Nastavljanje parametrov regulatorja z optimizacijskimi algoritmi

Dušan Fister
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
dusan.fister@student.um.si

Riko Šafarič
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, Maribor
riko.safaric@um.si

Iztok Jr. Fister
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, Maribor
iztok.fister1@um.si

Iztok Fister
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, Maribor
iztok.fister@um.si

POVZETEK

Vsak regulacijski sistem za optimalno delovanje zahteva pravilno nastavitev vhodnih parametrov. Uporabnik tega sistema strmi k temu, da bo njegov sistem deloval čim bolj optimalno. Za industrijske robote to pomeni, da sistem zagotavlja varnost in funkcionalnost. Velikokrat uvrščamo funkcionalnost pred varnost, kar povzroča nevšečnosti v praksi. Sami želimo robotu optimalno nastaviti regulacijske parametre tako, da bo zagotavljal tako varnost kot tudi funkcionalnost. Čeprav je naš robot miniaturna različica industrijskega robota, tudi ta za pravilno delovanje zahteva dovolj kakovostne nastavitve regulacijskih parametrov. Cilj naše študije je za iskanje optimalnih regulacijskih parametrov razviti algoritem po vzoru obnašanja netopirjev in ga uspešno uporabiti v praksi.

Ključne besede

regulacije, mehatronika, optimizacijski algoritmi

1. UVOD

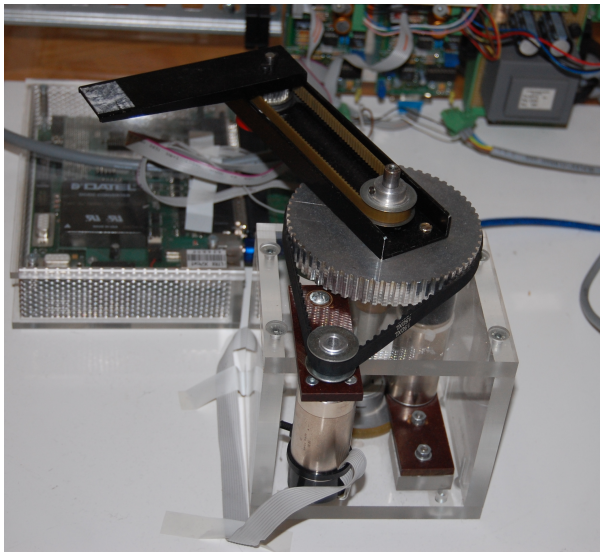
Optimizacijske algoritme uporabljamo za povečanje produktivnosti, storilnosti, kakovosti in hitrosti dela robotskih strojev. Vsak robot mora skozi dobo obratovanja delovati zanesljivo, varno ter funkcionalno. Pričakovati je, da bo robot zaradi povečanja produktivnosti deloval karseda hitro, vendar zaradi tega ne bo ogrožal varnosti ljudi v njegovi bližini. Pomembni faktor za delovanje robotskih mehanizmov predstavlja regulator, tj. element, ki zmanjšuje napako, oz. razliko med želeno in dejansko vrednostjo. Poznamo več vrst regulatorjev, od najpreprostejšega P-regulatorja, ki omogoča le ojačanje napake, do nekoliko kompleksnejših PI- in PD-regulatorjev, do najkompleksnejšega PID-regulatorja. V kraticah imen teh regulatorjev pomeni I-člen integriranje in D-člen diferenciranje napake. Da zagotovimo pravilno nastavljen regulator, moramo pravilno nastaviti vsak posamezen člen posebej. Robot optimalno deluje šele, ko so glede na njegove zahteve dela optimalno nastavljeni vsi členi regulatorja. Enačba (1) matematično prikazuje regulacijske člene in njegove nastavljive parametre, tj.

$$u = K_P(y_{zel} - y_{dej}) + K_I \int (y_{zel} - y_{dej}) + K_D(y_{zel} - y_{dej})', \quad (1)$$

kjer y_{zel} predstavlja želeno referenčno vrednost, medtem ko y_{dej} dejansko vrednost prenihaja. Naš uporabljen regulator, opisan z enačbo (1), sestoji iz izključno PI-člena, kar pomeni, da D-člen v našem primeru odpade.

Najpreprostejši način nastavljanja parametrov je ročno nastavljanje, ki pa zahteva izkušenega tehnika in obilo potrpežljivosti. Obstajajo različne avtomatske metode nastavljanja, kjer največkrat v te namene uporabljamo optimizacijske algoritme po vzoru iz narave. Korenine teh algoritmov segajo v leto 1871, ko je Charles Darwin objavil znanstveno delo o naravni selekciji [1]. Ta je navdihnila Alana Turinga pri razvoju t.i. *genetskega iskanja*, ki uporablja Darwinovo evolucijsko teorijo pri reševanju optimizacijskih problemov [14]. Leta 1988 je John Holland algoritem s selekcijo in mutacijo poimenoval *genetski algoritem*, kar se je ohranilo vse do danes [7]. Podrobnejši zgodovinsko pregled je naveden v [4].

Z robotom SCARA (angl. Selective Compliance Assembly Robot Arm) [12], prikazanim na sliki 1, se je prvi začel ukvarjati Albin Jagarinec. Leta 2005 je uspešno razvil adaptivni regulator [8], medtem ko je leto kasneje Marko Kolar preizkusil algoritem z mehko logiko [11]. Jure Čas se je ukvarjal z zveznim nevronskim sliding-mode regulatorjem [15], medtem ko je Tomaž Slanič iskal optimalne nastavitve parametrov omenjenega robota z genetskim algoritmom [13]. V tem prispevku opisujemo avtomatsko nasta-



Slika 1: Dvoosni robot.

vljanje parametrov PI-položajnega regulatorja na dvoosnem robotu SCARA z algoritmom po vzoru obnašanja netopirjev (angl. Bat Algorithm, krajše BA) [18]. Slednji obljublja enostavno implementacijo algoritma za reševanje realnega problema in hkrati ponuja hitro konvergenco rešitev. Gre za novejšega predstavnika algoritmov iz družine inteligence rojev (angl. Swarm Intelligence), ki za svoje delovanje uporabljajo biološko-socialno obnašanje insektov (npr. roji čebel, družine mravelj, ipd.) oz. živali (npr. jate ptic, rib, ipd.) [10].

2. REGULACIJSKA PROGA

Glavnino regulacijske proge predstavlja dvoosni robot SCARA, ki pooseblja gibanje človeške roke okoli rame in komolca, za kar skrbita dva enosmerna motorja. H glavni gredi vsakega motorja je prigrajen tudi inkrementalni merilnik položaja, ki služi kot dajalnik povratnega signala. Za učinkovito gibanje sklopa skrbita gonilnika motorjev, ki združujeta položajni regulator, komparator in smerni diskriminator za identifikacijo signalov iz inkrementalnega dajalnika ter ojačevalnik referenčnega signala. Slednjega določa vmesnik med računalnikom in robotom, katerega imenujemo DSP-2 Roby kartica [16]. Razvita je bila na UM, FERI in je neposredno povezana z računalnikom, od koder pridobiva zahtevane podatke in kamor pošilja povratne informacije. Optimizacija zato v celoti teče na računalniku.

2.1 Model robota

Regulacijska proga robota je nelinearna, saj model robota zapišemo kot člen drugega reda [17]. Pomeni, da običajni metodi nastavljanja regulatorja, npr. Bodejeva metoda [9] in krivulja lege korenov [3] pri načrtovanju nista več uporabni Enačba (2) predstavlja model robota.

$$\begin{bmatrix}\tau_{mot1}\\ \tau_{mot2}\end{bmatrix} = \begin{bmatrix} J_{m1}n_1 + \frac{a_1+2a_2cos(q_2)}{n_1} & \frac{a_3+a_2cos(q_2)}{n_1} \\ \frac{a_3+a_2cos(q_2)}{n_2} & J_{m2}n_2 + \frac{a_3+J_{3o}}{n_2}\end{bmatrix} \cdot \begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix} + 2 + \begin{bmatrix} \frac{-a_2\dot{q_2}(2\dot{q_1}+\dot{q_2})sin(q_2)}{n_1} \\ \frac{a_2sin(q_2)\dot{q_1}^2}{n_2}\end{bmatrix} = \begin{bmatrix} a & b \\ c & d\end{bmatrix} \cdot \begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix} + \begin{bmatrix}e\\ f\end{bmatrix} \quad (2)$$

Osnovni parametri za izračun modela robota so dolžine l, mase m in težiščni vztrajnosti momenti J posameznih osi ter gonil, medtem ko dodatni členi predstavljajo položaj q, hitrost $\dot{q}$ in pospešek osi $\ddot{q}$.

Optimizacijski problem je definiran kot $OP = \{I, S, f, goal\}$, kjer I predstavlja množico vseh nalog $\mathbf{x} \in I$, ki se lahko pojavijo na vhodu, S je množica dopustnih rešitev $\mathbf{x} \in S$ in f kriterijska funkcija. *goal* določa, ali kriterijsko funkcijo minimiziramo ali maksimiziramo. Vhod v optimizacijski problem je podan z dvema dvojicama parametrov:

$$\mathbf{x} = \{Kp_1, Kp_2, Kv_1, Kv_2\}, \quad (3)$$

kjer parametra Kp_1 in Kp_2 označujeta velikost proporcionalnega dela (P-člena) prve oz. druge osi, ter Kv_1 oz. Kv_2 velikost integralnega dela (I-člena) prve oz. druge osi.

Med optimizacijskim postopkom pri znanem modelu in znanem izhodu iščemo optimalne vhodne spremenljivke [2]. Pri tem izhodne spremenljivke razvrstimo v tri različne kategorije, tj.:

- $Over_i$ - dejanski prenihaj posamezne osi,
- Ess_i - statični pogrešek posamezne osi in
- $Time_i$ - nastavitveni čas posamezne osi.

Vse tri izhodne spremenljivke pridobivamo neposredno iz kartice DSP-2 Roby, medtem ko jih na računalniku dodatno obdelamo in uporabljamo za vrednotenje kakovosti vhodnih podatkov. Ti predstavljajo osnovo za vrednotenje kakovosti poljubnega odziva na stopnično funkcijo. Robotu tako podajamo navodila za delo, npr. premakni obe osi za določen kot. Robot poskuša doseči najhitrejši premik ob upoštevanju zahtevanega prenihaja, minimalnega statičnega pogreška in minimalnega nastavitvenega časa. Zahtevan prenihaj vnese uporabnik programa pred začetkom optimizacije.

Vrednotenje, oz. ocenjevanje odziva in posledično kakovosti izhodnih podatkov (posredno vhodnih) izvaja ocenjevalna funkcija (angl. objective function), prikazana v enačbi (4):

$$f_i = E_{1i}\cdot(1-|P_i - Over_i|)+E_{2i}\cdot(1-Time_i)+E_{3i}\cdot(1-Ess_i), \quad (4)$$

ki jo maksimiziramo. Zaradi medsebojno mehanske sklopljenosti dvoosnega robota večkriterijske optimizacije ne moremo vršiti. Zato uporabimo uteženo vsoto kriterijev, ki utežijo posamezen parameter tako, da na koncu znaša maksimalna vrednost (popolni odziv) ocenjevalne funkcije $f_i = 1$. Bližje enici torej smo, kakovostnejše rezultate pridobivamo.

3. ALGORITEM PO VZORU OBNAŠANJA NETOPIRJEV

Algoritem po vzoru obnašanja netopirjev (angl. Bat Algorithm, krajše BA) je novejši predstavnik optimizacijskih algoritmov po vzoru iz narave. Nastal je leta 2010 pod okriljem matematika Xin-She Yanga. Algoritem temelji na sposobnosti navigacije netopirjev v popolni temi. Netopirji predstavljo eno redkih bioloških vrst, ki se v naravi orientirajo s pomočjo t.i *eholokacije*. Pojav označuje periodično oddajanje kratkih ultrazvočnih pulzov, odbijajočih se od plena ali

ovire in merjenjem časa odmeva. Posledično lahko ob znani hitrosti potovanja zvoka po zraku določimo oddaljenost od plena ali ovire.

Obnašanje netopirja lahko zapišemo z matematičnim modelom, ki temelji na naslednjih osnovnih pojmih:

- dejanski frekvenci oddajanja Q_i,
- dejanski hitrosti leta $\mathbf{v}_i^{(t)}$ in
- dejanskemu položaju netopirja $\mathbf{x}_i^{(t)}$,

medtem ko naključni let netopirja zapišemo kot kombinacijo treh enačb:

$$Q_i = Q_{min} + (Q_{min} - Q_{min}) \cdot \beta, \tag{5}$$

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} + \left[\mathbf{x}_i^{(t)} - \mathbf{x}_{best}^{(t)}\right] \cdot Q_i, \tag{6}$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}, \tag{7}$$

kjer Q_{max} in Q_{min} predstavljata zgornjo in spodnjo mejo frekvence inicializiranih ob zagonu, krmilni parameter β je naključno število generirano v intervalu $[0, 1]$ in $\mathbf{x}_{best}^{(t)}$ trenutna najboljša rešitev.

Proces iskanja v algoritmu BA je odvisen od dveh procesov, tj. preiskovanja (angl. exploration) in izkoriščanja (angl. exploitation). Preiskovanje je močnejše zlasti ob začetku optimizacije, ko so položaji posameznikov razmetani naključno po preiskovalnem prostoru. Izkoriščanje pomeni, da začne preiskovalni proces izboljševati najdeno rešitev običajno z metodami lokalnega iskanja. Vsak posameznik se pri preiskovanju prostora giba v smeri trenutne najboljše rešitve, kar v naravi pomeni, da netopir oddaja ultrazvočne pulze visokih glasnosti A_i, ki posledično potujejo dlje. Te pulze oddaja le nekajkrat v sekundi, npr. osem do desetkrat, kar definiramo s parametrov emisija pulzov r_i. Ko netopir opazi plen, se mu začne približevati, glasnost začne upadati, medtem ko emisija pulzov raste. V tej fazi začne preiskovalni algoritem preiskani prostor rešitev izkoriščati. Matematično obstajata torej dva računska postopka za opis obeh komponent preiskovalnega proces, tj. preiskovanje je zapisano v enačbah (5)-(7), medtem ko izkoriščanje sledi zapisu v enačbi (8):

$$\mathbf{x}_{novi} = \mathbf{x}_{stari} + \epsilon \cdot \bar{A}^{(t)}. \tag{8}$$

Glasnost A_i in emisija pulzov r_i se sicer v originalnem algoritmu spreminjata, vendar se je za potrebe naše optimizacije izkazalo najbolje, da oba parametra nastavimo fiksno in s tem uravnavamo mejo med procesoma preiskovanja in izkoriščanja. S tem dosežemo nekoliko nižjo konvergenco v začetnih generacijah. Oba parametra sta v originalnem algoritmu odvisna od funkcijskih predpisov, ki pa začetno vrednost parametra v nekaj generacijah fiksirata na konstantne vrednosti, zato se parametra v poznejših generacijah obnašata podobno, kot če bi ju fiksirali. Vse omenjene enačbe lahko strnemo v optimizacijski algoritem, katerega psevdokod je prikazan v algoritmu 1. Algoritem začnemo z inicializacijo naključnih posameznikov. Populacijo ovrednotimo ter poiščemo posameznika z najkakovostnejšo rešitvijo.

Algoritem 1 Algoritem na osnovi obnašanja netopirjev

Vpis: Populacija netopirjev $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ za $i = 1 \ldots Np$, MAX_FE.
Izpis: Najboljša rešitev $\mathbf{x}_{best}$ in njena vrednost $f_{max} = \max(f(\mathbf{x}))$.

```
1: init_bat();
2: eval = vrednoti_novo_populacijo;
3: f_max = išči_najboljšo_rešitev(x_best);
4: while termination_condition_not_met do
5:   for i = 1 to Np do
6:     y = generiraj_novo_rešitev(x_i);
7:     if rand(0,1) < r_i then
8:       y = izboljšaj_najboljšo_rešitev(x_best)
9:     end if{ lokalno iskanje }
10:    f_new = vrednoti_novo_rešitev(y);
11:    eval = eval + 1;
12:    if f_new ≤ f_i and N(0,1) < A_i then
13:      x_i = y; f_i = f_new;
14:    end if{ shrani najboljšo rešitev pogojno }
15:    f_max=išči_najboljšo_rešitev(x_best);
16:  end for
17: end while
```

Slednji ima najvišjo vrednost ocenjevalne funkcije in predstavlja trenutno najboljšo globalno rešitev. Jedro algoritma predstavlja globalna zanka, ki se izvaja dokler ne presežemo maksimalnega števila iteracij, oz. dovolj kakovostne predpisane rešitve. V tej zanki iz vsakega posameznika generiramo novo rešitev s pomočjo preiskovanja. Glede na verjetnost, da emisija pulzov pade pod določeno mejo če je ta dovolj blizu rešitve pa nadaljujemo tudi na proces izkoriščanja, imenovanega tudi lokalno iskanje. Novo pridobljeno rešitev shranimo le, če je glasnost oddajanja ultrazvočnih pulzov dovolj majhna, kar pomeni, da se netopir nahaja blizu plena. V nasprotnem primeru rešitev zavržemo.

Algoritem išče optimalne parametre za nastavitev regulatorja v omejenem definicijskem območju, ki preprečuje nestabilno delovanje robota. Po vsakem izračunanem premiku novo nastale parametre vpiše v regulator ter počaka na izvršitev le-teh. Kriterijska funkcija vsako dvojico parametrov ovrednoti ter rezultat posreduje algoritmu. Vsako ovrednotenje zaradi mehanskih omejitev traja natančno deset sekund, zato za nas hitrost izvajanja algoritma ni pomembna. Vsekakor pa s povečevanjem števila posameznikov ter generacij premo sorazmerno povečujemo tudi čas potreben za optimizacijo.

4. REZULTATI

V tem poglavju predstavljamo rezultate pridobljene na realni laboratorijski aplikaciji. Zaradi stohastičnosti algoritma je posnetih več odzivov na stopnično funkcijo, čeprav se v praksi uporablja le osnovno testiranje, ko deluje robot brez prenihaja. Rezultati so bili posneti s pomočjo orodja MATLAB/Simulink ter vmesniške kartice DSP-2 Roby. Prikazani rezultati veljajo za dvoosnega robota ($n = 2$). Krmilni parametri algoritma BA so bili nastavljeni med eksperimenti na naslednje vrednosti:

- število posameznikov $N_p = 10$,
- število iteracij $n_{gen} = 10$,
- emisijo pulzov $r_i = 0.1$,
- glasnost oddajanja pulzov $A_i = 0.9$ in

- faktor skaliranja (frekvenca) $Q_i = \{0.5, 1.5\}$.

V sklopu eksperimentalnega dela smo ozvedli tri teste, ki se med seboj razlikujejo po različnih vrednostih zahtevanih prenihajev:

- prvo testiranje: $P_1 = 0$ %, $P_2 = 0$ %,
- drugo testiranje: $P_1 = 10$ %, $P_2 = 10$ % in
- tretje testiranje: $P_1 = 15$ %, $P_2 = 25$ %.

Vsa testiranja so prikazana tabelarično in grafično.

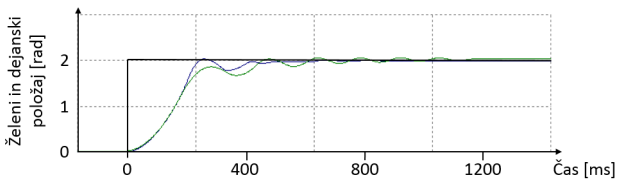
4.1 Prvo testiranje

Prvo in najosnovnejše testiranje je prikazano v tabeli 1 in sliki 2. Vsaka tabela in slika na kratko opisujeta dogajanje po končani optimizaciji, vhodni podatki v realni laboratorijski sistem sta optimizirani dvojici. Velja omeniti tudi zanimivost aplikacije, katero predstavlja povratni gib robota, izveden s preprostim P-regulatorjem za gib nazaj na ničelno točko.

Tabela 1: Prvo testiranje

Izmerjeni rezultati	1. os	2. os
Vrednost ocenjevalne funkcije f_i	0.96516	0.98324
Vrednost prenihaja $Over_i$	0.0187	0
Vrednost statičnega pogreška Ess_i	0.00058	0.00544
Vrednost nastavitvenega časa $Time_i$	0.906	0.504

Testiranje je bilo dokaj uspešno izvedeno, kar dokazuje tudi visoka vrednost povprečne ocenjevalne funkcije. Prenihaj prve osi je bil nastavljen nekoliko nenatančno, medtem ko je prenihaj druge osi natančno zadel zahtevano nastavitev. Statični progrešek je pomemben del robotove natančnosti. Višjo natančnost je dosegel s prvo osjo, saj ta beleži nižjo stopnjo napake. Nastavitveni čas je parameter, ki pove kako robot zaniha blizu rešitve. Daljši kot je nastavitveni čas, slabša je kakovost parametrov. Določa ga ozko tolerančno območje, tj. hitreje kot se robot ustali v tem območju, krajši nastavitveni čas ga odlikuje.



Slika 2: Prvo testiranje

Iz slike 2 je razvidno, da se je robot uspešno premaknil za kot dva radiana, vendar v okolici nekoliko zanihal in povzročil dolg nastavitveni čas, kar se sklada s tabelo 1. Z dodatnim eksperimentalnim nastavljanjem krmilnih parametrov bi lahko nastavitveni čas drastično izboljšali, vendar oteževali preprosto uporabo naše aplikacije. Poleg tega bi bilo potrebno nekajkrat ponastaviti spekter možnih števil iz katerih pridobivamo posamezne člene dvojic parametrov in jih čimbolj približati k optimalnim nastavitvam, kar dodatno zaostruje uporabo aplikacije. Predpostavili smo, da mora biti algoritem in njegova uporaba enostavna in funkcionalna, kar pa nekoliko slabša kakovost posameznih rezultatov. V oči bode tudi podnihaj, ki se pojavi takoj po dosegu zahtevane vrednosti in dodatno dodeljuje manevrski prostor za finejšo obdelavo.

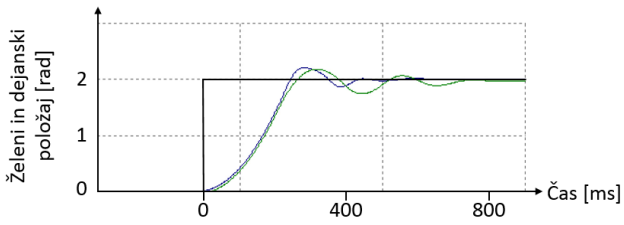
4.2 Drugo testiranje

Drugo testiranje predstavljata tabela 2 in slika 3. Testiranje je bilo izvedeno za zahtevana prenihaja $P_1 = 10$ % in $P_2 = 10$ %. Nastavljanje parametrov se za ta režim v praksi ne uporablja več, mi smo ga uporabili kot dodaten optimizacijski problem za podkrepitev rezultatov.

Tabela 2: Drugo testiranje

Izmerjeni rezultati	1. os	2. os
Vrednost ocenjevalne funkcije f_i	0.9739	0.9854
Vrednost prenihaja $Over_i$	0.10733	0.10237
Vrednost statičnega pogreška Ess_i	0.00579	0.00113
Vrednost nastavitvenega časa $Time_i$	0.714	0.444

Tabela 2 kaže končne rezultate drugega testiranja. Razvidno je, da je v tem primeru nekoliko kakovostneje nastavljena druga os. O tem priča višja vrednost ocenjevalne funkcije. Tudi prenihaja sta v veliki meri natančno načrtana, maksimalna napaka znaša zgolj 0.7 %. Statični pogrešek je nekoliko ohlapneje načrtan v primerjavi s prvim primerom. Ta sicer znaša precej manj za drugo os, medtem ko se je za prvo os enormno povečal. Nastavitveni čas je v obeh primerih krajši, kar dodatno izboljšuje vrednost ocenjevalne funkcije.



Slika 3: Drugo testiranje

Grafično predstavljen odziv dopolnjuje kakovost tabelarično predstavljenih rezultatov. Odziv je iz grafičnega stališča kakovostneje optimiziran kot prvi, največ k temu pripomore krajši nastavitveni čas. Iz fizikalnega stališča je treba omeniti, da so začetna eksperimentalna testiranja krmilnih parametrov potekala prav na tem režimu delovanja - $P_1 = 10$ % in $P_2 = 10$ %. Pomeni, da so ti parametri najugodnejši za opisovano testiranje, medtem ko za druga testiranja ne veljajo več v popolni meri. To pa je prednost in obenem slabost nastavljanja krmilnih parametrov.

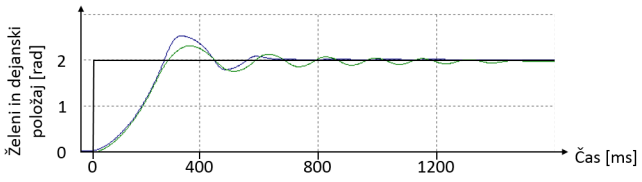
4.3 Tretje testiranje

Tretje testiranje poosebljata tabela 3 in slika 4, zahtevana kombinirana prenihaja znašata $P_1 = 15$ % in $P_2 = 25$ %. Tabelarični rezultati tretjega testiranja prinašajo glede na vrednost ocenjevalne funkcije pozitivne lastnosti, vendar zahtevajo glede na natančnost načrtanega prenihaja in

Tabela 3: Tretje testiranje

Izmerjeni rezultati	1. os	2. os
Vrednost ocenjevalne funkcije f_i	0.95527	0.97839
Vrednost prenihaja $Over_i$	0.16721	0.25872
Vrednost statičnega pogreška Ess_i	0.00202	0.00027
Vrednost nastavitvenega časa $Time_i$	1.222	0.606

trajanja nastavitvenega časa prve osi dodatno optimizacijo. Kakovostno so načrtani vsi izhodni podatki za drugo os. Pojav dolgega nastavitvenega časa in posledično nižje oce-



Slika 4: Tretje testiranje

njevalne funkcije prikazuje tudi slika 4. Čeprav je druga os uspešno nastavljena, zaradi velikega prenihaja in vztrajnostnih mas vpliva na prvo os. Znano je namreč, da sta obe osi medsebojno mehansko sklopljeni, kar posledično zmanjšuje kakovost pridobljenih podatkov druge osi.

5. SKLEP

Z našo aplikacijo smo hoteli pokazati, da so optimizacijski algoritmi po vzoru iz narave, konkretneje algoritem BA, primerni za nastavljanje parametrov regulatorjev. Algoritem BA, ki je znan zaradi svoje enostavne implementacije ter hitre konvergence, poleg zvezne optimizacije matematičnih modelov ponuja oporo tudi takim vrstam problemov. Četudi regulator v vseh treh primerih ni bil optimalno nastavljen, izstopajo namreč rezultati prve osi, smo bili objektivno z dobljenimi rešitvami zadovoljni. Vsekakor bi s povečanjem števila posameznikov v populaciji ter števila iteracij lahko pričakovali kakovostnejše rezultate, vendar bi po drugi strani povečali potreben optimizacijski čas.

Rezultati poskusov so pokazali, da najbolj izstopa dolg nastavitveni čas prve osi, saj ta predstavlja nosilno oporo tudi drugi osi. S hitrim spreminjanjem položaja in vztrajnostnih momentov pa slednja vpliva na prvo, kar predstavlja glavno mehansko težavo. Težavo bi lahko rešili z manjšanjem regulacijskega I-člena, oz. ponovno optimizacijo z ožjim naborom razpoložljivih vrednosti tega člena (Kv_1).

V prihodnje želimo algoritem BA hibridizirati s strategijami diferencialne evolucije in tako izboljšati rezultate optimizacije [6]. Trenutno so v teku tudi testiranja genetskega algoritma (GA), optimizacije z roji delcev (PSO) in diferencialne evolucije (DE). Dodatno izboljšavo predstavlja adaptacija krmilnih parametrov, kjer postopek ročnega nastavljanja parametrov avtomatiziramo ter s tem eliminiramo predčasno konvergenco v lokalne optimume [5]. Prav ta lastnost predstavlja glavno težavo omenjenega algoritma, saj optimizacija teče dokler izboljšujemo trenutno najboljšo najdeno rešitev.

6. REFERENCES

[1] C. Darwin. R.(1859): On the origin of species by means of natural selection. *Murray. London*, 1871.

[2] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.

[3] W. R. Evans. Control system synthesis by root locus method. *American Institute of Electrical Engineers, Transactions of the*, 69(1):66–69, 1950.

[4] D. Fister. *Načrtovanje samonastavljivega regulatorja 2 DOF robota s pomočjo BA algoritma*. Diplomsko delo : Univerza v Mariboru, Fakulteta za strojništvo, 2015.

[5] I. Fister, D. Strnad, X.-S. Yang, and I. Fister Jr. Adaptation and hybridization in nature-inspired algorithms. In *Adaptation and Hybridization in Computational Intelligence*, pages 3–50. Springer, 2015.

[6] I. Fister Jr, D. Fister, and X.-S. Yang. A hybrid bat algorithm. *arXiv preprint arXiv:1303.6310*, 2013.

[7] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[8] A. Jagarinec. *Adaptivni regulator z mehko logiko za dvoosni SCARA mehanizem*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[9] L. H. Keel and S. P. Bhattacharyya. A bode plot characterization of all stabilizing controllers. *Automatic Control, IEEE Transactions on*, 55(11):2650–2654, 2010.

[10] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.

[11] M. Kolar. *Vodenje SCARA robota z mehko logiko*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[12] H. Makino, N. Furuya, K. Soma, and E. Chin. Research and development of the scara robot. In *Proceedings of the 4th International Conference on Production Engineering*, pages 885–890, 1980.

[13] T. Slanič. *Genetski regulator za dvoosnega SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[14] A. M. Turing. Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, page 105, 1948.

[15] J. Čas. *Izdelava zveznega nevronskega sliding-mode regulatorja za teleoperiranje SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[16] M. Čurkovič. *Vgrajeni sistemi DSP/FPGA v sistemih vodenja*. Magistrsko delo Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2010.

[17] R. Šafarič and A. Rojko. *Inteligentne regulacijske tehnike v mehatroniki*. Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[18] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.