Anca MIRON
Andrei C. CZIKER

Ștefan UNGUREANU
Horia G. BELEIU

Aplicații ale IA în managementul energiei

Îndrumar de laborator

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-791-0

Anca MIRON **Ștefan UNGUREANU**
Andrei C. CZIKER **Horia G. BELEIU**

Aplicații ale IA în managementul energiei

Îndrumar de laborator

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-791-0

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Radu-Adrian Tîrnovan
Prof.dr.ing. Sorin G. Pavel

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-791-0

PREFAŢĂ

Prezentul material didactic este destinat să acopere activitatea practică din programa analitică a disciplinei *Aplicații ale Inteligenței Artificiale în Managementul Energiei,* pentru anul I, studii de master, specializarea energetică.

Îndrumarul de laborator a fost alcătuit ca un material didactic complementar orelor de curs, astfel încât noţiunile teoretice şi practice acumulate să aducă studenţilor masteranzi un plus de cunoştinţe în domeniul tehnicilor de inteligență artificială aplicate în inginerie energetică.

Concepute cu această abordare aplicativă, toate lucrările acestui material didactic sunt logic structurate și au o componentă teoretică corespunzătoare, necesară înțelegerii componentei practice.

Fiecare lucrare de laborator începe prin prezentarea clară a scopului lucrării, urmată de partea teoretică de care studenții au nevoie pentru soluționarea exercițiile propuse. Toate lucrările de laborator vor fi realizate prin utilizarea mediului de programare, modelare și simulare MATLAB, și a uneltelor predefinite dedicate dezvoltării de aplicații folosind tehnici de inteligență artificială.

Lucrările de laborator sunt împărțite în trei categorii, care se referă la cele trei tehnici de inteligență artificială prezentate în cadrul orelor de curs, și anume Logica Fuzzy, Rețele Neuronale Artificiale și Algoritmi Genetici. Astfel, fiecare categorie cuprinde două ședințe de laborator, în care nivelul de dificultate va crește gradual. Orele de laborator dedicate tehnicilor de IA menționate anterior vor începe cu o aplicație simplă, care apoi vor continua cu o aplicație mai complexă. Spre exemplu, în cazul Logicii Fuzzy prima aplicație este un controler fuzzy, urmată de un sistem complex în care se folosește și testează controlerul dezvoltat anterior.

Competențele dobândite la finalul orelor aplicative sunt menite să le permită absolvenților abordarea și soluționarea cu succes a problemelor întâlnite în calitate de specialişti energeticieni folosind tehnici de inteligență artificială.

Cu toate eforturile depuse, autorii sunt convinşi că această formă a prezentului material didactic poate fi îmbunătăţită prin continuarea preocupărilor în domeniu, prin sugestiile studenţilor care îl vor utiliza precum şi ale altor specialişti.

Autorii

CUPRINS

1

CONTROLUL FUZZY EXEMPLU NUMERIC

1.1. Scopul lucrării

Înțelegerea modului în care se proiectează, dezvoltă și funcționează un controler fuzzy de temperature.

Rolul controlerului este de a crește eficiența energetică a procesului de încălzire a unui spațiu de locuit.

1.2. Descrierea problemei

Reglarea temperaturii spațiului de locuit la consumatorii casnici de energie electrică se face în prezent, în funcție de temperatura interioară (din spațiu de locuit), fără să se aibă în vedere temperatura experioară (din afara spațiului, care depinde de factori climatici și geografici).

Pentru creșterea eficienței energetice a procesului de încălzire prin utilizarea eficientă a surselor de energie, o soluție este adaptarea temperaturii agentului termic (din instalația de încălzire) în funcție atât de termepratura interioară, cât și de cea exterioară. Această soluție se poate implementa prin folosirea unui controler (regulator) fuzzy de temperatură.

1.3. Controler fuzzy

Regulator (controlerul) fuzzy de temperatură are ca date de intrare temperatura interioară din încăpere și temperatura exterioară. Controlerul va furniza la ieșire tensiunea de alimentare a rezistenței folosită pentru a încălzi agentul termic (care curge prin calorifere). Se consideră că încălzirea se face prin intermediul unei instalații care cuprinde un dispozitiv electrotermic dedicat, care încălzește agentul termic, și un sistem de distribuție a agentului termic până la calorifere.

Dezvoltarea controlerului fuzzy presupune parcurgerea a șase etape:

1.3.1. Determinarea datelor de intrare și de ieșire

Alegerea datelor de intrare și ieșire, respectiv definirea caracteristicilor acestora se realizează după analiza procesului energetic care este controlat, și determinarea variabilelor care influențează procesul energetic și soluțiile posibile pentru creșterea eficienței acestuia.

Pentru definirea caracteristicilor datelor de intrare și ieșire este necesar să existe informații privind modul de variație a acestora.

Se consideră că datele de intrare ale controlerului de temperatură sunt T_{in} și T_{ext}, iar ieșirea este notată cu U.

1.3.2. Fuzzificarea datelor de intrare și de ieșire

Fuzzificarea este etapa prin care datele de intrare și ieșire, care inițial au valori singulare, sunt transformate în date fuzzy prin utilizarea unor variabile lingvistice și a unor numere fuzzy atașate acestora. Numere fuzzy care sunt folosite în cadrul laboratorului sunt numere fuzzy triunghiulare, trapezoidale, respectiv rampă / pantă.

T_{in} – temperatura interioară

T_{in} = [8, 24] = {rece [8, 12], bine [10, 20], cald [18, 24]}

Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

$$\mu_{T_{in}-rece} = \begin{cases} 1, & 8 \le T_{in} < 10 \\ \frac{12-T_{in}}{12-10}, & 10 \le T_{in} \le 12 \\ 0, & T_{in} > 12 \end{cases} \quad (1.1)$$

$$\mu_{T_{in}-bine} = \begin{cases} \frac{T_{in}-10}{15-10}, & 10 \le T_{in} \le 15 \\ \frac{20-T_{in}}{20-15}, & 15 \le T_{in} \le 20 \\ 0, & T_{in} > 20, T_{in} < 10 \end{cases} \quad (1.2)$$

$$\mu_{T_{in}-cald} = \begin{cases} 1, & 20 \le T_{in} < 24 \\ \frac{T_{in}-18}{20-18}, & 18 \le T_{in} \le 20 \\ 0, & T_{in} < 18 \end{cases} \quad (1.3)$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. 1.1.

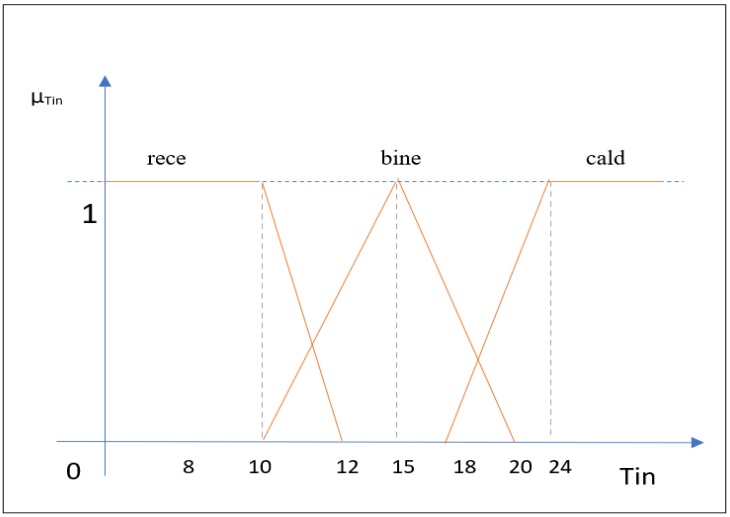


Fig. 1.1. Reprezentarea grafică a variabilei lingvistice T_{in}

T_{ext} – temperatura exterioară

T_{ext} = [-20, 21]
T_{ext} = {rece [-20, 0], bine [0, 15], cald [10, 21]}

Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

$$\mu_{T_{ext}-rece} = \begin{cases} 1, & -20 \leq T_{ext} < -10 \\ \frac{0-T_{ext}}{0-(-10)}, & -10 \leq T_{ext} \leq 0 \\ 0, & T_{ext} > 0 \end{cases} \quad (1.4)$$

$$\mu_{Text-bine} = \begin{cases} \frac{T_{ext}-(-5)}{0-(-5)}, & -5 \leq T_{ext} \leq 0 \\ 1, & 0 < T_{ext} \leq 10 \\ \frac{15-T_{ext}}{15-10}, & 10 < T_{ext} \leq 15 \\ 0, & T_{ext} > 15, T_{ext} < -5 \end{cases} \quad (1.5)$$

$$\mu_{T_{ext}-cald} = \begin{cases} 1, & 20 \leq T_{ext} < 21 \\ \frac{T_{ext}-10}{20-10}, & 10 \leq T_{ext} \leq 20 \\ 0, & T_{ext} < 10 \end{cases} \quad (1.6)$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. 1.2.

U – tensiunea

U = [180, 250] = {min [180, 210], med [205, 235], max [225, 250]}

Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

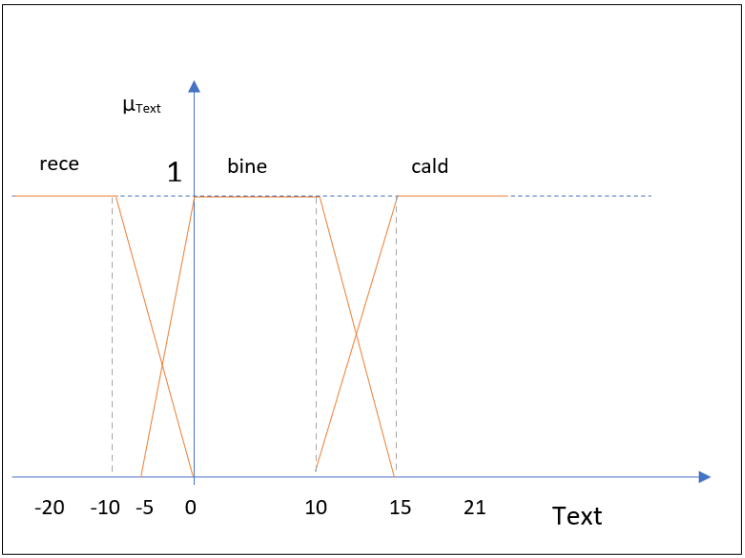


Fig. 1.2. Reprezentarea grafică a variabilei lingvistice T_{ext}

$$\mu_{U-min} = \begin{cases} \frac{U-180}{195-180}, 180 \leq U \leq 195 \\ \frac{210-U}{210-195}, 195 < U \leq 210 \\ 0, U > 210, U < 180 \end{cases} \quad (1.7)$$

$$\mu_{U-med} = \begin{cases} \frac{U-205}{220-205}, 205 \leq U \leq 220 \\ \frac{235-U}{235-220}, 220 < U \leq 235 \\ 0, U > 235, U < 205 \end{cases} \quad (1.8)$$

$$\mu_{U-max} = \begin{cases} \frac{U-235}{235-240}, 235 \leq U \leq 240 \\ \frac{250-U}{250-240}, 240 < U \leq 250 \\ 0, U > 250, U < 235 \end{cases} \quad (1.9)$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. 1.3.

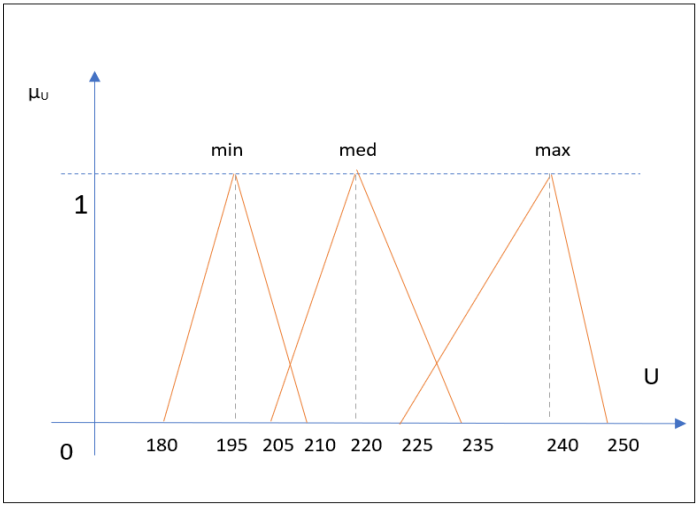


Fig. 1.3. Reprezentarea grafică a variabilei lingvistice U

1.3.3. Definierea tabelului de inferențe / a regulilor de inferență

Tabelul de inferențe definește relațiile dintre datele de intrare și ieșire, și reprezintă o formă compactă de a reprezenta regulile de inferență dintre date. În exemplu din lucrare, o regulă de inferență are forma:

Dacă T_{in}=rece ȘI T_{ext}=bine, Atunci U=med.

Tabelul de inferențe este detaliat în continuare, în tabelul 1.1.

Tabelul 1.1. Tabelul de inferențe a controlerului fuzzy de temperatură

U		T_{ext}		
		rece	**bine**	**cald**
T_{in}	**rece**	max	max	med
	bine	max	med	min
	cald	med	min	min

1.3.4. Alegerea metodei de rezolvare a inferențelor

Metodele de rezolvare a inferențelor prezentate în cadrul laboratorului sunt metoda Max-Min, Max-Prod și Sum-Prod. În continuare este exemplificată numeric fiecare metodă de rezolvare a inferențelor.

Metoda Max-Min

În folosirea metodei Max-Min, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – minim
SAU – maxim
Atunci – minim
SAU (dintre reguli) - maxim

Exemplu

Reguli selectate

1. Dacă T_{in}=rece (0,2) ȘI T_{ext}=rece(0,3), Atunci U=max
2. Dacă T_{in}=bine (0,3) ȘI T_{ext}=bine(0,4), Atunci U=med
3. Dacă T_{in}=cald (0,2) ȘI T_{ext}=cald(0,8), Atunci U=min
4. Dacă T_{in}=rece (0,5) ȘI T_{ext}=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Minim (0,2 ; 0,3) => 0,2 => minim (U=max ; 0,2) => număr fuzzy trapezoidal cu înălțimea 0,2
2. Minim (0,3 ; 0,4) => 0,3 => minim (U=med ; 0,3) => număr fuzzy trapezoidal cu înălțimea 0,3
3. Minim (0,2 ; 0,8) => 0,2 => minim (U=min ; 0,2) => număr fuzzy trapezoidal cu înlălțimea 0,2
4. Minim (0,5 ; 0,5) => 0,5 => minim (U=max ; 0,5) => număr fuzzy trapezoidal cu înlălțimea 0,5
5. Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umin^{0,2}$, $Umax^{0,5}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,3}$, $Umin^{0,2}$, $Umax^{0,5}$)

Metoda Max-Prod

În folosirea metodei Max-Prod, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – produs
SAU – maxim

Atunci – produs
SAU (dintre reguli) - maxim

Exemplu
Reguli selectate

1. Dacă T_{in}=rece (0,2) ȘI T_{ext}=rece(0,3), Atunci U=max
2. Dacă T_{in}=bine (0,3) ȘI T_{ext}=bine(0,4), Atunci U=med
3. Dacă T_{in}=cald (0,2) ȘI T_{ext}=cald(0,8), Atunci U=min
4. Dacă T_{in}=rece (0,5) ȘI T_{ext}=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Produs (0,2 ; 0,3) => 0,06 => Produs (U=max ; 0,06) => număr fuzzy triunghiular cu înălțimea 0,2
2. Produs (0,3 ; 0,4) => 0,12 => produs (U=med ; 0,12) => număr fuzzy triunghiular cu înălțimea 0,12
3. Produs (0,2 ; 0,8) => 0,16 => produs (U=min ; 0,16) => număr fuzzy triunghiular cu înlălțimea 0,16
4. Produs (0,5 ; 0,5) => 0,25 => produs (U=max ; 0,25) => număr fuzzy triunghiular cu înlălțimea 0,25
5. Maxim ($Umax^{0,06}$, $Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$)

Metoda Sum-Prod

În folosirea metodei Sum-Prod, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – produs
SAU – sumă
Atunci – produs
SAU (dintre reguli) - sumă

Exemplu
Reguli selectate

1. Dacă T_{in}=rece (0,2) ȘI T_{ext}=rece(0,3), Atunci U=max
2. Dacă T_{in}=bine (0,3) ȘI T_{ext}=bine(0,4), Atunci U=med
3. Dacă T_{in}=cald (0,2) ȘI T_{ext}=cald(0,8), Atunci U=min
4. Dacă T_{in}=rece (0,5) ȘI T_{ext}=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Produs (0,2 ; 0,3) => 0,06 => Produs (U=max ; 0,06) => număr fuzzy triunghiular cu înălțimea 0,2
2. Produs (0,3 ; 0,4) => 0,12 => produs (U=med ; 0,12) => număr fuzzy triunghiular cu înălțimea 0,12
3. Produs (0,2 ; 0,8) => 0,16 => produs (U=min ; 0,16) => număr fuzzy triunghiular cu înlălțimea 0,16
4. Produs (0,5 ; 0,5) => 0,25 => produs (U=max ; 0,25) => număr fuzzy triunghiular cu înlălțimea 0,25
5. Suma ($Umax^{0,06}$, $Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25+0,06}$)

1.3.5. Alegerea metodei de defuzzificare a suprafeței fuzzy de ieșire – determinarea valorii singulare de ieșire .

Pentru defuzzificare, în cadrul laboratorului sunt exemplificate metoda centrului de greutate, a centrului sumelor, respectiv a înălțimii. Relațiile matematice corespunzătoare fiecărei metode sunt enumerate în continuare.

Metoda centrului de greutate

Prin folosirea acestei metode, se determină valoarea singulară de ieșire ca fiind centrul de greutate a suprafeței fuzzy de ieșire. Dificultatea acestei metode constă în determinarea funcției de apartenență a suprafeței totale de ieșire.

$$U = \frac{A}{B} \tag{1.10}$$

Unde

$$A = \sum \int u \cdot \mu(u) du \tag{1.11}$$

$$B = \sum \int \mu(u) du \tag{1.12}$$

U – valoarea singulară de ieșire, u – valoare variabilă pe axa 0x, μ(u) – funcția de apartenență a suprafeței fuzzy

Metoda centrului sumelor

Prin folosirea acestei metode, se determină valoarea singulară de ieșire ca fiind centrul de greutate a fiecărei suprafețe fuzzy și apoi se determină suma algebrică. Comparativ cu metoda precedentă, în cazul acestei metode nu este necesar să se determine funcția suprafeței totale, ci se vor folosi funcțiile de apartenență a suprafețelor componente.

$$U = \frac{A}{B} \tag{1.13}$$

Unde

$$A = \sum \int u \sum \mu(u) du \tag{1.14}$$

$$B = \sum \int \sum \mu(u) du \tag{1.15}$$

Metoda înălțimii

În cazul utilizării metodei de defuzzificare a suprafeței de ieșire prin folosirea metodei înălțimii se vor folosi valorile înălțimilor suprafețelor componente ale suprafeței totale, și proiecțiile acestora pe axa 0x (valorile de pe axa 0x corespunzătoare acestora)

$$U = \frac{A}{B} \tag{1.16}$$

Unde

$$A = \sum u \cdot H \tag{1.17}$$

$$B = \sum H \tag{1.18}$$

H – înălțimea unei suprafețe componente, u – proiecția înălțimii pe axa 0x.

Se presupune suprafața fuzzy din fig. 1.4. Prin aplicarea celor trei metode de defuzzificare vom avea soluțiile:

Metoda centrului de greutate – relația (1.19)

$$U = \frac{\int_{180}^{195} u \cdot 0{,}5\frac{u-180}{5}du + \int_{195}^{208} u \cdot \left(0{,}44\frac{208-u}{13} + 0{,}06\right)du + \int_{208}^{220} u \cdot \left(0{,}44\frac{u-208}{12} - 0{,}14\right)du + \int_{220}^{228} u \cdot \left(0{,}44\frac{228-u}{8} - 0{,}14\right)du + \int_{228}^{240} u \cdot \left(0{,}64\frac{u-228}{12} + 0{,}06\right)du + \int_{240}^{250} u \cdot 0{,}7\frac{250-u}{10}du}{\int_{180}^{195} 0{,}5\frac{u-180}{5}du + \int_{195}^{208}\left(0{,}44\frac{208-u}{13} + 0{,}06\right)du + \int_{208}^{220}\left(0{,}44\frac{u-208}{12} - 0{,}14\right)du + \int_{220}^{228}\left(0{,}44\frac{228-u}{8} - 0{,}14\right)du + \int_{228}^{240} u \cdot \left(0{,}64\frac{u-228}{12} + 0{,}06\right)du + \int_{240}^{250} 0{,}7\frac{250-u}{10}du} \cong$$

$$223\ V \tag{1.19}$$

Metoda centrului sumelor – relația (1.20)

$$U = \frac{\int_{180}^{195} u \cdot 0{,}5\frac{u-180}{5}du + \int_{195}^{210} u \cdot \left(0{,}5\frac{210-u}{15} + 0{,}3\frac{u-205}{15}\right)du + \int_{205}^{220} u \cdot \left(0{,}3\frac{u-205}{15} + 0{,}5\frac{210-u}{15}\right)du + \int_{220}^{235} u \cdot \left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{225}^{240} u \cdot \left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{240}^{250} u \cdot 0{,}7\frac{250-u}{10}du}{\int_{180}^{195} 0{,}5\frac{u-180}{5}du + \int_{195}^{210}\left(0{,}5\frac{210-u}{15} + 0{,}3\frac{u-205}{15}\right)du + \int_{205}^{220}\left(0{,}3\frac{u-205}{15} + 0{,}5\frac{210-u}{15}\right)du + \int_{220}^{235}\left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{225}^{240}\left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{240}^{250} 0{,}7\frac{250-u}{10}du} \cong$$

$$222\ V \tag{1.20}$$

Metoda înălțimii – relația (1.21)

$$U = \frac{0{,}5*195+0{,}3*220+0{,}7*240}{0{,}7+0{,}5+0{,}3} = 221\ V \tag{1.21}$$

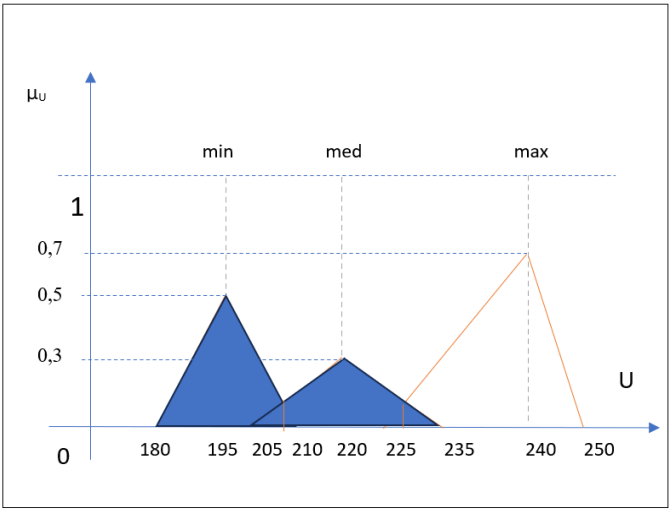


Fig. 1.4. Reprezentare grafică a suprafeței de ieșire

1.3.6. Testarea regulatorului.

Temperatura interioară se dorește a se menține la valoarea de 21°C.

Se consideră valoriile efective ale datelor de intrare:

T_{in} = 19 °C, T_{ext} = -3 °C.

Valoriile funcțiilor de apartenență a celor două date de intrare sunt:

$T_{in}^{bine} = 0,2$, $T_{in}^{cald} = 0,5$

$T_{ext}^{rece} = 0,3$, $T_{ext}^{bine} = 0,4$

Regulile corespunzătoare datelor de intrare sunt:

1. Dacă T_{in}=bine ȘI T_{ext}=rece, Atunci U=max
2. Dacă T_{in}=cald ȘI T_{ext}=rece, Atunci U=med
3. Dacă T_{in}=bine ȘI T_{ext}=bine, Atunci U=med
4. Dacă T_{in}=cald ȘI T_{ext}=bine, Atunci U=min

Rezolvarea inferențelor din regulile selectate

Metoda Max-Min

Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umed^{0,2}$, $Umin^{0,4}$) => suprafață fuzzy (formată din numere fuzzy trapezoidale) de ieșire formată din ($Umed^{0,3}$, $Umin^{0,4}$, $Umax^{0,2}$)

Metoda Max-Prod

Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umed^{0,2}$, $Umin^{0,4}$) => suprafață fuzzy (formată din numere fuzzy triunghiulare) de ieșire formată din ($Umed^{0,3}$, $Umin^{0,4}$, $Umax^{0,2}$)

Metoda Sum-Prod

Suma ($Umax^{0,06}$, $Umed^{0,15}$, $Umin^{0,2}$, $Umed^{0,08}$) => suprafață fuzzy (formată din numere fuzzy triunghiulare) de ieșire formată din ($Umed^{0,23}$, $Umin^{0,2}$, $Umax^{0,06}$)

1.4. Mersul lucrării

În cadrul acestei lucrări se vor realiza următoarele operații:

1. Se va determina valoarea de ieșire a controlerului fuzzy pentru datele de intrare:
    a. Temperatura exterioară este -15°C, temperatura interioară 21°C.
    b. Temperatura exterioară este 12°C, temperatura interioară 14°C.
    c. Temperatura exterioară este 5°C, temperatura interioară 11°C.

2. Se vor folosi cele trei metode de rezolvare a inferențelor, respectiv metode de defuzzificare.

3. Rezultatele se trec în tabelul 1.2.

Tabelul 1.2. Valorile testării controlerului fuzzy

Datele de intrare	**Metode de inferență**	**Defuzzificare**	**Date ieșire**
	Min-Max	Înălțimii	
	Max-Prod		
	Sum-Prod		
	Min-Max	Maximului din mijloc	
	Max-prod		
	Sum-Prod		
	Min-Max	Înălțimii	
	Max-Prod		
	Sum-Prod		
	Min-Max	Maximului din mijloc	
	Max-prod		
	Sum-Prod		
	Min-Max	Înălțimii	
	Max-Prod		
	Sum-Prod		
	Min-Max	Maximului din mijloc	
	Max-prod		
	Sum-Prod		

1.5. Prelucrarea datelor şi interpretarea rezultatelor

Datele din tabelul 1.2. se vor folosi pentru a determina care variantă de rezolvare a inferențelor, respectiv de defuzzificare este mai bună pentru problema studiată.

1.6. Concluzii

Se vor formula concluzii asupra utilității metodelor de defuzzificare și rezolvare a inferențelor.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2

CONTROLER FUZZY

2.1. Scopul lucrării

În lucrare se urmărește înțelegerea aspectele legate de realizarea unei aplicații dedicate modelării și simulării unui controler fuzzy folosind MATLAB/ Fuzzy Logic Designer.

2.2. Obiectivele lucării

Principalul obiectiv al lucrării este prezentarea noțiunilor teoretice privind realizarea unui controler fuzzy folosind instrumentul dedicat a mediului de modelare și simulare MATLAB. Deasemenea se vor prezenta pașii necesari în dezvoltarea controlerului fuzzy și se va testa operarea modelului fuzzy în funcție de parametrii controlerului.

2.3. Noţiuni teoretice. Fuzzy Logic Designer

Un controler fuzzy este compus din patru componente: blocul de fuzzificare, baza de reguli, blocul de rezolvare a inferențelor și blocul de defuzzificare.

Blocul de fuzzificare are rolul de a transforma valorile reale (clare) a mărimilor cu care lucrează controlerul în variabile fuzzy la care se asociază variabile lingvistice cu funcții de apartenență (numere fuzzy).

Baza de reguli cuprinde toate regulile care guvernează funcționarea controlerului și arată legătura dintre valorile datelor de intrare și ieșire.

Blocul de rezolvare a inferențelor are în componență un algoritm rezolutiv care combină regulile asociate unui scenariu de funcționare a controlerului obținute din baza de reguli. Ieșirea acestui bloc este reprezentată de suprafața fuzzy de ieșire.

Blocul de defuzzificare are implementat o metodă de defuzzificare prin care din suprafața fuzzy obținută după soluționarea inferențelor se extrage o valoare clară (singulară) a mărimii de ieșire, care va fi utilizată ulterior în procesul de control.

Mediul de modelare și simulare MATLAB oferă o unealtă dedicată implementării digitale a unui controler fuzzy. Această unealtă este Fuzzy Logic Designer (fig. 2.1. – MATLAB R2023a), care se accesează prin scrierea în linia de comandă a textului:

”fuzzy”

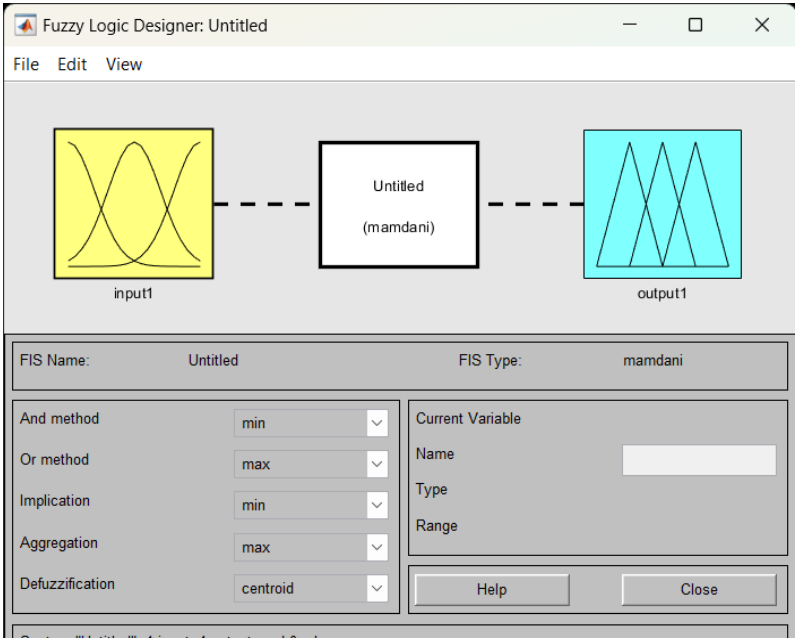


Fig. 2.1. Interfața grafică cu utilizatorul – Fuzzy Logic Designer

Unealta Fuzzy Logic Designer (FLD) oferă posibilitatea de a dezvolta un controler de tip Mamdani sau Sugeno:
File > New FIS > Mamdani sau Sugeno

Fișierele dezvoltate folosind FLD au extensia FIS, iar salvarea aplicației fuzzy se realizează prin acțiunea:
File > Export > To File

Deschiderea unui fișier FIS se face prin acțiunea:
File > Import > From File

2.3.1. Etapele dezvoltării unei aplicații FIS

Pașii necesari care trebuie parcurși pentru a realiza o aplicație folosind FLD sunt enumerați în continuare.

A. Variabilele de intrare și ieșire

Prima etapă este definirea numărului de variabile de intrare și ieșire prin acțiunea:
Edit > Add Variable > Input / Output

Numele fiecărei variabile se notează în fereastra principală a FIS. Această acțiune va avea drept rezultat deschiderea ferestrei de configurare a caracteristicilor variabilelor de intrare și ieșire, care se explică la pasul 4.

B. Selectarea modului de operare a blocului de inferență

Formatarea modului de operare a blocului de infereță poate fi realizată în orice etapă a dezvoltării controlerului.

Accesarea opțiunilor care țin de această etapă se poate face direct din fereastra principală FLD – fig. 2.1. sau prin acținuea
Edit > FIS Properties

Fig. 2.2. ilustrează focalizat opțiunile blocului de inferență.

”And method” se referă la operatorul logic ”ȘI” din interiorul regulilor. Acesta poate avea valorile: min (minim), prod (produs), sau Custom (definit de utilizator).

”Or method” se referă la operatorul logic ”SAU” din interiorul regulilor. Acesta poate avea valorile: max (maxim), probor (suma algebrică), sau Custom.

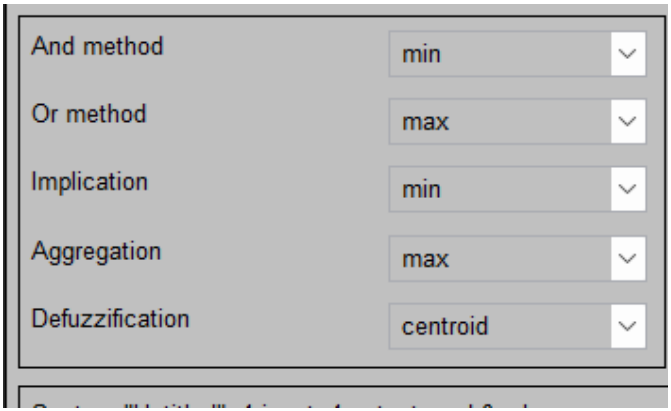


Fig. 2.2. Atributele blocului de inferență

”Implication” este atributul care este în legătură cu implicația ”Atunci” din interiorul unei reguli. Această implicație poate fi considerată ca min (minim), prod (produs), sau Custom.

”Aggregation” este operatorul prin care utilizatorul alege modul în care sunt agregate regulile folosite de blocul de inferență pentru a obține suprafața fuzzy de ieșire. Operatorul poate lua valorile max (maxim), sum (suma), probor (suma algebrică), sau Custom.

C. Selectarea metodei de defuzzificare

Metoda de realizare a defuzzificării se alege din fereastra principală a aplicației FIS, așa cum se observă în fig. 2.2.

”Defuzzification” are mai multe variante de operare, și anume

- Centroid – centrul de greutate al suprafeței fuzzy de ieșire.
- Bisector – determină valoarea bisectoarei care împarte suprafața de ieșire în două suprafețe de arii egale.
- Mom – maximul din mijloc.
- Lom – cel mai mare dintre valorile maxime.
- Som – cel mai mic dintre valorile maxime.

- Custom – definit de utilizator.

D. Definirea caracteristicilor variabilelor

Caracteristicile variabilelor sunt definite prin intermediul unei ferestre dedicate ilustrate în fig. 2.3. Această fereastră poate fi accesată prin acțiunea:

Edit > Membership Functions

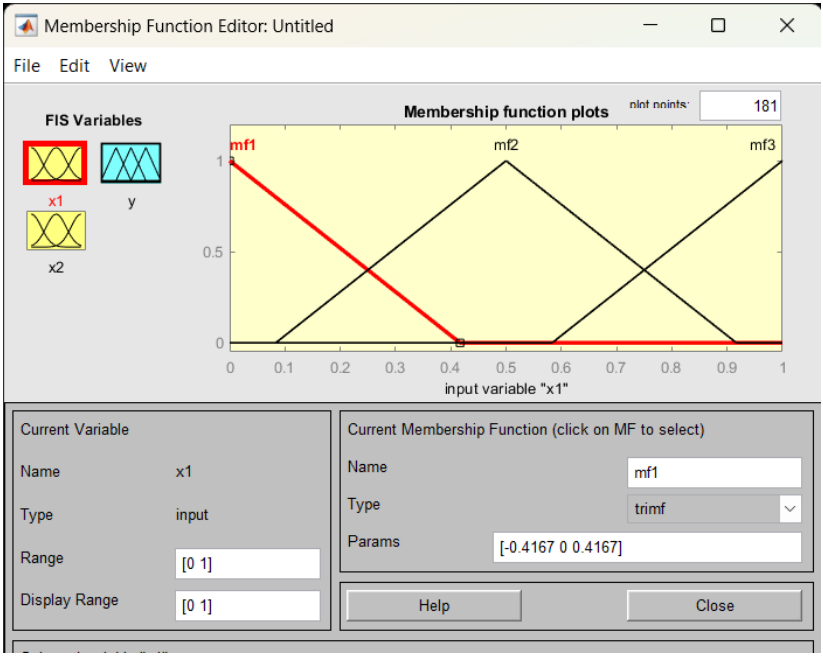


Fig. 2.3. Interfața cu utilizatorul dedicată funcțiilor de apartenență

Acțiunile corespunzătoare unei variabile se poate realiza după accesarea variabilei dorite. Astfel, pentru fiecare variabilă în parte se va defini intervalul de încredere (Range) și intervalul care se va afișa (Display Range). Tot aici se alege numărul de variabile lingvistice asociate fiecărei variabile, precum și tipul acestora și parametrii numărelor fuzzy.

Adăugarea de variabile lingvistice pentru o variabilă se înfăptuiește prin acțiunea:

Edit > Add MFs… (Membership Functions)

Această acțiune deschide o nouă fereastră, fig. 2.4. unde utilizatorul alege numărul și tipul de funcții de apartenență.

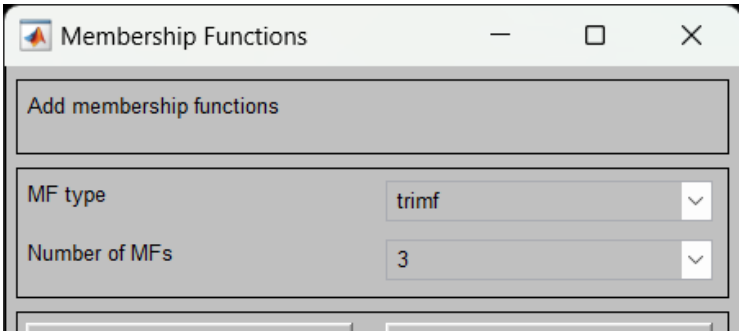
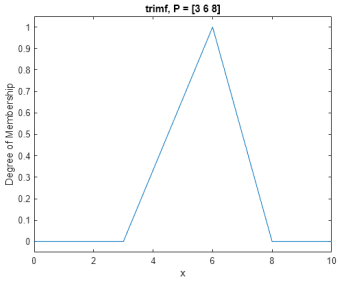


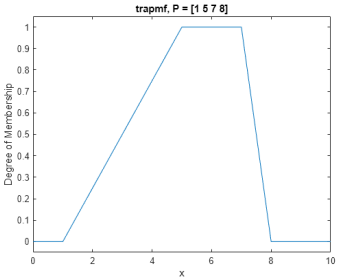
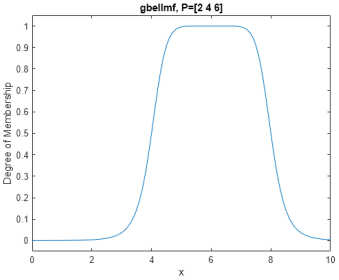
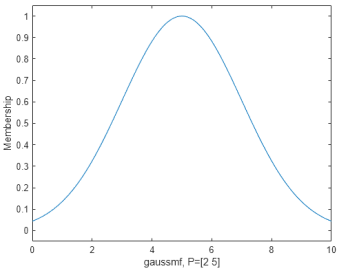
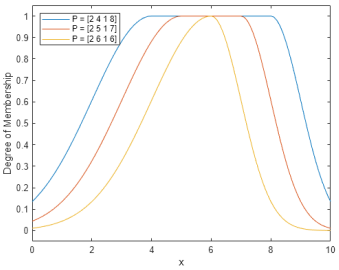
Fig. 2.4. Fereastra de adăugare a funcțiilor de apartenență

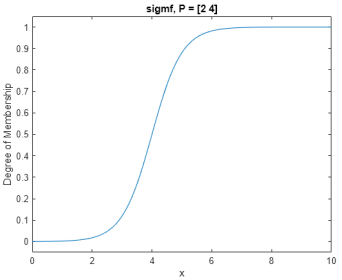
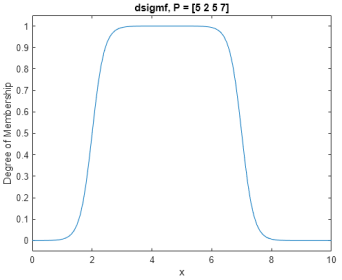
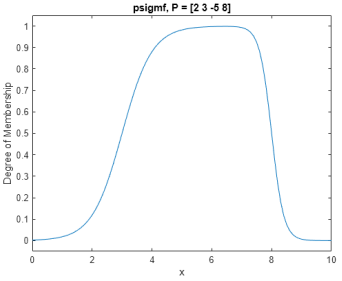
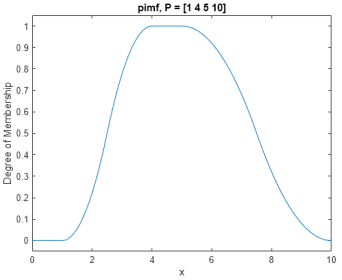
FLD oferă un număr mare de funcții de apartenență predefinite, care pot fi folosite de utilizatori. Aceste funcții sunt enumerate în tabelul 2.1.

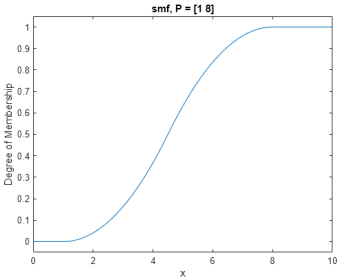
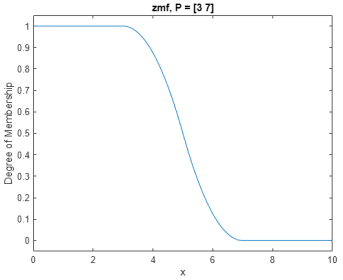
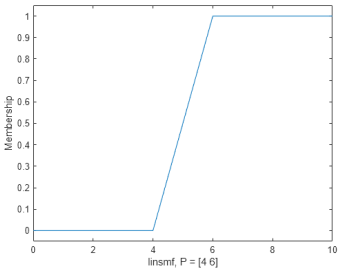
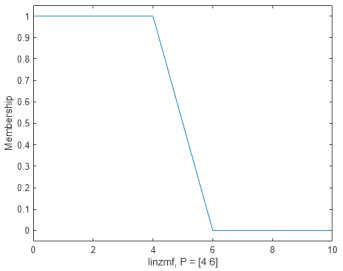
Pentru fiecare funcție de apartenență selectată, se poate modifica numele prin câmpul ”Name”, tipul prin câmpul ”Type”, respectiv se introduc parametrii în ”Params”. Acești parametrii sunt sunt specifici fiecărei funcții de apartenență. De exemplu, pentru numărul fuzzy triunghiular ”trimf” din fig. 2.4. trebuie introduse trei numere care corespund nucleului (funcția de apartenență are valoarea 1), respectiv extremităților (intersecția cu axa 0x).

Tabelul 2.1. Funcții de apartenență predefinite [1]

Simbol	Denumire	Reprezentare grafică
Trimf	Funcția triunghiulară	

Trapmf	Funcția trapezoidală	
Gbellmf	Funcția clopot generalizată	
Gaussmf	Funcția Gauss	
gauss2mf	Funcția Gauss de gradul 2	

Sigmf	Funcția Sigma	
Dsigmf	Diferența a 2 funcții sigma	
Psigmf	Produsul a 2 funcții sigma	
Pimf	Funcția Pi	

Smf	Funcția ”S”	
Zmf	Funcția ”Z”	
Linsmf	Funcția S liniară	
Linzmf	Funcția Z liniară	

Tot prin acțiunea amintită anterior se pot adăuga funcții de apartenență definite de utilizator, sau șterge funcții de apartenență care sunt asociate variabilelor de intrare sau ieșire.

E. Scrierea regulilor

Ultima etapă în modelarea controlerului fuzzy simulat (controler fuzzy virtual) este introducerea regulilor prin intermediul ferestrei din fig. 2.5. Această interfață pentru completarea bazei de reguli se accesează prin acțiunea

Edit > Rules…

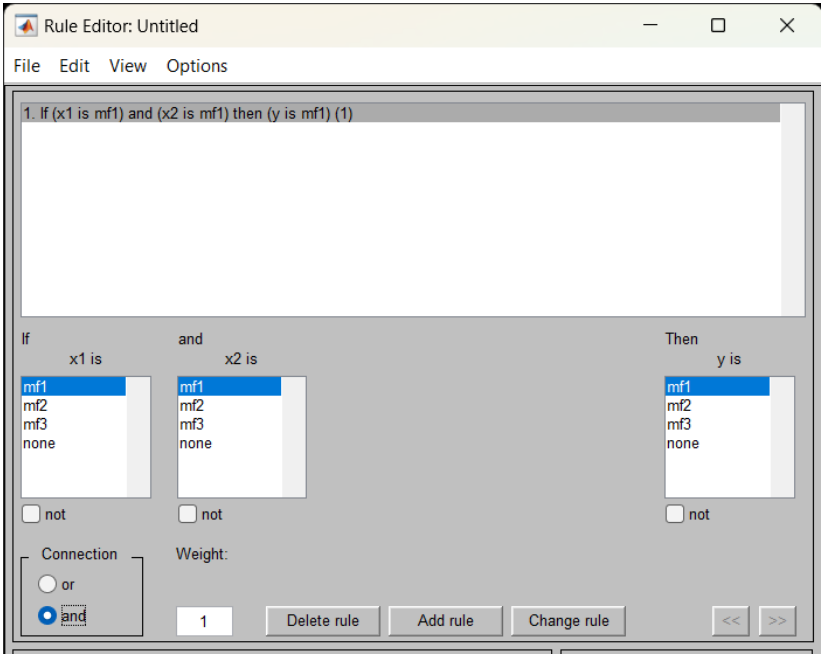


Fig. 2.5. Interfața cu utilizatorul dedicată bazei de reguli

Pentru a obține informații mai amănunțite despre interfețele și funcțiile de apartenență oferite de MATLAB se scrie în linia de comandă textul:

”help fuzzy”

2.3.2. Testarea locală a unei aplicații FIS

Testarea locală a controlerului fuzzy se referă la analiza suprafeței de ieșire rezultată în urma completării bazei de reguli, respectiv interogarea blocului de inferențe prin introducerea unor valori particulare a datelor de intrare. Astfel, fig. 2.6. ilustrează suprafața de ieșire a unui controler fuzzy cu două intrări și o ieșire, iar fig. 2.7. fereastra blocului de inferențe.

Suprafața fuzzy de ieșire se obține prin realizarea următoarei acțiuni:

View > Surface

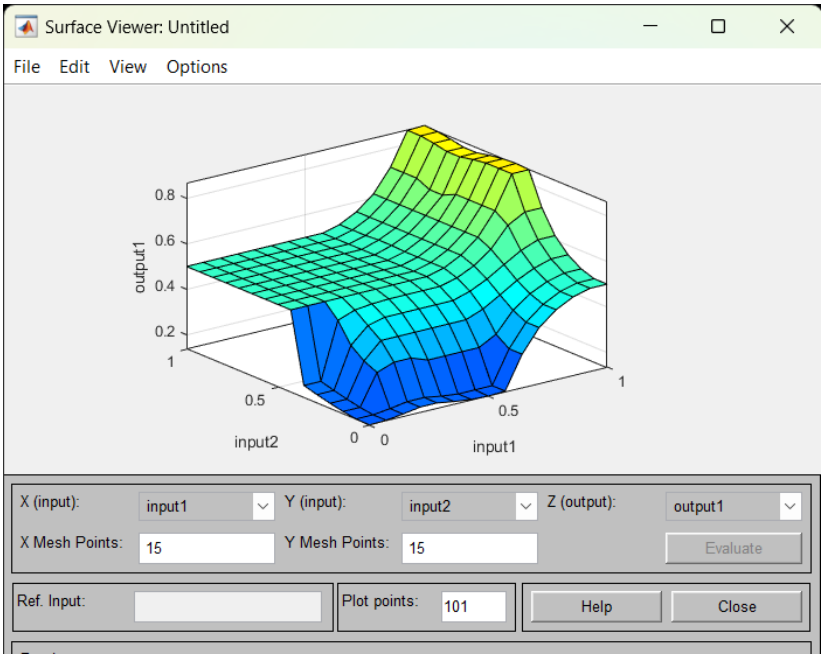


Fig. 2.6. Suprafața de ieșire a controlerului fuzzy

Fereastra blocului de inferență se accesează prin realizarea acțiunii

View > Rules

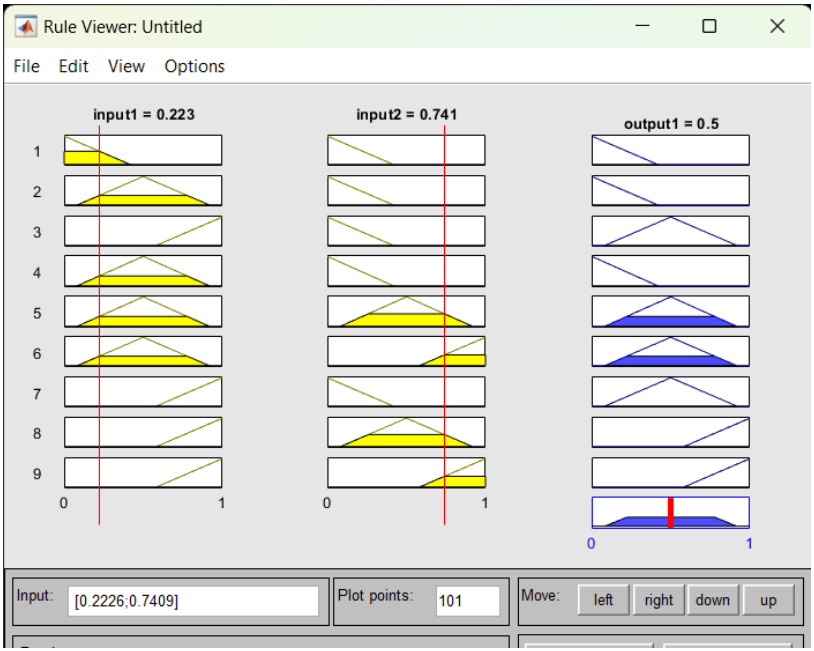


Fig. 2.7. Feresatra blocului de inferențe

Fereastra blocului de inferențe pune la dispoziția utilizatorului posibilitatea să analizeze funcționarea controlerului prin introducerea unor valori clare a datelor de intrare, acesta afișează valoarea de ieșire. Aici se afișează și numerele fuzzy a variabilelor de intrare și ieșire care sunt folosite, regulile corespunzătoare, și la final suprafața fuzzy obținută prin realizarea inferențelor.

Cazul particular din fig. 2.7. caracterizat prin
input1 = 0.223 și
input2 = 0.741
are la ieșire output1 = 0.5.

Trebuie menționat faptul că versiunile mai noi (actuale) de MATLAB conțin o variantă îmbunătățită (mai ușor de utilizat) a celei prezentate anterior. Aceasta este prezentată în fig. 2.8., și se accesează prin scrierea în linia de comandă a textului:
”fuzzyLogicDesigner”

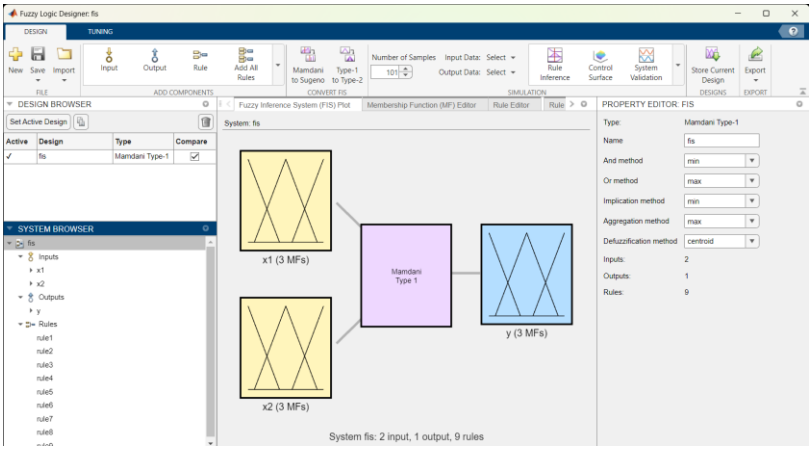


Fig. 2.8. Interfața cu utilizatorul a Fuzzy Logic Designer

Prin utilizarea noii variante a FLD se pot dezvolta mai multe tipuri de sisteme și controlere fuzzy, așa cum arată fereastra din fig. 2.9. Astfel, utilizatorul are posibilitatea să dezvolte controlere predifinite Mamdani de ordinul 1, Mamdani de ordinul 2, Sugeno de ordinul 1 și Sugeno de ordinul 2, dar și să construiască controlere digitale particulare.

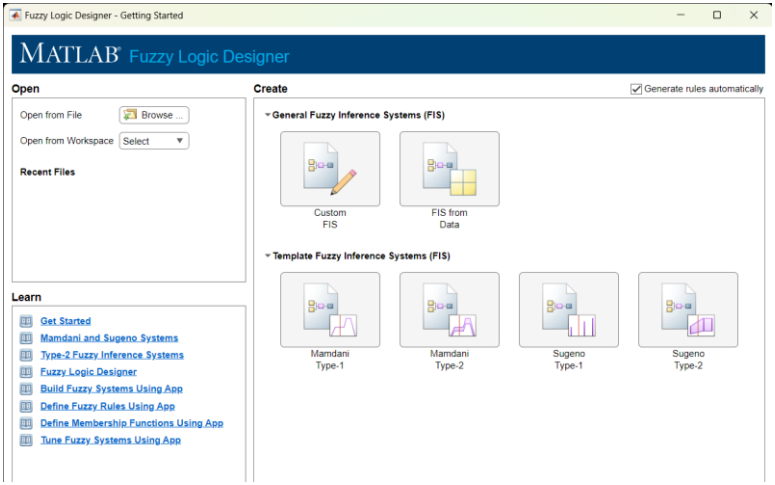


Fig. 2.9. Interfața cu uitilizatorul – fereastra inițială FLD

2.4. Mersul lucrării

În cadrul acestei lucrări se va implementa și studia funcționalitatea unui controler fuzzy folosind Fuzzy Logic Design. Pentru atingerea acestui obiectiv se vor realiza activitățile enumerate în continuare.

2.4.1. Controler fuzzy

Primul pas este dezvoltarea unui controler fuzzy urmând pașii descriși în capitolul 2.3. Controlerul va avea caracteristicile:

- Tip – Mamdani de ordinul 1.
- Variabile – două mărimi de intrare, notate x1 și x2, respectiv o mărime de ieșire y.
- Toate mărimile au un singur interval de încredere, și anume [-1, 1]
- Mărimile de intrare și ieșirea vor avea câte 3 variabile lingvistice N, Z, P. Aceste variabile lingvistice vor avea într-o primă etapă funcții de apartenență de tipul funcție triunghiulară, definite astfel:
  - N (-1.2 -1 -0.2),
  - Z (-0.8 0 0.8),
  - P (0.2 1 1.8).
- Regulile care leagă mărimile de intrare de ieșire se definesc folosind tabelul 2.2.

Tabelul 2.2. Regulile de inferență a controlerului fuzzy

Y		**x2**		
		N	Z	P
x1	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Pasul al doilea în dezvoltarea controlerului va fi testarea inițială a controlerului creat prin vizionarea suprafeței de ieșire integrale, și apoi a funcționării blocului de inferențe. Datele de intrare de testare vor fi x1= -0,4 și x2= 0,4.

2.4.2. Testarea controlerului

Odată finalizată dezvoltarea controlerului, se va testa luând în considerare datele de la subcapitolul precedent (x1= -0,4 și x2= 0,4) prin modificarea metodelor de defuzzificare, a tipului de funcții de apartenență folosite, respectiv atributele blocului de inferențe. Rezultatele se completează în tabelul 2.3.

Tabelul 2.3. Rezultatele testării controlerului fuzzy

Nr.	Blocul de inferențe	Defuzzificare	Tip număr fuzzy	Rezultate
1	Min, max, min, max	Centroid	Triunghiular	
2			Trapezoidal	
3			Gauss	
4			Clopot generalizat	
5		Bisector	Triunghiular	
6			Trapezoidal	
7			Gauss	
8			Clopot generalizat	
9	Prod, probor, prod, sum	Centroid	Triunghiular	
10			Trapezoidal	
11			Gauss	
12			Clopot generalizat	
13		Bisector	Triunghiular	
14			Trapezoidal	
15			Gauss	
16			Clopot generalizat	

2.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultatele obținute vor fi comparate între ele și cu datele de referință, apoi se va ridica un grafic care să evidențieze diferența dintre datele obținute și cele reale.

2.6. Concluzii

Se vor formula concluzii cu privire la

- Influența tipului de numere fuzzy asupra controlerului fuzzy ;
- Impactul metodei de defuzzificare asupra controlerului dezvoltat ;
- Cum afectează funcționarea controlerului fuzzy caracteristicile blocului de inferențe.

.....

.....

.....

.....

.....

.....

.....

.....

.....

3

CONTROLUL FUZZY A UNUI PROCES ENERGETIC

3.1. Scopul lucrării

Scopul lucrării este de a înțelege modul în care se dezvoltă și se utilizează un controler fuzzy folosind Matlab - Script.

3.2. Obiectivele lucrării

Lucrarea își propune dezvoltarea unui controler fuzzy pentru controlul intensității iluminatului artificial într-un spațiu interior (birou, clasă, hală etc.) astfel încât, iluminarea totală (naturală + artificială) să rămână în jurul unei valori optime (de ex. 500 lux), indiferent de variațiile luminii naturale (ex: nori, apus, trecerea unei persoane pe lângă fereastră).

3.3. Noţiuni teoretice

Iluminatul natural prezintă un potențial promițător în ceea ce privește reducerea consumului de energie în instalațiile de iluminat, și a devenit o alternativă atractivă la sistemele convenționale de iluminat interior artificial. Sistemele de reglare clasice, bazate pe modificarea continuă a fluxului luminos, prezintă o serie de dificultăți în a-și ajusta performanțele la modificarea rapidă a luminii

naturale în funcție de sezon, locație sau latitudine și gradul de acoperire a cerului[1].

Având în vedere aceste aspecte, reglarea fuzzy este o soluție potrivită în implementarea iluminatului mixt, un proces ce nu poate fi reprezentat ușor prin modelare matematică deoarece datele necesare pot fi prea complexe, iar în unele cazuri sunt incomplete sau chiar lipsesc. În acest context, în continuare se prezintă structura aplicației fuzzy pentru controlul iluminatului mixt.

Structura aplicației fuzzy pentru controlul iluminatului

Pentru a realiza o aplicație fuzzy este necesară parcurgerea următorii pași. Mai mult, pentru fiecare pas este integrat și sintaxa (codul) MatLab care se folosește pentru construcția aplicației. Trebuie precizat că sunt folosite funcțiile predefinite oferite de Matlab, cu care lucrează și instrumentul Fuzzy Logic Designer.

Pasul 1. Măsurători și achiziția datelor

Primul pas este reprezentat de obținerea datelor de intrare cu care se pot defini variabilele și valorile lingvistice, precum și numerele fuzzy asociate. Activitățile care se realizează la acest pas sunt enumerate succint în continuare.

- Senzori de lumină (fotodiode, luxmetre digitale), cu ajutorul cărora se măsoară:
    - Iluminarea naturală ($E_{natural}$, [lux]).
    - Iluminarea necesară în zona de lucru ($E_{necesar}$, [lux]).
- Calculul iluminării artificiale:

$$E_{artificial}=E_{necesar}-E_{natural} \quad (3.1)$$

[1] Cziker, A., Chindris, M., Miron, Anca. Pop Mihaela, *Utilizarea logicii fuzzy în controlul iluminatului mixt artificial/natural*, Simpozionul Naţional de iluminat, SINAIA 2007, „Soluţii moderne, de calitate pentru un iluminat eficient", 19 octombrie 2007, Sinaia, pag.48 – 55;

```
% Se creează sistemul fuzzy
fis = mamfis ('Name', 'RegulatorIluminare');
```

Ținând cont de cunoștințele dobândite în cadrul laboratoarelor anterioare, se poate trece la pasul 2.

Pasul 2. Fuzzyficarea datelor de intrare și ieșire

Variabilele fuzzy, care se recomandă să se definească prin folosirea numerelor trapezoidale pentru extremități și triunghiulare în rest:

2.1. Intrările

1. Iluminarea naturală – domeniu [0, 1000] [lux].
    - Valori lingvistice: Slabă, Normală, Puternică.

```
%% Intrarea 1: Iluminare Naturala [0, 1000]
fis = addInput(fis, [0 1000], 'Name', 'Iluminare_naturala');
fis = addMF(fis, 'Iluminare_naturala', 'trapmf', [-100 0 200 400], 'Name', 'Slaba');
fis = addMF(fis, 'Iluminare_naturala', 'trimf', [300 500 700], 'Name', 'Normala');
   fis = addMF(fis, 'Iluminare_naturala', 'trapmf', [600 800 1000 1100], 'Name', 'Puternica');
```

2. Eroarea iluminării – diferența între valoarea dorită și cea totală (iluminarea totală măsurată în zona de lucru: lumina naturală + artificială)
    - Eroare=$E_{tinta}-E_{total}$, domeniu [-500, +500] [lux].
    - Valori lingvistice: Negativa_Mare, Negativa_Mica. Zero, Pozitiva_Mica, Pozitiva_Mare.

```
%% Intrarea 2: Eroare Iluminare [-500, 500]
fis = addInput(fis, [-500 500], 'Name', 'Eroare_iluminare');
```

```
fis = addMF(fis, 'Eroare_iluminare', 'trapmf', [-600 -500 -400 -200], 'Name', 'Negativa_Mare');
fis = addMF(fis, 'Eroare_iluminare', 'trimf', [-300 -150 0], 'Name', 'Negativa_Mica');
fis = addMF(fis, 'Eroare_iluminare', 'trimf', [-50 0 50], 'Name', 'Zero');
fis = addMF(fis, 'Eroare_iluminare', 'trimf', [0 150 300], 'Name', 'Pozitiva_Mica');
fis = addMF(fis, 'Eroare_iluminare', 'trapmf', [200 400 500 600], 'Name', 'Pozitiva_Mare');
```

2.2. Ieșirea

1. Comanda iluminării artificiale (ex. dimmer)

   - domeniu [0, 100] [%].
   - Valori lingvistice: Scade mult, Scade puțin, Menține, Crește puțin, Crește mult.

```
%% Iesire: Comanda Dimmer [0, 100]
fis = addOutput(fis, [0 100], 'Name', 'Comanda_dimmer');
fis = addMF(fis, 'Comanda_dimmer', 'trapmf', [-10 0 10 30], 'Name', 'Scade_Mult');
fis = addMF(fis, 'Comanda_dimmer', 'trimf', [20 35 50], 'Name', 'Scade_Putin');
fis = addMF(fis, 'Comanda_dimmer', 'trimf', [45 50 55], 'Name', 'Mentine');
fis = addMF(fis, 'Comanda_dimmer', 'trimf', [50 65 80], 'Name', 'Creste_Putin');
fis = addMF(fis, 'Comanda_dimmer', 'trapmf', [70 90 100 110], 'Name', 'Creste_Mult');
```

Pasul 3. Reguli fuzzy

Al treilea pas îl reprezintă definirea și scrierea regulilor fuzzy, astfel încât, toate valorile lingvistice de intrare să se combine între

ele pentru a determina ieșierea corespunzătoare. Aceste reguli sunt descrise sub forma tabelară în tabelul 3.1.

Tabelul 3.1. Regulile fuzzy

Iluminare artificială		**Iluminare naturală**		
		Slabă	Normală	Puternică
Eroarea iluminării	Negativă Mare	Scade mult	Scade mult	Scade mult
	Negativă Mică	Scade putin	Scade putin	Scade putin
	Zero	Mentine	Mentine	Scade putin
	Pozitivă Mică	Creste putin	Creste putin	Mentine
	Pozitivă Mare	Creste mult	Creste mult	Creste putin

```
%% Reguli fuzzy (15)
rules = [
  "Iluminare_naturala==Slaba &
      Eroare_iluminare==Negativa_Mare =>
      Comanda_dimmer=Scade_Mult (1)"
  "Iluminare_naturala==Normala &
      Eroare_iluminare==Negativa_Mare =>
      Comanda_dimmer=Scade_Mult (1)"
  "Iluminare_naturala==Puternica &
      Eroare_iluminare==Negativa_Mare =>
      Comanda_dimmer=Scade_Mult (1)"
  "Iluminare_naturala==Slaba &
      Eroare_iluminare==Negativa_Mica =>
      Comanda_dimmer=Scade_Putin (1)"
  "Iluminare_naturala==Normala &
      Eroare_iluminare==Negativa_Mica =>
      Comanda_dimmer=Scade_Putin (1)"
  "Iluminare_naturala==Puternica &
      Eroare_iluminare==Negativa_Mica =>
      Comanda_dimmer=Scade_Putin (1)"
```

```
    "Iluminare_naturala==Slaba & Eroare_iluminare==Zero =>
        Comanda_dimmer=Mentine (1)"
    "Iluminare_naturala==Normala & Eroare_iluminare==Zero =>
        Comanda_dimmer=Mentine (1)"
    "Iluminare_naturala==Puternica & Eroare_iluminare==Zero
        => Comanda_dimmer=Scade_Putin (1)"
    "Iluminare_naturala==Slaba &
        Eroare_iluminare==Pozitiva_Mica =>
        Comanda_dimmer=Creste_Putin (1)"
    "Iluminare_naturala==Normala &
        Eroare_iluminare==Pozitiva_Mica =>
        Comanda_dimmer=Creste_Putin (1)"
    "Iluminare_naturala==Puternica &
        Eroare_iluminare==Pozitiva_Mica =>
        Comanda_dimmer=Mentine (1)"
    "Iluminare_naturala==Slaba &
        Eroare_iluminare==Pozitiva_Mare =>
        Comanda_dimmer=Creste_Mult (1)"
    "Iluminare_naturala==Normala &
        Eroare_iluminare==Pozitiva_Mare =>
        Comanda_dimmer=Creste_Mult (1)"
    "Iluminare_naturala==Puternica &
        Eroare_iluminare==Pozitiva_Mare =>
        Comanda_dimmer=Creste_Putin (1)"
];

fis = addRule(fis, rules);
```

Pasul 4. Defuzzificarea suprafeței fuzzy de la ieșire

Ultimul pas este reprezenta de definirea metodei de defuzzificare (în cazul în care nu se definește, automat este considerată metoda *centroid*). Sintaxa MatLab este următoarea.

```
fis.DefuzzificationMethod = 'centroid';
```

Informațiile prezentate în acest capitol, mai exact sintaxele MatLab, vor fi folosite pentru a realiza aplicația din cadrul ședinței de laborator.

3.4. Mersul lucrării

Pentru implementarea regulatorului fuzzy pentru controlul iluminatului artificial ținând cont de iluminarea naturală, se vor parcurge următorii pași:

1. Se scrie codul regulatorului fuzzy pentru controlul iluminatului artificial ținând cont de iluminatul natural.
2. Pentru a studia funcționarea regulatorului se vor definii valorile de intrare, respectiv cel de ieșire.

```
%% Introducere valori
prompt1 = 'Introdu valoarea iluminarii naturale [0-1000 lx]: ';
iluminare_nat = input(prompt1);

prompt2 = 'Introdu valoarea iluminarii existente [lx]: ';
iluminare_total = input(prompt2);

% Iluminare tinta fixa: 500 lx
etinta = 500;
eroare_iluminare = etinta - iluminare_total;

%% Evaluare FIS
iesire_dimmer = evalfis(fis, [iluminare_nat eroare_iluminare]);

fprintf('\nComanda dimmer: %.2f %%\n', iesire_dimmer);
```

3.5. Prelucrarea datelor și interpretarea rezultatelor

La final, controlerul fuzzy se va testa pentru a determina eficiența lui în procesul de control a iluminării spațiului interior. Pașii necesari sunt următorii:

1. Se va rula codul, și se vor introduce diferite valori pentru datele de intrare.
2. Se vor nota rezultatele obținute în tabelul 3.2. folosind două / trei metode de defuzzificare. În coloanele 5 și 6 se vor nota metodele de defuzzificare utilizate.

Tabelul 3.2. Rezultatele testării controlerului

Nr. Crt.	**Intrare**	**Valoare**	**Ieșire**		
			Centroid		
1	**2**	**3**	**4**	**5**	**6**
Varianta (3, 5, 5)					
1	Iluminare Naturală				
	Eroare Iluminare				
2	Iluminare Naturală				
	Eroare Iluminare				
3	Iluminare Naturală				
	Eroare Iluminare				
4	Iluminare Naturală				
	Eroare Iluminare				
Varianta (7, 7, 7)					
1	Iluminare Naturală				
	Eroare Iluminare				
2	Iluminare Naturală				
	Eroare Iluminare				
3	Iluminare Naturală				

Nr. Crt.	Intrare	Valoare	Ieșire		
			Centroid		
1	2	3	4	5	6
	Eroare Iluminare				
4	Iluminare Naturală				
	Eroare Iluminare				

Valorile notate în coloanele 5 și 6 se vor compara, pentru a determina cât de mult se abat de la metoda de defuzzificare centroid, metoda cea mai potrivită în cazul acestei aplicații.

Se va relua modelarea regulatorului, considerând definirea intrărilor și ieșirea prin mai multe valori lingvistice și numere fuzzy (7, 7, 7). La final, se vor compara rezultatele obținute cu cele din varianta (3, 5, 5).

3.6. Concluzii

Se vor formula concluzii cu privire la:

- Prin câte numere fuzzy să fie caracterizate mărimile de intrare și ieșirea.
- Ce metodă de defuzzificare se recomandă pentru ca regulatorul fuzzy să funcționeze optim pentru diferite situații.

.....

.....

.....

.....

.....

.....

.....

.....

.....

4

REȚELE NEURONALE ARTIFICIALE

4.1. Scopul lucrării

În lucrare se prezintă noțiunile de bază ale unei rețele neuronale artificiale și o aplicație dedicată implementării unui algoritm bazat pe rețele neuronale folosind MATLAB/ Deep Learning Toolbox, Statistics și Machine Learning Toolbox.

4.2. Obiectivele lucrării

Principalul obiectiv al lucrării este estimarea cu ajutorul rețelelor neuronale a unei funcții folosind strict date numerice obținute aleator pe baza ecuației (4.1).

$$Y = e^{(0.5 \cdot X)} \cdot sin(5 \cdot X) \tag{4.1}$$

Datele generate pentru antrenarea rețelei neuronale sunt reprezentate de un vector de 300 de elemente obținute prin suprapunerea datelor rezultate din ecuația (4.1) cu valori aleatori. Astfel, se poate compara vizual modul în care rețelele neuronale reușesc să generalizeze evoluția datelor. Scopul rețelei neuronale construite în această lucrare este să aproximeze o funcție care să descrie cât mai corect evoluția valorilor numerice.

$$Y = e^{(0.5 \cdot X)} \cdot sin(5 \cdot X) + 0.2 \cdot (|X| + 1) \cdot \beta \tag{4.2}$$

unde $\beta \in [0..1]$ este alocat aleator în intervalul [0..1]. În fig. 4.1 se pot vizualiza valorile obținute cu funcțiile prezentate în 1 (funcția ideală) și 2 (funcție generată aleator).

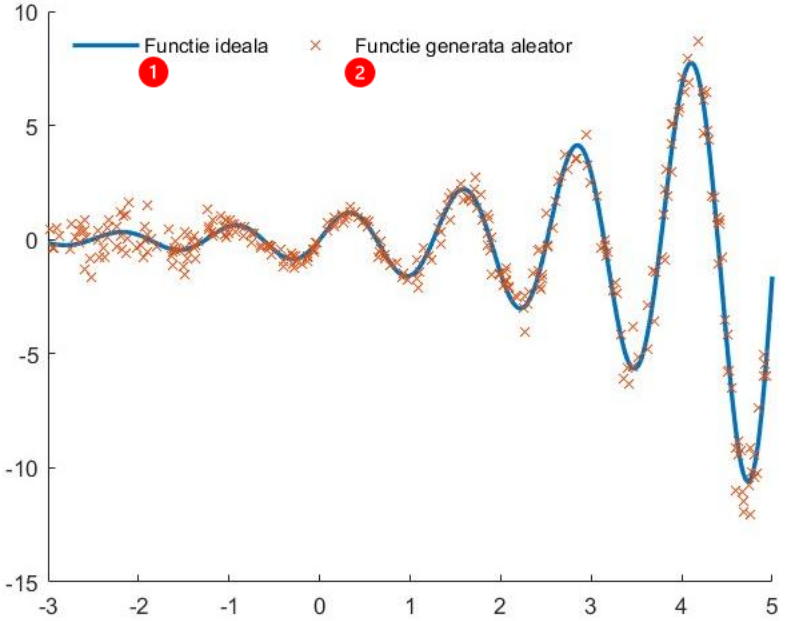


Fig. 4.1. Reprezentarea grafică a valorilor funcției ideale și valorile aleatoare

4.3. Noțiuni teoretice. Rețele neuronale artificiale

Rețelele neuronale artificiale, RNA sunt algoritmi matematici care fac parte din întreg conceptul de inteligență artificială (IA) și care încearcă să imite modul de funcționare a unei rețele neuronale biologice. Există un vast spectru terminologic pentru evidențierea algoritmilor de IA, respectiv învățare automată (machine learning), învățare profundă (deep learning), rețele neuronale artificiale (artificial neural networks) sau învățare profundă (deep learning). În fig. 4.2 se prezintă o clasificare a conceptelor de IA.

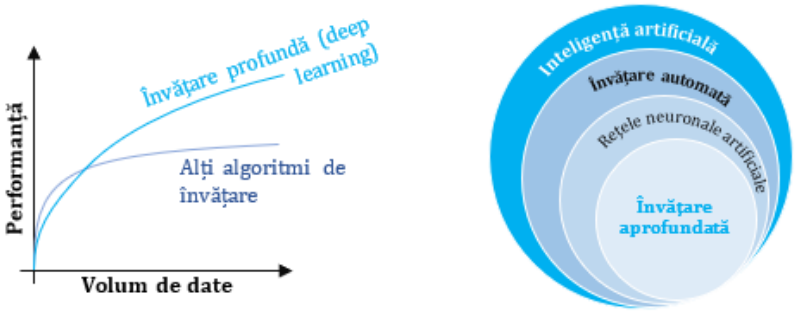


Fig. 4.2. Clasificare algoritmi de IA

Tehnicile bazate pe învățare automată prezentate în fig. 4.2 pot fi grupate în două grupe mari: învățare supravegheată și nesupravegheată. Metodele care utilizează învățarea supravegheată se bazează pe configurarea parametrilor specifici utilizând setul de date de antrenare împărțit în date de intrare și valori de ieșire cunoscute (reale, țintă). În fig. 4.3 se prezintă clasificarea rețelelor neuronale. În această lucrare se implementează rețele neuronale de tipul perceptron multistrat.

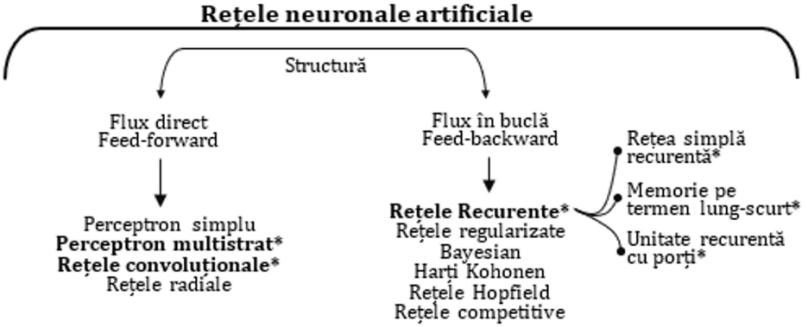


Fig. 4.3. Clasificare rețele neuronale

Elementele principale al rețelelor neuronale artificiale sunt neuronii artificiali (perceptronii). Stratul de intrare este format din N neuroni interconectați între ei în stratul ascuns. Neuronii din stratul ascuns sunt conectați la fiecare neuron din stratul de ieșire. Fiecare

neuron primește intrări de la alți câțiva neuroni care se înmulțesc cu ponderile și erorile aleatorii (bias) atribuite, care apoi sunt adunate și transmise unuia sau mai multor neuroni. Toată această structură sintetizată poate fi evidențiată grafic în fig. 4.4.

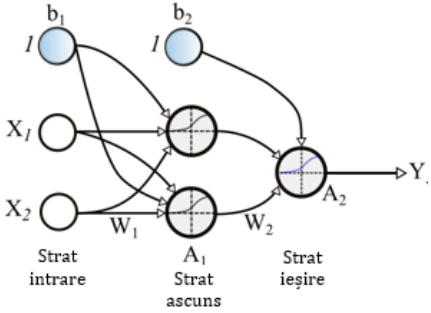


Fig. 4.4. Rețea neuronală perceptron multistrat

Neuronii artificiali sunt reprezentați printr-o funcție de activare care determină transmiterea calculului către următorul strat al rețelei. Pentru funcția de activare se pot folosi mai multe variante, în (4.3) se folosește o funcție prag:

$$Y = f(x) = W_1 \cdot X + b$$
$$Y = \begin{cases} 0, dacă\ Y \leq prag \\ 1, dacă\ Y > prag \end{cases} \quad (4.3)$$

Conexiunile sunt asociate cu ponderi (W) și erori aleatorii (b), care sunt ajustate în etapa de învățare pentru a obține rezultatul dorit în așa fel încât rezultatele întregului model să fie optimizate. Alegerea unei funcții de activare depinde de analiza empirică a rezultatelor, în (5.4) se prezintă funcția de activare sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$ieșire_1 = \sigma(W_1 \cdot X + b_1)$$
$$ieșire_2 = \sigma(W_2 \cdot A_1 + b_2) \quad (4.4)$$

unde X = intrări; W_1, W_2, b_1, b_2 = ponderi și erori aleatorii; $ieșire_1$, $ieșire_2$ = ieșirile din fiecare strat; σ – funcția de activare sigmoid; A_1 –

ieșirea din stratul ascuns. Eroarea aleatorie este o măsură care indică cât de ușoară este activarea neuronului.

4.3.1. Etapele dezvoltării unei aplicații cu RNA

Pași necesari care trebuie parcurși pentru a realiza o aplicație folosind Deep Learning Toolbox sunt enumerați în continuare. Se deschide un fișier Matlab (*.m) – fig. 4.5.

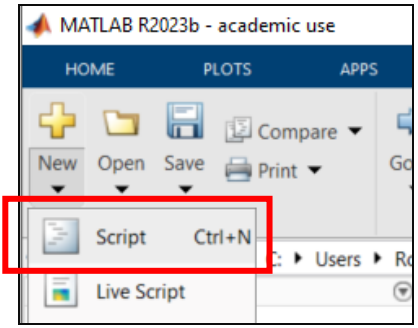


Fig. 4.5. Deschiderea fișierului unde se scriu liniile de cod

a. Definirea setului de date

Prima etapă este definirea variabilelor de intrare și ieșire prin generarea eșantionului de date pentru variabila X și obținerea rezultatului Y:

```
esant=300;
a=-3;
b=5;
x_ideal=-3:0.02:5;
y_ideal=exp(0.5*x_ideal).*sin(5*x_ideal);
x_aleator=a+(b-a).*rand(nsamp,1);
y_aleator=exp(0.5*x_aleator).*sin(5*x_aleator)  +
            0.2*(abs(x_aleator)+1).*randn(nsamp,1);
```

Se obțin două seturi de date, unul care este rezultat direct din funcția prezentată în (4.1) (x_ideal și y_ideal), iar al doilea set

cuprinde valorile determinate din această funcție la care se adaugă elementele valorile aleatorii ((4.2), respectiv x_aleator și y_aleator).

b. Vizualizarea grafică a datelor

```
figure; hold on;
plot1=plot(x_ideal, y_ideal, 'LineWidth',2);
plot2=plot(x_aleator, y_aleator,'x');
M1 = "Functie ideala";
M2 = "Functie generata aleator";
legend([plot1,plot2], [M1, M2]);
legend('Orientation','horizontal')
legend('boxoff')
legend('Location','northwest')
```

În urma rulării liniilor de cod anterior se pot vizualiza seturile de date folosite în antrenare (fig. 4.6.):

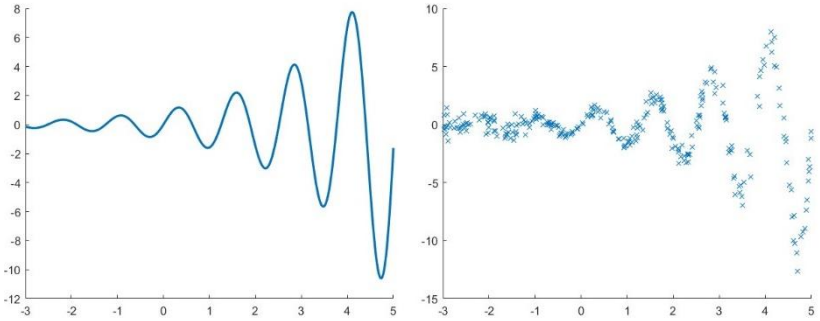


Fig. 4.6. Setul de date

c. Configurarea rețelei neuronale

Pentru configurarea rețelei neuronale trebuie setați parametrii cu ajutorul funcției *netconf*. În codul de mai jos, *netconf=[20 20 30]* ceea ce determină o rețea cu 3 straturi ascunse cu un număr de 20, 20 și 30 electroni în fiecare strat ascuns. Selectarea unei RNA feedforward se realizeaza cu funcția *feedforwardnet*. În procesul de antrenare se folosesc 1000 epoci, fiind o valoare default pentru

feedforwardnet. Pentru setarea manuală a numărului de epoci se poate folosi *net.trainParam.epochs = 1000.* Procesul de antrenare se oprește automat în modul *default* datorită unei setări *stop-loss,* care identifică dacă eroarea MSE *(mean square error)* începe să crească după o perioadă în care a scăzut. În cazul nostru, pentru evidențiere se va seta să antreneze continuu 1000 de epoci.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (4.5)$$

```
netconf=[20 20 30];
net=feedforwardnet(netconf);
net=trainbr(net,x_aleator.',y_aleator.');
```

Antrenarea rețelei neuronale se realizează cu funcția *trainbr,* care este un algoritm de învățare bazat pe regularizare Bayesiană. Pentru antrenarea rețelei se folosește setul de date pentru intrarea *x_aleator* și setul de date de ieșire *y_aleator.* În acest fel „potrivim" datele de intrare cu cele de ieșire prin configurarea ponderilor și bias-ului.

Tabelul 4.1. Funcții folosite pentru antrenarea RNA

Algoritm	Descriere
trainlm	Levenberg-Marquardt
trainbr	Bayesian Regularization
trainbfg	BFGS Quasi-Newton
trainrp	Resilient Backpropagation
trainscg	Scaled Conjugate Gradient
traincgb	Conjugate Gradient with Powell/Beale Restarts
traincgf	Fletcher-Powell Conjugate Gradient
traincgp	Polak-Ribiére Conjugate Gradient
trainoss	One Step Secant
traingdx	Variable Learning Rate Gradient Descent
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent

d. Realizarea estimării folosind RNA

Pentru testarea rețelei se folosește funcția *net* creată pe datele de intrare pentru a realiza estimări.

```
y_estim=net(x_ideal);
```

În acestă etapă rețeaua neuronală este construită și poate fi folosită pentru estimarea funcției care descrie evoluția datelor.

e. Vizualizarea grafică a datelor

În urma rulării liniilor de cod anterior și plotarea din codul de mai jos, se pot vizualiza grafic în fig. 4.7 setul de date folosite în antrenare și acuratețea cu care rețeaua neuronală aproximează funcția:

```
figure; hold on;
plot3=plot(x_aleator, y_aleator, 'x');
plot4=plot(x_ideal, y2pred, 'LineWidth',0.5);
M3 = "Functie generata aleator";
M4 = "Functie estimata de RNA";
legend([plot3,plot4], [M3, M4]);
legend('Orientation','horizontal')
legend('boxoff')
legend('Location','northwest')
```

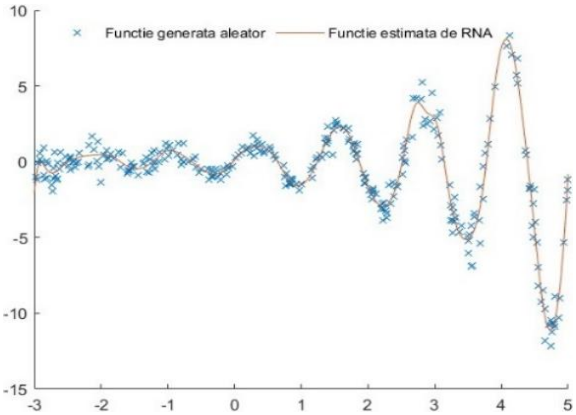


Fig. 4.5. Estimarea funcției de către RNA

Pentru o mai bună evidențiere în fig. 4.8 se reprezintă grafic valorile generate cu funcția din (1) – *y_ideal* și valorile generate cu funcția aproximată de RNA – *y_estim*.

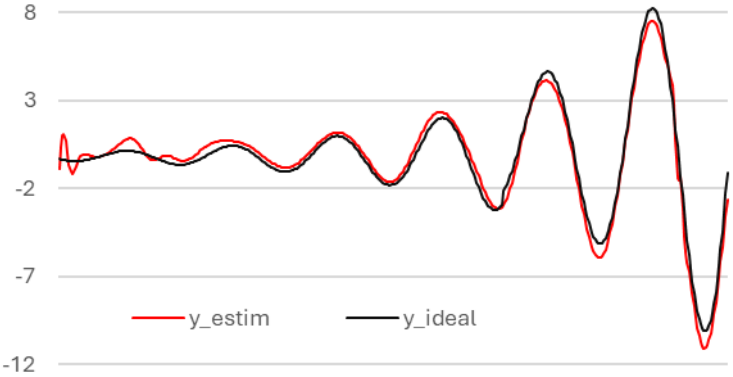


Fig. 4.8. Estimarea funcției de către RNA

4.3.2. Vizualizarea RNA folosite și parametrii antrenării

În cadrul toolbox-ului Deep Learning există o serie de opțiuni pentru vizualizarea și evaluarea performanțelor algoritmilor de IA. După rularea codului de la etapa **4.1.c** o să fie afișată o fereastră cu mai multe opțiuni conform fig. 4.8.

4.4. Mersul lucrării

În cadrul acestei lucrări se va implementa și studia funcționalitatea unei rețele neuronale artificiale cu ajutorul Toolbox-ului Deep Learning. Pentru a realiza acest obiectiv se vor realiza activitățile din următoarele subcapitole.

4.4.1. Rețea neuronală artificială

Primul pas este dezvoltarea unei RNA urmând pașii descriși în capitolul 4.3. Codul complet pentru dezvoltarea RNA este prezentat în secțiunea următoare.

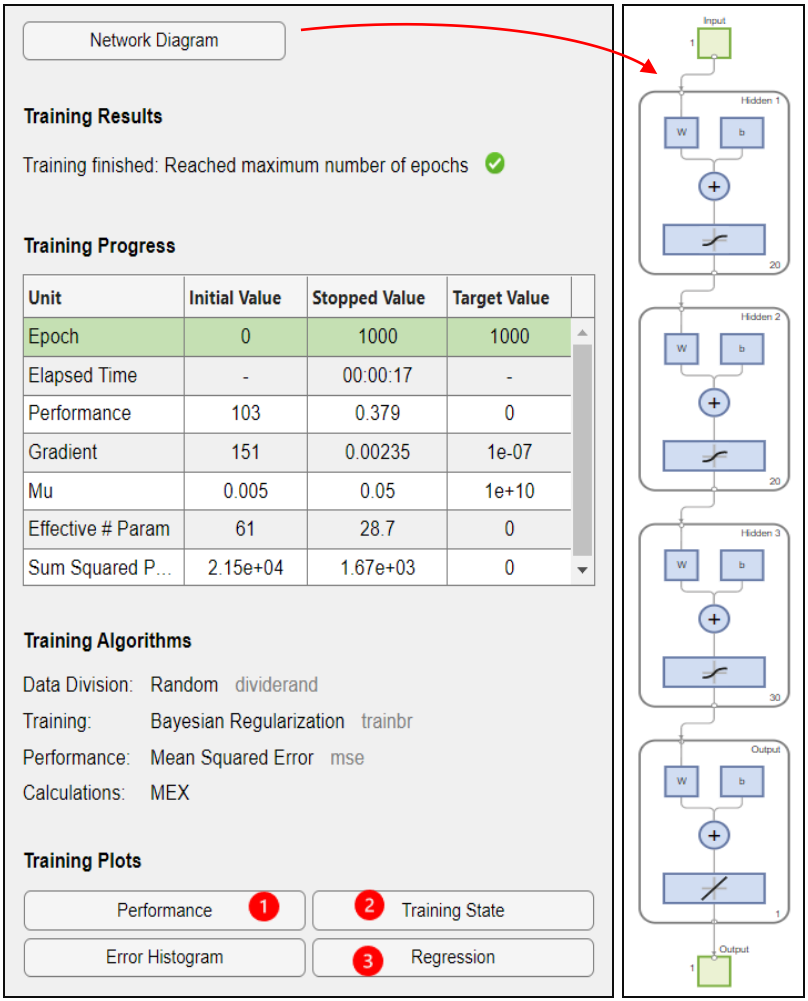


Fig. 4.8. Antrenarea și structura RNA

În fig. 4.9 se prezintă mai mulți indicatori pentru evaluarea performanței cu care RNA estimează valorile funcției. Accesând butoanele din fig. 4.8 se vor deschide ferestrele aferente prezentate în fig. 4.9.

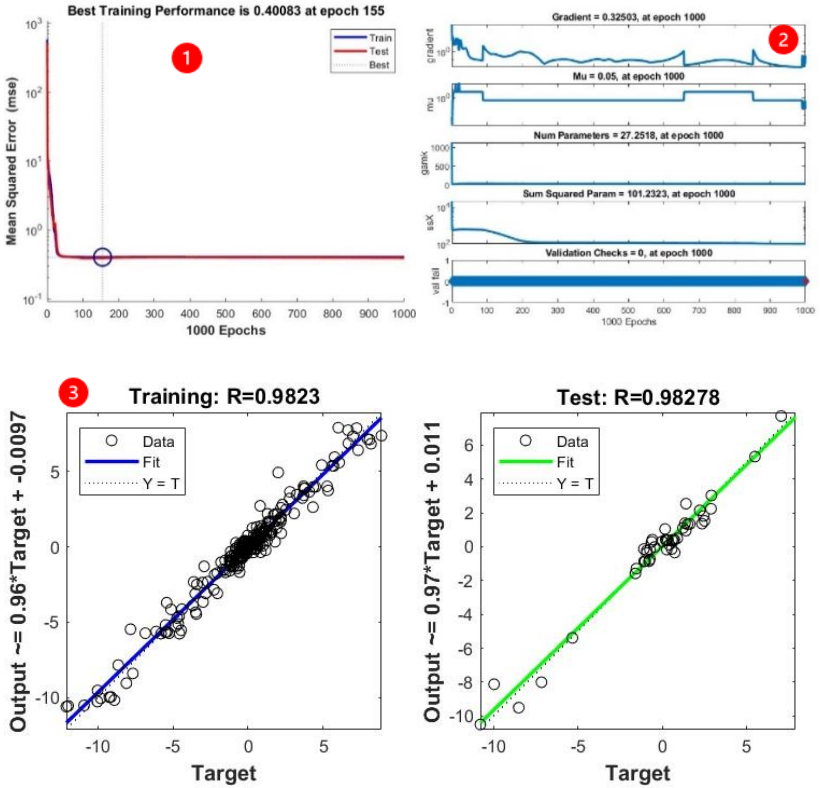


Fig. 4.9. Evaluarea performației RNA

Atât 1 cât și 2 au legatură directă cu procesul de învățare (antrenare) și modalul în care acesta reușește să minimizeze eroarea MSE evidențiată în 1.

Codul pentru dezvoltarea RNA

```
esant=300; %eșantion pentru datele generate aleator
a=-3;
b=5;
%generare valori aleatorii
x_aleator=a+(b-a).*rand(esant,1);
y_aleator=exp(0.5*x_aleator).*sin(5*x_aleator)+0.2*(abs(x_aleator)
                +1).*randn(esant,1);
%generare valori conform funcției
```

```
x_ideal=-3:0.02:5; %eșantion cu pas de 0.02 pentru funcția din Ecuația(1)
y_ideal=exp(0.5*x_ideal).*sin(5*x_ideal); % Ecuația(1)

%vizualizare grafică a valorilor
figure; hold on;
plot1=plot(x_ideal, y_ideal, 'LineWidth',2);
plot2=plot(x_aleator, y_aleator,'x');
M1 = "Functie ideala";
M2 = "Functie generata aleator";
legend([plot1,plot2], [M1, M2]);
legend('Orientation','horizontal')
legend('boxoff')
legend('Location','northwest')

%configurarea rețelei neuronale
netconf=[20 20 30]; %cRNA cu 3 straturi ascunse și 20, 20, 30 neuroni
net=feedforwardnet(netconf);%alegerea RNA feedforward
net=trainbr(net,x_aleator.',y_aleator.');%alocarea datelor de intrare și ieșire

%testarea rețelei neuronale
y_estim=net(x_ideal);

%vizualizare grafică a valorilor
figure; hold on;
plot3=plot(x_aleator, y_aleator, 'x');
plot4=plot(x_ideal, y_estim, 'LineWidth',0.5);
M3 = "Functie generata aleator";
M4 = "Functie estimata de RNA";
legend([plot3,plot4], [M3, M4]);
legend('Orientation','horizontal')
legend('boxoff')
legend('Location','northwest')
```

4.4.2. Testarea rețelei neuronale

Odată finalizată scrierea codului si debuging-ul potențialelor erori, se va testa pentru datele de la subcapitolul precedent modificarea numărului de straturi ascunse, numărului de neuroni și tipul de algoritm folosit pentru antrenare. Rezultatele se completează în tabelul 4.2.

Tabelul 4.2. Rezultatele testării controlerului fuzzy

Nr.	Algoritm de antrenare	Straturi ascunse	Număr de neuroni	Valoarea R	Timp
1	**trainlm**	1	10		
2		2	50 | 50		
3		3	100 | 100 | 100		
4		4	100 | 100 | 100 | 100		
5	**trainbr**	1	10		
6		2	50 | 50		
7		3	100 | 100 | 100		
8		4	100 | 100 | 100 | 100		
9	**traingdx**	1	10		
10		2	50 | 50		
11		3	100 | 100 | 100		
12		4	100 | 100 | 100 | 100		
13	**traingdm**	1	10		
14		2	50 | 50		
15		3	100 | 100 | 100		
16		4	100 | 100 | 100 | 100		
13	**traingd**	1	10		
14		2	50 | 50		
15		3	100 | 100 | 100		
16		4	100 | 100 | 100 | 100		

4.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultatele obținute vor fi comparate între ele și cu datele de referință, apoi se va ridica un grafic care să evidențieze diferența dintre datele obținute și cele reale.

4.6. Concluzii

Se vor formula concluzii cu privire la

- Influența dimensiunii RNA asupra timpului de antrenare ;
- Influența dimensiunii RNA asupra performanței ;
- Impactul metodei de antrenare asupra preformanței RNA ;

……...

……...

……...

……...

……...

……...

……...

……...

.....

.....

5

PROGNOZA FOLOSIND RNA EXEMPLU NUMERIC

5.1. Scopul lucrării

Înțelegerea modului în care se proiectează și utilizează o rețea neuronală artificială (RNA) de tipul multistrat perceptron cu învățare supravegheată și propagarea înapoi a erorii (back-propagation).

Rolul rețelei neuronale artificiale este de a realiza prognoza pe termen scurt a consumului de energie electrică a unui consumator comercial.

5.2. Descrierea problemei

Pentru creșterea calității activității de management energetic a unui consumator de energie electrică comercial, o soluție este estimarea consumului de energie electrică pe ziua următoare, adică prognoza pe termen scurt.

În acest scop, se va folosi o rețea neuronală artificială de tipul multistrat Perceptron cu un singur strat ascuns. Neuroni artificiali ai RNA sunt modelați după modelul neuronului static, care este activat prin funcția sigmoid.

Antrenarea (învățarea) rețelei se realizează prin implicarea metodei de învățarea supravegheată și propagarea înapoi a erorii (error back-propagation).

5.3. Noțiuni teoretice. Rețeaua Neuronală Artificială

5.3.1. Arhitectura RNA

Rețeaua neuronală artificială este formată din trei straturi de neuroni, fig. 5.1. Primul strat, stratul de intrare cuprinde cinci neuroni de intrare, al doilea strat, stratul ascuns are trei neuroni, iar ultimul strat, stratul de ieșire are un singur neuron.

Stratul de intrare are rolul de a prelua datele de intrare și a le transmite mai departe, către al doilea strat, cel ascuns. Cei 5 neuroni de intrare, x_1....x_5 corespund factorilor care influențează consumul de energie electrică, și anume: temperatura mediului ambiant, umiditatea mediului ambiant, nivelul de iluminare naturală, tipul de zi și numărul de angajați.

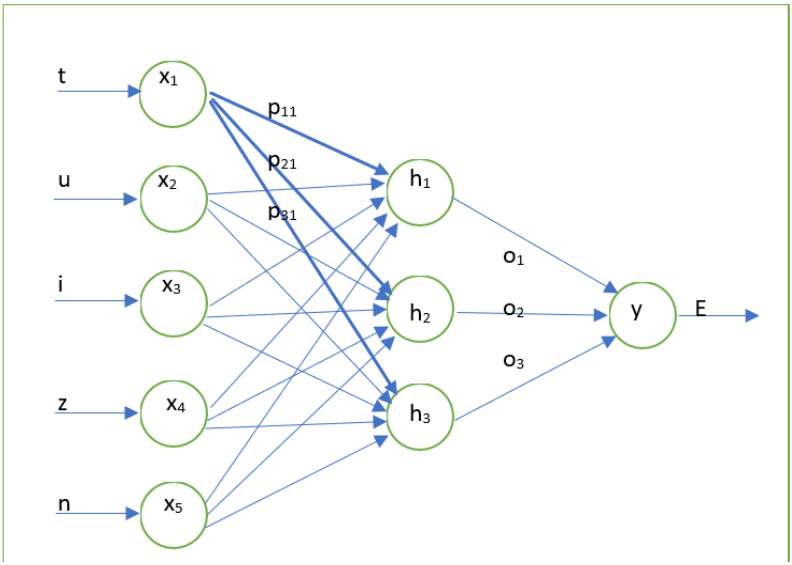


Fig. 5.1. Arhitectura RNA

Valoarea neuronilor de intrare se modifică în funcție de valorile datelor de intrare, ci nu prin calcule în cadru procesului de antrenare. Valorile datelor de intrare sunt preluate de la exterior, de cele mai

multe ori fiind scalate pentru a ușura calculele, altfel mărimile lor sunt neschimbate.

Temperatura mediului ambiant, t [°C] are intervalul de variație [-20, 35], este un factor care influențează consumul de energie electrică prin intermediul receptoarelor necesare pentru încălzire în timpul iernii și răcire în timpul verii, respectiv răcirea produselor perisabile pe tot parcursul anului.

Umiditatea mediului ambiant, u [%] variază în intervalul [40, 80]. Acest factor influențează consumul de energie electrică prin faptul că afectează procesul de răcire și încălzire.

Nivelul de iluminare naturală, i [lx] poate varia în intervalul [200, 1000], el afectând consumul de energie electrică a consumatorului prin intermediul aparatelor de iluminat.

Tipul zilei este un factor notat, cu n [1, 2, ..., 10], care a fost introdus deoarece studiile de specialitate au evidențiat faptul că forma curbei de sarcină a consumatorilor comerciali este diferită în funcție de ziua săptămânii (zi de lucru, zi de weekend) și de perioade speciale ale anului (concedii, sărbători).

Numărul de angajați, notat cu z, care ia valori numere naturale între 1 și 10, este un factor care influențează consumul de energiei electrică a consumatorului, întrucât numărul de angajați prezenți infuențează numărul de receptoare care sunt folosite.

Stratul ascuns este compus din trei neuroni, notați cu h_1, h_2 și h_3. Rolul acestor neuroni este de a înmagazina informațiile intermediare în etapele de antrenare, testare și folosire a RNA. Diferit de neuronii din stratul de intrare, valorile neuronilor de stratul ascuns se modifică permanent.

Stratul de ieșire a RNA conține un singur neuron, y, care corespunde consumului de energie electrică (în valori scalate), E [kWh] și va indica valoarea acesteia în intervalul [2, 10].

Observație

Pentru a dezvolta o aplicație RNA este necesară o bază consistentă de date. Aceste date vor fi folosite pentru a identifica factorii care influențează prognoza, intervalele de variație a acestora, precum și a ieșirii, dar în primul rând pentru a antrena și testa rețeaua.

Datele care sunt prezentate în acest exemplu sunt în scop didactic, deci nu caracterizează în mod fidel consumul de energie electrică a unui consumator comercial.

Precum se observă în fig. 5.1, straturile rețelei au neuroni total interconectați, astfel toți neuroni din stratul de intrare sunt conectați cu toți neuronii din stratul ascuns, iar aceștia din urmă sunt conectați cu neuronul de ieșire. Deci, rețeaua este total interconectată prin intermediul unor săgeți unidirecționale, denumite p_{ij} (i este indicele neuronului sursă și j este indicele neuronului receptor), respectiv o_1, o_2 și o_3. Săgețile indică faptul că informația va pleca de la neuroni din straturile inferioare, înspre neuroni din straturile superioare. Aceste conexiuni dintre neuroni se numesc ponderile neuronilor (weights) și indică modul în care o informație care trece de la un neuron dintr-un strat inferior înspre unul superior are importanță (influențează) în determinarea valorii finale a neuronului receptor. Ponderile au valori pozitive subunitare, adică aparțin intervalului [0, 1].

Datele de intrare și ieșire, precum și ponderile sunt reprezentate sub forma de matrici uni sau bidimensionale. Relațiile (5.1) – (5.5) descriu forma de prezentare a neuronilor din cele trei straturi și a ponderilor dintre neuroni când RNA este implementată matematic și informatic.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \tag{5.1}$$

$$H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \tag{5.2}$$

$$Y = [y] \tag{5.3}$$

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \\ p_{51} & p_{52} & p_{55} \end{bmatrix} \quad (5.4)$$

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix} \quad (5.5)$$

5.3.2. Datele de antrenare și de testare

Datele de antrenare și testare a RNA sunt cei mai importanți factori care influențează eficiența RNA. Astfel reiese atenția care trebuie acordată acestora.

În acest exemplu au fost considerate cinci seturi de date, trei pentru antrenare și două pentru testare. Un set conține valorile datelor de intrare și ieșirea corespunzătoare. Tabelul 5.1 prezintă seturile de antrenament și testare.

Tabelul 5.1. Datele de antrenament și testare ale RNA

Set	**t**	**u**	**i**	**z**	**n**	**E**
1	20	60	1000	1	5	6
2	-20	40	200	1	5	8
3	35	80	500	2	5	10
4	10	40	800	10	10	5
5	5	70	500	7	1	2

Din aceste seturi de antrenament se vor folosi setul 1, 3 și 5 pentru antrenarea rețelei și setul 2 și 4 pentru testarea ei.

Având în vedere că rețeaua lucrează cu numere subunitare în modul, datele trebuie scalate pentru a corespunde acestei cerințe. Metoda cea mai folosită în literatura de scpecialitate este scalarea după valoarea maximă a unui parametru. Însă, în acest exemplu se va

folosi scalarea cu multimpli de 10. Astfel, valorile lui t și u vor fi împărțite cu 100, ale lui z, n și E cu 10, iar ale lui i cu 1000. În consecință, noul set de date care va fi folosit mai departe de rețea este cel din tabelul 5.2.

Tabelul 5.2. Datele de antrenament și testare ale RNA scalate

Set	**t**	**u**	**i**	**z**	**n**	**E**
1	0,2	0,6	1	0,1	0,5	0,6
2	-0,2	0,4	0,2	0,1	0,5	0,8
3	0,35	0,8	0,5	0,2	0,5	1
4	0,1	0,4	0,8	1	1	0,5
5	0,05	0,7	0,5	0,7	0,1	0,2

Seturile de antrenament și testare scriese sub forma unor matrici, care sunt notate cu A (setul de antrenament) și T (setul de testare) au valorile din relațiile (5.6) și (5.7).

$$A = \begin{bmatrix} 0,2 & 0,6 & 1 & 0,1 & 0,5 & 0,6 \\ 0,35 & 0,8 & 0,5 & 0,2 & 0,5 & 1 \\ 0,05 & 0,7 & 0,5 & 0,7 & 0,1 & 0,2 \end{bmatrix} \tag{5.6}$$

$$T = \begin{bmatrix} -0,2 & 0,4 & 0,2 & 0,1 & 0,5 & 0,8 \\ 0,1 & 0,4 & 0,8 & 1 & 1 & 0,5 \end{bmatrix} \tag{5.7}$$

5.3.3. Antrenarea RNA

Antrenarea supravegheată a RNA este procesul prin care rețeaua ”învață” să asocieze datele de intrare la ieșirea corespunzătoare. Această învățare se realizează prin modificarea și adaptarea continuă în timpul antrenării a ponderilor dintre neuroni.

În prima etapă a antrenării sunt date valori aleatorii ponderilor în intervalul 0, 1. Aceste valori apoi prin relațiile matermatice corespunzătoare (propagarea înapoi a erorii) se vor modifica, într-un

final, la sfârșitul procesului de antrenare vor avea niște valori care rămân fixe în procesul de testare și urilizare ulterioară a RNA.

Valorile ponderilor P și O în proma fază a antrenării sunt cele din relațiile (5.8) și (5.9).

$$P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix} \tag{5.8}$$

$$O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix} \tag{5.9}$$

Relațiile matematice necesare pentru a determina activarea neuronilor, respectiv porpagarea înapoi a erorii sunt (5.10) – (5.18).

Relația (5.10) se folosește pentru a determina valoarea corespunzătoare corpului unui neuron static, iar relația (5.12) pentru a calcula activarea lui, adică valoarea care se transmite mai departe, la neuronii din stratul superior.

$$net_{h_i} = \sum x_j \cdot p_{ji} \tag{5.10}$$

unde i = 1, 2, 3, iar j = 1,.., 5
iar pentru determinarea valorii lui h_1

$$net_{h_1} = x_1 \cdot p_{11} + x_2 \cdot p_{21} + x_3 \cdot p_{31} + x_4 \cdot p_{41} + x_5 \cdot p_{51} \tag{5.11}$$

$$h_i = \frac{1}{1+e^{-net_{hi}}} \tag{5.12}$$

unde i = 1, 2, 3.

Valoarea corpului neuronului de ieșire se determină cu relațiile (5.13) și (5.14), iar valoarea de ieșire, notată cu yc, deoarece este obținută prin calcule se obține din relația (5.15).

$$net_{yc} = \sum h_i \cdot o_i \tag{5.13}$$

unde i = 1, 2, 3,
iar forma desfășurată este

$$net_{yc} = h_1 \cdot o_1 + h_2 \cdot o_2 + h_3 \cdot o_3 \tag{5.14}$$

$$y_c = \frac{1}{1+e^{-net_{yc}}} \tag{5.15}$$

Prin relațiile prezentate anterior, adică (5.10) – (5.15), se realizează transmiterea informațiilor de la stratul de intrare înspre stratul de ieșire. În continuare, ieșirea calculată este comparată cu ieșirea din setul de antrenament, iar diferența obținută (dacă este diferită de zero, sau o limită foarte mică) va fi considerată eroarea, care se propagă înapoi pentru a modifica valorile ponderilor. După adaptarea ponderilor, acestea vor corespunde setului de antrenament folosit.

$$err = y_D - y_c \tag{5.16}$$

$$o_i^* = o_i + \Delta o_i \tag{5.17}$$

unde noua valoare adaptată a ponderii este o_i^*, α este rata de învățare, care ia valori în intervalul [0, 1], $\Delta o_i = \alpha \cdot h_i \cdot \delta o_i$, $\delta o_i = h_i(1 - h_i) \cdot err$. Δo_i este valoarea cu care se modifică ponderea, iar δo_i este ponderea din eroare care revine lui o_i, iar *err* este eroarea mărimii calculate față de cea dorită, care se regăzește în setul de antrenament, y_D – ieșirea dorită, y_c – ieșirea calculată.

$$p_{ij}^* = p_{ij} + \Delta p_{ij} \tag{5.18}$$

unde noua valoare adaptată a ponderii este p_{ij}^*, α este rata de învățare, care ia valori în intervalul [0, 1],

$\Delta p_{ij} = \alpha \cdot x_j \cdot \delta p_{ij}$, $\delta p_{ij} = x_j(1 - x_j) \cdot \delta o_i \cdot o_i$; Δp_{ij} este valoarea cu care se modifică ponderea, iar δp_{ij} este ponderea din eroare care revine lui p_{ij}.

Relațiile (5.16) - (5.18) arată raționamentul matematic prin care se propagă înapoi eroarea dintre ieșirea calculată și cea dorită. Cu ajutorul relației (5.17) se determină noile valori adaptate ale ponderilor care leagă neuronii din stratul ascuns de cel de ieșire. Relația (5.18) arată modul în care se modifică ponderile dintre neuronii din stratul de intrare și neuronii din stratul ascuns. Se poate observa că noua valoare este influențată de modificările realizate la ponderile din etapa precedentă.

Antrenarea începe prin parcurgerea seturilor de antrenament o dată, și adaptarea ponderilor, astfel se realizează o epocă. În antrenarea unei RNA este nevoie de mai multe epoci. Aici se va folosi dor o epocă.

Setul 1

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0,2 \\ 0,6 \\ 1 \\ 0,1 \\ 0,5 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0,6],$$

$$P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix}, O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0,31 \\ 0,55 \\ 0,79 \end{bmatrix}, H = \begin{bmatrix} 0,57 \\ 0,63 \\ 0,68 \end{bmatrix}, Y_D = [0,6], y_D = 0,6,$$

$net_{yc} = 0{,}38, y_c = 0{,}595 \cong 0{,}6$

Întrucât $err = 0$, nu este nevoie ca ponderile să fie adaptate.

Setul 2

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0{,}35 \\ 0{,}8 \\ 0{,}5 \\ 0{,}2 \\ 0{,}5 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [1],$$

$$P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix}, O = \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0{,}33 \\ 0{,}57 \\ 0{,}8 \end{bmatrix}, H = \begin{bmatrix} 0{,}58 \\ 0{,}63 \\ 0{,}69 \end{bmatrix},$$

$Y_D = [1], y_D = 1, net_{yc} = 0{,}39, y_c = 0{,}6$

Întrucât $err = 0{,}4$ este nevoie ca ponderile să fie adaptate. Astfel, după realizarea calculelor și propagarea înapoi a erorii, noile ponderi au valorile enumerate în continuare.

Se poate observa din noilor valori ale ponderilor faptul că, ponderile dintre stratul de ieșire și cel ascuns (care sunt mai aproape de eroare) au fost mai puternic afectate, față de ponderile dintre stratul de intrare și cel ascuns.

$$O^* = \begin{bmatrix} 0{,}1056 \\ 0{,}2058 \\ 0{,}3059 \end{bmatrix} \cong \begin{bmatrix} 0{,}11 \\ 0{,}21 \\ 0{,}31 \end{bmatrix},$$

$$P^* = \begin{bmatrix} 0{,}1005 & 0{,}2005 & 0{,}3005 \\ 0{,}2005 & 0{,}3005 & 0{,}4005 \\ 0{,}1005 & 0{,}2005 & 0{,}3005 \\ 0{,}2005 & 0{,}3005 & 0{,}4005 \\ 0{,}1005 & 0{,}2005 & 0{,}3005 \end{bmatrix}$$

$$\cong \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix},$$

Setul 3

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0{,}05 \\ 0{,}7 \\ 0{,}5 \\ 0{,}7 \\ 0{,}1 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0{,}2],$$

$$P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix}, O = \begin{bmatrix} 0{,}11 \\ 0{,}21 \\ 0{,}31 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0{,}34 \\ 0{,}55 \\ 0{,}75 \end{bmatrix}, H = \begin{bmatrix} 0{,}58 \\ 0{,}63 \\ 0{,}68 \end{bmatrix}, \quad Y_D = [0{,}2], \quad y_D = 0{,}2,$$
$net_{yc} = 0{,}4, y_c = 0{,}6$

Întrucât $err = -0{,}4$ este nevoie ca ponderile să fie adaptate. Astfel, după realizarea calculelor și propagarea înapoi a erorii, noile ponderi au valorile enumerate în continuare.

$$O^* = \begin{bmatrix} 0{,}1043 \\ 0{,}2041 \\ 0{,}3040 \end{bmatrix} \cong \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix}, \quad P^* = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix},$$

Se poate observa din noilor valori ale ponderilor faptul că, ponderile O s-au modificat foarte puțin, iar ponderile P foarte puțin. Acest proces de adaptare a ponderilor continuă până la îndeplinirea condiției de oprire. În cazul de față condiția de oprire este parcurgerea unei epoci, dar în mod uzual, se efectuează peste 50 de epoci pentru ca antrenarea să dea rezultate bune.

După finalizarea antrenării RNA, ponderile rețelei vor fi adaptate pentru a corespunde tuturor datelor de intrare, deci rețeaua va fi capabilă să asocieze datele de intrare la ieșirea corespunzătoare. Aceste ponderi sunt utilizate ulterior în etapa de testare a rețelei.

5.3.4. Testarea RNA

În etapa de testare, rețeaua are următoarea formă

$$X = \begin{bmatrix} 0{,}05 \\ 0{,}7 \\ 0{,}5 \\ 0{,}7 \\ 0{,}1 \end{bmatrix} \rightarrow P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix} \rightarrow H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \rightarrow$$

$$O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix} \rightarrow Y = [y]$$

Seturile de testare ale rețelei cu ajutorul cărora se va determina eficacitatea rețelei sunt descrise în relația (5.7).

$$T = \begin{bmatrix} -0,2 & 0,4 & 0,2 & 0,1 & 0,5 & 0,8 \\ 0,1 & 0,4 & 0,8 & 1 & 1 & 0,5 \end{bmatrix}$$

Aceste valori de testare for fi folosite pentru a transmite informația dinspre stratul de start, înspre stratul de ieșire. Rezultatul obținut este comparat cu datele de ieșire din seturile de testare. Dacă rezultatele au o deviație mai mică de 1% (datele calculate au o deviație față de cele dorite) în peste 90% din cazurile (seturi) testate.

Setul 1

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} -0,2 \\ 0,4 \\ 0,2 \\ 0,1 \\ 0,5 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0,8],$$

$$P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix}, O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0,15 \\ 0,25 \\ 0,35 \end{bmatrix}, H = \begin{bmatrix} 0,53 \\ 0,56 \\ 0,59 \end{bmatrix}, Y_D = [0,8], y_D = 0,8, y_c = 0,58$$

Deviația ieșirii calculate față de cea dorită este de *dev* = 0,22, valoare care depășește procentul de 1%.

Setul 2

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0,1 \\ 0,4 \\ 0,8 \\ 1 \\ 1 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0,4],$$

$$P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix}, O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0,47 \\ 0,8 \\ 1,13 \end{bmatrix}, H = \begin{bmatrix} 0,61 \\ 0,69 \\ 0,76 \end{bmatrix}, Y_D = [0,4], y_D = 0,4, y_c = 0,63$$

Deviația ieșirii calculate față de cea dorită este de *dev* = -0,22, valoare care depășește procentul de 1%.

Din rezultatele obținute se poate observa că RNA nu a fost antrenată corespunzător, astfel nu a reușit să treacă etapa de testare. În această situație, RNA va fi antrenată din nou pe mai multe epoci, apoi testată din nou, procesul se va repeta până când datele obținute în urma testării corespund cerințelor.

5.4. Mersul lucrării

În cadrul acestei lucrări de laborator se vor realiza următoarele activități:

5. Se va îmbogăți baza de antrenament prin adăgarea altor 7 seturi de antrenament cu valori variate. La final va fi o bază de antrenament cu 10 seturi.
6. Se vor relua pașii de la 5.3.3 prin care se realizează antrenarea rețelei, iar pentru ponderi se vor lua în considerare valori cu 4 zecimale (ex. 0,0078).
7. Valorile ponderilor P și O vor avea valoarea 0,1.
8. Se vor parcurge două epoci.
9. Rezultatele obținute pentru ponderi se vor trece în tabelul 5.3.

Tabelul 5.3. Evoluția valorile ponderilor în procesul de antrenare

Epoca 1		
Setul	**Ponderile P**	**Ponderile O**
1		
2		

3		
4		
5		
6		
7		

8		
9		
10		
Epoca 2		
Setul	**Ponderile P**	**Ponderile O**
1		

2		
3		
4		
5		
6		

7		
8		
9		
10		

5.5. Prelucrarea datelor și interpretarea rezultatelor

Datele din tabelul 5.3. vor fi folosite pentru a testa rețeaua dezvoltată. Mai exact, cu valorile ajustate ale ponderilor, adică ultima linie a tabelului 5.3. se vor realiza operațiile din capitolul 5.3.4.

5.6. Concluzii

Se vor formula concluzii asupra eficienței procesului de antrenare a rețelei, respectiv asupra prognozei realizate.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6

PROGNOZA FOLOSIND CALCULUL NEURONAL

6.1. Scopul lucrării

Prognoza consumului de energie electrică oferă informații utile participanților la piața de energie electrică cât și operatorilor de sistem:

- echilibrarea balanței producție-consum și transferul optim de electricitate între rețelele regionale;
- achiziția energiei electrice
- reducerea consumului propriu tehnologic în rețele,
- identificarea rapidă a defectelor și mentenanță preventivă.

Pentru a răspunde la această provocare în această lucrarea se va implementa un algoritm RNA pentru prognoza curbelor de sarcină pentru următoarelor 24 de ore.

6.2. Obiectivele lucrării

Principalul obiectiv al lucrării este prezentarea noțiunilor teoretice privind realizarea prognozelorde consum folosind instrumentul dedicată a mediului de modelare și simulare MATLAB. Deasemenea se vor prezenta pașii necesari în dezvoltarea unei rețele enuronale și

se va testa operarea modelului în funcție de diferte variabile și parametrii de antrenare a rețelei.

6.3. Noţiuni teoretice. Neural Network Time Series

Pentru realizarea prognozei cu algoritmi bazați pe rețele neuronale trebuie urmate mai multe etape:

Pasul 1: Definirea problemei

Definirea cu atenție a problemei necesită o înțelegere a modului în care vor fi folosite prognozele și beneficiarii prognozelor.

Pasul 2: Colectarea informațiilor și prelucrarea datelor

Pentru realizarea prognozelor sunt necesare două tipuri de informații: (a) date statistice și (b) expertiza acumulată a specialiștilor. Construirea unui model statistic consistent implică date istorice corecte și suficiente.

Pasul 3: Analiza preliminară (exploratorie)

Acest pas presupune înțelegerea curbelor de sarcină prin reprezentarea grafică comparativă a datelor orare. Stabilirea variabilelor exogene, tiparele de consum, identificarea trend-ului sau a sezonalității sunt necesare.

Pasul 4: Alegerea și implementarea modelelor

Alegerea celui mai bun model se realizează după compararea rezultatelor de la mai multe metode matematice, de la cele mai simple până la cele consacrate în literatură. Fiecare model reprezintă un construct artificial care se bazează pe un set de ipoteze (explicite și implicite) și implică de obicei unul sau mai mulți parametri care trebuie evaluați folosind datele istorice cunoscute.

Pasul 5: Utilizarea și evaluarea unui model de prognoză.

Există multe probleme și obstacole în utilizarea și implementarea prognozelor în practică, cel mai mare obstacol fiind expunerea la risc. Metodele de evaluare se bazează pe calculul indicatorilor statistici pentru evidențierea erorilor.

6.3.1. Etapele dezvoltării unei aplicații RNA

Pași necesari care trebuie parcurși pentru a realiza o aplicație folosind NTSTOOL sunt enumerați în continuare.

a. Pregătirea și importul setului de date

Conform pasului 2 se analizează istoricul de consum al unei companii care își desfășoară activitatea în industria prelucrătoare a lemnului. În această lucrare se lucrează cu un istoric orar de consum pe 3 ani de zile și este prezentată în fig. 6.1 o săptămână tipică.

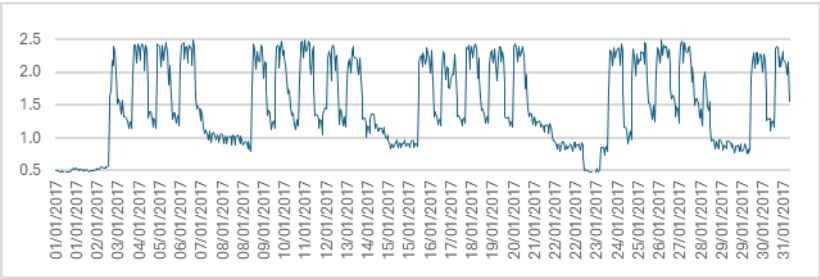


Fig. 6.1. Curbă de sarcină folosită în prognoză – luna ianuarie

Din motive de spațiu, în tabelul 6.1 se prezintă sumar antetul de tabel folosit în analize, care reprezintă un fișier Excel. Acest fișier trebuie pregătit înainte de a fi inserat în Matlab.

Tabel 6.1. Exemplificare date luate în calcul pentru prognoză.

Data/oră	Consum [MWh]	Temp [°C]
01/01/2017 - 1	0.502	4.500
01/01/2017 - 2	0.499	4.500
01/01/2017 - 3	0.500	4.600
01/01/2017 - 4	0.491	4.500
...	...	...
31/12/2019 -23	0.469	4.800
31/12/2019 - 24	0.408	-2.500

Acest fișier excel trebuie împărțit în două fișiere care trebuie inserate în Matlab, respetiv un fișier cu date de intrare și date de ieșire.

date_intrare.xls
date_ieșire.xls

În fig. 6.2 se prezintă împărțirea datelor în set de intrare și ieșire, deci practic vom asigura rețelei date cunoscute atât la intrare cât și la ieșire.

	DATE DE IEȘIRE	DATE DE INTRARE							
data/oră	Consum [MWh]	Temp [°C]	T-1	T-2	T-3	T-4	T-5	T-6	T-7
08/01/2017 - 1	1.054	-2.1	1.47	1.369	1.394	1.304	0.539	0.506	0.502
08/01/2017 - 2	1.039	-2	1.43	1.312	1.294	1.255	0.548	0.51	0.499
08/01/2017 - 3	0.905	-2.3	1.26	1.226	1.166	1.146	0.548	0.52	0.5
08/01/2017 - 4	1.028	-2	1.429	1.347	1.305	1.246	0.549	0.509	0.491
08/01/2017 - 5	1.005	-1.9	1.367	1.303	1.272	1.236	0.549	0.51	0.487
08/01/2017 - 6	0.916	-1.8	1.155	1.186	1.141	1.142	0.586	0.506	0.486
08/01/2017 - 7	1.015	-1.4	1.07	2.134	2.045	1.944	1.418	0.507	0.494
08/01/2017 - 8	1.032	-1	1.129	2.408	2.423	2.415	1.623	0.513	0.488
08/01/2017 - 9	0.999	-1	1.097	2.44	2.41	2.422	1.713	0.497	0.479
08/01/2017 - 10	1.021	-0.5	1.06	2.453	2.399	2.408	2.181	0.497	0.469
08/01/2017 - 11	0.887	0.6	0.965	2.193	2.081	2.135	2.094	0.488	0.476
08/01/2017 - 12	1.055	2.1	1.114	2.369	2.393	2.425	2.391	0.501	0.475

Fig. 6.2. Împărțirea setului de date

Din momentul încărcării fișierelor în Matlab, datele vor fi prelucrare sub formă de matrici. Fișierele trebuie inițial încărcate în *Workspace.* Se selectează folder-ul în care sunt localizate fișierele și apoi prin *drag-and-drop* către Workspace sau *click dreapta > import data* (1) se va deschide următoarea fereastră (fig. 6.3):

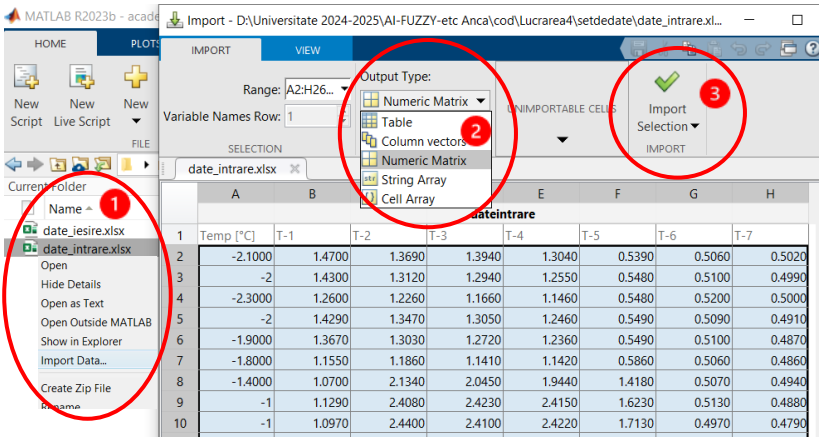


Fig. 6.3. Inserarea datelor în Workspace

Selectăm la Output Type > Numeric Matrix (2), apoi apăsăm pe butonul Import Selection (3). În acest moment în workspace vor fi inserate cele doua fișiere sub formă de matrice numerică double cu 26112 de valori după cum este prezentat în fig. 6.4.

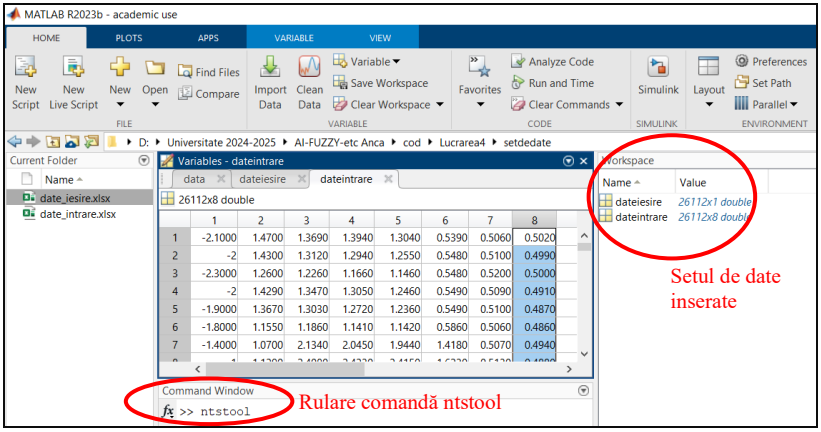


Fig. 6.4. Inserare a datelor și rularea ntstool

b. Configurarea RNA

În acest moment datele sunt pregătite și folosind comanda *ntstool (conform fig.4)* accesăm aplicația dedicată implementării unei rețele

neuronale. Această unealtă este Neural Network Time Series (fig. 6.5. – MATLAB R2023a), care se accesează prin scrierea în linia de comandă ntstool.

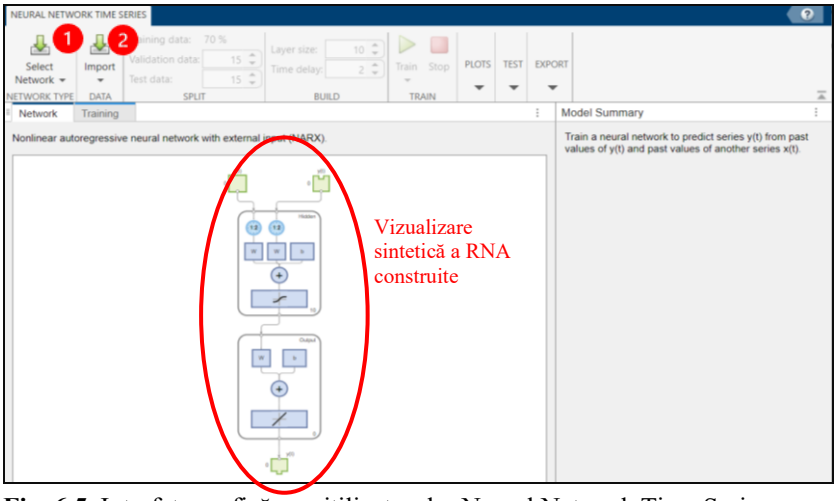


Fig. 6.5. Interfața grafică cu uitilizatorul – Neural Network Time Series

Instrumentul Neural Network for Time Series oferă posibilitatea de a dezvolta o rețea neuronală cu multiple straturi și parametrii, iar în fig. 6.5 ❶ se prezintă alegerea unui tip de rețea care ține cont de istoricul trecut și variabile independente:

Select Network > NARX Network

Pentru importul datelor se folosește butonul Import Data fig. 6.5 și se urmează pașii prezentați în fig. 6.6:

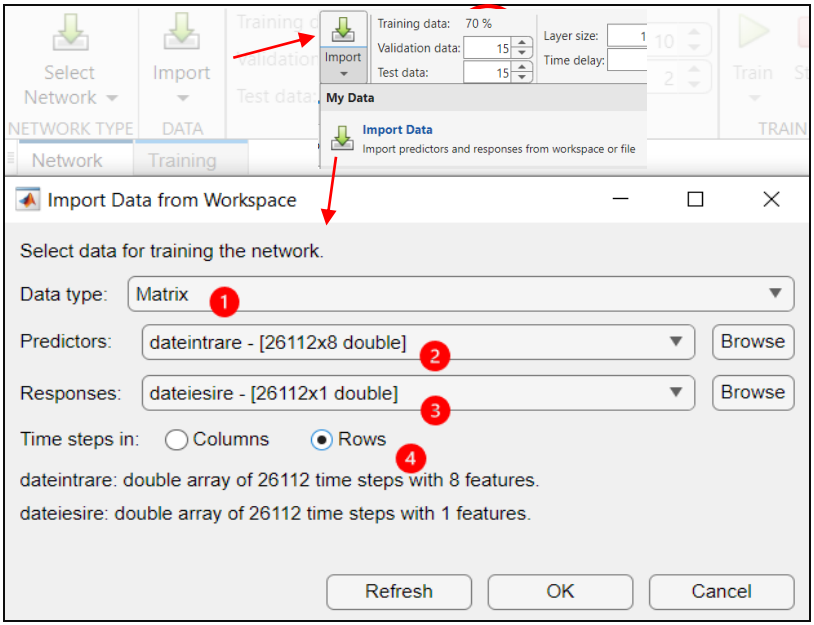


Fig. 4.6. Import date

1 Se selectează din dropdown list > Matrix

2 Predictors sunt variabile de intrare pe baza cărora se realizează prognoza (temperatură și consum istoric T-7)

3 Responses sunt variabila de ieșire, adică consumul

4 Se selectează orientarea valorilor pe linii (rows)

În fig. 6.7 se prezintă setările necesare pentru configurarea RNA, aceste setări trebuie introduse manual:

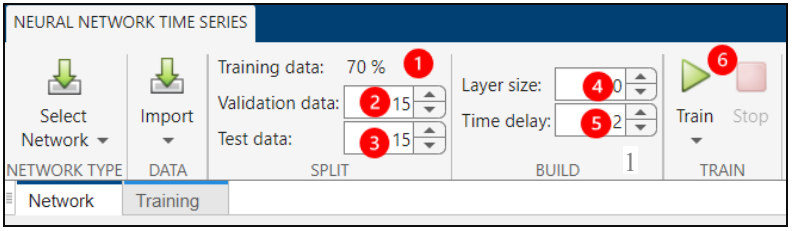


Fig. 6.7. Setare parametrii RNA

1 Se împarte tot setul de date în valori de antrenare, validare și testare. Se modifică automat în funcție de 2) și 3)

2 Setul de validare – se validează antrenare pe setul de antrenare

3 Setul de testare – aceste date nu sunt folosite deloc la antrenare

4 Numărul de straturi ascunse

5 Buton train

6 Time delay se referă la folosirea unei întârzieri între datele de intrare și cele de ieșire (ex: Td=2, RNA va fi antrenată să prognozeze a doua valoare, în cazul nostru se seteaza pe 1)

c. Antrenarea, testarea și validarea RNA

După rularea codului prin click pe Train se deschide fereastra prezentată în fig. 6.8. Numărul de epoci reprezintă iterațiile în care se realizează calculul eroriivreţelei și incrementare cu rata de învățare a ponderilor si bias-ului

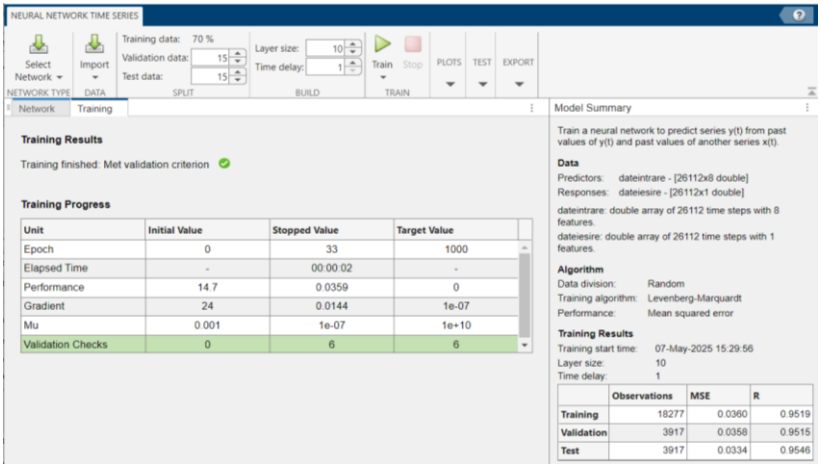


Fig. 6.8. Rezultate antrenare RNA pentru fiecare set de date

Prin accesarea meniului aferent *Ntstool* disponibil după antrenarea RNA se pot evalua performanțele în termeni de acuratețe conform fig. 6.9.

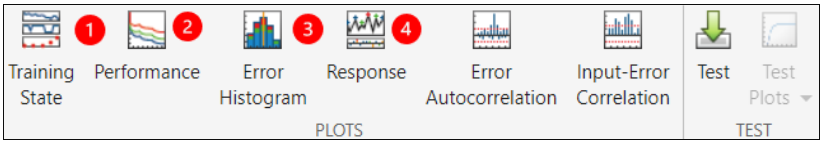


Fig. 6.9. Rezultate antrenare RNA

1 Training State – evidențiază acuratețea procesului de antrenare pe setul de antrenare, în fig. 6.10 sunt prezentați indicatorii care evaluează procesul de minimizare al erorii:

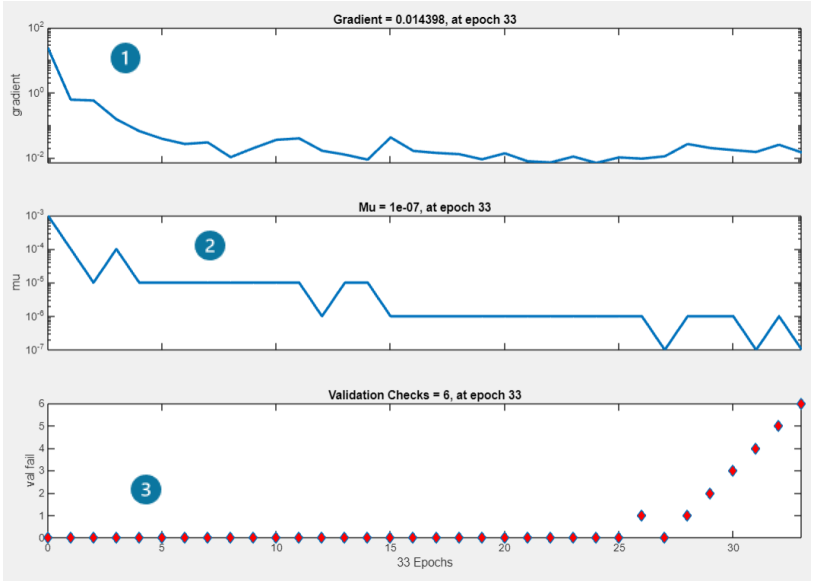


Fig. 6.10. Rezultate procesului de antrenare RNA în funcție de epoci

1 Graficul gradientului

O scădere constantă a magnitudinii gradientului este un semn bun de convergență. Gradienții mari persistenți sau oscilațiile pot necesita ajustări ale ratei de învățare sau reducerea gradientului pentru a stabiliza formarea.

2 Graficul mu (μ)

Oscilațiile frecvente ale lui „μ” ar putea sugera că optimizarea nu reușește să găsească o soluție stabilă, posibil din cauza unei suprafețe complexe al pierderilor. Un „μ” constant ridicat ar putea indica faptul că modelul are probleme de convergență și poate necesita ajustări, cum ar fi o inițializare sau o rată de învățare diferită.

3 Graficul verificărilor de validare

Scăderea valorilor funcției cost la validare: indică faptul că modelul se generalizează bine la datele nevăzute. Creșterea valorilor funcției cost la validare: ar putea sugera o supraadaptare (overfitting), atunci când modelul funcționează bine pe datele de antrenare, dar slab pe datele de validare. Plafonarea parametrilor de validare indica faptul că modelul și-a atins capacitatea cu arhitectura și datele actuale.

2 Performance – se evidențiază valorile funcției cost în funcție de numărul de epoci folosite în antrenare (fig. 6.11.)

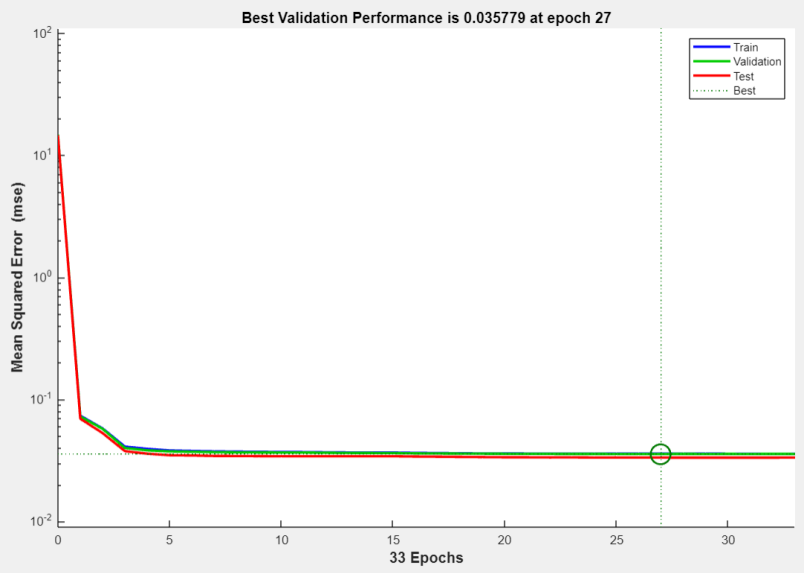


Fig. 6.11 Evoluția valorilor funcției cost

Formula de calcul pentru eroarea RNA este MSE (mean squared error sau eroarea medie pătratică și este definită prin (6.1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{6.1}$$

3 Error Histogram - Histograma erorilor este histograma erorilor dintre valorile țintă și valorile previzionate după antrenarea unei rețele neuronale feedforward, prezentată în fig. 6.12. Deoarece aceste valori ale erorii indică modul în care valorile previzionate diferă de valorile țintă.

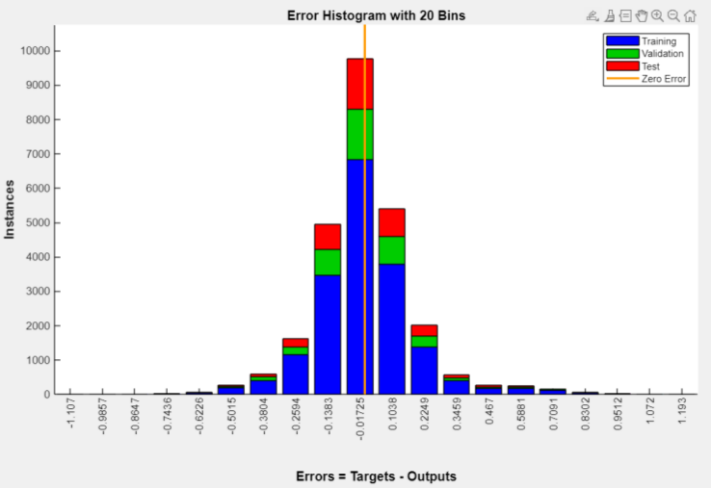


Fig. 6.12 Distribuția erorilor în funcție de setul de date

4 Response – accesarea acestui buton afișează rezultatele, răspunsurile (țintele) și erorile în funcție de timp. De asemenea, indică punctele de timp care au fost selectate pentru formare, testare și validare. Pentru vizualizarea grafică precisă a valorilor prognozate de RNA se va folosi zoom-ul în GUI aferent, în fig. 6.13 este o captură doar pentru evidențierea opțiunii.

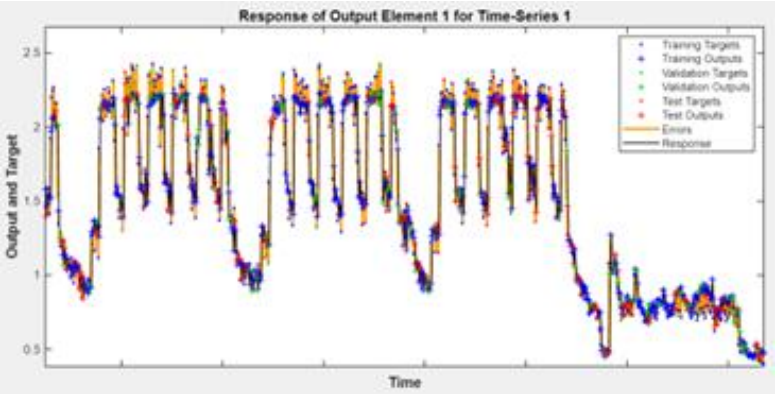


Fig. 6.13 Evaluarea grafică a prognozei pe fiecare set de date

6.4. Mersul lucrării

În cadrul acestei lucrări se va implementa și studia funcționalitatea unei rețele neuronale artificiale cu ajutorul Neural Network Time Series pentru prognoza consumului de energie electrică. Pentru a realiza acest obiectiv se vor realiza activitățile din următoarele subcapitole.

6.4.1. Rețea neuronală artificială

Pentru dezvoltarea aplicației RNA pentru prognoza consumului de energie electrică se vor urma pașii descriși în capitolul 3, aceștia fiind sumarizați aici:

- Inserare fișiere excel în workspace - date_intrare.xls și date_ieșire.xls
- Deschidere instrumentului neural network time series *(command window > ntstool)*
- Import date în ntstool – date de intrare (*predictors*) și date de ieșire (*responses*)
- Selectare procente pentru setul de date de antrenare, validare și testare
- Setare parametrii RNA – dimensiune strat ascuns (*layer size*)
- Click pe butonul *Train*

• Analiză rezultate

6.4.2. Evaluarea performanțelor RNA

Rularea pașilor prezentați în capitolul 3 secțiunea **c**, va determina atât antrenarea cât și validarea și testarea datelor. După acest pas se vor analiza opțiunile detaliate în Figura 9. Se vor testa pentru același set de date diferite tipuri de antrenare și se va modifica numărului de straturi ascunse, numărului de neuroni. Rezultatele se completează în tabelul 64.2.

Tabelul 6.2. Rezultatele testării controlerului fuzzy

Nr.	Algoritm de antrenare	Straturi ascunse	Număr de neuroni	Valoarea MSE		Timp
1	Levenberg-Marquardt	1	10	antrenare		
				validare		
				testare		
2	Bayesian Regularization	1	100	antrenare		
				validare		
				testare		
3	Scaled Conjugate Gradient	1	200	antrenare		
				validare		
				testare		

6.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultatele obținute vor fi comparate între ele și cu datele de referință, apoi se va ridica un grafic care să evidențieze diferența dintre datele obținute și cele reale.

6.6. Concluzii

Se vor formula concluzii cu privire la

- Influența dimensiunii RNA asupra timpului de antrenare ;
- Influența dimensiunii RNA asupra performanței ;
- Impactul metodei de antrenare asupra preformanței RNA .

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7

OPTIMIZAREA CU ALGORITMI GENETICI EXEMPLU NUMERIC

7.1. Scopul

Înțelegerea modului în care se proiectează, construiește și funcționează un algoritm genetic, AG.

Algoritmul genetic are rolul de a ajuta la creștea eficienței energetice a consumatorului industrial în procesul de management energetic și economic a acestuia.

7.2. Descrierea problemei

Un consumator industrial folosește în procesul tehnologic de producție 12 echipamente dedicate obținerii unor produse. Acestea se caracterizează prin atribute specifice care le clasifică în patru clase de calitate. Echipamentele electrice se împart în mai patru categorii în funcție de calitatea produselor care le produc, puterea activă și numărul de produse care îl produc zilnic.

Producția și funcționarea (echipamentele folosite) consumatorului industrial depinde de client și contractul cu acesta, deoarece, în contract se specifică numărul de produse și calitatea acestora. Pe

baza acestor doi parametrii (numărul și calitatea produselor), consumatorul va pune în funcțiune anumite echipamente.

În această situație, obiectivul consumatorului industrial este de a folosi combinația optimă de punere în funcțiune a echipamentelor electrice astfel încât să obțină produsele din contract cu costurile cele mai mici, adică consum minim de energie electrică.

În problemă ne intereseaază doar trei caracteristici ale echipamentelor electrice, și anume puterea activă, *P* (10 și 200 kW), calitatea produselor *q*, care poate lua valorile 1, 2, 3, 4 (1 – calitatea cea mai bună, și 4 – calitatea cea mai slabă), și numărul de produse obținute într-o zi, *n*, variază între 2 și 10 bucați/zi.

Tabelul 7.1 descrie cele 12 echipamente electrice ale consumatorului industrial cu cele trei atribute ale lor.

Tabelul 7.1. Echipamentele consumatorului insutrial.

Echipament	P [kW]	q	n [bucăți/zi]
1	10	1	2
2	80	1	10
3	200	1	20
4	20	2	5
5	100	2	30
6	150	2	50
7	50	3	10
8	90	3	45
9	180	3	80
10	70	4	20
11	12	4	60
12	200	4	100

Pentru înțelegerea mai ușoară a modului de operare a algoritmului genetic, acest exemplu se particularizează pentru un contract specific, care este descris în paragraful următor.

Un client prin intermediul unui contract cere următoarele produse: 50 bucăți porduse calitatea 1, 100 bucăți produse calitatea 2, 150 produse calitatea 3 și 200 produse calitatea 4. În plus, se specifică faptul că produsele de calitate inferioară pot fi înlocuite (în număr limitat) cu produse de calitate superioară.

Se va folosi un algoritm genetic pentru a determina soluția optimă de utilizare a echipamentelor pentru a obține produsele din contract în cel mai scurt timp și utilizând energia electrică în mod eficient. Mai mult, dacă va fi necesar, se va folosi acordul din contract privind înlocuirea produselor de calitate inferioară cu cele de calitate superioară.

7.3. Noțiuni teoretice. Algoritmul Genetic

Algoritmul genetic lucrează cu soluțiile problemei, adică funcționarea echipamentelor consumatorului industrial. Prima etapă în dezvoltarea algoritmului este codificarea soluțiilor, care se referă la modul în care se vor reprezenta acestea. Următorul pas este determinarea funcției obiectiv și de adaptare, apoi dezvoltarea primei populații, evaluarea indivizilor și construirea noilor generații.

7.3.1. Codificarea soluției

Codificarea soluției se va realiza prin folosirea valorilor {0, 1}, adică codificarea binară. Astfel, pentru fiecare echipament se va folosi una din cele două valori, care va indica starea de funcțiune a echipamentului corespunzător. Mai exact, 0 – echipamentul nu funcționează, iar 1 – echipamentul funcționează.

O soluție este reprezentată sub forma unui cromozom cu 12 gene, alele fiind 0 și 1, fig. 7.1. În figură, se observă echipamentele notate E1,...., E12., care corespund genelor cromozomului.

E1	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**E12**
1	0	0	1	1	0	0	0	1	1	0	1

Fig. 7.1. Reprezentarea unei soluții / cromozom

Adaptarea unui cromozom (analiza soluției față de soluția optimă) este cuantificată prin folosirea unor criterii clar definite. În plus, față de această funcție de adaptare, este definită funcția obiectiv. De multe ori, funcția de adaptare derivă din funcția obiectiv. În problema descrisă se dorește un consum minim de energiei electrică, dar și obținerea produselor. Aceste funcții sunt prezentate în următorul subcapitol.

7.3.2. Funcțiile obiectiv și de adaptare

Obiectivul acestei probleme, a algoritmului genetic este descris prin trei relații matematice și o condiție restrictivă. Astfel, sunt definite trei funcții, Fo1, Fo2 și Fo3 și restricția R, prin (7.1) – (7.3).

$$Fo1_j = \sum_{i=1}^{m} c_q * n_i \tag{7.1}$$

Unde Fo1 este prima funcție care definește obiectivul prin care se determină apropierea cromozomului de obiectiv din punctul de vedere a numărului și calitatea propduselor, j reprezintă indicele cromozomul din mulțimea de soluții (populația de cromozomi / indivizi), i este indicele echipamentului / a genei din cromozom, m = 12 este numărul de echipamente, q – calitatea produsului, c_q = 10 pentru q=1, c_q=7 pentru q=2, c_q = 4 pentru q=3, și c_q = 1 pentru q=4. Se poate observa că la această funcție se urmărește maximizarea ei înspre valoarea maximă posibilă $Fo1_{max}$ = 2000. Această valoare s-a obținut pentru produsele din contract.

$$Fo2_j = \sum_{i=1}^{n} P_i \ [kW] \tag{7.2}$$

Unde $Fo2$ reprezintă a doua funcție a obiectivului prin care se determină puterea activă a echipamentelor în funcțiune ale soluției j, P_i este puterea echipamentului i.

$$Fo3_j = \frac{F1_{max}}{F2_j} \tag{7.3}$$

Unde $Fo3$ este timpul în zile necesare soluției j să realizeze produsele din contract.

Din relațiile (9.2) și (9.3) rezultă consumul de energie a unei soluții/cromozom, care se dorește a se minimiza.

$$Cons_j = Fo2_j \cdot Fo3_j \cdot 24\ [kWh] \tag{7.4}$$

Pentru a respecta contractul, se impune o condiție restrictivă, și anume funcționarea a cel puțin a unui echipament din fiecare categorie de produs. Dacă un cromozom nu respectă această condințe, atunci nu va fi considerat viabil. Matematic, valoarea lui *R* se determină prin relația următoare.

$$R_j = \begin{cases} 1, & toate\ E_i^q = 0,\ q = \overline{1,4} \\ 0, & contrar. \end{cases} \tag{7.5}$$

Unde E_i^q este gena *i* a cromozomului *j* corespunzătoare unui echipament care produce produse din categoria *q*. Din relație, rezultă că valoarea restricției poate fi 0 sau 1.

Luând în considerare aceste funcții, se definește funcția de adaptare (fitness function – funcția de potrivire) *F*, care arată cât de apropiată de soluția optimă este un cromozom.

$$F_j = \frac{Fo1_j}{\sum_{j=1}^{m} Fo1_j} \tag{7.6}$$

Din această relație, rezultă că funcția de adaptare este pozitivă, subunitară, iar valoarea ei se urmărește să se maximizeze. În concluzie, un cromozom cu cât are funcția de adaptare mai mare, cu atât este mai potrivit, și are șanse mari să fie folosit în continuare.

7.3.3. Prima populație

Următorul pas în dezvoltarea algoritmului genetic este determinarea numărului de cromozomi / indivizi ai populației, adică numărul de soluții din mulțimea de soluții cu care va lucra algoritmul genetic. Acest număr rămâne constant pe tot parcursul utilizării algoritmului. În acest exemplu, numărul de indivizi, *m* este 10.

Valorile primei populații sunt generate aleator, conform codificării binare.

Pentru exemplu rezentat s-a considerat ca populația are 10 indivizi. Prima populație este generată aleator și prezentată în fig. 7.2.

Primul cromozom al populației are următoarea semnificație - întrucât echipamentele E1, E3, E5, E7, E9 și E11 au asociat valoarea 1, rezultă că acestea vor fi luate în considerare ca fiind în funcțiune, deci atunci când se calculează valoriile funcțiilor obiectiv, din tabelul 1 se vor extrage caracteristicile acestor echipamente.

Odată determinată prima populație, pentru fiecare dintre cromozomi se determină valorile funcțiilor Fo1, Fo2, Fo3 și R. Rezultalele obținute pentru acești cromozomi sunt ilustrate în tabelul 7.2.

	E1	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**E12**
C1	1	0	1	0	1	0	1	0	1	0	1	0
C2	0	1	0	1	0	1	0	1	0	1	0	1
C3	1	1	0	0	1	1	0	0	1	1	0	0
C4	1	1	1	0	0	0	1	1	1	0	0	0
C5	0	0	0	1	1	1	0	0	0	1	1	1
C6	0	0	1	1	0	0	1	1	0	0	1	1
C7	1	1	0	1	1	0	1	1	0	1	1	0
C8	0	0	1	0	0	1	0	0	1	0	0	1
C9	1	0	0	1	0	0	1	0	0	1	0	0
C10	0	1	1	0	1	1	0	1	1	0	1	1

Fig. 7.2. Prima populație de indivizi. C - cromozom

Tabelul 7.2. Valorile funcțiilor obiectiv pentru prima populație

Cromozom	**Fo1**	**Fo2**	**Fo3**	**R**
C1	560	660	3,6	1
C2	740	610	2,7	1
C3	1020	590	1,96	1
C4	860	610	2,32	0

C5	775	660	2,8	0
C6	615	680	3,25	1
C7	665	540	3	1
C8	970	730	2,06	1
C9	115	150	17,4	1
C10	1115	1120	1,8	1

Valorile funcției de adaptare calculată folosind relația matematică (9.6), a cromozomilor primei populații sunt:

$F_1 = 0,07$, $F_2 = 0,09$, $F_3 = 0,13$, $F_4 = 0,11$, $F_5 = 0,1$, $F_6 = 0,082$, $F_7 = 0,09$, $F_8 = 0,13$, $F_9 = 0,015$, $F_{10} = 0,15$.

Analiza rezultatelor obținute ne indică faptul cinci dintre cei 10 cromozomi au funcția de adaptare peste 0,1 și restricția 1, astfel acești cromozomi C2, C3, C7, C8 și C10 vor fi folosiți mai departe în următoarea populație. Trebuie specificat că doi (C4 și C5) dintre cei 10 cromozomi au fost eliminați de la început, din cauza faptului că valoarea lui R a fost 0.

Această metodă de selecție a cromozomilor care sunt folosiți la pasul următor se numește metoda elitistă.

7.3.4. Generarea unei noi populații

Noua populație (mulțime de soluții) va fi formată din cei cinci cei mai adaptați cromozomi din populația veche, plus alți cinci cromozomi noi, care vor fi obținuți din cei din populația anterioară la care se folosesc operatorii genetici de împerechere și mutație. Noua populație este ilustrată în fig. 7.3 cuprinde C2, C3, C7, C8 și C10, care vor deveni C1-C5, iar C6-C10 vor fi cromozomi noi obșinuți astfel:

- C6 și C7 – doi copii ai lui C2 și C3,
- C8 și C9 – doi copii ai lui C8 și C10,
- C10 – mutația a lui C7.

Copiii sunt obținuți prin operatorul genetic de împerechere în două puncte, mai exact genele 1-3 și 10-12 sunt păstrați de la un

părinte, iar genele 4-9 de la celălalt părinte. Astfel, de la doi părinți se pot obține doi copii.

Copiii rezultați prin operatorul genetic de mutație se obțin prin schimbarea valorii unei gene din 1 în 0, sau din 0 în 1. Mutația realizată s-a efectuat asupra genei 3.

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
C1	0	1	0	1	0	1	0	1	0	1	0	1
C2	1	1	0	0	1	1	0	0	1	1	0	0
C3	1	1	0	1	1	0	1	1	0	1	1	0
C4	0	0	1	0	0	1	0	0	1	0	0	1
C5	0	1	1	0	1	1	0	1	1	0	1	1
C6	0	1	0	0	1	1	0	0	1	1	0	1
C7	1	1	0	1	0	1	0	1	0	1	0	0
C8	0	0	1	0	1	1	0	1	1	0	0	1
C9	0	1	1	0	0	1	0	0	1	0	1	1
C10	1	1	1	1	1	0	1	1	0	1	1	0

Fig. 7.3. Noua populație de indivizi. C - cromozom

Noua populație este evaluată folosind aceeași metodă, adică prin determinarea valorilor funcțiilor obiectiv și a funcției de adaptare. Rezultatele obținute sunt prezentate în tabelul 7.3.

Tabelul 7.3. Valorile funcțiilor obiectiv pentru noua populație

Cromozom	Fo1	Fo2	Fo3	R
C1	740	610	2,7	1
C2	1020	590	1,96	1
C3	665	540	3	1

C4	970	730	2,06	1
C5	1115	1120	1,8	1
C6	610	490	3,27	1
C7	705	420	2,83	1
C8	1360	920	1,47	1
C9	1130	930	1,76	1
C10	865	740	2,31	1

Valorile funcției de adaptare calculată folosind relația matematică (6), a cromozomilor primei populații sunt:

$F_1 = 0,08$, $F_2 = 0,11$, $F_3 = 0,07$, $F_4 = 0,1$, $F_5 = 0,12$, $F_6 = 0,065$, $F_7 = 0,75$, $F_8 = 0,148$, $F_9 = 0,123$, $F_{10} = 0,094$.

Valorile restricției sunt 1 pentru toți cromozomi.

Acest proces de formarea de noi populații este repetat până la un număr maxim de generații, de obicei între 100 și 10000. În acest exemplu, procesul de generare de noi cromozomi / soluții s-a oprit la această populație.

Următorul pas este selecția celor mai adaptași cromozomi. În acest exemplu, cromozomii cei mai adaptați sunt C2, C4, C5, C8 și C9. După ordonarea în ordine descrescătoare a acestor cromozomi rezultă în C8, C9, C5, C2 și C4. Pentru aceste cromozomi, se calculează consumul de energie, rezultatele sunt prezentate în tabelul 7.4.

Tabelul 7.4. Cromozomii elitiști a ultimei generații

Nr. ordine	**Cromozom**	**Adaptare**	**Consum MWh**	**Observație**
1	C8	0,148	32,45	Punctajul cel mai mare
2	C9	0,123	39,28	Mediu

3	C5	0,12	48,38	Consumul cel mai mare
4	C2	0,11	27,75	Consumul cel mai mic
5	C4	0,1	36,09	Punctajul cel mai mic

Din analiza tabelului 9.4 se observă că există trei posibile soluții în funcție de ceea ce se dorește. Astfel, dacă se pune accent pe contract, atunci varianta cea mai bună este C8, iar soluția este punerea în funcțiune a echipamentelor 3, 5, 6, 8, 9, și 12.

În cazul în care se dorește consumul minim, atunci cromozomul C2 este soluția problemei, iar echipamentele 1, 2, 5, 6, 9 și 10 vor fi puse în funcțiune.

Ultima variantă este C9, care are punctajul al doilea, și consumul al doilea ca mărime. În această situație, se vor pune în funcțiune echipamentele 2, 3, 6, 9, 11 și 12.

7.4. Mersul lucrării

În cadrul acestei lucrări se vor realiza două activități.

Prima activitate constă în următoarele:

1. Plecând de la a doua populație obținută în capitolul 9.3, se vor realiza încă două generații, luând în considerare operațiile realizate în capitolul 7.3 privind prima populație și a doua populație, respectiv generație.

2. Rezultatele se vor completa în fig. 7.4 și 7.5, respectiv tabelele 7.5 și 7.6.

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
C1												
C2												
C3												
C4												
C5												
C6												
C7												
C8												
C9												
C10												

Fig. 7.4. A treia populație de indivizi. C - cromozom

Tabelul 7.5. Valorile funcțiilor obiectiv pentru a treia populație

Cromozom	Fo1	Fo2	Fo3	R
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
C1												
C2												
C3												
C4												
C5												
C6												
C7												
C8												
C9												
C10												

Fig. 7.5. A patra populație de indivizi. C - cromozom

Tabelul 7.6. Valorile funcțiilor obiectiv pentru a patra populație

Cromozom	Fo1	Fo2	Fo3	R
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

A doua activitate prespune următoarele:

1. Plecând de la prima populație, se va crea o nouă generație unde urmașii vor fi obținuți folosind încrucișarea într-un punct.
2. Rezultatele se completează în fig. 7.6 și tabelul 7.7.

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
C1												
C2												
C3												
C4												
C5												
C6												
C7												
C8												
C9												
C10												

Fig. 7.6. A patra populație de indivizi. C - cromozom

Tabelul 7.7. Valorile funcțiilor obiectiv pentru a cincea populație

Cromozom	Fo1	Fo2	Fo3	R
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

7.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultalele obținute în cele cinci generații se vor compara între ele pentru a determina cum au evoluat cromozomii, respectiv soluțiile.

7.6. Concluzii

Se vor formula concluzii asupra convergenței algoritmului, cu privire la necesitatea tipului criteriului de oprire.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

8

ALGORITMI GENETICI

8.1. Scopul lucrării

În lucrare se urmărește înțelegerea aspectele legate de realizarea unei aplicații de optimizare folosind algoritmi genetici în mediul de modelare și simulare MATLAB.

8.2. Obiectivele lucrării

Principalul obiectiv al lucrării este prezentarea noțiunilor teoretice privind realizarea unei aplicații de optimizare folosind unealta dedicată a mediului de modelare și simulare MATLAB – Global Optimization Toolbox. Deasemenea, se vor prezenta pașii necesari pentru dezvoltarea aplicației și se va testa operarea acesteia în funcție de parametrii ei.

8.3. Noţiuni teoretice.

Algoritmii genetici (AG) intră în categoria algoritmilor evolutivi, care sunt folosiți pentru rezolvarea problemelor de optimizare. Astfel, aceștia vor determina soluția optimă globală dintr-o mulțime de soluții, depășind nivelul optimelor locale.

Algoritmii genetici își au originea în teoria evoluționistă, în consecință, ei se bazează pe soluții care sunt codificate sub forma unor cromozomi care formează populații. Elementele definitorii ale unui algoritm genetic sunt:

- Populația (population) – mulțimea de indivizi (soluții) care trăiesc și se adapează la condițiile cerute.
- Cromozom (chromosone) – este codificarea unei soluții. Cromozomii sunt alcătuiți din gene, care definesc caracteristicile unui individ. Precum în genetică, pozițiile genelor sunt denumite loci, iar valorile lor alele.
- Generație (generation) – reprezintă o iterație în procesul de evoluție.
- Adaptare (fitness) – reprezintă gradul de satisfacere a condițiilor corespunzătoare mediului (problemei). Toți indivizii unei populații sunt testați pentru a se determina adaptarea lor.
- Selecție (selection) – reprezintă operația prin care din populația de indivizi sunt aleși cei mai adaptați indivizi, adică cei care vor merge mai departe în următoarea generație, respectiv vor fi folosiți pentru a se reproduce.
- Reproducere/ împerechere/ încrucișare (reproduction / crossover) – este operația prin care se formează noi indivizi din cei selectați. Acești indivizi vor moșteni din caracteristicile părinților în funcție de strategia de împerechere.
- Mutație (mutation) – reprezintă operația genetică prin care unui individ i se modifică valoarea unui gene.

Algoritmii genetici sunt procese repetitive, prin care, pornind de la o populație inițială cu indivizi inițializați aleator, se construiesc noi indivizi și populații folosind operatori genetici specifici (selecția, reproducere, mutație). Acest proces se va opri când este îndeplinit criteriul de oprire, precum atingerea unui număr maxim de generații. În cadrul acestui proces, pentru fiecare individ a populației se calculează funcția de adaptare. Un algoritm genetic este caracterizat printr-o funcție obiectiv, respectiv restricții (constrângeri).

Există o mare varietate de algoritmi genetici, care sunt construiți pentru a răspunde fidel problemei corespunzătoare lor. Totuși toți AG pot fi caracterizați printr-o structură de bază, care este ilustrată în fig. 8.1.

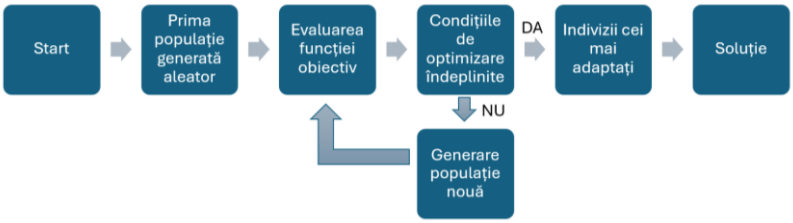


Fig. 8.1. Digrama funcțională a unui algoritm genetic

Un cromozon este caracterizat prin mai multe gene, care în esență sunt variabile ale soluției. Deci, fiecare soluție este influențată de valorile variabilelor. Fig. 8.2. prezintă structura generală a unui cromozon.

Var 1	**Var 2**	**Var 3**	**....**	**Var *i***	**....**	**Var *n***

Fig. 8.2. Structura generală a unui cromozon. Var 1 – prima variabilă a cromozomului, *i* – indicele unei variabile oarecare, *n* – numărul de variabile a cromozomului

Mediul de simulare MATLAB conține o unealtă pentru rezolvarea problemelor de optimizare prin Global Optimization Toolbox.

Global Optimization Toolbox oferă posibilitatea utilizării unor funcții prin care se caută soluțiile globale pentru probleme de optimizare. Printre algoritmii de rezolvare (solvers) se numără: surogatul, algoritmul genetic, roiul de particule și căutarea globală. Accesul la această unealtă se face prin intermediul interfeței din fig. 8.3, și anume ”Live Editor” realizând acțiunea

New > Live Script

urmată de

Task > Optimize

Fișierul nou deschis va avea extensia ”mlx”.

Așa cum se observă în fig. 8.3., Optimize are două variante de lucru, și anume optimizarea bazată pe problemă (Problem-based) și

optimizarea bazată pe algoritmul de rezolvare (Solver-Based). Varianta cea mai populară (recomandată) este prima abordare, și anume optimizarea bazată pe problemă.

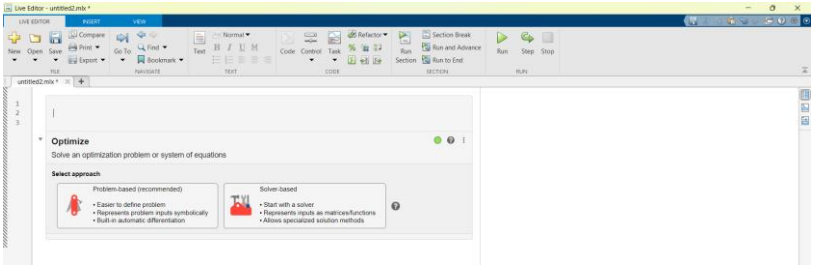


Fig. 8.3. Interfața MATLAB – Live Editor

8.3.1. Etapele dezvoltării aplicației

Dezvoltarea unei aplicații de optimizare prin folosirea Live Editor și Optimize se realizează urmând etapele rezumate în fig. 8.4. În continuare se vor descrie pașii necesari pentru construirea aplicației în varianta de utilizare a abordării bazate pe probleme, adică ”Problem-based approach”.

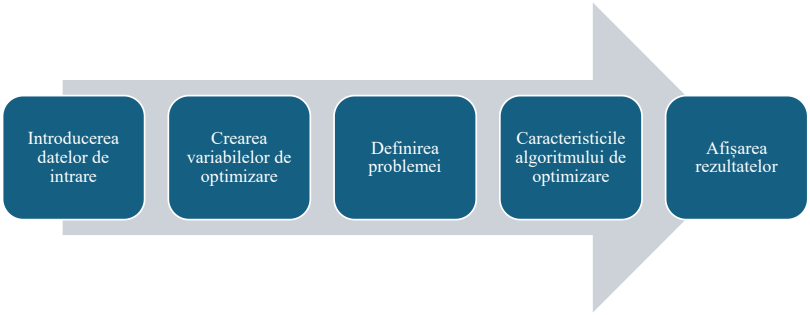


Fig. 8.4. Etapele dezvoltării aplicației folosind Live Editor și Optimize

Prima etapă constă în introducerea datelor de intrare, adică a variabilelor de intrare. Acestea trebuie definite înainte de deschiderea interfeței Optimize, mai exact în liniile de comandă notate 1, 2, 3… din fig. 8.3. Trebuie specificat, că definirea variabilelor poate să lipsească, dar este recomandat ca acestea să se definească înainte de introducerea problemei de optimizat pentru a preveni erori ulterioare.

Următoarele etape din fig. 8.4. se realizează prin intermediul uneltei Optimize ilustrată de fig. 8.5., unde se observă două ferestre, și anume – fereastra de definire a problemei și a caracteristicilor algoritmului de optimizare, respectiv ferestra de afișare a problemei și rezolvării în timp real (”live script”).

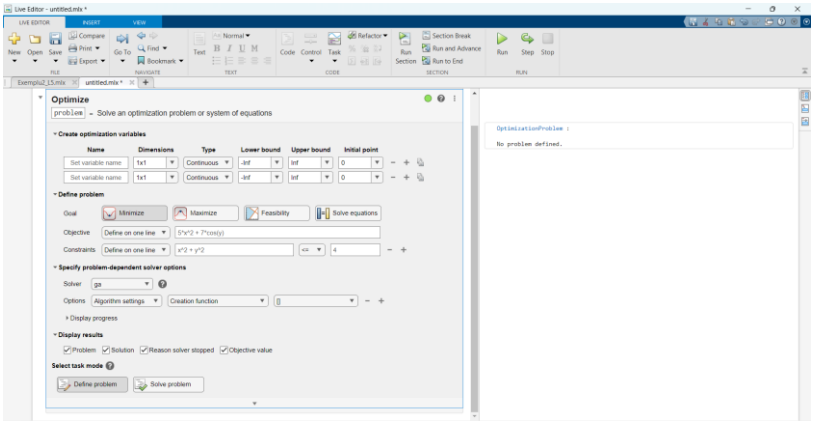


Fig. 8.5. Interfața Optimize – Problem-Based

A. Crearea variabilelor de optimizare (Create optimization variables) se realizează prin intermediul indicatorilor de intrare corespunzători: nume, dimensiunea, tipul, limita inferioară, limita superioară și valoarea inițială. Se poate observa în fig. 8.6. definirea a două variabile.

B. Definirea problemei (Define problem) constă în introducerea scopului, obiectivului și constrângerilor problemei.

Scopul problemei poate să fie minimizarea (determinarea valorii minime), maximizarea (determinarea valorii maxime), determinarea fezabilității (problema se poate rezolva) sau rezolvarea ecuațiilor.

Obiectivul problemei este reprezentat printr-o funcție, care poate fi definită printr-o linie de comandă, așa cum se poate vedea în fig. 8.6., dar și printr-o funcție locală, sau un fișier care conține funcția. Aceste variante sunt clar ilustrate în fig. 8.7.

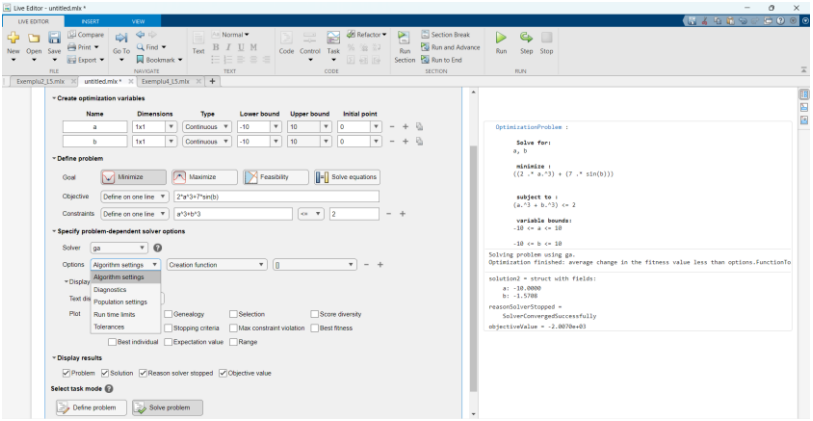


Fig. 8.6. Interfața Optimize – exemplu 1. Optimizarea unei probleme cu două variabile a și b

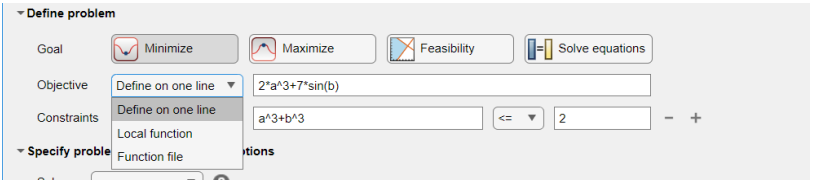


Fig. 8.7. Interfața Optimize – Opțiunile de introducere a funcției obiectiv

C. Caracteristicile algoritmului de optimizare (Specify problem-dependent solver options) se modifică în funcție de algoritmul de optimizare ales (solver). În cazul de față se va alege algoritmul ”ga”, care corespunde rezolvării prin angajarea unui algoritm genetic. Astfel, caracteristicile algoritmului (fig. 8.8.) ”ga” se referă la setările algoritmului, funcția de ieșire, toleranța, setările populației, criteriul de oprire.

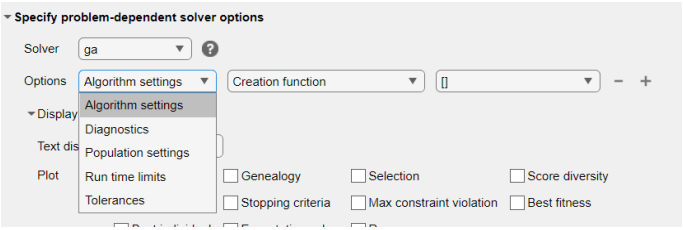


Fig. 8.8. Interfața Optimize – Opțiunile pentru caracteristicile algoritmului

Setările algoritmului ilustrate în fig. 8.9. se referă la funcțiile de creație, împerechere, adaptare, mutație, selecție, dar și la posibilitatea de a evalua funcțiile în paralel, introducerea unei optimizări hibride, sau a unei constrângeri neliniare. Pentru fiecare dintre aceste setări se oferă posibilitatea de a alege mai multe variante predefinite.

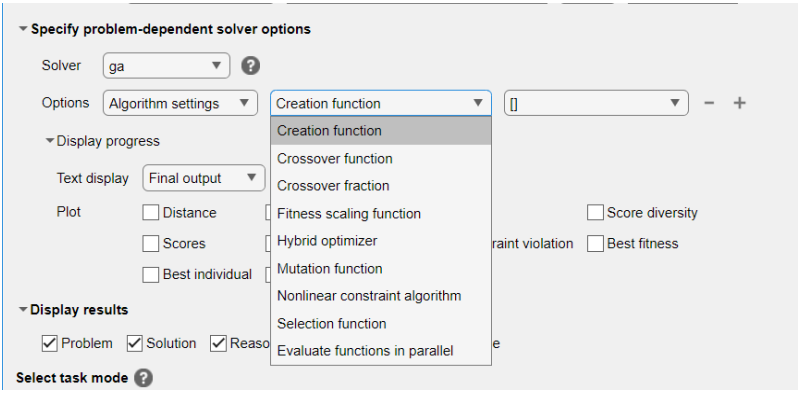


Fig. 8.9. Interfața Optimize – Setările algoritmului

Funcția de creație este opțiunea care ajută la generarea populației inițiale, de la care pornește AG. Varianta predefinită a funcției de creație este ”gacreationuniform”, prin care se generează indivizi uniform, aleator, între limitele impuse. Celelalte variante sunt: gacreationlinearfeasible (generează indivizi fezabili în raport cu constrângerile liniare, gacreationsobol (generează indivizi folosind secvența Sobol, care este o secvențo cvasi-aleatoare cu discrepanță redusă) și gacreationuniformint (generează indivizi de valori întregi, uniform, eșanționați uniform între limitele impuse).

Funcția de selecție este folosită pentru a selecta indivizii care vor intra în procesul de obținere de noi soluții / urmași. Selecția se poate face alegând dintr-o listă de cinci funcții predefinite: selectionstochunif (selecția proporțională uniformă în relație cu valoarea adaptării), selectionremainder (fiecare individ are un număr de reproducere atribuit în funcție de fitness), selectionuniform (selecția uniformă – ignora valoarea de adaptare și selectează aleator

părinții), selectionroulette (operatorul de selecție prin metoda ruletei) și selectiontournament (selecția după rang).

Funcția de încrucișare definește modul în care sunt obținuți cromozomi urmași din cromozomi părinți. Această funcție se poate alege din mai multe variante predefinite: crossoverscattered (crează urmași prin alegerea genelor în mod aleator de la părinți), crossoverheuristic (creează urmași prin extrapolarea dincolo de părintele mai bun.), crossoversinglepoint (încrucișarea într-un punct), crossovertwopoint (încrucișarea în două puncte), crossoverarithmetic (creează urmași prin calcularea unei medii ponderate a doi vectori părinți), și crossoverlaplace (creează urmași folosind o distribuție Laplace în jurul părintelui mai bun, introducând mai multă explorare decât o simplă încrucișare aritmetică).

Fracția de încrucișare este proporția de indivizi din fiecare generație care sunt creați folosind funcția de încrucișare (adică prin combinarea soluțiilor părinților). Indivizii rămași sunt creați doar folosind mutația. Aceasta ia valori în intervalul [0,1 1], o valoare uzuală este 0,8.

Funcția de scalare a adaptării este folosită pentru a transforma valorile brute ale funcției de adaptare într-o formă mai potrivită pentru selecție. Este o parte critică a procesului de selecție și influențează direct care indivizi sunt mai predispuși să fie aleși drept părinți pentru următoarea generație. În MATLAB sunt patru funcții predefinite : fitscalingrank (indivizii sunt ordonați în ordine descrescătoare după rang în funcție a adaptare), fitscalingshiftlinear (se aplică o transformare liniară deplasată la valorii de adaptare brute), fitscalingprop (adaptarea scalată este proporțională cu adaptarea brută) și fitscalingtop (doar câțiva indivizi de top primesc o valoare de fitness scalată diferită de zero).

Funcția de mutație are rolul operatorului genetic de mutație, astfel introduce modificări în valoarea genelor cromozomilor aleși pentru a obține noi cromozomi. MATLAB oferă cinci funcții predefinite: mutationgaussian (adaugă zgomot Gaussian la valoarea genelor mutante), mutationpower (mutații distribuite prin legea puterii -), mutationuniform (înlocuiește uniform valoarea genelor mutante cu valori aleatorii), mutationpositivebasis (folosește vectori cu bază pozitivă pentru mutații informate direcțional) și

mutationadaptfeasuble (realizează mutații aleatorii prin respectarea constrângerilor).

Algoritmul de constrângere neliniară este utilizat pentru a gestiona constrângerilor neliniare în timpul procesului de optimizare. Aceste constrângeri sunt furnizate de utilizator printr-o funcție de constrângere neliniară. Algoritmul are două variante predefinite: auglag și penalty. Auglag este strategia prin care se transformă problema constrânsă într-o secvență de probleme neconstrânse folosind o funcție lagrangiană augmentată. Penalty este o abordare folosită prin care cromozomilor care nu respect constrângerile le este modificată forțat valoarea funcție de adaptare pentru fi mai puțin probabil să fie selectați. Astfel, când se urmărește minimizarea obiectivului, funcția este mărită, iar când se urmărește maximizarea, valoarea funcției este micșorată.

Evaluarea funcțiilor în paralel poate să fie activate prin bifarea căsuție corespunzătoare.

Optimizare hibridă este folosită după ce algoritmul genetic își termină căutarea globală. Deci, funcția hibridă preia cea mai bună soluție globală și efectuează o căutare locală pentru a o ajusta fin. Această operație se poate realiza prin intermediul a patru funcții predefinite: fmincon (optimizator local neliniar constrâns), fminunc (optimizator local neliniar fără constrângeri), fminserach (metoda simplex neconstrânsă) și patternserach (căutare locală robustă, fără derivate).

Funcția de ieșire se referă la diagnosticarea algoritmului. Aceasta este dată de utilizator.

Toleranțele sunt parametrii care definesc momentul în care algoritmul ar trebui să se termine - adică, când îmbunătățirile suplimentare sunt neglijabile sau se detectează convergență. Așa cum se observă în fig. 5.10 sunt trei variante dintre care se poate alege: toleranța funcțională, limita valorii funcției de adaptare, toleranța la contrângeri.

Toleranța funcțională se referă la oprirea algoritmului când schimbarea valorii optime de fitness între generații devine mai mică decât această toleranță. Valoarea predefinită este 1e-6 = 0,000001.

Toleranța la constrângeri va opri algoritmul și o soluție este considerată viabilă atunci când este determinată încălcarea maximă permisă a constrângerii.

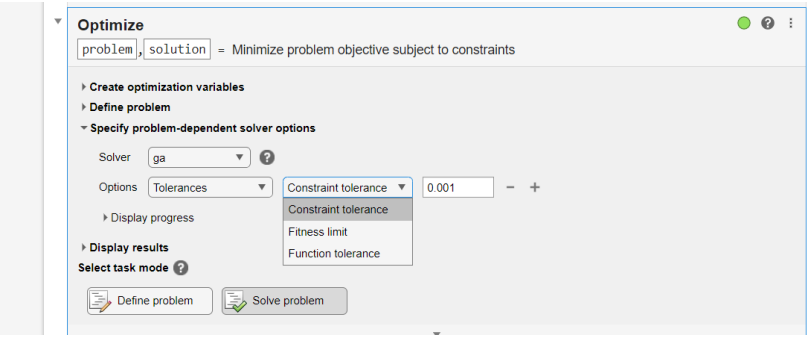


Fig. 8.10. Interfața Optimize – Toleranțe

Limita valorii fitness este o obțiune pentru oprirea algoritmului, care este valoare numerică ce indică cea mai bună valoare a adaptării care se acceptă. Dacă cea mai bună valoare fitness a individului din populație atinge sau scade sub această valoare, algoritmul se oprește.

Setările populației implică alegerea dintre trei variante: mărimea populației, numărul de elite și intervalul inițial al populației. Mărimea populație este deobicei un număr între 50 și 200, care reprezintă numărul de indivizi ai unei populații. Numărul de elite este numărul de cromozomi cu performanțe de top care sunt transferați direct neschimbați în generația următoare. Ultima setare definește intervalul de valori utilizat pentru inițializarea aleatorie a populației

Limitele de execuție sunt condiții de oprire care permit să se controleze durata de funcționare a algoritmului de optimizare - fie prin timp, generații sau evaluări ale funcțiilor. Aceste setări sunt utile în special pentru sarcinile sensibile la timp, sistemele în timp real sau problemele mari în care AG ar putea rula ore sau zile. Cea mai populară condiție de oprire este numărul maxim de generații, care deobicei este 100 generații. O altă posibilă condiție de oprire este timpul maxim de rulare a algoritmului. Alte condiții de oprire se referă la numărul maxim de generații permise fără ca valoarea optimă să se modifice substanțial sau la faptul că valoarea optimă de fitness nu s-a îmbunătățit timp de un număr specificat de secunde.

D. Afișarea progresului (Display progress) controlează câte informații despre progresul algoritmului (rezultatele intermediare)

sunt afișate în Fereastra de Comandă în timpul procesului de optimizare. Aceasta ajută la monitorizarea comportamentului, performanței și convergenței algoritmului în timp real. Fig. 8.11 ilustrează fereastra cu posibilitățile de afișare oferite prin interfața Optimize. În imaginea de mai jos se pot identifica două aspecte, și anume afișarea textului, respectiv a reprezentării grafice (plot). Prin afișare text (Text display) se oferă posibilitatea afișării sub formă de text a valorii de ieșire, a fiecărei iterații, respectiv a diagnosticării. În cazul reprezentării grafice, se pot selecta 11 parametrii care să fie afișa grafic: distanța, generalogia, selectia, scorul de diversitate, scorurile (valorile de fitness), criteriul de oprire, valoarea maximă în cazul violării constrângerilor, cea mai bună valoare a funcției de adaptare, cel mai bun cromozom, valoarea așteptată și intervalul de soluții.

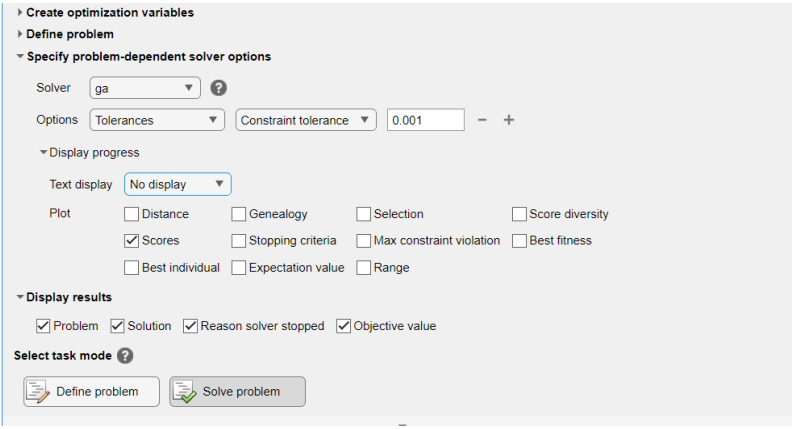


Fig. 8.11. Interfața Optimize – Afișarea progresului

E. Afișarea rezultatelor (Display results) sunt rezumatul final al procesului de optimizare afișat în Fereastra de comandă. Așa cum ilustrează fig. 8.11, utilizatorul poate alege să primească informații sintetizate despre problemă, soluție, cauza opririi algoritmului și valoarea funcției obiectivului.

8.3.2. Testarea algoritmului genetic

Testarea unui AG se poate face eficient urmând o abordare structurată, care nu depinde de funcția obiectiv.

Pentru testarea unui AG se parcurg pașii A-E de la capitolul precedent păstrându-se variantele implicite (default). Apoi se modifică setările algoritmului și ceilalți parametrii enumerați pentru a scoate în evidență diferențele dintre rezultate. La final, rezultatele se compară între ele cu scopul de a determina setările optime pentru problema de rezolvat.

8.4. Mersul lucrării

În cadrul acestei lucrări se va implementa și studia funcționalitatea unui algoritm genetic prin utilizarea uneltei Optimize din MATLAB. Pentru atingerea acestui obiectiv se vor realiza activitățile enumerate în continuare.

8.4.1. Dezvoltarea aplicației

Se va dezvolta aplicația a cărui scop este determinarea punctuluide minim a funcției din fig. 8.12. Primul pas este definirea variabilelor funcției, x și y pe intervalul [-2, 2]. Relația matematică a funcției este

f = @(x,y) x.*exp(-x.^2-y.^2)+(x.^2+y.^2)/20.

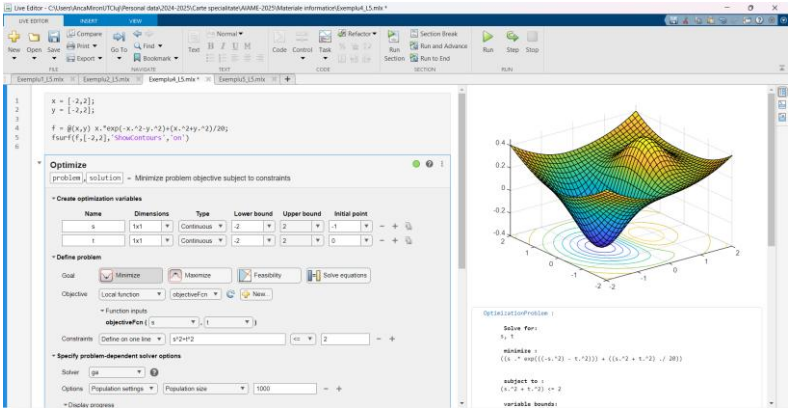


Fig. 8.12. Interfața Optimize – Aplicația test

Pentru afișarea funcției se folosește linia de comandă:
fsurf(f,[-2,2],'ShowContours','on').

Se va accesa unelata Optimize și se va dezvolta aplicația conform fig. 8.13.

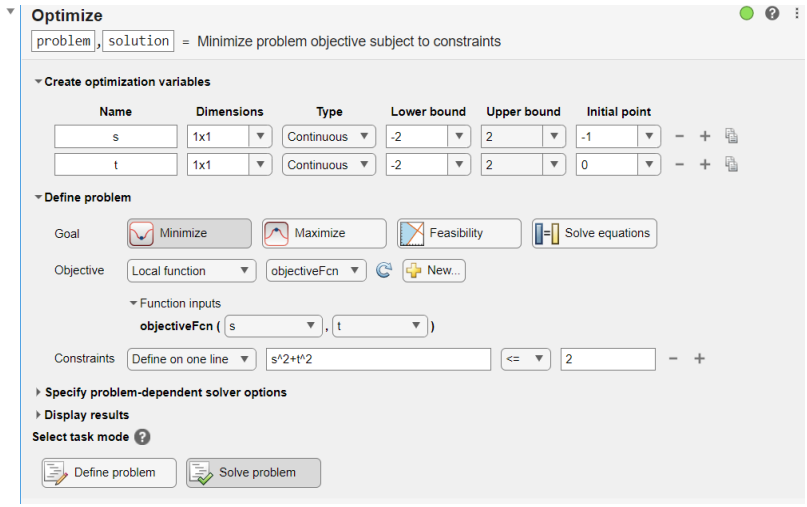


Fig. 8.13. Interfața Optimize – Aplicație test– definirea problemei

Se va selecta algoritmul de rezolutiv, solver, ga, iar apoi se acționează butonul Solve problem, și se va nota punctul de minim și valoarea funcției obiectiv în tabelul 8.1 pentru AG 0.

8.4.2. Testarea aplicației

Pentru testarea aplicației dezvoltate se vor modifica valorile implicite ale setărilor algoritmului și a celorlalți parametri cu valori / variante predefinite sau utilizate în mod uzual. Rezultatele sunt trecute în tabelul 5.1, începând de la AG 1. Opțiunile modificate față de varianta 0 a AG dezvoltat se vor realiza conform coloanei 2, Modificări, respectiv discutate în timpul ședinței de laborator. În coloana 3, Rezultate se vor trece valoarea punctului de minim și valoarea funcției obiectiv.

Tabelul 8.1. Rezultatele testării AG

AG	Modificări	Rezultate	Observații
0			
1	100 indivizi Oprire în 10 sec		
2	200 indivizi Oprire în 20 sec		
3	200 generații		
4	500 generații		
5			
6			

7			
8			
9			
10			
11			
12			
13			
14			

8.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultatele din tabelul 8.1 se compară și se consideră modificările aduse variantei 0 a aplicației pentru a completa coloana 4, Observații.

8.6. Concluzii

Se vor formula concluzii asupra dezvoltării aplicației și a rezultatelor obținute.

.....

.....

.....

.....

.....

.....

.....

9

ALGORITM GENETIC PENTRU OPTIMIZAREA UNUI PROCES ENERGETIC

9.1. Scopul lucrării

În lucrare se urmărește înțelegerea aspectele legate de realizarea unei aplicații dedicate modelării și simulării unui algoritm genetic folosind MATLAB, cu scopul de a crește eficiența energetică prin optimizarea procesului energetic.

Lucrarea se bazează pe un scenariu care presupune un proces tehnologic într-o fabrică, care produce componente electronice din patru categorii, cu ajutorul a 20 de aparate profesionale. Țelul fabrici este de a produce componentele într-o săptămână de lucru de 6 zile cu cel mai mic consum de energie.

9.2. Obiectivele lucrării

Principalul obiectiv al lucrării este prezentarea modului în care se poate dezvolta un algoritm genetic în MATLAB și descrierea explicită a pașilor necesari pentru construirea și testarea AG. Astfel, plecând de la scenariu descris anterior, se urmărește dezvoltarea unui AG care să dea soluția optimă de utilizare a aparatelor pentru a obține componentele necesare.

9.3. Noţiuni teoretice

Optimizarea unui proces energetic se referă la procesul prin care se îmbunătățește modul în care energia este folosită, astfel încât acesta să funcționeze mai eficient, mai fiabil, mai economic sau mai sustenabil - în funcție de obiectiv.

Optimizarea implică găsirea celor mai bune condiții de funcționare sau a parametrilor de proiectare care duc la cea mai bună performanță conform unei funcții obiectiv specifice, dar și luând în considerare constrângerile necesare, precum limitele fizice sau standardele de siguranță.

Fabrica din scenariu implicat are 20 de aparate care sunt caracterizate prin datele din tabelul 9.1.

Tabelul 9.1. Caracteristicile aparatelor fabricii

Aparat	P [kW]	Nr. componente produse zilnic	Tip componente
1	25	100	A
2	30	120	A
3	40	200	A
4	28	150	A
5	50	220	A
6	35	180	B
7	45	250	B
8	38	170	B
9	42	200	B
10	55	300	B
11	22	90	C
12	30	150	C
13	33	180	C
14	48	210	C
15	52	240	C
16	27	130	D
17	32	160	D
18	40	200	D
19	60	300	D
20	70	400	D

Realizarea aplicației AG presupune parcurgerea următorilor pași:

Pasul 1. Definirea problemei

Acest pas este cel mai important, întrucât de el depinde rezolvarea problemei cu ajutorul AG. La acest pas se definește obiectivul AG, adică minimizarea sau maximizarea unor valori.

În cazul scenariului din lucrare este vorba de minimizarea costurilor producției prin programarea eficientă a utilizării aparetelor dispoznibile.

Pasul 2. Definirea caracteristicilor AG

Caracteristicile AG se referă la:

- codificarea soluției (binară, cu valori numerice sau simboluri). Se va folosi codificarea prin valori numerice naturale pozitive, care corespund numărului de zile care funcționează aparatele. Deci, se vor folosi cromozomi cu 20 de gene, a căror valori sunt între 0 și 6 (maxim 6 zile pe săptămână).
- definirea funcției obiectiv, a funcției fitness și a constrângerilor. Funcția obiectiv se determină pentru fiecare cromozom ca fiind consumul total de energie a cromozomului.
- alegerea operatorilor genetici – selecția, încrucișarea și mutația. Caracteristicile AG sunt: 100 de indivizi, 100 de generații, număr de elite – 2, funcția de creație este varianta uniformă, încrucișarea laplace, selecția după rang și mutația după putere, fracția de încrucișare 0,8, iar de mutație 0,2.
- alegearea strategiei de construcție a noilor populații. Întrucât este folosită o strategia elitistă, cu numărul de elite 2, atunci noua populație este construită astfel: cele mai bune 2 soluții trec nechimbate în următoarea generație, iar din restul 80% sunt obținute prin încrucișare, iar 20% prin mutație.
- alegerea criteriului de oprire este numărul maxim de generații, care este 100.

Pasul 3. Construirea primei generații

Construirea primei generații începe prin alegerea numărului de indivizi care construiesc o populație, apoi inițializarea indivizilor cu valori aleatorii în funcție de codificarea soluțiilor.

Prima populație este evaluată prin funcția de adaptare / fitness, iar cu ajutorul opertorilor genetici se selectează indivizii părinți, și se produc noi soluții prin încrucișare și mutație.

Pasul 4. Evoluția următoarelor generații

Se formează noua populație folosind cromozomii urmași și cromozomii din vechea generație. La noile generații se reiau acțiunile de evaluare, selecție, încrucișare, mutație și formare de noi populații.

Pasul 5. Finalizarea algoritmului

Satisfacerea criteriului de oprire duce la finalizarea calculelor. Dintre indivizii ultimei populații se alege soluția cea mai bună, care reprezintă ieșirea AG.

9.3.1. Implementarea AG

Pentru implementarea AG dedicat scenariului descris se folosește Live Script, care se accesează din pagina principală MATLAB.

În prima etapă se va defini funcția fitness, care se va salva într-un fișier separat, în același director cu programul principal, urmând ca ulterior să fie apelată. Sintaxa care se scrie în linia de comandă este:

```
function cost = fitness_aparate(x, aparate, demand)
    x = round(x);  % Zile întregi de funcționare
    putere = aparate(:,1);
    iesire = aparate(:,2);
    tip_comp = aparate(:,3);

    % Consum total de energie
    totalEnergy = sum(putere .* x');

    % Urmărirea producției
    prod = zeros(4,1);
    for i = 1:length(x)
        q = tip_comp(i);
        prod(q) = prod(q) + x(i) * iesire(i);
    end

    % Logică de substituție: A poate înlocui B/C/D, B poate înlocui C/D etc.
    remaining = demand(:);
    for q = 1:4
        available = prod(q);
        for d = q:4
            needed = remaining(d);
            used = min(available, needed);
```

```
            remaining(d) = remaining(d) - used;
            available = available - used;
        end
    end
    % Penalizare pentru cererea nesatisfăcută
    penalty = sum(remaining .* 10000);  % Penalitate mare

    % Cost Final
    cost = totalEnergy + penalty;
end
```

În a doua etapă se scrie programul principal într-un fișier *.mlx, care reprezintă AG, și se afișează rezultatul.

9.4. Mersul lucrării

În cadrul lucrării se vor realiza următoarele activități:

A. Se va scrie și salva fișierul *.mlx cu funcția fitness.

B. Se ca scrie și salva programul principal a cărui sintaxă este:

```
clc; clear;

% Caracteristicile aparatelor: [Putere kW, Componente pe zi, Tipul componentelor]
% Tipul componentelor: A=1, B=2, C=3, D=4
aparate = [
    25, 100, 1;
    30, 120, 1;
    40, 200, 1;
    28, 150, 1;
    50, 220, 1;
    35, 180, 2;
    45, 250, 2;
    38, 170, 2;
    42, 200, 2;
    55, 300, 2;
    22,  90, 3;
    30, 150, 3;
    33, 180, 3;
```

```
    48, 210, 3;
    52, 240, 3;
    27, 130, 4;
    32, 160, 4;
    40, 200, 4;
    60, 300, 4;
    70, 400, 4;
];

% Cererea săptămânală pentru componentele A-D
demand = [1200, 1800, 1600, 2000];

nVars = size(aparate, 1);
lb = zeros(1, nVars);
ub = 6 * ones(1, nVars);  % Maxim 6 zile pe săptămână

fitnessFcn = @(x) fitness_aparate(x, aparate, demand);

options = optimoptions('ga', ...
    'PopulationSize', 100, ...
    'MaxGenerations', 100, ...
    'EliteCount', 2, ...
    'Display', 'iter', ...
    'PlotFcn', {@gaplotbestf});

[x_opt, fval] = ga(fitnessFcn, nVars, [], [], [], [], lb, ub, [], 1:nVars,
options);

% Output
disp('--- Programul optim de funcționare (Zile per aparat) ---');
disp(array2table(x_opt', ...
    'VariableNames', {'Zile'}, ...
    'RowNames', strcat("Aparat_", string(1:nVars))));

disp(['Consum total de energie: ', num2str(fval), ' kWh']);

% Numărul total de componente produse
prod = zeros(1,4);
for i = 1:nVars
    q = aparate(i,3);
    prod(q) = prod(q) + x_opt(i) * aparate(i,2);
```

```
end

disp('--- Componente Produse Tip componente ---');
disp(array2table(prod, 'VariableNames', {'A','B','C','D'}));

bar(prod, 'FaceColor', [0.2 0.6 0.5]);
set(gca, 'XTickLabel', {'A','B','C','D'});
ylabel('Componente produse');
title('Producția săptămânală de componente');
grid on;
```

Rezultatul obținut se va trece în tabelul 9.2. În continuare, se vor modifica caracteristicile AG prin adăugarea de noi linii de comandă la opțiunile AG.

Tabelul 9.2. Rezultatele obținute cu AG

AG	**Caracteristici AG**	**Rezultat Cromozom, kWh**	**Observații**
0			
1	50 indivizi		
2	50 generații		
3			

4			
5			
6			
7			
8			
9			
10			
11			

9.5. Prelucrarea datelor și interpretarea rezultatelor

Rezultatele din tabelul 9.2 se compară și se consideră modificările aduse variantei 0 a aplicației pentru a completa coloana 4, Observații.

9.6. Concluzii

Se vor formula concluzii asupra aplicației, a eficienței ei și legăturii dintre rezultate și caracteristicile AG.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

BIBLIOGRAFIE

1. https://www.mathworks.com/
2. Laura Ivanciu, Gabriel Olten, Sisteme fuzzy. Îndrumător de laborator / Fuzzy logic sistems. Laboratory manual, UTPRESS, Cluj-Napoca, 2024, ISBN: 978-606-737-711-8.