Univerza v Mariboru

Fakulteta za strojništvo

Načrtovanje samonastavljivega regulatorja 2 DOF robota s pomočjo BA algoritma

Diplomsko delo

Študent(ka): Dušan FISTER

Študijski program: Univerzitetni študijski program 1. stopnje Mehatronika

Mentor FS: red. prof. dr. Miran BREZOČNIK

Mentor FERI: red. prof. dr. Riko ŠAFARIČ

Somentor FERI: doc. dr. Iztok Fister

Maribor, 2015

Univerza v Mariboru

Fakulteta za strojništvo

Smetanova ulica 17
2000 Maribor, Slovenija

Številka: MEH-80052
Datum in kraj: 27.05.2015, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 46/2012)
izdajam

SKLEP O DIPLOMSKEM DELU

DUŠANU FISTERJU, študentu **univerzitetnega** študijskega programa **1. STOPNJE MEHATRONIKA**, se dovoljuje izdelati diplomsko delo.

Mentor FS: **red. prof. dr. Miran Brezočnik**
Mentor FERI: **red. prof. dr. Riko Šafarič**
Somentor FERI: **doc. dr. Iztok Fister**

Naslov diplomskega dela: **Načrtovanje samonastavljivega regulatorja 2 DOF robota s pomočjo BA algoritma**

Naslov diplomskega dela v angleškem jeziku: **Self-tuning controller design of 2 DOF robot using BA algorithm**

Diplomsko delo je potrebno izdelati skladno z »Navodili za izdelavo diplomskega dela« in ga oddati v treh izvodih do **30.09.2015** v referatu za študentske zadeve članice.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:

red. prof. dr. Niko Samec

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv

I Z J A V A

Podpisani Dušan Fister izjavljam, da:

- je bilo predloženo diplomsko delo opravljeno samostojno pod mentorstvom Mirana Brezočnika in Rika Šafariča;
- predloženo diplomsko delo v celoti ali v delih ni bilo predloženo za pridobitev kakršnekoli izobrazbe na drugi fakulteti ali univerzi;
- soglašam z javno dostopnostjo diplomskega dela v Knjižnici tehniških fakultet Univerze v Mariboru.

Maribor, _____ Podpis: _____

ZAHVALA

Zahvaljujem se mentorjema Miranu Brezočniku in Riku Šafariču ter somentorju Iztoku Fistru za pomoč in vodenje pri opravljanju diplomskega dela. Zahvaljujem se tudi vsem, ki so me v kakršnikoli obliki vzpodbujali pri nastanku tega dela.

Posebna zahvala velja staršem, ki so mi omogočili študij.

Načrtovanje samonastavljivega regulatorja 2 DOF robota s pomočjo BA algoritma

Ključne besede: algoritmi po vzoru obnašanja netopirjev, mehatronika, regulacije

UDK: 004.021:681.51(043.2)

POVZETEK

Algoritmi po vzoru iz narave dandanes pokrivajo več raziskovalnih področij. Aplikacije, s katerimi rešujemo različne optimizacijske probleme v industriji in drugih področjih človekove dejavnosti, povečujejo kakovost proizvoda, zmanjšujejo časovni okvir načrtovanja ali kako drugače lajšajo reševanja problema. Naš problem predstavlja načrtovanje parametrov položajnega regulatorja na dvoosnem robotskem mehanizmu s pomočjo algoritmov po vzoru iz narave. Pri tem med seboj primerjamo genetski algoritem ter algoritem po vzoru obnašanja netopirjev. Spoznamo in potrdimo osnovne značilnosti obeh algoritmov, ju preizkusimo na realni laboratorijski aplikaciji, ter ugotovimo, da algoritem po vzoru obnašanja netopirjev izboljšuje rezultate genetskega algoritma na našem problemu. S tem smo prišli do sklepa, da lahko algoritme po vzoru iz narave uspešno uporabimo za reševanje realnih problemov iz prakse.

Self-tuning controller design of 2 DOF robot using BA algorithm

Key words: nature-inspired algorithms, mechatronics, controlling

UDK: 004.021:681.51(043.2)

ABSTRACT

Nature-inspired algorithms present a bright star of researching. Implemented on different optimizational, industrial or researchable applications they offer an excellent opportunity to increase quality of product, reduce time needed for design or ease the procedure of solving modern problems. Our main topic presents the 2 dof robotic mechanism, equipped with nature-inspired algorithms in order to design self-tuning controller. We tested and compared two algorithms - Genetic algorithm and Bat algorithm, met their basics of working and according to real laboratory tests confirmed the execution of optimizational process. Bat algorithm did prove better working in the laboratory experiments. We conclude, that it was worth to use the nature-inspired algorithms for our problem.

KAZALO

UPORABLJENI SIMBOLI

F	–	sila
T	–	navor sile
m	–	masa
ρ	–	gostota
l	–	dolžina
I	–	težiščni vztrajnostni moment

UPORABLJENE KRATICE

BA	–	Algoritem po vzoru obnašanja netopirjev
GA	–	Genetski algoritem
SCARA	–	Selective Compliance Assembly Robot Arm
UM	–	Univerza v Mariboru
FS	–	Fakulteta za strojništvo
FERI	–	Fakulteta za elektrotehniko, računalništvo in informatiko

1 UVOD

1.1 Opis splošnega področja diplomskega dela

Navzočnost mehatronike v praksi dandanes prinaša nov tehnološki napredek. Inženirji so to tehniško disciplino uporabljali sprva za lajšanje pri reševanju vsakodnevnih problemov ter nadomeščanju človeške funkcije, medtem ko jo danes nevede uporablja širša množica ljudi tudi izven tehniških okvirov – npr. avtomatski zaviralni sistem (angl. Automatic Break System, krajše ABS), palični mešalnik, itd. V današnjem času ima pri razvoju novih naprav odločilno vlogo čas, pri čemer je naš glavni cilj opraviti nalogo v čim krajšem času dovolj kvalitetno.

Glavna tema naše diplomske naloge je dvoosni robot, ki mu želimo z optimizacijskim algoritmom poiskati optimalne parametre PI-položajnega regulatorja za vsako os posebej. Parametri regulatorja sestavljeni iz dveh dvojic (za vsak P in I člen) na vsako os (prva in druga os) morajo biti načrtani v skladu z željami uporabnika. Načrtovanje parametrov poteka s pomočjo algoritmov računske inteligence. Ta zajema v splošnem sledeče vrste algoritmov: evolucijsko računanje (Evolutionary Computing, krajše EC), umetne nevronske mreže (Artificial Neural Networks, krajše ANN), mehko logiko (angl. Fuzzy Logic, krajše FL), umetne imunske sisteme (angl. Artificial Immune Systems, AIS) ter algoritme inteligence roja (angl. Swarm Intelligence, krajše SI) [23]. V skupino evolucijskega računanja spadajo: genetski algoritmi (angl. Genetic Algorithms, krajše GA) [26], evolucijske strategije (angl. Evolution Strategies, krajše ES) [3], genetsko programiranje (angl. Genetic Programming, krajše GP) [33], evolucijsko programiranje (angl. Evolutionary Programming, krajše EP) [2] ter diferencialna evolucija (angl. Differential Evolution, krajše DE) [40], medtem ko med algoritme inteligence roja štejemo naslednje vrste algoritmov: optimizacijo z roji delcev (angl. Particle Swarm Optimization, krajše PSO) [24, 31], algoritme po vzoru obnašanja netopirjev (angl. Bat Algorithm, krajše BA) [46], kukavičje iskanje (angl. Cuckoo Search, krajše CS) [16, 22], optimizacijo s kolonijami mravelj (angl. Ant Colony Optimization, krajše ACO) [10] ter več drugih.

V našem delu se osredotočamo predvsem na algoritme po vzorih iz narave, tj. evolucijske algoritme in algoritme inteligence roja. Algoritem po vzoru obnašanja netopirjev (angl. Bat Algorithm, krajše BA) predstavlja enega novejših, popularnejših ter učinkovitih algoritmov uporabljenega na več aplikacijskih domenah. Zaradi nezahtevne implementacije ter hitre

konvergence algoritma smo ga uporabili v sklopu naše diplomske naloge in njegove rezultate primerjali z rezultati genetskega algoritma, ki so ga v preteklosti razvili že naši predhodniki. Princip delovanja algoritma BA temelji na naravnem obnašanju netopirjev. Ti svoj plen lovijo ponoči z oddajanjem kratkih ultrazvočnih pulzov in merjenjem odmeva. Ta pojav, ki mu pravimo tudi eholokacija, smo zapisali z matematičnim modelom ter ga uporabili pri reševanju optimizacijskih problemov.

1.2 Opredelitev diplomskega dela

Diplomska naloge je zasnovana multi-disciplinarno, saj zajema področja mehatronike, regulacij, strojništva in računalništva. Naloga, s katero se spopadamo v njej, je nastavljanje parametrov položajnega regulatorja na realnem robotu SCARA. Ta je sestavljen iz dveh električnih motorjev in dveh, togo povezanih osi. Na motorja sta priključena inkrementalna dajalnika, ki služita za meritev položaja. Iz položaja lahko z upoštevanjem preprostih matematičnih enačb izrazimo vrednost prenihaja, statičnega pogreška ter nastavitvenega časa posameznih osi. Osnovno zahtevo pri ocenjevanju kakovosti obnašanja robota predstavlja hitrejše gibanje obeh osi za določen zasuk ob upoštevanju začetnih pogojev. Ti so podani kot želeni procent prenihaja ob minimalnem statičnem pogrešku ter nastavitvenem času. S pomočjo optimizacijskega algoritma nastavljamo dvojico parametrov, ki jo preizkusimo in vrednotimo v realni aplikaciji. V grobem postopek ponavljamo, dokler ne zadovoljimo vseh kakovostnih omejitev ali dosežemo maksimalno število ponavljanj. Končni rezultat in vhod v regulator predstavlja optimalna dvojica parametrov, ki jo določa kriterijska, oz. ocenjevalna funkcija.

1.3 Struktura diplomskega dela

Struktura naloge v nadaljevanju je naslednja: najprej opišemo uporabljeno aparaturno opremo, skupaj z opisom robota SCARA, na katerem izvajamo preizkuse ter vhodno/izhodne kartice DSP-2 Roby, ki služi za komunikacijo med robotom in računalnikom. Opišemo tudi močnostna ojačevalnika ter napajalnik. Nadaljujemo z opisom in razlago optimizacijskih algoritmov GA ter BA, nakar predstavimo simulacijske rezultate testiranj in rezultate, dobljene na realni, laboratorijski aplikaciji. Nalogo sklenemo z diskusijo ter iztočnicami za nadaljnje delo.

2 PREGLED STANJA OBRAVNAVANE PROBLEMATIKE

Področje našega raziskovanja je zanimivo iz dveh vidikov. Prvi vidik predstavlja robot kot mehatronski stroj, medtem ko drugi vidik predstavljajo implementirani optimizacijski algoritmi. V nadaljevanju opisujemo oba vidika podrobneje.

2.1 2-DOF robot

2-DOF robot smo v celoti razvili in izdelali na UM, FERI. Testiranja na tem robotu je pričel Albin Jagarinec leta 2005 z razvojem adaptivnega regulatorja. Njegovi izsledki so zbrani v njegovi diplomski nalogi [29]. Leto kasneje je njegovo delo nadaljeval Marko Kolar [32]. Istega leta sta Jure Čas ter Tomaž Slanič razvila zvezni nevronski drsni regulator (angl. sliding-mode regulator) [6] ter genetski algoritem za vodenje dvoosnega robota [39], ki hkrati predstavlja iztočnico našega raziskovalnega dela.

2.2 Predstavitev algoritmov po vzorih iz narave

2.2.1 Genetski algoritmi

Ideja genetskih algoritmov sloni na Darvinovi evoluciji [11, 9], po kateri se biološke vrste skozi generacije spreminjajo in prilagajo vplivom iz okolja. To pomeni, da imajo v naravi večje možnosti za preživetje najbolje prilagojeni (najuspešnejši) posamezniki [41]. Princip prilagajanja lastnosti posameznika vplivom iz okolja tehnologi poznajo že zelo dolgo. Prvi, ki mu je uspelo darvinistične metode preusmeriti v reševanje problemov je bil Turing, ki je leta 1948 predlagal t.i. genetsko oz. evolucijsko iskanje. Leta 1962 je Bremermann uspel njegovo zamisel implementirati kot računalniški algoritem in uspešno opravil prva testiranja optimizacije z evolucijo in mutacijo [4]. Do konca šestdesetih let so se na področju evolucijskih algoritmov oblikovale tri optimizacijske veje. Osnovno vejo je predstavljalo evolucijsko programiranje [2], ki so jo sooblikovali ameriški profesorji Fogel, Owens ter Walsh. Njihov rojak Holland je svojo idejo poimenoval genetski algoritem [26], medtem ko sta Rechenberg ter Schwefel na evropskih tleh razvila t.i. evolucijske strategije [37, 38]. V zgodnjih 1990 se je tem trem vejam pridružilo genetsko programiranje [33] in kasneje še diferencialna evolucija [40].

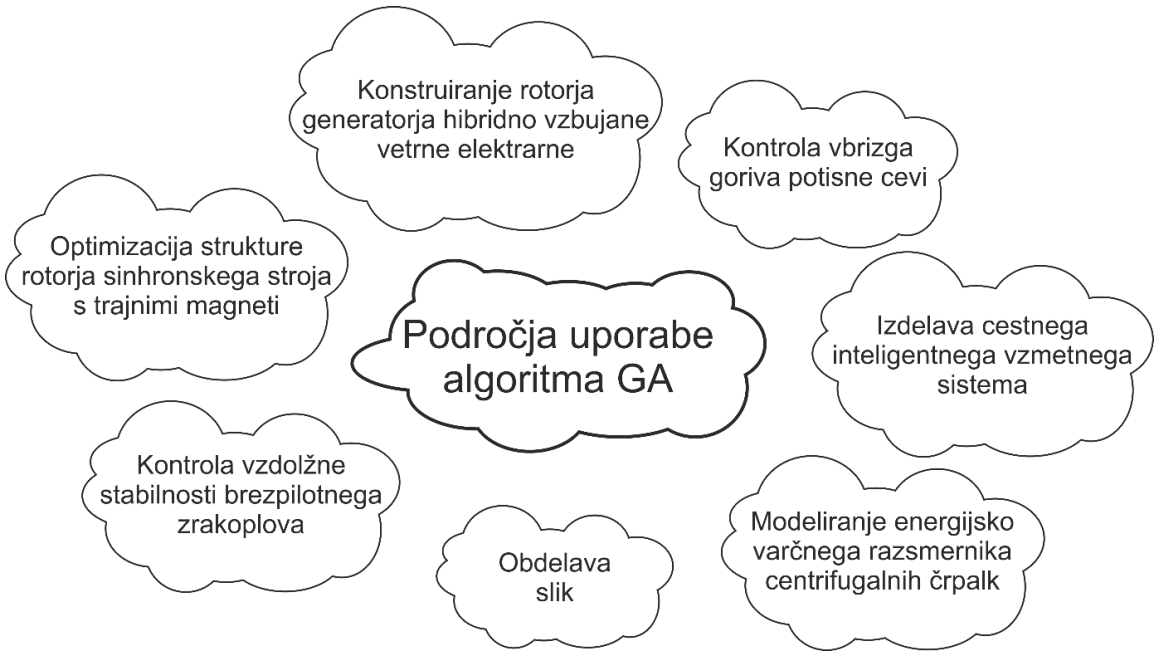
Vsem tem vejam je skupno, da algoritem razvija populacijo rešitev, ki predstavljajo posameznike, s pomočjo variacijskih operatorjev križanja in mutacije. Pri tem preživijo samo

najuspešnejši, ki prenesejo svoje najboljše lastnosti v naslednjo generacijo evolucijskega algoritma. Uspešnost rešitve (posameznika) v populaciji ocenimo glede na vrednost t.i. funkcije uspešnosti.

Do leta 1990 so se vse omenjene veje razvijale posamično, nakar so se vsa področja računalniške znanosti počasi združevale pod skupnim imenom evolucijsko računanje. Področja uporabe algoritma GA so zelo široka, saj ti algoritmi nudijo zanesljivo delovanje.

- kontrola vbrizga goriva potisne cevi (angl. ramjet fuel control) [27],
- obdelava slik (angl. image processing) [42],
- optimizacija strukture rotorja sinhronskega stroja s trajnimi magneti (angl. optimization of rotor structure in permanent magnet synchronous motors) [28],
- konstruiranje rotorja generatorja hibridno vzbujane vetrne elektrarne (angl. design of an outer rotor hybrid excited generator for wind energy conversion systems) [8],
- izdelava cestnega inteligentnega vzmetnega sistema (angl. intelligent road adaptive suspension design) [30],
- modeliranje energijsko varčnega razsmernika centrifugalnih črpalk (angl. energy-saving control model of inverter for centrifugal pump systems) [34],
- kontrola vzdolžne stabilnosti brezpilotnega zrakoplova (angl. UAV longitudinal control law design) [13],…

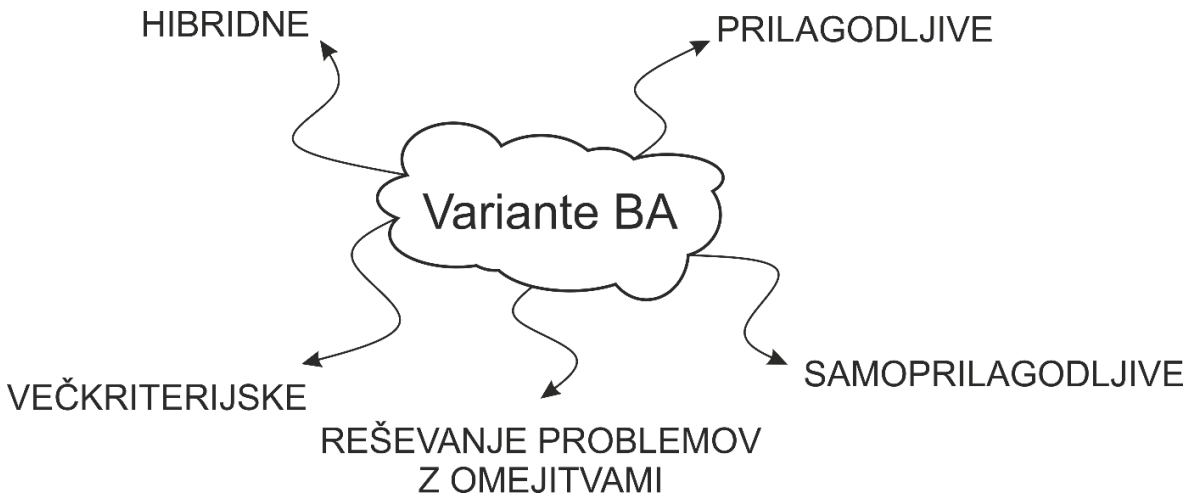
Najpomembnejša področja uporabe algoritmov GA so prikazana na sliki 2.1.



Slika 2.1: Področja uporabe algoritmov GA

2.2.2 Algoritmi na osnovi obnašanja netopirjev

Algoritem na osnovi obnašanja netopirjev je novejšega datuma, saj ga je razvil in implementiral Xin-SheYang [45, 46] šele leta 2010. Najprej so ga raziskovalci začeli uporabljati za potrebe numerične ter diskretne optimizacije. Kasneje je Yang razvil še algoritem BA za večkriterijsko optimizacijo [44], medtem ko so Gandomi in ostali [25] razvili algoritem BA za optimizacijo problemov z omejitvami. Fister in ostali [17] so izpostavili šibkosti algoritma BA, predvsem njegovo neučinkovitost pri reševanju problemov velikih dimenzij, in predlagali hibridni algoritem BA s strategijami diferencialne evolucije. Kasneje so originalni algoritem BA nekoliko izboljšali še s hibridizacijo z naključnimi gozdovi [18]. Poleg hibridizacije so, Fister in ostali [20, 21], vključili tudi samoprilagajanje kontrolnih parametrov, ki trenutno predstavlja eno izmed najuspešnejših variant algoritma BA. Slika 2.2 predstavlja variante BA, tj. hibridne, prilagodljive, samoprilagodljive, večkriterijske ter variante za reševanje problemov z omejitvami, grafično.

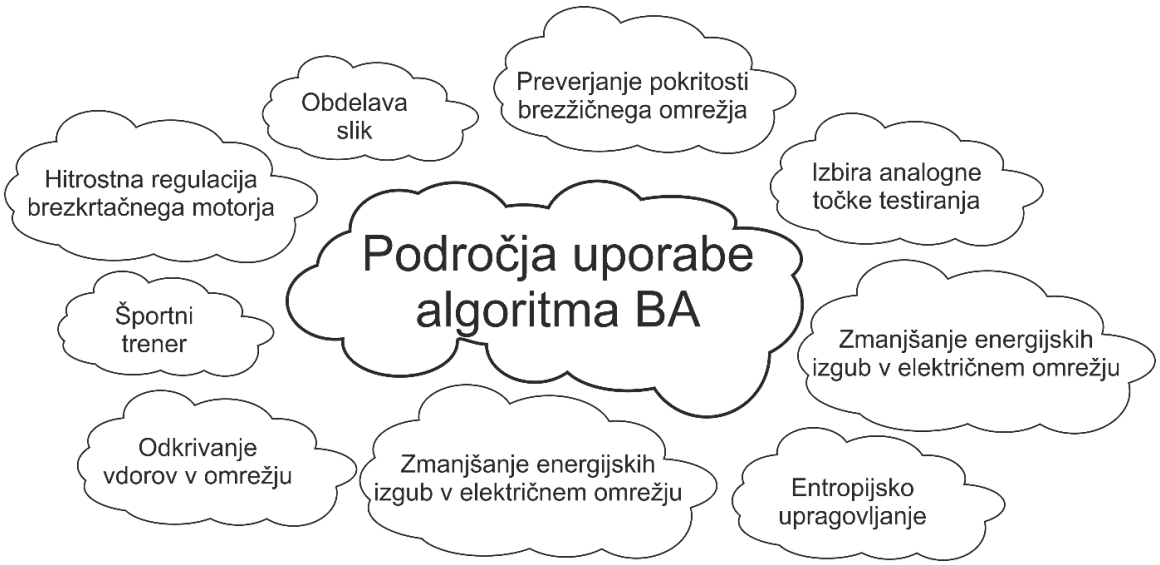


Slika 2.2: Variante BA

Kot kaže slika 2.3, je algoritem BA kljub njegovi mladosti moč uporabiti že na širokem spektru aplikacij. Številni raziskovalci so uporabili BA algoritem na naslednjih področjih:

- odkrivanje vdorov v omrežju (angl. intrusion detection)[12],
- preverjanje pokritosti brezžičnega omrežja (angl. coverage of wireless sensor network) [5],
- izbira analogne točke testiranja (angl. analog test point selection) [48],
- hitrostna regulacija brezkrtačnega motorja (angl. speed control of brushless DC motor) [36],

- zmanjšanje energijskih izgub v električnem omrežju (angl. power loss reduction in electrical distribution system) [1],
- entropijsko upragovljanje (angl. entropy based optimal thresholding) [47],
- obdelava slik (angl. image processing) [19, 35],
- športni trener (angl. planning sport training sessions) [15],…



Slika 2.3: Področja uporabe algoritma BA

3 JEDRO DIPLOMSKEGA DELA

3.1 Aparaturna oprema

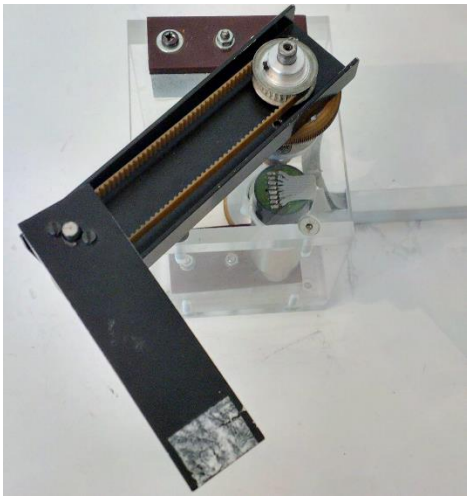
V nalogi smo uporabili sledečo aparaturno opremo:

- robota SCARA,
- vhodno/izhodna kartico DSP2-Roby,
- močnostna ojačevalnika,
- napajalnik in
- računalnik.

Vsi našteti, pravilno povezani deli, tvorijo celoto – sklop za testiranje optimizacijskih algoritmov. V nadaljevanju vse posamezne dele opisujemo natančneje.

3.1.1 Robot SCARA

Robot SCARA (angl. Selective Compliance Assembly Robot Arm) je štiriosni robotski mehanizem, ki je namenjen za natančnejša opravila v proizvodni liniji [6]. Razvil ga je profesor Hiroshi Makino z univerze Yamanashi [43]. Robot, ki ga uporabljamo za testiranja, je enostavnejše izvedbe in sestoji iz izključno dveh prostostnih stopenj, zato je vodenje motorjev nekoliko enostavnejše (slika 3.1).



Slika 3.1: Robot SCARA

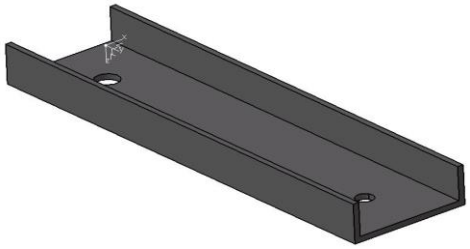
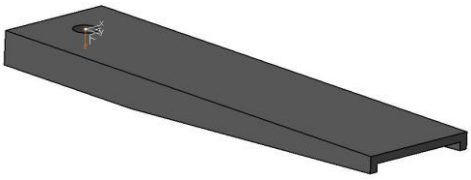
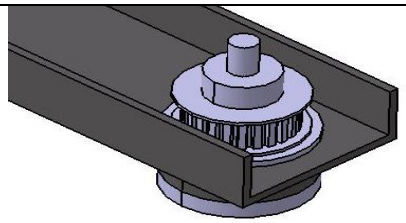
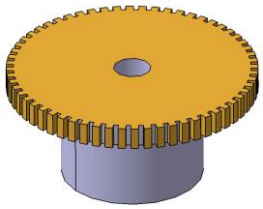
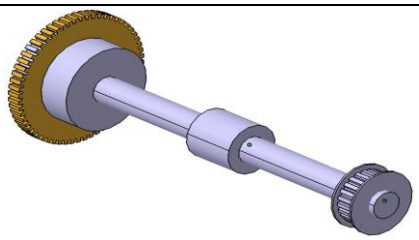
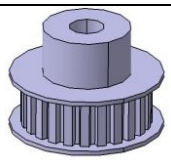
Zaradi svoje ne-industrijske izdelave ponuja robot SCARA odprt vstop v krmilno elektroniko. To pomeni, da lahko na njem izvajamo številne preizkuse, ki jih sicer na industrijskem robotu ne bi mogli. Med drugim lahko nastavljamo parametre položajnega regulatorja in jih po želji

optimiziramo, spreminjamo ter analiziramo. Osnovo za dobro poznavanje sistema predstavlja dinamični model robota, katerega je v [6] že izpeljal Jure Čas. V nadaljevanju diplomske naloge predstavljamo dinamični model robota SCARA nekoliko podrobneje.

3.1.1.1 Dinamični model robota SCARA

Dinamični model izpeljemo iz osnovnih mehanskih lastnosti robota SCARA. V tabeli 3.1 so razdelani in prikazani vsi sestavni deli tega robota. Poznavanje njihovih lastnosti je nujno za nadaljnjo analizo robota.

Tabela 3.1: Mehanske lastnosti robota

Ročica 1 $[m_2, I_2]$		gostota: ρ = 2710 kg/m³ (Aluminij) masa: m_2 = 0.053 kg težišče: l_{1T} = 57 mm dolžina: l_1 = 120 mm težiščni vztrajnostni moment za os rotacije: I_2 = 1.159e-4 kgm²
Ročica 2 $[m_4, I_4]$		gostota: ρ = 2710 kg/m³ (Aluminij) masa: m_4 = 0.024 kg težišče: l_{2T} = 45 mm dolžina: l_2 = 95 mm težiščni vztrajnostni moment za os rotacije: I_4 = 2.875e-5 kgm²
Sklop ležaja in jermenice $[m_3, I_3]$		gostota: ρ = 2710kg/m³ (Aluminij) gostota: ρ = 8750 kg/m³ (Jeklo) skupna masa: m_3 = 0.0737 kg težišče: 0 mm (od 2. osi rotacije) težiščni vztrajnostni moment za os rotacije: zanemarljiv
Zobnik 1 $[m_1, I_1]$		gostota: ρ = 2710 kg/m³ (Aluminij) masa: m_1 = 0.0667 kg težišče: 0 mm (od osi rotacije) težiščni vztrajnostni moment za os rotacije: I_1 = 1.280e-5 kgm²
Sklop jermenice, zobnika 2 in gredi $[I'_{30}]$		gostota: ρ = 2710 kg/m³ (Aluminij) gostota: ρ = 8750 kg/m³ (Jeklo) skupna masa: m = 0.1477 kg težišče: 0 mm (od osi rotacije) težiščni vztrajnostni moment za os rotacije: I'_{30} = 1.450e-5 kgm²
Jermenica $[I''_{30}]$		gostota: ρ = 2710 kg/m³ (Aluminij) masa: m = 0.012 kg težišče: 0 mm (od osi rotacije) težiščni vztrajnostni moment za os rotacije: I''_{30} = 4.50e-7 kgm²

Sledijo osnovne enačbe (enačbe 3.1-3.3), ki se pojavljajo v analizi dinamičnega modela. Posamezne spremenljivke so prikazane v tabeli 3.1.

$$a_1 = I_1 + I_2 + I_4 + m_2 \cdot l_{1T}^2 + (m_3 + m_4) \cdot l_1^2 + m_4 \cdot l_{2T}^2 \tag{3.1}$$

$$a_2 = m_4 \cdot l_1 \cdot l_{2T} \tag{3.2}$$

$$a_3 = I_4 + m_4 \cdot l_{2T}^2 \tag{3.3}$$

Predpostavimo, da lahko izraze 3.1, 3.2 in 3.3 zapišemo krajše. Iz teh okrajšanih enačb lahko dinamični model našega robota zapišemo kot matriko

$$\begin{bmatrix} \boldsymbol{T_1} \\ \boldsymbol{T_2} \end{bmatrix} = \begin{bmatrix} a_1 + 2 \cdot a_2 \cdot cos\,(q_2) & a_3 + a_2 \cdot cos\,(q_2) \\ a_3 + a_2 \cdot cos\,(q_2) & a_3 + I_{3O} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -a_2 \cdot \dot{q}_2 (2 \cdot \dot{q}_1 + \dot{q}_2) \cdot sin\,(q_2) \\ a_2 \cdot sin\,(q_2) \cdot \dot{q}_1^2 \end{bmatrix}. \tag{3.4}$$

3.1.1.2 Lastnosti motorjev

Za poznavanje obnašanja robota in hitrosti premikanja je treba podati tudi specifikacije motorjev z reduktorji. Lastnosti enosmernih motorjev s trajnimi magneti proizvajalca ESCAP so podane v tabeli 3.2.

Tabela 3.2: Lastnosti motorjev ESCAP

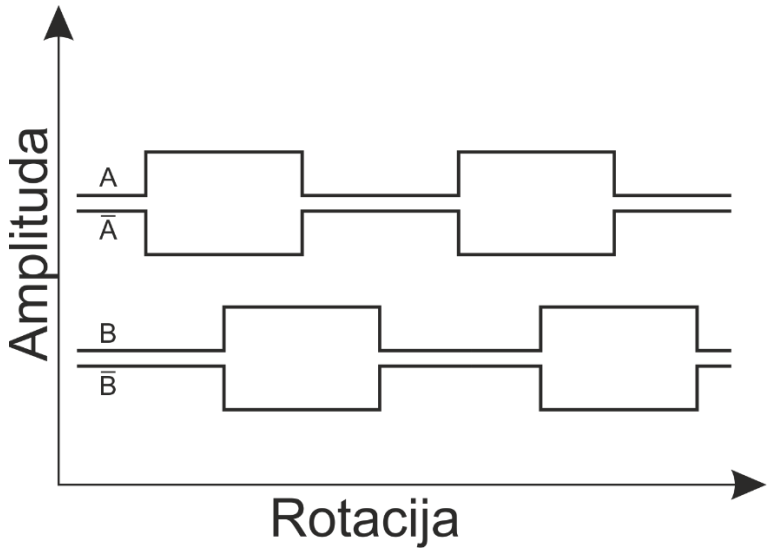
Proizvajalec in tip	ESCAP 28D11-219P
Moč [W]	15
Masa [g]	190
Nazivna napetost [V]	12
Nazivni tok [A]	1.5
Nazivni moment [mNm]	28.4
Vrtljaji prostega teka [min^{-1}]	5800
Tok prostega teka [mA]	44
Maksimalni radialni pospešek [10^3 rad/s^2]	48
Momentna konstanta [mNm/A]	19.6
Mehanska časovna konstanta [ms]	12
Vztrajnostni moment rotorja [$kgm^2\ 10^{-7}$]	17.6
Induktivnost rotorja [mH]	0.3

Komutacija motorja je mehanska, kar pomeni, da motorju za želeno hitrost vrtenja dovajamo enosmerno napetost. Magnituda napetosti je proporcionalna številu vrtljajev, oz. hitrosti vrtenja, medtem ko je tok proporcionalen navoru. Za pridobivanje podatkov o položaju, hitrosti ter smeri vrtenja je k vsakemu motorju povezan inkrementalni dajalnik. Podatke o zasuku

posreduje komparatorju in smernemu diskriminatorju, ki podatke obdela ter jih nadalje pošlje vhodno/izhodni kartici DSP-2 Roby. Ta z izračunavanjem prvega in drugega odvoda, kakor prikazano v enačbi 3.4, izvaja regulacijo sistema in obenem skrbi za pravilen potek optimizacije.

3.1.1.3 Inkrementalni dajalnik

Inkrementalni dajalnik deluje na principu štetja pulzov v pozitivno ter negativno smer. Sestavljen je iz svetlobne diode, inkrementalne plošče z črticami (režami) ter foto-tranzistorja. Naš robot meri zasuk s pomočjo dvo-kanalnega inkrementalnega dajalnika, kar pomeni, da sestoji iz dveh svetlobnih diod ter dveh foto-tranzistorjev. Generira signala A in B ter obe njuni negaciji, kar je prikazano na sliki 3.2. Dvojni signal uporabljamo za ugotavljanje smeri vrtenja. Če signal A prehiteva signal B pomeni, da se motor vrti v pozitivno smer ter obratno. Za dodatno zanesljivost delovanja si tri-kanalni inkrementalni dajalniki pridržujejo še signala I, oz. Z ter negacijo. Naša inkrementalna ploščica ima 100 črtic na obrat, smerni diskriminator natančnost poveča za faktor 4. Ločljivost našega inkrementalnega dajalnika tako znaša 400 črtic/obrat, kar pomeni natančnost pozicioniranja na $0.9°$. Ob uporabi reduktorskega sklopa se natančnost pozicioniranja osi mehanizma še poveča za stopnjo prestavnega razmerja reduktorja, saj je inkrementalni dajalnik prigrajen direktno na os enosmernega motorja.



Slika 3.2: Delovanje inkrementalnega dajalnika

Inkrementalni dajalnik za svoje delovanje uporabljamo devet bitni konektor RS-232, medtem ko je na vhodno/izhodno kartico DSP-2 Roby povezan deset bitni konektor. Zaradi neskladnosti obeh konektorjev nastajajo med njima določene razlike, ki jih je treba upoštevati med

načrtovanjem. V tabeli 3.3 je podan pomen vseh nogic obeh konektorjev. Iz tega sledi, da priključka s kablom nista neposredno vezana, ampak so žice prevezane. Uspešnost prevezave obeh kablov signalizirata dve diodi močnostnega ojačevalnika. Če z ročnim vrtenjem motorja spreminjata svetilnost pomeni, da je kabel pravilno povezan med oba priključka.

Tabela 3.3: Pomen devet in deset pinskih konektorjev

Številka nogice	*Pomen 9-bitnega konektorja*	*Pomen 10-bitnega konektorja*
1	Enc. A	NC
2	Enc. $\overline{A}$	Vcc (+5V)
3	Enc. B	GND
4	Enc. $\overline{B}$	$\overline{NC}$
5	GND	Enc. A
6	Enc. Z	Enc. $\overline{A}$
7	Enc. $\overline{Z}$	Enc. B
8	5 V	Enc. $\overline{B}$
9	Ni povezan	Enc.I
10		Enc. $\overline{I}$

3.1.2 Vhodno/izhodna kartica DSP-2 Roby

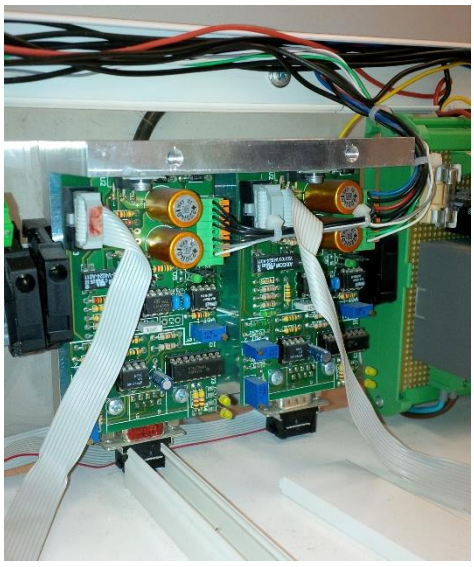
Vhodno/izhodna kartica DSP-2 Roby (v nadaljevanju kartica DSP-2 Roby) je namenjena za sprejemanje, obdelavo ter pošiljanje signalov. Medtem ko računalnik skrbi za optimizacijo krmilnih parametrov položajnega regulatorja, DSP-2 Roby (slika 3.3) skrbi za pravilno vodenje obeh motorjev. Njena sestava je podrobneje opisana v [7], medtem ko so poglavitne značilnosti omenjene v nadaljevanju.

Slika 3.3: Vhodno/izhodna kartica DSP-2 Roby

Jedro kartice DSP-2 Roby predstavlja digitalni signalni procesor tipa Texas TMS320C32. Odlikuje ga frekvenca takta 60 Mhz, podpora za računanje v formatu s plavajočo vejico, 256kB pomnilnik Flash ter 128 kB statični pomnilnik RAM. Na kartici DSP-2 Roby najdemo tudi programabilno logično vezje FPGA – XILINX Spartan XCS40-4. Kartico lahko na računalnik povežemo s povezavo RS-232, USB ter ethernet povezave, medtem ko kartica omogoča tudi povezavi RS-485 in CAN.

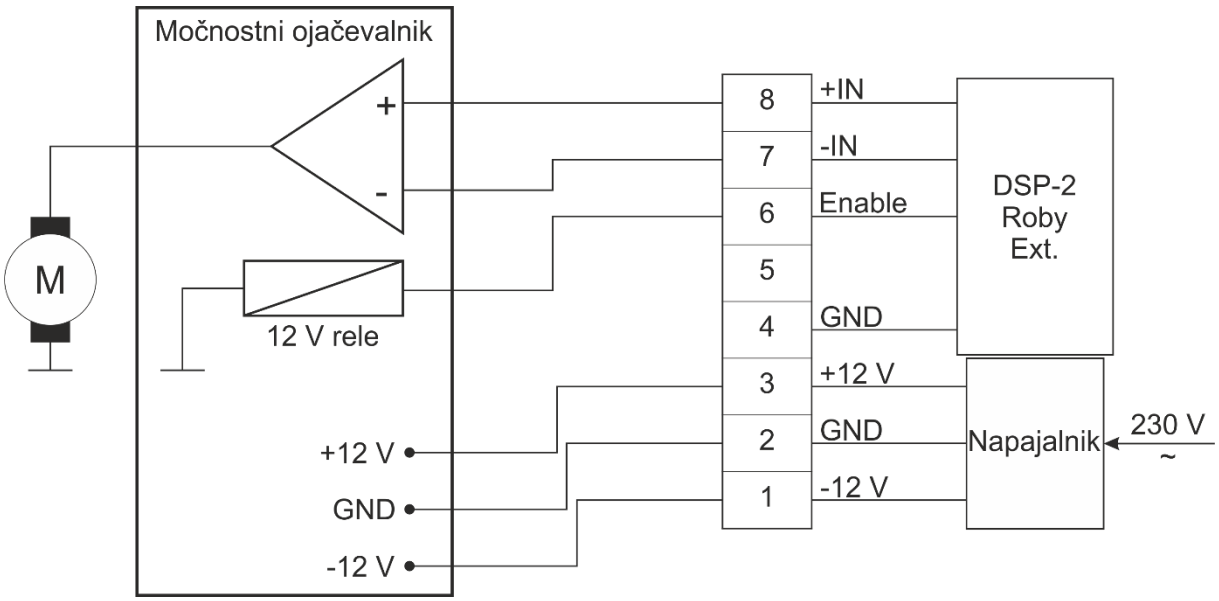
3.1.3 Močnostna ojačevalnika

Močnostna ojačevalnika (slika 3.4) služita za pretvorbo signalov nizkih moči v signale višjih moči ter obdelavo signalov inkrementalnih dajalnikov. Signali višjih moči napajajo motorje, medtem ko signali inkrementalnih dajalnikov služijo kot povratna zanka.



Slika 3.4: Močnostna ojačevalnika

Poleg RS-232 ter deset bitnega konektorja močnostni ojačevalnik za delovanje potrebuje še dodatno priključitev na napajalnik ter kartico DSP-2 Roby. Na sliki 3.5 so shematsko predstavljene povezave priključkov.

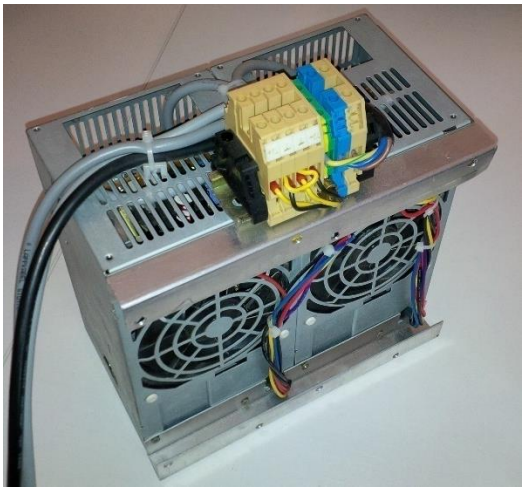


Slika 3.5: Povezave močnostnega ojačevalnika

Močnostni ojačevalnik pridobiva vir električne energije iz napajalnika. Zaradi možnega vrtenja motorja v pozitivno ter negativno potrebuje ojačevalnik tako pozitivno kot negativno napetost. Kartica DSP-2 Roby preko vmesne kartice DSP-2 Roby Ext. ojačevalniku posreduje nizko-energijski signal, ta pa jih s pomočjo ojačevalnih vezij ojači ter pošlje neposredno do motorja. Dodaten rele je namenjen za aktivacijo močnostnega ojačevalnika (angl. enable) in tako opravlja varnostno funkcijo.

3.1.4 Napajalnik

Napajalnik (slika 3.6) dobavlja energijo za mehansko in električno delo. Kot že omenjeno ga priključimo na omrežje, ta pa izmenično napetost usmeri v enosmerno velikosti dvanajst voltov.

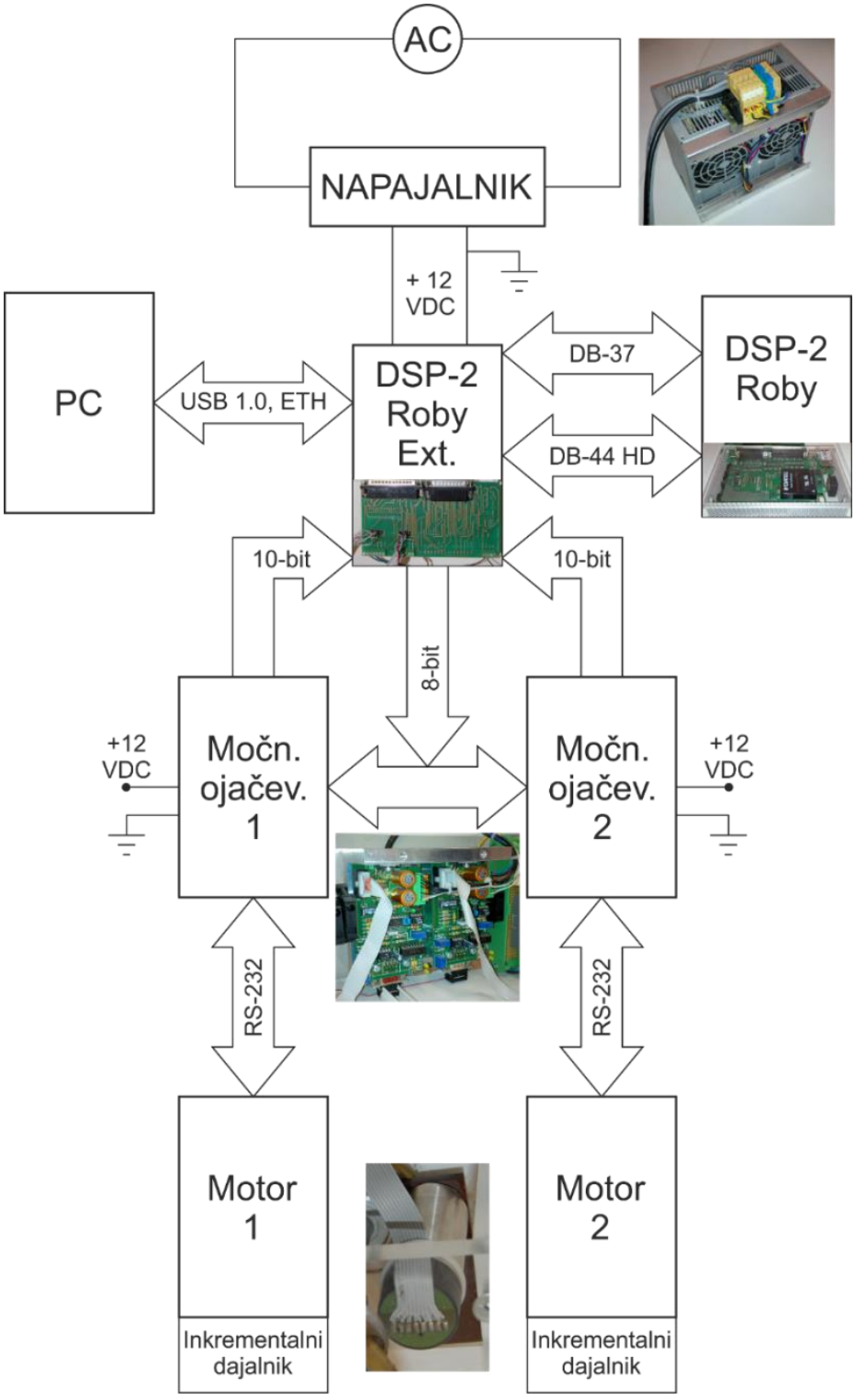


Slika 3.6: Napajalnik

3.1.5 Računalnik

Računalnik predstavlja najpomembnejšo komponento sklopa, saj omogoča modeliranje regulatorja, pisanje programske kode in izvajanje optimizacije. Tesno sodeluje s kartico DSP-2 Roby ter omogoča spremljanje optimizacijskega postopka in nadaljnjo analizo. Uporabljen računalnik ima za poganjanje zahtevane programske kode nameščen operacijski sistem Windows XP ter kompatibilne gonilnike za povezavo USB 1.0.

Slika 3.7 podrobneje prikazuje električni podsistem, v katerem osrednjo vlogo prevzema kartica DSP-2 Roby. Na kratko je opisan tudi način delovanja. Napajalnik omrežno napetost usmerja ter jo znižuje na 12V. Neposredno z napajalnikom je povezana razširitvena kartica DSP-2 Roby Ext., ki preko vodila DB-37 napaja kartico DSP-2 Roby. Razširitvena kartica prek USB 1.0 ali svetovnega spleta sprejema podatke od računalnika in jih posreduje kartici DSP-2 Roby. Ti podatki služijo kot napotki za delo, saj računalnik prevzema večino procesorskega dela. Razširitvena kartica podatke prenaša preko vodila z visoko gostoto DB-44 HD do kartice DSP-2 Roby, ta pa obdelane podatke pošilja prek osem bitnega kabla, po štiri za vsako os, do obeh močnostnih ojačevalnikov. Ti signal ojačijo ter takega posredujejo do obeh motorjev, ki opravljata mehansko delo. Kot povratna zanka služita inkrementalna dajalnika, ki signal pošiljata preko povezave RS-232 nazaj do močnostnih ojačevalnikov. Ta signal obdela ter pošlje preko deset bitnega konektorja nazaj na razširitveno kartico, ki jih kot regulacijski signal pošlje v računalnik.



Slika 3.7: Električni ter mehanski sistem

3.2 Genetski algoritem za optimizacijo parametrov položajnega regulatorja dvoosnega robota

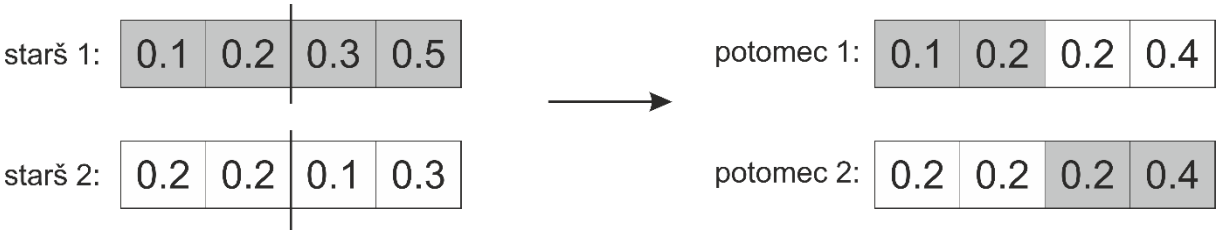
Genetski algoritem je eden izmed najosnovnejših vrst evolucijskih algoritmov [14]. Značilnost algoritma GA je, da rešitev išče s pomočjo populacije posameznikov, zato mu pravimo tudi populacijski algoritem. Osnovni elementi algoritma GA so:

- predstavitev posameznikov,
- selekcija staršev,
- križanje,
- mutacija,
- ocenitev novih rešitev ter
- selekcija preživelih.

Posamezniki v tipičnem GA so predstavljeni kot binarna števila. Kasneje so se razvili tudi realno kodirani genetski algoritmi.

V predlaganem GA za optimizacijo parametrov robota SCARA predstavimo posameznika kot vektor s štirimi realnimi števili, tj. parametri regulatorja, ki predstavljajo možno rešitev problema. Posameznike v začetni populaciji generiramo naključno. Selekcija staršev določa posameznike, ki vstopajo v proces križanja. Te posamezniki imenujemo tudi starši. Starše v GA lahko izbiramo na več načinov, npr. turnirska selekcija, ruleta, ipd. Pri tem najslabše posameznike glede na vrednost funkcije uspešnosti izločamo iz simuliranega evolucijskega procesa. Selekcija poteka na verjetnostni način, tj. če ima starš nižjo vrednost funkcije uspešnosti, ima večje možnosti, da ga v naslednji generaciji ne izberemo in obratno.

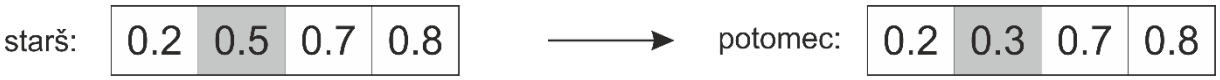
Križanje je najpomembnejši del algoritma GA. Ta operacija omogoča mešanje genetskega materiala in omogoča kreiranje kakovostnejših rešitev. Običajno uporabljamo aritmetično oz. diskretno križanje [11]. Pri aritmetičnem križanju dobimo vsak element potomca kot povprečje istoležnih elementov staršev, medtem ko pri diskretnem križanju izberemo vsak element potomca iz enega od obeh staršev z enako verjetnostjo. Križanje se ne izvaja na vseh starših, ampak je pogojeno s parametrom verjetnostjo križanja p_c. Slika 3.8 prikazuje križanje, torej dva starša, ki z križanjem njunega genetskega materiala tvorita dva potomca.



Slika 3.8: Enostavno aritmetično križanje

Pri aritmetičnem križanju (slika 3.8) najprej naključno določimo mesto križanja, od koder se elementi obeh potomcev določijo kot enostavno povprečje istoležnih elementov staršev. Tako lahko z določeno verjetnostjo napovemo, da iz dveh staršev pridobimo potomca, ki združuje pozitivne lastnosti obeh staršev in predstavlja boljšo rešitev problema. Z večanjem števila generacij neposredno vplivamo na kakovost optimizacijskih rezultatov.

Mutacija (slika 3.9) je naključna sprememba vrednosti poljubnega elementa vektorja rešitve. Izvedena je na izključno enem posamezniku, rezultat pa predstavlja potomca - mutanta.



Slika 3.9: Mutacija

Posamezniku naključno izberemo element vektorja in ga naključno spremenimo. Na sliki 3.9 smo naključno izbrali drugi element vektorja, katerega vrednost smo iz 0.5 naključno spremenili na 0.3. Fizikalno predstavlja mutacija izhod iz lokalnega optimuma. Aplicirana je le na peščici posameznikov, njeno pogostost določa parameter verjetnost mutacije p_m.

Ocenjevanje novih rešitev poteka s pomočjo ocenjevalne funkcije, ki vrednoti uspešnost rešitve. Ocenjevalno funkcijo za problem optimizacije parametrov robota SCARA zapišemo kot

$$f_i = E_{1i} \cdot \left(1 - \left|P_i - Over_i\right|\right) + E_{2i} \cdot \left(1 - Time_i\right) + E_{3i} \cdot \left(1 - Ess_i\right), \tag{3.5}$$

kjer pomenijo

E_{1i} – utež prenihaja posamezne osi,

P_i / rad – želeni prenihaj posamezne osi,

$Over_i$ / rad – izmerjeni prenihaj posamezne osi,

E_{2i} – utež nastavitvenega časa posamezne osi,

$Time_i$ / s – izmerjen nastavitveni čas posamezne osi,

E_{3i} – utež statičnega pogreška posamezne osi in

Ess_i / rad – statični pogrešek posamezne osi.

Domena vrednosti ocenjevalne funkcije v enačbi (3.5) je $f_i \in [0, 1]$, kjer ničla predstavlja najslabšo možno rešitev, medtem ko je ena idealna rešitev problema. Slednji se lahko le približamo, saj jo zaradi vztrajnostni elementov robota ter neidealnosti aplikacije v praksi ne moremo doseči.

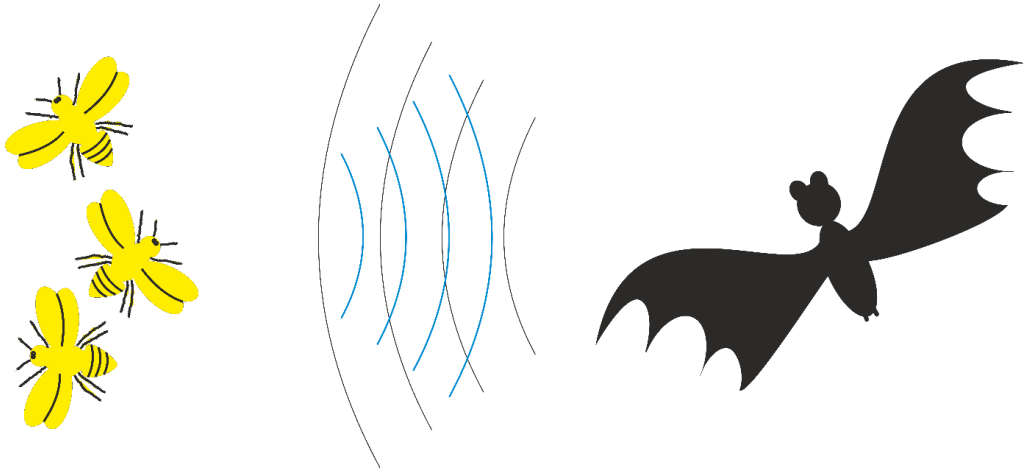
Uteži predstavljajo pomembnost posameznih izmerjenih parametrov. Vsota vseh treh uteži je ena, kar prikazuje enačba 3.6

$$\sum_{a=1}^{a=3} E_{ai} = 1. \tag{3.6}$$

Selekcija preživelih predstavlja zadnjo fazo evolucijskega cikla v GA. Pomen te selekcije je določiti najkakovostnejše posameznike, ki prenesejo svoje najboljše lastnosti v naslednjo generacijo, medtem ko slabše posameznike izločimo iz evolucijskega cikla. Selekcija preživelih je za razliko od selekcije staršev deterministična metoda, kar pomeni, da vedno izbiramo najuspešnejše posameznike za novo generacijo.

3.3 Algoritem na osnovi obnašanja netopirjev za optimizacijo parametrov položajnega regulatorja dvoosnega robota

Algoritem BA je eden izmed populacijskih algoritmov, ki posnemajo obnašanje inteligence rojev. Algoritem temelji na obnašanju t.i. mikronetopirjev, biološki vrsti s sposobnostjo eholokacije, ki omogoča zaznavanje in lovljenje plena, umikanje oviram na poti ter orientacijo v popolni temi. Pojav prikazuje slika 3.10.



Slika 3.10: Oddajanje ultrazvočnih signalov ter prejemanje odmeva

Za lažjo implementacijo uporablja algoritem nekaj poenostavitev glede na realni svet, ki so navedene v naslednjih alinejah:

- Netopirji so s pomočjo eholokacije sposobni določiti razdaljo do plena. Poleg tega znajo razločiti ali se odmev odbija od plena ali ovire.
- Netopir i med naključnim poletom leti s hitrostjo v_i na poziciji x_i s konstantno frekvenco oddajanja pulzov f_{min} ter spremenljivo valovno dolžino λ in glasnostjo A_0. Med približevanjem plenu lahko netopir spremeni tudi emisijo pulzov r, parameter ki pove kolikrat v sekundi netopir odda signal.
- Glasnost pulzov A_0 se med lovljenjem plena spreminja, zato določimo zgornjo mejo A_0 ter spodnjo mejo A_{min}.

Psevdo-kod je prikazan v algoritmu 1.

V algoritmu najprej inicializiramo novo populacijo, ki jo ovrednotimo z ocenjevalno funkcijo prikazano v enačbi 3.5. Za pravilno izvajanje algoritma je treba določiti tudi končni pogoj. To sta lahko bodisi doseženo predpisano število generacij ali dovolj kakovostna rešitev. Dokler eden izmed omenjenih pogojev ni izpolnjen, algoritem teče. Mi smo za končni pogoj uporabili predpisano število generacij N_p (vrstica 5 v algoritmu 1), ki jo je pred zagonom potrebno inicializirati. Glavni del optimizacije predstavlja vrstica 6 (funkcija generiraj_novo_rešitev), kjer v vsaki generaciji t posodobimo naključno pozicijo netopirja ter njegovo hitrost v skladu z enačbami 3.7, 3.8 ter 3.9

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \cdot \beta, \tag{3.7}$$

$$v_i^j(t) = v_i^j(t-1) + \left[\hat{x}^j - x_i^j(t-1)\right] \cdot f_i, \tag{3.8}$$

$$x_i^j(t) = x_i^j(t-1) + v_i^j(t), \tag{3.9}$$

kjer pomenijo

f_i – dejanska frekvenca, s katero bo netopir oddal zvočni impulz,

f_{min} in f_{max} – minimalna ter maksimalna frekvenca, s katero netopir lahko oddaja, razpon med 20 kHz in 200 kHz (ultrazvok – neslišno področje),

β – kontrolni parameter, $\beta \in [0,1]$,

$v_i^j(t)$ – dejanska hitrost posameznega netopirja,

$\hat{x}^j$ – najuspešnejša globalna pozicija in

x_i^j – dejanska pozicija.

Algoritem 1 Algoritem na osnovi obnašanja netopirjev

Vpis: Populacija netopirjev $\mathbf{x}_i = (x_{i1, \ldots,} xi_D)^T$ *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 N_p 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 {naključni sprehod}
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:        if f_new ≥ f_max then
16:            x_best = y; f_max = f_new;
17:        end if {shrani trenutno najboljšo rešitev}
18:        f_max = išči_najboljšo_rešitev(x_best);
19:    end for
20: end while
```

V zadnjem koraku prištejemo trenutni poziciji hitrost ter pridobimo novo pozicijo navideznega netopirja. Ta metoda predstavlja iskanje novih rešitev v preiskovalnem prostoru (angl. exploration).

Če je emisija pulzov *r* višja od naključnega števila rand(0,1) (vrstica 7 v algoritmu 1) poskušamo najboljšo rešitev izboljšati (vrstica 8 v algoritmu 1) z metodo naključnega sprehoda (angl. Random Walk). Yang [44] je s to metodo poskušal izkoristiti (angl. exploitation) znanje pridobljeno med procesom preiskovanja, zato je usmeril preiskovalni proces v izboljšanje trenutne najboljše rešitve. V tem primeru preoblikujemo posameznika $\mathbf{x}_{stari}$ po naslednji enačbi

$$\mathbf{x}_{novi} = \mathbf{x}_{stari} + \varepsilon \cdot \bar{A}(t), \tag{3.10}$$

kjer pomenita

ε – naključno število, $\varepsilon \in [-1,1]$ in

$\bar{A}(t)$ – povprečna glasnost vseh netopirjev posamezne generacije.

S parametrom r_i eksplicitno nadzorujemo procesa raziskovanja in izkoriščanja v preiskovalnem procesu. V vrstici 12 algoritma 1 predstavimo pogoj za shranjevanje najboljše rešitev populaciji. Le-to shranimo, če je vrednost ocenjevalne funkcije višja od trenutno najvišje ter če je naključna vrednost višja od inicializirane glasnosti. V naslednji vrstici najboljšo rešitev pogojno shranimo. Globalno najboljšo rešitev shranimo brez-pogojno (vrstica 16 algoritma 1).

Glasnost netopirjev A_i v vsaki generaciji spreminjamo po enačbi 3.11, medtem ko emisijo pulzov *r* po 3.12.

$$A_i(t+1) = \alpha \cdot A_i(t), \tag{3.11}$$

$$r_i(t+1) = r_i(0) + \left[1 - e^{-\gamma \cdot t}\right], \tag{3.12}$$

kjer pomenita

$\alpha, \gamma \in [0,1]$ – konstanti.

4 REZULTATI

Rezultate smo razdelili v dva dela. Prvi del zajemajo testiranja pridobljena na računalniškem modelu robota robota SCARA, drugi del pa z realnim laboratorijskim robotom SCARA.

4.1 Simulacijski rezultati

Spoznavanje z algoritmom BA, njegovimi osnovami, prednostmi ter slabostmi smo preizkusili s pomočjo simulacijskega modela izdelanega v orodju MATLAB/Simulink. Za primerjavo smo zaganjali tudi algoritem GA. Simulacijska testiranja so potekala na dva načina. Sprva smo algoritma primerjali po rezultatih, pridobljenimi z več zagoni, nakar smo opravili še testiranja po zahtevanih prenihajih. V obeh primerih smo ocenjevali uspešnost pri reševanju problema premika obeh osi za želeni kot dva radiana.

V našem primeru smo utež prenihaja nastavili na $E_{1i} = 0.4$, utež nastavitvenega časa ter statičnega pogreška pa $E_{2i} = 0.3$ ter $E_{3i} = 0.3$. Oba algoritma sta zaradi zagotavljanja poštenosti meritev imela enako nastavljena parametra velikost populacije $Np = 10$ ter število generacij $n_{gen} = 10$. Algoritem GA je med testiranji tekel z naslednjima parametroma: verjetnost mutacije $p_m = 0.4$ in verjetnost križanja $p_c = 0.8$. Algoritmu BA smo njegove parametre nastavili na: hitrost pulza $r = 0.1$, glasnost $A = 0.9$ ter faktor skaliranja $Q \in [0.5,\ 1.5]$.

4.1.1 Primerjava rezultatov testiranj algoritmov po več zagonih

V tem sklopu testiranj smo izvedli deset ponovnih zagonov vsakega algoritma. Beležili smo ocenjevalni funkciji prve osi f_1, druge osi f_2 ter optimizirane parametre, ki nakazujejo razvoj optimizacije in so pripravljene za vpis v regulator. Rezultate smo povzeli v tabeli 4.1.

Tabela 4.1: Rezultati algoritma GA

Zagon	f_1	f_2	Kp_1	Kp_2	Kv_1	Kv_2
1	0.9519	0.8649	48.12	35.89	14.46	9.14
2	0.9445	0.8803	106.06	179.35	16.01	20.35
3	0.9455	0.8813	126.48	126.39	22.79	15.09
4	0.9496	0.8845	171.85	177.23	22.14	19.33
5	0.9492	0.8812	166.55	56.79	24.71	12.30
6	0.9496	0.8852	54.65	56.79	9.88	12.30
7	0.9469	0.8835	125.83	185.16	21.36	20.61
8	0.9484	0.8817	177.62	130.71	21.81	8.18
9	0.9493	0.8824	164.93	91.43	23.39	20.36
10	0.9489	0.8891	46.12	130.71	23.39	33.31
$\overline{x}$	**0.9484**	**0.8814**	**118.82**	**117.05**	**19.99**	**17.10**

Povzeti so najboljši rezultati iz posameznih testiranj. Med parametri nastajajo določene razlike zaradi različno generirane začetne populacije. Izvedenih je deset testiranj, medtem ko krepko napisana vrstica prikazuje skupno povprečje. Za primerjavo smo izvedli tudi deset testiranj algoritma BA, čigar rezultati so predstavljeni v tabeli 4.2.

Tabela 4.2: Rezultati algoritma BA

Zagon	f_1	f_2	Kp_1	Kp_2	Kv_1	Kv_2
1	0.9468	0.8900	126.72	606.62	18.27	43.99
2	0.9515	0.8895	225.85	583.13	25.94	56.58
3	0.9518	0.8920	245.37	259.92	25.86	44.91
4	0.9492	0.8905	210.85	562.22	22.86	51.34
5	0.9479	0.8882	148.37	469.42	19.68	51.00
6	0.9513	0.8914	218.33	532.08	25.50	46.72
7	0.9525	0.8885	207.67	385.63	29.67	42.71
8	0.9489	0.8861	157.30	285.75	21.49	37.65
9	0.9507	0.8902	200.55	401.75	24.27	40.36
10	0.9505	0.8891	208.43	406.58	25.44	43.31
$\overline{x}$	**0.9501**	**0.8896**	**194.94**	**449.31**	**23.90**	**45.86**

Iz obeh testiranj je razvidna razlika – algoritem BA načeloma dosega višjo ocenjevalno funkcijo, kar pomeni, da se za določene zahteve odraža boljše kot algoritem GA. Dosega precej višje konstante parametrov, kar pomeni, da se lahko robot hitreje premika, zaradi tega pa dosega krajše odzivne čase. Da bi dokazali uspešnejše delovanje algoritma BA smo izvedli tudi Wilcoxonov ne-parametrični test in ugotovili signifikantno boljše rezultate algoritma BA pri stopnji zaupanja 0.05.

4.1.2 Primerjava rezultatov testiranj po zahtevanih prenihajih

Druga primerjava rezultatov omenjenih algoritmov predstavlja testiranja po zahtevanih prenihajih, ki je v praksi najbolj razširjena. Čeprav je za resno delo z robotom pomemben izključno osnovni test, ko zahtevana prenihaja za obe osi znašata nič, so bili zaradi stohastičnosti algoritmov opravljeni naslednji dodatni testi:

1. testiranje - zahtevana prenihaja $P_1 = 0$ %, $P_2 = 0$ %,
2. testiranje - zahtevana prenihaja $P_1 = 15$ %, $P_2 = 15$ %,
3. testiranje - zahtevana prenihaja $P_1 = 25$ %, $P_2 = 15$ % in
4. testiranje - zahtevana prenihaja $P_1 = 30$ %, $P_2 = 30$ %.

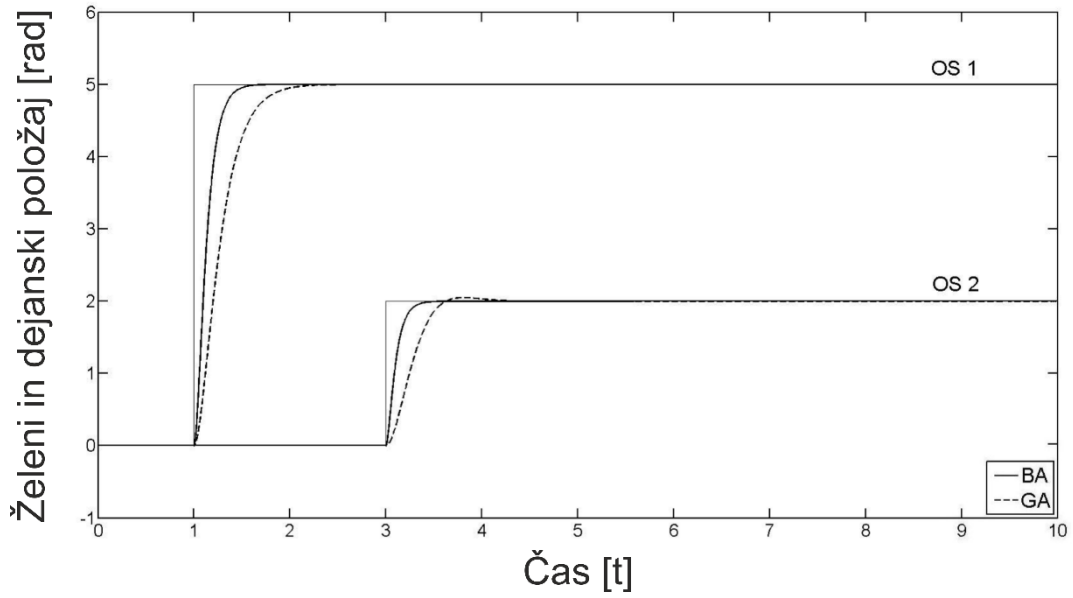
Delovanje z zahtevanim prenihajem 0 % je značilno za vse industrijske robote. Ti med opravljanjem dela velikokrat zamenjujejo orodje in s tem spreminjajo vztrajnostne momente ročic. Ob napačnem določanju parametrov regulatorja lahko robot povzroči katastrofalne posledice. Rezultati testiranja so prikazani na dva načina – tabelarično in grafično.

4.1.2.1 Testiranje 1

Rezultate prvega testiranja z zahtevanima prenihajema $P_1 = 0$ %, $P_2 = 0$ % prikazujeta tabela 4.3 ter slika 4.1.

Tabela 4.3: Tabelarična upodobitev prvega testiranja ($P_1 = 0$ %, $P_2 = 0$ %)

os	*algoritem*	*zahtevani prenihaj [%]*	*izmerjeni prenihaj [%]*
1	GA	0	0
	BA	0	0
2	GA	0	2.45
	BA	0	0



Slika 4.1: Grafična upodobitev prvega testiranja (P_1 = 0 %, P_2 = 0 %)

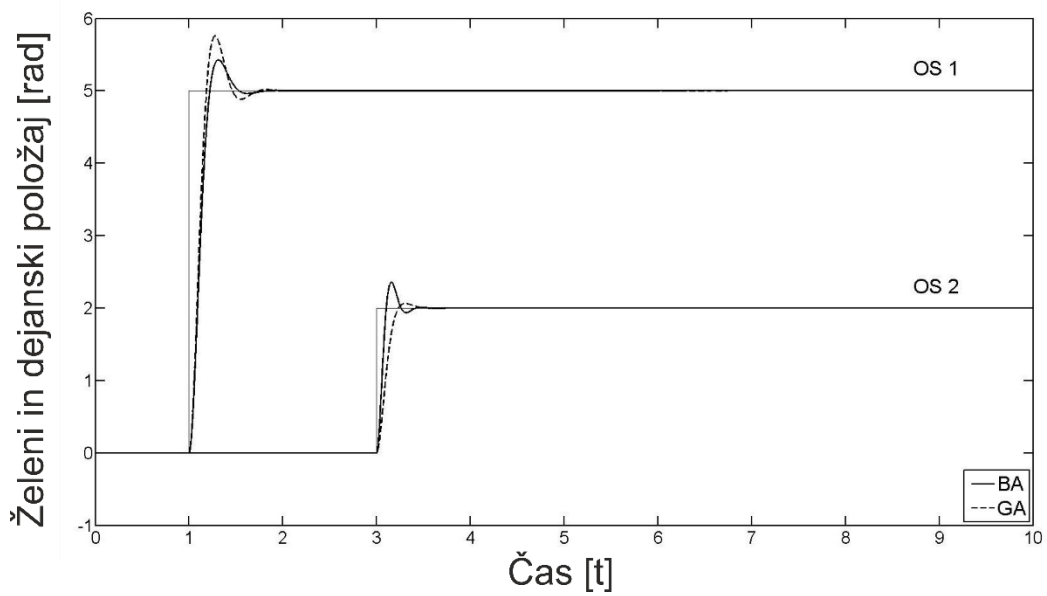
Algoritem BA je prikazan s polno črto, algoritem GA pa s črtkano. Glede na odziv vidimo, da za prvo os noben izmed algoritmov ni zabeležil prenihaja. Kljub temu je algoritem BA dosegel hitrejši vzponski čas. Prenihaj druge osi je natančneje načrtal algoritem BA, saj algoritem GA ni upošteval zahtevanega prenihaja 0 %. Poleg tega je dosegel daljši vzponski čas, kar dodatno poslabšuje njegov odziv. V tem primeru je algoritem BA izkazoval kakovostnejše delovanje.

4.1.2.2 Testiranje 2

Rezultati drugega testiranja z zahtevanima prenihajema P_1 = 15 % in P_2 = 15 % so prikazani v tabeli 4.4 ter sliki 4.2.

Tabela 4.4: Tabelarična upodobitev drugega testiranja (P_1 = 15 %, P_2 = 15 %)

os	*algoritem*	*zahtevani prenihaj [%]*	*izmerjeni prenihaj [%]*
1	GA	15	15.18
	BA	15	8.58
2	GA	15	3.15
	BA	15	17.90



Slika 4.2: Grafična upodobitev drugega testiranja (P_1 = 15 %, P_2 = 15 %)

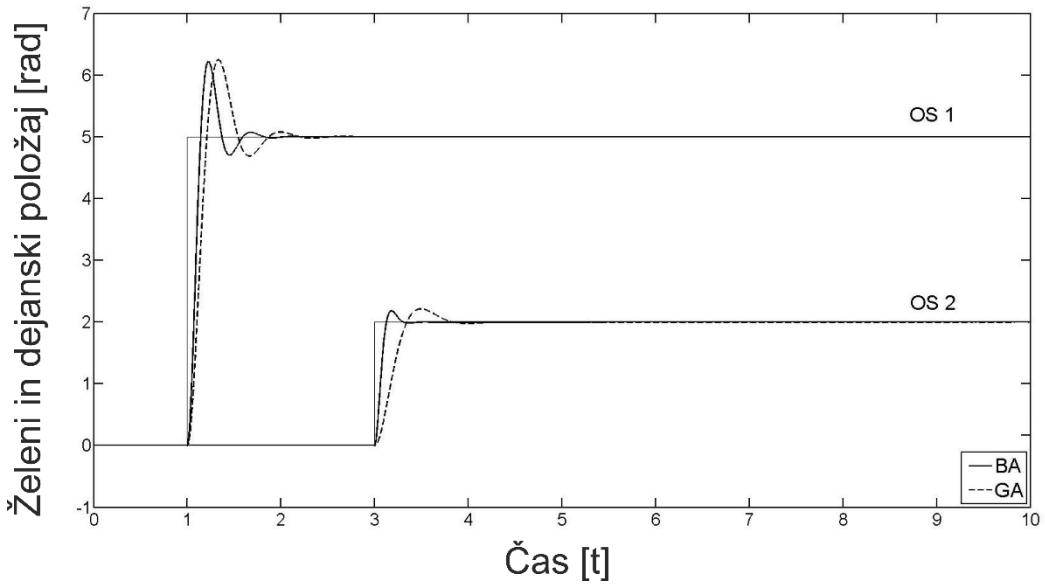
V tem testu noben izmed algoritmov ni izpolnil zahtevane naloge. Algoritem BA je za prvo os načrtal premajhni prenihaj, medtem ko za drugo os nekoliko prevelikega. Algoritem GA je prenihaj za prvo os natančno načrtal, vendar za drugo os napovedal veliko premajhnega. Nastavitvena časa za prvo os sta podobna, razlike ni opaziti. Nekoliko krajši nastavitveni čas druge osi je dosegel algoritem BA.

4.1.2.3 Testiranje 3

Rezultati tretjega testiranja z zahtevanima prenihajema P_1 = 25 % in P_2 = 10 %, ki je zajemalo preizkušanje pri neenakih prenihajih prve in druge osi, so prikazani v tabeli 4.5 in sliki 4.3.

Tabela 4.5: Tabelarična upodobitev tretjega testiranja (P_1 = 25 %, P_2 = 10 %)

os	*algoritem*	*zahtevani prenihaj [%]*	*izmerjeni prenihaj [%]*
1	GA	25	24.96
	BA	25	24.40
2	GA	10	10.65
	BA	10	9.20



Slika 4.3: Grafična upodobitev tretjega testiranja (P_1 = 25 %, P_2 = 10 %)

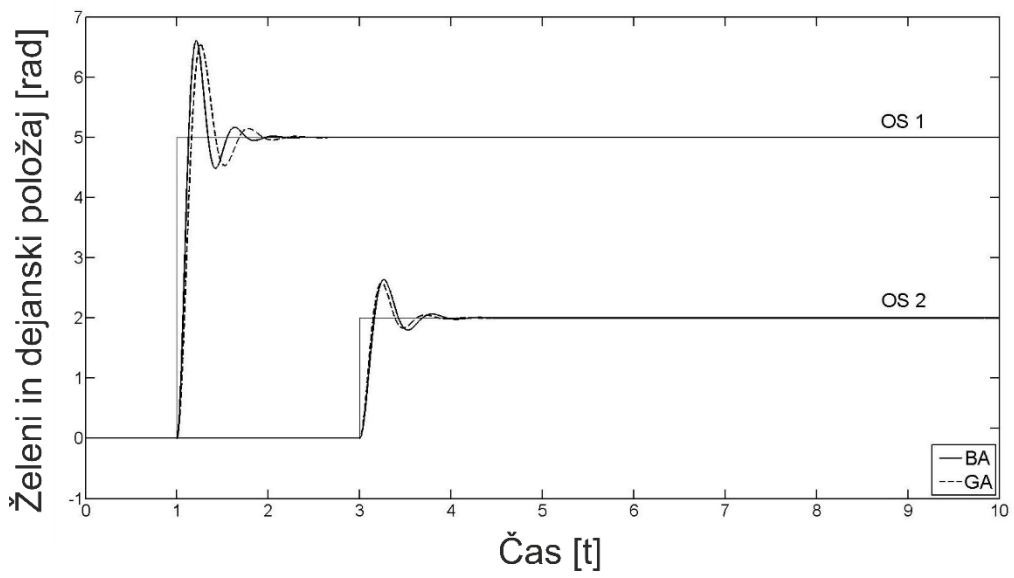
V tem primeru sta oba algoritma natančno načrtala prenihaj prve osi ter dokaj natančno prenihaj druge osi. Algoritem GA je načrtal nekoliko prevelikega, medtem ko algoritem BA nekoliko premajhnega. Iz grafa je razviden hitrejši nastavitveni čas algoritma BA. Tudi v tem primeru je algoritem BA izkazoval kakovostnejše delovanje.

4.1.2.4 Testiranje

Zadnje testiranje je potekalo pri največjih zahtevanih prenihajih P_1 = 30 %, P_2 = 30 %. Rezultati testov so predstavljeni v tabeli 4.6 ter grafu 4.4.

Tabela 4.6: Tabelarična upodobitev četrtega testiranja (P_1 = 30 %, P_2 = 30 %)

os	*algoritem*	*zahtevani prenihaj [%]*	*izmerjeni prenihaj [%]*
1	GA	30	31.80
	BA	30	32.08
2	GA	30	29.20
	BA	30	30.66



Slika 4.4: Grafična upodobitev četrtega testiranja (P_1 = 30 %, P_2 = 30 %)

Rezultati četrtega testiranja so nekoliko porazdeljeni. Algoritem GA izkazuje natančnejše načrtovanje prenihaja prve osi, a zabeležuje daljši nastavitveni čas. Prenihaja druge osi sta oba algoritma skrbno načrtala. Nekoliko krajši nastavitveni čas je sicer zabeležil algoritem GA ter s celostno podobo izboljšal rezultat algoritma BA.

4.2 Rezultati testiranj realne laboratorijske aplikacije

Algoritem BA smo testirali tudi na realnem laboratorijskem robotu SCARA. Za testiranje je bilo potrebno poleg programskega orodja MATLAB/Simulink namestiti tudi knjižnico DSP-2 ter potrebne gonilnike. Postopek in osnovne značilnosti dodatkov je podrobneje opisal Jure Čas v [6]. Testiranja so sestavljena iz treh testov glede na zahtevane prenihaje:

1. testiranje - zahtevana prenihaja P_1 = 0 %, P_2 = 0 %,
2. testiranje - zahtevana prenihaja P_1 = 10 %, P_2 = 10 % in
3. testiranje - zahtevana prenihaja P_1 = 15 %, P_2 = 25 %.

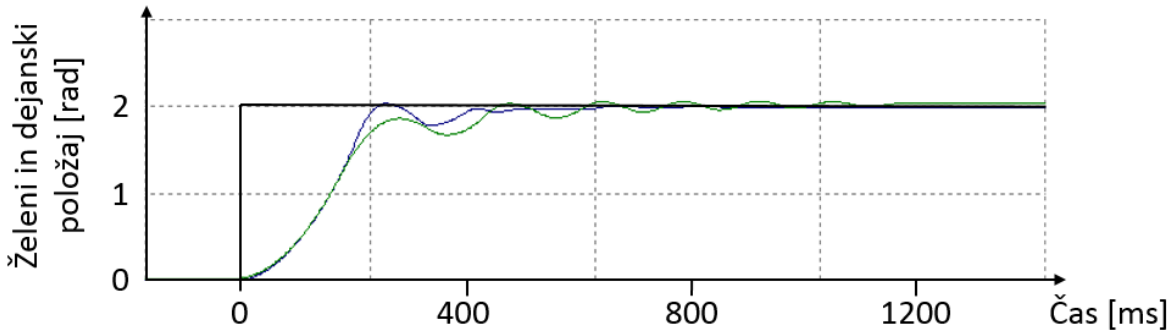
Zaradi težav s pogonskim sklopom nadaljnjih testov po več zagonih in testiranj algoritmov GA nismo opravili.

4.2.1 Testiranje 1

Rezultati s prenihajema P_1 = 0 %, P_2 = 0 % so po vzoru prejšnjih testov zbrani v tabeli 4.7, prikazan pa je tudi pripadajoči graf 4.5.

Tabela 4.7: Tabelarična upodobitev prvega testiranja (P_1 = 0 %, P_2 = 0 %)

os	*P* [%]	*Over* [%]	*Time* [s]	*Ess* [rad]	*f*
1	0	1.87	0.906	0.00058	0.9832
2	0	0	0.504	0.00544	0.9651



Slika 4.5: Grafična upodobitev prvega testiranja (P_1 = 0 %, P_2 = 0 %)

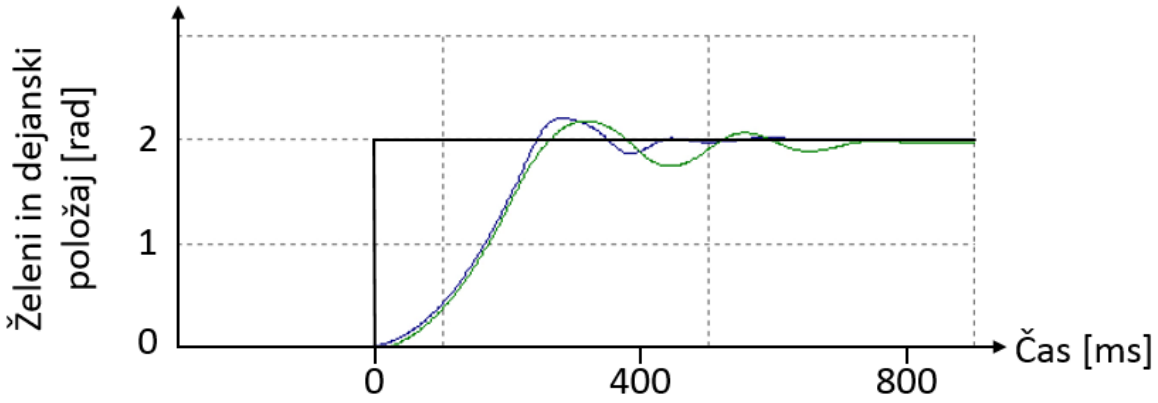
Pridobljene rezultate realne aplikacije smo vrednotili na enak način, kot smo vrednotili simulacijske rezultate, izvedene na računalniškem modelu. Ocenjevalna funkcija je vrednotila posamezne rešitve premika obeh osi za dva radiana. Algoritmu BA smo podali široki interval vrednosti možnih parametrov, zato je rešitev tega problema manj kvalitetna. Pojavlja se dolg nastavitveni čas, katerega bi lahko z manjšanjem intervala vrednosti parametrov odpravili, a s tem zmanjšali število možnih kombinacij vrednosti prenihajev. Prenihaja, ki sta znašala nič, sta bila natančno načrtana. Tudi statični pogrešek je znašal minimalno vrednost. Odziv bi lahko še nekoliko izboljšali s ponovnim testiranjem, saj bi meje parametrov nastavili na okolico desetih odstotkov trenutne najboljše rešitve. Algoritem bi zato rešitev iskal v tem zelo skrajšanem obsegu.

4.2.2 Testiranje 2

Rezultate drugega testiranja s prenihajema P_1 = 10 %, P_2 = 10 % ponazarjata tabela 4.8 ter slika 4.6.

Tabela 4.8: Tabelarična upodobitev drugega testiranja

os	*P* [%]	*Over* [%]	*Time* [s]	*Ess* [rad]	*f*
1	10	10.73	0.714	0.0058	0.9739
2	10	10.24	0.444	0.0011	0.9854



Slika 4.6: Grafična upodobitev drugega testiranja (P_1 = 10 %, P_2 = 10 %)

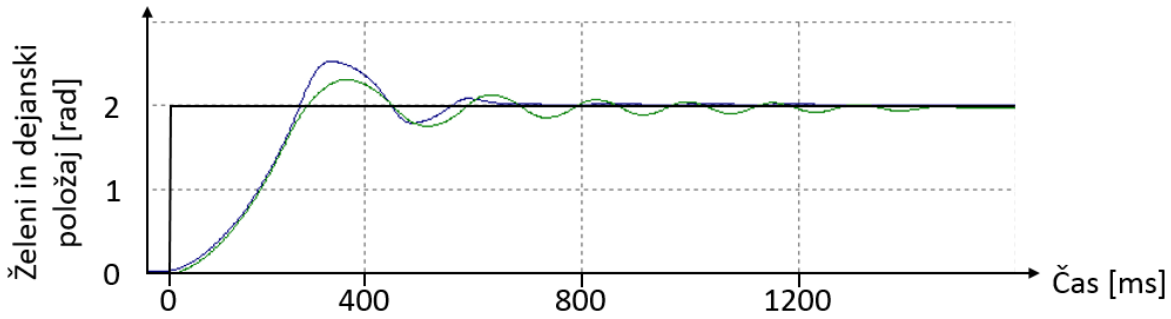
Druga optimizacija je bila kakovostneje opravljena kot prva. Prenihaja sta bila dokaj natančno načrtana, nastavitveni čas je padel na minimalno vrednost, tudi statični pogrešek najboljše rešitve je od množice izstopal kot kakovostnejši.

4.2.3 Testiranje 3

Zadnje testiranje s prenihajema P_1 = 15 %, P_2 = 25 % predstavlja testiranje pri kombiniranih prenihajih. Izsledki ter obnašanje robota je predstavljeno v tabeli 4.9 ter sliki 4.7.

Tabela 4.9: Tabelarična upodobitev tretjega testiranja

os	*P* [%]	*Over* [%]	*Time* [s]	*Ess* [rad]	*f*
1	15	16.72	1.222	0.0020	0.9784
2	25	25.87	0.606	0.0003	0.9553



Slika 4.7: Grafična upodobitev tretjega testiranja (P_1 = 15 %, P_2 = 25 %)

Pri sklepnem testiranju se pojavlja podobna težava kot v prvem testiranju. Pojavlja se dolg nastavitveni čas, medtem ko sta prenihaja dokaj natančno načrtana. Statični pogrešek je za obe osi skoraj popolnoma odpravljen.

5 DISKUSIJA

Z našimi eksperimenti smo potrdili osnovni značilnosti algoritma BA. To sta enostavna implementacija na realni sistem ter hitra konvergenca. Slednjo lastnost smo primerjali z znanim algoritmom GA ter na podlagi simulacijskih rezultatov obeh algoritmov ugotovili izkazovanje kakovostnejšega delovanje algoritma BA. To pomeni morebiti pridobitev enakovrednih (podobnih) rešitev v krajšem času ali boljših rezultatov v enakem času. Vendar algoritem BA ob danem enakem času vedno ne izkazuje kakovostnejših rešitev, kar pripisujemo stohastičnosti.

Najpomembnejša za nas je bila optimizacija parametrov položajnega regulatorja na realni aplikaciji. Rezultati potrjujejo, da so algoritmi po vzoru iz narave primerni za takšna opravila. Tekom eksperimentalnega dela smo ugotovili, da se je med testiranji pojavljal dolg nastavitveni čas. Z zmanjšanjem tega bi občutno pripomogli k izboljšanju vrednosti ocenjevalne funkcije. Dodatno prednost k zmanjšanju slednjega bi predstavljal dodatni člen ocenjevalne funkcije, ki bi ocenjeval, recimo, valovitost. Nekoliko enostavneje bi problem rešili s stalnim nastavljanjem krmilnih parametrov algoritma BA za vsak režim delovanja posebej.

Za še učinkovitejšo optimizacijo bi želeli uvesti večkriterijsko funkcijo, npr. za vsako os posebej (prvo in drugo), oz. za vsak izhodni parameter posebej (vrednost prenihaja, nastavitveni čas ter statični pogrešek). Težava, ki se ob tem pojavlja je, da morajo biti medsebojni členi večkriterijske funkcije neodvisni, kar v našem primeru ne velja popolnoma.

Končno bi lahko rezultate optimizacije izboljšali s hibridizacijo algoritma BA, npr. z uporabo strategij diferencialne evolucije, kombinacija katerih se je v preteklosti že izkazala kot uspešna. Dodatno rešitev predstavlja tudi samoadaptacija krmilnih parametrov, ki povečuje raznolikost populacije in s tem preprečuje algoritmu prehitro konvergenco proti morebitnim lokalnim optimumom.

6 SKLEP

Na začetku raziskovalnega dela smo si zadali problem, tj. načrtati optimalne parametre položajnega regulatorja dvoosnega robota za različne režime delovanja. Kot gonilo optimizacije smo izbrali algoritme po vzoru iz narave. Naloga je bila delno rešena z uspešno optimizacijo algoritma GA na simulacijskem modelu dvoosnega robota, ki je služil kot približek realne aplikacije za začetna testiranja posameznih algoritmov. Naša glavna iztočnica raziskovalnega dela je bila pridobiti kvalitetne rezultate v čim krajšem času. Rezultate optimizacijskega postopka predstavlja vektor, ki za vsako os (prvo in drugo) vsebuje konstante, oz. parametre posameznih členov (P in I člen). Ugotovili smo, da lahko realna aplikacija služi kot pripomoček za testiranje optimizacijskih algoritmov, zato smo na simulacijski model ter realno aplikacijo implementirali tudi enega izmed algoritmov roja delcev, t.i. algoritem BA. Pridobljeni simulacijski rezultati so se nagibali v korist algoritmu BA, vendar smo s simulacijo dokazali tudi nekaj slabosti algoritma BA. V določenih primerih je namreč zanesljivejše delovanje izkazoval algoritem GA. Nadaljnja analiza algoritmov je bila sestavljena iz testiranj realne aplikacije. Pridobili smo želene rezultate algoritma BA, medtem ko za algoritem GA testiranja zaradi mehanske okvare nismo opravili. Boljšega delovanja algoritma BA zato v celoti ne moremo potrditi. Smo pa z aplikacijo dosegli želeno – kvalitetne rezultate v čim krajšem času. Ob tem bi radi dodali, da bomo v nadalje spremenili zasnovo dvoosnega robota. Namesto zobatih reduktorjev bomo namestili jermenske, s čimer se bomo izognili največji težavi, ki nas je med testiranji pestila – izpadanjem zobov na zobniških gonilih prve osi. Poleg tega želimo v nadaljnje na realni aplikaciji preizkusiti ter primerjati dodatna dva algoritma: algoritem z roji delcev (PSO) ter strategije diferencialne evolucije (DE).

SEZNAM UPORABLJENIH VIROV

[1] Amon Djossou Adeyemi. *A Modified Bat Algorithm for Power Loss Reduction in Electrical Distribution System*. TELKOMNIKA Indonesian Journal of Electrical Engineering, 2015, vol. 14, no. 1, str. 55-61.

[2] Bäck Thomas, David B. Fogel, Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[3] Bäck Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[4] Bremermann Hans J. *Optimization through evolution and recombination*. Self-organizing systems 93, 1962.

[5] Cai Xingjuan, Wang Lei, Kang Qi, Wu Qidi. *Adaptive bat algorithm for coverage of wireless sensor network*. International Journal of Wireless and Mobile Computing, 2015, vol. 8, no. 3, str. 271-276.

[6] Čas Jure. *Izdelava zveznega nevronskega sliding-mode regulatorja za teleoperiranje SCARA robota*, Diplomsko delo. Maribor: Univerza v Mariboru, 2006.

[7] Čurkovič Milan. *Vgrajeni sistemi DSP/FPGA v sistemih vodenja*, Magistrsko delo. Maribor : Univerza v Mariboru, 2010.

[8] Dastani Mohammadreza, Mohammad Ardebili. *Optimal design and analysis simulation of an outer rotor hybrid excited generator for wind energy conversion systems.* IEEE, Electrical Engineering (ICEE), 23rd Iranian Conference on. IEEE, 2015.

[9] Dobnik-Dubrovski Polona, Brezočnik Miran. *Inženirsko načrtovanje tekstilij*. Maribor, Univerza v Mariboru, Oddelek za tekstilne materiale in oblikovanje, 2012.

[10] Dorigo, Marco, Mauro Birattari. *Ant colony optimization*. V : Encyclopedia of machine learning, Springer US, 2010, str. 36-39.

[11] Eiben Agoston E., Smith James E. *Introduction to evolutionary computing*. Natural Computing Series. Springer. Amsterdam, 2003.

[12] Enache Adriana-Cristina, & Sgârciu Valentin. *An Improved Bat Algorithm Driven by Support Vector Machines for Intrusion Detection*. International Joint Conference, Springer International Publishing, 2015, str. 41-51.

[13] Feng Lin, Yuxi Wang, Xiaoguang Qu. *The small UAV longitudinal control law design based on genetic algorithms*, 2015.

[14] Fister Dušan. *Optimizacija parametrov regulatorja z algoritmom na osnovi obnašanja netopirjev*, Elektrotehniška in računalniška konferenca ERK 2014, Portorož, 2014.

[15] Fister Iztok, Rauter Samo, Yang Xin-She, Ljubič Karin. *Planning the sports training sessions with the bat algorithm*. Neurocomputing, 2015, vol. 149, str. 993-1002.

[16] Fister Jr, Iztok, Dušan Fister, Iztok Fister. *A comprehensive review of cuckoo search: variants and hybrids*. International Journal of Mathematical Modelling and Numerical Optimisation, 2013, vol. 4, no. 4, str. 387-409.

[17] Fister Jr. Iztok, Dušan Fister, Xin-She Yang. *A Hybrid Bat Algorithm*. Elektrotehniški vestnik. Ljubljana : Elektrotehniška zveza Slovenije, 2013, vol. 80, no. 1-2, str. 1-7.

[18] Fister Jr. Iztok, Fister Dušan, Fister Iztok. *Differential evolution strategies with random forest regression in the bat algorithm*. V : Proceedings of the 15th annual conference companion on Genetic and evolutionary computation, ACM, 2013, str. 1703-1706.

[19] Fister Jr. Iztok, Fister Dušan, Mlakar Uroš, Ljubič Karin, Brest Janez, Fister Iztok: Netopirji, kresnice in kukavice pri obdelavi slik. Rosus, 2014, str. 28-33.

[20] Fister Jr. Iztok, Fong Simon, Brest Janez, Fister Iztok. *A novel hybrid self-adaptive bat algorithm*. V : The Scientific World Journal, *2014*.

[21] Fister Jr. Iztok, Fong Simon, Brest Janez, Iztok Fister. *Towards the self-adaptation in the bat algorithm*. V : Proceedings of the 13th IASTED international conference on artificial intelligence and applications, 2013.

[22] Fister Jr. Iztok, Xin-She Yang, Dušan Fister, Iztok Fister. *Cuckoo search: a brief literature review*. V : Cuckoo search and firefly algorithm, Springer International Publishing, 2014, str. 49-62.

[23] Fister Jr. Iztok, Yang Xin-She, Fister Iztok, Brest Janez, Fister Dušan. *A Brief Review of Nature-Inspired Algorithms for Optimization.* Elektrotehniški vestnik. Ljubljana : Elektrotehniška zveza Slovenije, 2013, vol. 80, no. 3, str. 116-122.

[24] Fister Jr. Iztok, Yang Xin-She, Ljubič Karin, Fister Dušan, Brest Janez, Fister Iztok. *Towards the novel reasoning among particles in pso by the use of rdf and sparql*. The Scientific World Journal, 2014.

[25] Gandomi Amir Hossein, Yang Xin-She, Alavi Amir H., Talatahari Siamak. *Bat algorithm for constrained optimization tasks*. V : Neural Computing and Applications, 2013, str. 1239-1255.

[26] Goldberg David E., John H. Holland. *Genetic algorithms and machine learning*. Machine learning, 1988, vol. 3, no. 2, str. 95-99.

[27] Huang Hui Xian, Zeng Sha, Zhao Wei Juan, Chen Ren. *Optimization of ramjet fuel control system based on GA-PSO*. V : Information, Computer and Application Engineering: Proceedings of the International Conference on Information Technology and Computer Application Engineering (ITCAE 2014). Hong Kong, China, 2015, vol. 10, no. 11, str. 89.

[28] Ishikawa Takeo, Peijie Xie, Nobuyuki Kurita. *Topology Optimization of Rotor Structure in Permanent Magnet Synchronous Motors Considering Ease of Manufacturing*. IEEJ Journal of Industry Applications, 2015, vol. 4, no. 4, str. 469-475.

[29] Jagarinec Albin. *Adaptivni regulator z mehko logiko za dvoosni SCARA mehanizem*, Diplomsko delo. Maribor : Univerza v Mariboru, 2005.

[30] Kanarachos Stratis, Kanarachos Andreas. *Intelligent road adaptive suspension system design using an experts' based hybrid genetic algorithm*. Expert Systems with Applications, 2015.

[31] Kennedy James, Russell Eberhart. *Particle swarm optimization*. IEEE International Conference on, 1995, vol. 4.

[32] Kolar Marko. *Vodenje SCARA robota z mehko logiko*, Diplomsko delo. Maribor : Univerza v Mariboru, 2005.

[33] Koza, John R. *Genetic programming: on the programming of computers by means of natural selection*, MIT press, 1992, vol. 1.

[34] Luo Yin, Yuan Shouqi, Sun Hui, Guo Yihang. *Energy-saving control model of inverter for centrifugal pump systems*. Advances in Mechanical Engineering, 2015, vol. 7, no. 7.

[35] Nakamura Rodrigo Yuji, Pereira Luis A., Costa Kelton, Rodrigues, D., Papa Joau Paulo, Yang, Xin-She. *A binary bat algorithm for feature selection*. V : Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on IEEE, 2012, str. 291-297.

[36] Premkumar Kamaraj, Manikandan B. V. *Speed control of Brushless DC motor using bat algorithm optimized Adaptive Neuro-Fuzzy Inference System*. Applied Soft Computing, 2015, vol. 32, str. 403-419.

[37] Rechenberg Ingo. *Evolution strategy: Nature's way of optimization*. V : Optimization: Methods and applications, possibilities and limitations. Springer Berlin Heidelberg, 1989, str. 106-126.

[38] Schwefel Hans-Paul Paul. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[39] Slanič Tomaž. *Genetski regulator za dvoosnega SCARA robota*, Diplomsko delo. Maribor, Univerza v Mariboru, 2006.

[40] Storn Rainer, Kenneth Price. *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*. V : Journal of global optimization, 1997, vol. 11, no. 4, str. 341-359.

[41] Šafarič Riko, Rojko Andreja: *Inteligentne regulacijske tehnike v mehatroniki*. Maribor: Univerza v Mariboru, 2005.

[42] Wang Y. L., J. X. Wang, H. B. Kang. *Application of the genetic algorithm in image processing*. V : Information, Computer and Application Engineering: Proceedings of the International Conference on Information Technology and Computer Application Engineering (ITCAE 2014), Hong Kong, China, 2015, vol. 10, no. 11.

[43] Wikipedia : SCARA. Wikipedia, The Free Encyclopedia. Dostopno na WWW: http://en.wikipedia.org/wiki/SCARA/ [7.7.2015].

[44] Xin-She Yang. *Bat algorithm for multi-objective optimisation*. International Journal of Bio-Inspired Computation. Inderscience, 2011, vol. 3, no. 5, str. 267 – 274.

[45] Yang Xin-She, Amir Hossein Gandomi. *Bat algorithm: a novel approach for global engineering optimization*. Engineering Computations, 2012, vol. 29, no. 5, str. 464-483.

[46] Yang Xin-She. *A new metaheuristic bat-inspired algorithm*. V : Gonzales J. R., et. al. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Berlin : Springer Link, 2010, str. 65-74.

[47] Ye Zhi-Wei, Wang Ming Wei, Liu Wei, Chen Shao Bin. *Fuzzy entropy based optimal thresholding using bat algorithm*. Applied Soft Computing, 2015, vol. 31, str. 381-395.

[48] Zhao Dongsheng, Yuzhu He. *Chaotic binary bat algorithm for analog test point selection*. Analog Integrated Circuits and Signal Processing, 2015, str. 1-14.

Življenjepis

Ime in priimek: Dušan Fister

Rojen: 18.05.1993

OŠ: Bakovci

SŠ: Gimnazija Franca Miklošiča Ljutomer

Fakulteta: UM, FS, Program: Mehatronika.

Priloga (BA algoritem):

```
int i, j;
double c, u, w, q;
double rndm;


/*--------------------------- INICIALIZACIJA ---------------------------*/

if(xD[0]<1) {
    bpf=0;
    ipf=0;
    srand(2);

    Lb[0] = 0;      // Kp1 - lower bound
    Lb[1] = 0;      // Kv1 - lower bound
    Lb[2] = 0;      // Kp2 - lower bound
    Lb[3] = 0;      // Kv2 - lower bound
    Ub[0] = 600;    // Kp1 - upper bound
    Ub[1] = 50;     // Kv1 - upper bound
    Ub[2] = 600;    // Kp2 - upper bound
    Ub[3] = 50;     // Kv2 - upper bound

    for(i=0;i<Pop;i++) {

        for(j=0;j < d;j++) {
            v[i][j]=0;
            rndm=((double)rand()/((double)(RAND_MAX)+(double)(1)));
            s[i][j]=(Ub[j]-Lb[j])*rndm+Lb[j];
        }
        fitness1[i]=0;
        fitness1[i]=0;
    }

    f_min1=0;
    f_min2=0;

    for(i=0;i<d;i++)    {
        best[i] = s[0][i];    // declare the best individual
    }

    ff1 = 0;
    ff2 = 0;

    xD[0]=1;    // Init marker
    xD[1]=-1;   // pop
}
//konec inicializacije

/*+++++++++++++++++++++++++++++++ FITNES +++++++++++++++++++++++++++++++*/
xD[40]=0;
xD[34]=0;               /* zmanjšanje navora ob prevelikem prenihaju za obe osi-Eliminator*/

if(I1[1]>I2[11])
    xD[34]=1;

if(I1[0]>I2[13])
    xD[40]=1;

if(xD[1]>=0) {
    if(xD[1]<I2[15]) {
        if(xD[3]<I2[16]) {
            i=xD[1];
            if((xD[34]==0)||(xD[40==0]))  {
                /* posiljanje parametrov na izhod */
                xD[5]=s[i][0];
                xD[6]=s[i][1];
                xD[7]=s[i][2];
                xD[8]=s[i][3];

                if(I3[0] != xD[33]) {
                /* izracun casov zakasnitve prve in druge osi */
                Time11=1-((*Tsim)-I1[2])/(*Tsim);
                Time22=1-((*Tsim)-I1[3])/(*Tsim);
```

```
                /* absolutno */
                xD[15]=I2[14]-I1[0];

                if(xD[15]<0)
                    xD[15]=-xD[15];
                xD[16]=I2[12]-I1[1];

                if(xD[16]<0)
                    xD[16]=-xD[16];

                /* fitnes funkcija za prvo in drugo os*/
                ff1=I2[0]*(1-xD[15])+I2[1]*(1-Time11)+I2[2]*(1-I1[4]);
                if(ff1 >= 1)
                    ff1=0.1;
                else
                    xD[32]=ff1;

                ff2=I2[3]*(1-xD[16])+I2[4]*(1-Time22)+I2[5]*(1-I1[5]); //I2[3] - E1B
                if(ff2 >= 1)
                    ff2=0.1;
                else
                    xD[21]=ff2; // izpis pod fitness 2

                xD[30]=(ff1+ff2)/2;


                rndm=((double)rand()/((double)(RAND_MAX)+(double)(1)));

                if((ff1 >= fitness1[i] && rndm > A) || fitness1[i] == 0) {
                        pop[i][0] = s[i][0];
                        pop[i][1] = s[i][1];
                        fitness1[i] = ff1;
                    }
                if((ff2 >= fitness2[i] && rndm > A) || fitness2[i] == 0) {
                        pop[i][2] = s[i][2];
                        pop[i][3] = s[i][3];
                        fitness2[i] = ff2;
                    }
                    // Update the current best solution
                if (ff1 > f_min1)  {

                        best[0] = s[i][0];
                        best[1] = s[i][1];
                        f_min1 = ff1;

                        //parametri na izhod
                        xD[9] = best[0];    // najboljsi Kp1
                        xD[10] = best[1];   // najboljsi Kv1
                    }

                if(ff2 > f_min2)  {
                        best[2] = s[i][2];
                        best[3] = s[i][3];
                        f_min2 = ff2;

                        xD[11] = best[2];   // najboljsi Kp2
                        xD[12] = best[3];   // najboljsi Kv2
                    }
                }
            }
        }

else {

    if(u0[0] != xD[2])
    {
        i=xD[13];
        /* izracun povprecnega fitnesa*/
        pf[i][0]=0;
        /* postavitev blokade na 0 */
        xD[34]=0;
        xD[40]=0;
            }
        }
    }
}
```

```
/* po koncu 10 sekundnega intervala skoci na naslednji set parametrov */
if(u0[0] != xD[2])
    xD[1]++;
if(xD[1]==I2[15])
{
    xD[3]++;
    xD[1]=0;
}
/*+++++++++++++++++++++++++++++++ NEWGEN ++++++++++++++++++++++++++++++++*/

if(xD[3] != xD[4]){
    if(xD[3]<I2[16]){ //*ITER    {
        ipf=xD[31];
        xD[18]=0;//apf=0

        for(i=0;i<Pop;i++) {
            rndm=((double)rand()/((double)(RAND_MAX)+(double)(1)));
            Q[i]=Qmin+(Qmax-Qmin)*rndm;
            for(j=0;j<d;j++) {
                v[i][j] = v[i][j]+(pop[i][j]-best[j])*Q[i]; // v(i,:)=v(i,:)+(Sol(i,:)-
best)*Q(i);
                s[i][j] = fabs(pop[i][j]+v[i][j]);

                // Apply simple bounds/limits
                if(s[i][j] < Lb[j])
                    s[i][j] = Lb[j];
                if(s[i][j] > Ub[j])
                    s[i][j] = Ub[j];
            }

            rndm = ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) );
            // Pulse rate - RWDE step
            if(rndm < r) {
                // The factor 0.001 limits the step sizes of random walks
                for(j=0;j<d;j++) {
                    // Sample normal
                    do {
                        u = ((double) rand() / (RAND_MAX)) * 10 - 1;
                        w = ((double) rand() / (RAND_MAX)) * 10 - 1;
                        q = u * u + w * w;
                    } while (q == 0 || q > 1);
                    c = sqrt(-2 * log(q) / q);
                    s[i][j] = best[j]+1*(u*c);
                    // Apply simple bounds/limits
                    if(s[i][j] < Lb[j])
                        s[i][j] = Lb[j];
                    if(s[i][j] > Ub[j])
                        s[i][j] = Ub[j];
                }
            }
        } //end for
    }
}

/* Zaustavitev simulacije ob doseženem številu iteracij*/
if(xD[3]==I2[16]){

   if(u0[0] != xD[2]) xD[17]++;
   if(xD[17]==2) xD[14]=1;
}

/* belezimo prejsnji stanji */
xD[2]=u0[0];
xD[4]=xD[3];
xD[33]=I3[0];
```

UNIVERZA V MARIBORU

Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek diplomanta-tke: Dušan Fister

Vpisna številka: _____

Študijski program: 1. STOPNJA MEHATRONIKA

Naslov diplomskega dela: Načrtovanje samonastavljivega regulatorja 2 DOF robota s pomočjo BA algoritma

Mentor: Miran Brezočnik, Riko Šafarič

Somentor: Iztok Fister

Podpisani-a Dušan Fister izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Diplomsko delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija diplomskega dela je istovetna elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, varstva industrijske lastnine ali tajnosti podatkov naročnika: _____
ne sme biti javno dostopno do _____ (datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela).

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah UM.

Datum in kraj: Maribor, 26.08.2015

Podpis diplomanta-tke: 

Podpis mentorja _____
(samo v primeru, če delo ne sme biti javno dostopno):

Podpis odgovorne osebe naročnika in žig: _____
(samo v primeru, če delo ne sme biti javno dostopno)