Proiect

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

**Îndrumător:**  **Student:**

Prof. Cristian Boldișor Patrania Bogdan Andrei

Grupa 4MF132

2023-2024

Cuprins

[1. Tema proiectului 2](#_Toc153036448)

[2. Indicații, recomandări, cerințe suplimentare 6](#_Toc153036449)

[3. Condiții pentru predarea proiectului 7](#_Toc153036450)

[4. Considerații teoretice 8](#_Toc153036451)

[5. Rezolvare proiect 12](#_Toc153036452)

[5.1 Implementarea regulatorului fuzzy de tip PD în MATLAB 12](#_Toc153036453)

[5.2. Comparație rezultate program C++ cu MATLAB 15](#_Toc153036454)

[5.3. Implementarea prin tabele de decizie Mamdani 16](#_Toc153036455)

[5.4. Comparație rezultate programe C++ și MATLAB 19](#_Toc153036456)

[5.5. Rezolvare proiect (D) 20](#_Toc153036457)

[Anexa 1 Program realizat în C++ (A) 21](#_Toc153036458)

[Anexa 2 Cod Matlab (B) 30](#_Toc153036459)

[Anexa 3 Program realizat în C++ (C) 31](#_Toc153036460)

[Anexa 4 Program realizat în C++ (D) 42](#_Toc153036461)

# 1. Tema proiectului

**A)**

Fie un sistem de inferență fuzzy de tip Mamdani cu două variabile de intrare – *eroarea* și *derivata\_erorii* – și o variabilă de ieșire – *comanda*. Termenii lingvistici ai variabilelor de intrare sunt descriși în figura 1, iar baza de reguli este descrisă în tabelul 1. Termenii lingvistici ai variabilei de ieșire sunt identici cu cei ai variabilei *eroare* (figura 1a). Pentru fiecare din cele trei variabile mulțimea de bază este intervalul .

Metoda de inferență propusă este de tip Mamdami, adică: a) operațiile de intersecție și implicație fuzzy sunt realizate cu operatorul *min*, iar operațiile de reuniune cu operatorul *max*; b) evaluarea inferențelor conține o etapă de defuzzyficare (metoda de defuzzyficare nu este impusă).

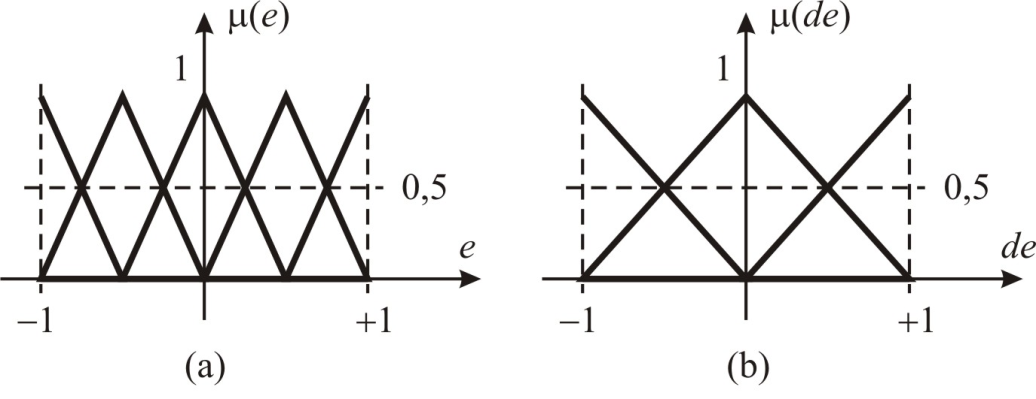


Fig. 1

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 |

**Realizați un program pentru implementarea sistemului de inferență considerat.**

Pentru verificare, comparați rezultatele obținute prin program cu cele obținute în Matlab.

Observație. Sistemul de inferență considerat este specific unui regulator fuzzy de tip proporțional-derivativ (PD) utilizat în structura convențională de reglare automată .

**B)**

Pentru optimizarea implementării numerice a unui sistem de inferență fuzzy, o soluție este aproximarea sistemului de inferență printr-un tabel de inferență Mamdani. Pentru acest lucru, fiecărei variabile de intrare i se stabilește o mulțime de bază discretă, păstrând un număr de puncte din mulțimea de bază continuă dată inițial. Astfel:

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

|  |  |  |
| --- | --- | --- |
|  |  |  |

b) pentru variabila *derivata\_erorii* se aleg 5 valori

|  |  |  |
| --- | --- | --- |
|  |  |  |

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

**Se cere să se calculeze valorile din tabelul de inferență 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 inferență, implementarea numerică a algoritmului de reglare proporțional-derivativ (din programul de la punctul A) se schimbă.

**Realizați un al doilea program care implementează algoritmul de inferență fuzzy pe tabela de decizie determinată anterior.**

Pentru verificare, testați rezultatele obținute cu cele două programe, pentru câteva perechi de valori (e, de) din mulțimea discretă . Dacă implementarea este corectă, valoarea mărimii de ieșire *comanda* obținută pentru o anumită pereche de valori cu primul program și cea obținută cu al doilea sunt egale.

**D)**

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

unde este valoarea derivatei erorii corespunzătoare iterației curente, și sunt valorile mărimii de eroare de la iterația curentă, respectiv de la iterația anterioară, iar este perioada de eșantionare (intervalul de timp dintre două iterații consecutive).

Pentru calculul derivatei numerice a erorii, considerăm perioada de eșantionare .

În fișierul ”date.txt” se găsește un set de valori ”înregistrate” ale mărimii de *eroare* dintr-un sistem automat. **Calculați valorile *derivatei\_erorii* pentru fiecare eșantion înregistrat**. Apoi, având setul de perechi de valori formate din valori ale *erorii* și valorile calculate ale *derivatei*, **calculați toate valorile mărimii de *comandă* separat prin cele două programe**. **Completați programele realizate astfel încât, să măsoare intervalul de timp necesar calculului valorilor semnalului de comandă pentru toate eșantioanele din fișierul de date.**

# 2. Indicații, recomandări, cerințe suplimentare

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

b) Pentru evaluarea inferențelor fuzzy pe baza tabelei de inferență Mamdani, este necesară definirea unei funcții care să indice distanța de la o pereche de valori numerice reale la o combinație de valori din mulțimea de bază discretă. Practic, este necesară o funcție aleasă convenabil care descrie distanța dintre două puncte în plan.

c) Programele modificate pentru setul de date de la punctul D trebuie să cronometreze durata execuției 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 și supoziții), iar sistemele de comandă și control sunt dezvoltate în strânsă legătură cu acestea.

Aplicațiile curente în inginerie și mai exact în control automat apelează doar o parte restrânsă a teoriei mulțimilor fuzzy, care se dovedește relativ ușor accesibilă.

Aplicațiile practice în domeniul ingineriei electrice, bazate pe teoria mulțimilor 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 și matematica 0-1 la un număr limitat de cazuri între elementele relaționale. De fapt, matematica de tip fuzzy înseamnă valoarea multiplă sau multivalență.

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

*Dacă e este E și de este DE atunci u este U.*

O propoziție 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 propoziție strictă este descrisă printr-o egalitate e = s. Propozițiile fuzzy conțin cuvântul ”este”, care ține loc semnului ”=” din relațiile stricte.

Valoarea de adevăr a propoziției fuzzy este descrisă de funcția de apartenență a mulțimii fuzzy ”E”, adică, pentru o valoare strictă e\* putem preciza în ce măsură ”e\* este E” prin valoarea lui .

O regulă fuzzy descrie legături între mărimi într-o manieră calitativă, folosind propoziții fuzzy și/sau propoziții stricte. Ca o definiție inițială, o regulă fuzzy este o construcție de forma: ”dacă propoziție-fuzzy atunci propoziție-fuzzy/propoziție-strictă”. Prima propoziție este denumită *premisă* și cuprinde evaluări calitative ale mărimilor de intrare. Cealaltă este denumită *consecință* și descrie evaluări calitative sau exacte (cantitative) ale mărimilor de ieșire.

Defuzzificarea este operația efectuată pentru obținerea unei valori ferme pe baza funcției de apartenență a unei mulțimi fuzzy. Altfel spus, defuzzificarea extrage o informație fermă dintr-o informație fuzzy. Există mai multe metode de defuzzificare, și anume:

* Metoda centrului de greutate;
* Metoda înălțimii;
* 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 comparație ușoara a performanțelor regulatoarelor fuzzy cu cele are regulatoarelor clasice.

Structura de reglare convențională este un sistem cu reacție negativă, considerată unitară pentru o simplificare inițială. Pe calea de reacție este preluată mărimea de ieșire a părții fixe, y(t), care practic este un semnal achiziționat corespunzător mărimii supuse reglării. Mărimea de ieșire măsurata este comparată cu o mărime de referință, r(t), care definește evoluția dorită în timp a sistemului, iar diferența celor două reprezintă eroarea de reglare:

|  |  |
| --- | --- |
| e(t) = r(t) – y(t) | (1) |

|  |
| --- |
|  |
| **Figura 1.** Structura clasică de reglare |

Regulatoarele fuzzy sunt una din aplicațiile importante a teoriei mulțimilor fuzzy. Un sistem de reglare cu regulator fuzzy are principalul avantaj al faptului că permite toleranță la incertitudine și imprecizie, aspect demonstrat de numeroase aplicații practice.

|  |
| --- |
|  |
| **Figura 2.** Structura unui regulator fuzzy – caz general |

Modulul de calcul al derivatei și / 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 relația:

|  |  |
| --- | --- |
|  | (2) |

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

Legea de reglare a regulatoarelor convenționale este:

|  |  |
| --- | --- |
|  | (3) |

Unde kd = Td (constanta de timp de derivare). Remarcăm aici faptul că la regulatoarele fuzzy, ca de altfel și 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 și de este DE atunci u este U.*

Regulatorul fuzzy primește ca intrare eroarea și derivata erorii, iar la ieșire va indica mărimea de comandă, universul de discurs fiind .

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

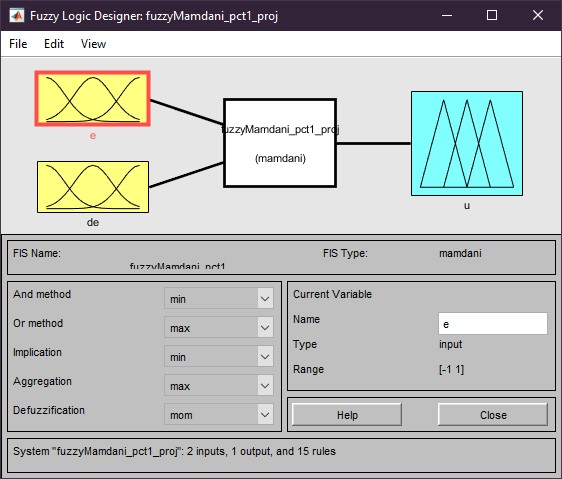
NB – trimf([-1, -1, -0.5])

NS – trimf([-1, -0.5, 0])

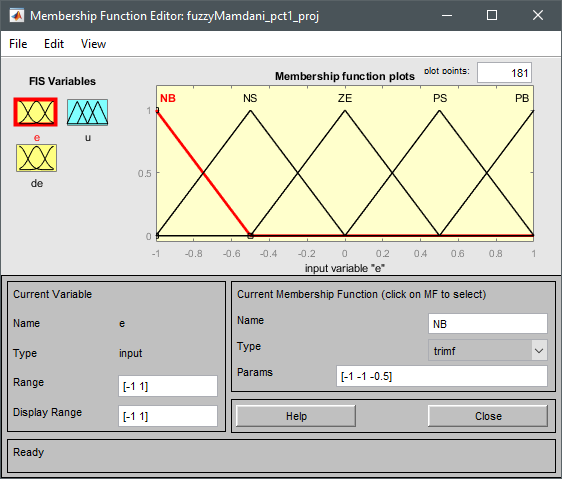
ZE – trimf([-0.5, 0, 0.5])

PS – trimf([0, 0.5, 1])

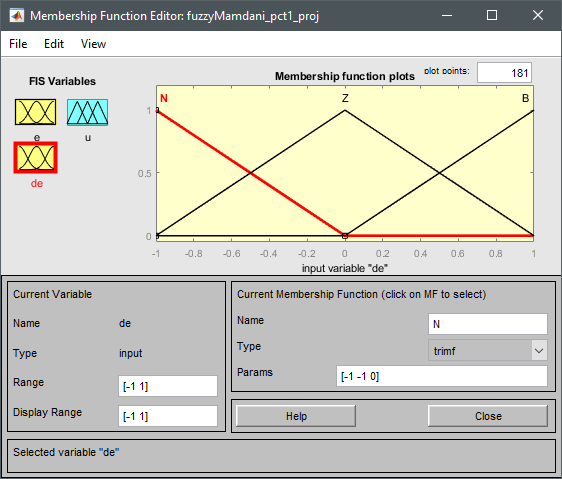
PB – trimf([0.5, 1, 1])



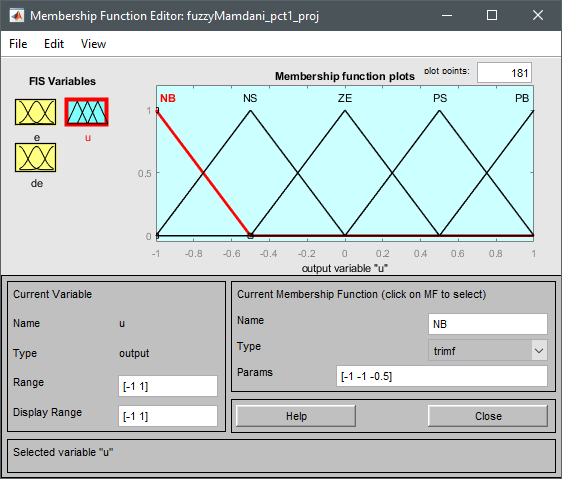
**Figura 4.** Fuzzy Logic Designer



**Figura 5.** Funcțiile de apartenență – eroare

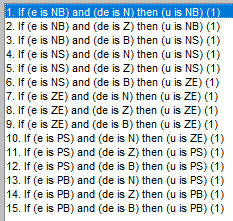


**Figura 6.** Funcțiile de apartenență – derivata erorii



**Figura 7.** Funcțiile de apartenență – mărimea de comandă

Baza de reguli (Figura 8) este cea impusă în Tabelul 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++ și mediul de dezvoltare Visual Studio.

Programul curpinde funcții de reprezentare a mulțimilor fuzzy, baza de reguli, metodele de fuzificare si defuzificare.

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

Defuzificarea a fost implementată folosind metoda mijlocul maximelor.

Se pot observa următoarele rezultate:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabelul 3.** Comparație rezultate MATLAB și 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 inferență fuzzy, o soluție este aproximarea sistemului de inferență printr-un tabel de inferență Mamdani. Pentru acest lucru, fiecărei variabile de intrare i se stabilește o mulțime de bază discretă, păstrând un număr de puncte din mulțimea de bază continuă dată inițial. Astfel:

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

|  |  |
| --- | --- |
|  | (6) |

b) pentru variabila *derivata\_erorii* se aleg 5 valori

|  |  |
| --- | --- |
|  | (7) |

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

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

|  |
| --- |
|  |
| **Figura 9.** Rezultat script MATLAB. |

Aceste rezultate au fost completate și în tabelul 4.

**Tabelul 4.** Tabelul de inferență 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 inferenței pentru o valoare strictă a variabilelor de intrare e\* și de\* se face după cum urmează.

Se consideră că valoarea e\* se află între valori consecutive din tabel, ei și ei+1. La fel și pentru de\*, dej și dej+1. În acest caz, în calculul valorii de ieșire u\* ar trebui să fie considerate valorile ui, j, ui+1, j, ui, j+1, ui+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 ei, cu atât valoarea corespunzătoare lui ui trebuie să contribuie mai mult în calculul lui u\*. Apropierea lui e\* de ei este direct proporțională cu depărtarea lui e\* de ei+1.

Ponderile celor 4 valori se vor calcula astfel:

|  |  |
| --- | --- |
|  | (8) |
|  |
|  |
|  |

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

|  |  |
| --- | --- |
|  | (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 și pe baza perioadei de eșantionare. În varianta cea mai simplă, relația de calcul a derivatei este:

|  |  |
| --- | --- |
|  | (10) |

unde este valoarea derivatei erorii corespunzătoare iterației curente, și sunt valorile mărimii de eroare de la iterația curentă, respectiv de la iterația anterioară, iar este perioada de eșantionare (intervalul de timp dintre două iterații consecutive).

Pentru calculul derivatei numerice a erorii, considerăm perioada de eșantionare .

În fișierul ”date.txt” se găsește un set de valori ”înregistrate” ale mărimii de *eroare* dintr-un sistem automat. Din cauza perturbațiile 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 eșantion înregistrat au fost calculate cu ecuația (10).

În urma calculării tuturor valorilor mărimii de comandă prin cele două programe realizate la punctele A și C ale proiectului, au rezultat timpi de calcul diferiți. Aceștia sunt prezentați 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 diferență 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)

Funcția de citire a valorilor din fișier și funcția 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;  } |