навчену мережу.
6. Як побудувати графіки передбачень мережі та похибок цих передбачень?

ПРАКТИКУМ 9

ВИКОРИСТАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ СИНТЕЗУ СИСТЕМИ КЕРУВАННЯ

Мета роботи: навчитися застосовувати генетичні алгоритми для синтезу системи керування засобами *MATLAB*.

Стислі теоретичні відомості

Генетичні алгоритми (*GA – Genetic Algorithms*) відносять до еволюційних, їх використовують, зокрема, для розв'язання задач оптимізації [9].

Математичний апарат генетичних алгоритмів передбачає використання такої термінології [9, 10].

Популяція – множина об'єктів – індивідів (особин).

Індивіди (особини), що входять у популяцію, – хромосоми з закодованими в них властивостями об'єктів, представлених значеннями параметрів – точки у просторі пошуку всіх можливих рішень.

Хромосоми – впорядковані послідовності генів.

Ген – закодована певною частиною хромосоми властивість особини.

Алель – значення конкретного гена.

Генотип або структура – набір хромосом особини.

Фенотип – набір зовнішніх спостережуваних ознак, які відповідають даному генотипу, тобто декодована структура або безліч параметрів.

На кожній ітерації генетичного алгоритму пристосованість кожної особини даної популяції оцінюється за допомогою функції пристосованості (цільової функції), і на цій основі створюється наступна популяція особин, що складають безліч потенційних рішень задачі. Чергова популяція в генетичному алгоритмі називається поколінням, а новостворювана популяція особин – нове покоління.

Класична схема генетичного алгоритму складається з таких етапів [10]:

1. формування початкової популяції хромосом розміром N;
2. оцінка пристосованості хромосом у популяції;
3. перевірка умови зупинення алгоритму;
4. селекція хромосом;
5. застосування генетичних операторів (схрещування, мутації);

6. формування нової популяції;

7. перехід до п. 2, якщо популяція не зійшлася, інакше – зупинка.

Розглянемо детально кожний крок цього алгоритму.

Формування початкової популяції. Стандартний генетичний алгоритм починає свою роботу з формування початкової популяції – кінцевого набору припустимих рішень задачі. Ці рішення можуть бути вибрані випадковим чином або отримані за допомогою простих наближених алгоритмів. Якщо відсутні припущення про місце розташування глобального оптимуму, то індивіди з початкової популяції бажано розподілити рівномірно по всьому простору пошуку рішення.

Оцінка пристосованості хромосом у популяції (оцінка особин популяції). Для задачі оптимізації з використанням генетичних алгоритмів, потрібно задати міру якості для кожного індивіда у просторі пошуку. Для цієї мети використовують функцію пристосованості. В задачах максимізації цільова функція часто сама виступає як функція пристосованості; для задач мінімізації цільову функцію слід інвертувати. Якщо вирішувана задача має обмеження, виконання яких неможливо контролювати алгоритмічно, то функція пристосованості, як правило, включає також штрафи за невиконання обмежень (вони зменшують її значення).

Перевірка умови зупинення алгоритму. Умова зупинення генетичного алгоритму залежить від його конкретного застосування. В задачах оптимізації зупинка алгоритму відбувається, якщо відомо максимальне (або мінімальне) значення функції пристосованості, після досягнення очікуваного оптимального значення, можливо, з заданою точністю.

Селекція (відбір) хромосом. Існує декілька підходів до вибору батьківської пари (чергового покоління), для кожного з яких застосовують відповідні оператори селекції [10].

Застосування генетичних операторів (схрещування, мутації). Оператори рекомбінації (схрещування) застосовують відразу ж після оператора відбору батьків для здобуття нових особин-нащадків. Сенс рекомбінації полягає в тому, що створені нащадки повинні успадковувати генну інформацію від обох батьків. Розрізняють дискретну рекомбінацію і кросинговер.

Після того, як закінчиться стадія кросинговера, нащадки можуть піддаватися випадковим модифікаціям, названим мутаціями.

Формування нової популяції (нового покоління). Після схрещування і мутації особин необхідно вирішити, які з нових особин увійдуть до наступного покоління, а які - ні, і що робити з їх предками. Є два найбільш поширених способи.

1. Нові особини (нащадки) займають місця своїх батьків. Після чого настає наступний етап, в якому нащадки оцінюються, відбираються, дають потомство і поступаються місцем своїм «дітям».

2. Наступна популяція включає як батьків, так і їх нащадків.

Зупинка алгоритму. Робота генетичного алгоритму є ітераційним процесом, який продовжується до тих пір, поки не пройде задане число поколінь або не виконається який-небудь інший критерій зупинки. В задачах оптимізації традиційними критеріями зупинки алгоритму є, наприклад, тривала відсутність прогресу (поліпшення значення середньої (або кращої) пристосованості популяції), мала різниця між кращим і гіршим значенням пристосованості для поточної популяції тощо.

Розглянемо задачу визначення оптимальних параметрів регулятора з використанням генетичних алгоритмів.

Схему одноконтурної системи керування наведено на рис. 9.1.

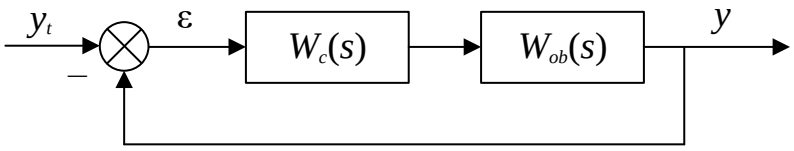


Рис. 9.1. Схема одноконтурної системи керування:
$W_{ob}(s)$, $W_c(s)$ – передавальні функції об’єкта і регулятора відповідно, y_t – завдання, y – вихід об’єкта, ε – похибка керування

Для прикладу використаємо такі передавальні функції об’єкта керування і ПІД-регулятора відповідно.

$$W_{ob}(s) = \frac{1}{7s^2 + 30s + 1} e^{-15s};$$

$$W_c(s) = K_p + \frac{T_i}{s} + T_d s,$$

де K_p – коефіцієнт підсилення; T_i – стала часу інтегрування; T_d – стала часу диференціювання.

Відповідно передавальна функція ПІ-регулятора має вигляд

$$W_c(s) = K_p + \frac{T_i}{s}.$$

За критерій оптимізації, J, обрано суму абсолютних значень відхилень y від y_t:

$$J = \sum_0^t t|y_t - y|dt,$$

де t – час спостереження, dt – крок за часом.

Наведемо терміни, які використовують генетичні алгоритми, та відповідні їм аналоги для такої задачі.

Популяція – вибірка з діапазону зміни коефіцієнтів регулятора. Якщо пошук коефіцієнтів ПІД-регулятора здійснюють в таких діапазонах: K_p є $[K_{p,\min}, K_{p,\max}]$, T_i є $[T_{i,\min}, T_{i,\max}]$, T_d є $[T_{d,\min}, T_{d,\max}]$, популяцією буде множина S є $\{K_p, T_i, T_d\}$. Звернемо увагу, що популяція не складає множину всіх можливих поєднань $\{K_p, T_i, T_d\}$. Її формують як вибірку S коефіцієнтів регулятора з діапазону пошуку.

Хромосоми – закодована послідовність коефіцієнтів регулятора (K_p, T_i, T_d).

Ген – значення коефіцієнта регулятора, подане не в чисельному, а закодованому значенні, наприклад, у вигляді двійкового коду, тобто кожне число з діапазону пошуку подають набором 0 і 1. Способи кодування можуть бути різними, їх у цій роботі не розглядають.

Алель – значення конкретного гена: 0 або 1.

Генотип або структура – набір коефіцієнтів регулятора в закодованому вигляді.

Фенотип – набір коефіцієнтів регулятора в дійсних значеннях.

Наприклад, нехай коефіцієнти регулятора є цілими числами і змінюються в діапазоні [0, 3]. Наведемо в таблиці 9.1 приклад кодування у вигляді двійкових чисел.

У даному випадку популяція – вибірка із закодованих значень, наведених у таблиці 9.1. Хромосоми (набір K_p, T_i, T_d) записані в стовпці «Кодовані значення», індивіди (особини) – хромосоми з популяції, ген – частина хромосоми, алель – значення, яке приймає конкретний ген у хромосомі. Генотипом є хромосома або набір хромосом, фенотипом – відповідні дійсні значення генотипу.

Таблиця 9.1. **Приклад кодування коефіцієнтів регулятора**

№	Кодоване значення	Дійсне значення	№	Кодоване значення	Дійсне значення	№	Кодоване значення	Дійсне значення
1	000000	[0; 0; 0]	23	011001	[1; 2; 1]	45	011111	[1; 3; 3]
2	010000	[1; 0; 0]	24	100101	[2; 1; 1]	46	101100	[2; 3; 0]
3	100000	[2; 0; 0]	25	101001	[2; 2; 1]	47	101101	[2; 3; 1]
4	000100	[0; 1; 0]	26	100110	[2; 1; 2]	48	101110	[2; 3; 2]
5	001000	[0; 2; 0]	27	101010	[2; 2; 2]	49	110000	[3; 0; 0]
6	000001	[0; 0; 1]	28	000011	[0; 0; 3]	50	110001	[3; 0; 1]
7	000010	[0; 0; 2]	29	000111	[0; 1; 3]	51	110010	[3; 0; 2]
8	010100	[1; 1; 0]	30	001011	[0; 2; 3]	52	110011	[3; 0; 3]
9	100100	[2; 1; 0]	31	0011111	[0; 3; 3]	53	110100	[3; 1; 0]
10	011000	[1; 2; 0]	32	010011	[1; 0; 3]	54	111000	[3; 2; 0]
11	101000	[2; 2; 0]	33	011011	[1; 2; 3]	55	111100	[3; 3; 0]
12	000101	[0; 1; 1]	34	011111	[1; 3; 3]	56	110101	[3; 1; 1]
13	001001	[0; 2; 1]	35	100011	[2; 0; 3]	57	110110	[3; 1; 2]
14	000110	[0; 1; 2]	36	100111	[2; 1; 3]	58	110111	[3; 1; 3]
15	001010	[0; 2; 2]	37	101011	[2; 2; 3]	59	111001	[3; 2; 1]
16	010001	[1; 0; 1]	38	1011111	[2; 3; 3]	60	111010	[3; 2; 2]
17	100001	[2; 0; 1]	39	001100	[0; 3; 0]	61	111011	[3; 2; 3]
18	010010	[1; 0; 2]	40	001101	[0; 3; 1]	62	111101	[3; 3; 1]
19	100010	[2; 0; 2]	41	001110	[0; 3; 2]	63	111110	[3; 3; 2]
20	010101	[1; 1; 1]	42	011100	[1; 3; 0]	64	111111	[3; 3; 3]
21	011001	[1; 2; 1]	43	011101	[1; 3; 1]			
22	010110	[1; 1; 2]	44	011110	[1; 3; 2]			

Розглянемо спосіб розв’язання задачі пошуку оптимальних параметрів налаштування ПІ- (ПІД-) регулятора з використанням генетичних алгоритмів у *MATLAB*.

9.1. Створимо функцію пристосованості (назвемо її “*gen_syst*”) для розрахунку поточного значення критерію оптимізації *J*. В неї входять передавальні функції ОК і регулятора.

```
function [J] = gen_syst(x)
s = tf('s');
W_ob = (13 / (7*s^2 + 30*s + 1))*exp(-15*s); % передавальна функція об'єкта керування
Kp = x(1); % коефіцієнт пропорційності
```

Ti = *x*(*2*); % стала часу інтегрування

%*Td* = *x*(3); % стала часу диференціювання

W_c = *Kp* + *Ti/s*; % передавальна функція ПІ- (ПІД-регулятора); *W_c* = *Kp* + *Ti/s* + *Td* * *s*

step(*feedback*(*W_ob*W_c*,*1*)); % подача одиничного ступінчатого сигналу

dt = 1;

t = 0:*dt*:150;

eps = *1* - *step*(*feedback*(*W_ob*W_c*,*1*),*t*); % різниця між вхідним і вихідним сигналами

J = *sum*(*t*'.*abs*(*eps*)*dt*)

end

Зберігаємо скрипт у файл з назвою "*gen_syst.m*".

9.2. Визначення оптимальних значень параметрів налаштування регулятора здійснимо з використанням пакету *Optimize MATLAB*.

Для цього створимо «живий» скрипт (*Live Script*) командою *Home → New Live Script*. На вкладинці *Live Editor* виконаємо виклик застосунку *Optimize* відповідною командою *Task → Optimize*. У вікні, яке з'являється (рис. 9.2), оберемо підхід *Solver-based*.

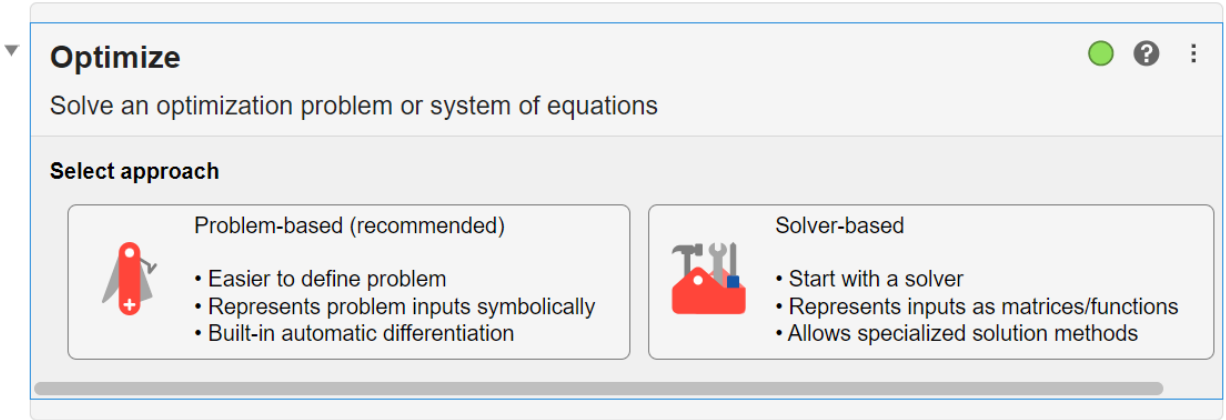


Рис. 9.2. Вікно для вибору методу розв'язання задачі

У результаті з'являється вікно, наведене на рис. 9.3.

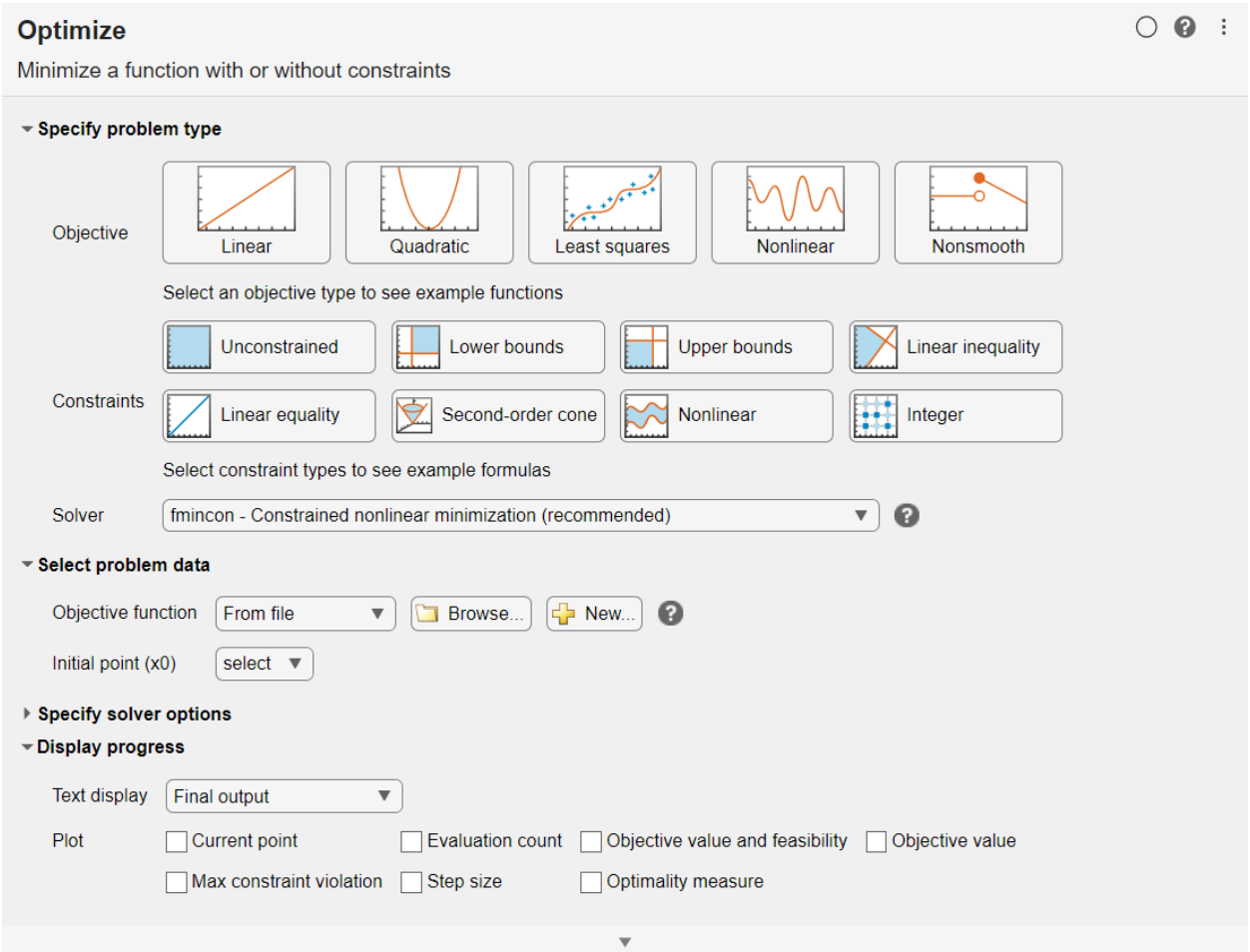


Рис. 9.3. Початкове вікно для розв'язування задач оптимізації

9.3. Подання задачі оптимізації у вікні *Optimize*. Для наведеного прикладу використано такі параметри налаштування.

У зоні *Specify problem type* існує три опції: тип функції (*Objective*); обмеження (*Constraints*); вибір методу оптимізації – *Solver*.

- Тип функції – нелінійний (*Nonlinear*).

- Обмеження (*Constraints*) для параметрів налаштування регулятора – нижня (*Lower bounds*) і верхня (*Upper bounds*) межі.

- *Solver* – генетичний алгоритм (*ga – Genetic Algorithm*).

У зоні *Select problem data* існує дві опції: функція мети (*Objective function*); вибір кількості змінних (*Number of variables*).

- Існує декілька варіантів вибору функції мети: записати в скрипті (*New…*) або викликати з файлу (*From file*).

При виборі з файлу натискаємо "*Browse to File…*" і вказуємо шлях до файлу "gen_syst.m".

- Кількість змінних *n* для оптимізації. Для ПІ-регулятора *n* = 2, для ПІД-регулятора, відповідно, *n* = 3. Кількість змінних потрібно задати через командний рядок (*Command Window*).

У *зоні Specify solver options* задаємо, якщо потрібно, додаткові параметри для пошуку. Наприклад, задамо розмір популяції. *Add → Population settings → Population size*. За замовчуванням вказано значення "*default*". Якщо $n \leq 5$, це значення за замовчуванням дорівнює 50, якщо Якщо $n > 5$, популяція дорівнює 200.

У зоні *Display progress* існує дві опції: відображення тексту (*Text display*); графіки (*Plot*).

- Зазначаємо дані виведення: без виведення (*No display*); виведення інформації про кожну ітерацію (*Each iteration*); виведення інформації про кожну ітерацію та додаткові відомості про можливі помилки та зміни в ключових параметрах алгоритму (*Diagnose*); виведення вихідних значень (*Final output*).

- Зазначаємо, які дані виводити графічно.

Distance – інтервал між значеннями індивідів у поколінні;

Scores – рейтинг кожного індивіда у поколінні;

Best individual – найкращий представник покоління при найкращому значенні критерія оптимізації в кожному з поколінь;

Genealogy – генеалогічне дерево індивідів (графік виводить після завершення всіх обчислень);

Stopping criteria – інформація про стан всіх параметрів, які впливають на критерії зупинки;

Expectation value – ряд ймовірностей та відповідні їм індивіди поколінь;

Selection – гістограма батьків;

Max constraint violation – максимальне значення обмежень, які не задовольняють значення функції;

Range – найменше, середнє і найбільше значення функції оптимізації в кожному поколінні;

Score diversity – гістограма рейтингу в кожному поколінні;

Best fitness – найкраще значення функції в кожному поколінні.

Вікно з налаштуваннями у п. 9.3 наведено на рис. 9.4.

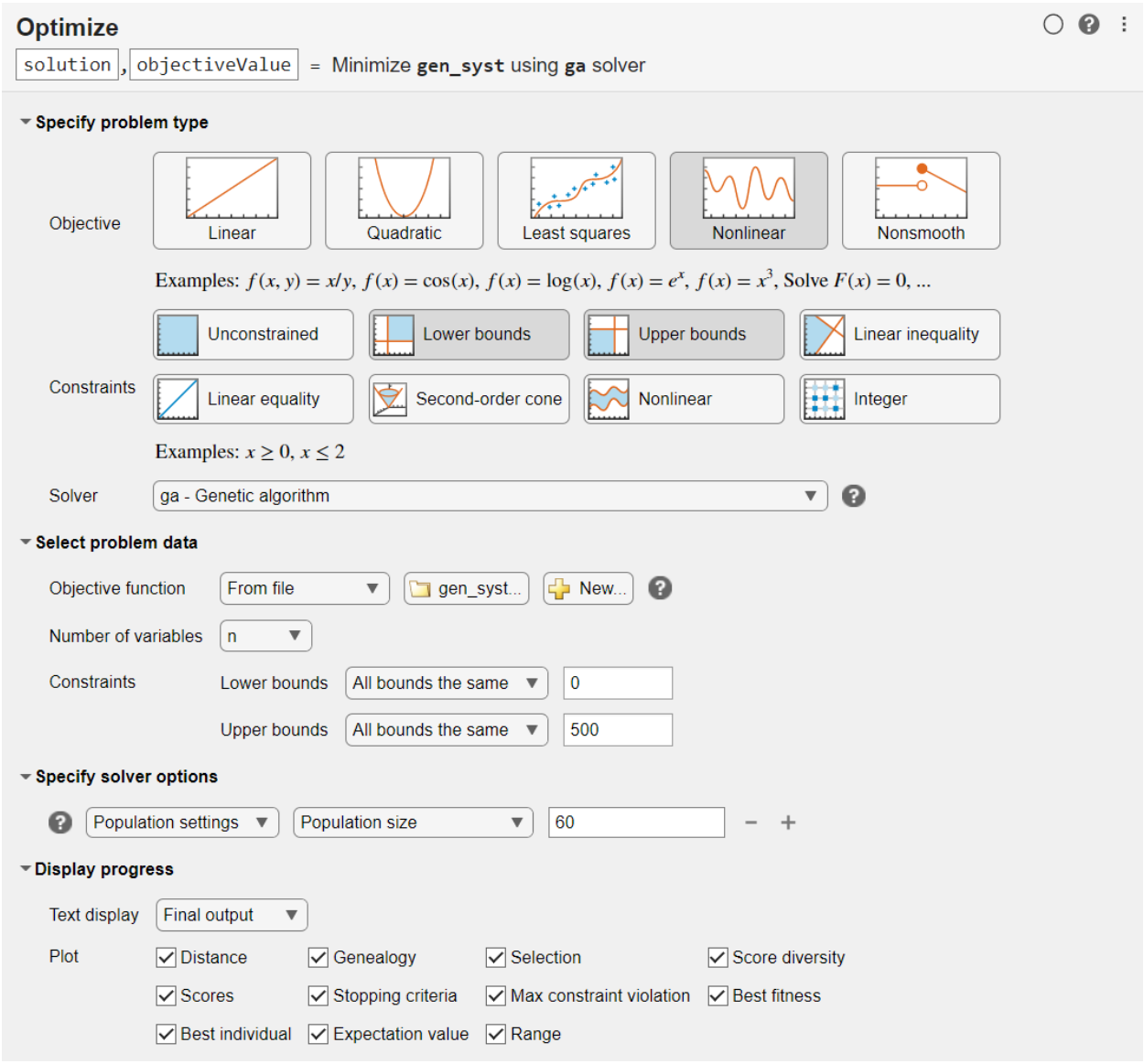


Рис. 9.4. Вікно з налаштуваннями параметрів пошуку коефіцієнтів регулятора

Для запуску алгоритму натискаємо кнопку “*Run*”, розташовану на панелі інструментів *Live Editor MATLAB*.

Після виконання програми в робочому просторі (*Workspace*) створюється вектор оптимальних коефіцієнтів налаштування регулятора “solution”.

solution =

0.0845 0.0026

Для наведеного прикладу K_p = 0.0845, T_i = 0.0026.

На рис. 9.5 можна переглянути параметри пошуку, задані в блоці “*Display progress*”.

Повторно використаємо скрипт, підставивши в нього оптимальні значення регулятора.

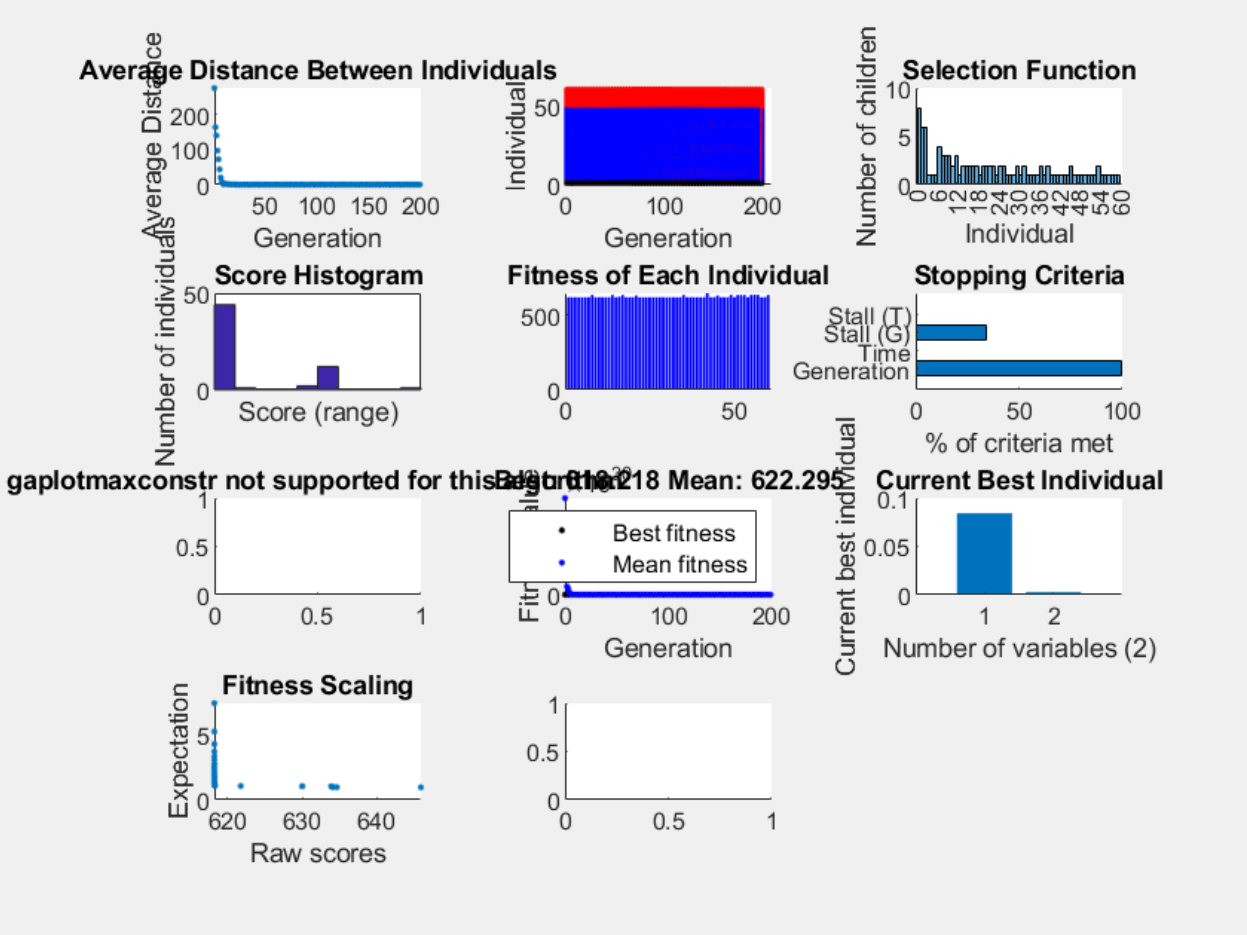


Рис. 9.5. Вікно з результатами налаштування системи керування

На рис. 9.6, наведено перехідну характеристику системи з оптимальними налаштуваннями ПІ-регулятора.

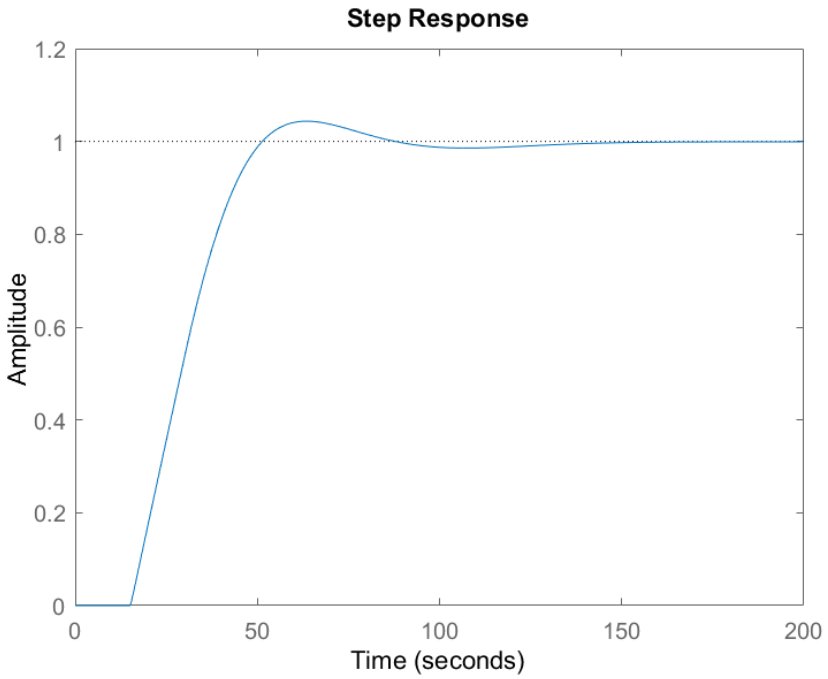


Рис. 9.6. Перехідна характеристика системи керування

Порядок виконання роботи

1. Створити функцію за методикою, описаною у посібнику.

Студентам треба назвати функцію не “gen_syst.m”, а за шаблоном “Прізвище_Назва_групи_gen_syst.m” (назву файла латиницею).

2. Визначити оптимальні налаштування ПІ- і ПІД-регулятора для власного технологічного об’єкта керування.

3. Дослідити зміну діапазону параметрів налаштування регулятора.

4. Дослідити вплив додаткових параметрів на результати пошуку (розмір популяції (*Population size*), кількість індивідуумів (*Number of elite members*) тощо).

5. Побудувати перехідні характеристики замкненої системи з оптимальними налаштуваннями ПІ- і ПІД-регуляторів. Порівняти їх.

Вміст звіту

Передавальні функції об’єкта керування, ПІ-, ПІД-регуляторів, функція мети, скрипт функції “gen_syst.m”, зображення вікна налаштування параметрів пошуку, графіки з результатами пошуку коефіцієнтів регулятора, перехідні характеристики системи керування з ПІ- та ПІД-регуляторами.

Контрольні запитання

1. Що таке функція пристосованості, як її формують?

2. Що таке оцінка пристосованості особин?

3. Як впливає розмір популяції на досягнення оптимального значення функції мети?

4. Як впливає діапазон зміни обмежень на досягнення оптимального значення функції мети?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Експертні методи в автоматизованих системах керування : Формування та напрями використання експертних знань : [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: Л. Д. Ярощук. – 2-ге вид., допов. – Київ : КПІ ім. Ігоря Сікорського, 2022. – 43 с.

2. Інтелектуальні системи управління: Експертні системи – основи проектування та застосування в системах автоматизації [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: Л. Д. Ярощук. –Київ : КПІ ім. Ігоря Сікорського, 2019. – 136 с.

3. Апостолюк В. О. Інтелектуальні системи керування: конспект лекцій [Текст] / В.О. Апостолюк, О.С. Апостолюк. – К.: НТУУ «КПІ», 2008. – 88 с.

4. Системи фуцці-керування/В.І. Архангельський, І.М. Богаєнко, Г.Г. Грабовський, М.О. Рюмшин.- К.: Техніка, 1997. – КПІ ім. Ігоря Сікорського. 208 с.

5. Кирик В. В. Математичний апарат штучного інтелекту в електроенергетичних системах: підручник / В. В. Кирик. – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка» 2019. – 224 с.

6. Time Series Data Augmentation for Deep Learning: A Survey / Q. Wen та ін. *Thirtieth International Joint Conference on Artificial Intelligence {IJCAI-21}*, м. Montreal, Canada, 19–27 серп. 2021 р. California, 2021. URL: https://doi.org/10.24963/ijcai.2021/631 (дата звернення: 02.03.2025).

7. High-Performance Electrostatic Coalescer – A Novel Technology for Improving the Economics of Oil-Water Separation / A. S. M. Ismail та ін. *Abu Dhabi International Petroleum Exhibition & Conference*, м. Abu Dhabi, UAE. 2020. URL: https://doi.org/10.2118/203475-ms (дата звернення: 02.03.2025).

8. Putiatin R., Tsapar V. Modelling of high performance electrocoalescer based on experimental data. *Proceedings of the NTUU "Igor Sikorsky KPI". Series: Chemical engineering, ecology and resource saving*. 2023. № 2. С. 39–52. URL: https://doi.org/10.20535/2617-9741.2.2023.283523 (дата звернення: 02.03.2025).

9. Simon D. Evolutionary Optimization Algorithms. Biologically-Inspired and Population-Based Approaches to Computer Intelligence, 1st ed. / D. Simon. New Jersey: John Wiley & Sons, Inc., Hoboken, 2013. – 742 p.

10. Черняк О. І. Інтелектуальний аналіз даних: Підручник / О. І. Черняк, П. В. Захарченко. – Київ : Знання, 2010. – 837 с.