Optimizacija parametara vektorske regulacije korištenjem HIL testing metodologije

Nedić, Adriana

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij elektrotehnike

Optimizacija parametara vektorske regulacije korištenjem HIL testing metodologije

Diplomski rad

Adriana Nedić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 14.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Adriana Nedić
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika
Mat. br. Pristupnika, godina upisa:	D-1417, 07.10.2021.
OIB studenta:	96608390935
Mentor:	prof. dr. sc. Marinko Barukčić
Sumentor:	dr. sc. Tin Benšić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Željko Hederić
Član Povjerenstva 1:	dr. sc. Tin Benšić
Član Povjerenstva 2:	izv. prof. dr. sc. Emmanuel-Karlo Nyarko
Naslov diplomskog rada:	Optimizacija parametara vektorske regulacije korištenjem HIL testing metodologije
Znanstvena grana diplomskog rada:	**Elektrostrojarstvo (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Potrebno je izvesti matematičke modele za sintezu regulatora sinkronog stroja s permanentnim magnetima. Sintetizirati regulacijski krug za vektorsko upravljanje PMSM na dSpace platformi, izvesti HIL simulaciju motora na OPAL-RT platformi te u kosimulacijskom okruženju optimirati rad u različitim pogonskim točkama sustava. Rezervirano za: Adriana Nedić Sumentor s FERITa: dr.sc.Tin Benšić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	14.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Adriana Nedić
Studij:	Diplomski sveučilišni studij Elektrotehnika
Mat. br. studenta, godina upisa:	D-1417, 07.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Optimizacija parametara vektorske regulacije korištenjem HIL testing metodologije**

izrađen pod vodstvom mentora prof. dr. sc. Marinko Barukčić

i sumentora dr. sc. Tin Benšić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. Uvod **1**

1.1. Pregled područja teme 2

2. Sinkroni motor s permanentnim magnetima **4**

2.1. Matematički model sinkronog motora s permanentnim magnetima . . . 4

2.2. Troosni model sinkronog motora 5

2.2.1. Dvoosna teorija PMSM-a 6

2.2.2. Dvoosni model sinkronog motora 7

2.3. Vektorska regulacija 8

2.3.1. Regulator struje 8

2.3.2. Regulator brzine 11

3. HIL testing metodologija **13**

3.1. RCP 14

3.1.1. dSpace MicroLabBox 15

3.1.2. Modificiranje modela regulatora u Simulink-u 16

3.1.3. Automatizacija ControlDesk-a 19

3.2. HIL 22

3.2.1. OPAL-RT stvarnovremenski sustv 22

3.2.2. Modificiranje modela PMSM-a u Simulnik-u 23

3.2.3. RT-LAB 24

4. Optimizacija parametara vektorske regulacije **26**

4.1. Jednociljni optimizacijski problem 26

4.2. Višeciljni optimizacijski problem 27

4.2.1. Pareto optimum 27

4.3. MIDACO optimizator 28

4.3.1. Metaheuristički algoritam kolonije mrava. 29

4.3.2. Postavke MIDACO optimizatora za rješavanje optimizacije parametara vektorske regulacije 30

5. Rezultati optimizacijskog procesa **34**

5.1. Analiza utjecaja parametara vektorske regulacije na dinamička stanja motora 37

6. Zaključak **41**

Literatura. **42**

Sažetak **44**

Abstract **45**

1. UVOD

Automobilska industrija u zadnjih nekoliko godina, s ciljem reduciranja emisije CO_2 i slijedeći zakone o očuvanju okoliša, sve se više posvećuje proizvodnji energetski učinkovitih električnih i hibridnih električnih vozila kojima nastoje zamijeniti tradicionalna vozila. Takva vozila posjeduju sinkrone motore sa permanentnim magnetima koji se preferiraju zbog visoke učinkovitosti i pouzdanosti. Nagli i ubrzani razvoj spomenutih vozila stavlja pred inženjere značajne izazove [1, 2].

Izgradnja takvih vozila podrazumijeva mnoge nove i kompleksne sustave koje treba razviti, integrirati i ispitati. Konvencionalne metode testiranja, koje uključuju dugo čekanje na izradu električnih motora i upravljačkih jedinica (eng. *electronic control unit - ECU*), čiji rezultati znaju biti nevaljani zbog loše dizajniranih sustava, u posljednje vrijeme zamjenjuje testiranja u virtualnom svijetu na modelima [3]. HIL (*Hardver In Loop*), hardver u petlji, jedan je od procesa testiranja koji se pokazao kao industrijski standard za testiranje i odobravanje sustava. Osim njega, RCP (*Rapid Control Prototyping*) metoda omogućuje brzo razvijanje i testiranje kontrolnih algoritama u stvarnom vremenu prije nego što se ti algoritmi integriraju u stvarna vozila. Dodatno, u automobilskoj industriji precizna regulacija elektromotora električnih vozila ključna je za napredne sustave upravljanja. Kako bi se prethodno spomenuti zahtjevi nad preciznošću ispunili potrebno je odabrati odgovarajuće parametre regulatora. Sve je češća upotreba mateheurističkih optimizacijskih algoritama za precizno procjenjivanje tih parametara. Metaheuristički algoritmi su sposobni pretraživati velike i kompleksne prostore rješenja te pronaći optimalne vrijednosti parametara koje poboljšavaju učinkovitost i performanse elektromotora. Ovaj pristup razvijanja i testiranja rezultira smanjenjem vremena razvoja, nižim troškovima i visokokvalitetnim proizvodima.

Tema ovog rada je optimizacija parametara vektorske regulacije korištenjem HIL testing metodologije. Izvedeni simulacijski modeli motora i pripadajućeg regulatora za vektorsko upravljanje implementirani u stvarnovremenske simulatore povezani su u kosimulaciju u HIL sustav s ciljem testiranja i optimizacije parametara regulatora.

U drugom poglavlju je dan uvod u sinkrone strojeve gdje su pobliže objašnjeni sinkroni motori s permanentnim magnetima. Zatim je definiran troosni matematički model motora kao polazišni matematički model iz kojeg će nastati simulacijski model. Prethodno je dan kratak ogled *dq* transformacije primjenom koje se iz troosnog sustava prelazi u dvoosni *dq* sustav. Temeljem dvoosnog modela motora parametriran je regulator za vektorsko upravljanje. U konačnici su na osnovu spomenutih matematičkih modela napravljeni simulacijski modeli unutar Simulink-a.

Treće poglavlje sadrži detaljno objašnjene metode testiranja (HIL) i prototipiranja kontrolnih algoritama (RCP). Prikazan je postupak implementacije simulacijskih modela u stvarnovremenske simulatore dSpace MicroLabBox i OPAL-RT.

U četvrtom poglavlju objašnjen je pojam optimizacije kao i vrste optimizacijskih problema. Definirani su jednociljni i višeciljni optimizacijski problemi u teoriji kao i konkretni optimizacijski problemi koji se rješavaju u ovom radu. Način rada optimizatora kao i mateheurističkog algoritma na kojem temelji svoj rad su objašnjeni. Prikazani su odabrani parametra optimizatora.

Peto poglavlja sadrži analizu dobivenih rezultata optimizacije. Prije same analize dan je detaljniji opis izvođenja eksperimenta. Usporedbom svih dobivenih opcija te služeći se teorijskim osnovama dinamike stroja analizirani su ishodi svih promatranih slučaja.

1.1. Pregled područja teme

Kombinacija HIL testiranja hibridnih električnih vozila te istovremena uporaba RCP metode za razvijanje kontrolnih algoritama prezentirana je u radu [1]. Hibridna električna vozila mogu imati dva ili više pogona čime sustav upravljanja postaje mnogo složeniji u odnosu na tradicionalna vozila. Time je razvoj i testiranje algoritama upravljanja mnogo izazovnije u razvojnoj fazi. Zbog složenosti, ti sustavi moraju proći kroz opsežno testiranje prije nego što se instaliraju u prototipno vozilo za testiranje na cestama. Radom u virtualnom okruženju moguće je simulirati uvjete ispitivanja i ponašanje vozila, čime se razvojnim inženjerima pruža metoda za provjeru strategije upravljanja bez potrebe za stvarnim vozilom ili stvarnim komponentama vozila u mnogim slučajevima.

RCP se inače koristi prije HIL testiranja dok ovaj rad prikazuje istovremeno djelovanje s HIL testiranjem kako bi se razvijali kontrolni algoritmi kada stvarni proizvodni ECU nije dostupan. U ovom radu istaknute su brojne prednosti korištenja RCP regulatora. Primjerice, posjeduje korisničko sučelje preko kojeg se može upravljati parametrima regulatora, mijenjati model i sve to za vrijeme testiranja. Za razliku od stvarnih regulatora, simulatori na kojima je implementiran model regulatora, imaju veću memoriju, više I/O jedinica i brži procesor što osigurava izvršenje svih zadataka. Simulatori odnosno RCP hardveri su sposobni proizvoditi signale za upravljanje različitim vrstama električnih motora (asinkronim motorima, sinkronim motorima s permanentnim magnetima) i ostalim komponentama hibridnog pogonskog sustava. Budući da je cilj RCP da omogući razvijanje što boljih regulatora pristupa se optimizacijskim metodama za pronalazak optimalnih parametara regulatora. Rad [4] prikazuje načine parametriranja regulatora, konkretno PID regulatora, koristeći različite metaheurističke algoritme. Zbog svoje jednostavne implementacije, dobrih rezultata te rješavanja kompleksnih problema metaheuristički algoritmi pokazali su se prikladnima za parametriranje regulatora.

Razvoj hibridnih vozila kompanija Ford Motor također temelji na HIL testiranju gdje je naglasak stavljen na testiranje elektromotornih pogona. U [2] su istaknuti izazovi s kojima se susrećem prilikom izvođenja HIL simulacija elektromotornih pogona. Najčešće korišteni pogon sastoji se od sinkronog motora s permanentnim magnetima i IGBT pretvarača. Zbog visokih frekvencija pretvarača simulacije imaju vrlo malo vrijeme uzorkovanja što može predstaviti problem za simulator koji to vrijeme ne može postići. Za rješenje spomenutih problema pristupa se TSAM (*Time-Step Averaging Method*) metodi ili se primjenjuju simulatori sa FPGA (eng. *Field programible gate array*) čipovima koji mogu postići kratko vrijeme uzorkovanja. Još jedan važan aspekt HIL simulacije, koji ističu u ovom radu, je testiranje ponašanja sustava u nenormalnim scenarijima ili načinima upravljanja. Pa je tako provedeno nekoliko testova kako bi se provjerila učinkovitost kontrolne strategije, kao i neki osnovni zaštitni sustavi, poput zaštite od preopterećenja strujom. Tijekom testiranja koriste OPAL-RT simulatore, koji su u iste svrhe korišteni i u ovom diplomskom radu.

Osim napretka u pogledu prelaska na električna vozila, potpuno autonomna vozila također su jedan od ciljeva mnogih proizvođača automobila. ADAS (Advanced Driver Assistance Systems) sustavi primjenjuju se u vozilima kako bi se poboljšala sigurnost i udobnost vozača. Kako bi ovi sustavi temeljeni na viziji bili ostvarivi moraju biti ispravno testirani, što znači da mora biti provedeno mnogo testova sa scenarijima iz stvarnog života. Različiti scenariji vožnje, s različitim uvjetima, moraju se uzeti u obzir. Svi ovi testovi su vremenski zahtjevni i skupi kako bi se izveli u stvarnom prometu. U radu [5] je predstavljeno potpuno rješenje za uređaj za snimanje videozapisa u automobilima kroz HIL simulaciju. Ovakav uređaj može značajno pojednostaviti i smanjiti vrijeme razvoja algoritama koji su potrebni za autonomnu vožnju.

U početku razvoja ADAS sustavi su bili usmjereni na udobnost vozača dok su posljednjih godina usmjerili istraživanja u pogledu sigurnosti. Rad [6] koristi HIL metodu za razvoj i ispitivanje sustava za sprječavanje sudara (eng. *pre-crash system - PCS*). U tu svrhu, testno

vozilo je stavljeno u simulirano okruženje prometa. Kako bi senzori testnog vozila primili podatke, jedan od sudionika u simuliranom prometu predstavljen je vozilom kojim se kroz različite scenarije ispituje reagiranje testnog vozila.

2. SINKRONI MOTOR S PERMANENTNIM MAGNETIMA

Sinkroni strojevi pripadaju rotacijskim električnim strojevima. Rotor sinkronog stroja okreće se sinkronom brzinom (ω_s) koja se definira kao kutna brzina kojom okretno magnetsko polje rotira u zračnom rasporu. Sinkrona brzina rotora određena je frekvencijom napajanja (f) te brojem pari polova stroja (p) [7].

$$\omega_s = \frac{2\pi f}{p} \tag{2-1}$$

S obzirom na primjenu sinkronih strojeva uvelike dominiraju sinkroni generatori koji proizvode gotovo cjelokupnu energiju koja se troši u svijetu [8]. Sinkroni motori nalaze primjenu u području velikih snaga (crpke, kompresori) ili izrazito malih snaga primjerice kod računala, u robotici i sl. Podjela sinkronih motora može se izvršiti prema više kriterija, pa se tako s obzirom na vrstu uzbude koja se nalazi na rotoru stroja dijele na [9, 10]:

1. Sinkroni motori s vanjskom uzbudom
2. Sinkroni motori s permanentnim magnetima
3. Reluktantni motori
4. Histerezni motori

U ovom radu naglasak je stavljen na sinkrone motore s permanentnim magnetima (eng. *Permanent Magnet Synchronous Motor - PMSM*). Navedeni motori nailaze na široku primjenu zbog prednosti koje pružaju. Neke od tih prednosti su precizno upravljanje, robusnost, energetska učinkovitost te korisnost [11]. Konstrukcijski su izvedeni kao trofazni sinkroni strojevi, osim što je uzbudni namot na rotoru, zajedno sa četkicama i kliznim kolutom, zamijenjen permanentnim magnetima čime se eliminiraju gubitci u bakru rotora, smanjuje masa stroja i samim tim povećava učinkovitost[12]. Ovisno o smještaju magneta na rotoru mogu se podijeliti u tri skupine [13]:

1. Sinkroni motori s vanjskim permanentnim magnetima (eng. *Surface mounted PMSM*)
2. Sinkroni motori s unutarnjim permanentnim magnetima (eng. *Interior PMSM*)
3. Sinkroni motori čiji se magnet nalazi unutar paketa limova rotora stroja

2.1. Matematički model sinkronog motora s permanentnim magnetima

Kao što je spomenuto u uvodu za potrebe raznih testiranja upotrebljavaju se modeli. Matematički model je skup matematičkih relacija (jednadžbi, nejednadžbi, graničnih uvjeta...) definiran unutar nekog interpretiranog formalnog sustava. Između varijabli matematičkog modela i varijabli originala moraju postojati pravila korespondencije. U protivnom, matematički model je bez svrhe jer se njegova valjanost ne može eksperimentalno provjeriti na originalu [14]. Zato se prilikom modeliranja PMSM-a u obzir uzimaju sljedeće pretpostavke:

1. Trofazni motor u zvijezda (Y) spoju
2. Statorski namoti simetrični i sinusoidalno raspodijeljeni duž zračnog raspora

3. Indukcija o položaju rotora ovisi sinusoidalno
4. Zanemaruje se magnetsko zasićenje, magnetska histereza i promjena parametara
5. U početnim (subtranzijentnim) i prijelaznim (tranzijentnim) stanjima brzina rotora je blizu sinkrone

Slijedeći navedene pretpostavke u idućem potpoglavlju izveden je troosni matematički model PMSM-a kao polazišna točka za daljnje modeliranje motora koji je korišten u radu. Sve relacije temelje se na knjigama [13, 15].

2.2. Troosni model sinkronog motora

Na osnovu nadomjesne sheme sinkronog motora izvedene su jednadžbe koje opisuju model istog. Pripadajuće statorske veličine označene su indeksom "s" dok su pripadajuće rotorske označene slovom "r".

Naponska jednadžba statora glasi:

$$\mathbf{u}_{abc} = \mathbf{R}_s \mathbf{i}_{abc} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_{abc} \tag{2-2}$$

gdje je: $\mathbf{u}_{abc} = \begin{bmatrix} u_a & u_b & u_c \end{bmatrix}^T$ napon statora, $\mathbf{i}_{abc} = \begin{bmatrix} i_a & i_b & i_c \end{bmatrix}^T$ struja statora, $\mathbf{R}_s$ matrica otpora (dimenzija 3×3) čiji elementi na glavnoj dijagonali imaju iznos otpora statora R_s koji je jednak u sve tri faze, a $\boldsymbol{\psi}_{abc} = \begin{bmatrix} \psi_a & \psi_b & \psi_c \end{bmatrix}^T$ ulančani magnetski tok.

$$\boldsymbol{\psi}_{abc} = \mathbf{L}_s \mathbf{i}_{abc} + \boldsymbol{\psi}_M \tag{2-3}$$

Izraz (2-3) predstavlja ulančani magnetski tok nastao protjecanjem struje kroz namote statora te pod djelovanjem permanentnog magneta ($\boldsymbol{\psi}_M$) dok $\mathbf{L}_s$ predstavlja matricu induktiviteta [15]:

$$\mathbf{L}_s = \begin{bmatrix} L_{ls} + L_A + L_B\cos(2\theta_r) & -\frac{1}{2}L_A + L_B\cos 2(\theta_r - \frac{\pi}{3}) & -\frac{1}{2}L_A + L_B\cos 2(\theta_r + \frac{\pi}{3}) \\ -\frac{1}{2}L_A + L_B\cos 2(\theta_r - \frac{\pi}{3}) & L_{ls} + L_A + L_B\cos(2\theta_r - \frac{2\pi}{3}) & -\frac{1}{2}L_A + L_B\cos 2(\theta_r + \pi) \\ -\frac{1}{2}L_A + L_B\cos 2(\theta_r + \frac{\pi}{3}) & -\frac{1}{2}L_A + L_B\cos 2(\theta_r + \pi) & L_{ls} + L_A + L_B\cos(2\theta_r + \frac{2\pi}{3}) \end{bmatrix} \tag{2-4}$$

gdje je θ_r kut koji označava položaj rotora, L_{ls} rasipni induktivitet, a L_A i L_B su induktiviteti faza dvofaznog sinkronog stroja temeljem kojeg je modeliran troosni model [15].

Promjena položaja rotora (θ_r) u vremenu jednaka je brzini gibanja rotora, odakle slijedi:

$$\frac{\mathrm{d}\theta_r}{\mathrm{d}t} = \omega_r \tag{2-5}$$

Gibanjem rotora magnetski tok permanentnog magneta ulančava se s namotima faza statora:

$$\boldsymbol{\psi}_{Mabc} = \psi_m \begin{bmatrix} \cos(\theta_r) \\ \cos(\theta_r - \frac{2\pi}{3}) \\ \cos(\theta_r + \frac{2\pi}{3}) \end{bmatrix} \tag{2-6}$$

Množenjem jednadžbe (2-2) matricom $\mathbf{i}_{abc}^T$ s lijeve strane dobivena je jednadžba koja predstavlja bilancu snage:

$$\mathbf{i}_{abc}^T \mathbf{u}_{abc} = \mathbf{i}_{abc}^T R_s \mathbf{i}_{abc} + \mathbf{i}_{abc}^T \frac{\partial \psi_{abc}}{\partial t} + \mathbf{i}_{abc}^T \omega_r \frac{\partial \psi_{abc}}{\partial \theta_r} \tag{2-7}$$

koja govori da je trenutna snaga motora jednaka zbroju članova s desne strane jednadžbe koji redom predstavljaju gubitke u bakru, gubitke u željezu te mehaničku snagu motora (P_e). Iz izraza za mehaničku snagu izveden je elektromagnetski moment stroja:

$$M_e = \frac{P_e}{\omega_r} = \mathbf{i}_{abc}^T \frac{\partial \psi_{abc}}{\partial \theta_r} \tag{2-8}$$

Za potpun model stroja u obzir se mora uzeti mehanička jednadžba gibanja stroja uslijed razvijenog elektromagnetskog momenta, otpora trenja te momenta tereta.

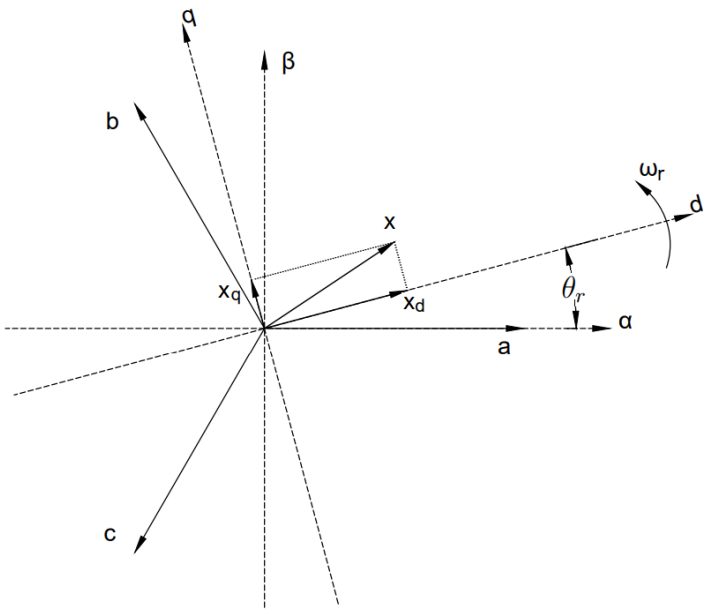
$$M_u = J\frac{\mathrm{d}\omega_r}{\mathrm{d}t} = M_e - B_{tr} - M_t \tag{2-9}$$

gdje je M_u moment ubrzanja, J inercija motora, B_{tr} otpor trenja te M_t moment tereta.

Analizom jednadžbi izvedenih u ovom potpoglavlju evidentno je da su varijable modela sinkronog motora ovisne o položaju rotora pa samim time i o vremenu čime je model nelinearan vremenski nepromjenjiv i kao takav nepogodan za modeliranje i simulaciju. U narednim potpoglavljima prikazani su alati kojima je omogućeno modeliranje sinkronog motora.

2.2.1 Dvoosna teorija PMSM-a

dq transformacija, poznatija kao Parkova transformacija, matematička je metoda koja se primjenjuje kod upravljanja elektromotorima posebice kod vektorskog upravljanja. Njenom implementacijom trofazne vremenski promjenjive varijable (napone, struje i tokove) zamjenjuju se varijablama dvoosnog sustava koje rotiraju kružnom frekvencijom okretnog magnetskog polja rotora [9]. Korak koji prethodni Parkovoj transformaciji jeste Clarkina transformacija u kojoj se iz troosnog sustava prvo prelazi u dvoosni mirujući $\alpha\beta$ sustav gdje se umjesto tri vremenski promjenjive veličine dobivaju dvije vremenski promjenjive veličine. Zatim se Parkovom transformacijom spomenuti mirujući dvoosni sustav transformira u rotirajući dvoosni. Kao krajnji rezultat dobiju se dvije istosmjerne veličine[15]. Slika 2.1 prikazuje prethodno spomenute transformacije [12].



SL. 2.1: *Transformacija troosnog abc sustava u dvoosni rotirajući dq sustav*

Svaka od spomenutih varijabli bilo to napona, struje ili toka je zapravo prostorni vektor. Isti taj prostorni vektor je definiran pomoću svojih komponenti stvarnih trofaznih iznosa koji su mjerljivi u namotima stroja, a vektor je definiran pomoću linearne kombinacije troosne baze vektorskog prostora. Tada je moguće svaki vektor prikazati kao linearnu kombinaciju jediničnih vektora ortonomirane baze te veličine [16]:

$$\begin{aligned}\overrightarrow{x} = \left[x_a\cos(\theta_r) + x_b\cos(\theta_r - \tfrac{2\pi}{3}) + x_c\cos(\theta_r + \tfrac{2\pi}{3})\right]\overrightarrow{d} + \\ \left[x_a\sin(\theta_r) + x_b\sin(\theta_r - \tfrac{2\pi}{3}) + x_c\sin(\theta_r + \tfrac{2\pi}{3})\right]\overrightarrow{q}\end{aligned} \tag{2-10}$$

$$\mathbf{x}_{dq} = \mathbf{K}\mathbf{x}_{abc} \tag{2-11}$$

gdje vrijedi da je $\mathbf{x}_{dq} = \begin{bmatrix}\mathbf{x}_d & \mathbf{x}_q\end{bmatrix}^T$, a $\mathbf{x}_{abc} = \begin{bmatrix}\mathbf{x}_a & \mathbf{x}_b & \mathbf{x}_c\end{bmatrix}^T$. Matrica $\mathbf{K}$ definirana je kao:

$$\mathbf{K} = \frac{2}{3}\begin{bmatrix}\cos(\theta_r) & \cos(\theta_r - \frac{2\pi}{3}) & \cos(\theta_r + \frac{2\pi}{3}) \\ -\sin(\theta_r) & -\sin(\theta_r - \frac{2\pi}{3}) & -\sin(\theta_r + \frac{2\pi}{3})\end{bmatrix} \tag{2-12}$$

Jednadžba (2-11) primijenjena je tijekom izrade modela motora koji se simulira u ovom radu.

2.2.2 Dvoosni model sinkronog motora

Primjenom transformacije definirane jednadžbom (2-11) na jednadžbe koje opisuju model sinkronog motora dobije se model sinkronog motora u dvoosnom dq sustavu. Detaljan raspis jednadžbi može se pronaći u [15]. Pa tako naponska jednadžba (2-2) nakon primjene transformacije u krajnjem obliku glasi:

$$\mathbf{u}_{dq} = \mathbf{r}_s\mathbf{i}_{dq} + \omega_{el}\mathbf{J}_r\boldsymbol{\psi}_{dq} + \frac{\mathrm{d}}{\mathrm{d}}(t)\boldsymbol{\psi}_{dq} \tag{2-13}$$

gdje je $\mathbf{J}_r$ matrica rotacije:

$$\mathbf{J}_r = \begin{bmatrix}0 & -1 \\ 1 & 0\end{bmatrix} \tag{2-14}$$

Postupak se ponavlja za izraz (2-3) te se dobiva:

$$\boldsymbol{\psi}_{dq} = \mathbf{L}_{sdq}\mathbf{i}_{dq} + \boldsymbol{\psi}_{Mdq} \tag{2-15}$$

gdje je:

$$\mathbf{L}_{sdq} = \begin{bmatrix}L_{ls} + \frac{3}{2}(L_A + L_B) & 0 \\ 0 & L_{ls} + \frac{3}{2}(L_A - L_B)\end{bmatrix} = \begin{bmatrix}L_d & 0 \\ 0 & L_q\end{bmatrix} \tag{2-16}$$

Novi dvoosni dq sustav orijentiran je prema položaju rotora ($\theta_r = 0$), kao jedan od preduvjeta vektorske regulacije, odnosno d os novonastalog koordinatnog sustava poravnata je sa permanentnim magnetom, stoga vrijedi da je:

$$\boldsymbol{\psi}_{Mdq} = \psi_m\begin{bmatrix}1 \\ 0\end{bmatrix} \tag{2-17}$$

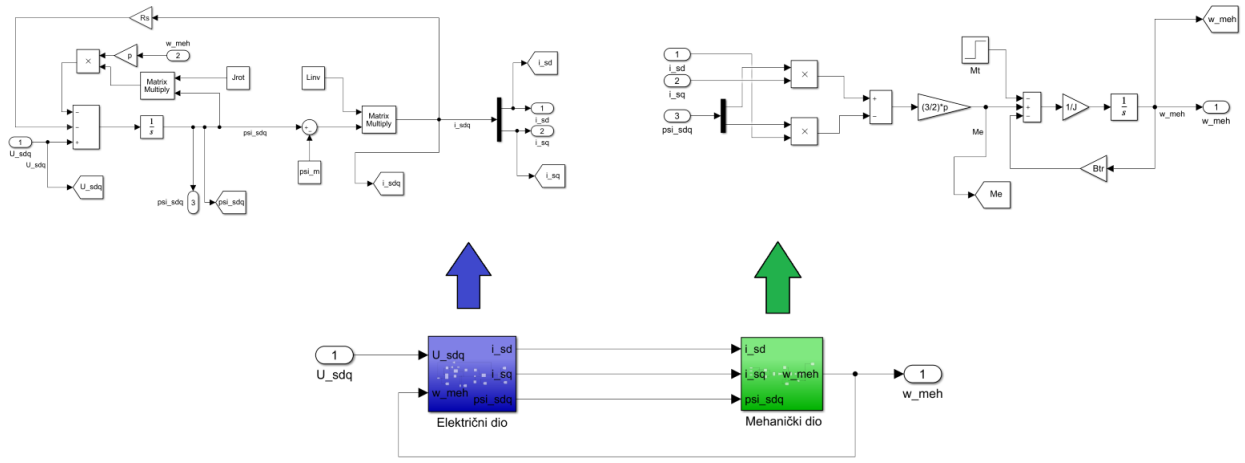
Zadnji izraz nad kojim se provodi transformacija jeste (2-8) - elektromagnetski moment. Primjenom transformacije dobiva se:

$$M_e = \frac{3}{2}p(\psi_d i_q + \psi_q i_d) \tag{2-18}$$

Elektromagnetski moment je moguće izraziti kao razliku uzbudnog momenta što ga stvara trajni magnet te reluktantnog momenta. Tada izraz za elektromagnetski moment poprima sljedeći oblik:

$$M_e = \frac{3}{2}(\psi_m i_q - (L_q - L_d) i_d i_q) \tag{2-19}$$

Model motora opisan jednadžbama (2-13), (2-15), (2-19) i (2-9) je model sinkronog motora s permanentnim magnetima zapisan u proizvoljno rotirajućem dq sustavu. Dobiven model je linearan, vremenski nepromjenjiv. Implementacija istog izvršena u Simulinku prikazana je slikom 2.2.



SL. 2.2: *dq model PMSM-a implementiranog u Simulinku*

2.3. Vektorska regulacija

Metoda vektorske regulacije (eng. *FOC - Field Oriented Control*) ima za svrhu zasebno upravljanje magnetskim poljem i elektromagnetskim momentom putem precizne kontrole d i q komponenti statorske struje [17]. Za primjenu ove metode potrebno je:

1. Poznavati linearan vremenski nepromjenjiv model motora
2. Postaviti q komponentu toka na nulu ($\psi_{rq} = 0$)
3. Poznavati položaj i iznos magnetskog toka rotora (ψ_{rd})

Ispunjavanjem prethodno navedenih uvjeta moguće je realizirati vektorsku regulaciju korištenjem PI regulatora, točnije njih tri. Model i svrha svakog od njih pojašnjeni su u daljnjem tekstu.

2.3.1 Regulator struje

Razdvajanjem statorske struje na dvije komponente omogućeno je da se kontrolom jedne (d komponente) upravlja magnetskim tokom, a kontrolom druge (q komponente) upravlja elektromagnetskim momentom odakle slijedi da se za regulaciju statorske struje primjenjuju dva nezavisna PI regulatora. Kako bi se poništio utjecaj reluktantnog momenta referenca struje u d osi postavlja se na vrijednost nula ($i_d^* = 0$) odakle slijedi da je izraz za elektromagnetski moment (2-19) postaje jednak:

$$M_e = \frac{3}{2}\psi_m i_q \tag{2-20}$$

Jednadžba 2-20 opravdava gore navedenu tvrdnju da kontrolom struje u q osi upravlja elektromagnetskim momentom.

Prvi korak za dobivanje prijenosnih jednadžbi regulatora je razdvajanje jednadžbe (2-13) na d i q komponentu.

$$u_d = R_s i_d - \omega_r L_q i_q + \frac{\mathrm{d}}{\mathrm{d}t} L_d i_d \tag{2-21}$$

$$u_q = R_s i_q + \omega_r L_d i_d + \omega_r \psi_m + \frac{\mathrm{d}}{\mathrm{d}t} L_q i_q \tag{2-22}$$

Radi izbjegavanja diferencijalnih jednadžbi prilikom modeliranja regulatora na jednadžbe (2-21) i (2-22) primijenit će se Laplaceova transformacija kojom se iz vremenske prelazi u kompleksnu (frekvencijsku) domenu. Detaljnije o samoj transformaciji se može pronaći u [18].

$$\mathcal{L}\left\{\frac{\mathrm{d}f(t)}{\mathrm{d}t}\right\} = sF(s) \tag{2-23}$$

Jednadžba (2-23) predstavlja transformiranu operaciju deriviranja gdje je $f(t)$ funkcija u vremenskoj, a $F(s)$ funkcija u kompleksnoj domeni. Njenom primjenom jednadžbe (2-21) i (2-22) iz diferencijalnog oblika prelaze u algebarski oblik:

$$U_d(s) = R_s I_d(s) - \omega_r L_q I_q(s) + s L_d I_d(s) \tag{2-24}$$

$$U_q(s) = R_s I_q(s) + \omega_r L_d I_d(s) + \omega_r \psi_m + s L_q I_q(s) \tag{2-25}$$

Daljnjim sređivanjem se dobije:

$$U_d(s) - E_d(s) = I_d(s)(R_s + sL_d) \tag{2-26}$$

$$U_q(s) - E_q(s) = I_q(s)(R_s + sL_q) \tag{2-27}$$

gdje E_d i E_q predstavljaju protuelektromotornu silu:

$$E_d(s) = -\omega_r L_q I_q(s) \tag{2-28}$$

$$E_q(s) = \omega_r L_d I_d(s) + \omega_r \psi_m \tag{2-29}$$

Prijenosna funkcija nekog sustava jeste omjer izlazne i ulazne funkcije. Ulazna funkcija u ovom slučaju jeste struja $I(s)$ dok je izlazna funkcija napon $U(s)$ umanjen za iznos protuelektromotorne sile $E(s)$. Stoga slijedi:

$$\frac{I_d(s)}{U_d(s) - E_d(s)} = \frac{\frac{1}{R_s}}{1 + sT_{sd}} \tag{2-30}$$

$$\frac{I_q(s)}{U_q(s) - E_q(s)} = \frac{\frac{1}{R_s}}{1 + sT_{sq}} \tag{2-31}$$

gdje su $T_{sd} = \frac{L_d}{R_s}$ i $T_{sq} = \frac{L_q}{R_s}$ vremenske konstante.

Unutarnji regulacijski krug u kom se regulira struja sastoji se od PI regulatora i prijenosnih funkcija sustava. Tako prijenosna funkcija otvorenog regulacijskog kruga glasi(daljnji postupak izveden je za struju u d osi dok se identičan izvodi i za struju u q osi):

$$G_o(s) = \frac{1 + sT_D}{sT_i} \frac{\frac{1}{R_s}}{1 + sT_{sd}} \tag{2-32}$$

PI regulator unutarnjeg regulacijskog kruga parametrira se prema tehničkom optimumu prema kojem slijedi da je $T_D = T_{sd}$ čime se poništava dominantna vremenska konstanta. Nakon primjene tih načela izraz za otvoreni regulacijski krug glasi:

$$G_o(s) = \frac{\frac{1}{R_s}}{sT_i} \tag{2-33}$$

Ulaz u regulator je razlika između referentne i mjerene vrijednosti struje I_d. Povratnom vezom struje se postiže zatvoreni regulacijski krug čija je prijenosna funkcija:

$$G_x(s) = \frac{I_d}{I_d*} = \frac{G_o(s)}{1 + G_o(s)} = \frac{1}{1 + sT_iR_s} \tag{2-34}$$

Prema tehničkom optimumu, osim što je $T_D = T_{sd}$, vrijedi da je $T_i = 2k_sT_x$ odnosno za ovaj konkretan slučaj:

$$T_i = 2\frac{1}{R_s}T_x = \frac{2T_x}{R_s} \tag{2-35}$$

gdje je T_x vremenska konstanta zatvorenog regulacijskog kruga.

PI regulatori korišteni u ovom radu izvedeni su kao suma proporcionalnog i integralnog člana te poprimaju sljedeći oblik:

$$Gx(s) = K_p + \frac{K_i}{s} = \frac{\frac{K_p}{K_i}s + 1}{\frac{1}{K_i}s} \tag{2-36}$$

gdje parametri po simetričnom optimumu poprimaju vrijednosti:

$$T_D = \frac{K_p}{K_i} \tag{2-37}$$

$$T_i = \frac{1}{K_i} \tag{2-38}$$

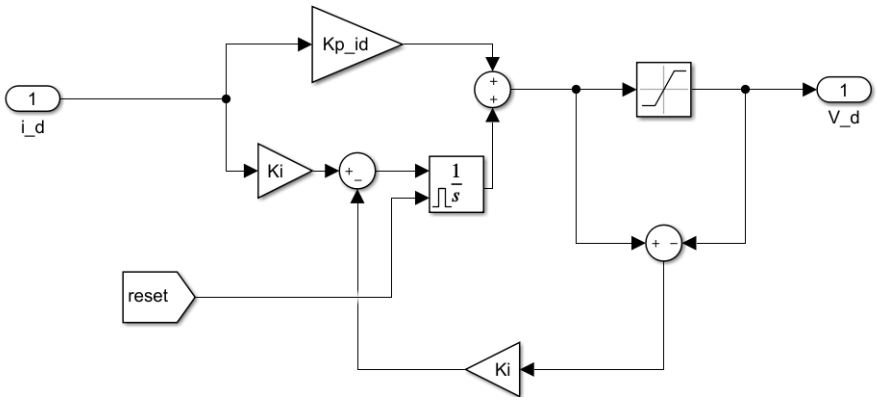
Daljnjim uvrštavanjem slijedi:

$$K_{id} = K_{iq} = \frac{R_S}{2T_x} \tag{2-39}$$

$$K_{pd} = \frac{L_d}{2T_x} \tag{2-40}$$

$$K_{pq} = \frac{L_q}{2T_x} \tag{2-41}$$

U ovom radu su prilikom parametriranja PI regulatora uvedena ograničenja. Prvo ograničenje odnosi se na iznos veličina koje regulator šalje. Kako bi simulacija sustava bila što sličnija stvarnom sustavu, u kojem aktuatori ne mogu dati više od maksimalne vrijednosti veličine, u Simulink modelu regulatora dodaje se blok *saturation* kojim se to ograničenje realizira. Zatim, kako bi se riješili problemi glede pojave nestabilnosti u sustavu, predugog vremena ustaljivanja ili prevelikog odstupanja uvodi se ograničenje izlazne veličine integratora (eng. *Anti-windup*). Naime, kada je limitator na graničnoj vrijednosti, integrator regulatora i dalje integrira regulacijsko odstupanje i postiže vrijednost koja je veća od granične vrijednosti limitatora, stoga dolazi do zasićenja integratora (eng. *integrator windup*). Rješenje je moguće realizirati na više načina, dok je u ovom slučaju odabran postupak povratnog računanja (eng. *tracking back calculation*) [10, 18].



SL. 2.3: *PI regulator struje sa anti-windup-om*

2.3.2 Regulator brzine

Regulacija brzine vrtnje vrši se kako bi pri utjecaju momenta tereta brzina ostala konstantna. Regulator brzine se kaskado spaja na regulator struje. Prijenosna funkcija vanjskog regulacijskog kruga rezultat je kaskadnog spajanja prijenosne funkcije PI regulatora brzine sa funkcijom $G_x(s)$ unutarnjeg regulacijskog kruga:

$$G_o(s) = \frac{T_{D\omega} + 1}{T_{i\omega} s} \frac{1}{1 + 2T_x s} \frac{1}{Js} \tag{2-42}$$

PI regulator brzine parametrira se prema simetričnom optimumu zbog integralnog djelovanja u sustavu. Parametri T_D i T_i tada poprimaju iznose:

$$T_{D\omega} = 4Tx \tag{2-43}$$

$$T_{i\omega} = 8k_s T_x^2 = \frac{8T_x^2}{J} \tag{2-44}$$

Još jedna promjena u odnosu na unutarnji regulacijski krug (regulaciju struje) jeste dodavanje predfiltera koji za svrhu ima kompenzirati nadvišenja odnosno usporiti prijelaznu pojavu. Njegova jednadžba glasi:

$$G_{pf} = \frac{k_{pf}}{1 + sT_{pf}} \tag{2-45}$$

gdje je $k_{pf} = 1$, a $T_{pf} = T_{D\omega} = 4T_x$. Zatvoreni regulacijski krug prikazan je jednadžbom (2-46)

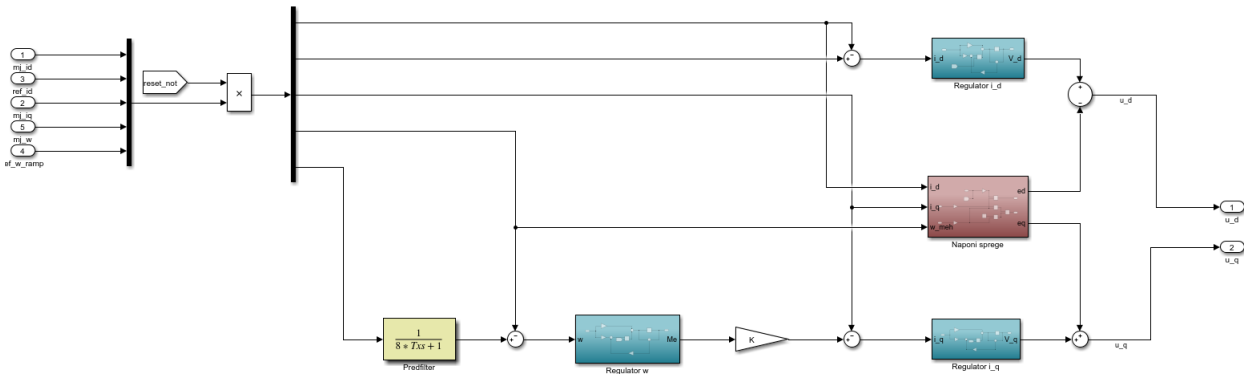
$$G_x(s) = \frac{4T_x s + 1}{16T_x^3 s^3 + 8T_x^2 s^2 + 4T_x s + 1} \tag{2-46}$$

PI regulator brzine prikazan je kao suma proporcionalnog i integralnog člana čiji parametri poprimaju sljedeće vrijednosti:

$$K_{p\omega} = \frac{J}{2T_x} \tag{2-47}$$

$$K_{i\omega} = \frac{J}{8T_x^2} \tag{2-48}$$

Kao i kod modela regulatora struje uvedeno je ograničenje izlazne veličine integratora i regulatora. Cjelokupni model regulatora prikazuje slika 2.4.



SL. 2.4: *Regulator*

Konačni model se sastoji od tri PI regulatora, jednog predfiltera i bloka "Naponi sprge". Blok je modeliran prema jednadžbama (2-30) i (2-31). Njihov negativan predznak osigurava poništenje protuelektromotornih sila u motoru.

3. HIL TESTING METODOLOGIJA

Simulacije imaju ključnu ulogu kada se govori o razvoju i dizajniranju velikog broja električnih sustava (od velikih elektroenergetskih postrojenja do optimizacije elektromotornih pogona). Napretkom tehnologije performanse simulacijskih alata značajno su porasle i osposobile ih za rješavanje kompleksnih problema [19]. U ovom radu provedena je stvarnovremenska kosimulacija što podrazumijeva izmjenu signala između simulatora u stvarnom vremenu (eng. *Real Time - RT*). Stvarnovremenske simulacije zbog uvjeta rada u stvarnom vremenu imaju određena ograničenja kao što su diskretno vrijeme i konstantan vremenski korak (eng. *fixed step size*). U kosimulacijskom postavu različiti modeli i alati rade zajedno stvarajući cjelovitu simulaciju. Ideja koja stoji iza kosimulacije jeste izgradnja sustava iz međusobno neovisnih i samostalnih modela čime se olakšava neovisno modificiranje komponenti te korištenje najprikladnijih alata za svaki od podsustava [20].

Simulacije se mogu definirati kao eksperimentiranje nad modelima, što znači da za provođenje kosimulacije treba modelirati sustav koji se želi ispitivati. U ovom radu ispituje se regulator za vektorsko upravljanje PMSM-om te je stoga bilo potrebno napraviti modele i PMSM-a i regulatora. Matematički model PMSM-a i regulatora, predstavljeni u prethodnom poglavlju, modelirani su prema MBD (eng. *model-based-design*) metodi. MBD je matematička i grafička metoda za rješavanje problema povezanih sa dizajnom složenih sustava. Kod implementiranja MBD metode razlikuju se dva tipa modela, a to su [21, 19]:

1. Objekt regulacije (eng. *plant*) - PMSM
2. Regulator (eng. *controller*)

MBD metoda se sastoji od četiri koraka koji se izvode redom[21]:

1. Izrada modela objekta regulacije
2. Analiza modela objekta regulacije i sinteza regulatora
3. Simulacija modela objekta regulacije i regulatora zajedno
4. Implementacija regulatora

U ovom radu odrađena su prva tri koraka. Primjena ove metode inače se odabire zbog mogućnosti da svi inženjeri uključeni u dizajn i modeliranje sustava imaju mogućnost provoditi ispitivanja tijekom faze razvoja kako bi potvrdili i ispunili dane zahtjeve nad sustavom. Tehnički rizici se umanjuju rano u razvojnom procesu [21, 19]. Na taj način testiranja se mogu provoditi vrlo rano u procesu stvaranja prije nego li stvarni fizički sustavi bivaju izgrađeni.

Spomenuta metoda obuhvaća nekoliko vrsta testiranja koje omogućuju provođenje ispitivanja vrlo rano u procesu stvaranja prije nego li stvarni fizički sustavi bivaju izgrađeni. Sva ta testiranja se mogu rasporediti po razinama, koje predstavljaju razvojni proces jednog proizvoda, pa tako polazeći od najniže razine postoje [19]:

1. Model u petlji (eng. *MIL - Model-in-loop*)
2. Softver u petlji (eng. *SIL - Software-in-loop*)
3. Procesor u petlji (eng. *PIL - Proccesor-in-loop*)
4. Hardver u petlji (eng. *HIL - Hardware-in-loop*)
5. Okolina u petlji (eng. *EIL - Enviroment-in-loop*)

6. Korisnik u petlji (eng. *CIL - Costumer-in-loop*)

U ovom radu provedeno je HIL testiranje odnosno varijacija HIL-a: implementacija regulatora koristeći RCP (eng. *Rapid control prototyping*) povezanog sa objektom regulacije (sinkronim motorom s permanentnim magnetima) kao HIL sustavom odnosno RCP uz objekt regulacije kao HIL sustav u kojem se posjeduje potpuna kontrola nad procesom regulacije. Koristeći se kombinacijom tih metoda ostvareni su uvjeti za provođenje kosimulacije.

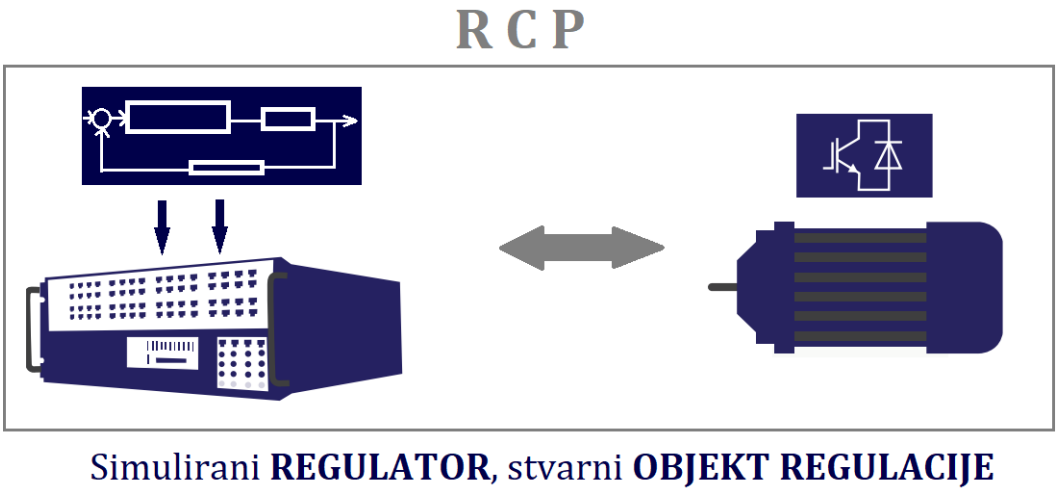
U [21] izdvojeno je niz prednosti koje nastaju kombiniranjem ovih dviju metodologija od koji su neke:

- Trajanje ciklusa projekta je smanjeno zbog mogućnosti istovremenog rada na pojedinim elementima sustava
- Troškovi su smanjeni zahvaljujući mogućnosti ponovnog korištenja stvarnovremenskih simulatora za različite projekte za razliku od fizičkih postava za testiranje koji nisu podložni modificiranju
- Rezultati se mogu ponavljati nebrojeno puta budući da se dinamika stvarnovremenskih simulatora ne mijenja kroz vrijeme na način na koji se fizički sustavi mijenjaju
- Zamijenjuju rizična i skupa ispitivanja koja bi se inače provodila na fizičkim ispitnim postavima

U nastavku teksta pojašnjene su RCP i HIL metoda, te implementacije simulacijskih modela regulatora u dSpace MicroLabBox stvarnovremenski simulator i PMSM-a u OPAL-ov OP5600 stvarnovremenski simulator čime su ostvareni uvjeti za izvođenje kosimulacije.

3.1. RCP

RCP igra ključnu ulogu u razvojnom ciklusu sustava ispitujući ispravnost algoritama koji se testiraju na stvarnovremenskim platformama koristeći stvarne senzore i aktuatore [22]. Inženjeri vrše rano, iterativno ispitivanje regulacije sustava na modelima prototipa prije nego što je hardver dostupan. To omogućava ubrzani razvoj regulatora temeljenih na modelima, postupnu integraciju i potvrdu ispravnosti dizajna [19]. Slika 3.5 ilustrira RCP metodu.



SL. 3.5: *RCP metoda*

RCP metoda podrazumijeva regulator implementiran u stvarnovremenski simulator i povezan sa stvarnim (fizičkim) objektom regulacije (motor + frekvencijski pretvarač).

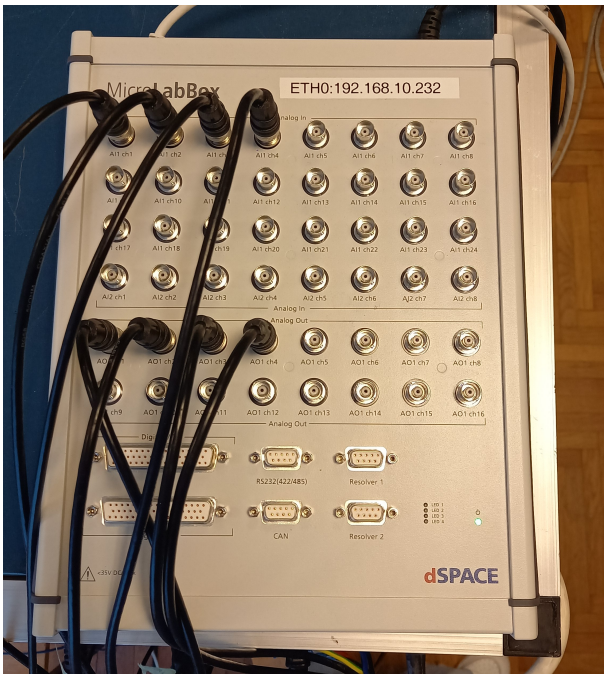
U ovom radu regulator služi za vektorsku regulaciju PMSM-a. Zahvaljujući RCP metodi moguće je umjesto pravog koristiti simulacijski model regulatora implementiran u stvarnovremenski simulator te na taj način vršiti ispitivanja nad regulatorom. Korišten stvarnovremenski simulator za potrebe RCP metode u ovom radu jeste dSpace MicroLabBox čiji je način rada pojašnjen u nastavku.

3.1.1 dSpace MicroLabBox

MicroLabBox je stvarnovremenski laboratorijski simulacijski sustav koji omogućava brzo i jednostavno ispitivanje, kontrolu te raznovrsna mjerenja, kako bi teorije upravljanja pretočili u stvarnost.

U ovom radu svrha MicroLabBox simulatora jeste omogućiti simulaciju regulatora za vektorsko upravljanje sinkronim motorom te u konačnici i optimizaciju parametara istog.

Za potrebe ovog rada upotrebljena je BNC varijanta MicroLabBox-a koja se sastoji se od FPGA (eng. —*field programable gate array*) i I/O (*input/output*) jedinice za simulaciju sa više od 100 I/O priključnica koje se dijele na analogne i digitalne. Naponi na analognim priključnicama mogu doseći najveću vrijednost od 10 V što zbog ograničenih mogućnosti simulatora što zbog sigurnosti osobe koja upravlja istim. Putem tih priključaka ostvarena je povezanost sa drugim simulatorom koji sudjeluje u kosimulaciji. Povezivanje sa računalom izvedeno je preko IP adrese pomoću RJ45 kabla koji je spojen na host PC priključak simulatora i Ethernet priključak računala. Za korištenje simulatora bilo je potrebno instalirati nekoliko programa od kojih će se izdvojiti MATLAB/Simulink programski paket te dSpace-ov ControlDesk. Budući da je model regulatora stvoren u MATLAB Simulnik-u softverske pakete dSpace-a bilo je nužno povezati sa MATLAB-om kako bi se dSpace biblioteke mogle koristiti u Simulink-u. Na slici 3.6 prikazan je korišteni dSpace-ov stvarnovremenski simulator.



SL. 3.6: *dSpace MicroLabBox simulator*

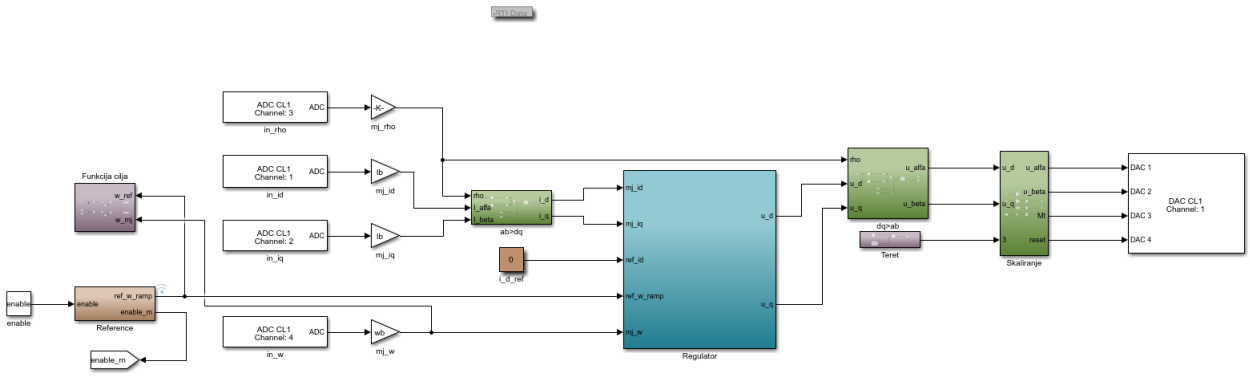
3.1.2 Modificiranje modela regulatora u Simulink-u

Kako je već spomenuto, model regulatora kojeg se simulira modeliran je u Simulinku-u. Nakon što je omogućena integracija MATLAB-a u dSpace-ove programe potrebno je modificirati model kako bi bio pogodan za implementaciju u simulator. Postupak modificiranja detaljno prati [10].

Prije izrade odnosno modifikacije modela bilo je potrebno prvo namjestiti simulacijsko okruženje. Stoga je pokretanjem Simulink programa prvo odabrano stvarnovremensko sučelje (eng. *real time interface, RTI*) koje predstavlja poveznicu između MATLAB-a i dSpace-ovog hardvera (ds1202), konkretno za ovaj simulator to je bio RTI1202 *template* čime su postavljeni uvjeti za razvoj modela za stvarnovremensku simulaciju. Nadalje su postavljeni konfiguracijski parametri:

- Konstantan vremenski korak (eng. *fixed step size*)
- Trajanje vremenskog koraka 0.0001 (10^{-4}) sekundi
- Metoda za numeričko rješavanje diferencijalnih jednadžbi (*solver*) - Euler (ode1)

Nakon spomenutih postavki dodani su novi blokovi i podsustavi nužni za daljnji rad. Slika 3.7 prikazuje finalni izgled simulacijskog modela koji je implementiran u MicroLabBox. U nastavku teksta do detalja je objašnjeno značenja svih blokova i podsustava.



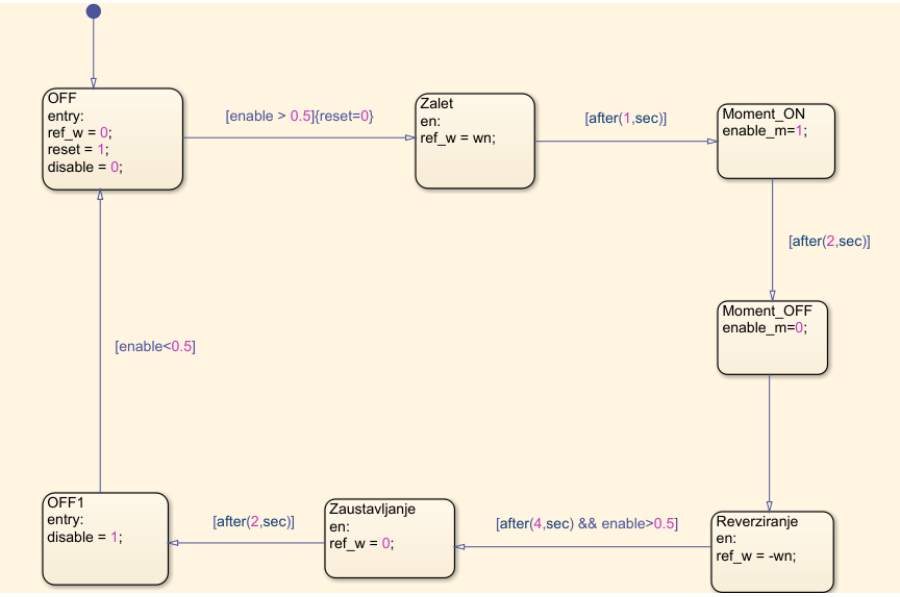
SL. 3.7: *Simulacijski model dSpace regulatora*

Razni dSpace blokovi su nakon integracije postali dio Simulink-ove biblioteke. Za potrebe ovog rada korišteni su blokovi za slanje i primanje signala drugog simulatora koji se nalazi u kosimulacijskom postavu. Signali koji se izmjenjuju su analogni pa tako analogni signali koje simulator prima u modelu prvo ulaze u blok *ADC* (*analog digital converter*) koji nakon pretvorbe u digitalne signale i skaliranja idu u regulator. Slično tome, kako regulator šalje digitalne signale oni se prvo skaliraju nakon čega idu u blok *DAC* (*digital analog converter*) koji ih pretvara u analogne signale nakon čega se mogu proslijediti drugom simulatoru. Za analogne signale vrijedi da ako je vrijednost signala u Simulink-u 1 tada će taj isti signal na analognom priključku imati vrijednost 10 V. Budući da je maksimalna vrijednost napona na simulatoru 10 V potrebno je skalirati sve signale koji se šalju kako ne bi prekoračili vrijednost 1 odnosno 10 V i na taj način oštetili uređaj pa i ugrozili osobu koja njim upravlja. Kao dodatno osiguranje nakon *gain* bloka, kojim se skaliraju vrijednosti, dodan je blok *saturation* kojim je osigurano da nakon skaliranja promatrani signal ima vrijednost iz intervala $\{-1, 1\}$. Blok

"Skaliranje" se odnosi na prethodno objašnjenu rekonstrukciju signala koji se šalju drugom simulatoru. Signali koje simulator prima nakon *ADC* bloka se prilagođavaju na način da se pomnože sa vrijednošću kojom su skalirani u simulatoru iz kojeg dolaze. To množenje se vrši pomoću bloka *gain*. Nakon osiguravanja uspješne razmjene signala model je spreman za implementaciju. Prije same implementacije urađene su dodatne preinake modela koje su omogućile što uspješnije daljnje rukovanje istim.

Vektorska regulacija podrazumijeva implementaciju regulatora koji osim signala mjerenih veličina koje dolaze iz motora (struja i brzine) također zahtjeva zadavanje referenci brzine vrtnje i struje. Referencama se zadaje ono što želimo da stvarni sustav prati. Pa je tako u potpoglavlju 2.3.1 objašnjeno kako se referenca struje postavlja na vrijednost 0 radi poništenja reluktantnog momenta. Blok "Reference" sadrži blok *Stateflow Chart* kojim je zadana referenca brzine vrtnje. Logika unutar bloka *Stateflow Chart* prikazana je na slici 3.8. Može se vidjeti kako se trajektorija koja predstavlja brzinu vrtnje sastoji od zaleta, terećenja, rasterećenja, stacionarnog stanja, reverziranja te zaustavljanje kako bi se tijekom provođenja eksperimenta mogao dobiti uvid u sva dinamička stanja motora.

Stateflow Chart je grafikon gdje stanja i prijelazi čine osnovne građevne blokove logičkog sustava. Stanja su prikazana unutar pravokutnih blokova, dok su prijelazi prikazani linijama. Prelazak iz jednog u drugi blok stanja osigurava ispunjenje uvjeta predstavljenim linijama.



SL. 3.8: *Logika unutar Chart bloka*

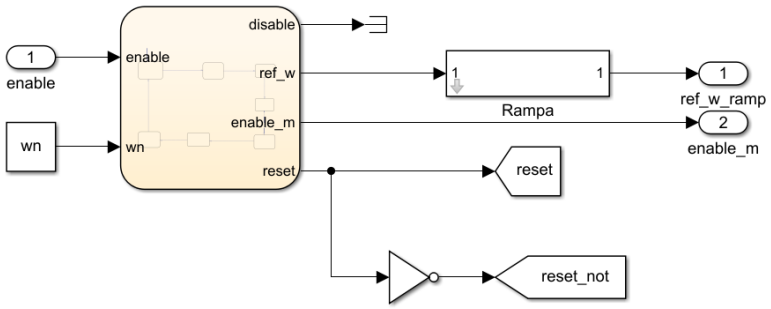
Prije nego li uslijedi objašnjene logike *Chart* bloka korisno je upoznati se sa značenjem pojedinih varijabli:

- *ref_w* - referentna brzina vrtnje
- *wn* - nazivna brzina vrtnje (314 rad/s)
- *enable_m* - varijabla koja služi za paljenje i gašenje tereta
- *reset* - varijabla koja kod integratora ima za cilj osiguravanje da je početno stanje (eng. *initial condition*) uvijek nula

Ostale varijable (*enable, disable*) nisu potrebne za razumijevanje logike ovog bloka stoga su one objašnjene u idućem poglavlju gdje je njihova primjena značajna.

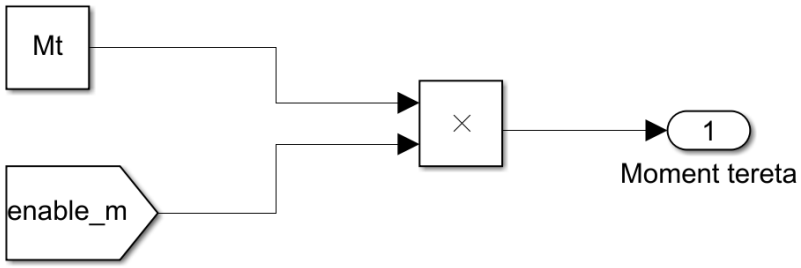
Početni blok sa strelicom iznad sadrži naslov "OFF" što znači da je početno stanje ono u kom je motor ugašen što se može zaključiti po referenci brzine vrtnje koja ima vrijednost nula čime se dokazuje da se motor ne vrti. Varijabla *reset* potrebna je u tom početnom stanju iz nekoliko razloga. Naime, kako će za potrebe rada biti nužno da se simulacija vrti nekoliko puta uzastopno potrebno je resetirati integratore jer bi u suprotnom po završetku svake simulacije početni uvjeti u integratorima zapravo bile vrijednosti koje je integrator zadnje integrirao od prethodne simulacije. Isto tako sve povratne veze, reference i mjerene vrijednosti vraćene su na vrijednost 0. Varijabla *reset* šalje se na drugi simulator gdje na isti način resetira motor (detaljnije objašnjeno u idućem poglavlju). Nakon što su ispunjeni uvjeti $[enable > 0.5]\{reset = 0\}$ (u uglatim zagradama nalaze se uvjeti koji se moraju zadovoljiti, dok se u vitičastim zagradama osigurava da prelaskom u novo stanje željena varijabla napravi određena promjena, kao što ovdje prelaskom iz bloka "OFF" u blok "Zalet" varijabla *reset* poprima vrijednost 0) prelazi se u idući blok odnosno stanje "Zalet".

Prelaskom u blok "Zalet" motor kreće ubrzavati do nazivne brzine vrtnje budući da je referenci brzine vrtnje pridodjeljena vrijednost *wn*. Zalet, kao i sve ostale promjene brzine vrtnje odvijaju se po rampi. Slika 3.9 prikazuje kako vrijednost koju poprimi varijabla *ref_w* po izlasku iz *Chart* bloka ulazi u blok "Rampa" nakon čega ista brzina koja se mijenja po rampi izlazi iz bloka "Reference" i kao takva se predaje u blok "Regulator".



SL. 3.9: *Sadržaj bloka Reference*

Nadalje nakon jedne sekunde započinje terećenje motora gdje u bloku "Moment_ON" varijabla "enable_m" poprima vrijednost 1 čime se osigurava paljenje tereta. Vodeći se istom logikom nakon dvije sekunde se prelazi u stanje "Moment_OFF" gdje se postavljanjem varijable "enable_m" na vrijednost 0 isti teret i gasi. Slika 3.10 prikazuje sadržaj bloka "Teret" te kako je riješeno spomenuto paljenje i gašenje tereta. *Constant* blok sadrži vrijednost momenta tereta,



SL. 3.10: *Sadržaj bloka Teret*

dok je *enable_m* varijabla koja dolazi iz *Chart* bloka. Naime, početna vrijednost varijable *enable_m* jeste 0 te kada se pomnoži sa vrijednošću momenta tereta daje nulu. Kada se u bloku stanja "Moment_ON" varijabla postavi na vrijednost 1 u bloku "Teret" sada izlazna vrijednost nema više vrijednost 0 nego vrijednost Mt odnosno tada nastupa terećenje motora.

Bitno je napomenuti kako je u trenutku paljenja tereta brzina vrtnje postala konstantna i ostala za vrijeme terećenja i rasterećenja stroja. Kako između blokova "Moment_OFF" i "Reverziranje" na liniji prijelaza nema uvjeta znači da odmah po prelasku u blok u kom se gasi teret nastupa reverziranje motora. Reverziranjem je motor prešao u generatorski režim rada budući da je brzina suprotnog predznaka što se može vidjeti i u bloku stanja gdje je referenca brzine vrtnje izjednačena sa vrijednošću $-wn$. Kod reverziranja motor prvo uspori do nule zatim se počinje vrtjeti u suprotnu stranu što je rezultat negativne vrijednosti brzine vrtnje. Nakon četiri sekunde prelazi se u stanje "Zaustavljanje" gdje motor slijedi zadanu referencu brzine vrtnje i polako usporava dok se ne zaustavi odnosno dok referenca ne poprimi vrijednost 0. Na taj način se ponovo dolazi u početno stanje "OFF" (blok stanja "OFF1" objašnjen u nastavku).

Osim zadavanja referenci u model prikazan na slici 3.7 dodan je blok "ab>dq" koji predstavlja transformaciju veličina koje dolaze iz motora u stacionarnom referentnom okviru u referentni okvir prilagođen brzini vrtnje rotora gdje kut, koji je potreban za transformaciju jedna od veličina koje dolaze iz drugog simulatora u kojem je implementiran model motora. Osim njega, prije bloka skaliranje dodan je blok "dq>ab" koji veličine koje se šalju u drugi simulator, na kojem je implementiran motor, transformira natrag u mirujući dvoosni $\alpha\beta$ sustav.

Prije same implementacije modela na simulator, izvršene su *offline* simulacije u Simulink-u kako bi se utvrdilo da je model ispravan i spreman za implementaciju. Utvrđivanjem ispravnosti modela regulatora, korištenjem Simulink Coder-a, koristeći naredbu *build* model je preveden u programski jezik C čime su nastale datoteke potrebne za daljnji rad na simulatoru. ControlDesk program korišten je za uređivanje simulacije nakon učitavanja na MicroLabBox. U njega je učitana *.sdf* datoteka koja sadrži sve varijable modela čime je omogućeno daljnje rukovanje simulacijom.

3.1.3 Automatizacija ControlDesk-a

ControlDesk jedan je od mnogih dSpace-ovih programa koji omogućava stvaranje HMI sučelja za kontrolu važnih veličina simulacijskog modela, snimanje podataka, prikaz kao i automatizaciju eksperimenta. Tijekom izrade ovog rada cilj je bio što više automatizirati cijeli eksperiment da bi u konačnici jedan klik miša bio dovoljan da se eksperiment izvrši. Automatizacija je izvršena koristeći Python skripte kojima se u nekoliko linija koda pokrenula simulacija i mjerenje, prikazali se i spremili podaci, izvršila optimizacija, zaustavilo mjerenje i simulacija. Prvobitna ideja bila je putem Python editora *Spyder*, vanjskog neovisnog editora, izvršiti automatizaciju eksperimenta. Prepreke koje su se pojavile u realizaciji te ideje, nekompatibilnost modula i biblioteka svojstvenih dSpace simulatoru i instaliranoj verziji Python-a, primorale su na korištenje internog editora koji se nalazi unutar programa ControlDesk. Pogreška (eng. *error*) koja se ispriječila za rad sa vanjskim editorom, *bad magic number*, indicira kako postoje razlike u Python verzijama između one koju posjeduje editor i one u kojem se nalaze moduli s kojima se želi raditi. Kako nije bilo moguće otkloniti pogrešku izabran je interni editor za daljnji rad.

Nakon stvaranja projekta i eksperimenta te učitavanja *.sdf* datoteke ostvareni su uvjeti za rad sa simulacijom u ControlDesk-u. Slijedeći priručnik pod nazivom "ControlDesk Automation" te služeći se Python skriptama sa primjerima korištenja dSpace funkcija (primjerice funkcije kojima se pokreće i zaustavlja mjerenje u određenom trenutku) složene su Python skripte potrebne za automatizaciju eksperimenta.

Prvi korak bio je omogućiti pristup varijablama modela. Varijable su bile sve vrijednosti unutar modela regulatora u obliku pojačanja (*gain*), konstanti (*Constant*) te izlaznih vrijednosti integratora, sustava i podsustava. Kako je glavni zadatak ovog rada optimizacija parametara vektorske regulacije morao se omogućiti pristup tim parametrima te parametrima *enable, disable*. Način na koji je pristupljeno varijablama pokazuje kod u nastavku:

Primjer koda 1: *Pristup varijablama modela*

```
1         #pristup platformi na koju je ucitan eksperiment
2         MyPl = Application.ActiveExperiment.Platforms['Platform']
3
4         #Varijable regulatora struje Id
5         Ki_id = MyPl.ActiveVariableDescription.Variables['Platform():/Model_
              ↪ Root/Regulator/Regulator_i_d/Gain5/Ki']
6         Kp_id = MyPl.ActiveVariableDescription.Variables['Platform():/Model_
              ↪ Root/Regulator/Regulator_i_d/Gain4/Kp_id']
7         Ki_id_p = MyPl.ActiveVariableDescription.Variables['Platform():/Model
              ↪ _Root/Regulator/Regulator_i_d/Gain1/Ki']
```

Isti postupak napravljen je za pristup svim potrebnim varijablama. Nakon toga započet je proces automatizacije eksperimenta. Prvo se pristupilo inicijalizaciji kako bi se korištenjem naredbi moglo upravljati programom, od pokretanja *online* simulacije do upravljanja mjerenjem i snimanjem podataka, dinamičkog mijenjanja parametara za vrijeme simulacije i optimizacije pa do zaustavljanja. Pozivanjem funkcije *Initialize* pokrenuta je *online* simulacija (model se počinje simulirati na MicroLabBox-u) i započet proces mjerenja (zadnje dvije linije koda):

Primjer koda 2: *Pristup varijablama modela*

```
1  def __init__(self):
2         self.ControlDeskApplication = None
3         self.MeasurementDataManagement = None
4         self.Enums = None
5
6  def Initialize(self):
7         self.ControlDeskApplication = Dispatch("ControlDeskNG.Application.7.5
              ↪ ")
8         self.MeasurementDataManagement = DispatchWithEvents(self.
              ↪ ControlDeskApplication.MeasurementDataManagement,
              ↪ MeasurementDataManagementEvents)
9         self.Enums = Enums(self.ControlDeskApplication)
10        self.ControlDeskApplication.CalibrationManagement.
              ↪ StartOnlineCalibration()
11        self.MeasurementDataManagement.Start()
```

Nakon tog napravljene su razne postavke za mjerenje i snimanje podataka u čije detalje se neće ulazit. Najvažnija funkcija odnosno ona koja je zaslužna za veći dio automatizacije ovog eksperimenta prikazana je u nastavku:

Primjer koda 3: *Pokretanje i zaustavljanje simulacije*

```
def ConfigureMeasurement(self):
        while enable.ValueConverted == 0:
                if self.MeasurementDataManagement.IsMeasuring == True:
                        Wait(0.3)
                        enable.ValueConverted = 1
        while enable.ValueConverted == 1:
                if disable.ValueConverted > 0.5:
                        fc_w = f_cilja.ValueConverted
                        fc_iq = f_cilja_iq.ValueConverted
                        enable.ValueConverted = 0
```

Najprije će se objasniti varijable koje se pojavljuju unutar koda:

- *enable* - varijabla koja se nalazi unutar *Chart* bloka koja je jedan od uvjeta za rad motora. Kao što samo ima kaže ona omogućava (eng. enable) rad motora
- *disable* - varijabla koja označava prestanak rada motora

Svaka od varijabli "djeluje" kada poprimi vrijednost 1 (u *Chart* grafikonu je uvjet predan u obliku $enable > 0.5$ zbog toga što uvjeti u obliku jednadžbi često znaju stvarati problem pa se na ovaj način osigurava sigurno ispunjenje uvjeta). Inicijalne vrijednosti obje varijable su 0.

Prva *while* petlja kaže da dok je $enable = 0$ da se provjerava je li zadovoljen uvjet tj. je li mjerenje pokrenuto (mjerenje se pokreće pozivanjem funkcije *Initialize*) i ako jest da pričeka 0,3 sekunde te da varijabli *enable* pridoda vrijednost 1. Vraćajući se na *Chart* blok prikazan slikom 3.8 vidljivo je kako zalet motora kreće tek nakon što je zadovoljen uvjet $enable > 0.5$. Budući da dokle god je prethodno spomenuti uvjet ispunjen motor se vrti dopušteno je reći da se izvršavanjem prve *while* petlje pokreće motor.

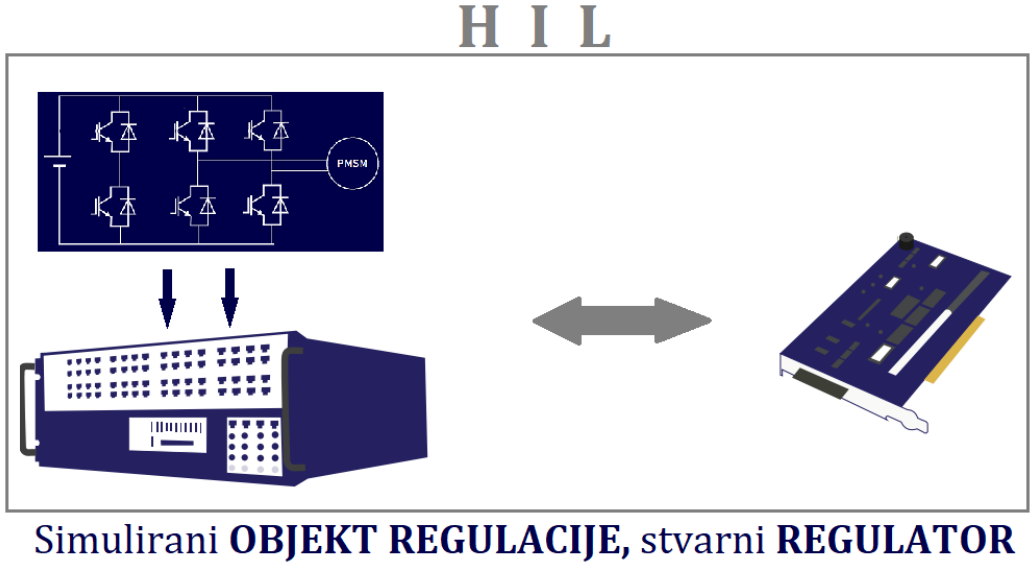
Druga *while* petlja kaže da dokle god je $enable = 1$ da se provjerava je li varijabla $disable = 1$. Vraćajući se na *Chart* grafikon vidljivo je kako se u bloku "OFF1" vrijednost *disable* varijable postavila na 1. Stanje opisano u tom bloku nastupa nakon što se motor zaustavio. Kada je *if* uvjet ispunjen, odnosno kada je motor zaustavljen, vrijednost varijable *enable* se ponovo postavlja na vrijednost 0. Prateći dalje grafikon na prijelazu iz "OFF1" u "OFF" zadan je uvjet $enable < 0.5$ odnosno da je vrijednost *enable* varijable 0 čime se dolazi opet u početno stanje. U bloku "OFF" vrijednost *disable* se vraća na nulu, a varijabla *reset*, čija je funkcija opisana u prethodnom poglavlju, poprima vrijednost 1 čima resetira regulator i motor te nakon 0,3 sekunde motor se ponovo pali. Simulacija se uzastopno pokreće na ovaj način samo ukoliko se uzastopno poziva funkcija *ConfigureMeasurements* (u nastavku rada prikazana je upotreba navedene funkcije na spomenut način).

U ovom potpoglavlju objašnjena je implementacija modela regulatora u MicroLabBox simulator te automatizacija procesa upravljanja, unutar ControlDesk programa, nad implementiranim regulatorom. U potpoglavlju koje slijedi je objašnjen postupak implementacije za pripadajući motor.

3.2. HIL

HIL je simulacijsko i testno okruženje za "ugrađene sustave" (eng. *embedded systems*). Za sustave koji se testiraju (eng. *SUT - systems under testing*) se pretpostavlja da rade sa signalnim ulazima i izlazima iz stvarnog svijeta [23]. Tako je omogućeno detaljno ispitivanje sustava prije nego što se fizički izgradi ili primijeni u stvarnom svijetu.

HIL metoda podrazumijeva objekt regulacije (u ovom slučaju PMSM) implementiran u stvarnovremenski simulator i stvarni (fizički) regulator. Ilustracija HIL metode nalazi se na slici 3.11.



SL. 3.11: *HIL metoda*

Kako je već spomenuto u ovom radu je primijenjena varijacija HIL metode stoga je regulator implementiran u simulator koristeći RCP metodu čiji su detalji objašnjeni u prethodnom poglavlju. Model PMSM-a implementiran je u stvarnovremenski simulator OPAL-RT koji je objašnjen u nastavku.

3.2.1 OPAL-RT stvarnovremenski sustv

Za potrebe HIL simulacije korišten je OP5031 stvarnovremenski simulator uparen sa OP5600 simulacijskim sustavom (slika 3.12). Ta vrsta simulatora se upotrebljava za stvarnovremenske simulacije, kosimulacije te XIL (*X-in-the-loop*) simulacije. OP5031 stvarnovremenski simulator sadrži mikroprocesor sa RT Linux operativnim sustavom, dok OP5600 sadrži FPGA (eng. *field-programable gate array*) i I/O jedinice (digitalne i analoge ulazno/izlazne priključke preko kojih se povezuje s drugom stvarnovremenskim sustavima). Povezivanje OP5600 stvarnovremenskog simulatora s drugim simulatorima odrađuje se preko DB37 konektora. OP5600 simulator se sastoji od 16 DB37 priključaka, podijeljenih u 4 grupe po 4 DB37 priključka. Unutar grupe postoje podgrupe koje predstavljaju ulazne i izlazne priključke. Ako se u podgrupi spajaju analogni signali tada podgrupa može maksimalno imati 16 signala dostupno, a u slučaju da se spajaju digitalni signali onda podgrupa može imati maksimalno 32 signala. Način spajanja kabla na DB37 konektor objašnjen je shematski i tablično u [24].

RT-LAB je OPAL-RT-ov softverski paket koji omogućava implementaciju modela u simulator, izvođenje simulacija u stvarnom vremenu i provođenje različitih testiranja, HIL testiranje. RT-LAB može se integrirati sa različitim alatima za modeliranje od kojih je jedan i

SL. 3.12: *OPAL-RT stvarnovremenski sustav*

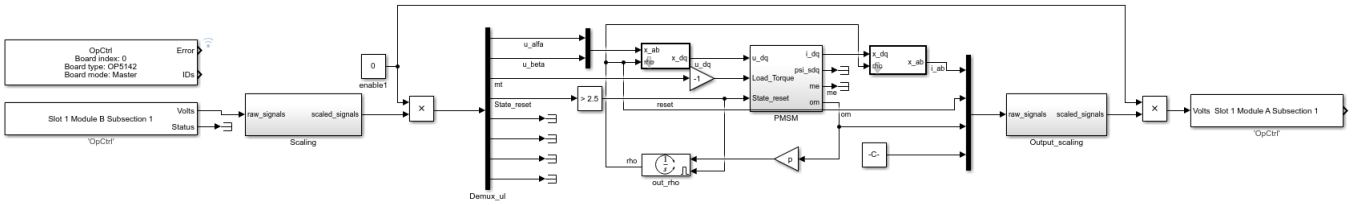
MATLAB/Simulink čime je omogućena implementacija modela motora u simulator. U nastavku su opisani postupci koji su prilagodili model motora za implementaciju u simulator pri čemu se detaljno prati [10].

3.2.2 Modificiranje modela PMSM-a u Simulnik-u

Kao što je prethodno rečeno simulatori su upareni sa programom RT-LAB te se zbog kompatibilnosti sa simulatorima uređivanju Simulink modela pristupa iz RT-LAB-a. Na taj način su dostupni svi alati koji omogućuju simulaciju, a ujedno se i biblioteka Simulink-a proširuje s dodatnim grupama blokova kao što su RT-EVENTS, RT-LAB I/O, RT-Drive i drugi. Budući da se radi o kosimulaciji s konstantnim iznosom koraka (eng. *fixed step size*) potrebno je postaviti, slično kao i kod regulatora, konfiguracijske parametre. Pa je tako za metodu rješavanja dierencijalnih matematički jednadžbi (*solver*) odabran ode3 (Bogacki-Shampine), a vremenski korak iznosi 0.00001 (10^{-5}) sekundi. U slučaju odabira dužeg vremenskog koraka gubi se na točnosti jer se samim time smanjuje broj uzoraka koji rekonstruiraju signale, a kod odabira kraćeg vremenskog koraka se riskira da se proračuni ne stignu izvršiti do kraja čime bi se ugrozila simulacija odnosno narušio željeni način rada u stvarnom vremenu.

Nakon postavljanja parametara simulacijskog okruženja uslijedile su promjene u modelu motora. Kao što je bio slučaj i prilikom modificiranja modela regulatora isto tako je i kod modela motora bilo potrebno dodati odgovarajuće blokove koji će omogućiti izmjene signala između simulatora točnije kosimulaciju. Za izmjenu signala korišten potrebno je prvo blok *OpCtrl* u kojem se povezuju ulazno/izlazni blokovi sa ulazno/izlaznim sučeljima preko kojih izmjenjuju signali tijekom izvođenja kosimulacije. Primanje signala od drugog simulatora i prosljeđivanje istih dalje u simulacijski model postiže se blokom *AnalogIn*, dok se slanje signala prema drugom simulacijskom sustavu postiže upotrebom bloka *AnalogOut*. Kod postavkih svakog od spomenutih blokova potrebno je odabrati određene parametre radi kompatibilnosti sa blokom *OpCtrl*, dodatno kod bloka *AnalogOut* potrebno je namjestiti naponsku razinu. Odabrana naponska razina je ±10 V zbog kompatibilnosti sa drugim simulatorom (MicroLabBox-om) i lakšeg preračunavanja prilikom skaliranja signala. Skaliranje je odrađeno po istom načelu kao i kod modela regulatora predstavljenom u potpoglavlju 3.1 kombinacijom *gain* i *saturation*

blokova. Razlika kod izlaznih signala u odnosnu na MicroLabBox jeste ta što je vrijednost digitalnog signala jednaka vrijednosti analognog. Znači ako je vrijednost digitalnog signala 1 na priključnici OPAL-a napon će biti 1 V a ne 10 V kao što je slučaj kod MicroLabBox-a stoga se interval vrijednosti unutar *saturation* bloka postavlja na $\{-10, 10\}$. Kako svi analogni ulazni signali imaju vrijednost 10 V potrebno ih je pomnožiti sa vrijednošću 10 puta manjom od one s kojom su skalirani unutar modela regulatora čime se dobije stvarna vrijednost digitalnog signala.



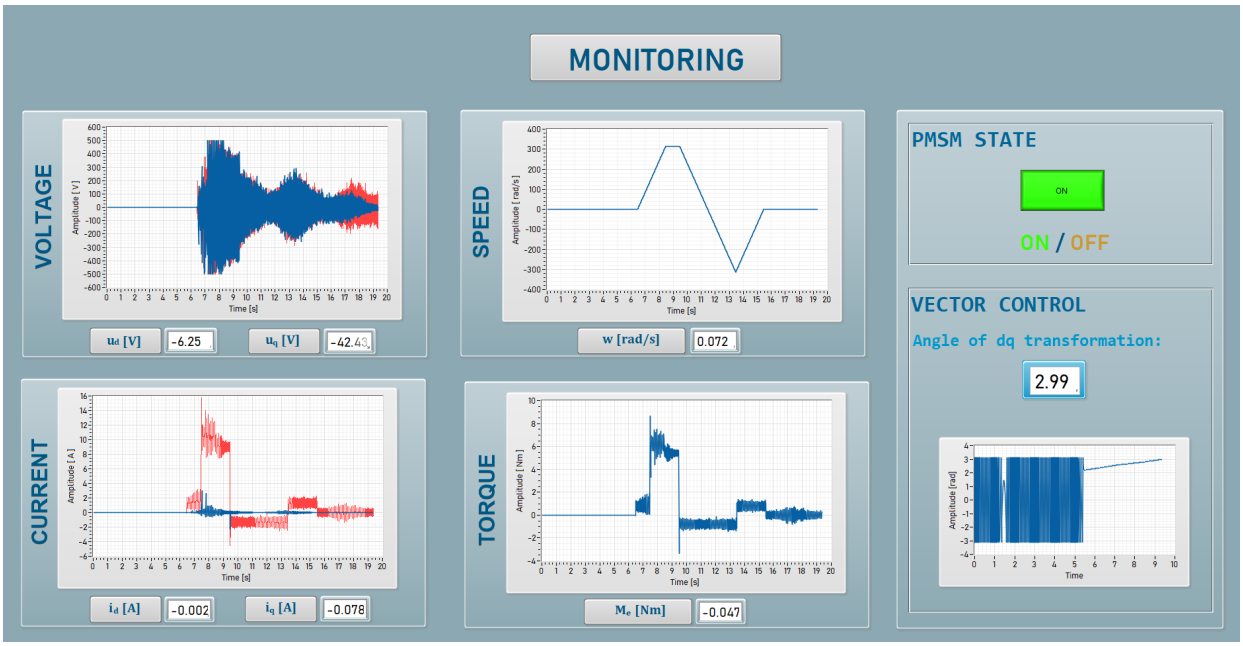
SL. 3.13: *Simulacijski model PMSM-a implementiran u OPAL-RT stvarnovremenski simulator*

Osim OPAL-RT blokova za slanje i primanje signala, blokova za skaliranje, u modelu na slici 3.13 dodani su također blokovi za transformaciju iz $\alpha\beta$ sustava u dq sustav i obrnuto. Kut transformacije izračunava se u modelu motora iz brzine vrtnje i šalje u regulator kao što je spomenuto u prethodnom poglavlju. Model motora prima iz MicroLabBox simulatora signale napona u $\alpha\beta$ sustavu, moment tereta i *reset* varijablu koja ima isti zadatak kao i kod regulatora, a to je resetirati sve integratore. Vrijednost primljenog analognog signala, koji predstavlja *reset* varijablu, je 5 V te kako bi dobili jedinicu koja će ispunjavati svoj zadatak prije ulaska u blok "PMSM" upotrijebljen je logički blok *Compare to Constant*. Kao što samo ime bloka kaže, on primljeni signal uspoređuje sa zadanom konstantom i ako je zadani uvjet točan izlaz iz bloka je 1, dok je u suprotnom 0. Dakle, kada u *Chart* bloku unutar modela regulatora varijabla *reset* poprimi vrijednost 1, u simulator dolazi signal od 5 V koji nakon usporedbe unutar bloka *Compare to Constant* daje 1 te se istovremeno resetiraju svi integratori i u motoru. Ovaj dio je vrlo važan za cijeli proces kosimulacije. Cijeli proces resetiranja može se poistovjetiti sa procesom razmagnetiziranja u stvarnim sustavima. Signali koji se šalju prema drugom simulatoru jesu struje u $\alpha\beta$ sustavu, brzina vrtnje i kut transformacije. Učitavanje Simulink modela odrađeno je koristeći se programom RT-LAB što je objašnjeno u nastavku.

3.2.3 RT-LAB

RT-LAB omogućuje Simulink modelima interakciju sa stvarnim svijetom u stvarnom vremenu njihovom implementacijom u OPAL-RT stvarnovremenske simulatore. RT-LAB-a omogućuje bolju kontrolu, vizualizaciju, pristup i prilagodbu simulacijskim modelima. Kako bi prethodno modificirani model motora bio implementiran u regulator potrebno je spomenuti model prevesti u odgovarajući programski jezik što se postže naredbom *build*, zatim se taj isti model učita na simulator (*load*) te se pokrene simulacija (*execute*). Vizualizacije se ostvaruje putem HMI (eng. *human machine interface*) sučelja čime se omogućuje praćenje i kontrola podataka. HMI sučelje se uređuje preko LabVIEW programa. HMI sučelje, izrađeno za potrebe ovog rada, prikazano je na slici 3.14.

Moguće je pratiti valne oblike napona, struje, momenta motora i brzine vrtnje motora. Osim nadgledanja ostvareno je i upravljanje motorom u pogledu paljenja i gašenja motora. To je omogućeno koristeći tipku ON/OFF koja povezana sa blokom "enable1" koji u obliku *Constant* bloka se množi sa ulaznim i izlaznim signalima motora. Inicijalno stanje motora je "OFF" te ja



SL. 3.14: *HMI sučelje u LabVIEW-u*

tako inicijalna vrijednost bloka "enable1" 0 što za rezultat ima to da su vrijednosti svih veličina u motoru također nula. Pritiskom na tipku se prelazi u stanje "ON" čime se vrijednost unutar bloka mijenja u 1 te se osigurava nesmetan prolazak signala odnosno motor je pokrenut.

Sve navede postavke, i unutar ControlDesk-a za upravljanje MicroLabBox-om kao i ove unutar RT-LAB-a i LabVIEW-a za upravljanje OPAL-RT-om, omogućile su izvođenje kosimulacije. Dodatno, napravljena je automatizacija cijelog procesa kosimulacije kao i HMI sučelje za kontrolu nad istim procesom. U nastavku rada prezentirano je dodatno proširenje samog procesa kosimulacije uz istovremenu optimizaciju parametara vektorske regulacije.

4. OPTIMIZACIJA PARAMETARA VEKTORSKE REGULACIJE

S ciljem postizanja što boljih dinamičkih odziva PMSM-a primijenjena je metoda vektorske regulacije. Modelirani regulatori imaju zadatak smanjiti oscilacije brzine vrtnje i nadvišenja momenta motora. Kako bi poboljšali performanse regulatora i ispunili prethodno navedene zahtjeve potrebno je pronaći optimalne parametre regulatora. Budući da je regulator definiran velikim brojem parametara pristupa se optimizacijskoj metodi rješavanja problema.

Optimizacija je pronalaženje rješenja nekog problema u obliku ekstrema funkcije (minimuma ili maksimuma). S obzirom na to rješava li optimizacija jedan ili više problema razlikuju se jednociljna i višeciljna optimizacija [25].

Problem jednociljne optimizacije definira se kao minimiziranje (ili maksimiziranje) funkcije $f(\boldsymbol{x})$ uz uvjet da su $g_i(\boldsymbol{x}) \geqslant 0$, gdje je $i = \{1, ..., m\}$, i $h_j(\boldsymbol{x}) = 0$, gdje je $j = \{1, ..., m\}$, za $\boldsymbol{x} \in \Omega$. Cilj je pronaći rješenje za koje će funkcija $f(\boldsymbol{x})$ biti što manja (ili veća), pri čemu je $\boldsymbol{x}$ n-dimenzionalni vektor varijable odluke $\boldsymbol{x} = (x_1, ..., x_n)$ iz skupa Ω koji sadrži sve moguće vrijednosti $\boldsymbol{x}$ koje zadovoljavaju postavljene uvjete. Prilikom optimizacije (minimizacije ili maksimizacije) $f(\boldsymbol{x})$ ograničenja predstavljena funkcijama $g_i(\boldsymbol{x}) \geqslant 0$ i $h_j(\boldsymbol{x}) = 0$ moraju biti zadovoljena. Broj ograničenja m ne smije biti veći od broja varijabli odluke n [25].

Problem višeciljne optimizacije definira se kao minimiziranje (ili maksimiziranje) funkcije $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_k(\boldsymbol{x}))$ uz uvjet da su $g_i(\boldsymbol{x}) \geqslant 0$, gdje je $i = \{1, ..., m\}$, i $h_j(\boldsymbol{x}) = 0$, gdje je $j = \{1, ..., m\}$, za $\boldsymbol{x} \in \Omega$. Rješenje za koje će funkcija $F(\boldsymbol{x})$ biti što manja (ili veća), pri čemu je $\boldsymbol{x}$ n-dimenzionalni vektor varijable odluke $\boldsymbol{x} = (x_1, ..., x_n)$ iz skupa Ω koji sadrži sve moguće vrijednosti $\boldsymbol{x}$ koje zadovoljavaju postavljene uvjete. Prilikom optimizacije (minimizacije ili maksimizacije) $F(\boldsymbol{x})$ ograničenja predstavljena funkcijama $g_i(\boldsymbol{x}) \geqslant 0$ i $h_j(\boldsymbol{x}) = 0$ moraju biti zadovoljena. Broj ograničenja m ne smije biti veći od broja varijabli odluke n [25].

U ovom radu je postavljen i jednociljni i višeciljni optimizacijski problem od kojih je svaki definiran u nastavku.

4.1. Jednociljni optimizacijski problem

Pilikom zadavanja jednociljnog optimizacijskog problema za funkciju cilja uzeto je regulacijsko odstupanje brzine vrtnje kako bi se smanjilo odstupanje stvarne od referentne brzine vrtnje odnosno kako bi stvarna brzina što bolje slijedila referentnu brzinu. Funkcija je definirana prema kriteriju integralne kvadratne pogreške (eng. *integral square error*):

$$f_\omega(\boldsymbol{\Theta}) = \int_0^\infty (\omega_r^* - \omega_r(\boldsymbol{\Theta}))^2 dt \tag{4-49}$$

gdje je $\boldsymbol{\Theta}$ vektor varijabli odluke, ω_r^* referentna veličina brzine vrtnje dok $\omega_r(\boldsymbol{\Theta})$ predstavlja mjerenu veličinu brzine vrtnje koja dolazi iz motora. Vektor varijabli odluke čine parametri regulatora koji se žele optimirati. Ukupno 9 parametara čini vektor varijabli odluke $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} = \begin{bmatrix} K_{id} & K_{pd} & K_{id_p} & K_{iq} & K_{pq} & K_{iq_p} & K_{i\omega} & K_{p\omega} & K_{i\omega_p} \end{bmatrix} \tag{4-50}$$

Parametri sa indeksom $K_{x_\mathbf{p}}$ odnose se na parametre regulatora koji se nalaze u povratnoj vezi, a tu su smješteni zbog primjene *anti-windup*a. Iako su oni po definiciji tehničkog i simetričnog optimuma jednaki po iznosu integralnim članovima pripadajućeg regulatora, prilikom optimizacije tretiraju se zasebno te tako mogu poprimiti različite vrijednosti.

Uz definiranu funkciju cilja i vektor varijabli odluke moguće je postaviti minizacijski problem:

$$\begin{aligned} &f_\omega(\boldsymbol{\Theta}) \rightarrow min \\ &\boldsymbol{\Theta}_{opt} = \underset{\boldsymbol{\Theta} \in S \subset \mathbb{R}}{argmin} f_\omega(\boldsymbol{\Theta}) \end{aligned} \tag{4-51}$$

Skupom S se definiraju ekspilicitna ograničenja nad varijablama odluke u obliku intervala $S \in [\mathbf{\Theta}_{dg}, \mathbf{\Theta}_{gg}]$ gdje $\mathbf{\Theta}_{dg}$ predstavlja vektor donje granice ograničenja, $\mathbf{\Theta}_{gg}$ vektor gornje granice ograničenja varijabli odluke.

Rješavanjem jednociljnog optimizacijskog problema dobije se jedinstveno optimalno rješenje u obliku vekotra $\mathbf{\Theta}_{opt}$ za koje se smatra da postiže globalni optimum.

4.2. Višeciljni optimizacijski problem

Drugi pristup bio je postavljanje višeciljnog optimizacijskog problema. Istovremena optimizacija, odnosno u ovom slučaju minimizacija, nekoliko funkcija cilja kao i dobivanje ne jednog jedinstvenog rješenja nego njih nekoliko ono je što razlikuje višeciljnu optimizaciju od jednociljne. Tako je uz već postojeću funkciju cilja, prikazanu izrazom (4-49), definirana još jedna funkcija cilja. Funkcija je definirana na isti način kao i prethodna, a to je prema kriteriju integralne kvadratne pogreške (eng. *integral square error*):

$$f_M(\mathbf{\Theta}) = \int_0^\infty (i_q^* - i_q(\mathbf{\Theta}))^2 dt \tag{4-52}$$

gdje je i_q^* referentna vrijednost struje statora u q osi dok je $i_q(\mathbf{\Theta})$ mjerena vrijednost struje statora u q osi koja dolazi iz motora. Ovom funkcijom cilja želi se smanjiti nadvišenje momenta motora. Vektor varijable odluke $\mathbf{\Theta}$ predstavljen je izrazom (4-50) odnosno jednak je kao i u prethodno postavljenom optimizacijskom problemu. Isto vrijedi i za postavljena ograničenja. Definiranjem pripadajućih funkcija cilja moguće je zapisati višeciljnu funkciju:

$$\mathbf{F}(\Theta) = (f_\omega(\mathbf{\Theta}), f_M(\mathbf{\Theta})) \tag{4-53}$$

Višeciljni minimizacijski problem nakon definiranja funkcije cilja glasi:

$$\begin{aligned} &\mathbf{F}(\mathbf{\Theta}) \rightarrow min \\ &\mathbf{\Theta}_{opt2} = \underset{\Theta \in S \subset \mathbb{R}}{argmin} \mathbf{F}(\mathbf{\Theta}) \end{aligned} \tag{4-54}$$

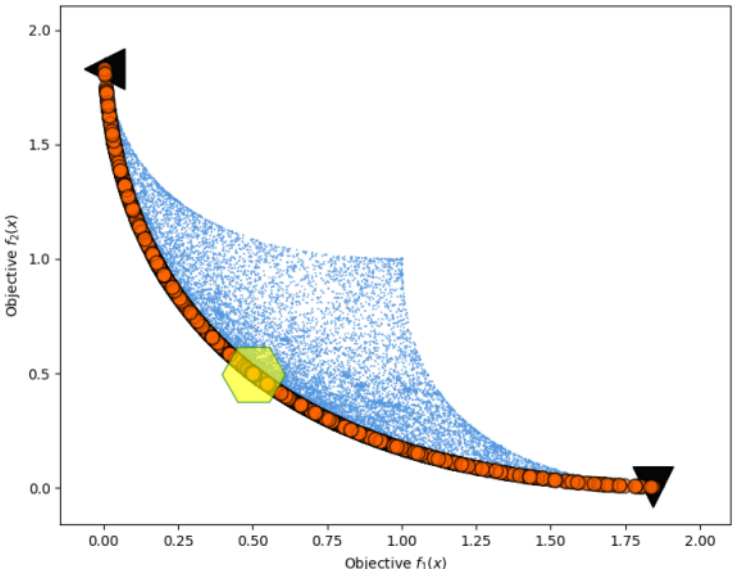
U ovom slučaju optimalni vektor varijable odluke $\mathbf{\Theta}_{opt2}$ ne predstavlja jedno nego set rješenja. Primjenom Paretove teorije optimalnosti (eng. *Pareto Optimality Theory*) odabire se set rješenja. U daljnjem tekstu objašnjeno je što je to Pareto optimum te kako se njegovom primjenom dolazi do rješenja problema.

4.2.1 Pareto optimum

Kod višeciljnih funkcija pojam "optimum" se mijenja jer je cilj pronaći dobre kompromise (eng. *trade-offs*), umjesto pojedinačnog rješenja kao u jednociljnoj optimizaciji. Najčešće prihvaćeni pojam "optimum" temelji se na prijedlogu koji je prvobitno dao Francis Ysidro Edgeworth, a kasnije ga je generalizirao Vilfredo Pareto. Iako neki autori nazivaju ovaj pojam Edgeworth-Pareto optimum najčešće prihvaćeni izraz je Pareto optimum [25].

Pareto optimum jedan je od ključnih koncepta u polju optimizacije. Ideja Pareto optimuma jeste generiranje Pareto optimizacijskog seta. Pareto optimizacijski set je stvoren kako bi vizualizirao kompromise između ciljeva kroz set odabranih rješenja. Pareto optimalno rješenje definira se kao skup rješenja u prostoru ciljeva koji definira granicu iza koje nijedan od ciljeva ne može biti poboljšan bez žrtvovanja barem jednog od preostalih ciljeva [26]. Na slici 4.15 prikazan je grafički set Pareto optimalnih rješenja (eng. *Pareto front*).

Narančastim su označena Pareto optimalna rješenja koja čine Pareto front, dok žuti šesterokut prikazuje jedno od mogućih rješenja. Ono što karakterizira višeciljne funkcije jeste



SL. 4.15: *Pareto front*

da ne postoji jedinstveno rješenje te da nema matematičkih alata koji bi dokazali zašto je jedno rješenje bolje od drugog. Pareto optimum ne daje konkretno rješenje nego sužava izbor nakon čega je odabir rješenja odluka donesena na osnovu intuitivne ljudske procjene. U ovom radu donošenje odluke je "prepušteno" MIDACO optimizatoru čiji je princip rada pojašnjen u nastavku.

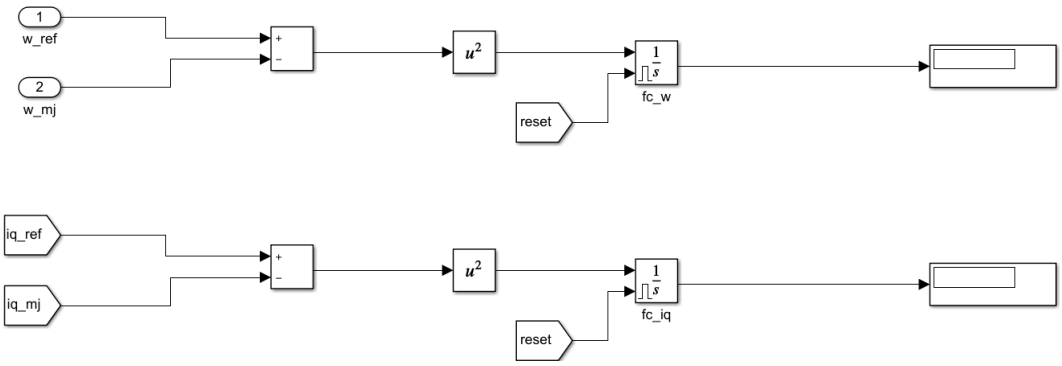
4.3. MIDACO optimizator

Za optimizaciju parametara vektorske regulacije PMSM-a korišten je MIDACO (eng. *Mixed Integer Distributed Ant Colony Optimization*) optimizator. To je je softverski alat utemeljen na metaheurističkom algoritmu distribuirane kolonije mrava za rješavanje problema optimizacije.

SL. 4.16: *Grafički prikaz optimizacijsko-kosimulacijskog procesa*

Optimizacija se vrši u kosimulaciji sa dSpace MicroLabBox stvarnovremenskim simulatorom. Optimizator je modul u Python programskom jeziku implementiran u ControlDesk IDE i generira novu populaciju varijabli odluke odnosno parametre. Regulator, čija je model implementiran na dSpace-ovom MicroLabBox-u, prima te parametre te u kosimulaciji sa OPAL-RT simulatorom na kojem je implementiran PMSM izvrti simulaciju, zatim po završetku simulacije izračunatu funkciju cilja unutar dSpace MicroLabBox-a šalje natrag optimizatoru. Cijeli proces prikazan je slikom 4.16. Spomenute funkcije cilja (definirane jednadžbama (4-49) i (4-52))

izračunate su unutar bloka "Funkcija cilja", prikazanog na slici 4.17, koji je dio simulacijskog modela regulatora implementiranog u MicroLabBox simulator.



SL. 4.17: *Izračun funkcija cilja*

Optimizator generira novi set parametara prema primljenoj vrijednosti funkcije/funkcija cilja. Taj proces se iterativno ponavlja sve dok se ne zadovolji kriterij koji ga zaustavlja [16, 27].

4.3.1 Metaheuristički algoritam kolonije mrava

Kao što je u prethodnom poglavlju rečeno MIDACO optimizator utemeljen je na metaheurističkom algoritmu distribuirane kolonije mrava. U nastavku će biti ukratko objašnjen pojam metaheuristike i samog algoritma radi boljeg razumijevanja načina rada optimizatora i njegovog odabira parametara.

Optimizacijske metode rješavanja problema mogu se podijeliti na one koje probleme rješavaju služeći se matematičkim alatima i one koje koriste metaheurističke algoritme. Metaheuristika je metoda koja rješava složene optimizacijske probleme u kratkom vremenskom roku. Dvije glavne komponente svakog metaheurističkog algoritma su intenzifikacija i diverzifikacija. Diverzifikacija znači generiranje raznolikih rješenja kako bi se globalno pretražio prostor varijabli odluke, dok intenzifikacija znači fokusiranje na lokalnu pretragu znajući da se dobro rješenje nalazi u tom području. Kako bi nađeno rješenje konvergiralo u optimum ključ je pronaći ravnotežu između diverzifikacije i intezifikacije. Dobra kombinacija ovih dviju glavnih komponenti obično će osigurati ostvarenje globalne optimalnosti [28]. Metaheurističke metoda obuhvaća različite algoritmime inspirirane prirodom. Postoji više vrsta podjela metaheurističkih algoritama s obzirom na različite karakteristike. Jedna od njih dijeli metaheurističke algoritme na [28]:

- Evolucijski algoritmi
- Algoritmi temeljeni na fizici
- Algoritam roja čestica

U ovom radu korišteni MIDACO optimizator utemeljen je na algoritmu kolonije mrava koji pripada skupini algoritama roja čestica. Optimizacija putem kolonije mrava (eng. Ant Colony Optimization, ACO) je stohastička metaheuristička metoda za rješavanje kombinatornih optimizacijskih problema koja koristi umjetne mrave. Cilj ACO-a je pronalaženje kraćih ruta od njihovih gnijezda do izvora hrane. Mravi ostavljaju za sobom kemijsku tvar zvanu feromon koja im omogućava komunikaciju s drugim mravima. Kada mrav pomakne, iza sebe ostavlja dosljednu količinu feromona koju drugi mravi mogu otkriti. Svaki mrav putuje na svoj jedinstven način, ali kad naiđe na trag feromona, mora odlučiti hoće li ga slijediti ili ne. Ako jedan mrav

prati trag, njegov vlastiti feromon ga ojačava, a povećanje feromona povećava vjerojatnost da će sljedeći mrav slijediti taj trag. Kao rezultat toga, što više mrava ide istom stazom, to postaje privlačnije za sljedeće mrave. Osim toga, mrav koji odabere kraći put do izvora hrane će se brže vratiti natrag do gnijezda [4].

4.3.2 Postavke MIDACO optimizatora za rješavanje optimizacije parametara vektorske regulacije

Prethodno je dan kratak pregled algoritma na kojem optimizator temelji svoj rad. Na početku ovog potpoglavlja opisan je način izvedbe optimizacije. Postavljanje idućih parametara vrši se u unutar editora ControlDesk-a gdje je u obliku skripte implementiran MIDACO optimizator. Sljedeće parametri postavljaju se redamo redom:

1. Funkcije cilja - MIDACO optimizator funkciju cilja kao i funkcije ograničenja smatra *black-box* funkcijama. Taj koncept omogućava korisniku potpunu slobodu prilikom računanja i definiranja funkcije cilja s obzirom da optimizatoru nisu bitne karakteristike funkcije i njen matematički zapis nego samo njena vrijednost.

2. Dimenzija problema - broj funkcija ciljeva (o) i varijabli odluke (n)

3. Donje (xl) i gornje (xu) granice ograničenja

4. Početna točka (x) - vrijednost parametara koje će biti uzete pri pokretanju optimizacije (najčešće se uzimaju vrijednosti donje granice ograničenja)

5. Kriterij zaustavljanja ($maxeval$) - kriterij čijim će zadovoljavanjem prekinuti optimizacijski proces (dosegnut broj iteracija, vrijednost funkcije cilja u zadnjih nekoliko iteracija jednaka i sl.)

Navedene postavke obavezne su kako bi se optimizacijski proces mogao izvršiti. Od dodatnih postavki postoje nekoliko parametara koji postavljaju dodatne uvjete za zaustavljanje procesa, parametri za fokus traženja rješenja oko trenutno najboljeg rješenja, parametri za postavke višeciljne optimizacije i slično. U ovom radu od svih ponuđenih su se koristila dva parametra:

- ANTS - ovaj parametar omogućava korisniku da fiksira broj mrava (iteracija) koje MIDACO generira unutar jedne generacije

- KERNEL - parametar koji ovisno o prirodi problema može pomoći da rješenje konvergira i dosegne globalni optimum (uvijek se koristi u kombinaciji sa parametrom $ANTS$)

Kako je prethodno navedeno prvi korak je predavanje funkcije cilja. U ovom radu funkcija odnosno funkcije cilja predane su u obliku Python funkcije *ConfigureMeasurement* čija je povratna vrijednost bila vrijednost funkcija cilja. Naime, predana funkcija imala je za argument vektor varijabli odluke te je na taj način nakon svake iteracije MIDACO poslao novi set parametara funkciji *ConfigureMeasurement* koja bi odradila simulaciju sa danim parametrima te po završetku poslala MIDACO-u novu vrijednost funkcije cilja dok bi istovremeno MIDACO poslao novi set parametara. Postupak se ponavljao sve dok zaustavni kriterij nije bio zadovoljen. Definicija proširene Python funkcije *ConfigureMeasurement* prikazana je u nastavku:

Primjer koda 4: *Tijelo funkcije ConfigureMeasurement*

```
def ConfigureMeasurement(self,x):

##Zapisivanje vrijednosti optimalnih parametara vektorske regulacije
        Ki_id.ValueConverted = x[0]
        Kp_id.ValueConverted = x[1]
        Ki_id_p.ValueConverted = x[2]
        Ki_iq.ValueConverted = x[3]
        Kp_iq.ValueConverted = x[4]
        Ki_iq_p.ValueConverted = x[5]
        Ki_w.ValueConverted = x[6]
        Kp_w.ValueConverted = x[7]
        Ki_w_p.ValueConverted = x[8]

##Pokretanje simulacije
        while enable.ValueConverted == 0:
                if self.MeasurementDataManagement.IsMeasuring == True:
                        Wait(0.3)
                        enable.ValueConverted = 1

        while enable.ValueConverted == 1:
                if disable.ValueConverted > 0.5:
                        fc_w = f_cilja.ValueConverted
                        fc_M = f_cilja_M.ValueConverted
                        enable.ValueConverted = 0

return fc_w, fc_M
```

Dok je predavanje funkcije cilja optimizatoru napravljeno na sljedeći način (u primjeru je pokazano predavanje funkcije cilja kod višeciljne optimizacije):

Primjer koda 5: *Predavanje funkcije cilja optimizatoru*

```
def problem_function(x):
        fc_w, fc_iq = a.ConfigureMeasurement(x) ##Predavanje vrijednosti
            ↪ funkcijama cilja
        f = [0.0]*2 # Vektor za spremanje funkcija cilja u F(X)
        g = [0.0]*1
        f[0] = fc_w
        f[1] = fc_iq
        print(f)
return f, g
```

Donje (xl) i gornje(xu) granice prostora pretrage definirane su kao $\pm 90\%$ vrijednosti varijabli po tehničkom i simetričnom optimumu. Početna točka (x) iz koje je započeo postupak optimizacije je mijenjana. Prvi set parametara iz kojih je započela optimizacija bile su vrijednosti koje varijable poprimaju po tehničkom i simetričnom optimumu. Drugi set parametara generiran je nasumičnim odabirom unutar definiranih granica.

Primjer koda 6: *Prikaz vrijednosti parametara ograničenja i početne točke*

```
problem['xl'] = [ 1625*0.1, 9*0.1, 1625*0.1, 1625*0.1, 17*0.1, 1625*0.1,
    ↪ 625*0.1, 2.5*0.1, 625*0.1 ]
problem['xu'] = [ 1625*1.9, 9*1.9, 1625*1.9, 1625*1.9, 17*1.9, 1625*1.9,
    ↪ 625*1.9, 2.5*1.9, 625*1.9 ]
y = [0.0]*2
y[0] = [1625, 9, 1625, 1625, 17, 1625, 625, 2.5, 625] #Tehnicki i smietricni
    ↪ optimum
a1 = problem['xl']
a2 = problem['xu']
y[1] = rndm = [] #Nasumicni brojevi
for i,j in zip(a1, a2):
        rndm.append(i+random.random()*(j-i))
k = [0,1] #Automatizacija izmjene pocetnih parametara
for i in k:
        problem['x'] = y[i]
```

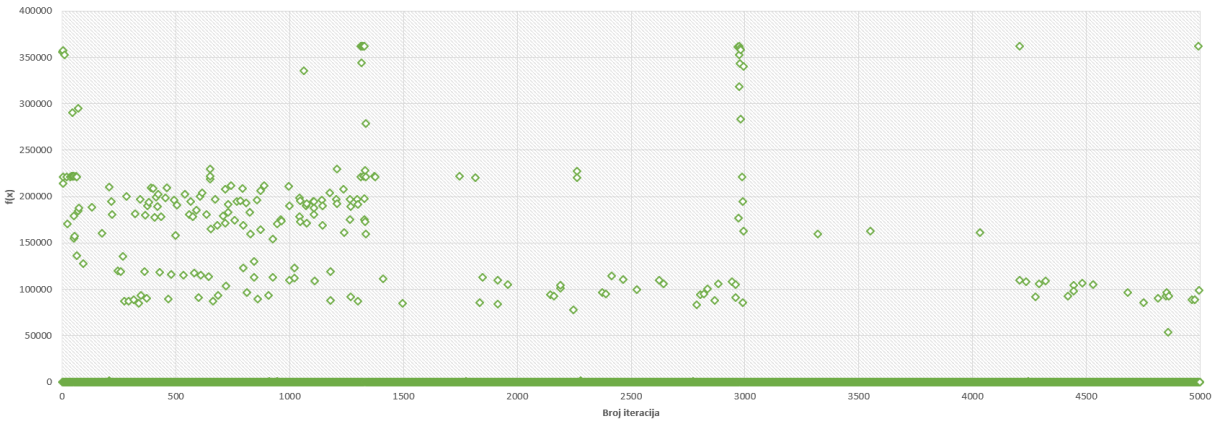
Vrijednosti ostalih postavljenih parametara prikazani su tablicom 4.1.

R.br.	Parametar	Vrijednost
1.	o	1(2)
2.	n	9
3.	m	0
4.	*maxeval*	3200
5.	*ANTS*	50
6.	*KERNEL*	10

Tab. 4.1: *Parametri MIDACO optimizatora*

Parametar pod rednim brojem 1 odnosi se na broj funkcija cilja pa s obzirom da je izvršena jednociljna i višeciljna optimizacija vrijednost tog parametra se mijenjala. Vrijednosti ostalih parametara su ostale isti tijekom izvršavanja obje optimizacije. Broj varijabli odluke (n) ima vrijednost 9 što dokazuje vektor varijabli odluka predstavljen izrazom (4-50). Iznos parametra m ima vrijednost nula s obzirom na to da ne postoje funkcije ograničenja, nego su ograničenja prostora pretrage postavljena putem definiranja gornjih i donjih granica. Četvrti po redu parametar *maxeval* koji definira broj iteracija postavljen je na 3200. Prilikom postavljanja

ovog parametra pristupilo se intuitivno pa je tako prva postavljena vrijednost bila 5000. Slika 4.18 prikazuje rezultate jednociljne optimizacije.



SL. 4.18: *Grafički prikaz vrijednosti funkcija cilja kroz 5000 iteracija*

Na njoj je vidljivo kako je optimizator prvo razbacao "mrave" u nekih prvih 1500 iteracija te da nakon toga sve više sužava prostor u kom daje rješenja da bi nakon nekih 3200 iteracija davao gotovo jednake vrijednosti. Time je pala odluka da se broj iteracija bude 3200. Vrijednost parametara 5. i 6. odabrana je na osnovu procjene s obzirom na vrstu problema iz uobičajenih kombinacija ova dva parametra.

Svim prethodno navedenim postavkama omogućen je proces optimizacije. Rješenja su predstavljena od strane MIDACO optimizatora u obliku tekstualnih datoteka. Dok kod jednociljne optimizacije uvijek postoji jedno jedinstveno rješenje, kod višeciljne optimizacije postoji set rješenja kao što je spomenuto u poglavlju 4.2.1. MIDACO optimizator slijedi Pareto teoriju određivanja seta rješenja, a odluku o najboljem rješenje donosi na način da odabrano rješenje predstavlja ono koje je najuravnoteženije među svim ciljevima.

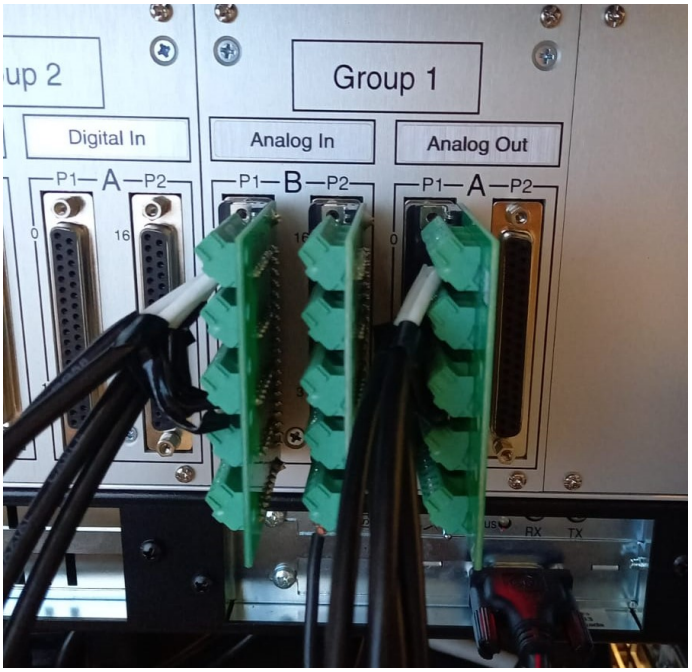
5. REZULTATI OPTIMIZACIJSKOG PROCESA

Glavni zadatak rada bio je dobiti optimalne parametre regulatora za vektorsko upravljanje PMSM-om, odnosno projektirati regulator za zadani motor koji će učiniti taj motor još učinkovitijim nego što je on sam bez regulatora.

Tim rečeno pristupilo se optimizacijskoj proceduri korištenjem MIDACO optimizatora. Optimizacija se vršila kao dio HIL kosimulacije, što je jedna od *state-of-the-art* odnosno najinovativnijih metoda koje se primjenjuju u današnjoj automobilskoj industriji.

Stvarnovremenska HIL kosimulacija odvijala se između dSpace-ovog MicroLabBox-a i OPAL-RT stvarnovremenskih simulatora. Prvi korak implementacije simulacijskih modela u stvarnovremenske simulatore, gdje je model regulatora implementiran u MicroLabBox, a model PMSM-a implementiran u OP5600 stvarnovremenski simulator, odrađen je u skladu s postupkom objašnjenim unutar poglavlja 3. Naime, riječ je o kombinaciji HIL i RCP metode čime se objašnjava korištenje simulacijskog modela ne samo PMSM-a nego i pripadajućeg regulatora, za razliku od klasične HIL metode gdje se koristi stvarni (fizički) regulator.

Implementacijom modela unutar stvarnovremenskih simulatora bilo je potrebno fizički povezati iste. Budući da su priključci na simulatorima različiti po vrsti (MicroLabBox za analogne signale koristi BNC priključke, a OP5600 DB37 priključke) upotrijebili su se BNC-BNC kablovi koji su se podijelili te su tako nastali kabeli koji s jedne strane imaju BNC priključak, dok su s druge strane izložene žice. Na određene utore DB37 priključka prema tablici iz OP5600 priručnika spojene su izložene žice. Slika 5.19 prikazuje ulazni i izlazni analogni priključak na OPAL-ovom simulatoru.



SL. 5.19: *Ulazni i izlazni priključci OP5600 simulatora*

Na ulazni priključak spojena su četiri signala koje MicroLabBox, odnosno regulator, šalje na OPAL-ov simulator odnosno PMSM:

- d i q komponente napona
- moment tereta
- *reset*

Na izlazni priključak spojena su četiri signala koje OPAL šalje MicroLabBox-u:

- d i q komponente struje statora
- brzinu vrtnje motora
- kut transformacije

Kako bi se dobili što bolji rezultati optimizacijski proces je proveden ukupno četiri puta. Redoslijed radnji za pokretanje eksperimenta tekao je na sljedeći način:

1. Pokretanje modela PMSM-a na OPAL-RT stvarnovremenskom simulatoru slijedom naredbi *build, load, execute* unutar RT-LAB-a
2. Paljenje motora pritiskom na tipku ON/OFF u HMI sučelju unutar LabVIEW-a
3. Pokretanje optimizacijske skripte unutar Python editora ControlDesk-a tipkom *Run*

Pristupilo se rješavanju jednociljnog i višeciljnog problema sa dva seta početnih parametara odnosno početnih točki od kojih je optimizacijski proces započinjao. Za vrijeme rješavanja jednog optimizacijskog problema izvršeno je 3200 iteracija što znači da je simulacija odnosno kosimulacija pokrenuta 3200 puta i to svaki put sa novim setom parametara regulatora prije nego li je optimizator izbacio najbolje rješenje. Nakon svake iteracije unutar ControlDesk sučelja *Interpreter* ispisivane su vrijednosti parametara dobivenih optimizacijom s kojima će se izvrtjeti zadana simulacija.

Zahvaljujući HMI sučelju bilo je moguće kontinuirano pratiti kako pojedini skup parametara utječe na dinamičke odzive motora. Osim praćenja dinamičkih stanja motora, HMI sučelje osposobljeno je za paljenje motora. Praćenje je omogućeno putem grafova i ispisa trenutnih numeričkih vrijednosti veličina. Prikazivane veličine napona, struje, brzine i momenta motora.

Slika 5.20 prikazuje laboratorijski postav koji se sastojao od jednog računala (na slici lijevo) sa kojeg se pokretao optimizacijski proces te upravljalo MicroLabBox simulatorom odnosno simulacijskim modelom regulatora, te drugogog računala (na slici desno) na kojem se upravljalo OPAL-RT simulatorom i simulacijskim modelom motora te putem HMI sučelja pratila trenutna stanja motora.

SL. 5.20: *Laboratorijski postav*

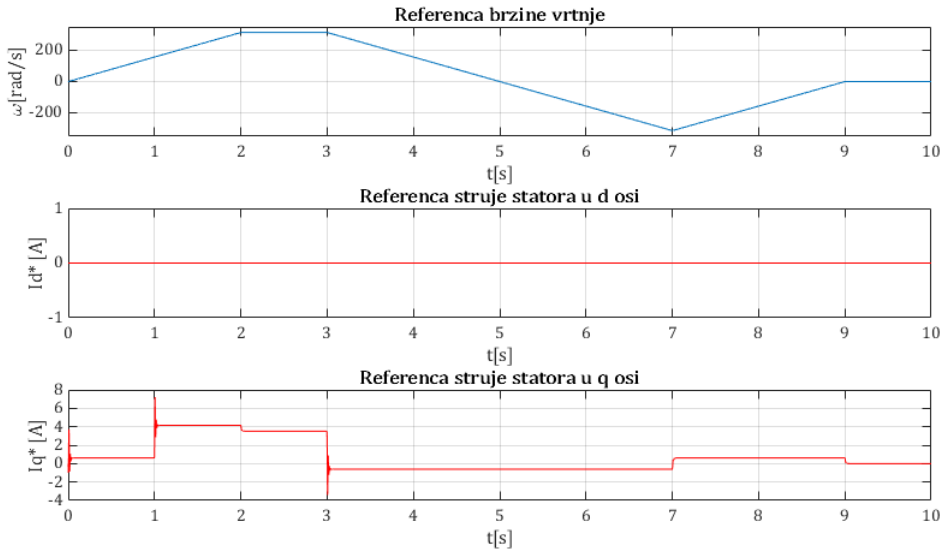
Nazivne podatke sinkronog motora s permanentnim magnetima korištenog u simulaciji prikazuje tablica 5.2.

R_s	3.25 Ω	U_n	400 V
L_d	18 mH	M_n	5.4 Nm
L_q	34 mH	n_n	3000 o/min
Ψ_m	0.341 Wb	P_n	1.7 kW
p	3	J	0.005 kgm^2

Tab. 5.2: *Nazivni podaci motora*

Slika 5.21 prikazuje odzive referentnih veličina unutar eksperimenta. Pozivajući se na njih objasnit će se tijek eksperimenta odnosno sva stanja i njihov slijed koji je prethodno zadan blokom *Chart* čije se objašnjenje nalazi u poglavlju 3.
Prateći linearnu promjenu referentne brzine vrtnje mogu se izdvojiti sljedeća stanja:

- Zalet motora - traje 2 sekunde
- Stacionarno stanje - motor se vrti nazivnom brzinom 1 sekundu
- Reverziranje - traje od 3. do 7. sekunde
- Zaustavljanje započinje u 7. i završava u 9. sekundi
- Zaustavljen motor u intervalu od 9. do 10. sekunde



SL. 5.21: *Referentne vrijednosti unutar eksperimenta*

Referentne vrijednost struje statora u *d* osi, koja je direktno proporcionalna momentu motora, oslikava promjene momenta motora odnosno:

- Terećenje - nastupa za vrijeme zaleta u 1. sekundi što uzrokuje nadvišenje struje
- Rasterećenje - nastupa u 3. sekundi i uzrokuje propad struje

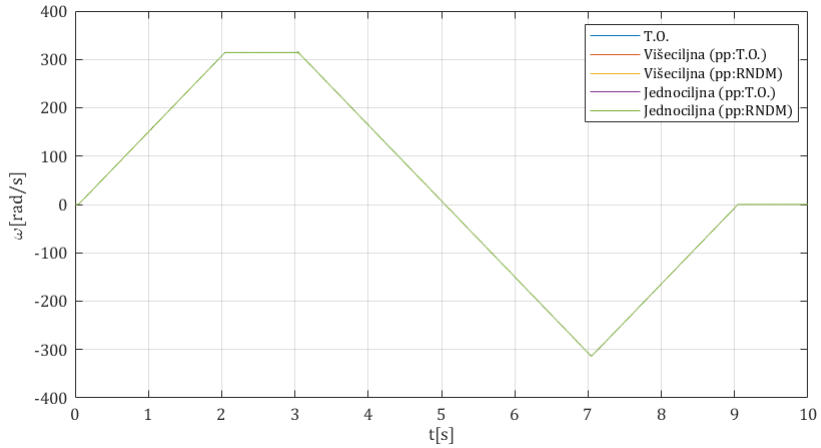
U nastavku su prikazani odzivi brzine vrtnje, struje i momenta PMSM-a nakon implementacije optimiranih parametara regulatora. Tablica 5.3 prikazuje 5 ispitnih slučajeva gdje svaki slučaj predstavlja regulator sa parametrima dobivenim kao rješenje određenog optimizacijskog problema. Slučaj 1 odnosi se na rezultate simulacije u kojoj korišteni parametri nisu rezultat optimizacijskog postupka nego su izračunati numeričkim metodama odnosno tehničkim optimumom (T.O.) i simetričnim optimumom (S.O.) kako bi se moglo ukazati na prednosti ili mane odabira parametara korištenjem metaheurističkih optimizacijskih algoritama.

	Vrsta optimizacije	Početni parametri (pp)
Slučaj 1	/	*T.O, S.O.*
Slučaj 2	*Višeciljna*	*T.O, S.O.*
Slučaj 3	*Višeciljna*	*nasumičan odabir*
Slučaj 4	*Jednociljna*	*T.O, S.O.*
Slučaj 5	*Jednociljna*	*nasumičan odabir*

Tab. 5.3: *Ispitni slučajevi*

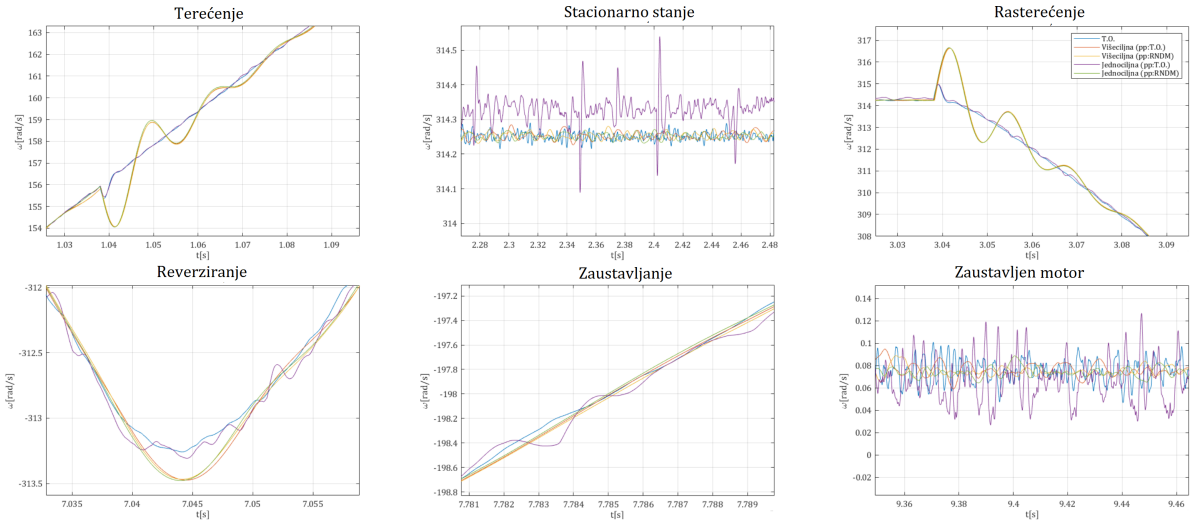
5.1. Analiza utjecaja parametara vektorske regulacije na dinamička stanja motora

Kako je jedan od glavnih ciljeva optimizacije bio smanjiti regulacijsku pogrešku brzine vrtnje kako bi ona što idealnije pratila zadanu referencu, prvo se pristupilo analizi odziva brzine vrtnje kod svih pet slučajeva. Slika 5.22 prikazuje odziv brzine vrtnje za svih 5 slučajeva. S obzirom da radi o neznatnim odstupanjima između svakog slučaja nije moguće vidjeti razlike koje postoje među signalima.



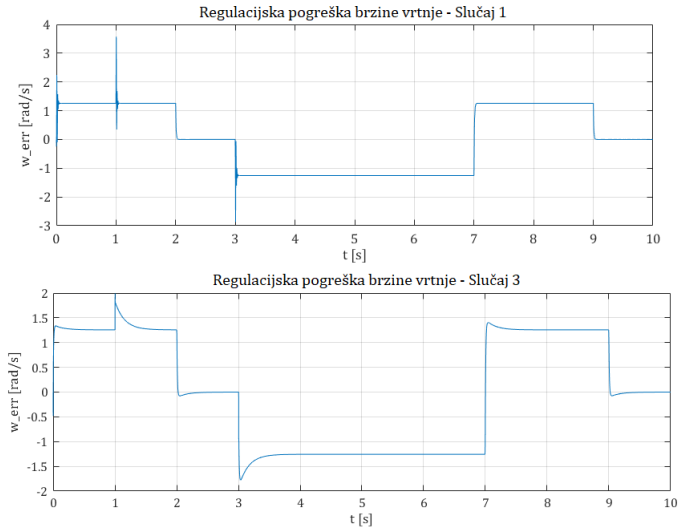
SL. 5.22: *Odzivi brzine vrtnje*

Na slici 5.23 se nalaze uvećani prikazi odziva brzine vrtnje u pojedinim stanjima motora kako bi se bolje vidjele tranzijentne promjene za svaki slučaj.



SL. 5.23: *Uvećani prikaz odziva brzine vrtnje za pojedina stanja motora*

Pozivajući se na prikazano moguće je zaključiti kako signali brzine vrtnje za *slučaj 1* i *slučaj 4* najbolje savladavaju promjene što se vidi prilikom terećenja i rasterećenja motora kao i prilikom reverziranja odnosno na prelasku iz reverziranja u zaustavljanje. Prethodno je zaključeno na osnovu toga što je prilikom prelazaka iz jednog u drugo stanje nadvišenje ili propad brzine vrtnje minimalano. Međutim, rezultati odziva brzine vrtnje za preostala tri slučaja prikazuju nešto lošije savladavanje promjene odnosno nešto veća nadvišenja i propadanja brzine vrtnje. Za vrijeme trajanja ostalih stanja motora može se primijetiti kako je kod signala za *slučaj 1* i *slučaj 4* šum zastupljen mnogo više nego kod preostala tri slučaja kada je regulator motora imao drugačije parametre. Oscilatorna priroda signala odnosno zastupljenost šuma u signalu može biti posljedica elektromagnetskih interferencija, elektromagnetske kompatibilnosti, mehaničkih vibracija, regulacije sustava. S obzirom da su šumovi manje prisutni kod signala čiji su regulatori imali različite parametre (*slučaj 2, 3* i *5*) postoji velika mogućnost da je izbor parametara regulatora odnosno poboljšan regulacijski sustav motora ono što je dovelo do smanjenja prisutnosti šuma.

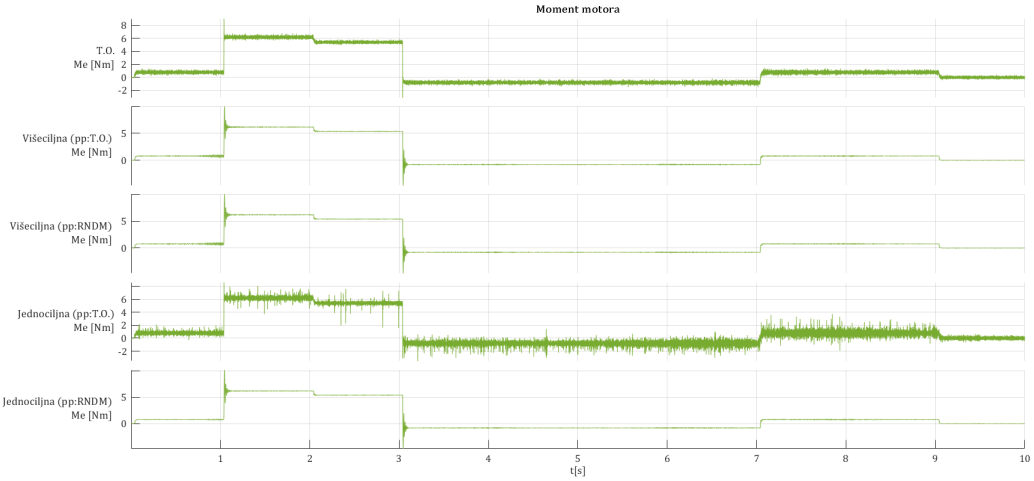


SL. 5.24: *Regulacijska pogreška brzine vrtnje za slučaj 1 i slučaj 3*

Slika 5.24 prikazuje regulacijsko odstupanje za *slučaj 1* i *slučaj 3*. Vidljivo je kako je regu-

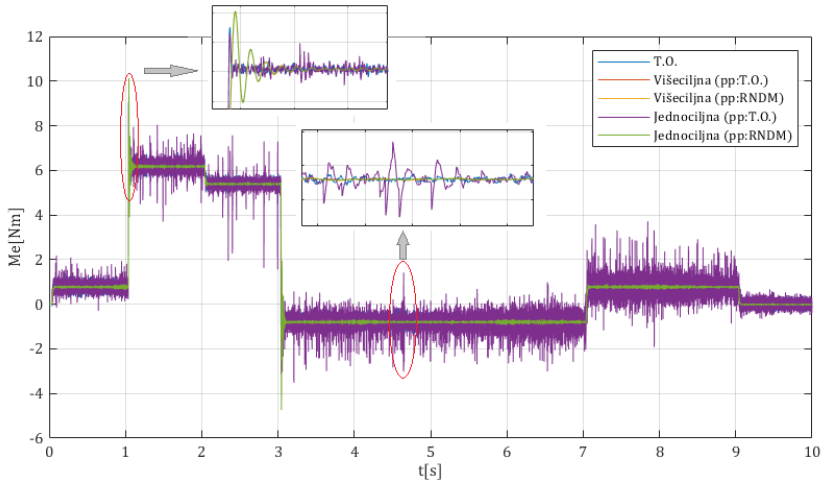
lacijsko odstupanje za *slučaj 3* puno manje što znači da je optimizacijski problem dobro riješen. Jedino negativno što se može izdvojiti jeste dulje vrijeme ustaljivanja. Dulje vrijeme ustaljivanja povezano je sa propadima i nadvišenjima odziva brzine vrtnje kao posljedica terećenja i rasterećenja motora.

Osim brzine, drugi cilj vektorske regulacije općenito pa i ovog optimizacijskog procesa bio je i smanjiti nadvišenje momenta.



SL. 5.25: *Odziv momenta motora*

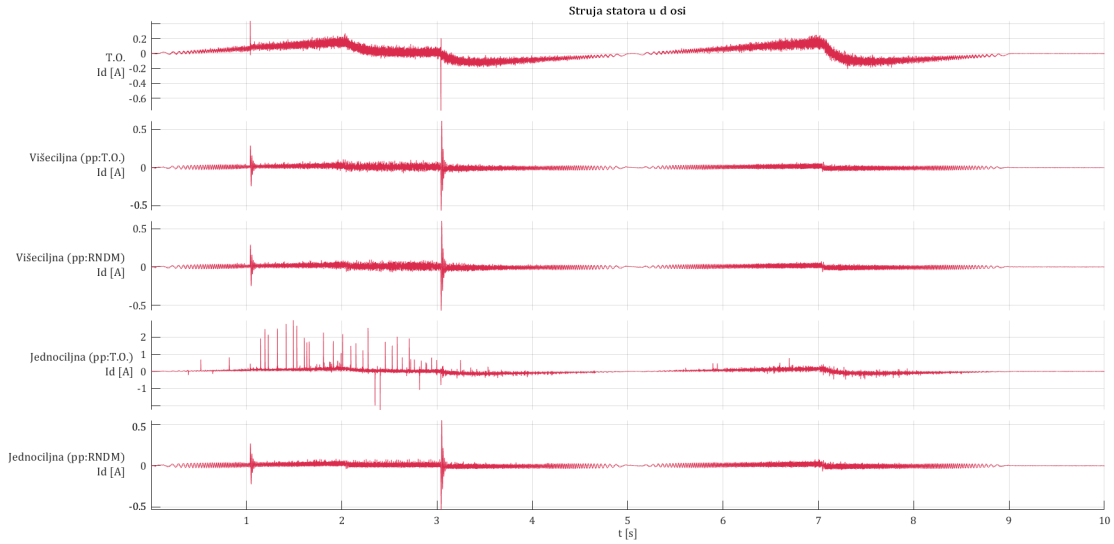
Slika 5.25 prikazuje odziv momenta za svih 5 ispitnih slučajeva. Prilikom zadavanja jednociljnog optimizacijskog problema funkcija cilja bila je smanjenje regulacijske pogreške brzine vrtnje, dok je tek kod višeciljnog optimizacijskog problema uz smanjenje regulacijske pogreške brzine vrtnje dodana funkcija cilja za smanjenje nadvišenja momenta. Uspoređujući odziv momenta motora čiji parametri regulatora nisu optimizirani (*slučaj 1*) sa ostala četiri slučaja vidljivo je kako optimizacijski postupak nije riješio problem nadvišenja momenta. Slika 5.26 prikazuje sve slučajeve na jednom mjestu kako bi se bolje pokazala razlika jednih odziva u odnosu na druge.



SL. 5.26: *Uvećani prikazi odziva momenta motora*

Iako problem nadvišenja momenta nije riješen, važno je naglasiti kako su sva nadvišenja momenta opet u dozvoljenim granicama, te da odzivi momenta za *slučaj 2, 3* i *5* prikazuje kraće trajanje prijelaznih pojava.

Kontrolom struje u d osi postiže se upravljanje magnetskim poljem. Isto tako, njenom kontrolom moguće je minimizirati gubitke bakra budući da q komponenta struje proizvodi elektromagnetski moment. Slika 5.27 prikazuje odzive struje u d osi.



SL. 5.27: *Odzivi struje statora u d osi*

Signala struje u d osi pokazuje povećanu prisutnost šuma kod svih navedenih slučajeva. Parametri dobiveni kao rješenje jednociljnog optimizacijskog problema, *slučaj 4*, daju najlošije rješenje ako se da suditi po oscilatornoj prirodi signala. Kako je ranije rečeno kontrolom struje d moguće je kontrolirati gubitke u bakru odnosno ona uzrokuje direktne gubitke u mreži. Što znači da će veća zastupljenost šuma u signalu rezultirati većim gubitcima u bakru što je jedna od negativnih karakteristika koja se u svakom slučaju želi izbjeći. Ono po čemu su svi odzivi dobiveni korištenjem regulatora sa optimiranim parametrima jeste bolje slijeđenje reference struje $i_d^* = 0$.

Radi boljeg shvaćanja koliko je šum zastupljeniji za *slučaj 1* u odnosu na *slučaj 2* izračunati su gubitci u bakru za oba slučaja čime su dobiveni rezultati pokazali kako su gubitci u bakru veći za 7 do 8 puta u *slučaju 1*.

6. ZAKLJUČAK

Postizanje visoke učinkovitosti i preciznosti kod električnih motora vrlo je zahtjevan zadatak. Dizajniranje takvih sustava je težak zadatak koji iziskuje neiscrpno provođenje testova koji ako se provode na stvarnim fizičkim sustavima iziskuju puno novca, vremena i mogu biti jako opasni. Testiranje u virtualnom svijetu, odnosno na modelima, nailazi na veću primjenu zbog mnogo prednosti kao što su ponovljivost eksperimenata, sigurnost, smanjeni troškovi, kraće trajanje razvojnog procesa. Na taj način otklanjaju sve poteškoće i negativne strane testiranja na stvarnim fizičkim modelima. Zadatak ovog rada jeste pronaći optimalne parametre regulatora. Kako bi pronašli odgovarajuće parametre koji će rezultirati što boljim performansama moramo biti u mogućnosti testirati popriličan broj kombinacija.

Tako je u radu prvo predstavljen sinkroni motor s permanentnim magnetima od njegovog opisa, matematičkog modela, preko *dq* transformacije i u konačnici dvoosnog modela koji je poslužio za modeliranje regulatora za vektorsko upravljanje koji je predmet optimizacije. Zatim su predstavljene metode testiranja nad modelima. Odabrana HIL metoda omogućava provođenje testiranja nad simuliranim hardverom, u ovom slučaju motorom. Budući da je naglasak u ovom radu stavljen na optimizaciju parametara vektorske regulacije bilo je potrebno prije svega i regulator staviti u simulacijsko okruženje. RCP metodom omogućeno je parametriranje odnosno optimizacija parametara regulatora u sklopu HIL simulacije. RCP i HIL metoda zajedno su rezultirale kosimulacijom odnosno provođenje eksperimenta isključivo na modelima bez prisustva fizičkog regulatora ili motora. Modeli su implementirani u stvarnovremenske simulatore dSpace i OPAL-RT.

Nadalje, kada su ostvareni uvjeti za kosimulaciju mogao je započeti optimizacije. Tako su nadalje definirani jednociljni i višeciljni optimizacijski problemi. Jednociljna optimizacija imala je cilj smanjiti regulacijsku pogrešku brzine vrtnje Višeciljni optimizacijski problem definirao je dvije funkcije cilja od kojih je jedna preuzeta iz jednociljnog optimizacijskog problema, dok je druga funkcija imala za cilj smanjiti nadvišenje momenta. Opisan je optimizator, koji svoj rad temelji na metaheurističkom algoritmu kolonije mrava, korišten za rješavanje optimizacijskih problema. Također je dan opis optimizacijskog procesa sa svim postavkama optimizacije.

Optimizacijski proces proveo se za 4 različita slučaja. Svaka optimizacija rezultirala je sa novim setom optimalnih parametara nakon čega je svaki set unesen u u regulator te je pokrenuta zadana simulacija. Simulacija se sastojala od zaleta, terećenja, rasterećenja, reverziranja i zaustavljanja čim je osigurano ispitivanje utjecaja parametara na sva dinamička stanja motora. Dobiveni odzivi brzine vrtnje, struje i momenta su potom analizirani za svaki od slučaja. Optimizacija je rezultirala poboljšanjima u vidu trajanja prijelazne pojave, bolje oscilatorne prirode signala, smanjene regulacijske pogreške brzine vrtnje u odnosu na dinamiku motora čiji su parametri regulatora dobiveni numeričkom metodom prema tehničkom i simetričnom optimumu. Nešto lošiji ishodi postignuti su prilikom savladavanje promjena u vidu nadvišenja i propadanja odziva momenta i brzine vrtnje. Sagledavajući generalnu sliku, u kojoj su optimirani parametri dali stabilne odzive uz više poboljšanja nego pogoršanja cijele slike dinamičkog stanja motora, može se reći kako metaheuristički algoritmi u polju parametriranja regulatora pronalaze mjesto. Daljnja istraživanja mogla bi uključivati primjenu različitih kriterija za izračun funkcije cilja čijom bi se optimizacijom možda dobili bolji rezultati ili primjenu nekih drugih metaheurističkih algoritama. Primjena metaheuristike na području parametriranja regulatora je nova te ostavlja puno prostora za daljnja istraživanja.

LITERATURA

[1] S.C. Nagaraj and B. Detrick. HIL and RCP tools for embedded controller development in hybrid vehicles. In *2009 IEEE Vehicle Power and Propulsion Conference*. IEEE, sep 2009.

[2] Ji Wu, Christian Dufour, and Linxiang Sun. Hardware-in-the-loop testing of hybrid vehicle motor drives at ford motor company. In *2010 IEEE Vehicle Power and Propulsion Conference*. IEEE, sep 2010.

[3] Junichi Kako, Soejima Shinichi, and Ohata Akira. Efficient engine development using model based development (MBD). In *2007 Chinese Control Conference*. IEEE, jul 2006.

[4] Stephen Bassi Joseph, Emmanuel Gbenga Dada, Afeez Abidemi, David Opeoluwa Oyewola, and Ban Mohammed Khammas. Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems. *Heliyon*, 8(5):e09399, may 2022.

[5] Stefan Koncar, Zeljko Lukac, Nevena Stojanovic, and Igor Kolak. Development of automotive video logger HIL device for ADAS/AD algorithms development and testing. In *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE, may 2020.

[6] Jeroen Ploeg, Falke M. Hendriks, and Niels J. Schouten. Towards nondestructive testing of pre-crash systems in a HIL setup. In *2008 IEEE Intelligent Vehicles Symposium*. IEEE, jun 2008.

[7] Veselko; Pužar Milica Mandić, Ivan; Tomljenović. *Sinkroni i asinkroni električni strojevi*. Tehničko Veleučilište u Zagrebu, 2012.

[8] Božidar Jadrić, Martin; Frančić. *Dinamika električnih strojeva*. Zagreb: Graphis, 2004.

[9] Domagoj-Krešimir Jukić. Modeliranje sinkronog motora s permanentnimmagnetima i upravljanje neizrazitim regulatorom. Master's thesis, Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, 2019.

[10] Kristian Knol. Kosimulacijska analiza rada sinkronog motora sapermanentnim magnetima i izrazenom reluktancijomu stvarnom vremenu. Master's thesis, Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, 2022.

[11] Shigeo Morimoto, Yoshinari Asano, Takashi Kosaka, and Yuji Enomoto. Recent technical trends in PMSM. In *2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA)*. IEEE, may 2014.

[12] Luka Pravica. Strukture upravljanja sinkronimmotorom s permanentnimmagnetima. Master's thesis, Sveučilište u Zagrebu, Fakultet Elektrotehnike i Računarstva, 2012.

[13] R. Krishnan. *Permanent Magnet Synchronous and Brushless DC Motor Drives*. CRC Press, dec 2017.

[14] Ivan Flegar. Uloga matematičkih modela u projektiranju električkih uređaja. *Povijest i filozofija tehnike*, 2019.

[15] Paul C. Krause. *Analysis of electric machinery and drive systems*. IEEE Press, 2002.

[16] T. Benšić. Analiza dinamičkih stanja sustava vlastite potrošnje hidroelektrane. Master's thesis, Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, 2014.

[17] Bimal K. Bose. *Modern Power Electronics and AC Drives*. Prentice Hall PTR, 2001.

[18] N Perić. *Automatsko upravljanje, predavanja*. Fakultet elektrotehnike i računarstva Zagreb, 2004.

[19] Vince Socci. Implementing a model-based design and test workflow. In *2015 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, sep 2015.

[20] Severin Sadjina, Lars Tandle Kyllingstad, Martin Rindarøy, Stian Skjong, Vilmar Æsøy, and Eilif Pedersen. Distributed co-simulation of maritime systems and operations. *Journal of Offshore Mechanics and Arctic Engineering*, 141(1), sep 2018.

[21] Philippe; Paquin Jean-Nicolas Belanger, Jean; Venne. The what, where, and why of real-time simulation. *OPAL-RT*, 2010.

[22] Jihas Khan. Rapid control prototyping (RCP) solutions for the validation of motor control applications. In *2016 International Conference on Emerging Technological Trends (ICETT)*. IEEE, oct 2016.

[23] Janne Keranen and Tomi Raty. Validation of model-based testing in hardware in the loop platform. In *2013 10th International Conference on Information Technology: New Generations*. IEEE, apr 2013.

[24] *OP5600 HILBOX USER GUIDE, 2011.*

[25] *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer US, 2007.

[26] *The Integration of Process Design and Control, Volume 17 (Computer Aided Chemical Engineering)*. Elsevier Science, 2004.

[27] Toni Varga. *Razvoj i primjena kosimulacijskih sustava za sintezu regulacijskih struktura asinkronog stroja*. PhD thesis, Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, 2022.

[28] Xin-She Yang. Metaheuristic optimization. *Scholarpedia*, 6(8):11472, 2011.

SAŽETAK

Ovaj rad istražuje primjenu metaheurističkih optimizacijskih algoritama za dobivanje parametara vektorske regulacije pri čemu se optimizacija provodi unutar HIL kosimulacije. Prvo je opisan sinkroni motor s permanentnim magnetima i dan matematički model istog te definiran matematički model regualatora za vektorsko upravljanje motorom. Zatim je definirana HIL testing metodologija kao i RCP metoda čijom primjenom su ostvareni uvjeti za provođenje kosimulacije čiji je krajnji cilj optimizacija parametara vektorske regulacije. U sklopu istog poglavlja objašnjen je kosimulacijski postav koji se sastoji od dva stvarnovremenska simulatora. Potom su definirani optimizacijski problemi te objašnjen način rada i postavke optimizatora na kojem će se vršiti proces optimizacije kao i algoritma na kojem optimizator temelji svoj rad. U konačnici je nakon provođenja eksperimenta i dobivanja rezultata dan zaključak s ciljem potvrđivanja odnosno opovrgavanja korištenja metaheurističkih optimizacijskih algoritama za parametriranje PI regulatora.

Ključne riječi:: HIL, kosimulacija, metaheuristika, optimizacija, RCP, sinkroni motor s permanentnim magnetima.

Optimization of vector control parameters using HIL testing methodology

ABSTRACT

This paper discuss the application of metaheuristic optimization algorithms for obtaining vector control parameters within the framework of Hardware-in-the-Loop (HIL) co-simulation. Firstly, a permanent magnets synchronous motor is described, along with its mathematical model, and the mathematical model of the controller is defined. Then, the HIL testing methodology is outlined, along with the Rapid Control Prototyping (RCP) method, which application enables co-simulation, with the ultimate goal of optimizing vector control parameters. In the same chapter, the co-simulation setup, consisting of two real-time simulators, is explained. Subsequently, optimization problems are defined along with operation and settings of the optimizer, as well as the algorithm on which the optimizer is based. In the last chapter, after conducting experiments and obtaining results, a conclusion is drawn to confirm or refute the use of metaheuristic optimization algorithms for tuning parameters of PI regulators.

Keywords: co-simulation, HIL, metaheuristic, optimization, permanent magnet synchronous motor, RCP.

ŽIVOTOPIS

Adriana Nedić rođena je 16.04.1999. godine. Nakon završene osnovne škole upisuje Srednju školu fra Martina Nedića Orašje, smjer opća gimnazija.

Tijekom osnovnoškolskog i srednjoškolskog obrazovanja učestvuje u natjecanjima iz matematike, hrvatskog jezike te završava glazbenu školu u trajanju od 6 godina. Po završetku srednje škole 2018. godine dodjeljuje joj se titula učenice generacije.

Iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer Elektrotehnika. Na drugoj godini bira smjer Elektroenergetika. Po završetku sveučilišnog preddiplomskog studija 2021. godine upisuje sveučilišni diplomski studij Elektroenergetika, smjer Industrijska elektroenergetika.