Universitatea *Transilvania* din Brașov

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Departamentul Automatică şi Tehnologia Informației

Proiect
Sisteme Inteligente de Control

Îndrumător:

Șef. Lucr. Dr. Ing. Cristian Boldișor

Student:

Vancea-Farcaș Alexandru

Grupa 4461

Proiect
Sisteme inteligente de control

1. Tema proiectului

A) Se va proiecta un regulator fuzzy pentru controlul unui proces descris de funcția de transfer:

$$G_p(s)=\frac{K_p}{(sT_1+1)(sT_2+1)} \quad (1)$$

în care parametrii procesului și performanțele impuse sunt preluate din tabelul din anexă.

B) Se va realiza o implementare a regulatorului proiectat, în limbajul C/C++, care ulterior se va testa pentru diverse valori ale mărimilor de intrare ale sistemului de inferență prin comparație cu funcționarea aceluiași sistem de inferențe realizat în Matlab.

Numărul temei proiectului și datele de proiectare sunt:

Tabelul 1. Datele de proiectare.

Tema nr.	Parametrii procesului			Performanțele impuse		
	K_p	T_1	T_2	e_{st}	M_v	t_s
28	1.5	120	12	1%	2%	180

2. Indicații și recomandări

a) Proiectarea și analiza sistemului de reglare se va face folosind mediul Matlab-Simulink. Implementarea în program se va face doar după determinarea sistemului de inferențe cu care se obțin performanțele impuse.

b) Nu sunt impuse restricții pentru tipul regulatorului fuzzy sau tipul inferențelor utilizate, decât aceea că alegerea acestora trebuie să conducă la obținerea performanțelor.

c) Programul realizat va fi testat prin comparație cu funcționarea sistemului implementat în mediul Matlab. Se vor lua diverse valori ale variabilelor de intrare ale sistemului de inferență pentru care se vor înregistra valorile de ieșire obținute în Matlab și cele obținute cu ajutorul programului.

3. Conținutul minimal al proiectului

a) Enunțul și descrierea problemei.

b) Prezentarea sistemului de reglare și a structurii regulatorului fuzzy ales. *Obs*: Se va justifica alegerea tipului de regulator.

c) Prezentarea sistemului de inferență obținut (variabile fuzzy, funcții de apartenență, reguli, tipul inferenței fuzzy, caracteristica statică, metoda de defuzificare – dacă este cazul –etc.). *Obs*: Se vor justifica toate alegerile referitoare la sistemul de inferență.

d) Prezentarea rezultatelor simulării sistemului de reglare proiectat (Matlab – Simulink).

e) Prezentarea particularităților de implementare, utilizate la realizarea programului.

f) Se vor anexa schema Simulink și programul realizat.

4. Condiții pentru susținerea proiectului

a) Proiectul va fi prezentat și susținut. Se vor adresa întrebări referitoare la noțiuni de bază și aspecte întâlnite în realizarea lui.

b) Proiectul predat trebuie să conțină: redactarea (format electronic și printat), fișierele obținute cu ajutorul aplicației de editare a sistemelor de inferență fuzzy, schemele Simulink, codul sursă al programului și fișierul executabil.

c) Proiectul poate fi predat până la data examenului scris.

5. Rezolvarea proiectului

A) Proiectarea regulatorului fuzzy

Noțiuni teoretice

Se consideră structura clasică de reglare, cu un singur regulator principal, aflat pe calea directă a sistemului. 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 măsurată 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:

$$e(t) = r(t) - y(t) \tag{2}$$

Pe baza erorii de reglare, se calculează mărimea de comandă aplicată părții fixate, aplicând o lege de reglare:

$$u(t) = f_R(e(t)) \tag{3}$$

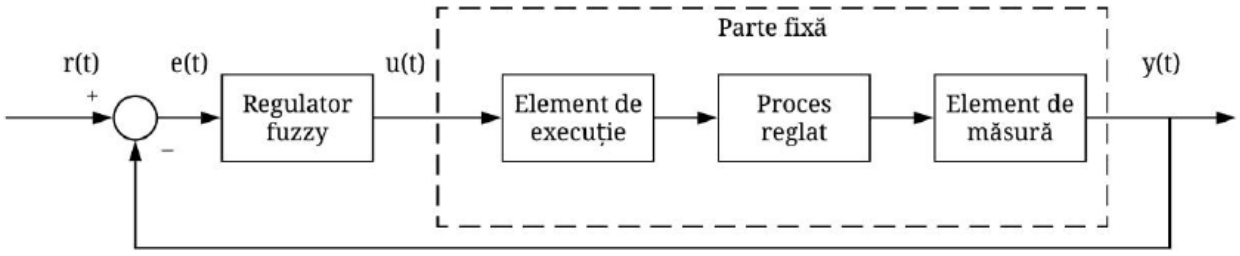

Figura 1. Structura convențională de reglare.

Funcționarea sistemelor fuzzy au la bază logica fuzzy. 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.

Structura unui regulator fuzzy în cazul general este prezentată în figura de mai jos.

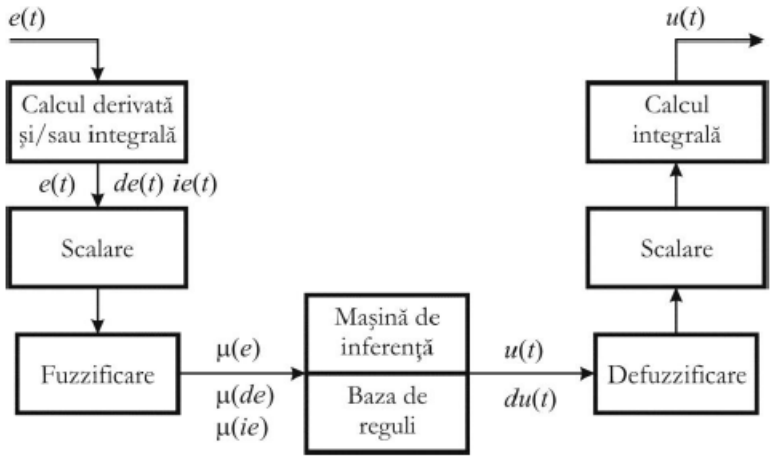

Figura 2. Structura unui regulator fuzzy.

Tipul regulatoarelor fuzzy este corelat de mărimile de intrare și de ieșire ale mașinii de inferență, mai exact de includerea derivatelor și integralelor calculate pentru eroarea de reglare și mărimea de comandă.

Alegerea regulatorului s-a realizat pe baza experimentelor, intuitiv. S-a încercat folosirea unui regulator de tip PI și a unui regulator de tip PD.

Regulatorul fuzzy de tip PI

Legea de reglare este

$$u = k_p \cdot e + k_i \cdot \int e \cdot dt \tag{4}$$

unde k_p și k_i sunt coeficienții de amplificare a componentei proporționale, respectiv a componentei integrale.

Variabilele lingvistice pentru regulatorul fuzzy PI vor fi:

- eroarea;
- derivata erorii;
- derivata mărimii de comandă.

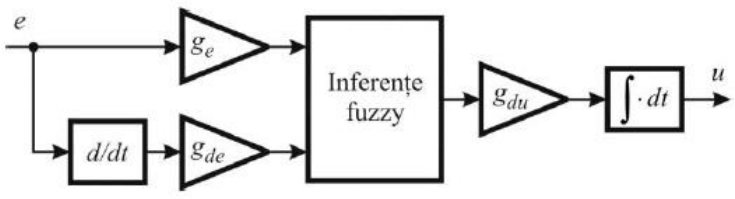

Figura 3. Regulatorul fuzzy PI.

Regulatorul fuzzy de tip PD

Legea de reglare este

$$u = k_p \cdot e + k_d \cdot \dot{e} \tag{5}$$

unde $k_d = T_d$ (constanta de timp de derivare).

Variabilele lingvistice pentru regulatorul fuzzy PD vor fi:

- eroarea;
- derivata erorii;
- comanda.

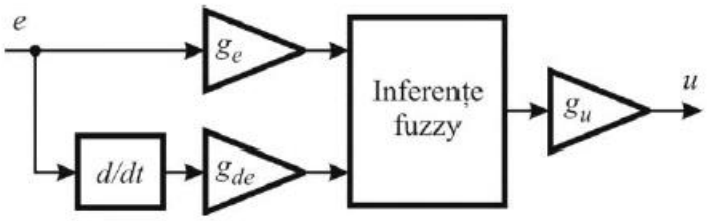

Figura 4. Regulatorul fuzzy PD.

Proiectarea regulatorului fuzzy de tip PD

Variabilele lingvistice pentru regulatorul fuzzy de tip PD, cele prezentate mai sus, prezintă diferite mulțimi fuzzy. În acest caz, mulțimiile fuzzy pentru cele trei variabile lingvistice sunt:

1. Pentru eroare
    - Negative Big (NB);
    - Negative Small (NS);
    - Zero (ZE);
    - Positive Small (PS);
    - Positive Big (PB).
2. Pentru derivata erorii
    - Negative (NE);
    - Zero (ZE);
    - Positive (PO).

3. Pentru comandă
    - Negative Big (NB);
    - Negative Small (NS);
    - Zero (ZE);
    - Positive Small (PS);
    - Positive Big (PB).

O regulă tipică din bază are formularea:

daca e este *NB* și de este *PO* atunci u este *NS*

Regula prezentată mai sus specifică faptul că, dacă mărimea de eroare, în acel moment de timp, aparține mulțimii NB și derivata erorii are sens pozitiv, atunci mărimea de comandă va avea o valoare ce se încadrează în mulțimea NS.

Regulatorul fuzzy PD ce folosește inferențe de tip Mamdani, primește ca intrare eroarea și derivata erorii, iar la ieșire va indica mărimea de comandă, universul de discurs fiind [-1, 1]. Sistemul de reglare este bazat pe un regulator fuzzy de tip PD ce folosește ca metodă de defuzificare metoda centrului de greutate. Operatorul pentru conjuncție este *min*, operatorul pentru disjuncție este *max,* tipul implicației este *min,* tipul agregării regulilor este *max.*

Proiectarea regulatorului fuzzy de tip PD a fost realizată, în prima etapă, în Matlab folosind toolbox-ul fuzzy. Mulțimile fuzzy ale variabilelor lingvistice și baza de reguli au fost create intuitiv, suferind apoi o serie de modificări în scopul încadrării performanțelor sistemului în limitele impuse. Cele mai bune performanțe s-au obținut cu datele prezentate mai jos.

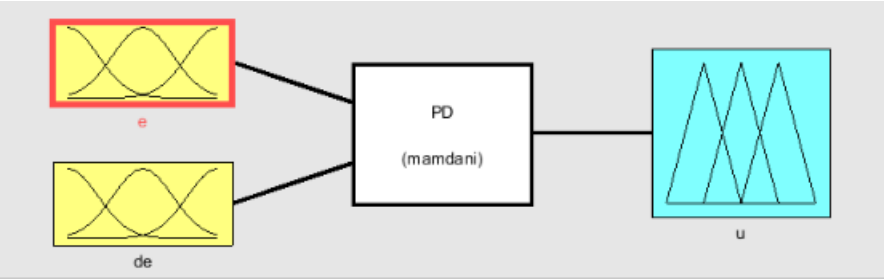

Figura 5. Regulatorul fuzzy PD.

Variabila lingvistică corespunzătoare erorii prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NB: funcție trapez cu parametrii [-1, -1, -0.4, -0.2];
- NS: funcție trapez cu parametrii [-0.3, -0.25, -0.1, 0];
- ZE: funcție triunghi cu parametrii [-0.01, 0, 0.01];
- PS: funcție trapez cu parametrii [0, 0.1, 0.25, 0.3];
- PB: funcție trapez cu parametrii [0.2, 0.4, 1, 1].

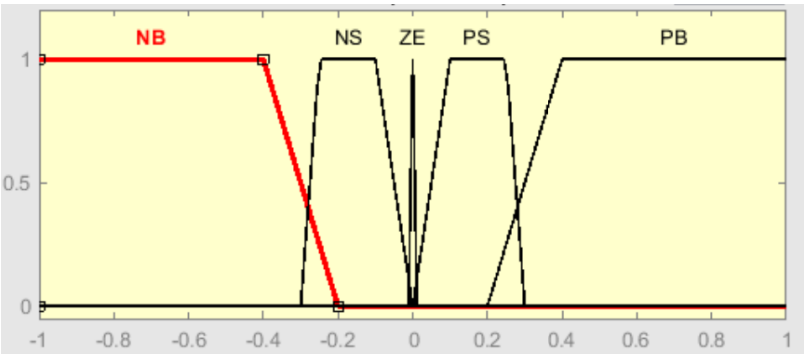

Figura 6. Mulțimile erorii.

Variabila lingvistică corespunzătoare derivatei erorii prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NE: funcție triunghi cu parametrii [-1, -1, 0.0001];
- ZE: funcție triunghi cu parametrii [-0.01 0, 0.01];
- PO: funcție triunghi cu parametrii [0.0001, 1, 1].

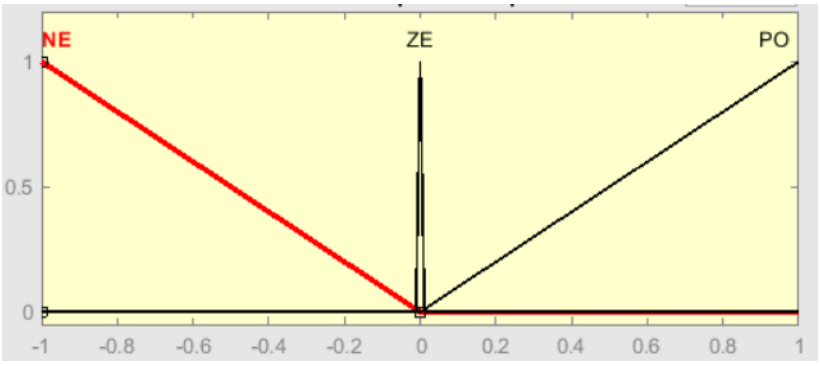

Figura 7. Mulțimile derivatei erorii.

Variabila lingvistică corespunzătoare mărimii de comandă prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NB: funcție triunghi cu parametrii [-1, -1, -0.5];
- NS: funcție triunghi cu parametrii [-1, -0.5, 0];
- ZE: funcție triunghi cu parametrii [-0.01, 0, 0.01];
- PS: funcție triunghi cu parametrii [0, 0.5, 1];
- PB: funcție triunghi cu parametrii [0.5,1, 1].

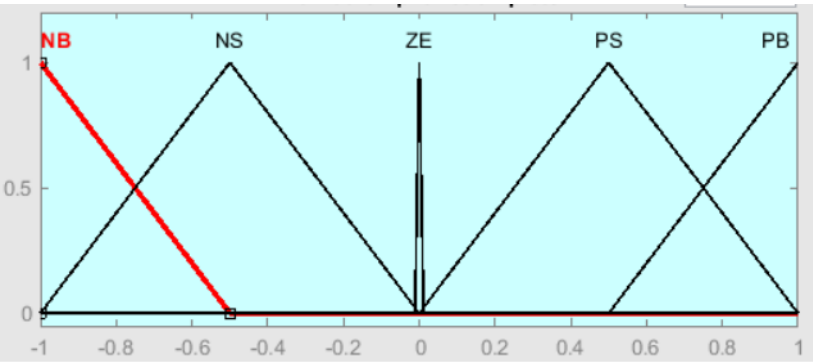

Figura 8. Mulțimile mărimii de comandă.

Baza de reguli aleasă este prezentată în tabelul de mai jos.

Tabelul 2. Baza de reguli.

de/e	NB	NS	ZE	PS	PB
NE	NB	NS	ZE	ZE	PS
ZE	NB	NS	ZE	PS	PB
PO	NS	ZE	ZE	PS	PB

În urma celor prezentate a fost proiectat un regulator fuzzy PD. Pentru a determina care sunt performanțele sistemului folosind acest regulator s-a realizat o simulare a sistemului de reglare folosind toolbox-ul Simulink. S-a considerat ca ge și gde să fie egal cu 1, iar gu egal cu 1.7.

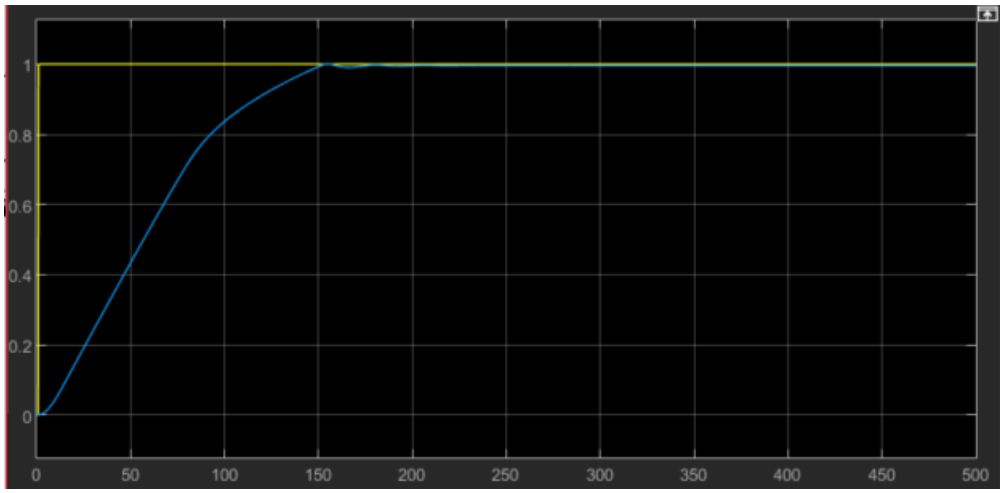

Figura 9. Răspunsul sistemului utilizând un regulator fuzzy de tip PD.

În urma simulării s-au obținut următoarele performanțe.

- Eroare staționară: $e_{st} = 0.5\%$;
- Suprareglaj : $M_v = 0.49\%$;
- Timp de stabilire: $t_s = 200s$.

După cum se poate observa, sistemul are un timp de stabilire mai mare față de valoarea impusă.

Schema de reglare pentru realizarea simulării sistemului de reglare este afișată mai jos:

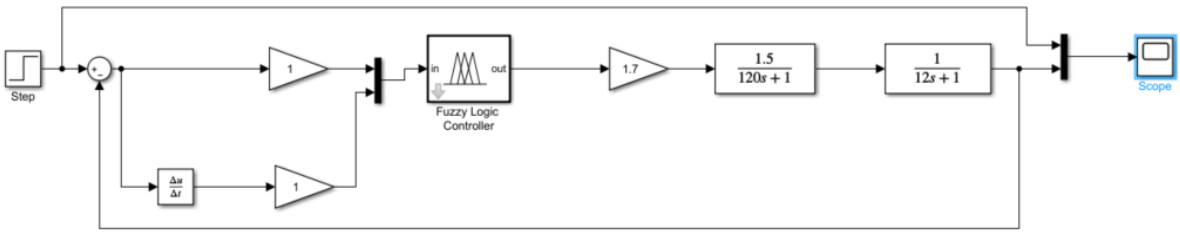

Figura 10. Structura de reglare în cazul unui regulator de tip PD.

.

Proiectarea regulatorului fuzzy de tip PI

Considerând aceleași date utilizate pentru proiectarea regulatorului PD, s-a realizat proiectarea unui regulator fuzzy de tip PI. Am realizat acest lucru pentru a observa diferența dintre cele două regulatoare proiectate pe baza acelorași date.

În figura de mai jos este prezentată simularea sistemului de reglare. Performanțele acestuia nu sunt deloc corespunzătoare.

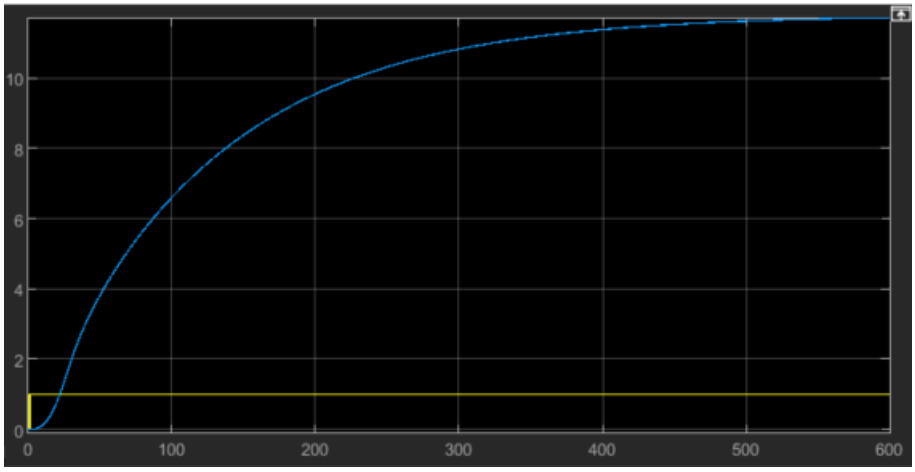

Figura 11. Răspunsul sistemului utilizând un regulator fuzzy de tip PI.

Am încercat să modific baza de date în vederea obținerii unor performanțe mai bune cu un regulator fuzzy de tip PI.

Variabilele lingvistice pentru regulatorul fuzzy de tip PI, cele prezentate mai sus, prezintă diferite mulțimi fuzzy. În acest caz, mulțimiile fuzzy pentru cele trei variabile lingvistice sunt:

1. Pentru eroare
    - Negative Big (NB);
    - Negative Small (NS);
    - Zero (ZE);
    - Positive Small (PS);
    - Positive Big (PB)

2. Pentru derivata erorii
    - Negative (NE);
    - Zero (ZE);
    - Positive (PO).
3. Pentru derivata mărimii de comandă
    - Negative Big (NB);
    - Negative Small (NS);
    - Zero (ZE);
    - Positive Small (PS);
    - Positive Big (PB).
    -

O regulă tipică din bază are formularea:

daca e este *NB* și de este *PO* atunci du este *NS*

Regula prezentată mai sus specifică faptul că, dacă mărimea de eroare, în acel moment de timp, aparține mulțimii NB și derivata erorii are sens pozitiv, atunci derivate mărimii de comandă va avea o valoare ce se încadrează în mulțimea NS.

Regulatorul fuzzy PI ce folosește inferențe de tip Mamdani, primește ca intrare eroarea și derivata erorii, iar la ieșire va indica derivata mărimii de comandă, universul de discurs fiind [-1, 1]. Sistemul de reglare este bazat pe un regulator fuzzy de tip PI ce folosește ca metodă de defuzificare metoda centrului de greutate. Operatorul pentru conjuncție este *min*, operatorul pentru disjuncție este *max,* tipul implicației este *min,* tipul agregării regulilor este *max.*

Proiectarea regulatorului fuzzy de tip PI a fost realizată, în prima etapă, în Matlab folosind toolbox-ul fuzzy. Mulțimile fuzzy ale variabilelor lingvistice sunt aproape aceleași de la regulatorul fuzzy de tip PD, suferind mici modificări, însă baza de reguli a fost modificată în vederea obținerii unor performanțe satisfăcătoare.

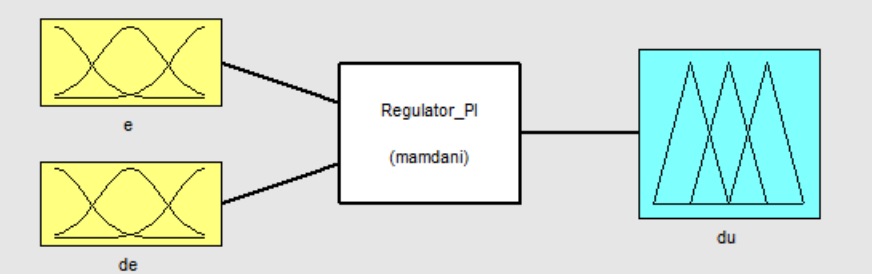

Figura 12. Regulatorul fuzzy PI.

Variabila lingvistică corespunzătoare erorii prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NB: funcție trapez cu parametrii [-1, -1, -0.3, -0.1];
- NS: funcție trapez cu parametrii [-0.2, -0.15, -0.1, 0];
- ZE: funcție triunghi cu parametrii [-0.01, 0, 0.01];
- PS: funcție trapez cu parametrii [0, 0.1, 0.15, 0.2];
- PB: funcție trapez cu parametrii [0.1, 0.3, 1, 1].

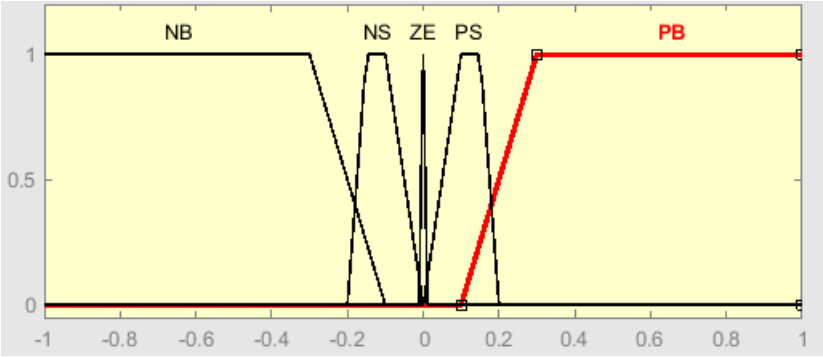

Figura 13. Mulțimile erorii.

Variabila lingvistică corespunzătoare derivatei erorii prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NE: funcție trapez cu parametrii [-1, -1, -0.9, 0];
- ZE: funcție triunghi cu parametrii [-0.9, 0, 0.9];
- PO: funcție trapez cu parametrii [0, 0.9, 1, 1].

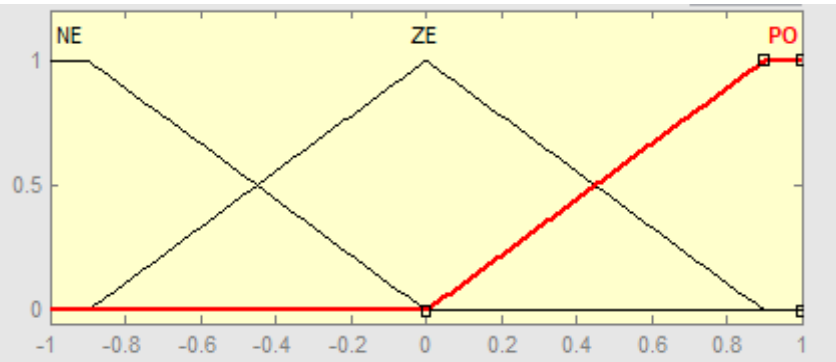

Figura 13. Mulțimile derivatei erorii.

Variabila lingvistică corespunzătoare derivatei mărimii de comandă prezintă, precum se poate observa și în figura de mai jos, următoarele mulțimi fuzzy:

- NB: funcție trapez cu parametrii [-1, -1, -0.95, -0.55];
- NS: funcție trapez cu parametrii [-0.95, -0.55, -0.45, -0.05];
- ZE: funcție triunghi cu parametrii [-0.1, 0, 0.1];
- PS: funcție trapez cu parametrii [0.05, 0.45, 0.55, 0.95];
- PB: funcție trapez cu parametrii [0.55, 0.95, 1, 1].

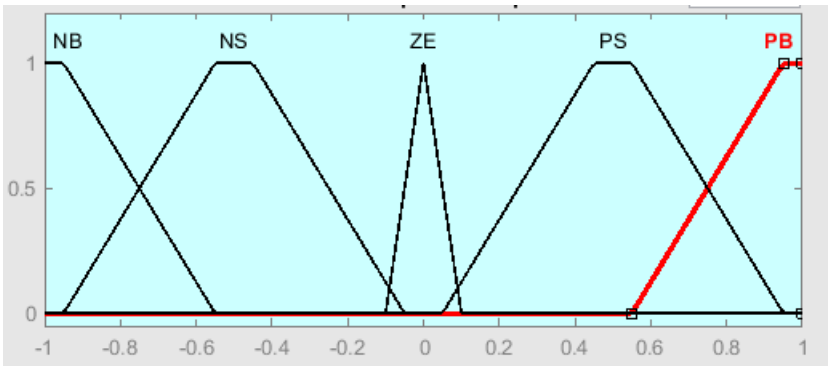

Figura14. Mulțimile derivatei mărimii de comandă.

Baza de reguli aleasă este prezentată în tabelul de mai jos.

Tabelul 3. Baza de reguli.

de/e	NB	NS	ZE	PS	PB
NE	NB	NS	ZE	ZE	PS
ZE	NB	NS	ZE	PS	PB
PO	NC	ZE	ZE	PS	PB

În urma celor prezentate a fost proiectat un regulator fuzzy PI. Pentru a determina care sunt performanțele sistemului folosind acest regulator s-a realizat o simulare a sistemului de reglare folosind toolbox-ul Simulink. S-a considerat ca ge și gde să fie egal cu 1, iar gdu egal cu 0.09.

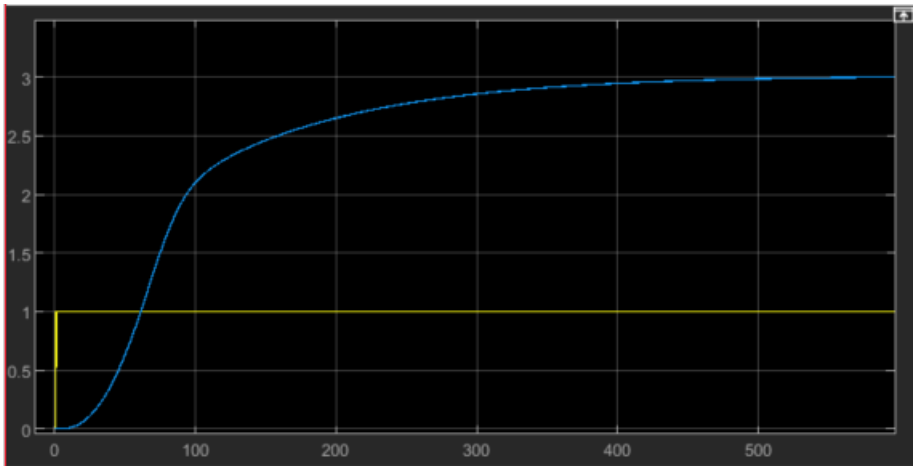

Figura 15. Răspunsul sistemului utilizând un regulator fuzzy de tip PI.

În urma simulării, performanțele obtinute nu sunt deloc satisfăcătoare, având un suprareglaj mare, eroare staționară mare și un timp de stabilire, de asemenea, mare.
Schema de reglare pentru realizarea simulării sistemului de reglare este prezentată mai jos:

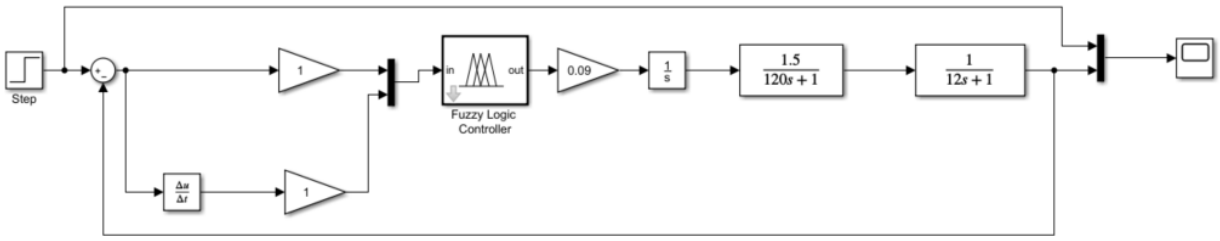

Figura 16. Structura de reglare în cazul unui regulator de tip PI.

În concluzie, în urma simulărilor am constatat că, cu un regulator fuzzy de tip PI nu am reușit să obțin performanțele impuse, însă cu un regulator fuzzy de tip PD am reușit să obțin performanțe cât mai aproapiate de cele impuse.

B) Implementarea regulatorului proiectat

Implementarea regulatorului obținut s-a realizat prin intermediul mediului de dezvoltare Visual Studio, folosind limbajul de programare C++. Procesul vizat are în vedere implementarea regulatorului proiectat folosind toolbox-ul din Matlab fuzzy. Programul conține atât funcțiile de reprezentare a mulțimiilor fuzzy și baza de reguli, cât și metodele de fuzificare, precum și defuzificare. Pentru procesul de defuzificare a fost implementată metoda centrelor de greutate. Codul implementării regulatorului este prezentat în Anexa B.
În tabelul de mai jos sunt reprezentate diferențele dintre implementările sistemului fuzzy în limbajul C++, precum și cele folosind utilitarul din Matlab.

Tabelul 4. Diferențele dintre cele două implementări.

Eroarea	Derivata erorii	Mărimea de comandă (C++)	Mărimea de comandă (Matlab)
-0.3	0.7	-0.592	-0.6
0	0	0.428	0.4
-0.9	0.9	-0.604	-0.625
0.2	-0.3	0.875	0.84

6. Bibliografie

[1] BOLDIȘOR C., Curs “Sisteme inteligente de control”.

[2] BOLDIȘOR C., Laborator “Sisteme inteligente de control”.

[1] Dafinca, L., Suciu, C., – Sisteme expert în automatică, Reprografia Universității Transilvania, Brașov, 2003.

Anexa A. Codul sursă al implementării regulatorului

main.cpp file

```
#include <iostream>
#include <string>
#include"regulator.h"


int main()
{
	float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE];
	float functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII];
	float iesire[NUMAR_FUNCTII_APARTENENTA_EROARE];
	float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII];
	float e, de;

	Initializari(functiiEroare, functiiDerivataEroare, matriceApartenenta, iesire);
	float c1, c2, c3, c4, c5;

	std::string reguli[NUMAR_FUNCTII_APARTENENTA_EROARE]
	[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII] =
	{ { "NB", "NB", "NB" },
	{ "NB", "NB", "NS" },
	{ "PS", "PS", "NS" },
	{ "PB" ,"PB", "PB" },
	{ "PB", "PB", "PB" } };

	std::cout << "Introduceti o valoare pentru eroare (intre -1 si 1):" << std::endl;
	std::cout << "e = ";
	std::cin >> e;

	std::cout << "Valoarea derivatei erorii (intre -1 si 1): " << std::endl;
	std::cout << "de = ";
	std::cin >> de;
	std::cout << std::endl;

	EroareFuzificare(e, functiiEroare);
	DerivataFuzificare(de, functiiDerivataEroare);

	std::string variabile[] = { "NB", "NS", "ZE", "PS", "PB" };
	std::cout << "Aparteneta eroare." << std::endl;
	for (int i = 0; i < NUMAR_FUNCTII_APARTENENTA_EROARE; i++)
	{
		std::cout << e << " apartine " << functiiEroare[i] << " " + variabile[i] <<
	std::endl;
	}
	std::cout << std::endl;

	std::string variabileDerivata[] = { "NE", "ZE", "PO" };
	std::cout << "Aparteneta derivata eroare" << std::endl;
```

```
for (int i = 0; i < NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII; i++)
	{
		std::cout << de << " apartine " << functiiDerivataEroare[i] << " " +
variabileDerivata[i] << std::endl;
	}

	//Se determina tabela de inferenta.
	std::cout << std::endl;
	std::cout << "Tabela de inferenta" << std::endl;
	Inferenta(functiiEroare, functiiDerivataEroare, matriceApartenenta);

	//Se realizeaza compunerea.
	for (int i = 0; i < NUMAR_FUNCTII_APARTENENTA_EROARE; i++)
	{
		iesire[i] = Compunere(matriceApartenenta, reguli, variabile[i]);
	}

	//Se calculeaza centrele.
	c1 = CalculCentruTriunghi(-1, -1,-0.5, iesire[0]);
	c2 = CalculCentruTriunghi(-1, -0.5, 0, iesire[1]);
	c3 = CalculCentruTriunghi(-0.01, 0, 0.01, iesire[2]);
	c4 = CalculCentruTriunghi(0, 0.5, 1, iesire[3]);
	c5 = CalculCentruTriunghi(0.5, 1, 1, iesire[4]);

	// Rezultatul defuzificarii.
	std::cout << std::endl;
	std::cout << "Defuzificare = " << CalculDefuzificare(iesire[0], iesire[1],
iesire[2], iesire[3], iesire[4], c1, c2, c3, c4, c5) << std::endl;

	system("pause");
	return 0;

}
```

regulator.h header

```
#pragma once

#define NUMAR_FUNCTII_APARTENENTA_EROARE          5
#define NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII 3


void Initializari(float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE],
                  float
functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII],
                  float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII],
                  float iesire[NUMAR_FUNCTII_APARTENENTA_EROARE]);
float triunghi(float a, float b, float c, float x);
float trapez(float a, float b, float c, float d, float x);
float CalculCentruTriunghi(float a, float b, float c, float y);
float CalculCentruTrapez(float a, float b, float c, float d, float y);
void EroareFuzificare(float e, float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE]);
void DerivataFuzificare(float de, float
functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII]);
void Inferenta(float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE],
```

```
                        float
functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII]
                        float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII]);
float Compunere(float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII],
                            std::string
reguli[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII],
                            std::string variabila);

float CalculDefuzificare(float y1, float y2, float y3, float y4, float y5, float c1,
float c2, float c3, float c4, float c5);
```

regulator.cpp file

```
#include <iostream>
#include"regulator.h"

// Functie care realizeaza initializarea tuturor matricelor si tablourilor folosite.
void Initializari(float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE],
	float functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII],
	float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII],
	float iesire[NUMAR_FUNCTII_APARTENENTA_EROARE])
{
	int i = 0;
	int j = 0;

	for (i = 0; i < NUMAR_FUNCTII_APARTENENTA_EROARE; i++)
	{
		functiiEroare[i] = 0;
		iesire[i] = 0;
	}

	for (i = 0; i < NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII; i++)
	{
		functiiDerivataEroare[i] = 0;
	}

	for (i = 0; i < NUMAR_FUNCTII_APARTENENTA_EROARE; i++)
	{
		for (j = 0; j < NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII; j++)
		{
			matriceApartenenta[i][j] = 0;
		}
	}
}

//Functia de apartenenta triunghi.
float triunghi(float a, float b, float c, float x)
{
	if ((x < a) || (x > c))
	{
		return 0;
	}
```

```
	else if ((x >= a) && (x <= c))
	{
		return ((x - a) / (b - a));
	}
	else
	{
		return ((c - x) / (c - b));
	}
}

//Functia de apartenenta trapez.
float trapez(float a, float b, float c, float d, float x)
{
	if (x < a)
	{
		return 0;
	}
	else if ((x >= a) && (x <= b))
	{
		return (x - a) / (b - a);
	}
	else if ((x > b) && (x < c))
	{
		return 1;
	}
	else if ((x >= c) && (x <= d))
	{
		return (d - x) / (d - c);
	}
	else
	{
		return 0;
	}
}

//Calcularea centrelor pentru defuzificare.
float CalculCentruTriunghi(float a, float b, float c, float y)
{
	float y1, y2;
	if (y != 0)
	{
		y1 = y * (b - a) + a;
		y2 = c - (y * (c - b));

		return (y1 + y2) / 2;

	}
	else return 0;
}

float CalculCentruTrapez(float a, float b, float c, float d, float y)
{
	float y1, y2;
	if (y != 0)
	{
		y1 = y * (b - a) + a;
		y2 = d - y * (d - c);
```

```
			return (y1 + y2) / 2;
		}
		else return 0;
}

/**
Functie de fuzificare a erorii.
Variabilele lingvistice folosite si definirea acestora se realizeaza in cadrul acestei
functii.
NB - Negative Big
NS - Negative Small
ZE - ZEro
PS - Positive Small
PB - Positive Big
*/

void EroareFuzificare(float e, float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE])
{
		functiiEroare[0] = trapez(-1, -1, -0.4, -0.2, e);			//NB
		functiiEroare[1] = trapez(-0.3, -0.25, -0.1, 0, e);			//NS
		functiiEroare[2] = triunghi(-0.01, 0, 0.01, e);				//ZE
		functiiEroare[3] = trapez(0, 0.1, 0.25, 0.3, e);			//PS
		functiiEroare[4] = trapez(0.2, 0.4, 1, 1, e);				//PB
}

/**
Functie de fuzificare a derivatei erorii.
Derivata erorii este folosita pentru a putea determina semnul erorii intr-un anumit
moment.
Ceea ce va ajuta foarte mult in a determina cum se comporta eroarea si ce trebuie facut
pentru
a reduce eroarea.
NE - Negative
ZE - Zero
PO - Positive
*/
void DerivataFuzificare(float de, float
functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII])
{
		functiiDerivataEroare[0] = triunghi (-1, -1, 0.0001, de);		//NE
		functiiDerivataEroare[1] = triunghi(-0.01, 0, 0.01, de);		//ZE
		functiiDerivataEroare[2] = triunghi (0.0001, 1, 1, de);			//PO
}

/**
In aceasta functie se realizeaza inferenta fuzzy.
Metoda folosita presupune ca pentru construirea tabelei de inferenta sa se aleaga minimul
valorii functiilor
de apartenenta ale erorii si derivatei erorii in functie de baza de reguli.
*/
void Inferenta(float functiiEroare[NUMAR_FUNCTII_APARTENENTA_EROARE],
		float functiiDerivataEroare[NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII],
		float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII])
{
```

```
	for (int i = 0; i < 5; i++)
	{
		for (int j = 0; j < 3; j++)
		{
			if (functiiEroare[i] > functiiDerivataEroare[j])
			{
				matriceApartenenta[i][j] = functiiDerivataEroare[j];
			}
			else
			{
				matriceApartenenta[i][j] = functiiEroare[i];
			}
		}
	}

	for (int i = 0; i < 5; i++)
	{
		for (int j = 0; j < 3; j++)
		{
			std::cout << matriceApartenenta[i][j] << " ";
		}
		std::cout << std::endl;
	}
}

//Se realizeaza functia de compunere.
float Compunere(float
matriceApartenenta[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_E
RORII],
				std::string
reguli[NUMAR_FUNCTII_APARTENENTA_EROARE][NUMAR_FUNCTII_APARTENENTA_DERIVATA_ERORII],
				std::string variabila)
{
	float min = 0;
	for (int i = 0; i < 5; i++)
	{
		for (int j = 0; j < 3; j++)
		{
			if (reguli[i][j]._Equal(variabila))
			{
				if (min < matriceApartenenta[i][j])
					min = matriceApartenenta[i][j];
			}
		}
	}

	return min;
}

//Functia care realizeaza defuzificarea.
float CalculDefuzificare(float y1, float y2, float y3, float y4,
	float y5, float c1, float c2, float c3, float c4, float c5)
{
	float defuzificare = 0;
	defuzificare = (y1 * c1 + y2 * c2 + y3 * c3 + y4 * c4 + y5 * c5) / (y1 + y2 + y3 +
	y4 + y5);
	return defuzificare;}
```