Ця система векторно-матричних диференціальних рівнянь містить $n+n(n+s)$ рівнянь та $n+n^2+ns$ невідомих функцій $x_m(t)$; $A_n(t)$ і $B_m(t)$.

Алгоритм настроювання є рівномірно асимптотично стійким, якщо виконані умови (1), (3) та (11).

Список літератури

1. *Вектор-функции* Ляпунова и их построение: Сб. науч. тр. СО АН СССР. – Новосибирск: Наука, 1980. – 288 с.

Стаття надійшла до редакції 19.03.01.

УДК 681.513.061:658.012.122
В 185.5+З 973.2-018

Ю.М. Мінаєв, О.Ю. Філімонова, Бенамур Лієс, Р.А. Ярош

АЛГОРИТМИ ТА ПРОГРАМИ РОЗВ'ЯЗУВАННЯ СИСТЕМ НЕЧІТКИХ АЛГЕБРИЧНИХ РІВНЯНЬ ТА ЇХ РЕАЛІЗАЦІЯ В СИСТЕМІ МОДЕЛЮВАННЯ MATLAB/SIMULINK

Розроблено алгоритм розв'язування нечіткої системи алгебричних рівнянь в нейромережному базисі на платформі MatLab/Simulink. Система нечітких рівнянь перетворюється в систему з (n×к) *чітких рівнянь, де к – кількість компонент для уявлення нечітких коефіцієнтів* ($\tilde{a}_{ij} \in \tilde{\mathbf{A}}$) *та правих частин* ($\tilde{b}_i \in \tilde{\mathbf{b}}$), i, j = n×n *(для чіткої системи). На* α*-рівні система має розмірність* n×n *і в загальному випадку стає несумісною (перевизначеною). Запропонований алгоритм дозволяє отримати наближений розв'язок систем рівнянь в нейромережному базисі незалежно від сумісності. Розв'язок загальної (несумісної системи) відшукується в області* $\{x^*, x^2\}$. *де* x^* *– дефадзифікований розв'язок системи, отриманий на* α*-рівнях;* x^2 *– розв'язок загальної (несумісної) системи. Такий підхід дозволяє суттєво розширити клас розв'язуваних задач ідентифікації та прогнозування в умовах невизначеності.*

До розв'язування систем лінійних рівнянь (СЛР) зводяться різні задачі, що виникають при моделюванні поведінки об'єктів, проектуванні систем керування, економічному аналізі, прогнозуванні та ін. Для розв'язування СЛР можуть застосовуватися різні обчислювальні методи та прийоми залежно від характеристик конкретної системи. Ефективність використовуваного апарата значно впливає на процес опрацювання даних і продуктивність обчислювальних систем [1].

У ситуаціях, коли система рівнянь як результат експерименту є несумісною (недовизначеною, перевизначеною) або параметри моделі (коефіцієнти $\tilde{a}_{ij} \in \tilde{\mathbf{A}}$, праві частини $\tilde{b}_i \in \tilde{\mathbf{b}}$) визначені наближено (інтервально) або у вигляді нечітких чи змінних чисел, доцільно рекомендувати природну генерацію деякого методу, що дозволяє одержати достовірний результат, адекватний поставленій задачі як за точністю, так і за формою.

На сьогоднішній день відомі спроби модифікації традиційних методів лінійної алгебри стосовно сучасних прикладних обчислювальних задач [1].Так, для сумісних СЛР потрібен точний розв'язок, тоді як для несумісних систем, що не мають розв'язку, результатом може бути деякий вектор (наближений розв'язок), що дає мінімальну похибку і водночас має можливість

отримати точний розв'язок, якщо система сумісна. Виконання даної умови (одночасний розв'язок сумісних та несумісих, до того ж ще й нечітких рівнянь) можливе, наприклад, при використанні нейронних структур, які мають, зокрема, властивості навченості і самонавченості, що дозволяє розв'язувати також задачі великої розмірності. Реалізація алгоритмів для розв'язування чітких сумісних СЛР, що мають точний розв'язок, в нейромережному базисі, докладно розглянута в сучасній науковій літературі [2;3], реалізації на нейронних мережах методів пошуку наближених розв'язків несумісих СЛР присвячені роботи [4;5], але методи розв'язування несумісних нечітких СЛР на даний час практично не розв'язані.

Можливість розв'язування нечітких систем лінійних алгебричних рівнянь (НСЛАР) здійснено зведенням їх до розв'язування сумісних (несумісних) чітких систем в нейромережному логічному базисі з використанням засобів систем моделювання MatLab/Simulink. Розв'язування НСЛАР в нейронних мережах можливо із застосуванням нечітких нейронів [6].

Нечітка система лінійних алгебричних рівнянь береться у вигляді

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}},$$

де $\widetilde{a}_{ij} \in \widetilde{\mathbf{A}}$; $\widetilde{b}_i \in \widetilde{b}$; $\widetilde{x}_i \in \widetilde{x}$, $\widetilde{\alpha}_{ij} \overset{\text{def}}{=} \bigcup_{\iota,j}(\alpha_{ij}^k/\mu_{\alpha_{ij}}^k)$; $\widetilde{b}_i \overset{\text{def}}{=} \bigcup_{\iota}(b_\iota^k/\mu_{b_\iota}^k)$; $\widetilde{x}_i \overset{\text{def}}{=} \bigcup_{\iota}(x_\iota^k/\mu_{x_\iota}^k)$,

де i=1,n; j=1,m; k=1,K; $\overset{\text{def}}{=}$ – дорівнює за визначенням.

Розглядаємо НСЛАР на α-рівнях, тобто для k α-рівнів ($\alpha = \bigcup(\alpha^k)$, k=1,K) сформуємо систему з k правил :

$$R_1 : \text{if } \mu_{a_{ij}}^1 \le \mu^{\alpha_1} \text{ and } \mu_{b_i}^1 \le \mu^{\alpha_1} \text{ then } \mathrm{A}^{\alpha_1} x = b^{\alpha_1};$$

$$R_k : \text{if } \mu_{a_{ij}}^k \le \mu^{\alpha_k} \text{ and } \mu_{b_i}^k \le \mu^{\alpha_k} \text{ then } \mathrm{A}^{\alpha k} x = b^{\alpha k},$$

де $A^{\alpha_k}, b^{\alpha_k}$, k =1,K – матриці та вектор правих частин сформованих систем рівнянь.

Отримуємо, з одного боку, k систем чітких лінійних алгебричних рівнянь, з іншого – одну перевизначену систему, яка складається з n × k рівнянь з n невідомими, причому ці розв'язки є абсолютно рівнозначними (залежно від умов задачі). Крім того, слід враховувати, що отримані на α-рівнях СЛАР можуть мати лінійно залежні рівняння, тобто потрібно отримувати розв'язок недовизначеної СЛАР (m < n).

Як відомо, нейронна мережа реалізує рівняння $y_i = \sum_j a_{ij} x_j$, де x_j (j – номер нейрона) – сигнали на входах всіх нейронів. Проходження вектора сигналів **x** через мережу зв'язків зводиться до множення матриці **A** =$\{ a_{ij} \}$ на вектор сигналів **x**. Практично всюди, де зустрічається операція множення матриці на вектор, можуть бути застосовані нейронні мережі. Зокрема, обчислення градієнта квадратичної форми H=1/2(x,Qx) може здійснюватися повно зв'язаною мережею із симетричною матрицею зв'язків: grad H=Qx ($a_{ij} = q_{ij} = q_{ji}$). Якщо існує можливість обчислювати градієнт квадратичної форми, то методом найшвидшого спуску можна шукати точку мінімуму багаточлена другого порядку.

Припустимо, що задано такий багаточлен: P(x)= 1/2 (x,Qx)+(b,x). Його градієнт дорівнює grad P=Qx+b. Цей вектор може бути отриманий при проходженні вектора **x** через мережу з вагами зв'язків ($a_{ij} = q_{ij}$) за умови, що на вхідний суматор кожного нейрона по додатковому зв'язку ваги **b** подається стандартний одиничний сигнал. Задамо тепер функціонування мережі формулою

$$x' = x - h \text{ grad } P = x - h(Qx + b).$$

Кожний j-й нейрон має вхідні ваги $a_{ij} = -hq_{ij}$ для зв'язків з іншими нейронами ij, вагу $-b_j$ для постійного одиничного вхідного сигналу і вагу $a_{ij} = 1 - hq_{ij}$ для зв'язку нейрона із самим собою (передачі на нейрон того ж самого сигналу з попереднього кроку). Вибір кроку h>0 може викликати утруднення (він залежить від коефіцієнтів мінімізованого багаточлена). Однак, є простий розв'язок: в кожний момент дискретного часу T вибирається своє значення і крок повинен збігатися згодом до нуля, а сума кроків h_T – до нескінченності, наприклад, $h_T=1/T$ або $h_T=1/(T)^{1/2}$. Отже, проста симетрична повно зв'язана мережа без нелінійних елементів може методом найшвидшого спуску відшукувати точку мінімуму квадратичного багаточлена. Розв'язування СЛР **Ax=b** зводиться до мінімізації багаточлена [2–4]

$$P = \frac{1}{2}\big((Ax-b),(Ax-b)\big) = \frac{1}{2}\big(x, A^T Ax\big) - \big(A^T b, x\big) - \frac{1}{2}(b,b).$$

Тому розв'язання системи може виконуватися нейронною мережею. Найпростіша мережа, що обчислює градієнт цього багаточлена, не є повно зв'язаною, а складається з двох шарів: перший з матрицею зв'язків **A**, другий з транспонованою матрицею $\mathbf{A}^T$. Постійний одиничний сигнал подається на зв'язки з вагами $-b_j$ на першому шарі. Мінімізацію цього багаточлена, а значить, і розв'язування СЛР може виконуватися так само, як і в загальному випадку, за формулою $x' = x - h \operatorname{grad} P$. Удосконалені варіанти алгоритму розв'язку можна знайти в роботі [3]. Певна модифікація дозволяє замість безумовного мінімуму багаточлена другого порядку P шукати точку умовного мінімуму з умовами $x_i = c_i$ для $i = i_1, \ldots, i_k$. Для цього замість формули $x' = x - h \operatorname{grad} P = x - h(Qx + b)$ слід використовувати:

$$x_i^1 = c_i \quad \text{при} \quad i = i_1, \ldots, i_k;$$

$$x_i^1 = x_i - h\frac{\partial P}{\partial x_1} = x_i - h\left(\sum_j q_{ij}x_j + b_i\right) \quad \text{при } i \neq i_1 \ldots i_k.$$

При розв'язуванні несумісних СЛР **Ax=b** основною задачею є обчислення результуючого вектора $\mathbf{x}^н = \{x_i^н\}, i = 1,n$, що мінімізує похибку E_1, яка характеризує відхилення вектора **b** від одержуваного у ($\mathbf{Ax}^н=\mathbf{y}$). Для певності будемо вважати, що матриця **A** має розмір N × M, вектор **b** – N × 1, вектор **x** – M × 1 (N, M ≥1). Метод та алгоритм розв'язування несумісних СЛР в нейромережному базисі розглянуто в роботі [5], згідно з яким одним з методів оцінки похибки E_1 є мінімізація квадрата відхилення компонент векторів **b** та **y**:

$$E1 = \Sigma (b - y)^2 \rightarrow \min .$$

На відміну від сумісних СЛР [2-4], де розв'язок вважається знайденим за умови, що ($\mathbf{b} - \delta_1 < E1 < \mathbf{b} + \delta_1$), де δ_1 – апріорно задана точність (похибка обчислень), у несумісній системі ця умова не може бути виконаною. Шуканим вектором тут вважається вектор $\mathbf{x}^н$, при якому похибка E_1 сягає мінімуму і стабілізується. Для пошуку наближеного розв'язку несумісної СЛР пропонується пошук розв'язку за допомогою процедур надбудови методом найменших квадратів, згідно з яким задача розв'язання СЛР розглядається як знаходження такого вектора **x**, який при множенні на матрицю **A** дає мінімальне (в розумінні квадрата відхилення помилки) значення щодо вектора **b**. Запропонований в роботі [5] алгоритм включає такі процедури:

– на вхід нейронної мережі подається шуканий вектор **x**, що у початковий момент часу дорівнює випадковому вектору, наприклад, одиничному ($\mathbf{x_0} = [1]$);

– на виході отримується вектор $\mathbf{y} = f(g)$, де $g = \mathbf{Ax-b}$, бажаний вихідний сигнал $\mathbf{y} = [0]$;

– критерієм якості є мінімізація функціонала F, що залежить від похибки **y**:

$$F(y) = y^T y \rightarrow \min .$$

Процедура пошуку розв'язку проводиться за допомогою ітераційного градієнтного методу, за яким $x(k+1) = x(k) - \Delta x(k)$:

$$\Delta x(k) = h\frac{\delta F}{\delta X}\Big|_{x=[k)} = 2hA^T y(k).$$

Цей метод є основою для поширення на випадок несумісної СЛР, якщо до контуру настроювання (блока, де реалізується ітераційна процедура) входить блок визначення досягнення мінімуму функціоналом якості.

Якщо СЛР є сумісною, то на деякому кроці вектор помилки **y** стане нульовим, отже, функціонал якості F також набуде значення 0 і перестане змінюватися [5]. Якщо ж СЛР є несумісною, то **y** нульового значення не набуде і його зміни будуть незначними, значення функціонала F також стабілізується. Таким чином, бажаним виходом є стабілізоване значення помилки **y**.

Відповідна нейронна мережа в графічній нотації MatLab показана на рис 1.

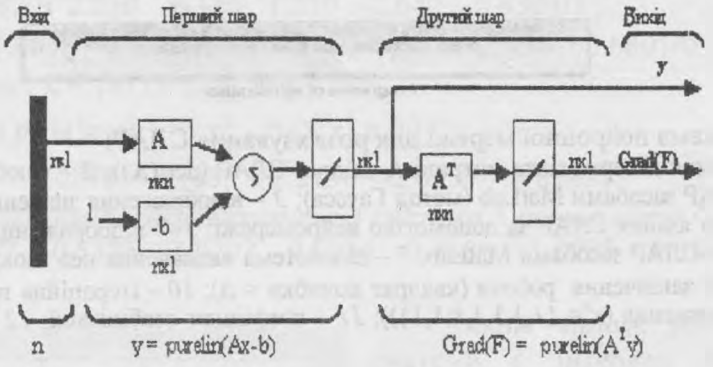

Рис.1. Схема нейронної мережі для розв'язання СЛАР

Слід зазначити, що запропонований метод є універсальним і може бути використаний також для розв'язання несумісних СЛР. У цьому разі на деякому кроці вектор **y** стане нульовим, отже функціонал якості F також набуде значення 0 і перестане змінюватися. Характерною особливістю розв'язування несумісних СЛР є те, що матриця **A** не є симетричною:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 9 & 1 & .4 & 0 & 0 & .02 & 4 \\ 2 & 3 & .7 & .9 & 0 & 0 & 1 \\ 2 & 0 & 5 & 7 & 1 & 4 & -1 \\ 9 & -2 & 0 & 0 & .1 & 5 & 7 \\ -2 & 3 & .6 & 8 & 9 & 2 & -3 \\ 0 & .01 & 0 & .2 & 3 & 3 & 4 \end{pmatrix}; \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 9 \\ 1 \\ 0 \\ 0.1 \\ -8 \\ -1 \end{pmatrix}.$$

Таким чином, поданий нейронний алгоритм дозволяє розв'язувати СЛР довільного виду. **Simu-link-схема розв'язування** чіткої СЛР сьомого порядку наведена на рис. 2–5.

Наведемо приклад розв'язування нечіткої системи сьомого порядку. Кожний елемент нечіткої матриці **A** та нечіткого вектора **b** створені з відповідних чітких за принципом $a_{ij} \pm (0.15 \ldots 0.25)$*rand(1)*$a_{ij}$ ± 0,1 та $b_i \pm$ (0.15 … 0.25)*rand(1)*$b_i \pm 0{,}1$, де rand(1) – генератор випадкових чисел в інтервалі [0,1], які являють собою інтервали (наприклад, нечітка одиниця може бути уявлена у вигляді нейронної мережі {0.75/0.1,1/1,1.25/0.1}):

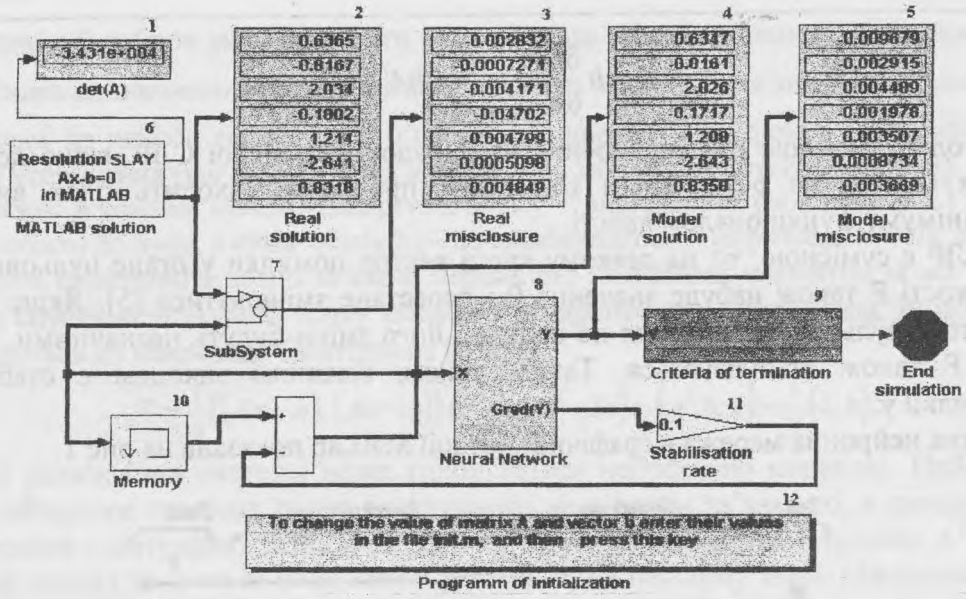

Рис.2. Simulink-схема нейронної мережі для розв'язування СЛАР:
1 – визначення детермінанта матриці **A** вхідної СЛАР (det (**A**)); *2* – відображення розв'язування вхідної СЛАР засобами MatLab (метод Гаусса); *3* – відображення значень нев'язки (метод Гаусса); *4* – розв'язання СЛАР за допомогою нейромережі; *5* – відображення значень нев'язки; *6* – розв'язання СЛАР засобами MatLab: *7* – підсистема визначення нев'язок; *8* – нейронна мережа; *9* – критерій закінчення роботи (квадрат похибки < Δ); *10* – ітераційна процедура (зміна початкового наближення (x^{Π}= {1,1,1,1,1,1,1})); *11* – коефіцієнт стабілізації; *12* –програма ініціалізації (*m*-file)

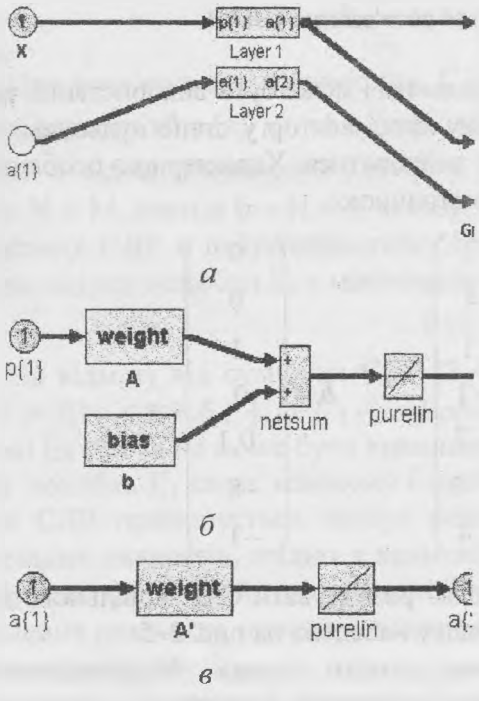

Рис. 3. Нейронна мережа (блок 8) :
а – загальна схема нейронної мережі; *б* – перший шар нейронної мережі; *в* – другий шар нейронної мережі

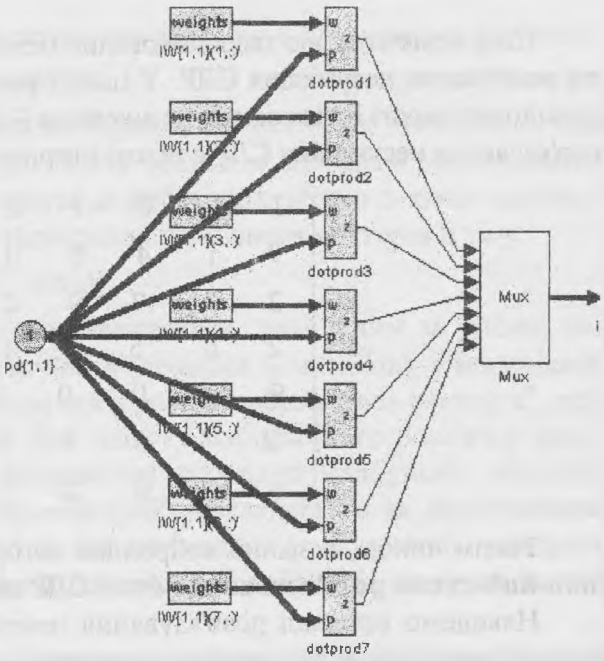

Рис. 4. Структура та значення вагових коефіцієнтів першого шару нейронної мережі (блок 8)

$$\tilde{A}=\begin{pmatrix} \tilde{1} & \tilde{1} & \tilde{0} & -\tilde{1} & \tilde{0} & \tilde{0} & \tilde{0} \\ \tilde{9} & \tilde{1} & \tilde{4} & \tilde{0} & \tilde{0} & .0\tilde{2} & \tilde{4} \\ \tilde{2} & \tilde{4} & .\tilde{7} & .\tilde{9} & \tilde{0} & \tilde{0} & \tilde{1} \\ \tilde{2} & \tilde{0} & \tilde{5} & \tilde{7} & \tilde{1} & \tilde{4} & -\tilde{1} \\ \tilde{9} & -\tilde{2} & \tilde{0} & \tilde{0} & \tilde{1} & \tilde{5} & \tilde{7} \\ -\tilde{2} & \tilde{3} & .\tilde{6} & \tilde{8} & \tilde{9} & \tilde{2} & -\tilde{3} \\ \tilde{0} & .0\tilde{1} & \tilde{0} & .\tilde{2} & \tilde{3} & \tilde{3} & \tilde{4} \end{pmatrix}; \quad \tilde{b}=\begin{pmatrix} \tilde{0} \\ \tilde{9} \\ \tilde{1} \\ \tilde{0} \\ 0.\tilde{1} \\ -\tilde{8} \\ -\tilde{1} \end{pmatrix}.$$

Чіткі системи, створені на α-рівнях 0.1(зліва), 1, 0.1(справа), дали розв'язки:
x1(α=0.1(зліва))={0.610 -0.800 2.990 -0.230 1.350 -2.870 0.840}/0.1 ;
x2(α=1)={0.640 -0.820 2.030 -0.180 1.210 -2.640 0.830}/1 ;
x3(α=0.1(справа)={0.650 -0.830 1.760 -0.170 1.130 -2.450 0.680}/0.1.

Нечіткий розв'язок : x = {x1 ∪ x2 ∪ x3}.

Дефадзифікований розв'язок : $= x^1_{def} = (\sum x^1_j.\alpha_j)/(\sum \alpha_j) =$

= { 0.6383 -0.8192 2.0875 -0.1833 1.2150 -2.6433 0.8183}.

Чіткий розв'язок загальної (перевизначеної) системи з розмірністю 7 × 21 відповідно становить x_{7x21} = {0.7757 -0.5325 1.0292 0.0034 0.7886 -1.8735 0.3693}.

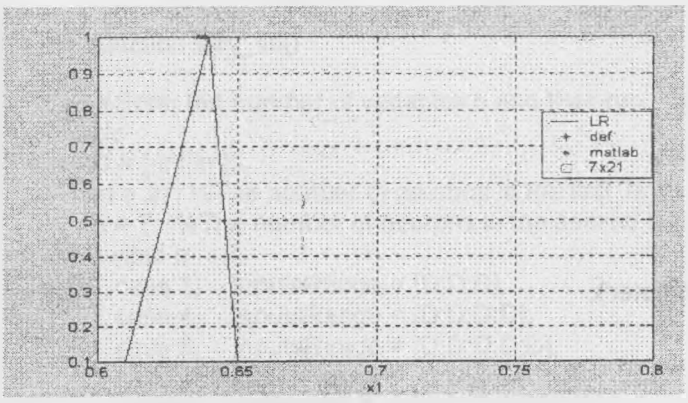

Рис. 5. Перший компонент нечіткого розв'язування СЛАР ($\tilde{x}_1$)

Таким чином, розв'язком нечіткої системи є інтервал [x_{def} , x_{7x21}] = = [0.6383,0.7757 ; -0.8192, -0.5325; 2.0875, 1.0292; -0.1833, 0.0034; 1.2150, 0.7886 ; -2.6433, -1.8735; 0.8183, 0.3693].

На рис.5 показано перший компонент нечіткого розв'язування СЛАР ($\tilde{x}_1$={0.610/0.1,0 640/1,0.650/0.1}).

Запропонований алгоритм розв'язування нечітких систем рівнянь, реалізований в середовищі MatLab/Simulink в нейромережному логічному базисі дозволяє суттєво розширити класи задач ідентифікацій та прогнозування в умовах невизначеності (див. програму розв'язку НСЛАР на вхідній мові MatLab).

```
% % Функція init_slay()  для  ініціювання СЛАР
function init_slay()
clc
A=[ 1   1    0   -1   0    0    0;...
   9   1  .4    0    0   .02   4;...
   2   4  .7   .9    0    0    1;...
   2   0   5    7    1    4   -1;...
   9  -2   0    0   .1    5    7;...
  -2   3  .6    8    9    2   -3;...
   0  .01  0   .2    3    3    4];

b=[ 0;  9;  .1;   0;.1;  -.8;  -1];
```

```
[m,n]=size(A);
for i=1:m
   for j=1:n
      if A(i,j)>=0
         A1(i,j) = A(i,j)-(0.3*rand(1))*A(i,j)-0.1;
         A2(i,j) = A(i,j)+(0.3*rand(1))*A(i,j)+0.1;
      else
         A1(i,j)=A(i,j)+(0.3*rand(1))*A(i,j)-0.1;
         A2(i,j)=A(i,j)-(0.3*rand(1))*A(i,j)+0.1;
      end
   end
    b1(i,:)=b(i)-(0.3*rand(1))*b(i)-0.1;
    b2(i,:)=b(i)+(0.3*rand(1))*b(i)+0.1;
end

%for initializing the SLAY matrix A and vector b
file_name = 'slay7x7';
NN='Neural Network 7x7';
h = 1;
switch h
    case 1% alpha level 0.1 from the left
       AA = [A1];
       bb = [b1];
    case 2% alpha level 1
       AA = [A];
       bb = [b];
    case 3   % alpha level 0.1 from the right
       AA = [A2];
       bb = [b2];
    case 4
        file_name = 'slay7x21';
        NN='Neural Network 7x21';
        AA = [A1; A; A2];
        bb = [b1; b; b2];
    otherwise
      disp(' You have to intered an invalid parameter');
end

[m,n] = size(AA);

%*************************************************************************************
%****** Fragment for changing the weights and the biases of the first layer
% loop for changing the weights of first layer
for i=1:m
  NewValue = mat2str(AA(i,:)');
  tmp = ['IW{1,1}/IW{1,1}(',int2str(i),',:)'"];
  str = [file_name,'/',NN,'/Layer 1/',tmp];
  set_param(str, 'Value', NewValue);
end
% changing biases of the first layer
NewValue = mat2str(-bb);
str = [file_name,'/',NN,'/Layer 1/b{1}'];
set_param(str, 'Value', NewValue);

%*************************************************************************************
%****** Fragment for changing the weights and the biases of the second layer
% loop for changing the weights of second layer
At = AA';
for i=1:n
  NewValue = mat2str(At(i,:));
```

```
  tmp = ['LW{2,1}/IW{2,1}(',int2str(i),',:)'"];
  str = [file_name,'/',NN,'/Layer 2/',tmp];
  set_param(str, 'Value', NewValue);
end

%*************************************************************************
%****** Fragment for initializing the memory initial condition (the vector X0)
X0(i:n,:)=1;
NewValue = mat2str(X0);
set_param([file_name,'/Memory'],'X0',NewValue)

%*************************************************************************
%****** Fragment for solving SLAY in MATLAB method Gauss
x = A\b;
NewValue = mat2str(x,6);
set_param([file_name,'/MATLAB solution/Solution Ax-b=0'],'Value',NewValue);
%******* Fragment for diplaying the determinant of A if it is square matrix
delta = det(A);
NewValue = mat2str(delta,6);
set_param([file_name,'/MATLAB solution/det(A)'],'Value',NewValue);

%*************************************************************************
%****** Savinng all changes to the model
save_system(file_name);
```

% % Функція slay_lib() для генерування нейронної мережі

```
function slay_lib()

%Enter the number of variables n and then run this program to generate a neural network block
clc
D = [-inf inf];
m = 21; %The number of neurons in the first  layer
n = 7;  %The number of neurons in the second layer
switch n
  case 3,   InputsRange = [D;D;D];
  case 4,   InputsRange = [D;D;D;D];
  case 5,   InputsRange = [D;D;D;D;D];
  case 6,   InputsRange = [D;D;D;D;D;D];
  case 7,   InputsRange = [D;D;D;D;D;D;D];
  case 8,   InputsRange = [D;D;D;D;D;D;D;D];
  case 9,   InputsRange = [D;D;D;D;D;D;D;D;D];
  case 10,  InputsRange = [D;D;D;D;D;D;D;D;D;D];
  case 11,  InputsRange = [D;D;D;D;D;D;D;D;D;D;D];
  case 12,  InputsRange = [D;D;D;D;D;D;D;D;D;D;D;D];
  case 13,  InputsRange = [D;D;D;D;D;D;D;D;D;D;D;D;D];
  case 14,  InputsRange = [D;D;D;D;D;D;D;D;D;D;D;D;D;D];
  case 21,  InputsRange = [D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D;D];

  otherwise
    disp('To creat a neural network block for SLAY greater than 21 variables');
    disp('modify the file slay_lib.m');
end

%*************** Architecture properties: ************************************
% Creating a new costum feedforward network of 2 layers with n neurons
net = network;
net.numInputs = 1;
net.inputs{1}.range=InputsRange; %The range of changing the inputs values
net.numLayers = 2;
```

```
%********** The first layer parameters **********
net.layers{1}.size = m;
net.layers{1}.transferFcn = 'purelin';
net.layers{1}.initFcn = 'initwb';
net.layers{1}.netInputFcn = 'netsum';
net.inputConnect(1,1)=1;
net.layerConnect(1,1)=0;
net.layerConnect(1,2)=0;
net.biasConnect(1) = 1;
net.biases{1}.learn=0;
net.inputWeights{1,1}.learn = 0;
net.layerWeights{1,1}.learn = 0;
net.outputConnect(1)=1;
net.targetConnect(1)=1;

%********** The second layer parameters **********
net.layers{2}.size = n;
net.layers{2}.transferFcn = 'purelin';
net.layers{2}.initFcn = 'initwb';
net.layers{2}.netInputFcn = 'netsum';
net.inputConnect(2,1)=0;
net.layerConnect(2,1)=1;
net.layerConnect(2,2)=0;
net.layerWeights{2,1}.learn = 0;
net.biasConnect(2) = 0;
net.biases{2}.learn=0;
net.outputConnect(2)=1;
net.targetConnect(2)=0,

%************************** PROGRAMM ************************************
net = init(net); % initializing the network
gensim(net,-1);  % generate a new neural network block of n variables

% % Функція graf() для  відтворення  результатів розв'язку НСЛАР
function graf()

mu=[0.1  1  0.1];
load x_def
load x_matlab
load x_7x21
load x_min
load x_nor
load x_max

for i=1:n
   x = [x_min(i)  x_nor(i)  x_max(i)];
  plot( x,mu,   x_def(i),1,'*',   x_matlab(i),1,'.', x_7x21(i),1,'s')
  legend('LR','def', 'matlab', '7x21');
  str = ['x' int2str(i) ];
  xlabel(str)
  grid on
  pause
end
```

Список літератури

1. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем.–М.: Мир, 1991.–312 с.

2. *Галушкин А.И., Судариков В.А.* Адаптивные нейросетевые алгоритмы решения задач линейной алгебры//Нейрокомпьютер.–1992 – № 1,2 – С.21–28.

3. *Сударикov В.А.* Исследование адаптивных нейросетевых алгоритмов решения задач линейной алгебры // Нейрокомпьютер – 1992. – №3,4 – С. 13–20.

4. *Горбань А.Н., Россиев Д.А.* Нейронные сети на персональном компьютере. – Новосибирск: Наука. Сиб. отд-ние, 1996.– 276 с.

5. *Агеев А.Д., Ильченкова З.В.* Решение систем линейных уравнений на нейронных структурах// Нейрокомпьютер.– 1997. – №1,2 – С. 35–38.

6. *A.Nikov, T.Georgiev. A* fuzzy neural network and its MatLab Simulation.Proc. of ITISS 21–st Intern. Conference Technology on Inform // Technology Interfaces. – 1999. – P. 413–418.

Стаття надійшла до редакції 13.06.01.

55К 6172.7

УДК 519.24:519.652.3

А.Я. Білецький, П.О. Приставка, О.О. Смойловська

АЛГОРИТМІЗАЦІЯ НЕПАРАМЕТРИЧНОЇ ОЦІНКИ ФУНКЦІЇ РОЗПОДІЛУ ЙМОВІРНОСТЕЙ ПРИ ОБРОБЦІ СТАТИСТИЧНИХ ДАНИХ

Запропоновано алгоритмізацію непараметричної оцінки функції розподілу та функції щільності ймовірності реалізацій випадкової величини за допомогою локальних поліноміальних сплайнів на основі В-сплайнів другого і третього порядку та обчислення стійких квантилів.

Задача відтворення розподілів є класичною. Необхідно відтворити функцію розподілу $F(x;\bar{\Theta}) = \mathrm{P}\{\omega: -\infty < \xi(\omega) < x\}$ імовірнісного простору $\langle \Omega, F, \mathrm{P} \rangle$ за даними вибіркового простору $\langle \Omega_n, \mathrm{A}, \mathrm{P}_n \rangle$, де $\Omega_n \in \Omega$, $\Omega_n = \{x_i; i = \overline{1,n}\}$ – вибірка. За P_n можливо брати оцінку емпіричної функції розподілу ймовірностей $F_n(x)$ або гістограмні оцінки.

Необхідність підвищення точності і достовірності статистичних висновків, одержуваних з умов експерименту, розв'язання ряду задач теорії надійності, масового обслуговування і розпізнавання образів обумовили введення сплайн-розподілів. Серед напрямків теоретичних і прикладних досліджень щодо сплайн-перетворень у задачах ймовірнісно-статистичного аналізу варто виділити відтворення і дослідження функцій розподілів (сплайн-розподілів) і функцій щільності (гістосплайнів) за вибірковими даними. Гістосплайни апроксимують гістограму або емпіричну функцію розподілу сплайн-функціями поліноміального або експоненціального типу і тим самим оцінюють щільність ймовірності за кінцевою вибіркою.

Інтерполювання і згладжування гістограм локальними поліноміальними сплайн-функціями, особливо параболічними і кубічними, проводяться за класичною схемою. Найбільш адекватні моделі функції щільності і розподілу отримують на основі усереднюючих сплайнів. Особливий клас функцій складають В-сплайни, які успішно реалізуються при відтворенні функції щільності за гістограмою.

Для відтворення функції щільності можна запропонувати такий підхід. Нехай задано масив спостережень $X = \{x_i, i = \overline{0, n-1}\}$, $n \geq 7$, за яким побудовано варіаційний ряд, розбитий на класи. Для цього фіксується рівномірне розбиття осі спостережень Δ_h на класи

$$x_i = x_{\min} + (i-1)h + h/2,$$