

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE



PROIECT DE LICENȚĂ

STUDIUL COMPARATIV AL UNOR REGULATOARE
METAEURISTICE BIDIMENSIONALE: PI, PID SI PID CU
FILTRARE MEDIANĂ

Absolvent : Dragos-Alexandru Ezeanu

Coordonator științific: Prof. Dr. Ing. Dan Ștefănoiu
Coordonator științific: Prof. Dr. Ing. Janetta Culită

Septembrie, 2024

CONTENTS

1	Introducere	1
2	Prezentarea Instalației	3
2.1	Descrierea Instalației	4
2.2	Configurația Instalației	4
3	Identificarea parametrilor instalației	6
3.1	Formularea problemei de identificare	6
3.2	Modele parametrice	7
4	Optimizare Metaeuristică	11
4.1	Prezentare Metaeuristici	11
4.2	Particle Swarm Optimization (PSO)	12
4.3	Tehnici de Menținere a Particulelor în Spațiul de Căutare	14
5	Control PID Multivariabil	20
5.1	Legea de control PID	20
5.2	Regulator PID Multi-variabil	21
5.3	Implementare în MATLAB-SIMULINK	23
6	Studiu de caz	25
6.1	Functia fitness	25
6.2	Implementarea Algoritmului Metauristic	26
6.3	Perturbații și referințe de tipul treaptă succesivă	36
7	Concluzii	44
8	Contribuții ale studentului	45

Bibliografie	45
------------------------	----

MULTUMIRI

Aș dori să mulțumesc profesorilor coordonatori, Prof. Dr. Ing. Stefănoiu Dan și Prof. Dr. Ing. Culică Janetta, pentru sprijinul esențial în înțelegerea și abordarea lucrării. Carisma și modul dumneavoastră de comunicare au fost de mare ajutor în finalizarea lucrării de licență și în aprofundarea cunoștințelor despre controlul sistemelor. Experiența dobândită va fi extrem de valoioasă în viitoarea mea carieră de inginer. Vă mulțumesc pentru implicarea și timpul acordat.

1 INTRODUCERE

Proiectul de licență de față se concentrează pe dezvoltarea unui regulator PID multi-variabil pentru instalația ASTANK2, utilizată în reglarea nivelului de apă. Reglementarea precisă a nivelului apei este esențială în numeroase aplicații din industria apei, cum ar fi tratarea și distribuția acesteia, managementul rezervelor de apă sau instalațiile industriale de răcire. O optimizare eficientă a proceselor automate din aceste sisteme poate duce la o utilizare mai eficientă a resurselor și la reducerea pierderilor.

Motivația acestui proiect provine din necesitatea găsirii unui regulator capabil să mențină stabil nivelul apei în rezervoare, în fața unor variabile și perturbări externe. Algoritmul metaeuristic ales pentru această sarcină este Particle Swarm Optimization (PSO), cunoscut pentru capacitatea sa de a explora eficient spațiul de căutare și de a oferi soluții precise într-un timp scurt. Obiectivul principal a fost de a găsi parametrii optimi pentru regulatorul PID care să răspundă adecvat acestor provocări.

Un aspect esențial în cadrul acestui proiect a fost dezvoltarea unei funcții de fitness care să ajute la evaluarea performanței regulatoarelor propuse. Acest pas este crucial în optimizarea metaeuristică, deoarece funcția de cost determină calitatea soluțiilor obținute. Simulările au generat diferite variante de regulatoare, printre care cel mai eficient s-a dovedit a fi un regulator de tip PI, unde componenta derivativă a fost nulă.

Cu toate acestea, pentru a explora complet posibilitățile oferite de reglarea PID, s-au reluat căutările, cu scopul de a găsi o soluție optimizată pentru regulatorul complet. Deși rezultatul inițial pentru PID a arătat un fitness mai mic decât cel al PI-ului, analiza ulterioară a indicat că un filtru median aplicat pe comenziile PID a îmbunătățit performanțele acestuia. În final, PID-ul optimizat a oferit un compromis bun între complexitate și eficiență, însă PI-ul s-a menținut drept cea mai simplă și robustă soluție pentru aplicația specifică de reglare a nivelului apei.

Această lucrare subliniază importanța optimizării sistemelor fluide, oferind o soluție aplicabilă pentru controlul proceselor complexe de reglare a nivelului apei, cu impact direct asupra eficienței operaționale și economiei de resurse.

Lucrarea este structurată pe 8 capitole, astfel:

- În capitolul 2 se prezintă instalația și configurația acesteia pentru proiectul prezent.
- În capitolul 3 se prezintă modul în care a fost identificat modelul parametric al instalației.
- În capitolul 4 se prezintă metoda metaeuristică utilizată.

- În capitolul 5 se prezintă modul alegerii regulatorului PID.
- În capitolul 6 se prezintă modul de lucru și rezultatele obținute în urma analizării a 3 regulatoare diferite.
- În capitolul 7 se prezintă contribuțiile studentului.
- În capitolul 8 se prezintă concluziile formulate în urma rezultatelor obținute.

2 PREZENTAREA INSTALATIEI

Instalația ASTANK2 constă într-un sistem hidraulic de recirculare a apei , cu 2 bazine ce sunt deschise, de aceeași înaltime, dar cu formă diferită . Primul bazin (cel din dreapta) are formă de paralelipiped dreptunghic, iar al doilea are un perete înclinat, asa cum ilustrează figura:

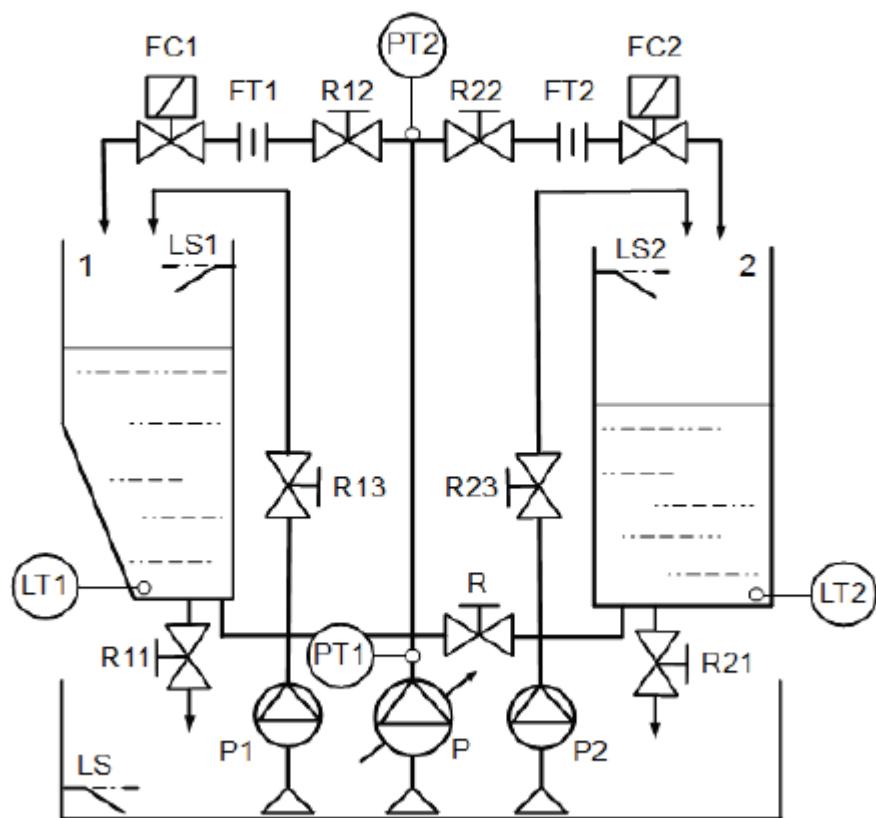


Figure 1: Sistemul ASTANK2 [1]

2.1 Descrierea Instalației

Instalația descrisă prezintă o configurare flexibilă, oferind posibilitatea de a funcționa fie ca un sistem interconectat, fie ca două sisteme independente. În cea mai complexă configurare, robinetul R permite comunicarea între cele două bazine superioare atunci când este deschis. În schimb, dacă acesta este închis, bazinele pot funcționa independent, fiecare fiind alimentat separat prin robinetele R13 și R23. De asemenea, golirea bazinelor se face prin robinetele R11 și R21, care pot fi deschise sau închise, la fel ca robinetul R.

Nivelul apei din bazine este monitorizat prin traductoarele LT1 și LT2. În cazul în care nivelul apei crește prea mult, senzorii de nivel LS1 și LS2 trimit semnale pentru oprirea pompei, prevenind astfel supra-umplerea. Senzorul LS asigură protecția pompei principale (P), astfel încât aceasta să nu funcționeze în gol, prevenind deteriorarea echipamentului.

Pompa principală (P) extrage apa din bazinul de acumulare și o distribuie în sistem prin conducta principală. Presiunea și debitul pot fi reglate prin ajustarea frecvenței pompei. Conducta principală se ramifică pentru a alimenta cele două bazine, iar presiunea este monitorizată cu ajutorul traductoarelor PT1 și PT2. De asemenea, debitul de apă pe fiecare ramură este măsurat de traductoarele FT1 și FT2 și reglat prin electrovalvele FC1 și FC2.

Pompele auxiliare P1 și P2 sunt utilizate pentru a adăuga suplimentar volum de apă în fiecare bazin, contribuind la menținerea unui debit constant prin comenzi de tip ON/OFF. Acestea introduc și perturbații controlate în sistem.

Astfel, instalația ASTANK2 poate fi modelată ca un sistem multivariabil MIMO (Multiple Input, Multiple Output), cu 3 intrări (tensiunile aplicate electrovalvelor și pompei principale) și 2 ieșiri (nivelul de apă din cele două bazine).

2.2 Configurația Instalației

După cum s-a menționat anterior, configurațiile posibile sunt variate în funcție de acționarea robinetelor. De exemplu, deschiderea robinetului RC determină o evoluție mai lentă a procesului de umplere și golire, deoarece nivelurile din cele două bazine se stabilizează după o perioadă mai lungă de timp.

Pe de altă parte, robinetele RP1 și RP2 sunt utilizate pentru a introduce perturbații în procesul din fiecare bazin, simulând astfel condiții variante de operare. Acest lucru permite o testare mai amplă și o adaptare a sistemului în funcție de dinamica impusă de aceste schimbări.

Pentru lucrarea prezentată, configurația a fost aleasă astfel:

- Toate robinetele prezente în instalație sunt în pozitie de **deschis**, excepție făcând robinetele **R11** și **R12** ce sunt deschise la 50%
- Pompele auxiliare nu sunt folosite ca parametrii de intrare, fiind setate pe **OFF**

În urma acestor configurații, procesul este de tip multivariabil, cu 3 intrări și 2 ieșiri.

3 IDENTIFICAREA PARAMETRILOR INSTALAȚIEI

În acest capitol, va fi prezentată metodologia de identificare experimentală a unui model matematic valid, care să descrie cu acuratețe procesul de umplere-golire al instalației ASTANK2. Cu alte cuvinte, scopul este găsirea unui model de identificare care să reflecte caracteristicile funcționale ale procesului. De obicei, procesul este tratat ca o cutie neagră, ceea ce înseamnă că primim semnale de ieșire atunci când aplicăm diverse comenzi sau stimuli. Modelul va fi identificat ulterior prin metode de identificare stabilite, utilizând aceste date experimentale.

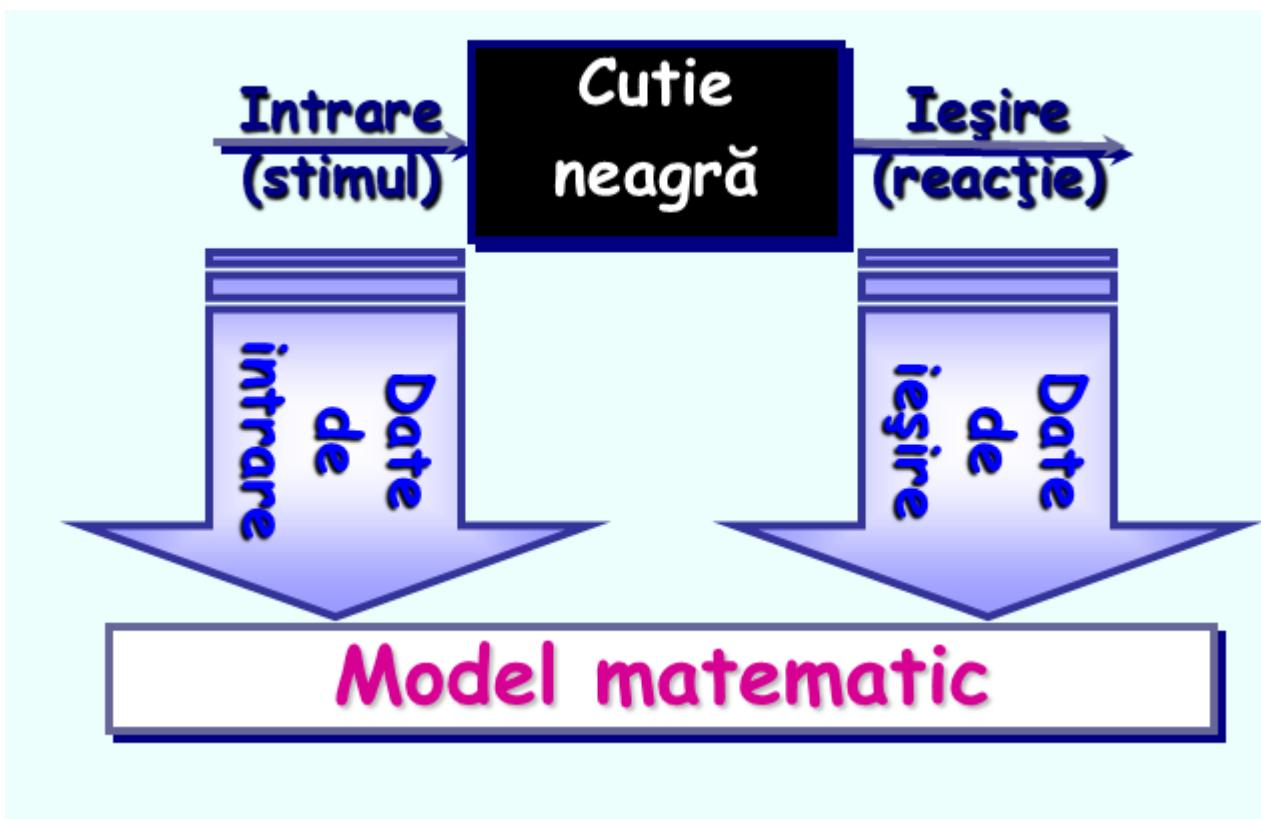


Figure 2: Model Cutie Neagră [5]

3.1 Formularea problemei de identificare

Obiectivul principal al identificării este determinarea unui model stocastic, cu structură și comportament necunoscut, și construirea unui model matematic care să fie adecvat procesului într-un sens bine definit (conform unor criterii de adecvară). În acest caz, modelul matematic este descris de un anumit număr de parametri necunoscuți ce trebuie determinați.

În urma stimulării cutiei negre cu un set de semnale, aceasta ne oferă un alt set de date experimentale. Prin urmare, procesul nostru poate fi analizat numai prin dependență intrare-iesire pe care o posedă. Deci, acest model va fi estimat cu ajutorul minimizării unei funcții criteriu ce ne va asigura că modelul nostru simulat este asemănător cu cel real.

În acest context, putem formula problema fundamentală prin prisma Teoriei Optimizării. În urma analizei asupra Teoriei Optimizării, un experiment de identificare urmărește următoarele maniere pentru a putea construi un model matematic:

- Analizarea informațiilor preliminare
 - Organizarea experimentului
 - Achiziția preliminară a datelor și prelucrarea acestora
- Alegerea clasei de modele a modelului specific
- Alegerea metodei de identificare
- Determinarea modelului adecvat
- Validarea modelului adecvat

Căutarea modelului implică o evaluare comparativă între diferite modele structurale, prin prisma unui criteriu de adecvare. Aceste modele sunt de indici structurali diferenți. Se evaluatează diferite modele cu structuri diferențiate cu același criteriu de optimizare. Din toate modelele posibile, se va alege cel care are valoarea optimă a criteriului de adecvare. Criteriul de adecvare poate fi de maximizare sau de minimizare.

Există diferite tehnici de căutare a optimului. Tehnicile inspirate din *Teoria Optimizării* pot fi de tipul metodelor clasice de optimizare (MCMMP) sau de tipul metodelor neconvenționale care se bazează pe metaeuristici (algoritmi inspirați din biologie). Rezultatul va reprezenta un model matematic adecvat, valid și optim pentru procesul de fată.

Cele trei direcții ale domeniului de identificare sunt:

- Modelele parametrice
- Semnalele de intrare
- Metodele de identificare

3.2 Modele parametrice

Modelele matematice de identificare sunt modele parametrice, unde, după cum se vede și în denumire, conceptele principale sunt **parametrii**. Cele mai folosite modele parametrice le reprezintă cele din clasa **ARMAX**. În cadrul acestei lucrări, folosim identificarea parametrică pentru modele din clasa **ARMAX**.

Forma generală a unui model ARMAX intrare-iesire este reprezentată în:

$$s_i y[n] = q^{-nk} G(q^{-1}) u[n] + H(q^{-1}) e[n], \quad (1)$$

unde $u[n]$ și $y[n]$ reprezintă valorile semnalelor de intrare și ieșire ale modelului la momentul n de timp, $e[n]$ reprezintă perturbatia aplicată sistemului, ce va fi considerată un zgomot alb de dispersie nulă. Întârzierea sistemului este indicată de către n_k , iar $G(q^{-1})$ reprezintă funcția părții utile a sistemului și $H(q^{-1})$ reprezintă funcția părții nedeterministe a sistemului.

$$G(q^{-1}) = \frac{C(q^{-1})}{D(q^{-1})} H(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} \quad (2)$$

Modelele din clasa **RSISO** extind clasa generală ARMAX, incluzând posibilitatea de a specifica poli diferenți în cadrul filtrelor de sistem și de zgomot. Forma unui model din clasa **RSISO** este:

$$\begin{cases} A(q^{-1})y[n] = \frac{B(q^{-1})}{F(q^{-1})}u[n] + \frac{C(q^{-1})}{D(q^{-1})}e[n] \\ E\{e[n]e[m]^T\} = \lambda^2 \delta_0(n-m), \end{cases} \quad (3)$$

unde λ^2 reprezintă dispersia necunoscută a zgomotului alb și δ_0 este impulsul unitar discret, iar q^{-1} reprezintă operatorul de întârziere temporală.

În continuare, vom avea definite polinoamele modelului astfel:

$$\begin{cases} A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \dots + a_{na} q^{-na}; \\ B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} + \dots + b_{nb} q^{-nb}; \\ F(q^{-1}) = 1 + f_1 q^{-1} + f_2 q^{-2} + \dots + f_{nf} q^{-nf}; \\ C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \dots + c_{nc} q^{-nc}; \\ D(q^{-1}) = 1 + d_1 q^{-1} + d_2 q^{-2} + \dots + d_{nd} q^{-nd}, \end{cases} \quad (4)$$

Clasa RSISO cuprinde, aşa cum este precizat anterior, o gamă largă de modele parametrice. Acestea se pot clasifica în:

- **ARMAX** - Acronimul provine de la **AutoRegresiv cu Medie Alunecătoare și control eXogen**. În această clasă de modele, polinoamele parametrice $F(q^{-1})$ și $D(q^{-1})$ sunt egale cu **1**, aşadar funcția sistemului a părții utile va avea poli identici cu cei ai funcției sistemului a părții stocastice.

$$A(q^{-1})y[n] = B(q^{-1})u[n] + C(q^{-1})e[n] \quad (5)$$

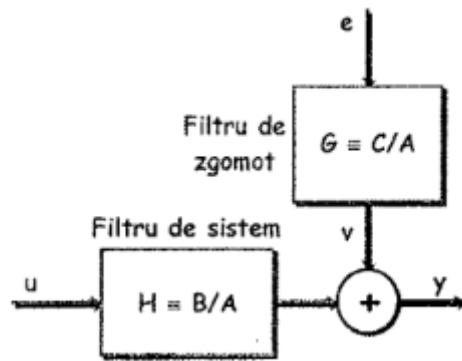


Figure 3: Model Structural ARMAX [5]

- **ARMA** - Acronimul provine de la **AutoRegresiv cu Medie Alunecătoare**. Reprezintă un caz particular al clasei de modele ARMAX, în care avem semnalul de comandă $u[n] = 0$, ceea ce conduce la "dispariția" din structură a polinoamelor $B(q^{-1})$ și $F(q^{-1})$.

$$A(q^{-1})y[n] = C(q^{-1})e[n] \quad (6)$$

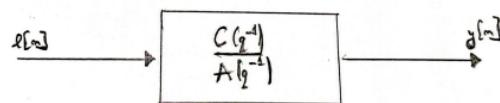


Figure 4: Model Structural ARMA

- **ARX** - Acronimul provine de la **AutoRegresiv cu control eXogen**. Este de asemenea un caz particular al clasei de modele ARMAX, în care valoarea polinomului $x(q^{-1})$ este egală cu **1**.

$$A(q^{-1})y[n] = B(q^{-1})u[n] + e[n] \quad (7)$$

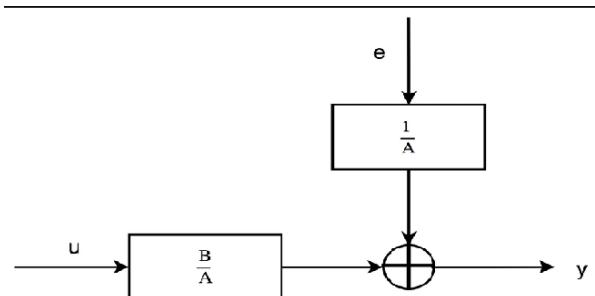


Figure 5: Model Structural ARX

Pentru a putea reuși identificarea modelului parametric valid și adekvat, este crucial să determinăm numărul și valorile parametrilor necunoscuți.

Modelele au fost obținute printr-o tehnică de optimizare neconvențională (metaeuristici). Din acestea, au fost folosite următoarele pentru a obține modelele parametrice necesare realizării proiectului:

- Metoda Celor Mai Mici Pătrate (MCMMP) - folosită pentru partea utilă a modelelor ARX
- Metoda Minimizării Erorii de Predictie (MMEP) - folosită pentru modelele ARMA, adică partea de zgromot.

Indicii strucuturali ai modelului au fost determinați prin optimizare metaeuristică. Așadar, pentru a soluționa problema modelului parametric adekvat și valid, vor fi implementate tehnici de optimizare, care vor fi prezentate în capituloarele următoare.

4 OPTIMIZARE METAEURISTICĂ

Identificarea unui model poate fi realizată dacă se cunosc indicii structurali ai modelului. Adică, metodele MMEP (Metoda Momentelor Efective de Proces) și MCMMMP (Metoda Criteriului de Minimizare a Modelului Parametric) se aplică pentru indicii structurali cunoscute. Deoarece nu cunoaștem indicii structurali ai modelului, nu avem prea multe informații despre "cutia neagră" care reprezintă funcționarea procesului; astfel, indicii structurali pot fi aleși într-un anumit domeniu de variație. În plus, deoarece datele măsurate sunt afectate de zgomot, este necesară utilizarea metaeuristicilor pentru găsirea indicilor structurali optimi ai modelului.

Astfel, ne dorim să găsim un model optim din punct de vedere al unui criteriu de adevarare care, implicit, să conducă la găsirea indicilor structurali optimi. Avantajul major al tehniciilor de optimizare metaeuristice este că oferă soluții satisfăcătoare într-un timp mai redus decât o căutare exhaustivă, care poate dura mult mai mult. În acest fel, se obține un compromis destul de bun între precizia modelului și timpul de găsire al aceluia model.

4.1 Prezentare Metauristică

Termenul de *euristică* provine din limbajul antic grecesc, unde *heuriskein* referă la acțiunea de a **descoperi**. Euristică se bazează pe căutarea unui optim într-un spațiu de căutare dat. Diferența între căutarea exhaustivă și cea heuristică este că, în timp ce prima analizează tot spațiul de căutare și găsește soluția globală după un timp îndelungat, cea din urmă analizează o mică parte din spațiul de căutare prin metode strategice, astfel încât, într-un timp redus de căutare, să fie găsit un individ suficient de bun pentru rezolvarea problemei de optimizare propusă.

Putem defini problema de optimizare sub forma unei funcții cost.

$$f : \mathbf{S} \rightarrow \mathbb{R}, \quad (8)$$

Unde S este spațiul de căutare și f este funcția de fitness.

Scopul este de a găsi optimul în spațiul de căutare predefinit. Funcția cost f împarte spațiul de căutare în mai multe subspații pentru a forma mai multe optimuri locale. De aceea, problema se poate rezuma la căutarea optimului doar într-un subspațiu, denumit D . Astfel, putem scrie problema sub forma:

$$\text{opt}_{x \in D \subseteq S} f$$

unde x reprezintă multimea de puncte din spatiul D în care noi vom căuta optimul.

Având în vedere că optimul va fi găsit, căutarea trebuie efectuată în timp rezonabil. Punctele de căutare ale lui D formulează o problemă de granulare optimă. Pentru rezolvarea problemei, trebuie găsită granula lui D care este localizată cel mai aproape de optimul global al lui f .

În general, toate metaheuristicile folosesc un motor pseudo-aleator pentru a selecta câțiva parametri sau operații care tind spre estimarea soluției optime. De aceea, procedura prin care se generează secvențe pseudo-aleatoare este importantă în design-ul metaheuristicilor. În contextul pseudo-aleator, trebuie luată în considerare densitatea de probabilitate ce trebuie să constituie o prioritate. Utilizarea numerelor pseudo-aleatoare în combinație cu metaheuristicile face imposibilă formularea unui rezultat general privind convergența. Pentru aceasta, trebuie să impunem conjecturi pentru acele metaheuristică ce pot avea cel mai mult succes în aplicare.

– **Locale:**

- Criteriul are un optim global.
- Căutarea este restricționată în subseturi înguste ce conțin optimul global.

– **Globale:**

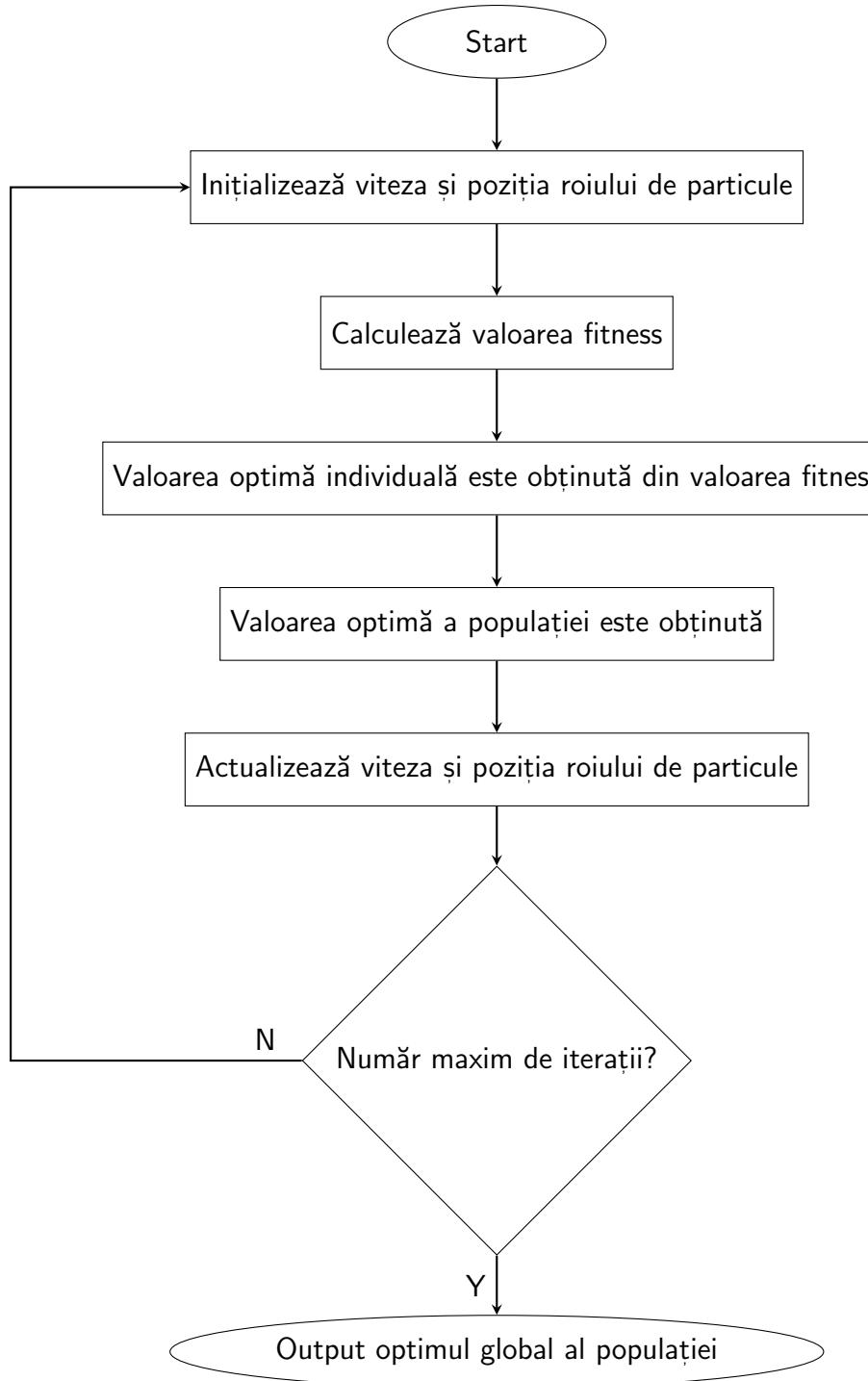
- Găsirea optimului general (sau o aproximare bună a acestuia) prin performarea unei căutări în anumite zone ale lui S .
- Găsirea formează un scenariu în care își permite să își schimbe focusul la alte zone în mod instantaneu, pentru a evita atracția degajată de optimul global.

4.2 Particle Swarm Optimization (PSO)

Principiul general al PSO (Particle Swarm Optimization) presupune o populație de particule ce este ghidată de o conștiință cognitivă specifică a fiecărei particule în parte și, în același timp, urmează o soluție optimă.

- La început, fiecare particulă este localizată la întâmplare în spațiul de căutare, cu o viteză selectată în aceeași manieră.
- Fiecare particulă este capabilă să evalueze calitatea poziției sale și să păstreze în memorie performanța cea mai bună. Toate particulele formează o expresie a unei conștiințe cognitive.
- La fiecare iterație, fiecare particulă poate întreba celelalte particule despre poziția și performanța stocată. Astfel, particulele devin conștiente de gândirea cognitivă a roiului.
- La fiecare iterăție, fiecare particulă își schimbă poziția și viteza în funcție de informațiile adunate.
- La sfârșit, particulele se aglomerează într-o mică vecinătate a unei particule, apoi se va optimiza punctul optim pentru criteriul presetat.

Pașii pentru a implementa PSO pot fi descriși de următoarea diagramă:



Modelul dinamic al comportamentului particulelor este :

$$x_p^{k+1} = x_p^k + v_p^k \Delta T_p^k, \forall k \in \mathbb{N} \quad (9)$$

Unde :

- x_p^k - reprezintă poziția particulei p la iterată k
- v_p^k - reprezintă viteza constantă a particulei p la iterată k

- ΔT_p^k - reprezintă timpul particulei p la iterația k pentru a se misca

În acest context, modul în care calculăm viteza și durata de timp este crucial. De obicei, $\Delta T_p^k = 1$ pentru fiecare particulă sau poate fi selectat aleatoriu. Problema apare deoarece nu putem garanta că particula rămâne în S după durata de timp, aşa că implementăm o tehnică prin care readucem particula în S:

- 1. Definim viteza v_p^k
- 2. Selectam $\Delta T_p^k \in [0, T]$
- 3. Cat timp $x_p^k + v_p^k \Delta T_p^k \in S$
 - alegem un alt $\Delta T_p^k \in [0, \Delta T_p^k]$

Prin acest mod, ΔT_p^k scade gradual, până când este validată condiția de apartenență în spațiul de căutare. O altă tehnică constă în asigurarea că poziția este viabilă, ce se realizează prin cunoașterea variației poziției curente $(x_{p,i}^{min}, x_{p,i}^{max})$, definită prin:

$$x_{p,i}^k = \min\{\max\{x_{p,i}^k + v_{p,i}^k \Delta T_p^k, x_{p,i}^{min}\}, x_{p,i}^{max}\}; \quad (10)$$

4.3 Tehnici de Menținere a Particulelor în Spațiul de Căutare

Tehnica asigură că acele componente "rebele" care ies din spațiul de căutare se comportă conform dorințelor, adică li se validează poziția.

O tehnică viabilă este combinarea celor două metode anterioare, prin alegerea unui ΔT_p^k care să nu fie constant și care să nu fie general valabil, adică să fie unic pentru fiecare particulă.

Pentru a implementa aceste două tehnici, trebuie să găsim o ecuație în care să se realizeze actualizarea vitezei particulelor. Aceasta poate fi scrisă astfel:

$$v_p^{k+1} = \mu_p^k v_p^k + \lambda_{p,c}^k v_{p,c}^k + \lambda_{p,s}^k v_{p,s}^k, \forall p \in [1, P], \forall k \in \mathbb{N} \quad (11)$$

Unde avem

- μ_p^k - se numește factor de mobilitate
- $\lambda_{p,c}^k$ - se numește conservativa particulei
- $v_{p,c}^k$ - este viteza către cea mai bună poziție
- $\lambda_{p,s}^k$ - oferă particulei motiv să urmeze un drum ales de informatorii
- $v_{p,s}^k$ - este viteza către optimul social, după informatorii

Calculul vitezei particulei reprezintă nucleul acestei metaeuristici. Pentru aceasta, trebuie aleasă o metodă de selecție a informatorilor care oferă roiului date despre pozițiile lor. În

general, selecția informatorilor se schimbă de fiecare dată când cea mai bună performanță a întregului roi nu s-a îmbunătățit. Odată ce o poziție optimă a fost găsită, viteza este actualizată prin observarea tendinței particulei respective. O particulă poate avea trei tendințe de a se comporta:

- aventuroasă - continuă drumul cu viteza curentă
- conservativă - merge în direcția particulei cu cea mai bună poziție găsită
- panurgiană - urmează orbește direcția către punctul optimal, așa cum îi indică informanții

O schemă care să succinte clasificarea particulelor este prezentată mai jos:

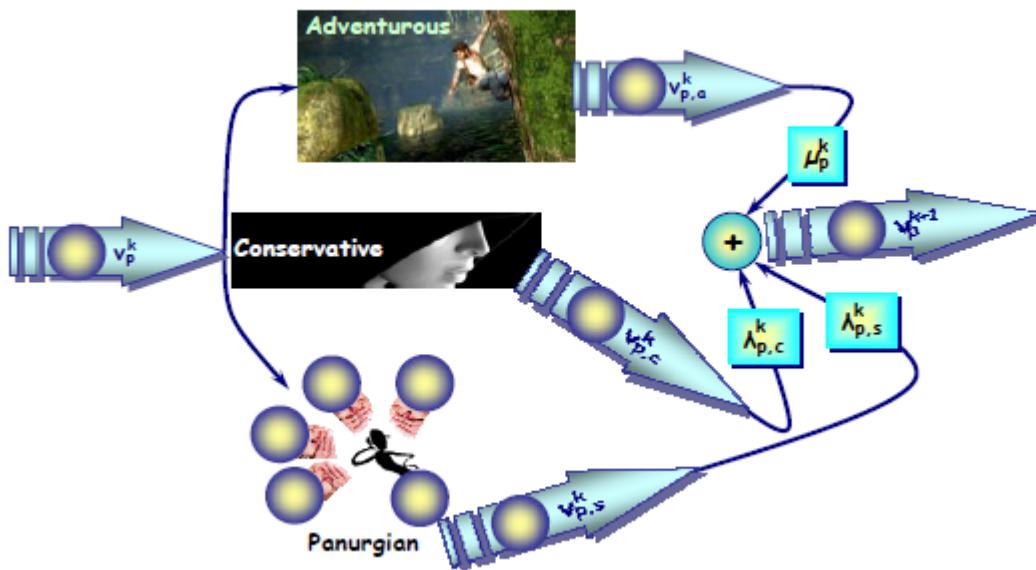


Figure 6: Tendințele particulelor [3]

Ne mai rămâne de analizat modul în care alegem informatorii roilui de particule. Pentru aceasta, se definește grupul de informatori prin ϵ_p^k .

Selectia de informatori se schimbă la fiecare iterare. Atunci, putem realiza selecția prin:

- Conservăm cel mai bun roi pentru a fi pasat următorului grup de informanți.
- Celelalte $P-1$ poziții din ϵ_p^{k+1} sunt vacante și trebuie înlocuite de particule din roi, depinzând de șansa fiecărui competitor și de un threshold h_s cunoscut. În general, h_s este ales cu valoarea 0.75.

În cadrul fiecărei iterării, pentru a elibera locuri în grupul de informatori, este necesară organizarea unei competiții. Această competiție constă în următoarele etape:

- Pentru fiecare particulă din roi, aleatoare, se alege o valoare $h \in [0, 1]$.

- Dacă $h \geq h_s$, atunci particula câștigă și ia locul vacant.
- Dacă nu, particula este refuzată în acest moment, dar poate reîntră în competiție în alt moment.

Problema revine la ideea principală de a aplica o metodă metaeuristică în buclă închisă. Acest lucru nu poate fi făcut decât în mod adaptiv. Valorile parametrilor algoritmului PSO trebuie să fie adaptate la fiecare iterare pentru a putea găsi un optim global într-un timp acceptabil.

Tehnica de PSO adaptiv cu strategie evoluționară ajută la menținerea balansului dintre diversitatea și capacitatea roiului de a converge către optimul căutat.

Pentru aceasta, parametrii de configurare principali, adică μ , λ_c , λ_s , nu pot fi setați ca și constante și, mai mult decât atât, nu trebuie determinați empiric. Plecând de la ideea că parametrii de mai sus ce definesc inertă, conștiința cognitivă și socială a particulei sunt concepte naturale în PSO, este indicat să le cuantificăm într-o manieră adaptivă, ce depinde direct de dinamica roiului.

Fie $p \in [1, P]$ indexul unei particule alese arbitrar al roiului și un $k \in \mathbb{N}$ reprezentând indexul iterării curente.

În încercările sale aventuroase, dinamica particulei ar trebui să fie condiționată de un parametru de mobilitate, μ_p^k , ce este opus de inertă particulei η_p^k . Relația dintre cele două este:

$$\mu_p^k = 1 - \eta_p^k; \quad (12)$$

Cu cât particula este mai departată de poziția cea mai bună, cu atât inertă este mai mică și, în consecință, mobilitatea crește. Particula este motivată să se mișeze mai rapid, fiind atrasă mai puternic de către poziția sa cea mai bună. În contrast, particula mai aproape de poziția sa cea mai bună nu are motive întemeiate să se îndepărteze și este tentată să rămână lipită de zona respectivă.

Așadar, în consecință, inertă relativă a unei particule poate fi definită astfel:

$$\eta_p^k = \frac{f(x_p^k) - f(x_p^{tpo,k})}{f(x_p^{opt,k}) - f(x_p^{tpo,k})} \in [0, 1], \forall p \in [1, P], \forall k \in \mathbb{N}. \quad (13)$$

Unde avem definite, încă din notările cunoscute:

- $x_p^{tpo,k}$ - reprezintă cea mai "rea" poziție a particulei; acronimul "tpo" provine de la prescurtarea "opt" scrisă invers pentru a putea semnala o poziție opusă celei optimale.
- $f(x_p^{tpo,k})$ - reprezintă cea mai "rea" valoare a criteriului de optimizare.
- $f(x_p^{opt,k})$ - reprezintă cea mai "bună" valoare a criteriului de optimizare.

Parametrii de configurare principali vor fi prezențați în continuare. Varianța cognitivă poate explica cum particula este asezată în apropierea sau în depărtarea drumului ce duce la punctul optimal. O posibilă definiție o consideră următoarea relație, ce preia valori relative la fel ca în cazul inerției:

$$\lambda_{p,c}^k = 2 \frac{\sigma_{p,c}^k - \sigma_{p,c}^{min,k}}{\sigma_{p,c}^{max,k} - \sigma_{p,c}^{min,k}} \in [0, 2], \forall p \in [1, P], \forall k \in \mathbb{N} \quad (14)$$

Unde avem :

- $\sigma_{p,c}^k$ - varianța cognitivă absolută a particulei
- $\sigma_{p,c}^{min,k}$ - varianța cognitivă minima a particulei
- $\sigma_{p,c}^{max,k}$ - varianța cognitivă maxima a particulei

Diferența între varianța minimă și maximă exprimă cât de mult este tentată particula să exploreze zona din jurul poziției celei mai bune (pentru o diferență mică) sau, în contrast, dacă particula este tentată să exploreze o zonă mai largă (pentru o diferență mare).

Putem defini varianța cognitivă absolută astfel:

$$\sigma_{p,c}^k = \frac{1}{k+1} \sum_{l=0}^k \|x_p^l - x_p^{opt,k}\|^2, \forall p \in [1, P], \forall k \in \mathbb{N} \quad (15)$$

Prin această definiție, putem cuantifica dispersia pozițiilor succesive ale particulelor pe drumul curent, $\{x_p^l\}_{l \in [0, k]}$, ținând cont de poziția curentă cea mai bună, $x_p^{opt,k}$.

Varianța relativă socială $\lambda_{p,s}^k$ exprimă intensitatea tendinței panurgiene a particulei, ce depinde de cât de concentrat sau cât de risipit este grupul de informanți ϵ_p^k în jurul punctului cel mai bun, $x_{\epsilon_p^k}^{opt,k}$. În cazul în care informanții sunt strâns grupați în jurul poziției lor celei mai bune, particula devine "sigură" pe informanți și este tentată să își mărească viteza panurgiană către ei. În alt caz, particula ar trebui să aibă dubii cu privire la poziția comunicată și, cu multă grijă, să își limiteze viteza panurgiană. Așadar, putem defini varianța relativă socială astfel:

$$\lambda_{p,s}^k = 2 \frac{\sigma_{p,s}^{max,k} - \sigma_{p,s}^k}{\sigma_{p,s}^{max,k} - \sigma_{p,s}^{min,k}} \in [0, 2], \forall p \in [1, P], \forall k \in \mathbb{N} \quad (16)$$

Unde

- $\sigma_{p,s}^k$ - varianța absolută socială a grupului de informanți
- $\sigma_{p,s}^{min,k}$ și $\sigma_{p,s}^{max,k}$ sunt valorile minime și maxime a variantei.

Varianța absolută socială este definită astfel:

$$\sigma_{p,s}^k = \frac{1}{P_i} \sum_{r=1}^{P_i} \|x_r^k - x_{\epsilon_p^k}^{opt,k}\|^2, \forall p \in [1, P], \forall k \in \mathbb{N} \quad (17)$$

Aceasta din urmă cuantifică modul în care informanții sunt dispersați în jurul poziției celei mai bune. Diferit față de varianța absolută cognitivă, în acest caz doar poziția curentă a informanților este luată în considerare, din moment ce doar dispersia actuală prezintă interes.

Roiul global de particule, ϵ , evoluează către un punct optim, aşa cum este sugerat în imaginea de mai jos:

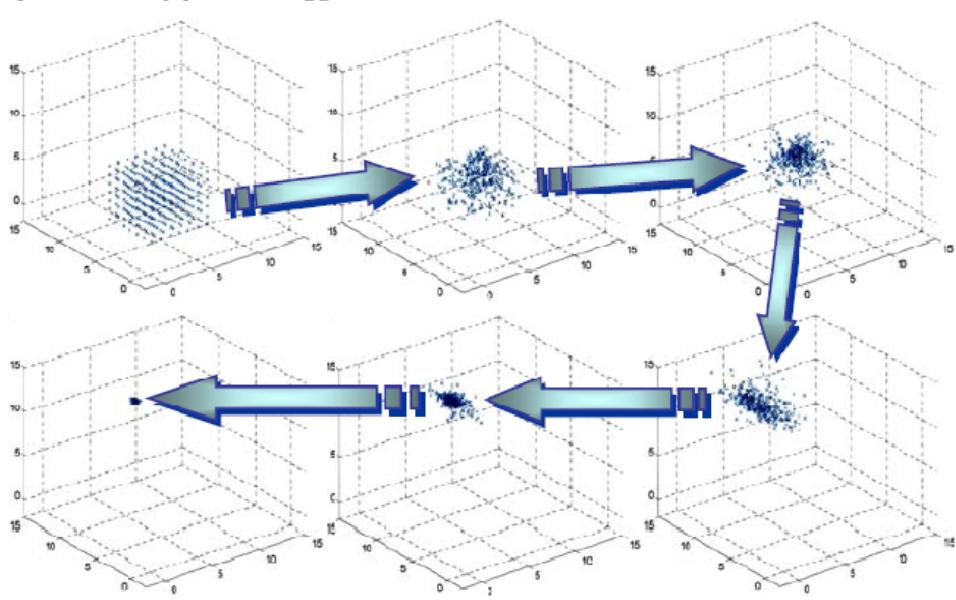


Figure 7: Principiul optimizării prin roiul de particule [3]

În timpul evoluției, contrar principiului general prezentat în figura de mai jos, dispersia roialui poate varia în numeroase moduri, ajungând chiar la a oscila. În figura prezentată, dispersia roialui doar descrește pe măsură ce particulele se grupează în jurul unui punct optim. Astfel, trebuie să definim varianta roialui care poate controla diversitatea particulelor.

$$\sigma_{\epsilon}^k = \frac{1}{P} \sum_{p=1}^P \|x_p^k - x_{\epsilon,q}^{opt,k}\|^2, \forall k \in \mathbb{N} \quad (18)$$

, unde $x_{\epsilon,q}^{opt,k}$ reprezintă poziția curentă optimă a întregului roi de particule, în timp ce $q \in [1, P]$ reprezintă indexul unei particule ce a atins această poziție.

Varianța roialui poate fi monitorizată cu scopul de a detecta tendințele roialui în sensul detectării dispersiei în jurul punctului curent optim. Pe de o parte, dacă varianța crește semnificativ după câteva iterații, atunci roial tinde să devină prea risipit, ceea ce conduce la dominarea explorării în fața exploatarii. Pe de altă parte, dacă varianța scade în mod sensibil după câteva iterații, atunci roial va fi prins într-o vecinătate îngustă a punctului curent optim, ceea ce nu coincide de fiecare dată cu punctul global optim. Acest lucru conduce la dominarea exploatarii în favoarea explorării.

Pentru a evita scenariile prezentate anterior, trebuie proiectată o alarmă care să trimită semnal

de fiecare dată când varianța relativă a roiului se schimbă brusc după doar câteva iterații, ceea ce impune setarea unui threshold. Varianța relativă a roiului poate fi definită astfel:

$$\rho_{\epsilon}^k = \frac{\sigma_{\epsilon}^k - \sigma_{\epsilon}^{k-1}}{\sigma_{\epsilon}^{k-1}}, \forall k \in \mathbb{N} \quad (19)$$

5 CONTROL PID MULTIVARIABIL

În această etapă, urmărim proiectarea unei legi de control care, în bucla închisă, să ne valideze indicii de performanță doriti, ceea ce înseamnă suprareglaj cât mai mic și atingerea erorii staționare nule. Condițiile vor fi analizate într-un proces ce primește o referință de tip treaptă. De asemenea, trebuie asigurat că sistemul respinge perturbațiile de diferite tipuri (treaptă, impuls) ce apar în timpul funcționării. [2]

5.1 Legea de control PID

Schema de control al sistemului reprezintă o structură de reglare formată din referință, proces și regulator. Schema clasică a unui proces cu o singură intrare și o singură ieșire este prezentată mai jos:

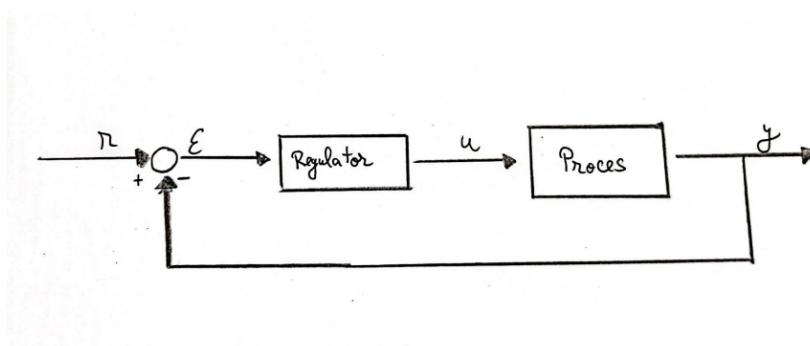


Figure 8: Structura clasică de reglare

În acest proiect, vom opta pentru folosirea unui regulator PID-R

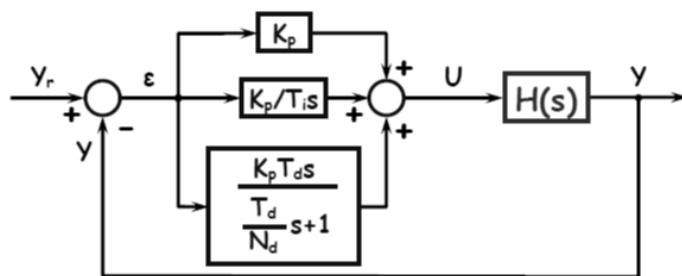


Figure 9: Structura Proportional Derivativ Integrativ [4]

Este bine cunoscut că pentru un PID, comanda se calculează pe baza erorii de referință:

$$u(t) = K_p \left(\epsilon(t) + \frac{1}{T_i} \int_0^t \epsilon(\tau) d\tau + T_d \frac{d\epsilon(t)}{dt} \right) \quad (20)$$

Putem reprezenta eroare de reglare prin formula: $\epsilon = y_r - y$, unde y_r definește referința aplicată regulatorului, iar y reprezintă ieșirea din proces.

Funcția de transfer a PID-ului este următoarea:

$$H_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \quad (21)$$

Vom aplica un element de filtrare constant funcției de transfer pentru a exclude problemele de realizare fizică întâmpinate de componenta derivativă T_d . În urma schimbărilor, putem defini funcția de transfer în modul:

$$H_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N_d} s + 1} \right) \quad (22)$$

N_d reprezintă, în cazul nostru, constanta de filtrare, setată prealabil la valoarea de 10.

Componentele regulatorului PID contribuie în moduri diferite asupra controlului instalațiilor.

Componenta proporțională K_p generează o ieșire direct proporțională cu eroarea curentă $\epsilon(t)$. Cu cât eroarea este mai mare, cu atât ieșirea proporțională va crește. Aceasta corectează deviațiile într-un timp rapid. Totodată, un câștig K_p mai mare reduce eroarea staționară, dar poate provoca un suprareglaj mărit. Iar dacă valoarea este foarte mare, poate duce chiar la instabilitatea sistemului.

Componenta integrativă $\frac{1}{T_i s}$ elimină eroarea staționară, compensând efectele generate de către cea proporțională. Funcția să este de a acumula erorile din trecut și de a "grăbi" sistemul în a ajunge la valorile dorite. În cazurile în care contribuția integrativă devine prea mare, aceasta poate cauza ca răspunsul sistemului să conțină suprareglaj, dar și oscilații. Acest fapt este datorat încercării de a corecta erorile trecute chiar și după ce sistemul se apropiște de referință.

Componenta derivativă $\frac{T_d s}{\frac{T_d}{N_d} s + 1}$ atenuază oscilațiile și reduce suprareglajul. Aceasta se bazează pe rata de schimbare a derivatei erorii în timp. Acționează ca un amortizor, încetinind evoluția sistemului în cazul intervenirii oscilațiilor.

5.2 Regulator PID Multi-variabil

Pe instalația ASTANK2, fiind un proces multi-variabil, am ales să implementez un regulator PID. Procesul poate fi controlat cu ajutorul unui regulator ce trebuie să urmărească 2 referințe simultan. Asadar, vom implementa un regulator PID multi-variabil care va analiza cele 2 erori $\epsilon_1 = y_{ref1} - y1$ și $\epsilon_2 = y_{ref2} - y2$ și va furniza comenzi folosite de către proces, mai precis

u_1 și u_2 . Valorile y_{ref1} și y_{ref2} reprezintă referințele de pe cele 2 intrări, adică cele 2 niveluri de apă din bazinele superioare. Prin utilizarea acestui model de regulator, componente sale nu vor fi valori numerice, ci vor deveni matrici de dimensiuni 2x2 ce conțin valori numerice:

$$K_p(s) = \begin{bmatrix} K_p^{11} & K_p^{12} \\ K_p^{21} & K_p^{22} \end{bmatrix} \quad (23)$$

$$T_i(s) = \begin{bmatrix} \frac{1}{T_i^{11}s} & \frac{1}{T_i^{12}s} \\ \frac{1}{T_i^{21}s} & \frac{1}{T_i^{22}s} \end{bmatrix} \quad (24)$$

$$T_d(s) = \begin{bmatrix} \frac{T_d^{11}s}{\frac{T_d^{11}}{N_d}s+1} & \frac{T_d^{12}s}{\frac{T_d^{12}}{N_d}s+1} \\ \frac{T_d^{21}s}{\frac{T_d^{21}}{N_d}s+1} & \frac{T_d^{22}s}{\frac{T_d^{22}}{N_d}s+1} \end{bmatrix}$$

Modelul instalației este discret, așa că vom discretiza regulatorul multi-variabil PID. Vom folosi astfel transformarea Tustin, ce ne ajută să conservăm stabilitatea sistemului. Așadar, un pol continuu s îi va corespunde un pol stabil în planul discret z :

$$s = \frac{2}{T_s} \frac{z - 1}{z + 1}. \quad (25)$$

T_s reprezintă perioada de eşantionare, aleasă convenabil ca 1.

Analizând formele matriciale ale componentelor PID-ului, putem observa faptul că nu trebuie aplicată transformarea Tustin componentei proportionale, K_p , fiindcă aceasta conține doar valori numerice. Transformarea va fi aplicată deci doar componentelor integrative și derivate, astfel:

$$\frac{1}{T_i s} \Rightarrow \frac{T_s}{2T_i} \frac{z - 1}{z + 1} \quad (26)$$

pentru componenta integrativă, iar

$$\frac{T_d s}{\frac{T_d}{N_d} s + 1} \Rightarrow \frac{2T_d N_d (z - 1)}{(N_d T_s + 2T_d) z + (N_d T_s - 2T_d)} \quad (27)$$

pentru componenta derivativă.

5.3 Implementare în MATLAB-SIMULINK

Pentru a simula controlul instalației ASTANK2, vom implementa schema de control cu ajutorul MATLAB-SIMULINK.

Parametrii instalației se vor introduce în blocuri de tipul discrete transfer function”, unde vom pune coeficienții modelelor ARX și ARMA (pentru filtrele de zgomot) sub formă de vector, dar vom și specifica perioada de eșantionare T_s . Astfel, vom avea definite următoarele polinoame:

$$A_{11}, A_{12}, A_{21}, A_{22}; B_{11}, B_{12}, B_{21}, B_{22} \quad (28)$$

pentru ARX, iar pentru ARMA:

$$C_1, D_1, C_2, D_2. \quad (29)$$

Filtrele de zgomot ce sunt alcătuite din $\frac{C_1}{D_1}$ și $\frac{C_2}{D_2}$ vor primi fiecare un semnal de intrare de la către un generator de zgomot alb (Band-Limited White Noise și bloc în Simulink). Generatoarele au menționate puterea zgomotului, definită ca λ_1^2 și λ_2^2 și perioada de eșantionare T_s .

Componenta proporțională K_p este reprezentată în schema de reglare printr-un bloc **Gain**, ce este setat pe multiplicare matricială de forma $K \cdot u$. **K** este reprezentat de către matricea componentei proportionale, K_p , iar **u** reprezintă un vector rezultat din compunerea celor 2 referințe, conectate împreună cu un MultiPlexor. Componentele derivate și integrative sunt reprezentate de către blocurile de tip ”LTI System”, ce acceptă ca și parametri matricile T_{dz} și T_{iz} , ce reprezintă discretizările cu Tustin aplicat fiecărui element matricial în parte din $T_d(s)$ și $T_i(s)$.

Așadar, avem definite următoarele:

$$T_i(z) = \begin{bmatrix} \frac{T_s}{2T_j^{11}} \frac{z+1}{z-1} & \frac{T_s}{2T_j^{12}} \frac{z+1}{z-1} \\ \frac{T_s}{2T_j^{21}} \frac{z+1}{z-1} & \frac{T_s}{2T_j^{22}} \frac{z+1}{z-1} \end{bmatrix} \quad (30)$$

, reprezentând componenta integrativă discretizată, și

$$T_d(z) = \begin{bmatrix} \frac{2T_d^{11}N_d(z-1)}{(N_dT_s+2T_d^{11})z+(N_dT_s-2T_d^{11})} & \frac{2T_d^{12}N_d(z-1)}{(N_dT_s+2T_d^{12})z+(N_dT_s-2T_d^{12})} \\ \frac{2T_d^{21}N_d(z-1)}{(N_dT_s+2T_d^{21})z+(N_dT_s-2T_d^{21})} & \frac{2T_d^{22}N_d(z-1)}{(N_dT_s+2T_d^{22})z+(N_dT_s-2T_d^{22})} \end{bmatrix} \quad (31)$$

pentru componenta derivativă discretizată.

Odată ce am stabilit parametrii pentru fiecare componentă a schemei de reglare, simularea se va aplica pe:

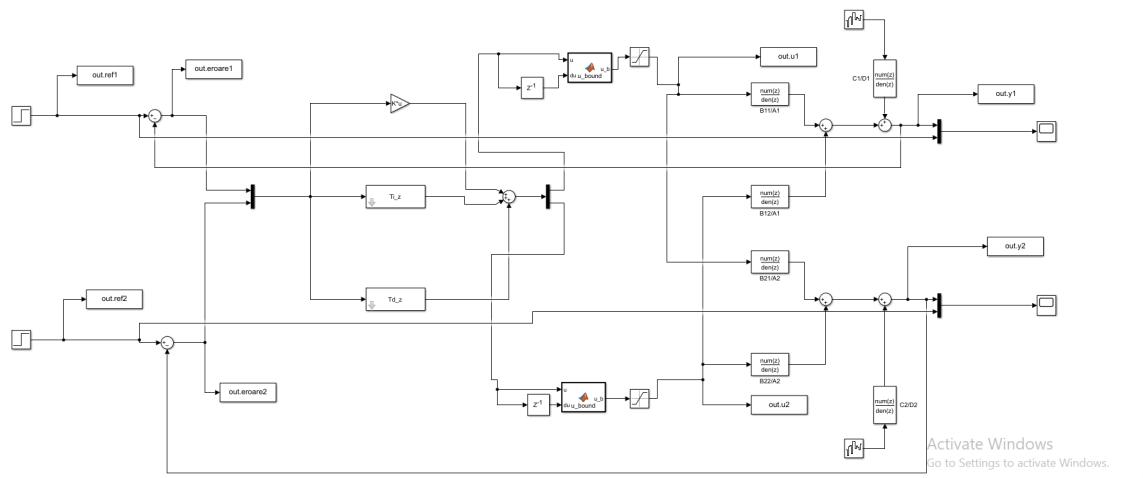


Figure 10: Structura Simulink PID

Din cauza limitărilor prezente în instalație (mai precis ale electrovalvelor), trebuie să saturăm comenziile înainte de a intra în sistem. Așadar, vom introduce o saturare pe fiecare comandă ce pleacă din regulator spre proces.

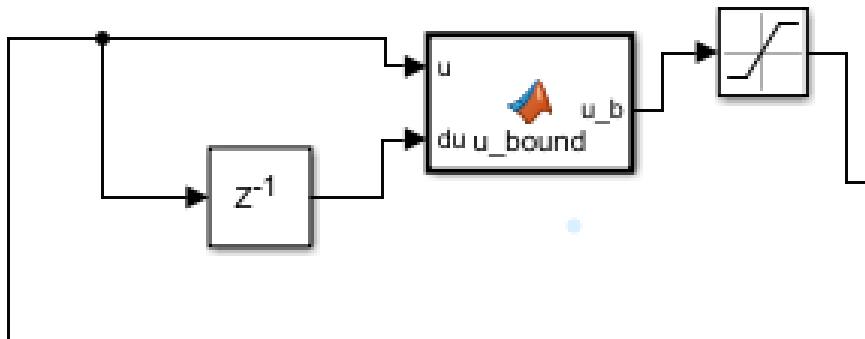


Figure 11: Saturatie pe comanda

Vom introduce o *Matlab Function* denumită u_{bound} ce implementează restricția ca diferența dintre două comenzi succesive să nu fie mai mare decât $\pm 3V$. Acest lucru este necesar deoarece electrovalvele nu suportă salturi mai mari decât $\pm 3V$ între două comenzi succeseive.

6 STUDIU DE CAZ

6.1 Funcția fitness

Funcția fitness va fi aleasă că urmare a structurii parametrice identificate. Vom utiliza partea utilă a acesteia, adică:

$$\begin{cases} A_1(q^{-1})y_1[n] = B_{11}(q^{-1})u_1[n] + B_{12}(q^{-1})u_2[n] + v_1[n] \\ A_2(q^{-1})y_2[n] = B_{21}(q^{-1})u_1[n] + B_{22}(q^{-1})u_2[n] + v_2[n] \end{cases} \quad (32)$$

Vom crea zgromotul rezidual colorat prin trecerea unui zgromot alb printr-un model de tip **ARMA**. Așadar, putem defini următoarele:

$$\begin{cases} v_{r1}[n] = \frac{C_1(q^{-1})}{D_1(q^{-1})}e_1[n] \\ E\{e1[n]e1[m]^T\} = \lambda^2\delta_0[n-m] \\ v_{r2}[n] = \frac{C_2(q^{-1})}{D_2(q^{-1})}e_2[n] \\ E\{e2[n]e2[m]^T\} = \lambda^2\delta_0[n-m] \end{cases} \quad (33)$$

Deci, vom crea din modelul general de tip **MIMO 2x2** o structură ce conține 2 modele de tip **MISO 2x1** separate. Pentru a ne putea calcula parametrii optimi ai instalației, alegerea funcției fitness utilizată în algoritmul metaheuristic are un rol crucial. Asadar, în acest proiect, funcția fitness a fost aleasă ca suma ponderată a normelor respective erorilor generate de cele 2 modele **MISO** și de comenziile oferite fiecărui model în parte. De asemenea, în formarea funcției de fitness, vom lua în considerare doar valoarea semnalului de la eșantionul cu numărul N_k , acest lucru rezultând din observarea evoluției semnalului care, până la pragul de aproximativ N_k , crește în regim staționar, ceea ce ne indică faptul că minimizarea trebuie efectuată după acest prag. Totodată, asupra comenziilor blocurilor li s-a aplicat aceeași limitare în eșantioane, împreună cu o deviație standard cu scopul de a ajusta amplitudinea în folosirea acestuia în calculul funcției fitness.

$$\begin{cases} J_1 = \frac{\|e_1^k\|}{\sqrt{N_y - N_k}} \\ J_2 = \frac{\|e_2^k\|}{\sqrt{N_y - N_k}} \\ J_3 = std(u_1(N_k : N_y)) \\ J_4 = std(u_2(N_k : N_y)) \\ J = \alpha J_1 + \alpha J_2 + \beta J_3 + \beta J_4; \end{cases} \quad (34)$$

e_1^k și e_2^k reprezintă eroarea raportată de la momentul N_k până la terminarea simulării. Prin erori ne referim la diferența dintre referința aplicată și răspunsul indicial oferit de parametrii instalației. J_3 și J_4 reprezintă deviațiile standard ale semnalelor de comandă u_1 respectiv u_2 analizate de la momentul N_k până la sfârșitul simulării. N_y reprezintă lungimea semnalului de ieșire. Am ales $N_k = 100$ fiindcă acesta reprezintă aproximativ momentul la care ieșirea atinge valoarea de regim staționar.

Coeficienții ce sunt aplicați fiecărui semnal în parte au scopul de a da o pondere mai mare semnalelor pe care le dorim mai netede. Aceștia au proprietatea:

$$\alpha + \beta = 0.5 \quad (35)$$

Prin urmare avem definită funcția cost pe care algoritmul metaheuristic trebuie să o minimizeze. Scopul este de a aduce această funcție cost la o valoare subunitară cât mai mică.

6.2 Implementarea Algoritmului Metaheuristic

Algoritmul metaheuristic va fi implementat în această lucrare prin folosirea funcției din Matlab, **particleSwarm**.

Plecăm de la o populație de particule ce înmagazinează posibili parametri pentru rezolvarea minimizării funcției criteriu prestabilită. Fiecare particulă reprezintă un cromozom care conține un număr de alele ce conțin valoarea indicilor strucurali ai modelului parametric cerut.

Inițializarea algoritmului se realizează prin:

- Configurarea parametrilor inițiali
- Formarea mărimii populației de particule
- Implementarea unei funcții fitness care urmează a fi minimizată

Configurarea parametrilor inițiali constă în predefinirea variabilelor ce sunt implicate direct în procesul de optimizare. Inițial, trebuie prestabilite limitele spațiului de căutare pentru fiecare variabilă a PID-ului multivariabil. Așadar, avem limitele stabilite astfel:

- $K_p \in [a_p, b_p]$ - reprezintă intervalul K_p
- $T_i \in [a_i, b_i]$ - reprezintă intervalul T_i
- $T_d \in [a_d, b_d]$ - reprezintă intervalul T_d

Variabilele precedente pot fi schimbate pentru a mări sau a micșora intervalul de căutare al PSO-ului, ceea ce ne oferă o mobilitate în plus pentru a găsi optimul dorit. Valorile folosite pentru acest proiect sunt:

- $a_p = 0$
- $b_p = 10$
- $a_i = 1$
- $b_i = 100$
- $a_d = 0$
- $b_d = 10$

Totodată, putem să împărțim aceste intervale de căutare pentru a putea lucra în paralel pe mai multe computere, ceea ce reduce semnificativ timpul de așteptare necesar rularii algoritmului și, implicit, găsirii optimului. Această metodă se va realiza prin împărțirea la jumătate a intervalor parametrilor K_p , T_i și T_d . Așadar, vom avea 8 configurații posibile:

- C1: $[ap, (ap+bp)/2], [ai, (ai+bi)/2], [ad, (ad+bd)/2]$ și restul intervalelor integrale;
- C2: $[ap, (ap+bp)/2], [ai, (ai+bi)/2], [(ad+bd)/2, bd]$ și restul intervalelor integrale;
- C3: $[ap, (ap+bp)/2], [(ai+bi)/2, bi], [ad, (ad+bd)/2]$ și restul intervalelor integrale;
- C4: $[ap, (ap+bp)/2], [(ai+bi)/2, bi], [(ad+bd)/2, bd]$ și restul intervalelor integrale;
- C5: $[(ap+bp)/2, bp], [ai, (ai+bi)/2], [ad, (ad+bd)/2]$ și restul intervalelor integrale;
- C6: $[(ap+bp)/2, bp], [ai, (ai+bi)/2], [(ad+bd)/2, bd]$ și restul intervalelor integrale;
- C7: $[(ap+bp)/2, bp], [(ai+bi)/2, bi], [ad, (ad+bd)/2]$ și restul intervalelor integrale;
- C8: $[(ap+bp)/2, bp], [(ai+bi)/2, bi], [(ad+bd)/2, bd]$ și restul intervalelor integrale.

Intervallele ce vor fi analizate prin alegerea configurației sunt introduse într-un vector coloană numit *params_ranges*, ce conține pe linii fiecare interval pentru cei 12 parametri.

După definirea configurațiilor, trecem la stabilirea parametrilor ce sunt implicați în procesul funcției **particleSwarm**. Astfel, avem următoarele variabile definite:

- Numărul de variabile pentru fiecare particulă: $nVars = 12$, ce reprezintă câte 4 pentru fiecare componentă a PID-ului multivariabil (4 pentru K_p , 4 pentru T_i și 4 pentru T_d)
- Limita inferioară pentru fiecare variabilă $lb = params_ranges(:, 1)$, semnificând extagerea limitelor inferioare ale fiecărui parametru

- Limita superioară pentru fiecare variabilă $ub = params_ranges(:, 2)$, semnificând extagerea limitelor superioare ale fiecărui parametru
- Dimensiunea roiului $Swarmsize = 150$
- Numărul maxim de iterații admisibile $MaxIterations = 1000$
- Factorul de inerție al roiului $InertiaRange = [0.4, 0.9]$
- Coeficientul de accelerare $SelfAdjustmentWeight c_1 = 1.19$
- Coeficientul de accelerare $SocialAdjustmentWeight c_2 = 1.19$

Valorile coeficientilor menționați anterior au fost alese în conformitate cu documentația oferită ca și suport și modificate în urma iterațiilor anterioare celei finale. Aceste modificări au fost aduse cu scopul mării vitezei de convergență a algoritmului și scăderii timpului de așteptare.

Pentru a putea analiza erorile și comenziile ce intră în alcătuirea funcției de fitness, utilizăm o funcție care să fie folosită la fiecare iteratie a algoritmului PSO. Funcția atribuie noile valori ale componentelor regulatorului, sustrase din vectorul de variabile înmagazinat de fiecare particulă. După ce are loc atribuirea, se extrag valorile semnalelor de referință și comenziile, se calculează funcția fitness și se returnează metaeuristicii. Odată ce s-a găsit o particulă cu un fitness mai bun decât cel mai bun global precedent, roiul este informat și reîncepe căutările. Acest ciclu se repetă până când roiul nu a putut ajunge la o valoare mai bună timp de 20 de iterații succesive.

Apelul algoritmului PSO returnează un vector linie x ce conține valorile componentelor PID, dar și o valoare a criteriului de fitness, denumită J .

Dupa aplicarea algoritmului PSO, se obține:

$$J = 0.1681. \quad (36)$$

Această valoare poate fi transformată într-un fitness procentual, folosind formula:

$$F = \frac{100}{(1 + 0.65 * J)}, \quad (37)$$

unde F reprezintă în cazul de față fitnessul procentual, egal cu valoarea de 90.15%. Pentru că nu se cunoaște o limită superioară a lui J (el variind de regulă între 0 și infinit), formula funcției cost reprezintă o hiperbolă care comprimă intervalul $(0, \infty)$ în intervalul $(0, 1)$. Valoarea de 0.65 a fost aleasă astfel încât regulatoarele de calitate să conducă la valori ale fitnessului de peste 80%.

Parametrii optimi ce sunt returnați au forma:

$$K_p = \begin{bmatrix} 0.685 & 0.3563 \\ 0.4208 & 0.6849 \end{bmatrix} \quad (38)$$

$$T_i = \begin{bmatrix} 100 & 100 \\ 71.94 & 74.96 \end{bmatrix} \quad (39)$$

$$T_d = \begin{bmatrix} 0.00001 & 0 \\ 0.000008 & 0.0000006 \end{bmatrix} \quad (40)$$

Putem observa că elementele matricilor PID-ului optim găsit respectă configurația C3, cu fiecare parametru în limitele predefinite rularii algoritmului metaheuristic. Pentru găsirea regulatorului optim, algoritmul a rulat timp de aproximativ 30 de minute, timp în care a soluționat problema în 53 de iterări din cele 1000 posibile:

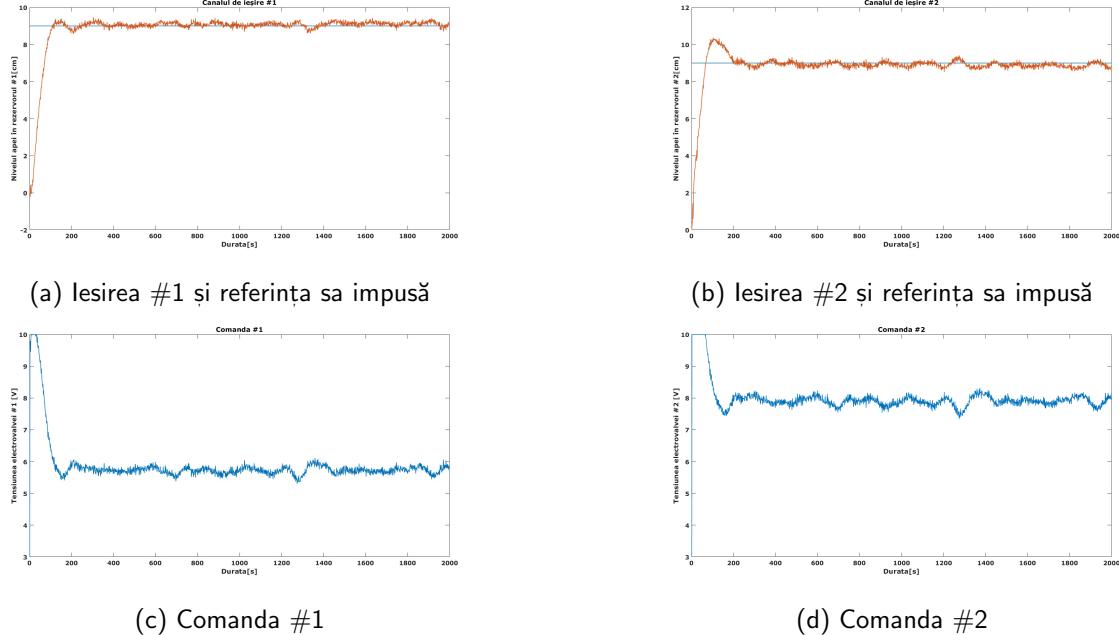


Figure 12: Comenzi și ieșiri în funcție de referințele impuse.

Se poate observa cum avem un suprareglaj destul de mare pe ieșirea #2, din cauza componente derivate care, în prealabil, se poate aproxima cu 0. Din această cauză, nu putem soluționa faptul că am găsit un PID optim, ci mai degrabă un PI optim. Urmărirea referinței este una bună, fără prea multe oscilații. Timpul de simulare este setat la 2000 [s], acesta neputând fi micșorat sub acest prag fiindcă nu am putea observa stabilizarea iesirilor în jurul referințelor.

Pentru a putea găsi un PID optim, trebuie să facem un compromis în defavoarea valorii fitnessului final pentru a putea găsi un regulator ce nu are componenta derivativă nulă. Acest lucru va duce la diminuarea suprareglajului de pe canalul 2.

Pentru a putea forța algoritmul să nu înlocuiască componenta derivativă cu 0, trebuie să impunem alt interval de căutare pentru T_d . Așadar, reluăm rularile cu:

- $a_d = 0$;
- $b_d = 4$;

Am scăzut limita superioară pentru a permite algoritmului să se concentreze pe o zonă mai restrânsă în găsirea optimului. Odată cu aceste schimbări, am putut obține un regulator cu un fitness mai slab, dar cu un suprareglaj mult mai mic. Valorile rezultate de către algoritm sunt:

$$f_{val} = 0.3756; F = 80.38\%; \quad (41)$$

Matricile componentelor regulatorului PID au devenit:

$$K_p = \begin{bmatrix} 2.75 & 0 \\ 0.947 & 2.464 \end{bmatrix} \quad (42)$$

$$T_i = \begin{bmatrix} 100 & 100 \\ 99.98 & 75.98 \end{bmatrix} \quad (43)$$

$$T_d = \begin{bmatrix} 0.0002 & 0.29 \\ 0 & 0 \end{bmatrix} \quad (44)$$

Din valorile matricilor, putem observa configurația pe care a fost găsită soluția, aceasta fiind C7. În acest exemplu, componenta derivativă nu este 0, putând astfel să concluzionăm că am găsit un regulator optimal PID.

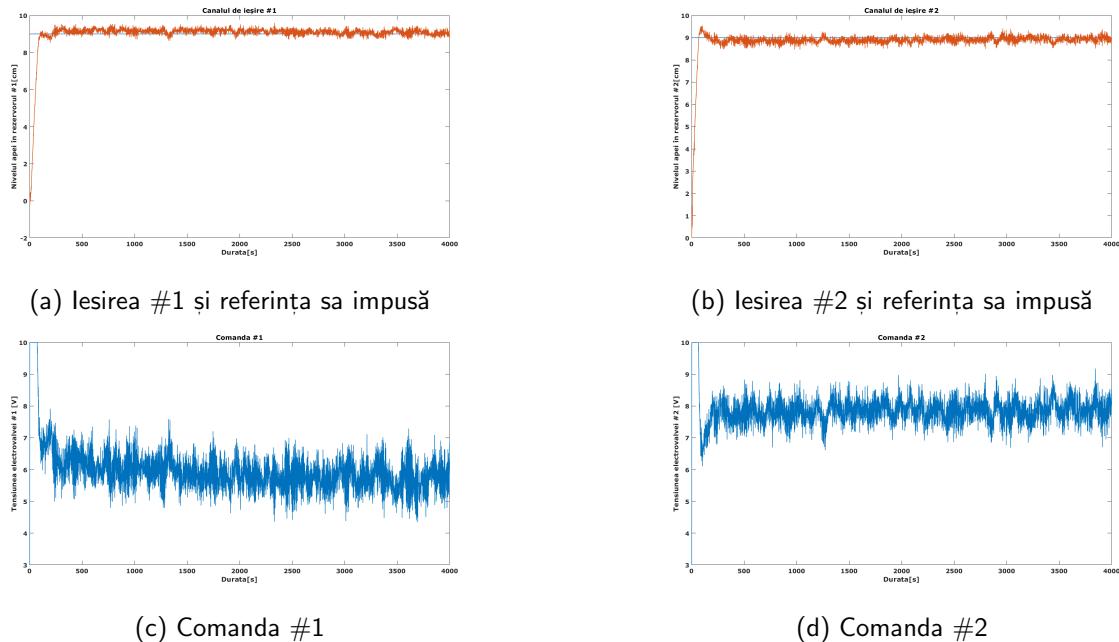


Figure 13: Comenzi și ieșiri în funcție de referințele impuse.

Timpul de simulare a fost mărit la 4000 [s] pentru a putea observa stabilizarea ieșirilor în jurul referințelor impuse. După cum ne-am așteptat, suprareglajul pe canalul de ieșire 2 s-a micșorat, prețul plătit fiind o comandă mult mai zgomotoasă (perturbată), dar și mărirea timpului de stabilizare.

Așadar, avem 2 optime găsite până în acest moment: un PI cu comandă netedă, dar suprareglaj pe canalele de ieșire, și un PID cu comandă zgomotoasă, dar cu suprareglaj micșorat pe

canalele de ieșire. Scopul este de a găsi acel PID cu suprareglaj scăzut, dar și cu comandă netedă.

Pentru a netezi comenziile transmise către proces, vom aplica un filtru median pe fiecare dintre cele două canale. Acest filtru înlocuiește fiecare eșantion cu valoarea mediană calculată din toate eșantioanele începând de la primul până la ultimul. După aceea, procedeul se repetă pentru fiecare eșantion, de la al doilea până la final, și aşa mai departe, până când toate eșantioanele sunt procesate.

Proprietățile filtrului sunt de a păstra caracteristicile semnalului și de a îndepărta zgomotul supus comenzi. Putem defini un filtru median prin schema SIMULINK: Ecuția filtrului median

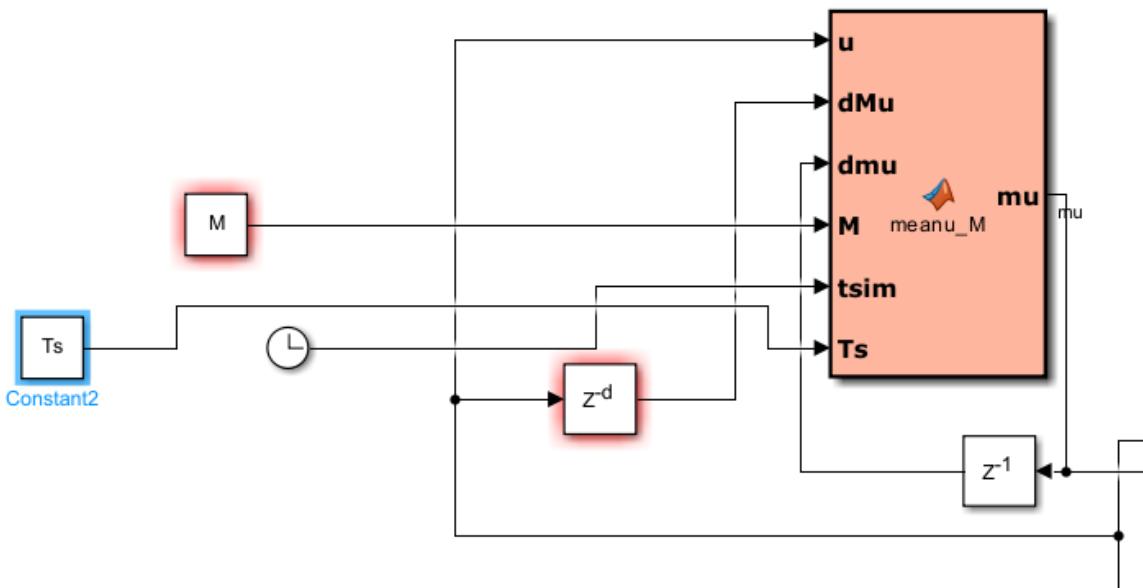


Figure 14: Filtru Median de ordin M

este:

$$u_{filtrat}[n] = \begin{cases} \frac{u[1]+\dots+u[n]}{n}, & \text{pentru } n \leq M \\ \frac{u[n-M+1]+\dots+u[n]}{M}, & \text{pentru } n > M \end{cases} \quad (45)$$

unde $u_{filtrat}[n]$ reprezintă variabila mu a filtrului. Filtrul primește comanda nesaturată, care intră printr-un bloc de întârziere. Variabila M reprezintă ordinul filtrului. Aceasta ne oferă posibilitatea să stabilim pentru ce valoare comenziile procesului devin mai netede, fără a introduce un suprareglaj suplimentar ieșirii. Explicații:

Cu schimbarile efectuate, schema simulink va arăta astfel:

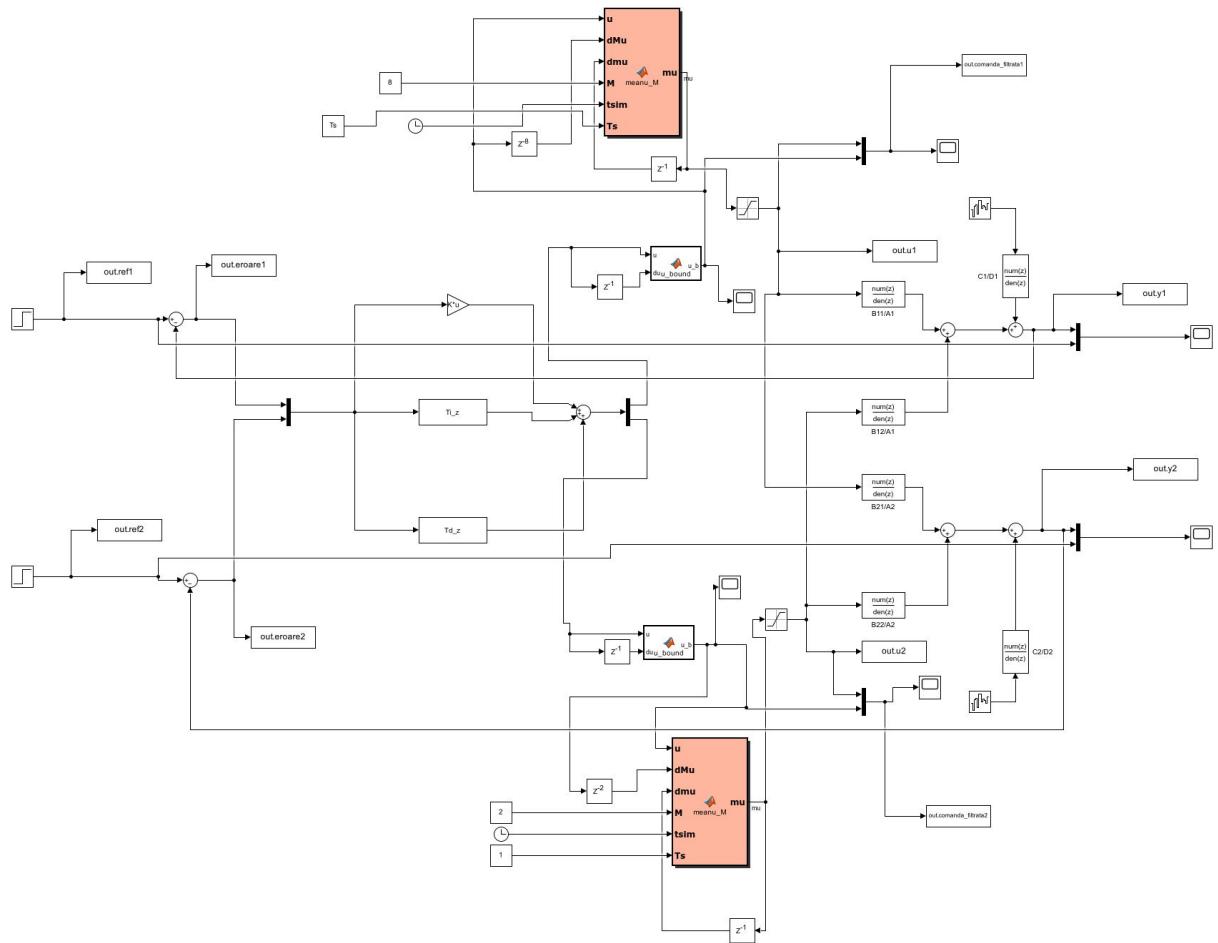


Figure 15: Schema Simulink cu filtru median de ordin M

Vom utiliza PID-ul optim găsit anterior și îi vom aplica filtrul median pe cele 2 comenzi. Pentru a ne ușura munca, am optat în căutarea exhaustivă a ordinelor celor două filtre mediane, analizând funcția fitness la fiecare iterare. Așadar, am obținut un filtru superior de ordin **12** și unul inferior de ordin **7**.

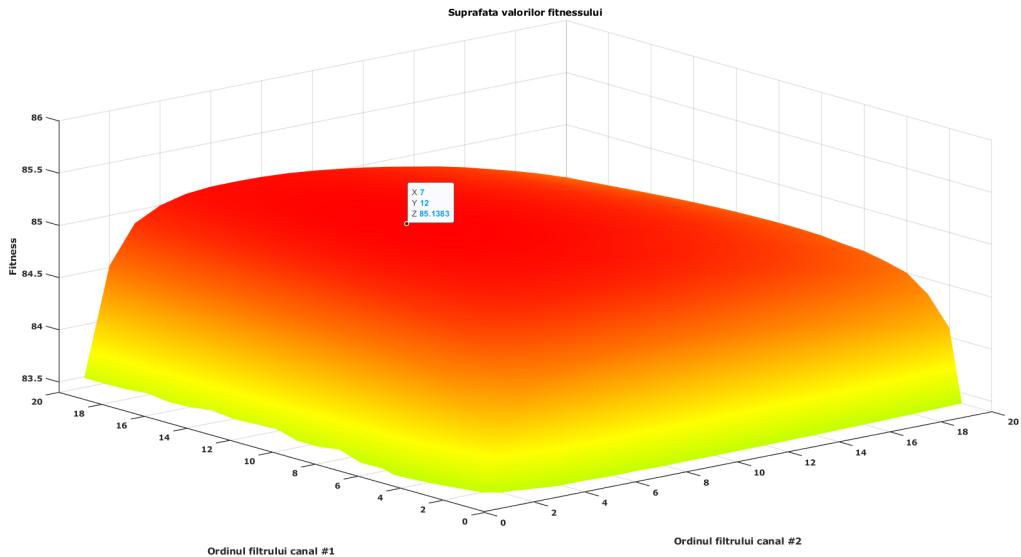


Figure 16: Suprafața valorilor fitnessului

Am generat o mapare a soluțiilor posibile pentru schema SIMULINK cu filtrele mediane, limitând memoria filtrului la 20, deoarece, după acest prag, nu se mai observă diferențe considerabile de fitness.

Rezultatele în urma setării ordinelor celor două filtre mediane sunt:

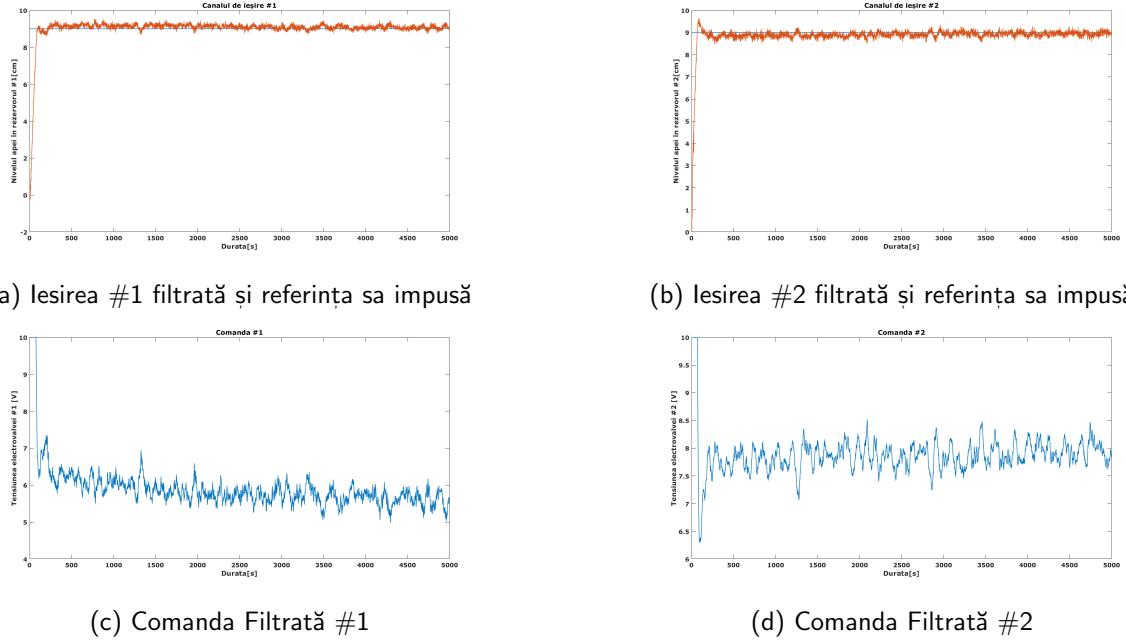
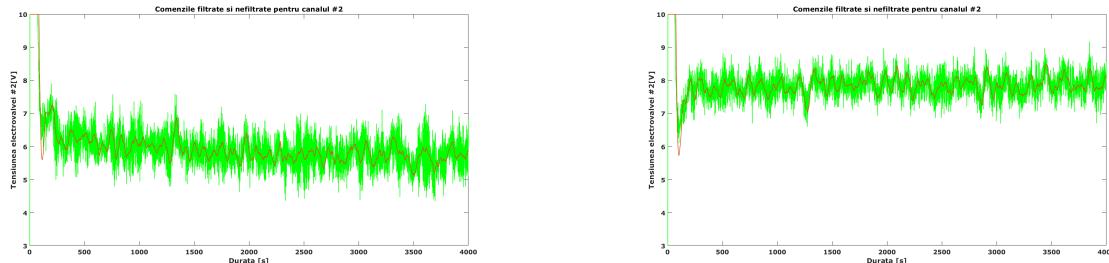


Figure 17: Comenzi și ieșiri filtrate .

Putem observa că, înainte de filtrare, comenziile au prezentat oscilații *Figura13(c) si(d)*, iar după filtrare, comenziile au devenit mai netede (c) și (d). După cum se poate observa și în figura 16, în urma netezirii comenziilor prin aplicarea filtrelor mediane, fitnessul a crescut cu aproximativ 5%, ajungând la valoarea nouă de **85.1383%**. Prețul pe care l-am plătit este mărirea suprareglajului pe canalul 2 de ieșire (d), iar timpul de stabilizare a crescut.

O comparație între cele 2 comenzi poate fi efectuată prin ilustrația:



(a) Comanda Filtrată și Nefiltrată #1

(b) Comanda Filtrată și Nefiltrată #2

Figure 18: Comparatie comenzi primite de la regulatorul PID

În urma filtrării, timpul de stabilizare s-a mărit, simularea fiind analizată pe o durată de 5000 [s]. Concluzionând efectele filtrării, putem afirma că, deși avem prezență unui suprareglaj puțin mărit și a unui timp de stabilizare mai mare, comenziile procesului s-au netezit și nu mai prezintă zgomot.

Tabelul 1: Clasamentul regulatoarelor pe baza valorilor de Fitness și f_val

	PI	PID filtrat	PID nefiltrat
Fitness(%)	90.15	85.13	80.38
J	0.1681	0.2692	0.3756

Clasamentul celor 3 regulatoare de mai sus ilustrează clar ipotezele la care am ajuns: un regulator PI are o valoare cea mai mare a fitnessului și, deși comenziile nu sunt zgomotoase, prezintă un suprareglaj mărit pe canalul de ieșire. PID-ul nefiltrat ne oferă un fitness satisfăcător, cu un suprareglaj nesemnificativ, dar cu zgomot mare pe comenzi. Regulatorul PID filtrat este satisfăcător din următoarele puncte de vedere: are un fitness ridicat, nu prezintă suprareglaj la fel de mare ca al PI-ului, iar sistemul primește comenzi netede de la acesta. În concluzie, putem afirma că acesta din urmă reprezintă PID-ul optim dorit.

6.3 Perturbații și referințe de tipul treaptă succesivă

Pentru a putea testa regulatoarele găsite în secțiunile anterioare, vom introduce perturbații pe canalele de ieșire, dar și referințe de tip treaptă succesivă.

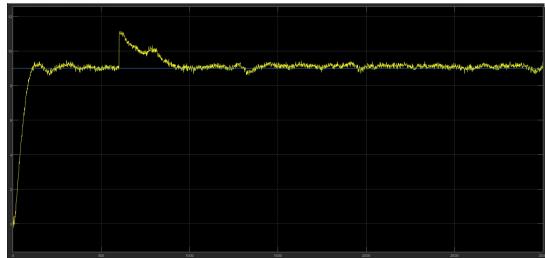
Vom avea 3 scenarii posibile pentru cele 3 regulatoare găsite prin optimizare metaeuristică.

Pentru că primelor 2 regulatoare nu li s-a adăugat filtrarea mediană pe comenzi, vom simula comportamentul lor în cazul perturbațiilor pe canalele de ieșire, conform cu Figura 10.

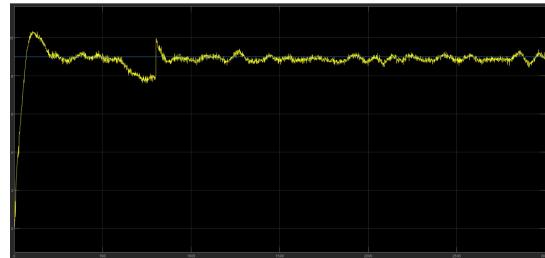
Pe canalele de ieșire a fost adăugat un semnal generat din workspace-ul Matlab, numit *pert1* și *pert2*. Valoarea acestora poate fi schimbată cu ușurință. Pentru testare, am optat pentru un semnal de tip treaptă cu o amplitudine de 20% din valoarea referinței și cu întârziere de 600 s pe canalul 1 și 800 s pe canalul 2.

Timpul de simulare a fost setat la 3000s pentru a putea observa cum se manifestă canalele de ieșire și comenzi cand sunt perturbate. În urma similarilor, avem rezultatele:

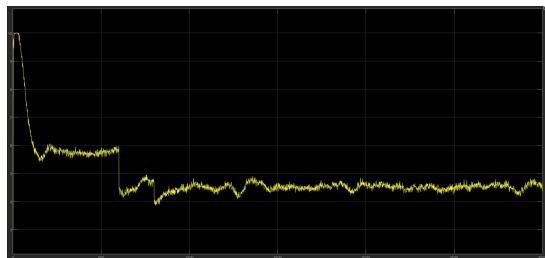
Pentru PI:



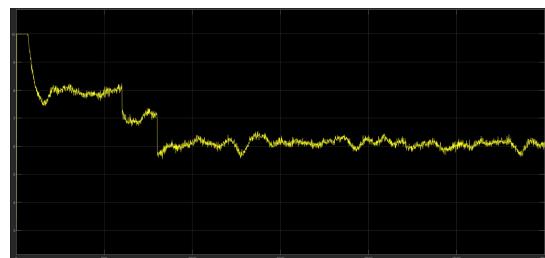
(a) Ieșirea #1 perturbată și referința sa impusă



(b) Ieșirea #2 perturbată și referința sa impusă



(c) Comanda perturbată #1

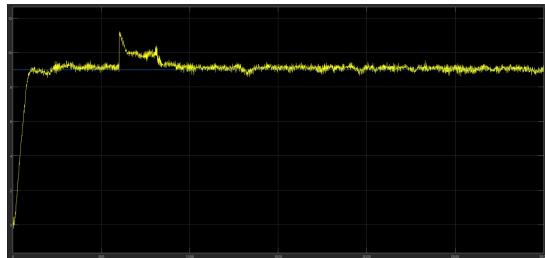


(d) Comanda perturbată #2

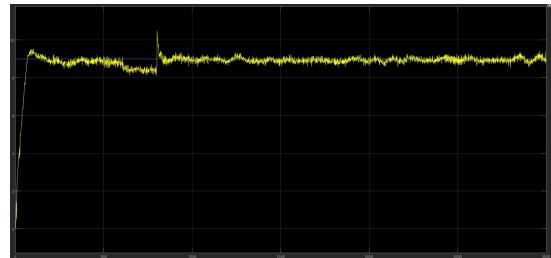
Figure 19: Comenzi și ieșiri perturbate PI

Putem observa cum regulatorul a respins perturbațiile apărute, revenind înapoi la referința dată. În cazul PI-ului, comenzi sunt netede, fără oscilații, dar avem un suprareglaj mare pe canalul 2 ceea ce, în cazul perturbației suplimentare, se stabilizează într-un timp mult mai îndelungat.

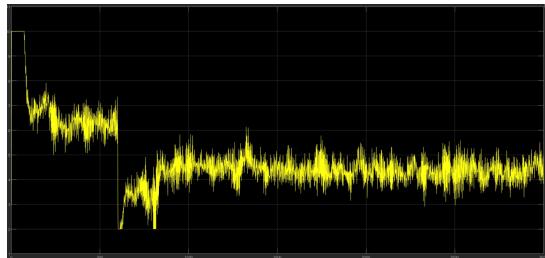
Pentru PID nefiltrat:



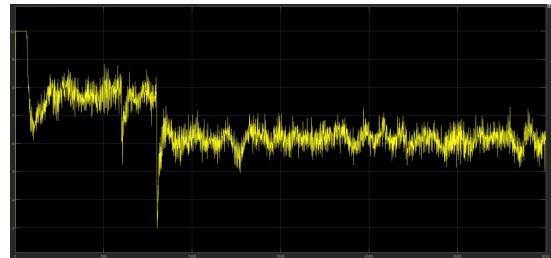
(a) Ieșirea #1 perturbată și referința sa impusă



(b) Ieșirea #2 perturbată și referința sa impusă



(c) Comanda perturbată #1

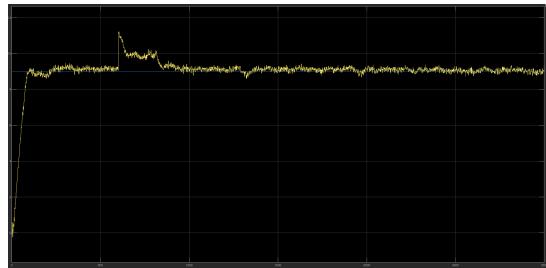


(d) Comanda perturbată #2

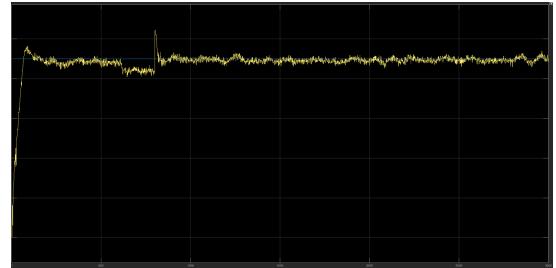
Figure 20: Comenzi și ieșiri perturbate PID nefiltrat

Și de această dată, regulatorul ne respinge perturbațiile suplimentare. Se poate observa în graficele (a) și (b) cum ieșirile se stabilizează rapid, fără a oscila în jurul referinței impuse. Pe de altă parte, graficele comenziilor (c) și (d) prezintă oscilații puternice cauzate de zgomot.

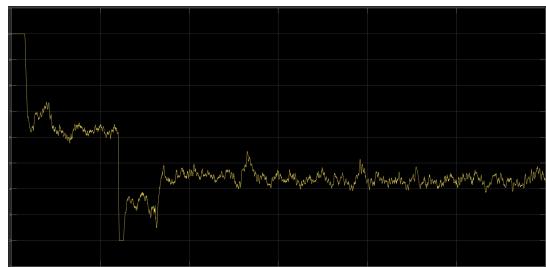
Pentru PID filtrat se va folosi schema de implementare prezentată în Figura 15, la care se adaugă în buclă perturbația pe fiecare canal de ieșire.



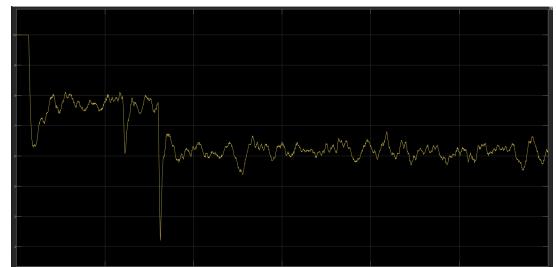
(a) Ieșirea #1 perturbată și referința sa impusă



(b) Ieșirea #2 perturbată și referința sa impusă



(c) Comanda perturbată #1



(d) Comanda perturbată #2

Figure 21: Comenzi și ieșiri perturbate PID filtrat

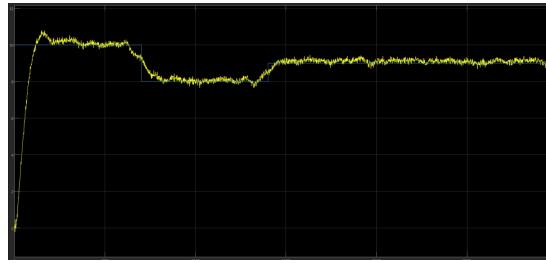
În comparație cu celelalte 2 regulatoare prezentate, pe canalele de ieșire (a) și (b) putem observa cum sunt respinse perturbațiile. Pe canalul 2 (b) suprareglajul este micșorat, ieșirea reușind să se stabilizeze, fără oscilații în jurul referinței. Comenzile (c) și (d) nu prezintă oscilații, sunt netede și sunt limitate în banda de 2-10 V. Putem afirma, în concordanță cu ceea ce a fost concluzionat în secțiunile anterioare, că PID-ul filtrat reprezintă cel mai bun regulator din cele 3 analizate în cazul suplimentării instalației cu perturbații pe canalele de ieșire.

Pentru a testa regulatoarele în cazul referințelor de tip treaptă succesivă, vom folosi 2 semnale extrase din workspace, *referinta1* și *referinta2* ce reprezinta semnale de tip treaptă aplicate în diferite momente.

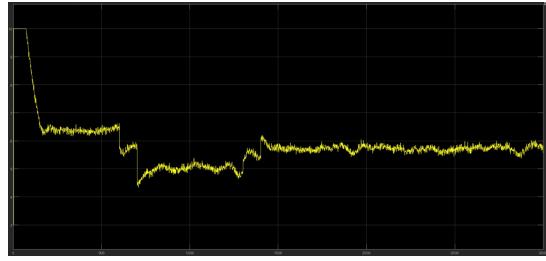
Vom înlocui referințele celor 2 scheme de simulare cu aceste valori.

Timpul de simulare este de 3000 s. În urma simulărilor, avem rezultatele:

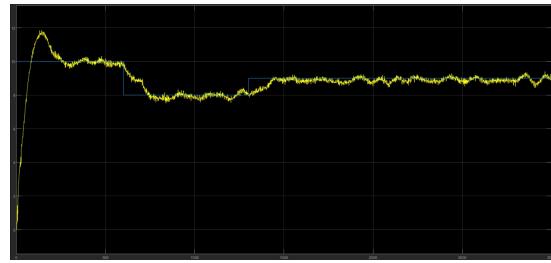
Pentru PI:



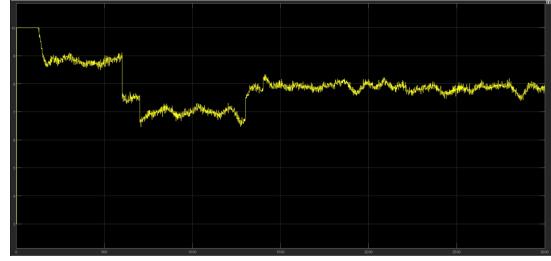
(a) Ieșirea #1 perturbată și referința sa impusă



(c) Comanda perturbată #1



(b) Ieșirea #2 perturbată și referința sa impusă

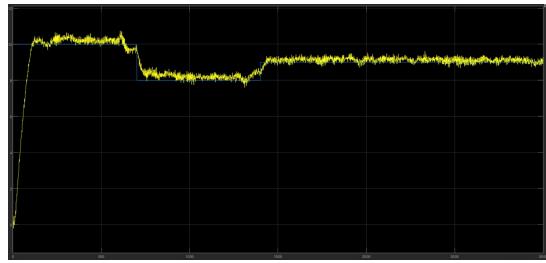


(d) Comanda perturbată #2

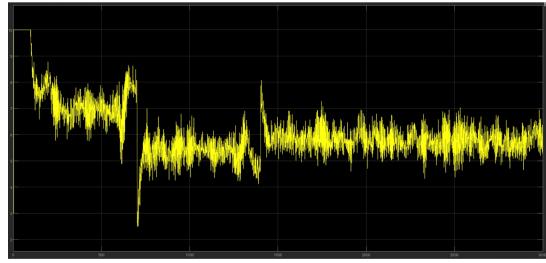
Figure 22: Comenzi și ieșiri la referința de tip treaptă succesivă PI

Regulatorul PI încearcă să urmărească referințele multiple, dar fără prea mult succes. Acest lucru se poate observa mai bine pe canalul 2, figura (b). Suprareglajul este mare, dar comenzi sunt netede, fără prea mult zgomot. Ieșirea de pe canalul 2 arată cum regulatorul încearcă să se stabilizeze la referințele succese, dar nu reușește în totalitate, existând oscilații în jurul acestora.

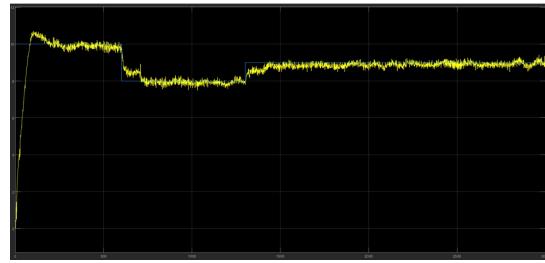
Pentru PID nefiltrat:



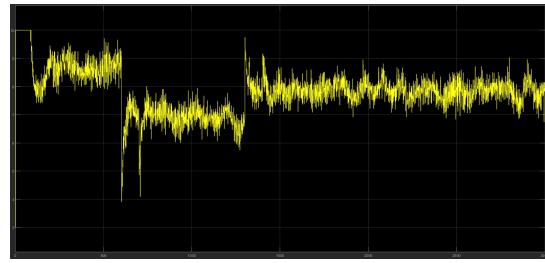
(a) Iesirea #1 perturbată și referința sa impusă



(c) Comanda perturbată #1



(b) Iesirea #2 perturbată și referința sa impusă

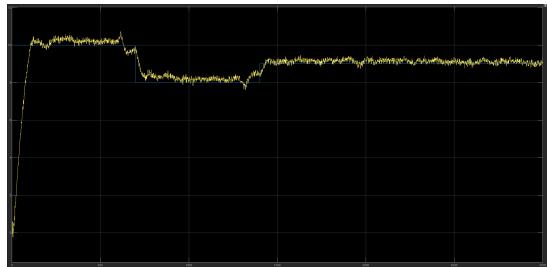


(d) Comanda perturbată #2

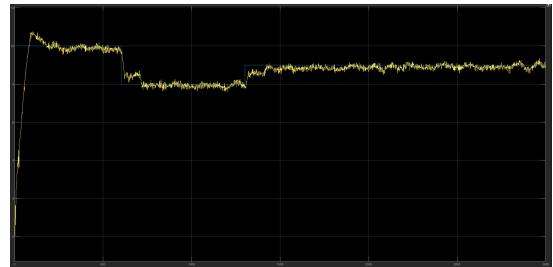
Figure 23: Comenzi și ieșiri la referință de tip treaptă succesivă PID nefiltrat

Regulatorul PID urmărește mai bine referințele succeseive, fără a avea un suprareglaj mărit. Acest lucru se poate observa în figurile (a) și (b). După cum putem observa, comenzi sunt mult mai zgomoioase, mai precis în figurile (c) și (d).

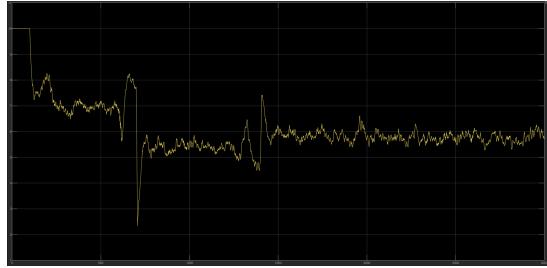
Pentru regulatorul PID filtrat, folosim schema de simulare din Figura 15, în care înlocuim referințele treaptă cu *referinta1* și *referinta2* din Figura 24. În urma simulărilor, obținem:



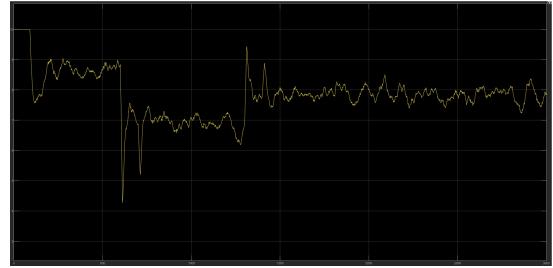
(a) Ieșirea #1



(b) Ieșirea #2



(c) Comanda #1



(d) Comanda #2

Figure 24: Comenzi și ieșiri la referința de tip treaptă succesivă PID filtrat

PID-ul filtrat ne oferă o urmărire satisfăcătoare a referinței de tip treaptă successive, dar și comenzi netezite ce nu prezintă zgomot. Suprareglajul de pe canalul 2 este micșorat (*b*) și oferă o urmărire fără oscilații a referințelor successive. Așadar, acest regulator are răspunsul cel mai bun din cele 3 analizate în cazul referinței de tip treaptă succesivă.

7 CONCLUZII

Obiectivul principal al lucrării este de a realiza un regulator PID multi-variabil pe cale metaeuristică pentru instalația ASTANK2. Parametrii optimi au fost găsiți prin metoda metauristică **particleSwarm** [3], sau pe scurt **PSO**, care oferă o exploatare riguroasă a spațiului de căutare într-un timp scurt și cu o precizie satisfăcătoare. Pentru a putea analiza performanțele valorilor componentelor reguloatoarelor găsite de către particule, trebuie să formăm o funcție fitness care să ne ajute să găsim valorile optime. Maniera construirii funcției de cost reprezintă nucleul căutării metaeuristice.

În urma simulării, s-au găsit mai multe tipuri de regulatoare cu fitnessuri diferite. Cel mai performant s-a dovedit a fi PI-ul, în care componenta derivativă este nulă. Căutările s-au reluat cu alte intervale pentru componenta derivativă astfel încât algoritmul metauristic să ne returneze un PID. Fitnessul PID-ului astfel găsit este mai mic decât cel al PI-ului aflat anterior. Din graficele comenziilor PID-ului am concluzionat faptul că putem introduce un filtru median pe comenzi astfel încât acestea să se netezească. Prin aplicarea filtrelor mediane, am obținut astfel un PID optimal, cu fitness mai mare decât PID-ul fără filtru, dar cu fitness mai mic decât PI-ul.

În urma studiului rejectării perturbațiilor suplimentare și a urmăririi referinței de tip treaptă succesivă, putem afirma că regulatorul PID cu filtru median pe comenzi oferă, în ambele ipostaze, cel mai bun răspuns. Acesta rejectează cu succes perturbațiile și urmărește cu precizie satisfăcătoare referința de tip treaptă succesivă, în timp ce oferă comenzi netede pentru proces.

8 CONTRIBUȚII ALE STUDENTULUI

- Utilizarea algoritmului PSO implementat în Matlab
- Definirea funcției cost pentru algoritm metaheuristic
- Implementarea schemei de simulare în SIMULINK
- Implementarea funcției de analizare a erorilor și a comenziilor sistemului
- Analiza rezultatelor obținute în urma simulărilor schemei de simulare
- Adaptarea schemei de simulare aplicând filtrul median
- Implementarea expresiilor perturbațiilor și referințelor de tip treaptă succesivă

BIBLIOGRAPHY

- [1] Janetta Culită, Dan Ștefănoiu, and Alexandru Dumitrașcu. Astank2: Analytical modeling and simulation. In *2015 20th International Conference on Control Systems and Computer Science*. IEEE, 2015. CuStDu-15.
- [2] Ioan Dumitrache. *Ingineria reglării automate*. Editura Politehnica Press, 2010.
- [3] Dan Ștefănoiu, Pierre Borne, Dumitru Popescu, Florin Gheorghe Filip, and Abdelkader El Kamel. *Optimization in Engineering Sciences, Metaheuristics, Stochastic Methods and Decision Support*. Wiley, 2014.
- [4] Dan Ștefănoiu and Janetta Culită. Optimal identification and metaheuristic pid control of a two-tank system. *Electronics*, 2021.
- [5] Dan Ștefănoiu, Janetta Culită, and Petre Stoica. Fundamentele modelarii si identificarii sistemelor. *Editura Printech, Bucuresti*, 2005.