Proiect
Tehnici de inteligență computațională în controlul proceselor
(2023-2024)

Îndrumător:

Prof. Cristian Boldișor

Student:

Patrania Bogdan Andrei

Grupa 4MF132

2023-2024

Cuprins

1. Tema proiectului

A)

Fie un sistem de inferenţă fuzzy de tip Mamdani cu două variabile de intrare – *eroarea* şi *derivata_erorii* – şi o variabilă de ieşire – *comanda*. Termenii lingvistici ai variabilelor de intrare sunt descrişi în figura 1, iar baza de reguli este descrisă în tabelul 1. Termenii lingvistici ai variabilei de ieşire sunt identici cu cei ai variabilei *eroare* (figura 1a). Pentru fiecare din cele trei variabile mulţimea de bază este intervalul $[-1;\ +1]$.

Metoda de inferenţă propusă este de tip Mamdami, adică: a) operaţiile de intersecţie şi implicaţie fuzzy sunt realizate cu operatorul *min*, iar operaţiile de reuniune cu operatorul *max*; b) evaluarea inferenţelor conţine o etapă de defuzzyficare (metoda de defuzzyficare nu este impusă).

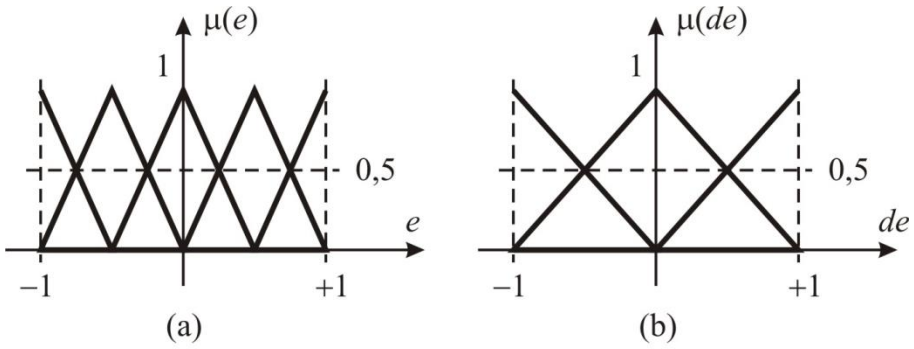

Fig. 1

Tabelul 1.

Comanda este ...		*Derivata erorii*		
		N	Z	B
Eroarea	NB	NB	NB	NS
	NS	NS	NS	ZE
	ZE	ZE	ZE	ZE
	PS	ZE	PS	PS
	PB	PS	PB	PB

Realizaţi un program pentru implementarea sistemului de inferenţă considerat.

Pentru verificare, comparaţi rezultatele obţinute prin program cu cele obţinute în Matlab.

Observaţie. Sistemul de inferenţă considerat este specific unui regulator fuzzy de tip proporţional-derivativ (PD) utilizat în structura convenţională de reglare automată .

B)

Pentru optimizarea implementării numerice a unui sistem de inferenţă fuzzy, o soluţie este aproximarea sistemului de inferenţă printr-un tabel de inferenţă Mamdani. Pentru acest lucru, fiecărei variabile de intrare i se stabileşte o mulţime de bază discretă, păstrând un număr de puncte din mulţimea de bază continuă dată iniţial. Astfel:

a) pentru variabila *eroare* se aleg 11 valori din mulţimea de bază continuă:

$$E = \{-1\ ;\ -0{,}8\ ;\ -0{,}6\ ;\ -0{,}4\ ;\ -0{,}2\ ;\ 0\ ;\ 0{,}2\ ;\ 0{,}4\ ;\ 0{,}6\ ;\ 0{,}8\ ;\ 1\}$$

b) pentru variabila *derivata_erorii* se aleg 5 valori

$$DE = \{-1\ ;\ -0{,}5\ ;\ 0\ ;\ 0{,}5\ ;\ 1\}$$

iar tabela de inferenţă Mamdani va avea structura din tabelul 2.

Se cere să se calculeze valorile din tabelul de inferenţă Mamdani pentru baza de reguli descrisă anterior.

Tabelul 2.

		eroarea										
		-1	-0,8	-0,6	-0,4	-0,2	0	0,2	0,4	0,6	0,8	1
derivata erorii	-1	?	?	?	?	?	?	?	?	?	?	?
	-0,5	?	?	?	?	?	?	?	?	?	?	?
	0	?	?	?	?	?	?	?	?	?	?	?
	0,5	?	?	?	?	?	?	?	?	?	?	?
	1	?	?	?	?	?	?	?	?	?	?	?

C)

Având tabela de inferenţă, implementarea numerică a algoritmului de reglare proporţional-derivativ (din programul de la punctul A) se schimbă.

Realizaţi un al doilea program care implementează algoritmul de inferenţă fuzzy pe tabela de decizie determinată anterior.

Pentru verificare, testaţi rezultatele obţinute cu cele două programe, pentru câteva perechi de valori (e, de) din mulţimea discretă $E \times DE$. Dacă implementarea este corectă, valoarea mărimii de ieşire *comanda* obţinută pentru o anumită pereche de valori cu primul program şi cea obţinută cu al doilea sunt egale.

D)

În sistemele de reglare numerice, derivata mărimii de eroare este calculată din valorile înregistrate ale erorii şi pe baza perioadei de eşantionare. În varianta cea mai simplă, relaţia de calcul a derivatei este:

$$de[k] = \frac{e[k] - e[k-1]}{T_e}$$

unde $de[k]$ este valoarea derivatei erorii corespunzătoare iteraţiei curente, $e[k]$ şi $e[k-1]$ sunt valorile mărimii de eroare de la iteraţia curentă, respectiv de la iteraţia anterioară, iar T_e este perioada de eşantionare (intervalul de timp dintre două iteraţii consecutive).

Pentru calculul derivatei numerice a erorii, considerăm perioada de eşantionare $T_e = 0{,}1$.

În fişierul ”date.txt” se găseşte un set de valori ”înregistrate” ale mărimii de *eroare* dintr-un sistem automat. **Calculaţi valorile *derivatei_erorii* pentru fiecare eşantion înregistrat**. Apoi, având setul de perechi de valori formate din valori ale *erorii* şi valorile calculate ale *derivatei*, **calculaţi toate valorile mărimii de *comandă* separat prin cele două programe**. **Completaţi programele realizate astfel încât, să măsoare intervalul de timp necesar calculului valorilor semnalului de comandă pentru toate eşantioanele din fişierul de date.**

2. Indicații, recomandări, cerințe suplimentare

a) Pentru calculul valorilor în tabela de inferenţă Mamdani se poate folosi orice aplicaţie de editare şi analiză a sistemelor de inferenţă fuzzy (cum ar fi editorul de sisteme de inferenţă fuzzy din pachetul Matlab).

b) Pentru evaluarea inferenţelor fuzzy pe baza tabelei de inferenţă Mamdani, este necesară definirea unei funcţii care să indice distanţa de la o pereche de valori numerice reale (e, de) la o combinaţie de valori (e^*, de^*) din mulţimea de bază discretă. Practic, este necesară o funcţie aleasă convenabil care descrie distanţa dintre două puncte în plan.

c) Programele modificate pentru setul de date de la punctul D trebuie să cronometreze durata execuţiei pentru întregul set de date.

3. Condiții pentru predarea proiectului

a) Proiectul va fi predat in format electronic (la școală sau prin email) și va conține:

- un document (versiuni Word/Latex și PDF) cu descrierea problemei, a metodelor și a programelor realizate;

- fișierele sursă, programele executabile și fișierele de date utilizate.

b) Documentul va respecta stilurile de redactare folosite în acest document.

c) Termenul de predare este 15 ianuarie 2021.

4. Considerații teoretice

Abordarea inginerească ”clasică”, strictă, a realității este una în special cantitativă, bazată pe modelări matematice exprimate în forme care inspiră exactitatea. Într-o astfel de abordare, aprecierea de ordin calitativ a mărimilor, valorilor, rezultatelor etc. este greu interpretabilă, deoarece acestea au valori stricte, bine precizate.

Într-un context strict, modelele disponibile sunt exacte (de fapt, cât mai exacte posibil, cu aproximări şi supoziţii), iar sistemele de comandă şi control sunt dezvoltate în strânsă legătură cu acestea.

Aplicaţiile curente în inginerie şi mai exact în control automat apelează doar o parte restrânsă a teoriei mulţimilor fuzzy, care se dovedeşte relativ uşor accesibilă.

Aplicaţiile practice în domeniul ingineriei electrice, bazate pe teoria mulţimilor fuzzy, includ un mecanism de evaluare numerică a aprecierilor calitative. Logica fuzzy se referă la faptul că toate lucrurile, faptele sunt caracterizate în grade de relativitate, astfel logica fuzzy transformă logica şi matematica 0-1 la un număr limitat de cazuri între elementele relaţionale. De fapt, matematica de tip fuzzy înseamnă valoarea multiplă sau multivalenţă.

În cazul proiectării sistemelor cu logică fuzzy există 2 metode de inferenţă des utilizate: Mandani şi Sugeno-Takagi. Inferenţa de tip Mandani este metoda principală folosită la prezentarea noţiunilor de bază privind raţionamentul fuzzy. Pentru descrierea metodei de inferenţă de tip Mandani, se consideră o bază de reguli de tip Mandani, cu două variabile de intrare(eroarea şi derivata acesteia) şi o variabilă de ieşire (mărimea de comandă). Regulile de tip Mandani sunt de forma:

Dacă e este E şi de este DE atunci u este U.

O propoziţie fuzzy simplă este o sintagmă de forma ”e este E”, în care ”e” este o variabilă fuzzy, iar ”E” este un termen lingvistic (atribut) ce poate caracteriza variabila. Prin analogie, o propoziţie strictă este descrisă printr-o egalitate e = s. Propoziţiile fuzzy conţin cuvântul ”este”, care ţine loc semnului ”=” din relaţiile stricte.

Valoarea de adevăr a propoziţiei fuzzy este descrisă de funcţia de apartenenţă a mulţimii fuzzy ”E”, adică, pentru o valoare strictă e^* putem preciza în ce măsură ”e^* este E” prin valoarea lui $\mu_E(e^*)$.

O regulă fuzzy descrie legături între mărimi într-o manieră calitativă, folosind propoziţii fuzzy şi/sau propoziţii stricte. Ca o definiţie iniţială, o regulă fuzzy este o construcţie de forma: ”dacă propoziţie-fuzzy atunci propoziţie-fuzzy/propoziţie-strictă”. Prima propoziţie

este denumită *premisă* şi cuprinde evaluări calitative ale mărimilor de intrare. Cealaltă este denumită *consecinţă* şi descrie evaluări calitative sau exacte (cantitative) ale mărimilor de ieşire.

Defuzzificarea este operaţia efectuată pentru obţinerea unei valori ferme pe baza funcţiei de apartenenţă a unei mulţimi fuzzy. Altfel spus, defuzzificarea extrage o informaţie fermă dintr-o informaţie fuzzy. Există mai multe metode de defuzzificare, şi anume:

- Metoda centrului de greutate;
- Metoda înălţimii;
- Metode de maxim: primul maxim, ultimul maxim, mijlocul maximelor.

Se consideră structura clasică de reglare (Figura 1), cu un singur regulator principal, aflat pe calea directă a sistemului. Această structură permite o comparaţie uşoara a performanţelor regulatoarelor fuzzy cu cele are regulatoarelor clasice.

Structura de reglare convenţională este un sistem cu reacţie negativă, considerată unitară pentru o simplificare iniţială. Pe calea de reacţie este preluată mărimea de ieşire a părţii fixe, y(t), care practic este un semnal achiziţionat corespunzător mărimii supuse reglării. Mărimea de ieşire măsurata este comparată cu o mărime de referinţă, r(t), care defineşte evoluţia dorită în timp a sistemului, iar diferenţa celor două reprezintă eroarea de reglare:

$$e(t) = r(t) - y(t) \qquad (1)$$

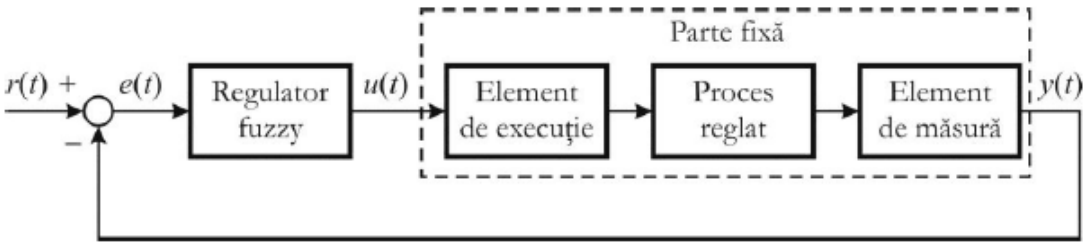

Figura 1. Structura clasică de reglare

Regulatoarele fuzzy sunt una din aplicaţiile importante a teoriei mulţimilor fuzzy. Un sistem de reglare cu regulator fuzzy are principalul avantaj al faptului că permite toleranţă la incertitudine şi imprecizie, aspect demonstrat de numeroase aplicaţii practice.

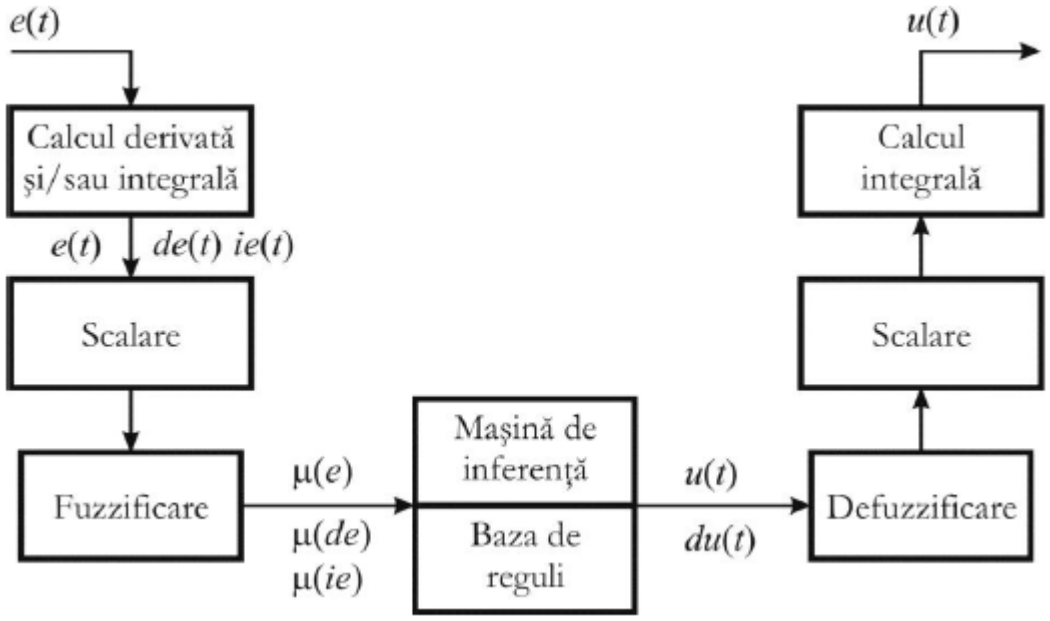

Figura 2. Structura unui regulator fuzzy – caz general

Modulul de calcul al derivatei şi / sau al integralei (MCDI), determină valorile derivatei, respectiv integralei numerice, pentru mărimea de intrare. Derivata numerică a semnalului de intrare e(t) este calculată cu relaţia:

$$de(t) = \frac{e(t) - e(t - T_e)}{T_e} \tag{2}$$

unde Te este perioada de eşantionare cu care se realizează înregistrarea semnalelor.

Legea de reglare a regulatoarelor convenţionale este:

$$u = k_p \cdot e + k_d \cdot \dot{e} \tag{3}$$

Unde $k_d = T_d$ (constanta de timp de derivare). Remarcăm aici faptul că la regulatoarele fuzzy, ca de altfel şi la cele numerice, nu este necesară filtrarea care se introduce la regulatoarele analogie din motive de realizabilitate fizică a acestora.

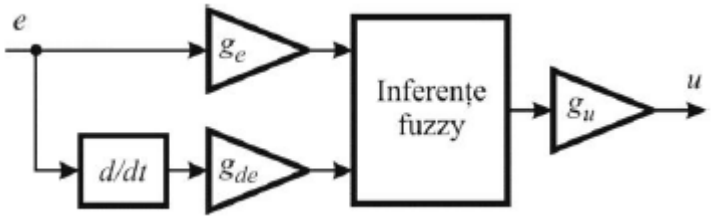

Figura 3. Regulator fuzzy de tip PD

Variabilele lingvistice pentru un regulator fuzzy PD vor fi:

- Eroarea;
- Derivata erorii;
- Mărimea de comandă.

5. Rezolvare proiect

5.1 Implementarea regulatorului fuzzy de tip PD în MATLAB

O regulă tipică din bază are formularea:

Dacă e este E şi de este DE atunci u este U.

Regulatorul fuzzy primeşte ca intrare eroarea şi derivata erorii, iar la ieşire va indica mărimea de comandă, universul de discurs fiind $[-1, 1]$.

Pentru definirea erorii (e), se vor folosi 5 funcţii triunghiulare astfel:

NB – trimf([-1, -1, -0.5])

NS – trimf([-1, -0.5, 0])

ZE – trimf([-0.5, 0, 0.5])

PS – trimf([0, 0.5, 1])

PB – trimf([0.5, 1, 1])

Pentru definirea derivatei erorii (de), se vor folosi 3 funcţii triunghiulare astfel:

N – trimf([-1, -1, 0])

Z – trimf([-1, 0, 1])

B – trimf([0, 1, 1])

Pentru calculul mărimii de comandă, s-a utilizat metoda de defuzzificare denumită mijlocul maximelor. Pentru definirea mărimii de comandă (u), se vor folosi 5 funcţii triunghiulare astfel:

NB – trimf([-1, -1, -0.5])

NS – trimf([-1, -0.5, 0])

ZE – trimf([-0.5, 0, 0.5])

PS – trimf([0, 0.5, 1])

PB – trimf([0.5, 1, 1])

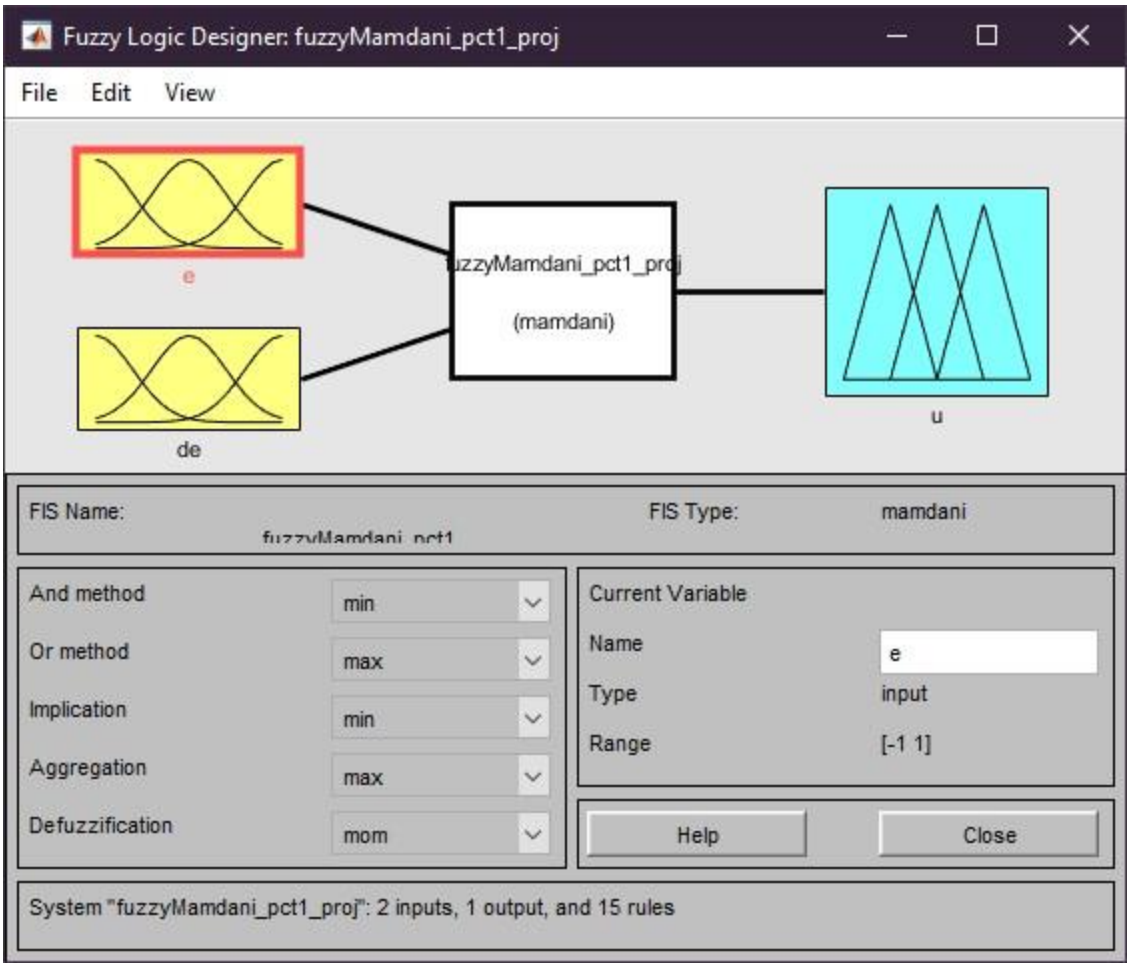

Figura 4. Fuzzy Logic Designer

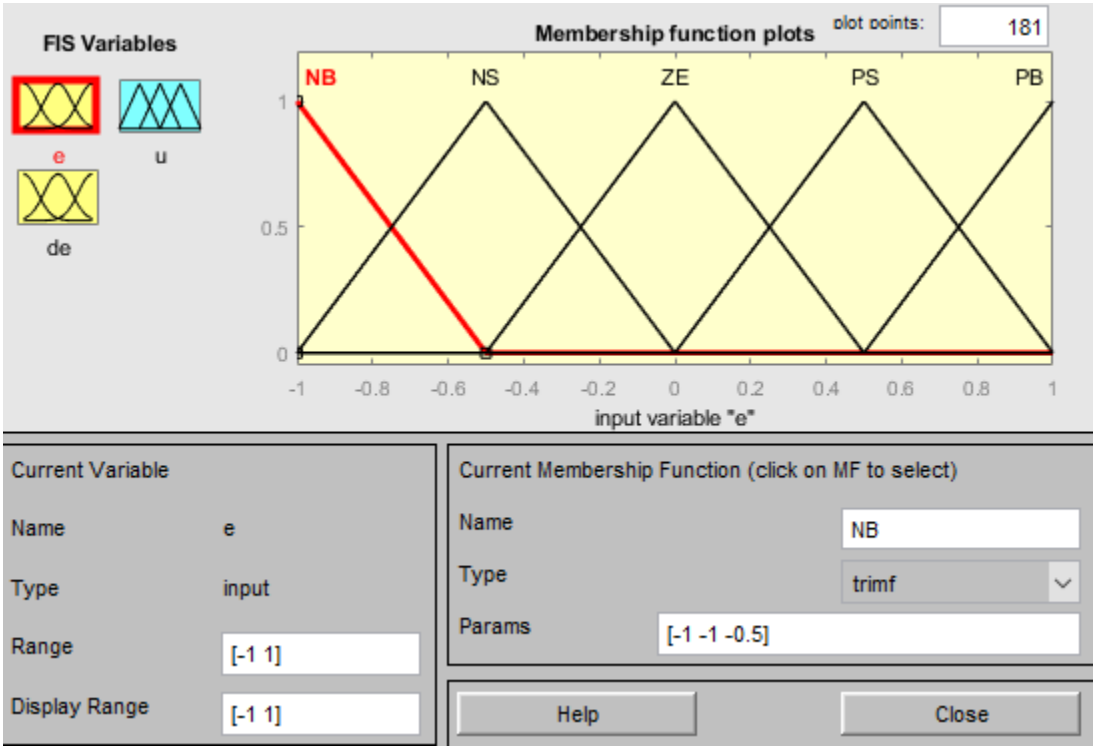

Figura 5. Funcţiile de apartenenţă – eroare

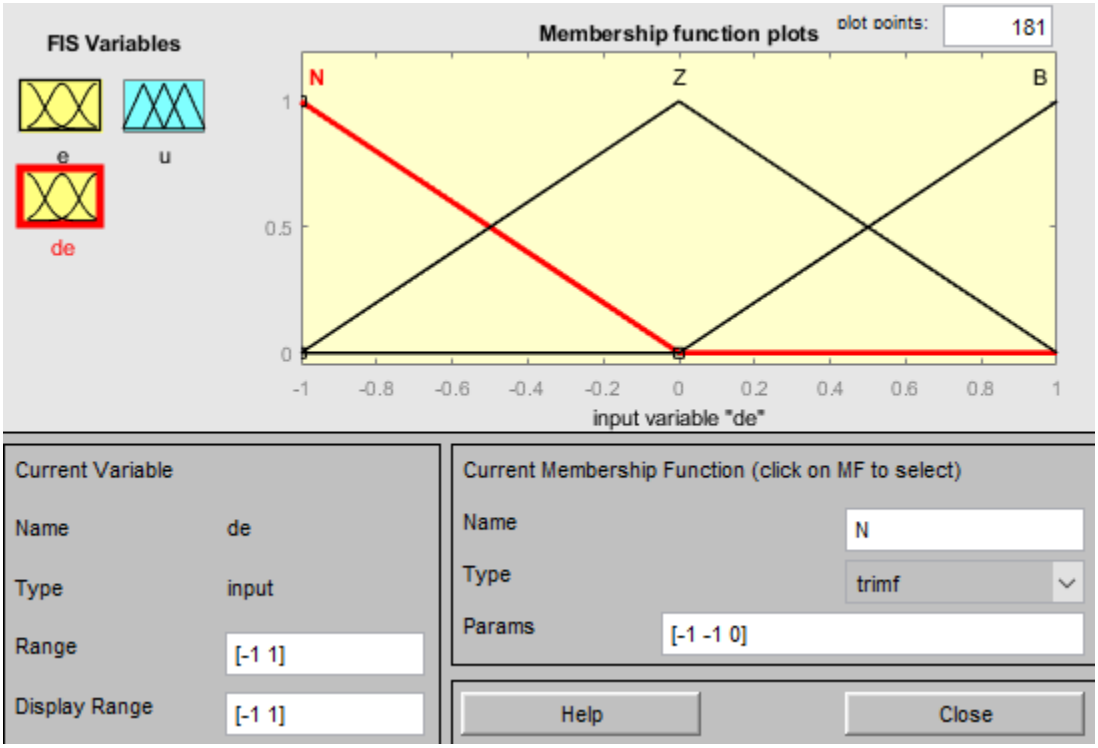

Figura 6. Funcţiile de apartenenţă – derivata erorii

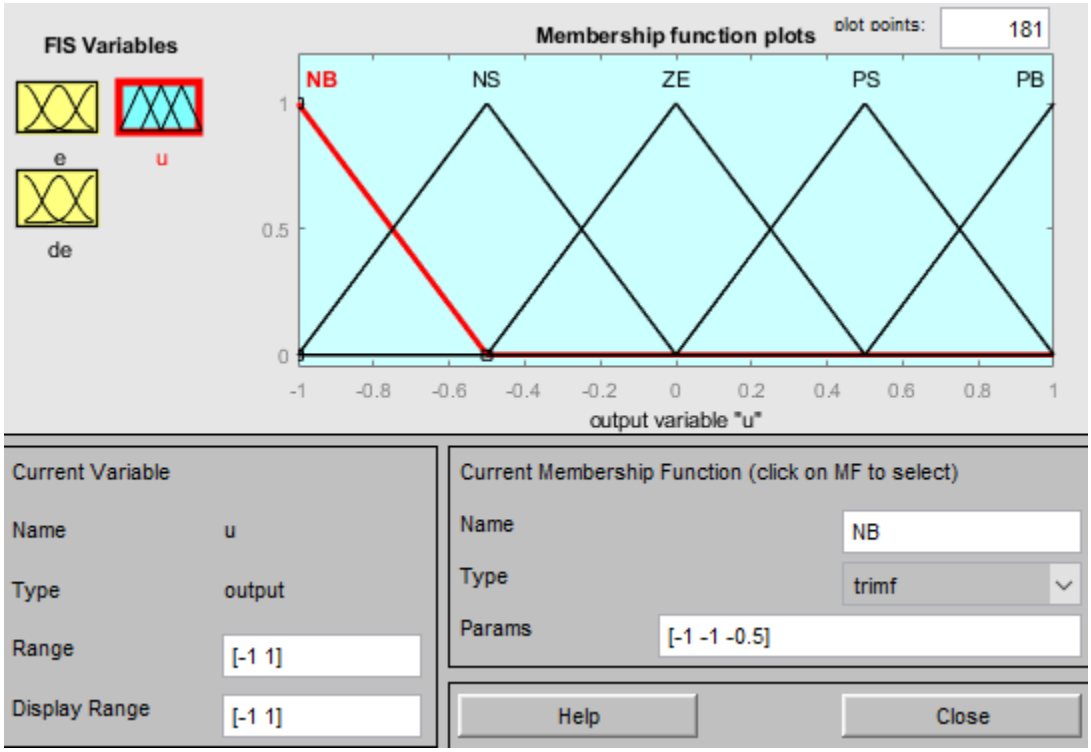

Figura 7. Funcțiile de apartenență – mărimea de comandă

Baza de reguli (Figura 8) este cea impusă în Tabelul 1.

1. If (e is NB) and (de is N) then (u is NB) (1)
2. If (e is NB) and (de is Z) then (u is NB) (1)
3. If (e is NB) and (de is B) then (u is NS) (1)
4. If (e is NS) and (de is N) then (u is NS) (1)
5. If (e is NS) and (de is Z) then (u is NS) (1)
6. If (e is NS) and (de is B) then (u is ZE) (1)
7. If (e is ZE) and (de is N) then (u is ZE) (1)
8. If (e is ZE) and (de is Z) then (u is ZE) (1)
9. If (e is ZE) and (de is B) then (u is ZE) (1)
10. If (e is PS) and (de is N) then (u is ZE) (1)
11. If (e is PS) and (de is Z) then (u is PS) (1)
12. If (e is PS) and (de is B) then (u is PS) (1)
13. If (e is PB) and (de is N) then (u is PS) (1)
14. If (e is PB) and (de is Z) then (u is PB) (1)
15. If (e is PB) and (de is B) then (u is PB) (1)

Figura 8. Baza de reguli a regulatorului fuzzy

5.2. Comparație rezultate program C++ cu MATLAB

Implementarea regulatorului a fost realizată folosind limbajul C++ şi mediul de dezvoltare Visual Studio.

Programul curpinde funcţii de reprezentare a mulţimilor fuzzy, baza de reguli, metodele de fuzificare si defuzificare.

Calculul mărimii de comandă se realizează pe baza funcţiilor triunghiulare ce definesc eroarea, derivata erorii şi mărimea de comandă.

Defuzificarea a fost implementată folosind metoda mijlocul maximelor.

Se pot observa următoarele rezultate:

Tabelul 3. Comparaţie rezultate MATLAB şi C++

Eroarea (e)	**Derivata erorii (de)**	**Mărimea de comandă (u) – Matlab**	**Mărimea de comandă (u) – C++**
0.3	-0.2	0.5	0.5
-0.2	0.3	0	0
0.7	0.4	0.5	0.5
-0.7	-0.6	-0.5	-0.5

5.3. Implementarea prin tabele de decizie Mamdani

Pentru optimizarea implementării numerice a unui sistem de inferenţă fuzzy, o soluţie este aproximarea sistemului de inferenţă printr-un tabel de inferenţă Mamdani. Pentru acest lucru, fiecărei variabile de intrare i se stabileşte o mulţime de bază discretă, păstrând un număr de puncte din mulţimea de bază continuă dată iniţial. Astfel:

a) pentru variabila *eroare* se aleg 11 valori din mulţimea de bază continuă:

$$E = \{-1\ ;\ -0{,}8\ ;\ -0{,}6\ ;\ -0{,}4\ ;\ -0{,}2\ ;\ 0\ ;\ 0{,}2\ ;\ 0{,}4\ ;\ 0{,}6\ ;\ 0{,}8\ ;\ 1\} \tag{6}$$

b) pentru variabila *derivata_erorii* se aleg 5 valori

$$DE = \{-1\ ;\ -0{,}5\ ;\ 0\ ;\ 0{,}5\ ;\ 1\} \tag{7}$$

iar tabela de inferenţă Mamdani va avea structura din Tabelul 2.

Valorile din tabelul de inferenţă Mamdani pentru baza de reguli descrisă iniţial sunt calculate în tabelul 4, folosind programul MATLAB realizat şi prezentat anterior, căruia i s-a adăugat un script care va realiza toate calculele, fără a modifica de fiecare dată valorile intrărilor. Scriptul se numeşte *pctB.m*, şi dă rezultatul din figura 9.

```
u =

  Columns 1 through 10

   -1.0000   -0.9000   -0.5000   -0.5000         0         0         0    0.0000    0.0000    0.5000
   -0.8800   -0.8800   -0.5000   -0.5000   -0.0000   -0.0000   -0.0000    0.2500    0.2500    0.6300
   -1.0000   -0.9000   -0.5000   -0.5000         0         0         0    0.5000    0.5000    0.9000
   -0.6300   -0.6300   -0.2500   -0.2500   -0.0000   -0.0000   -0.0000    0.5000    0.5000    0.8800
   -0.5000   -0.5000   -0.0100    0.0000         0         0         0    0.5000    0.5000    0.9000

  Column 11

    0.5000
    0.6300
    1.0000
    0.8800
    1.0000
```

Figura 9. Rezultat script MATLAB.

Aceste rezultate au fost completate şi în tabelul 4.

Tabelul 4. Tabelul de inferenţă Mamdani

		eroarea										
		-1	-0,8	-0,6	-0,4	-0,2	0	0,2	0,4	0,6	0,8	1
derivata erorii	-1	-1	-0.9	-0.5	-0.5	0	0	0	-0.01	-0.01	0.5	0.5
	-0,5	-0.88	-0.88	-0.5	-0.5	0	0	0	0.25	0.25	0.63	0.63
	0	-1	-0.9	-0.5	-0.5	0	0	0	0.5	0.5	0.9	1
	0,5	-0.63	-0.63	-0.25	-0.25	0	0	0	0.5	0.5	0.88	0.88
	1	-0.5	-0.5	-0.01	-0.01	0	0	0	0.5	0.5	0.9	1

Având tabela de decizie construită, evaluarea inferenţei pentru o valoare strictă a variabilelor de intrare e^* şi de^* se face după cum urmează.

Se consideră că valoarea e^* se află între valori consecutive din tabel, e_i şi e_{i+1}. La fel şi pentru de^*, de_j şi de_{j+1}. În acest caz, în calculul valorii de ieşire u^* ar trebui să fie considerate valorile $u_{i,j}$, $u_{i+1,j}$, $u_{i,j+1}$, $u_{i+1,j+1}$, cu ponderile corespunzătoare w. Restul elementelor din tabel nu vor intra în calcul.

Este intuitiv să considerăm că, cu cât e^* este mai aproape de e_i, cu atât valoarea corespunzătoare lui u_i trebuie să contribuie mai mult în calculul lui u^*. Apropierea lui e^* de e_i este direct proporţională cu depărtarea lui e^* de e_{i+1}.

Ponderile celor 4 valori se vor calcula astfel:

$$w_{i,j} = \frac{|e^* - e_{i+1}||de^* - de_{j+1}|}{|e_{i+1} - e_i||de_{j+1} - de_j|}$$

$$w_{i+1,j} = \frac{|e^* - e_i||de^* - de_{j+1}|}{|e_{i+1} - e_i||de_{j+1} - de_j|} \qquad (8)$$

$$w_{i,j+1} = \frac{|e^* - e_{i+1}||de^* - de_j|}{|e_{i+1} - e_i||de_{j+1} - de_j|}$$

$$w_{i+1,j+1} = \frac{|e^* - e_i|\left|de^* - de_j\right|}{|e_{i+1} - e_i|\left|de_{j+1} - de_j\right|}$$

Având cele 4 ponderi, se calculează valoarea de ieșire cu relația:

$$u^* = w_{i,j}u_{i,j} + w_{i+1,j}u_{i+1,j} + w_{i,j+1}u_{i,j+1} + w_{i+1,j+1}u_{i+1,j+1} \quad (9)$$

Ca o observație importantă: suma ponderilor trebuie să fie întotdeauna egală cu 1.

5.4. Comparație rezultate programe C++ și MATLAB

Tabelul 5. Comparație rezultate C++ (A), C++ (C) și MATLAB

Eroarea (e)	**Derivata erorii (de)**	**Mărimea de comandă (u) – C++ (A)**	**Mărimea de comandă (u) – C++ (C)**	**Mărimea de comandă (u) – Matlab**
0.3	-0.2	0.5	0.2	0.5
-0.2	0.3	0	0	0
0.7	0.4	0.5	0.692	0.5
-0.7	-0.6	-0.5	-0.692	-0.5

5.5. Rezolvare cerința (D)

În sistemele de reglare numerice, derivata mărimii de eroare este calculată din valorile înregistrate ale erorii şi pe baza perioadei de eşantionare. În varianta cea mai simplă, relaţia de calcul a derivatei este:

$$de[k] = \frac{e[k] - e[k-1]}{T_e} \tag{10}$$

unde $de[k]$ este valoarea derivatei erorii corespunzătoare iteraţiei curente, $e[k]$ şi $e[k-1]$ sunt valorile mărimii de eroare de la iteraţia curentă, respectiv de la iteraţia anterioară, iar T_e este perioada de eşantionare (intervalul de timp dintre două iteraţii consecutive).

Pentru calculul derivatei numerice a erorii, considerăm perioada de eşantionare $T_e = 0{,}1$.

În fişierul ”date.txt” se găseşte un set de valori ”înregistrate” ale mărimii de *eroare* dintr-un sistem automat. Din cauza perturbaţiile ce pot apărea în orice sistem, există câteva valori înregistrare ale mărimii de eroare ce nu se încadrează în universul de discurs. Pentru acestea, a fost nevoie de o aproximare la cea mai apropiată valoare a universului de discurs.

Valorile derivatei erorii pentru fiecare eşantion înregistrat au fost calculate cu ecuaţia (10).

În urma calculării tuturor valorilor mărimii de comandă prin cele două programe realizate la punctele A şi C ale proiectului, au rezultat timpi de calcul diferiţi. Aceştia sunt prezentaţi în tabelul 6, pentru modul Release al fiecărui program.

Tabelul 6 Intervalul de timp necesar calculului

Timp program (A) **(secunde)**	**Timp program (C)** **(secunde)**
0.238147	0.0093849

Se poate observa că este o diferenţă între destul de considerabilă între cele două programe din punct de vedere al timpului de rulare. Astfel, metoda optimă, mai rapidă, pentru calcularea valorilor semnalului de comandă pentru un număr mare de valori de intrare este cea de calcul folosind tabela de decizie Mamdani.

Anexa 1 Program realizat în C++ (A)

main.cpp

```
#include <iostream>
#include <algorithm>
#include <stdio.h>
#include <conio.h>


#include <vector>


double triunghi(double e, double a, double b, double c)
{
	if (e < a)
		return 0;

	if (e >= b && e <= c)
		return (c - e) / (c - b);

	if (e >= a && e <= b)
		return (e - a) / (b - a);

	if (e > c)
		return 0;
}

double triunghi_stanga(double e, double a, double b)
{
	if (e < a)
		return 0;
	if (e > b)
```

```
		return 0;


	if (e >= a && e <= b)
		return (b - e) / (b - a);
}


double triunghi_dreapta(double e, double a, double b)
{
	if (e < a)
		return 0;
	if (e > b)
		return 0;


	if (e >= a && e <= b)
		return (e - a) / (b - a);
}


void fuzzificare_eroare(double e, std::vector<double> &me)
{
	//vectorul me[] -> cuprinde rezultatele in urma apelarii functiilor pentru intrarea "eroare" (functiile de apartenenta)
	me.push_back(triunghi_stanga(e, -1.0, -0.5));	//NB
	me.push_back( triunghi(e, -1.0, -0.5, 0.0) );	//NS
	me.push_back( triunghi(e, -0.5, 0.0, 0.5) );	//ZE
	me.push_back( triunghi(e, 0.0, 0.5, 1.0) );	//PS

	me.push_back(triunghi_dreapta(e, 0.5, 1.0));	//PB
}
void fuzzificare_derivataEroare(double de, std::vector<double> &mde)
{
```

```
	//vectorul mde[] -> cuprinde rezultatele in urma apelarii functiilor pentru intrarea "derr_eroare" (functiile de apartenenta)

	mde.push_back(triunghi_stanga(de, -1.0, 0.0));	//N

	mde.push_back( triunghi(de, -1.0, 0.0, 1.0) );	//Z

	mde.push_back(triunghi_dreapta(de, 0.0, 1.0));	//B

}



void inferenta_Mamdani(std::vector<double> me, std::vector<double> mde, std::vector <std::vector <double> > &mu)

{

	//se aplica inferenta Mamdani (se determina minimul dintre functiile de activare ale intrarilor "eroare" si "derr_eroare"

	//rezultatele in urma inferentei se retin in vectorul mu[]

	for (size_t i = 0; i < 3; ++i)

		for (size_t j = 0; j < 5; ++j)

			mu[i][j] = std::min (me[j], mde[i]);



	//randuri -> i -> derr_err -> NE ZE PO

	//coloane -> j -> eroare -> NB NS ZE PS PB

	std::cout << "Tabela de inferenta: " << std::endl;

	for (int i = 0; i < 3; ++i)

	{

		for (int j = 0; j < 5; ++j)

		{

			std::cout << mu[i][j] << "\t";

		}

		std::cout << std::endl;

	}

	std::cout << std::endl;



}
```

```
double max_fctApartenenta(std::vector <std::vector <double> > mu, char reguli[3][5], char
cuvant)

{

	//se calculeaza valoarea maxima a fiecarei functii de apartenenta

	double max = 0.0;


	for (int i = 0; i < 3; ++i)

		for (int j = 0; j < 5; ++j)

			if (reguli[i][j] == cuvant)

				max = std::max(max, mu[i][j]);


	return max;

}


double primulMaxim(double a, double b, double c, double max)

{

	double firstMax = (b-a)*(max) + a;


	if (a == b)

		firstMax = a;


	return firstMax;

}


double ultimulMaxim(double a, double b, double c, double max)

{

	double lastMax = c - (c - b)*(max);


	if (b == c)
```

```
                lastMax = c;


        return lastMax;
}


void main()
{
        //baza de reguli, unde:
        //n -> NB; m -> NS; z -> ZE; p -> PS; b -> PB;
        char reguli[3][5] = { 'n', 'm', 'z', 'z', 'p',
                                                                                'n',
'm', 'z', 'p', 'b',
                                                                                'm',
'z', 'z', 'p', 'b' };


        std::cout << "Tabela de reguli: " << std::endl;
        for (int i = 0; i < 3; ++i)
        {
                for (int j = 0; j < 5; ++j)
                {
                        std::cout << reguli[i][j] << "\t";
                }
                std::cout << std::endl;
        }
        std::cout << std::endl;


        while (true)
        {
                double e, de;
                std::cout << "Introduceti valoarea erorii: e = "; std::cin >> e;
                if (e < -1 || e > 1)
```

```
		{
			std::cout << "Universul de discurs pentru eroare este [-1; 1]. 
Valoarea introdusa nu apartine acestuia." << std::endl;
			std::cout << "Introduceti valoarea erorii: e = "; std::cin >> e;
		}

		std::cout << "Introduceti valoarea derivatei erorii: de = "; std::cin >> de;
		if (de < -1 || de > 1)
		{
			std::cout << "Universul de discurs pentru derivata erorii este [-1; 
1]. Valoarea introdusa nu apartine acestuia." << std::endl;
			std::cout << "Introduceti valoarea derivatei erorii: de = "; std::cin 
>> de;
		}



		std::vector<double> me;
		me.reserve(5);
		std::vector<double> mde;
		mde.reserve(3);

		//apelare fct fuzzificare pentru a determina functiile de apartenenta
		fuzzificare_eroare(e, me);
		fuzzificare_derivataEroare(de, mde);

		std::cout << e << " apartine  " << me[0] << "  NB " << std::endl;
		std::cout << e << " apartine  " << me[1] << "  NS " << std::endl;
		std::cout << e << " apartine  " << me[2] << "  ZE " << std::endl;
		std::cout << e << " apartine  " << me[3] << "  PS " << std::endl;
		std::cout << e << " apartine  " << me[4] << "  PB " << std::endl;
		std::cout << std::endl;
```

```
std::cout << de << " apartine  " << mde[0] << " N " << std::endl;
std::cout << de << " apartine  " << mde[1] << " Z " << std::endl;
std::cout << de << " apartine  " << mde[2] << " P " << std::endl;
std::cout << std::endl;

std::vector <std::vector <double> > mu;
mu.resize(3);
for (size_t i = 0; i < mu.size(); ++i)
	mu[i].resize(5);

//tabela de inferenta
inferenta_Mamdani(me, mde, mu);

std::vector <double> max_apartenenta;
max_apartenenta.reserve(5);
max_apartenenta.push_back(max_fctApartenenta(mu, reguli, 'n'));
max_apartenenta.push_back(max_fctApartenenta(mu, reguli, 'm'));
max_apartenenta.push_back(max_fctApartenenta(mu, reguli, 'z'));
max_apartenenta.push_back(max_fctApartenenta(mu, reguli, 'p'));
max_apartenenta.push_back(max_fctApartenenta(mu, reguli, 'b'));

double max = 0.0;
for (size_t i = 0; i < 5; ++i)
{
	max = std::max(max, max_apartenenta[i]);
}

double som = 0.0;
if (max == max_apartenenta[0])
```

```
		{
			som = primulMaxim(-1.0, -1.0, -0.5, max);
			max_apartenenta[0] = 10; //label
		}
		else if (max == max_apartenenta[1])
		{
			som = primulMaxim(-1.0, -0.5, 0.0, max);
			max_apartenenta[1] = 10;
		}
		else if (max == max_apartenenta[2])
		{
			som = primulMaxim(-0.5, 0.0, 0.5, max);
			max_apartenenta[2] = 10;
		}
		else if (max == max_apartenenta[3])
		{
			som = primulMaxim(0.0, 0.5, 1.0, max);
			max_apartenenta[3] = 10;
		}
		else if (max == max_apartenenta[4])
		{
			som = primulMaxim(0.5, 1.0, 1.0, max);
			max_apartenenta[4] = 10;
		}



		double lom = 0.0;
		if (max == max_apartenenta[0])
			lom = ultimulMaxim(-1.0, -1.0, -0.5, max);
		else if (max == max_apartenenta[1])
```

```
                        lom = ultimulMaxim(-1.0, -0.5, 0.0, max);
                else if (max == max_apartenenta[2])
                        lom = ultimulMaxim(-0.5, 0.0, 0.5, max);
                else if (max == max_apartenenta[3])
                        lom = ultimulMaxim(0.0, 0.5, 1.0, max);
                else if (max == max_apartenenta[4])
                        lom = ultimulMaxim(0.5, 1.0, 1.0, max);
                else
                {
                        for (size_t i = 0; i < max_apartenenta.size(); ++i)
                        {
                                if (max_apartenenta[i] == 10)
                                {
                                        if (i == 0) lom = ultimulMaxim(-1.0, -1.0, -0.5, max);
                                        if (i == 1) lom = ultimulMaxim(-1.0, -0.5, 0.0, max);
                                        if (i == 2) lom = ultimulMaxim(-0.5, 0.0, 0.5, max);
                                        if (i == 3) lom = ultimulMaxim(0.0, 0.5, 1.0, max);
                                        if (i == 4) lom = ultimulMaxim(0.5, 1.0, 1.0, max);
                                }
                        }
                }

                double mom = (som + lom) / 2;
                std::cout << "Defuzzificare prin metoda MIJLOCUL MAXIMELOR (MOM): " << mom <<
"\n\n" << std::endl;
        }
}
```

Anexa 2 Cod Matlab (B)

pctB.m

```
fis = readfis("fuzzyMamdani_pct1_proj.fis");


u=zeros(5,11);
i=1;
for e = -1.0:0.2:1.0
    j=1;
    for de = -1.0:0.5:1.0
        inputs = [e de];
        u(j,i) = evalfis(fis,inputs);
        j=j+1;
    end
    i=i+1;
end
```

Anexa 3 Program realizat în C++ (C)

main.cpp

```
#include <iostream>
#include <algorithm>
#include <stdio.h>
#include <conio.h>

#include <vector>

void valori_eroare(std::vector <double> &me)
{
	me.push_back(-1);
	me.push_back(-0.8);
	me.push_back(-0.6);
	me.push_back(-0.4);
	me.push_back(-0.2);
	me.push_back(0);
	me.push_back(0.2);
	me.push_back(0.4);
	me.push_back(0.6);
	me.push_back(0.8);
	me.push_back(1);
}

void valori_derErr(std::vector <double> &mde)
{
	mde.push_back(-1);
	mde.push_back(-0.5);
```

```
	mde.push_back(0);

	mde.push_back(0.5);

	mde.push_back(1);

}


void valoriActive_comanda(double e, double de, std::vector<double> me,
std::vector<double> mde, double reguli[5][11], std::vector<double> &mu)

{

	for (size_t i = 0; i < mde.size(); ++i)

	{

		for (size_t j = 0; j < me.size(); ++j)

		{

			if ((e > me[j] && e < me[j + 1]) && (de > mde[i] && de < mde[i +
1]))

			{

				mu.push_back(reguli[i][j]);

				mu.push_back(reguli[i][j + 1]);

				mu.push_back(reguli[i + 1][j]);

				mu.push_back(reguli[i + 1][j + 1]);


				std::cout << "Casute active: " << std::endl;

				std::cout << mu[0] << "\t" << mu[1] << std::endl;

				std::cout << mu[2] << "\t" << mu[3] << std::endl;

			}


			if (e == me[j] && (de > mde[i] && de < mde[i + 1]))

			{

				mu.push_back(reguli[i][j]);

				mu.push_back(reguli[i + 1][j]);
```

```
                        std::cout << "Casute active: " << std::endl;
                        std::cout << mu[0] << "\n" << mu[1] << std::endl;
                }

                if ((e > me[j] && e < me[j + 1]) && de == mde[i])
                {
                        mu.push_back(reguli[i][j]);
                        mu.push_back(reguli[i][j + 1]);

                        std::cout << "Casute active: " << std::endl;
                        std::cout << mu[0] << "\t" << mu[1] << std::endl;
                }

                if (e == me[j] && de == mde[i])
                {
                        mu.push_back(reguli[i][j]);

                        std::cout << "Casute active: " << std::endl;
                        std::cout << mu[0] << std::endl;
                }
        }
    }
}

void main()
{
        //baza de reguli
        double reguli[5][11] = { -1, -0.9, -0.5, -0.5, 0, 0, 0, -0.01, -0.01, 0.5, 0.5,

        -0.88, -0.88, -0.5, -0.5, 0, 0, 0, 0.25, 0.25, 0.63, 0.63,
```

```
	-1, -0.9, -0.5, -0.5, 0, 0, 0, 0.5, 0.5, 0.9, 1,

	-0.63, -0.63, -0.25, -0.25, 0, 0, 0, 0.5, 0.5, 0.88, 0.88,

	-0.5, -0.5, -0.01, -0.01, 0, 0, 0, 0.5, 0.5, 0.9, 1 }; //y


	std::cout << "Tabela de reguli: " << std::endl;
	for (int i = 0; i < 5; ++i)
	{
		for (int j = 0; j < 11; ++j)
		{
			std::cout << reguli[i][j] << "\t";
		}
		std::cout << std::endl;
	}
	std::cout << std::endl;


	while (true)
	{
		double e, de;
		std::cout << "Introduceti valoarea erorii: e = "; std::cin >> e;
		if (e < -1 || e > 1)
		{
			std::cout << "Universul de discurs pentru eroare este [-1; 1].
Valoarea introdusa nu apartine acestuia." << std::endl;
			std::cout << "Introduceti valoarea erorii: e = "; std::cin >> e;
		}

		std::cout << "Introduceti valoarea derivatei erorii: de = "; std::cin >>
de;
```

```
		if (de < -1 || de > 1)

		{

			std::cout << "Universul de discurs pentru derivata erorii este [-
1; 1]. Valoarea introdusa nu apartine acestuia." << std::endl;

			std::cout << "Introduceti valoarea derivatei erorii: de = ";
std::cin >> de;

		}


		std::vector<double> me;

		me.reserve(11);

		valori_eroare(me);


		std::vector<double> mde;

		mde.reserve(5);

		valori_derErr(mde);


		std::vector<double> mu;

		mu.reserve(4);

		valoriActive_comanda(e, de, me, mde, reguli, mu);


		std::vector<double> ponderi_w;

		ponderi_w.reserve(4);

		for (size_t i = 0; i < mde.size(); ++i)

		{

			for (size_t j = 0; j < me.size(); ++j)

			{

				if ((e > me[j] && e < me[j + 1]) && (de > mde[i] && de <
mde[i + 1]))

				{

					double w0 = (abs(e - me[j + 1])*abs(de - mde[i +
1])) / (abs(me[j] - me[j + 1])*abs(mde[i] - mde[i + 1]));
```

```
                                    double w1 = (abs(e - me[j])*abs(de - mde[i + 1])) /
(abs(me[j] - me[j + 1])*abs(mde[i] - mde[i + 1]));

                                    double w2 = (abs(e - me[j + 1])*abs(de - mde[i])) /
(abs(me[j] - me[j + 1])*abs(mde[i] - mde[i + 1]));

                                    double w3 = (abs(e - me[j])*abs(de - mde[i])) /
(abs(me[j] - me[j + 1])*abs(mde[i] - mde[i + 1]));


                                    ponderi_w.push_back(w0);

                                    ponderi_w.push_back(w1);

                                    ponderi_w.push_back(w2);

                                    ponderi_w.push_back(w3);


                                    std::cout << "Ponderile corespunzatoare casutelor
active: " << std::endl;

                                    std::cout << ponderi_w[0] << "\t" << ponderi_w[1] <<
std::endl;

                                    std::cout << ponderi_w[2] << "\t" << ponderi_w[3] <<
std::endl;


                                    break;

                            }


                            if (e == me[j] && (de > mde[i] && de < mde[i + 1]))

                            {

                                    if (e == 1)

                                    {

                                            double w0 = (abs(e - me[j - 1])*abs(de - mde[i
+ 1])) / (0.2 * abs(mde[i] - mde[i + 1]));

                                            double w1 = (abs(e - me[j - 1])*abs(de -
mde[i])) / (0.2 * abs(mde[i] - mde[i + 1]));


                                            ponderi_w.push_back(w0);

                                            ponderi_w.push_back(w1);
```

```
                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;
                                        std::cout << ponderi_w[0] << "\n" <<
ponderi_w[1] << std::endl;


                                        break;
                                }
                                else
                                {
                                        double w0 = (abs(e - me[j + 1])*abs(de - mde[i
+ 1])) / (0.2 * abs(mde[i] - mde[i + 1]));
                                        double w1 = (abs(e - me[j + 1])*abs(de -
mde[i])) / (0.2 * abs(mde[i] - mde[i + 1]));


                                        ponderi_w.push_back(w0);
                                        ponderi_w.push_back(w1);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;
                                        std::cout << ponderi_w[0] << "\n" <<
ponderi_w[1] << std::endl;


                                        break;
                                }
                        }


                        if ((e > me[j] && e < me[j + 1]) && de == mde[i])
                        {
                                if (de == 1)
                                {
```

```
                                        double w0 = (abs(e - me[j + 1])*abs(de - mde[i
- 1])) / (abs(me[j] - me[j + 1]) * 0.5);

                                        double w1 = (abs(e - me[j])*abs(de - mde[i -
1])) / (abs(me[j] - me[j + 1]) * 0.5);


                                        ponderi_w.push_back(w0);

                                        ponderi_w.push_back(w1);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << "\t" <<
ponderi_w[1] << std::endl;


                                        break;

                                    }


                                    else

                                    {

                                        double w0 = (abs(e - me[j + 1])*abs(de - mde[i
+ 1])) / (abs(me[j] - me[j + 1]) * 0.5);

                                        double w1 = (abs(e - me[j])*abs(de - mde[i +
1])) / (abs(me[j] - me[j + 1]) * 0.5);


                                        ponderi_w.push_back(w0);

                                        ponderi_w.push_back(w1);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << "\t" <<
ponderi_w[1] << std::endl;


                                        break;

                                    }
```

```
                        }


                        if (e == me[j] && de == mde[i])

                        {

                                if (e == 1 && de == 1)

                                {

                                        double w0 = (abs(e - me[j - 1])*abs(de - mde[i
- 1])) / (0.2 * 0.5);

                                        ponderi_w.push_back(w0);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << std::endl;


                                        break;


                                }

                                if (e == 1 && de == mde[i])

                                {

                                        double w0 = (abs(e - me[j - 1])*abs(de - mde[i
+ 1])) / (0.2 * 0.5);

                                        ponderi_w.push_back(w0);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << std::endl;

                                        break;


                                }

                                if (de == 1 && e == me[j])

                                {
```

```
                                        double w0 = (abs(e - me[j + 1])*abs(de - mde[i
- 1])) / (0.2 * 0.5);

                                        ponderi_w.push_back(w0);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << std::endl;


                                        break;
                                }
                                else
                                {
                                        double w0 = (abs(e - me[j + 1])*abs(de - mde[i
+ 1])) / (0.2 * 0.5);

                                        ponderi_w.push_back(w0);


                                        std::cout << "Ponderile corespunzatoare
casutelor active: " << std::endl;

                                        std::cout << ponderi_w[0] << std::endl;


                                        break;
                                }
                        }
                }
        }


        double verificare_ponderi = 0.0;
        double u = 0.0;
        for (size_t i = 0; i < ponderi_w.size(); ++i)
        {
                verificare_ponderi += ponderi_w[i];
```

```
                        u += double(ponderi_w[i]) * double(mu[i]);

                }

                if (verificare_ponderi != 1.0)

                        std::cout << "Programul nu calculeaza corect ponderile! Suma lor
trebuie sa fie 1." << std::endl;



                std::cout << "u = " << u << "\n\n" << std::endl;

        }

}
```

Anexa 4 Program realizat în C++ (D)

Funcţia de citire a valorilor din fişier şi funcţia de calculare a derivatei erorii.

SCF_COD.cpp

```
std::vector<double> extragereValoriEroare_fisier()
{
	std::ifstream fin("date3.txt");

	std::vector<double> date; date.reserve(10003);
	double e = 0;

	while (!fin.eof())
	{
		fin >> e;

		if (e > 1.0)
			e = 1.00;

		date.push_back(e);
	}
	fin.close();

	return date;
}

std::vector<double> calculareDerErr(std::vector<double> eroare)
{
	std::vector<double> derErr;
	derErr.reserve(eroare.size());
```

```
	derErr.push_back(0.0); //pentru e[0] - e[-1]


	double t_e = 0.1;
	for (int i = 1; i < eroare.size(); ++i)
	{
		double esantion = (eroare[i] - eroare[i - 1]) / t_e;


		derErr.push_back(esantion);
	}


	return derErr;
}
```