1 UVOD

Regulatori u klasičnom sustavu za regulaciju napona sinkronih generatora izvedeni su u mikrokontrolerima. Izvedene su klasične regulacijske strukture zasnovane na PID regulatorima [6],[11].

U radu su prikazane osnove umjetnih neuronskih mreža, te njihova primjena u regulaciji napona i prigušenju niskofrekvencijskih oscilacija sinkronih generatora [2]. Opisano je nekoliko već realiziranih sustava za regulaciju napona sinkronih generatora zasnovanih na neuronskim mrežama[3],[4],[5],[29]. U ovom radu stavljeno je težište i na razvoj novog sustava uzbude sinkronog generatora razvijenog na platformi procesora za digitalnu obradu signala.

Zbog svojih karakteristika i mogućnosti ugradnje u sustav regulacije napona generatora, detaljno je prikazan neuronski regulator napona koji u sebi objedinjuje funkcije klasičnog regulatora napona i klasičnog stabilizatora elektroenergetskog sustava [4]. Neuronska mreža koja predstavlja regulator napona sinkronog generatora je unaprijedna perceptronska mreža koja sadrži dva sloja; šest neurona u unutarnjem, skrivenom sloju, te jedan neuron u izlaznom sloju. Podešavanje parametara neuronske mreže odvija se *on-line* tijekom rada sinkronog generatora pomoću algoritma povratnog prostiranja (eng. *BP algorithm*) uz modificirani oblik kriterijske funkcije. Neuronski regulator napona sinkronog generatora modeliran je u programskom paketu *Matlab Simulink*.[8]

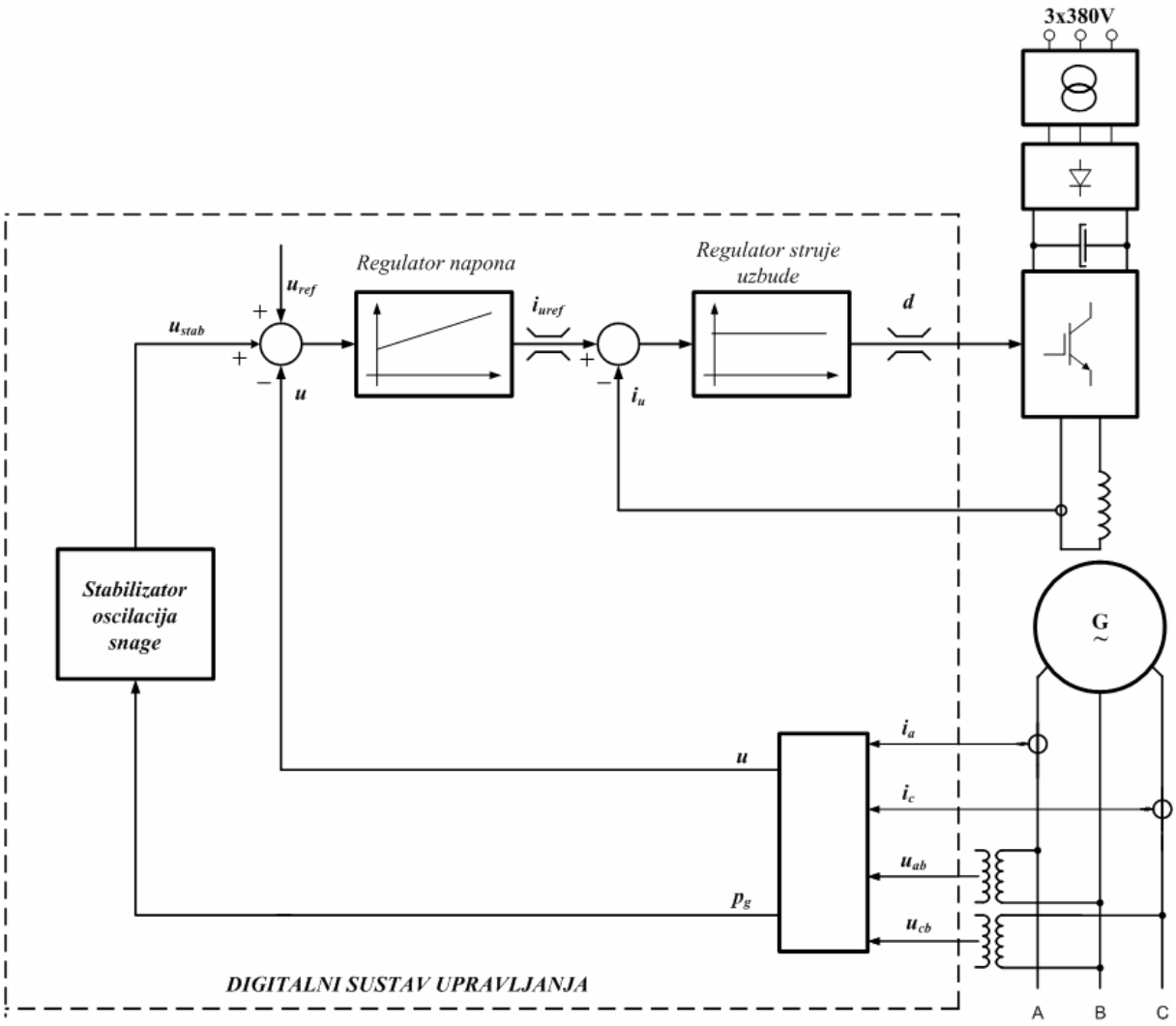
Za potrebe ovog rada izrađen je novi sustav uzbude na platformi digitalnog procesora za obradu signala tvrtke *Texas Instruments* TMS320F2812. Procesor je zasnovan na 32 bitnoj C2000 jezgri. Načinjeno je prilagodno sklopovlje za prilagodbu signala prema tranzistorskom pretvaraču.

Ispitano je djelovanje sustava regulacije napona sinkronog generatora s ugrađenim neuronskim regulatorom uzbude, te je uspoređeno s djelovanjem klasičnog sustava za regulaciju napona sinkronog generatora i s neizrazitim regulatorom napona realiziranom u digitalnom sustavu upravljanja uzbudom.[7]

Nakon simulacije djelovanja neuronskog regulatora napona sinkronog generatora, implementiran je regulacijski algoritam u procesor za obradu signala pomoću razvojnog alata *Code Composer Studio* tvrtke *Texas Instruments*. Cjelokupni regulacijski algoritam je načinjen u programskom jeziku C.

2 KLASIČNI SUSTAV REGULACIJE UZBUDE SINKRONOG GENERATORA

Standardni sustav za regulaciju napona sinkronog generatora prikazan je na slici 2.1.



Slika 2.1 Sustav regulacije napona sinkronog generatora

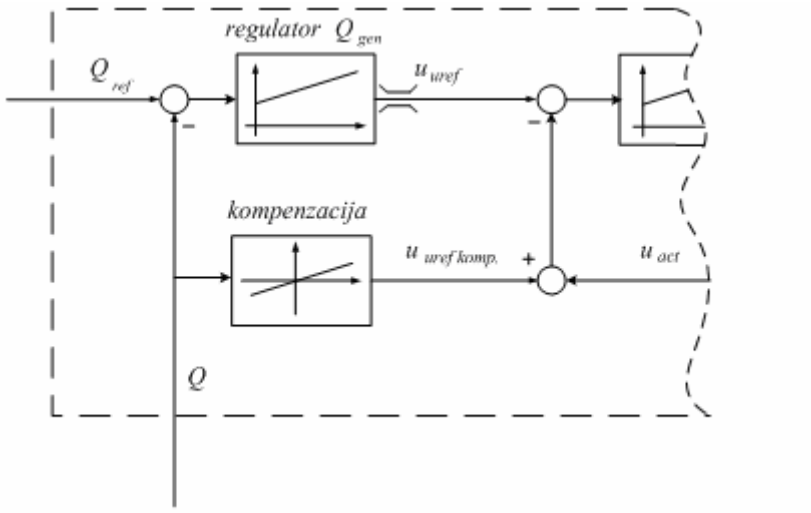
Sustav prikazan na slici 2.1. sadrži dvije povratne veze; povratnu vezu po naponu i unutarnju povratnu veza po struji uzbude sinkronog generatora. Mjere se dva linijska napona i dvije fazne struje generatora te struja uzbude generatora. Uzbuda se napaja iz čoperskog izvora. Regulator napona po svojoj karakteristici je PI tipa dok je regulator struje uzbude P tipa. Izlaz iz regulatora struje uzbude je referentna vrijednost pulsno-širinske modulacije *d*. Izlazi iz regulatora su ograničeni.

Osim osnovnih regulacijskih karakteristika, potrebno je u sustav uzbude dodati i zaštite, te osigurati kompenzaciju napona po jalovoj snazi generatora. Uvodi se ograničenje maksimalne struje uzbude i ograničenje maksimalne vrijednosti napona generatora.

U sustav uzbude sinkronog generatora uvodi se također i kompenzacija napona po jalovoj snazi. Kompenzacija napona po jalovoj snazi uključuje se kada je generator spojen na elektroenergetski sustav i ona osigurava potrebnu naponsku karakteristiku generatora. To se realizira tako da se signalu napona generatora dodaje signal proporcionalan jalovoj snazi (Slika 2.2)

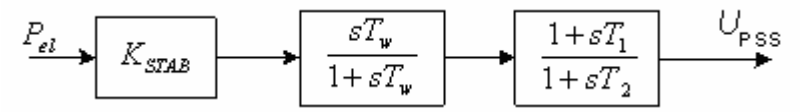
$$u_{komp} = u_g + q_g \cdot K \qquad (2.1)$$

U sustav regulacije uzbude sinkronog generatora moguće je uključiti i regulator jalove snage koji je nadređen regulatoru napona. Izlaz iz regulatora jalove snage predstavlja referentnu vrijednost regulatoru napona generatora (Slika 2.2.). Regulator jalove snage ima PI karakteristiku.



Slika 2.2. Regulator jalove snage i kompenzacija napona po jalovoj snazi

Suvremeni sustavi uzbude sinkronih generatora opremaju se sa stabilizatorom oscilacija radne snage. Zadatak stabilizatora oscilacija radne snage je da tijekom prijelaznog procesa, djelovanjem na uzbudu, priguši pojavu oscilacija radne snage. Stabilizator mora postići faznu kompenzaciju između ulaza u sustav uzbude i elektromagnetskog momenta u interesantnom spektru frekvencija njihanja (0.1-3 Hz). Uobičajeni signali koji se koriste kao ulazni signali stabilizatora su radna snaga generatora, brzina vrtnje ili frekvencija generatora. Stabilizator elektroenergetskog sustava sastoji se od bloka fazne kompenzacije, niskopropusnog filtra i pojačanja (Slika 2.3.).



Slika 2.3. Blokovska struktura stabilizatora oscilacija

3 UMJETNE NEURONSKE MREŽE

3.1 Osnovni modeli umjetnih neurona

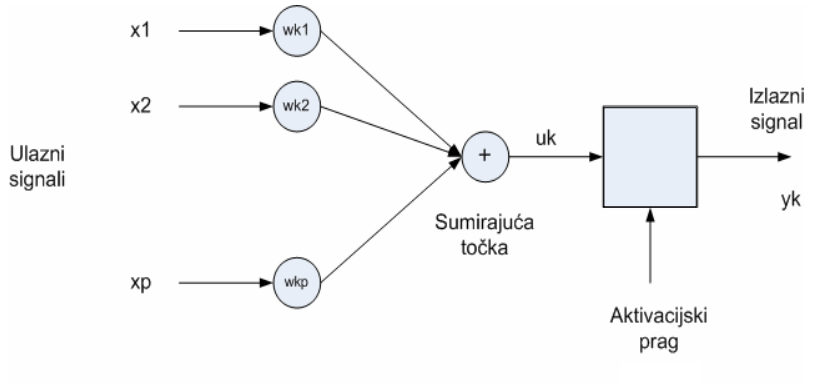
Većina dosad razvijenih modela umjetnih neurona i neuronskih mreža samo svojom strukturom podsjeća na biološke neurone (Dodatak A), bez pretenzije da budu njihovi stvarni modeli [2].

Na slici 3.1. prikazan je perceptron, umjetni neuron koji predstavlja jednostavan statički model biološkog neurona. Iz slike 3.1. možemo identificirati tri osnovna elementa modela neurona:

- skup sinapsi od kojih svaku karakterizira njezina težinska vrijednost,
- prag osjetljivosti ili sumacijska točka,
- aktivacijska funkcija za ograničenje amplitude izlaznog signala iz neurona.

Princip rada neurona može se objasniti na slijedeći način:

Svaki ulazni signal x_i množi se s težinskim koeficijentom w_i. Tako otežani ulazni signali se zbrajaju i njihov se zbroj uspoređuje s pragom osjetljivosti neurona w_{n+1}. Ako je zbroj otežanih signala veći od praga osjetljivosti, nelinearna aktivacijska funkcija ψ ga preslikava u izlazni signal neurona *y(t).* Izlazni signal najčešće je ograničen u području [-1,1]. Iako je veliki broj raznih aktivacijskih funkcija, funkcije *logsig* i *tansig* prihvaćene su kao standardne aktivacijske funkcije.



Slika 3.1. Shematski prikaz perceptrona

Matematički opisan neuron bi se sveo na sljedeće jednadžbe:

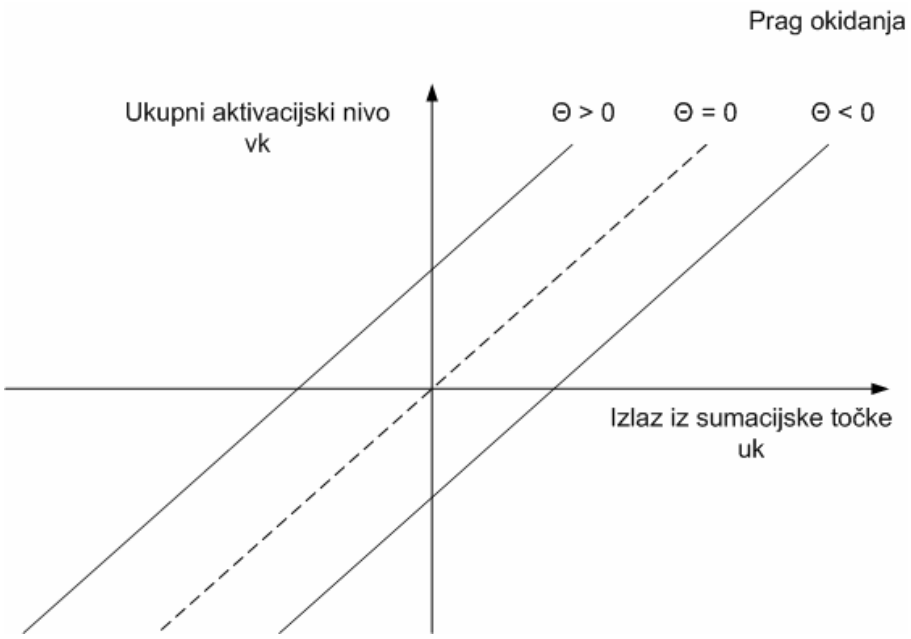
$$u_k = \sum_{j=1}^{p} w_{kj} x_j \,, \qquad (3.1)$$

$$y_k = \varphi(u_k - \theta_k) \,. \qquad (3.2)$$

Prag osjetljivosti dolazi do izražaja u afinoj transformaciji izlaza u_k. Ovisno o iznosu praga osjetljivosti vidimo (Slika 3.2.) da položaj grafa koji prikazuje ovisnost v_k od u_k ovisi o iznosu praga osjetljivosti.

$$v_k = u_k - \theta_k \qquad (3.3)$$

Osim prikazanog statičkog modela neurona na slici 3.1., postoji i dinamički model neurona koji u obzir uzima dinamička svojstva bioloških neurona. Takav dinamički model dobije se ako se statičkom neuronu doda na ulaz povratni signal(i) s izlaza neurona.



Slika 3.2. Izlazni signal iz neurona u ovisnosti o pragu okidanja

3.2 Tipovi aktivacijskih funkcija

Aktivacijska funkcija definira izlaz neurona u ovisnosti o na njenom ulazu. Mogu se identificirati tri osnovna modela aktivacijskih funkcija:

- ***Treshold* funkcija** s matematičkim opisom:

$$\varphi(v) = \begin{cases} 1 & za \quad v \geq 0 \\ 0 & za \quad v < 0 \end{cases}, \qquad (3.4)$$

te za izlaz iz neurona koji ima ovakvu aktivacijsku funkciju možemo pisati sljedeći izraz

$$y_k = \begin{cases} 1 & za \quad v_k \geq 0 \\ 0 & za \quad v_k < 0 \end{cases}, \qquad (3.5)$$

gdje je u_k unutarnji signal u neuronu prije prolaska kroz aktivacijsku funkciju

$$u_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \, . \qquad (3.6)$$

Neuron s takvom aktivacijskom funkcijom naziva se McCulloch-Pitts-ov model neurona.

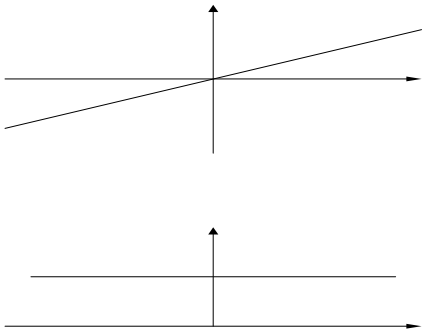
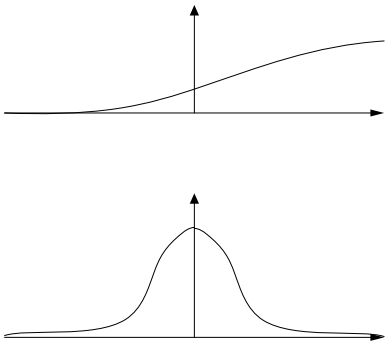
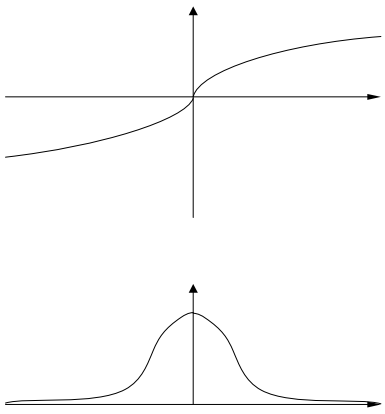
- **Linearna aktivacijska funkcija** (Slika 3.3 a)) možemo promatrati kao nelinearno pojačanje. Karakteristična su dva slučaja: linearno područje bez ulaska u zasićenje i zasićeno područje kada funkcija prelazi u *treshold* funkciju.

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ v, & -\frac{1}{2} \succ v \succ -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \qquad (3.7)$$

- ***Sigmoidalne* funkcija** koja je najčešća aktivacijska funkcija u neuronskim mrežama slika (Slika 3.3 b i c). Primjer *Sigmoidalne* funkcije:

$$\varphi(v) = \frac{1}{1+\exp(-av)}. \qquad (3.8)$$

Najčešće se upotrebljavaju *logsig* i *tansig* funkcija.

NAZIV FUNKCIJE	IZRAZ ZA FUNKCIJU I NJENU DERIVACIJU	GRAFIČKI PRIKAZ FUNKCIJE I DERIVACIJE
PURELIN	$\psi(v) = g_a v$ $\psi'(v) = g_a$	
LOGSIG	$\psi(v) = \frac{1}{1+e^{-gav}}$ $\psi'(v) = g_a \frac{e^{-gav}}{(1+e^{-gav})^2} = g_a \cdot \psi(1-\psi)$	
TANSIG	$\psi(v) = \frac{1}{1+e^{-gav}} - 1$ $\psi'(v) = g_a \frac{4e^{-2gav}}{(1+e^{-2gav})^2} = g_a \cdot (1-\psi^2)$	

Slika 3.3. Tipovi aktivacijskih funkcija

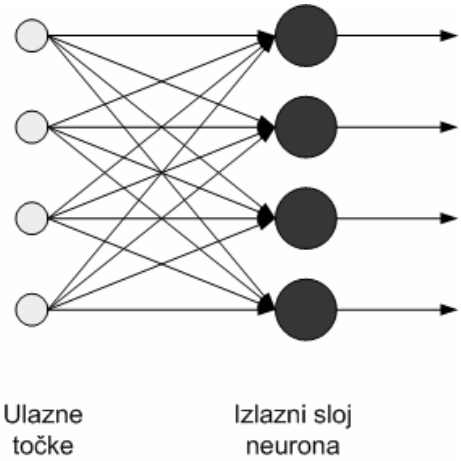
3.3 Arhitektura neuronskih mreža

Sa strukturnog stajališta, umjetne neuronske mreže se dijele na statičke (unaprijedne) i dinamičke (povratne), ovisno o modelu neurona od kojeg su građene, te o načinu prostiranja signala kroz mrežu. S obzirom na broj slojeva u kojima su raspoređeni neuroni, razlikuju se jednoslojne i višeslojne umjetne neuronske mreže. Kao zasebne strukture navode se još neizrazite neuronske mreže i nekonvencionalne mreže.

Za primjenu u identifikaciji i upravljanju nelinearnim dinamičkim procesima najčešće se koriste višeslojne statičke neuronske mreže

3.3.1 Jednoslojne unaprijedne mreže

Najjednostavniji oblik neuronskih mreža organiziranih u slojeve je mreža koja se sastoji od jednog sloja (Slika 3.4.).

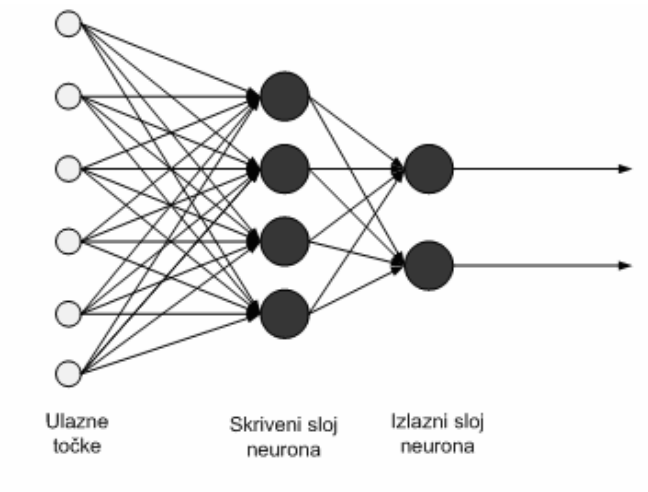


Slika 3.4. Jednoslojne unaprijedne mreže

Primjer jednoslojne unaprijedne mreže je asocijativna memorija. Izlaz iz mreže se formira u ovisnosti o ulaznom signalu i o težinama na ulazu.

3.3.2 Višeslojne unaprijedne mreže

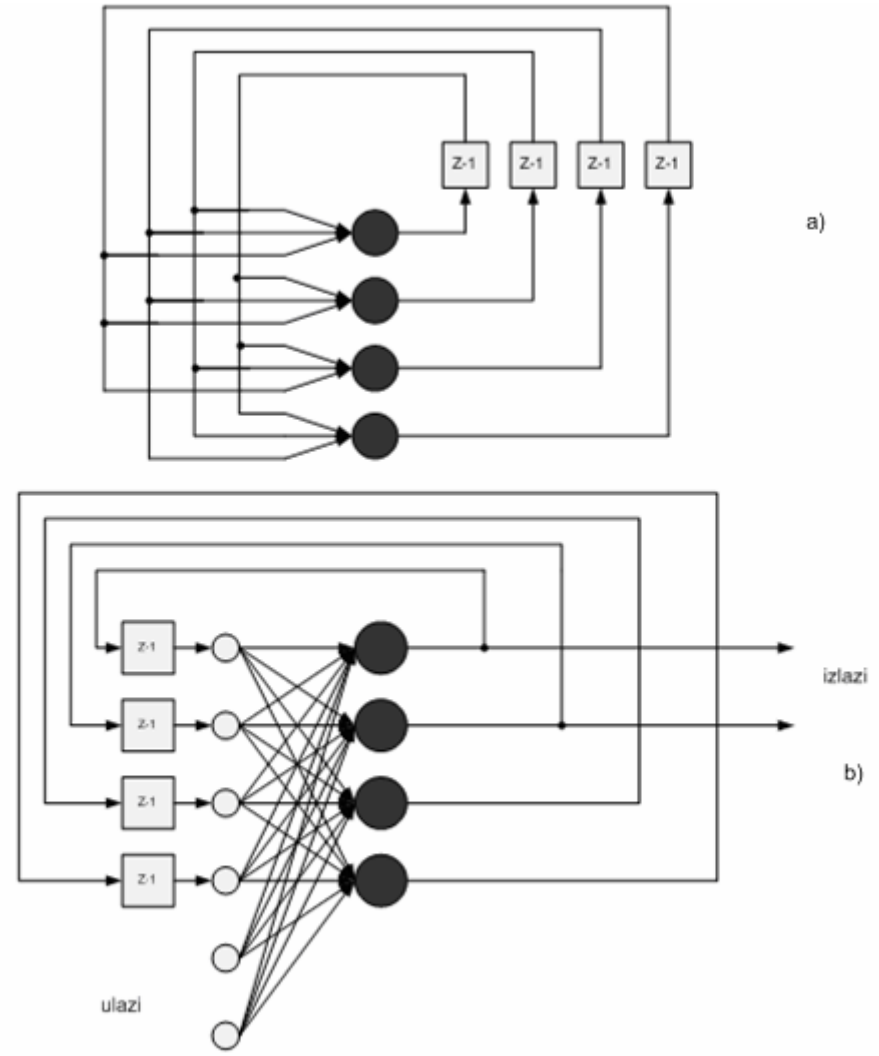
Za razliku od jednoslojnih mreža višeslojne mreže imaju i skriveni sloj (Slika 3.5.). Funkcija skrivenog sloja je da mreži omogući opisivanje nelinearnosti većeg reda. Višeslojne mreže mogu biti potpuno spojene, što znači da svaki ulaz u svaki neuron u svakom sloju je spojen sa svakim izlazom iz neurona u prethodnom sloju. Za razliku od potpuno spojenih kod djelomično spojenih mreža svaki ulaz u neuron ne mora biti spojen sa svakim izlazom iz neurona prethodnog sloja.



Slika 3.5. Višeslojne unaprijedne mreže

3.3.3 Dinamičke mreže

Dinamičke mreže su po strukturi iste kao i višeslojne mreže s tom razlikom da imaju barem jednu povratnu vezu (Slika 3.6.). Povratna veza podrazumijeva da se na ulaz dovodi zakašnjeli signal izlaza.

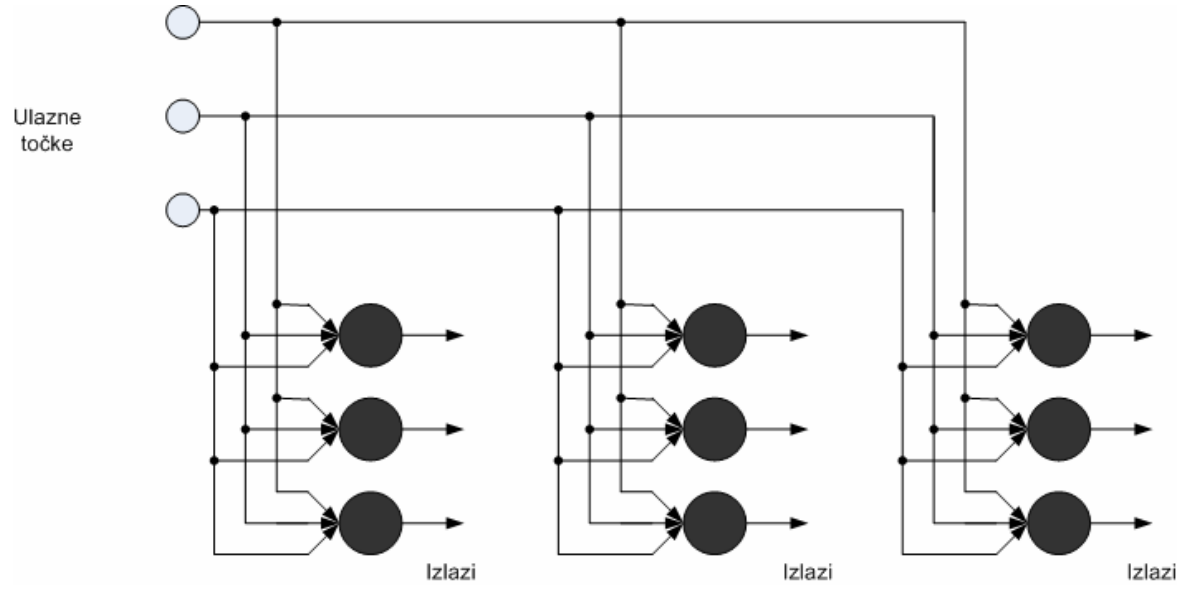


Slika 3.6. Dinamičke mreže, a) princip, b) primjer dinamičke mreže

Na slici 3.6 a) prikazan je princip po kojem se dovodi zakašnjeli signal s izlaza neuronske mreže na ulaz, dok je na slici 3.6. b) prikazana realizacija dinamičke neuronske mreže s četiri neurona.

3.3.4 Rešetkasta struktura (*Lattice*)

Lattice ili rešetkasta struktura sastoji se od jedno, dvo ili trodimenzionalnog skupa neurona na koji su dovedeni ulazni signali kako prikazuje slika 3.7.



Slika 3.7 Rešetkasta (*Lattice*) struktura

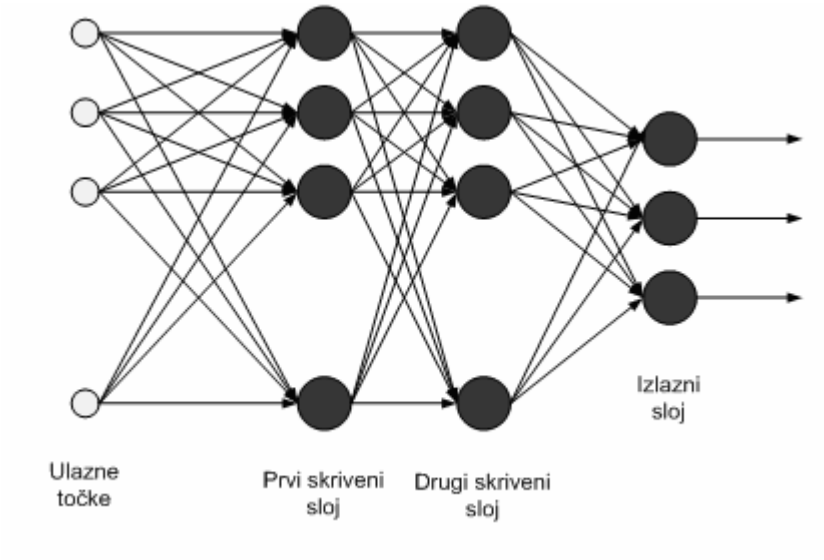
3.4 Algoritmi učenja neuronskih mreža

Algoritam učenja podešava parametre neuronske mreže s ciljem postizanja njezinog željenog ponašanja. Kod primjene neuronskih mreža u sustavima regulacije najčešće je poznato željeno vladanje neuronske mreže, pa se za njezino učenje primjenjuju algoritmi temeljni na pogrešci. Kao mjera pogreške koristi se kriterijska funkcija (kriterij kakvoće) $\Im(\theta)$ koja može biti bilo koja pozitivna skalarna funkcija ovisna o parametrima neuronske mreže. Algoritam učenja podešava parametre neuronske mreže tako dugo dok kriterijska funkcija ne poprimi minimalni iznos.

Osim algoritama učenja temeljenih na pogrešci postoje još algoritmi temeljeni na izlazu mreže i algoritmi s ojačanjem. Neuronske mreže s algoritmom učenja temeljenim na izlazu mreže uče samo na osnovi ulaznih signala u mrežu. Algoritmi učenja s ojačanjem zasnivaju se na tzv. signalu ojačanja koji daje kvalitativnu ocjenu vladanja neuronske mreže. Ovaj način učenja primjenjuje se kada vanjski referentni signal koji opisuje željeno vladanje mreže nije dostupan.

3.4.1 MLP neuronske mreže i algoritam povratnog prostiranja

Višeslojna perceptronska mreža (eng. *Multy Layer Perceptron*, MLP) izgrađena je od perceptrona organiziranih u serijski povezane slojeve (Slika 3.8.).



Slika 3.8. Višeslojna perceptronska mreža (MLP mreža)

Svi neuroni u nekom sloju povezani su sa svim neuronima u dva susjedna sloja preko jednosmjernih unaprijednih veza. Veze između neurona susjednih slojeva predstavljene su sinaptičkim težinskim koeficijentima koji djeluju kao pojačala signala na odgovarajućim vezama. Iznosi sinaptičkih težinskih koeficijenata određuju vladanje mreže. Izračunavanje njihovih odgovarajućih iznosa ostvaruje se algoritmima učenja.

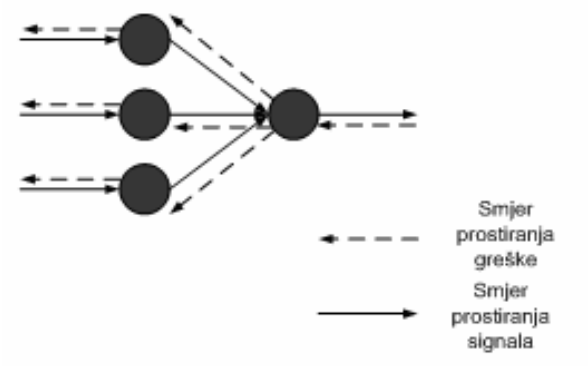
Iako nema ograničenja na broj slojeva od kojih se mreža sastoji, uglavnom se koriste dvoslojne i troslojne MLP mreže. Najčešće korištene aktivacijske funkcije su *tansig*, *logsig* ili *pureline* aktivacijska funkcija.

3.4.2 Osnovni pojmovi i definicije

Osnovne karakteristike višeslojne perceptronske mreže su sljedeći:

- svaki neuron u neuronskoj mreži sadrži nelinearnost tipa *tansig* ili *logsig*,
- mreža se sastoji od jednog ili više skrivenih slojeva koji ne pripadaju ni ulazu ni izlazu,
- mrežu odlikuje veliki stupanj spajanja među neuronima.

Najraširenij algoritam za podešavanje težina je algoritam povratnog prostiranja greške. U principu algoritam povratnog prostiranja greške ima dva prolaza kroz mrežu. U prostiranju prema naprijed signal prolazi kroz neuronsku mrežu formirajući izlazni signal, dok u povratnom prostiranju sve se težine podešavaju u skladu s primijenjenom metodom učenja (Slika 3.9.).



Slika 3.9. Smjer signala u mreži

3.4.3 Algoritam povratnog prostiranja

Signal greške na izlazu neurona j u iteraciji n je definiran s:

$$e_j(n) = d_j(n) - y_j(n). \tag{3.9}$$

Osim toga definiramo trenutnu vrijednost kvadrata greške za neuron j kao $\frac{1}{2} \cdot e^2(n)$. Suma kvadrata greške za sve izlazne neurone dobije se sumiranjem svih $\frac{1}{2} \cdot e^2(n)$ po izlaznim neuronima. Tako da vrijedi:

$$\xi(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n), \tag{3.10}$$

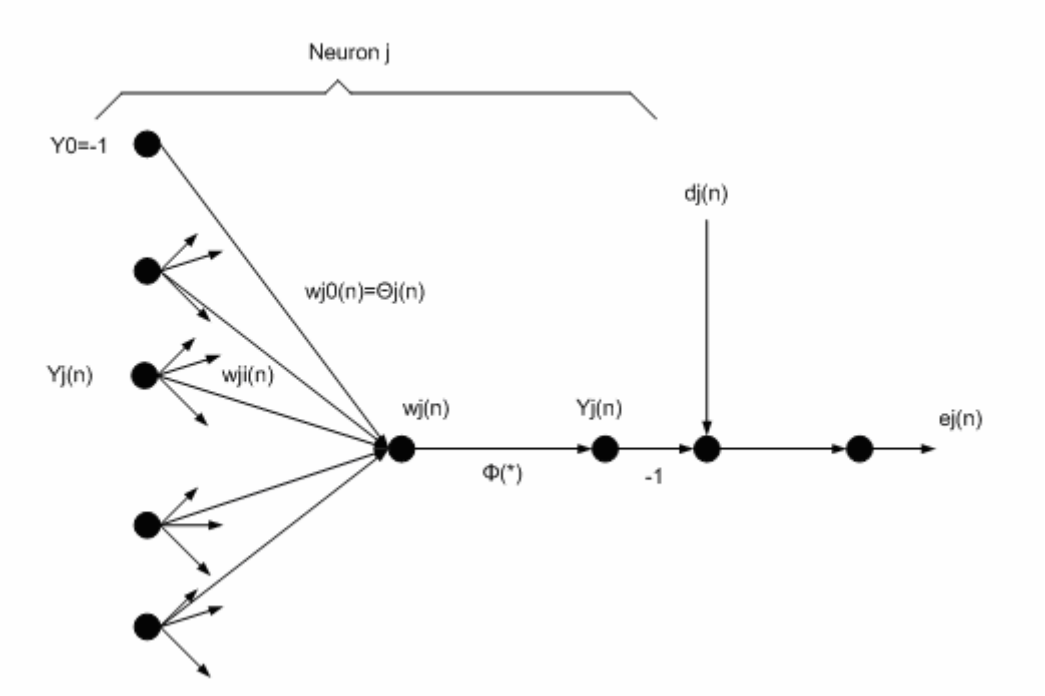
što predstavlja trenutnu sumu kvadrata greške za neuronsku mrežu. Srednja vrijednost kvadrata greške računa se po sljedećem izrazu :

$$\xi_{av} = \frac{1}{N} \sum_{n=1}^{N} \xi(n). \tag{3.11}$$

Promotrimo li na slici 3.10. prikaz jednog proizvoljnog neurona j u nekoj mreži. Za sumacijski signal $v_j(n)$ možemo pisati

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) \cdot y_i(n) \tag{3.12}$$

gdje p predstavlja ukupan broj ulaza u neuron j.



Slika 3.10. Proizvoljan neuron u neuronskoj mreži

Za signal $y_j(n)$ koji predstavlja izlaz iz neurona vrijedi:

$$y_j(n) = \varphi_j(v_j(n)) \tag{3.13}$$

Algoritam povratnog prostiranja vrši promjenu težina $\Delta w_{ji}(n)$ čija je promjena proporcionalna trenutnom gradijentu greške $\frac{\partial \xi(n)}{\partial w_{ji}(n)}$:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{3.14}$$

Podijelimo li obje strane jednadžbe (3.10) sa $e_j(n)$ dobije se:

$$\frac{\partial \xi(n)}{\partial e_j(n)} = e_j(n). \tag{3.15}$$

Nadalje ako podijelimo (3.9) sa $y_j(n)$ dobijemo

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1. \tag{3.16}$$

Ukoliko izraz (3.13) podijelimo s $v_j(n)$ dobijemo

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)), \tag{3.17}$$

Ako izraz (3.12) podijelimo s $w_{ji}(n)$ dobije se:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n). \tag{3.18}$$

Uvrštavanjem (3.15), (3.16), (3.17), (3.18) u izraz (3.14) dobije se:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot \varphi_j'(v_j(n)) \cdot y_i(n). \qquad (3.19)$$

Promjena težine $\Delta w_{ji}(n)$ za $w_{ji}(n)$ je definirana s

$$\Delta w_{ji}(n) = -\eta \cdot \frac{\partial \xi(n)}{\partial w_{ji}(n)}. \qquad (3.20)$$

gdje je η konstanta koja određuje korak učenja. Uvrštavanjem (3.19) u (3.20) dobije se sljedeći izraz za podešavanje težina neuronske mreže

$$\Delta w_{ji}(n) = \eta \cdot \delta_j(n) \cdot y_i(n). \qquad (3.21)$$

Gdje je $\delta_j(n)$ definiran s

$$\delta_j(n) = \frac{\partial \xi(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n) \cdot \varphi_j'(v_j(n)). \qquad (3.22)$$

Vidi se iz (3.21) i (3.22) da su za računanje promjene težine ključni signali greške i derivacija aktivacijske funkcije. Gledajući gornje jednadžbe vidimo da su moguća dva slučaja neuron *j* je u izlaznom sloju odnosno neuron j je u skrivenom sloju.

- **Neuron je u izlaznom sloju**

  U ovom slučaju postojeći izraz (3.9) za računanje greške koja se odnosi na taj neuron jednostavno je tako da se može odmah izračunati lokalni gradijent koristeći izraz (3.22)

- **Neuron je u skrivenom sloju**

  Kada se neuron nalazi u skrivenom sloju nema posebno definiranog željenog odziva za taj neuron. Iz tog razloga potrebno je odrediti signal greške za taj neuron rekurzivno preko svih neurona na koje je taj neuron spojen. Slika 3.11. prikazuje neuron j u skrivenom sloju. Ako se pogleda izraz (3.22) potrebno je redefinirati lokalni gradijent $\delta_j(n)$ za neuron u skrivenom sloju izraz .

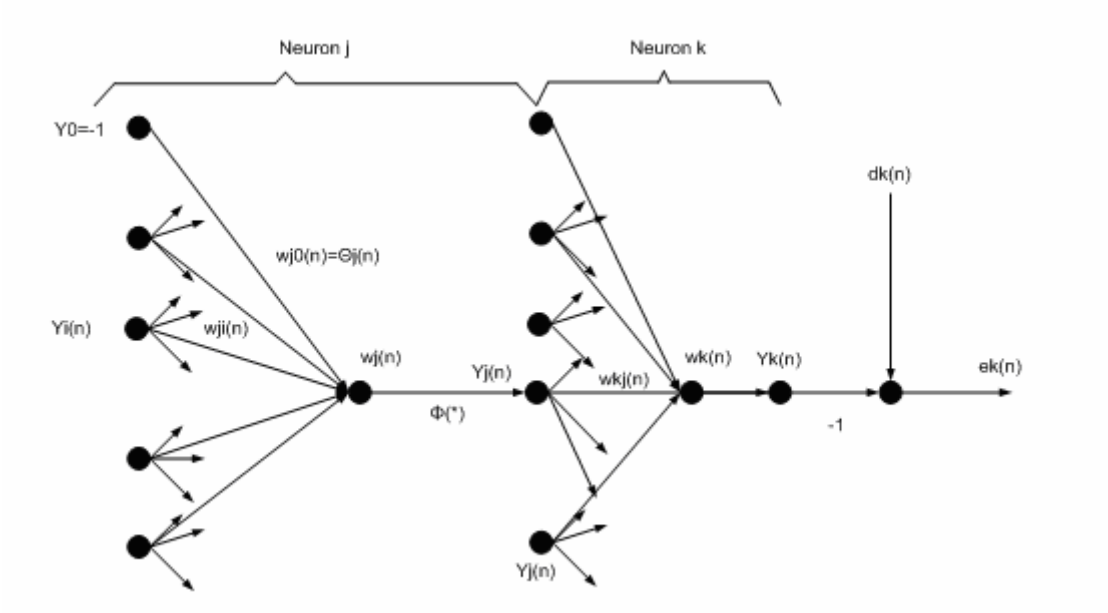
$$\delta_j(n) = \frac{\partial \xi(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial \xi(n)}{\partial y_j(n)} \cdot \varphi_j'(v_j(n)). \qquad (3.23)$$

Gledajući sliku 3.9 možemo pisati da je :

$$\xi(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n), \qquad (3.24)$$

gdje je k neuron koji se nalazi u izlaznom sloju, a j neuron koji se nalazi u skrivenom sloju. Ako se izraz (3.24) parcijalno derivira po $y_j(n)$, dobiva se:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}. \qquad (3.26)$$



Slika 3.11. Neuron u skrivenom i izlaznom sloju

Ako se jednadžba (3.26) proširi:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}. \qquad (3.27)$$

Nadalje sa slike 3.11. može se pisati:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \qquad (3.28)$$

i

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi'(v_k(n)). \qquad (3.29)$$

Za neuron *k* vrijedi da je $v_k(n)$ jednako

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) \cdot y_j(n). \qquad (3.30)$$

Ako se (3.30) parcijalno derivira po $y_j(n)$, dobije se

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \qquad (3.31)$$

Koristeći (3.29), (3.31) i (3.27) dobije se željena parcijalna derivacija

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_k e_k(n) \cdot \varphi_k'(v_k(n)) \cdot w_{kj}(n) = -\sum_k \delta_k(n) \cdot w_{kj}(n). \qquad (3.32)$$

Konačno uvrštavajući (3.32) u (3.24) dobije se sljedeći izraz za neuron u skrivenom sloju:

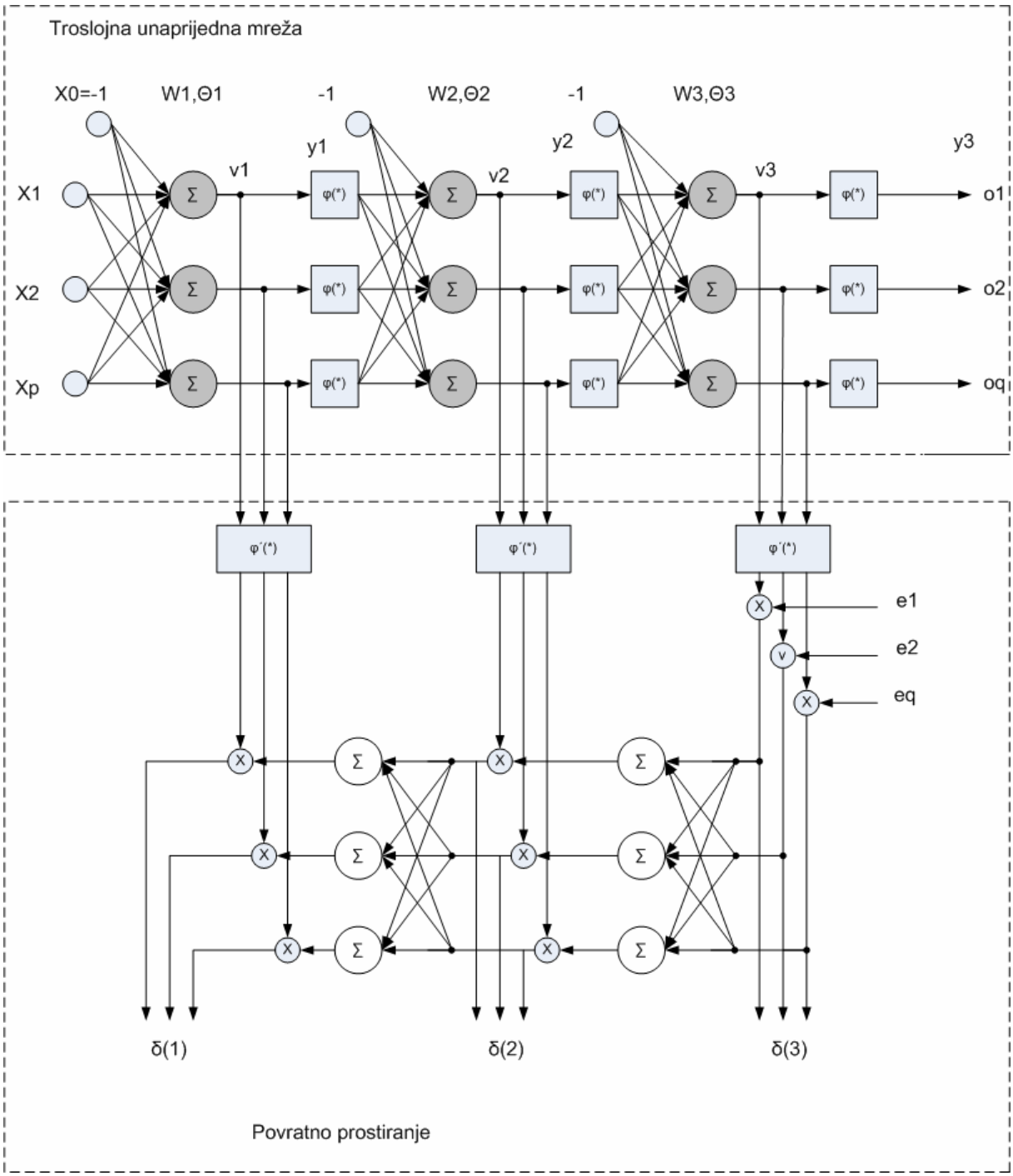
$$\delta_j(n) = \varphi_j'(v_j(n)) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n). \qquad (3.33)$$

Može se napisati za $\Delta w_{ji}(n)$ sljedeći izraz:

$$\begin{pmatrix} \text{Promjena} \\ \text{težine} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{Korak} \\ \text{ucenja} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{Lokalni} \\ \text{gradijent} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{Ulazni signal} \\ \text{u neuron} \\ y_j(n) \end{pmatrix} \qquad (3.34)$$

Može se primijetiti da je jedina razlika u izračunu promjene težina pojedinog neurona način izračuna lokalnog gradijenta koji ovisi o fizičkoj lokaciji neurona u neuronskoj mreži.

U primjeni neuronske mreže iz gornjih razmatranja vidljivo je da postoje dvije faze računanja. U unaprijednoj fazi težine ostaju nepromijenjene u mreži te se računa konačna vrijednost izlaza iz mreže. U povratnoj fazi računanja, počevši od izlaznog sloja preko skrivenih slojeva pa sve do ulaznog sloja, računaju se promjene težine pojedinog neurona neuronske mreže što se vidi na slici 3.12.



Slika 3.12. Arhitekturalni graf unaprijedne mreže s proračunom težina

3.4.4 Nelinearna aktivacijska funkcija

Da bi se mogao koristiti algoritam povratnog prostiranja, potrebno je u neuronskoj mreži koristiti derivabilnu aktivacijsku funkciju. Zato se često koristi *logsig* funkcija koja je već opisana u poglavlju 3.2.

$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1+\exp(-v_j(n))} \quad \text{, za} \quad -\infty < v_j(n) < \infty \,. \qquad (3.35)$$

Ako se izraz (3.35) parcijalno derivira po $v_j(n)$, dobije se sljedeći izraz:

$$y_j(n)\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) = \frac{\exp(-v_j(n))}{\left[1+\exp(-v_j(n))\right]^2}. \quad (3.36)$$

Koristeći izraz (3.35) da se eliminira član $\exp(-v_j(n))$ iz izraza (3.36). dobije se derivacija

$$\varphi_j'(v_j(n)) = y_j(n)\left[1-y_j(n)\right]. \quad (3.37)$$

Za neuron smješten u izlaznom sloju dobije se izraz

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = \left[d_j(n)-o_j(n)\right]o_j(n)\left[1-o_j(n)\right], \quad (3.38)$$

dok za izraz u izlaznom sloju vrijedi izraz

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) = y_j(n)\left[1-y_j(n)\right]\sum_k \delta_k(n)w_{kj}(n). \quad (3.39)$$

3.4.5 Korak učenja

Korakom učenja definiramo kojom brzinom se utječe na promjenu težina. Što je korak učenja manji, to će biti i manje promjene težina te će mreža sporije učiti. Međutim, ako se korak učenja postavi jako velikim može se dogoditi da izlaz iz mreže postane nestabilan, to jest oscilatoran.

3.4.6 Kriteriji zaustavljanja

Ne postoje jasno definirani kriteriji koji određuju u kojem trenutku je potrebno zaustaviti učenje neuronske mreže pa te kriterije određuju praktični uvjeti. Ne postoje kriteriji koji jasno određuju konvergenciju algoritma učenja.

Smatra se da algoritam konvergira kada Euklidska norma gradijenta vektora dosegne neku malu vrijednost. Ovakav način može rezultirati dugačkim vremenima učenja. Da bi se smanjila vremena, smatra se da je algoritam konvergirao kada razlika u grešci između zadnja dva podešenja bude dovoljno mala.

Hibridni kriterij zaustavljanja je Kramer i Sangiovanni-Vincentelli kriterij koji kaže da se učenje zaustavi kada greška upadne u okvir neke male nama prihvatljive vrijednosti.

3.4.7 Inicijalizacija mreže

Prvi korak kod metode povratnog prostiranja je taj da se postave neke inicijalne vrijednosti težina neurona, odnosno da se mreža inicijalizira.

Ovaj postupak je izrazito osjetljiv iz razloga što može doći do preranog zasićenja. To se može razmotriti na sljedećem primjeru ukoliko se koristi *tansig* aktivacijska funkcija u rasponu od -1 do 1 te ukoliko se početne vrijednosti težine neurona postave blizu vrijednosti -1 ili 1 tada se može reći da je taj neuron u zasićenju. Nadalje promotrimo slučaj kad izlaz iz neurona poprima vrijednost 1 za ulazni signal u neuron vrijednosti od -1. Povećavanjem vrijednosti ulaza u neuron iznad -1 sve do vrijednosti 1 neuron će davati izlazni signal iznosa 1. Takav neuron je nepravilno zasićen.

Ovaj slučaj se može izbjeći ukoliko se podešenja početnih vrijednosti težina neurona uniformno raspodijele unutar vrijednosti 50 % u odnosu na vrijednost zasićenja. Detaljnije o rješavanju problema zasićenja neurona može se naći u literaturi [1].

4 PREGLED SUSTAVA REGULACIJU UZBUDE S VIŠESLOJNIM UNAPRIJEDNIM NEURONSKIM MREŽAMA

U pregledu radova iz ovog područja obuhvatit će se samo radovi vezani za višeslojne neuronske mreže koje se koriste u regulaciji napona sinkronih generatora. Upotrebe ostalih vrsta mreža za rješavanje navedene problematike neće se obrađivati jer prelaze granice ovog rada.

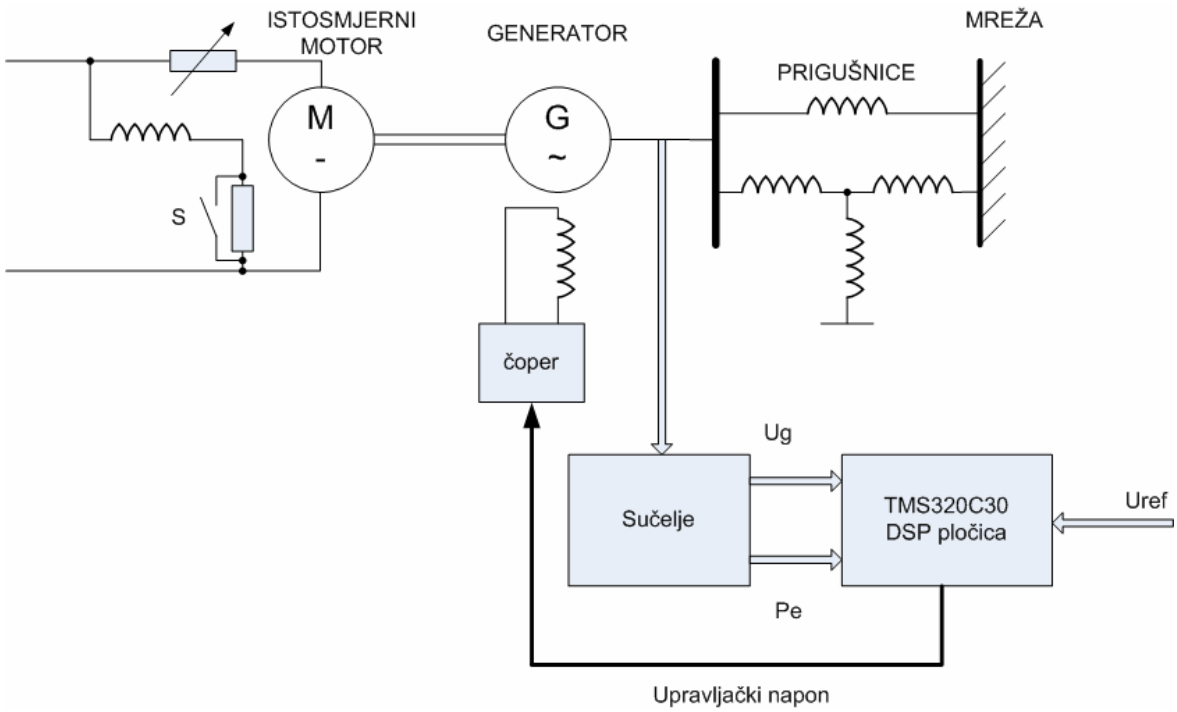
Iako su neke od ovih struktura upravljanja primijenjene u regulaciji napona sinkronih generatora, još je uvijek veliki broj neriješenih pitanja oko njihove implementacije u realni sustav. Tako npr. neuronske strukture regulacije ne osiguravaju uvijek zahtijevanu stabilnost i robusnost, pogotovo ako se neuronska mreža nalazi direktno u krugu regulacije. Problemi rastu i zbog nedovoljno razvijenih matematičkih alata za obradu nelinearnih tehnika upravljanja temeljenih na neuronskim mrežama, kao i zbog zahtjeva za matematičkim modelom procesa kod nekih struktura upravljanja.

U ovom pregledu prikazane su samo neuronske strukture regulacije sinkronih generatora koje izbjegavaju navedene probleme.

Potrebno je primijetiti razliku u načinu učenja pojedinih umjetnih neuronskih mreža. Učenje se može odvijati *on-line* ili *off-line*. *On-line* učenje odvija se tijekom rada sinkronog generatora, tj. u interakciji je sa sinkronim generatorom. *Off-line* učenje umjetne neuronske mreže temelji se na prethodno izmjerenim podacima, bez direktne interakcije sa procesom. Mjerni podaci moraju opisivati sinkroni generator u što više radnih točaka. Jednom naučena neuronska mreža, kad uđe u regulacijski sustav, više ne mijenja svoje naučeno ponašanje.

4.1 Regulator napona zasnovan na jednom perceptronu [3],[29]

U sustavu (Slika 4.1.) načinjen je regulator napona sinkronog generatora zasnovan na jednom perceptronu. Regulacijska struktura prema [3] implementirana je u procesor za digitalnu obadu signala TMS320C30. Na stezaljkama generatora su mjereni naponi i struje. Pomoću zasebne elektroničke pločice (sučelja) ti su se signali obradili i poslali u DSP. Sustav prema slici 4.1. izveden je u laboratoriju sveučilišta u Calgary-u. Generator je snage 3kVA i 220 V a pogonjen je istosmjernim motorom 220 V, 30 A. Na mrežu je bio spojen preko prigušnica koje su simulirale spojne vodove.



Slika 4.1 Eksperimentalni postav u laboratoriju sveučilišta u Calgary-u

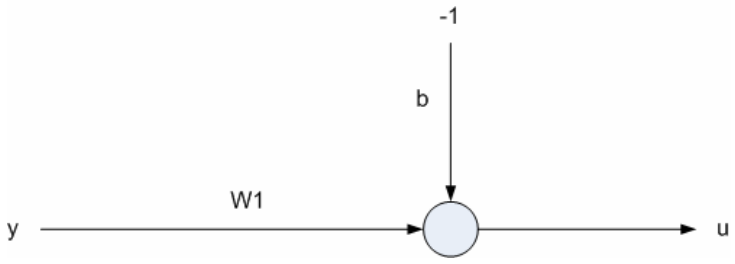
Regulator napona zasnovan na jednom neuronu prema [3] i [29] sastoji se od jednog pojačanja i jednog pomaka te je prikazan na slici 4.2**.** Matematički neuron sa slike 4.2. opisan je sljedećim izrazom:

$$u = y \cdot W - \theta \tag{4.1}$$

Budući da je ovo adaptivni neuron, promjena njegove težine i pomaka se kontinuirano događa ovisno o funkciji greške. Računanje funkcije greške kao i promjene težina i pomaka odvija se po algoritmu povratnog prostiranja.

$$\Delta W = \eta \cdot error \cdot y \tag{4.2}$$

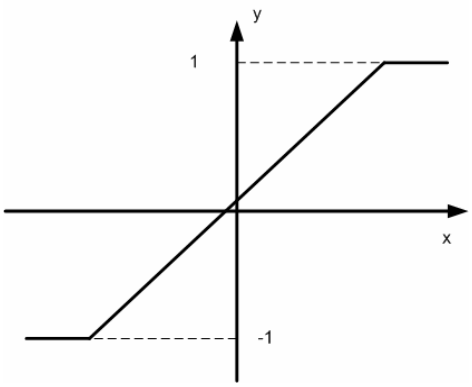
$$\Delta\theta = -\eta \cdot error \tag{4.3}$$



Slika 4.2 Adaptivni neuron

Izlazna aktivacijska funkcija je linearna sa zasićenjem i prikazana je na slici 4.3. Funkcija greške je modificirana kriterijska funkcija te se kontinuirano vrši promjena parametara

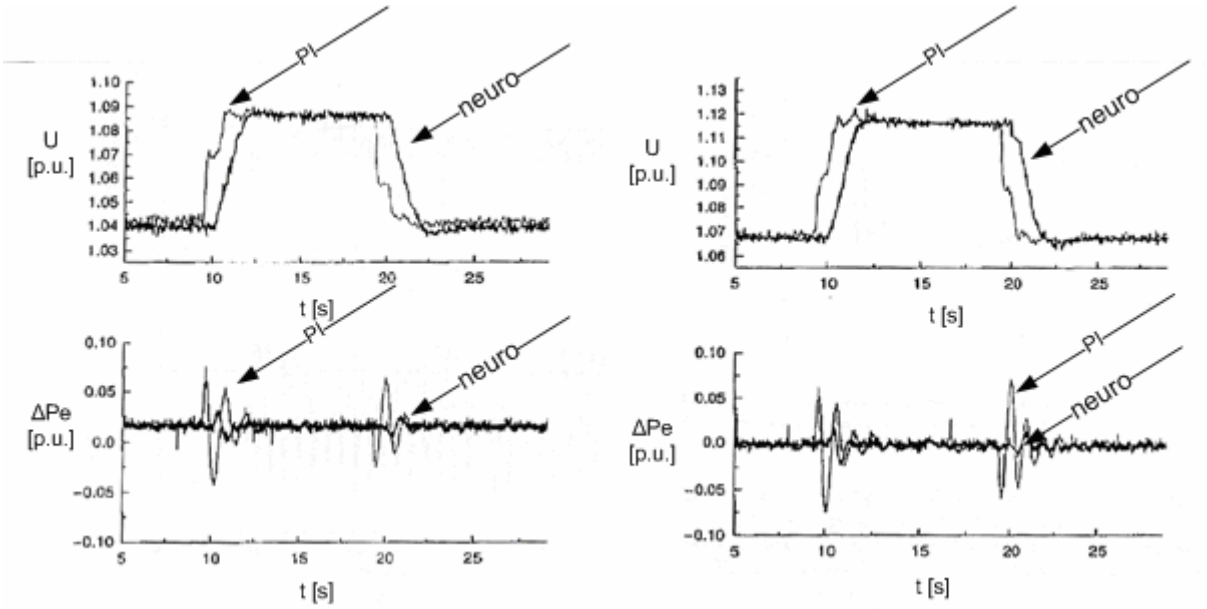
regulatora. Ulazne varijable u kriterijsku funkciju su referentna vrijednost napona generatora, stvarna vrijednost napona generatora te devijacija radne snage.



Slika 4.3 Linearna aktivacijska funkcija

$$error = \left[\left(U_{ref} - U_g \right) - k_c \left(\frac{dU_g}{dt} \right) \right] - \left[k_1 (\Delta P_e) + k_2 \left(\frac{dP_e}{dt} \right) \right] \qquad (4.4)$$

Prvi dio jednadžbe (4.4.) služi za regulaciju napona sinkronog generatora, dok drugi dio jednadžbe (4.4.) služi za stabilizaciju njihanja radne snage. U ovom slučaju ulazni signal u neuron je referentna vrijednost napona, dok je izlaz upravljački signal. Za sustav uzbude generatora zasnovanom na adaptivnom neuronu izvršena su snimanja djelovanja regulatora te je izvršena usporedba s klasičnim sustavom uzbude zasnovanim na proporcionalno integralnom regulatoru što je prikazano na slici 4.4.[3]



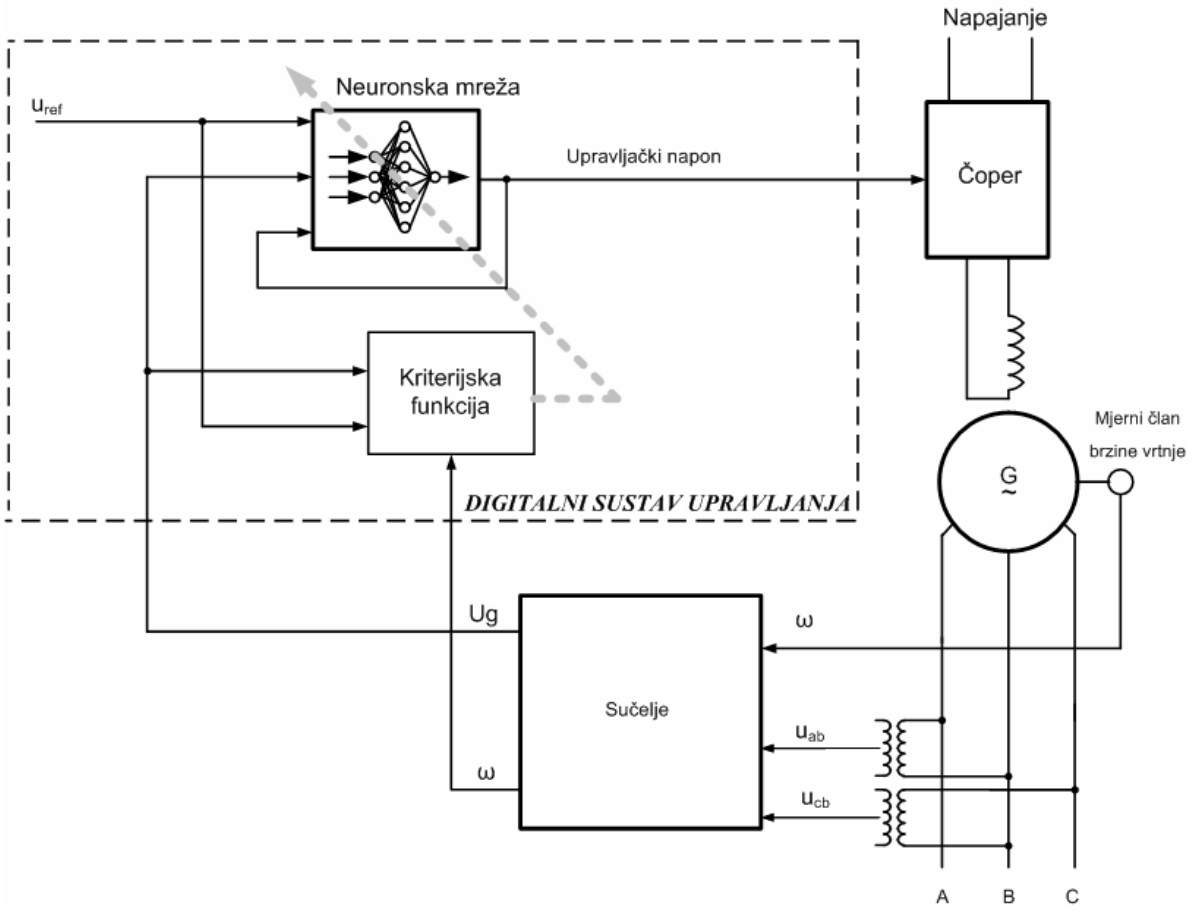
Slika 4.4 Eksperimentalni rezultati preuzeti iz [3]

Iz eksperimentalnih rezultata prema [3] je vidljivo da regulator s adaptivnim neuronom postiže kvalitetnije regulacijsko djelovanje u odnosu na PI regulator zbog boljeg prigušenja

oscilacija radne snage. Pokusi prema [3] rađeni su kad je generator bio spojen na mrežu preko prigušnica uz udarnu promjenu reference napona.

4.2 Regulator napona zasnovan na višeslojnoj neuronskoj mreži

Neuronski regulator napona predstavljen u [4] zamjenjuje regulator napona i stabilizator elektroenergetskog sustava u klasičnom sustavu regulacije napona sinkronog generatora. U ovoj izvedbi nema potrebe za identifikacijom parametara, ni dodatnim regulatorom. Struktura sustava za regulaciju napona sinkronog generatora s neuronskim regulatorom napona prikazana je na slici 4.5.



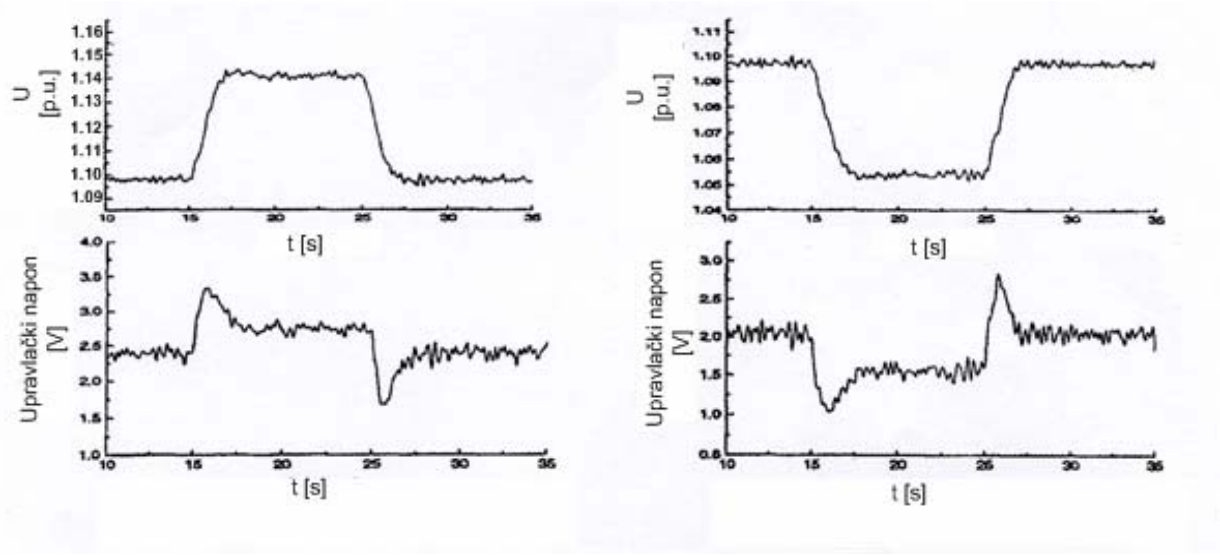
Slika 4.5 Sustav regulacije napona sinkronog generatora s neuronskim regulatorom napona prema [4]

Učenje se odvija *on-line* prema algoritmu povratnog prostiranja (BP algoritam). Umjesto standardne kriterijske funkcije u BP algoritmu, koristi se modificirana kriterijska funkcija (4.5.) koja je slična modificiranoj funkciji greške (4.4) u poglavlju 4.1. s tom razlikom da se umjesto signala devijacije radne snage koristi signal brzine vrtnje:

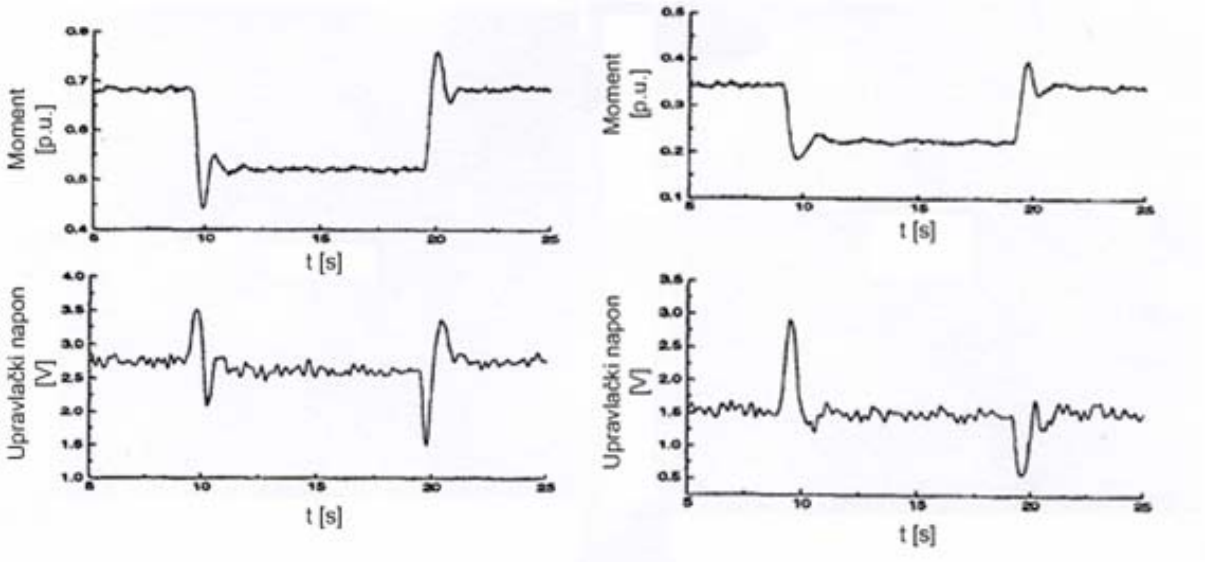
$$error = \left[\left(U_{ref} - U_g \right) - k_c \left(\frac{dU_g}{dt} \right) \right] - \left[\omega + k_\omega \left(\frac{d\omega}{dt} \right) \right]. \qquad (4.5.)$$

Neuronska mreža koja se koristi kao regulator napona sinkronog generatora ima tri ulazna signala: referentnu vrijednost napona sinkronog generatora (U_{ref}), napon sinkronog generatora (U_g) i prošlo stanje izlaza neuronske mreže(y_{t-1}); šest neurona u skrivenom sloju i jedan neuron u izlaznom sloju. Sinkroni generator kao i sustav u koji je implementirano rješenje isti je kao i u poglavlju 4.1.

Laboratorijska ispitivanja izvedena su prilikom rada generatora spojenog na mrežu preko prigušnica. Ispitivanja su izvedena za slučaj udarne promjene referentne vrijednosti napona. (Slika 4.6.). Osim udarne promjene referentne vrijednosti napona vršena su i ispitivanja kod udarne promjene momenta na osovini sinkronog generatora.(Slika 4.7.).



Slika 4.6. Eksperimentalni rezultati preuzeti iz [4] za promjenu naponske reference od 0.05 p.u. kad generator: a) predaje u mrežu 0.7 p.u. radne snage i b) predaje u mrežu 0.33 p.u. radne snage



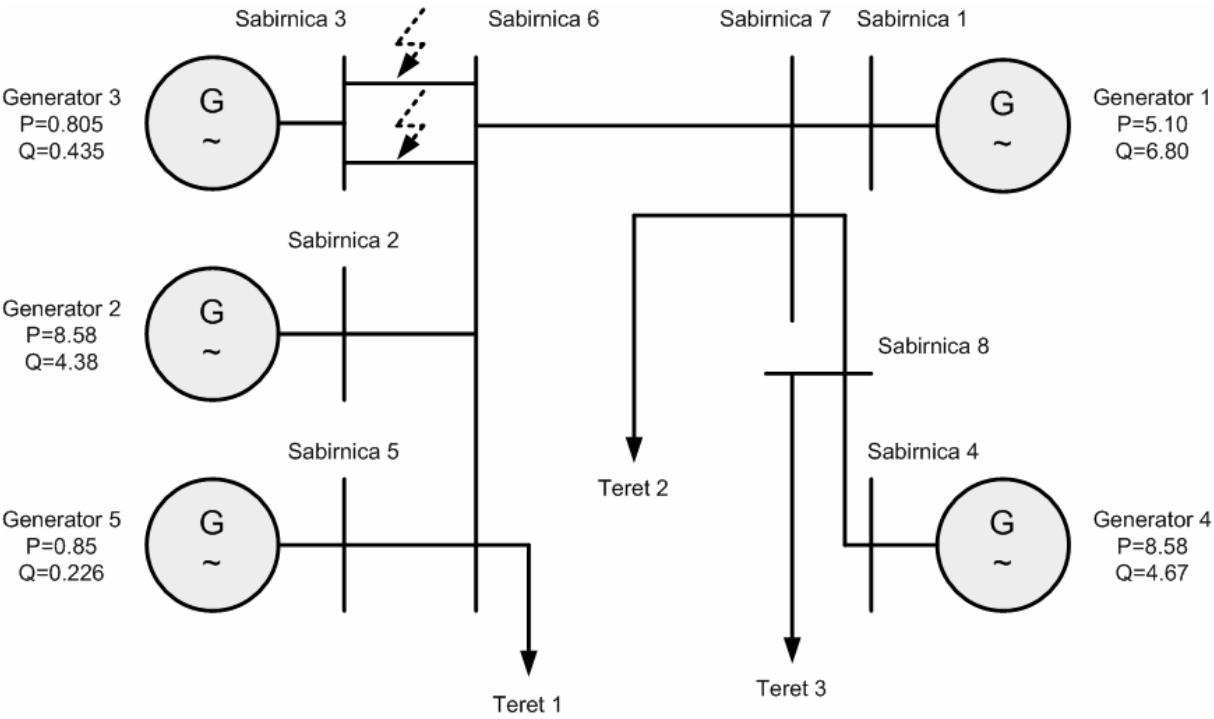
Slika 4.7. Eksperimentalni rezultati preuzeti iz [4] za promjenu reference momenta od 0.05 p.u. kad generator: a) predaje u mrežu 0.7 p.u. i b) predaje u mrežu 0.33 p.u. radne snage

Eksperimentalna ispitivanja prikazana na slici 4.6 i slici 4.7, pokazuju da ovaj neuronski regulator ostvaruje regulaciju napona sinkronog generatora i prigušenje oscilacija radne snage u uvjetima promjene naponske reference i reference momenta.

4.3 Simulacija ponašanja paralelnog rada generatora s regulatorima uzbude zasnovanim na neuronskoj mreži

U [5] je izvedena simulacija djelovanja regulatora napona zasnovanog na neuronskoj mreži (Slika 4.5.) u uvjetima kada je generator spojen na elektroenergetsku mrežu prikazanu na slici 4.8.

Realizirana neuronska mreža sastoji se od tri ulaza, šest neurona u skrivenom sloju i jednim izlaznim neuronom. Modificirana funkcija greške pomoću koje se odvija podešavanje neuronske mreže je dana u jednadžbi 4.5 poglavlja 4.2. Podešavanje težina neuronske odvija se *on-line* po algoritmu povratnog prostiranja.



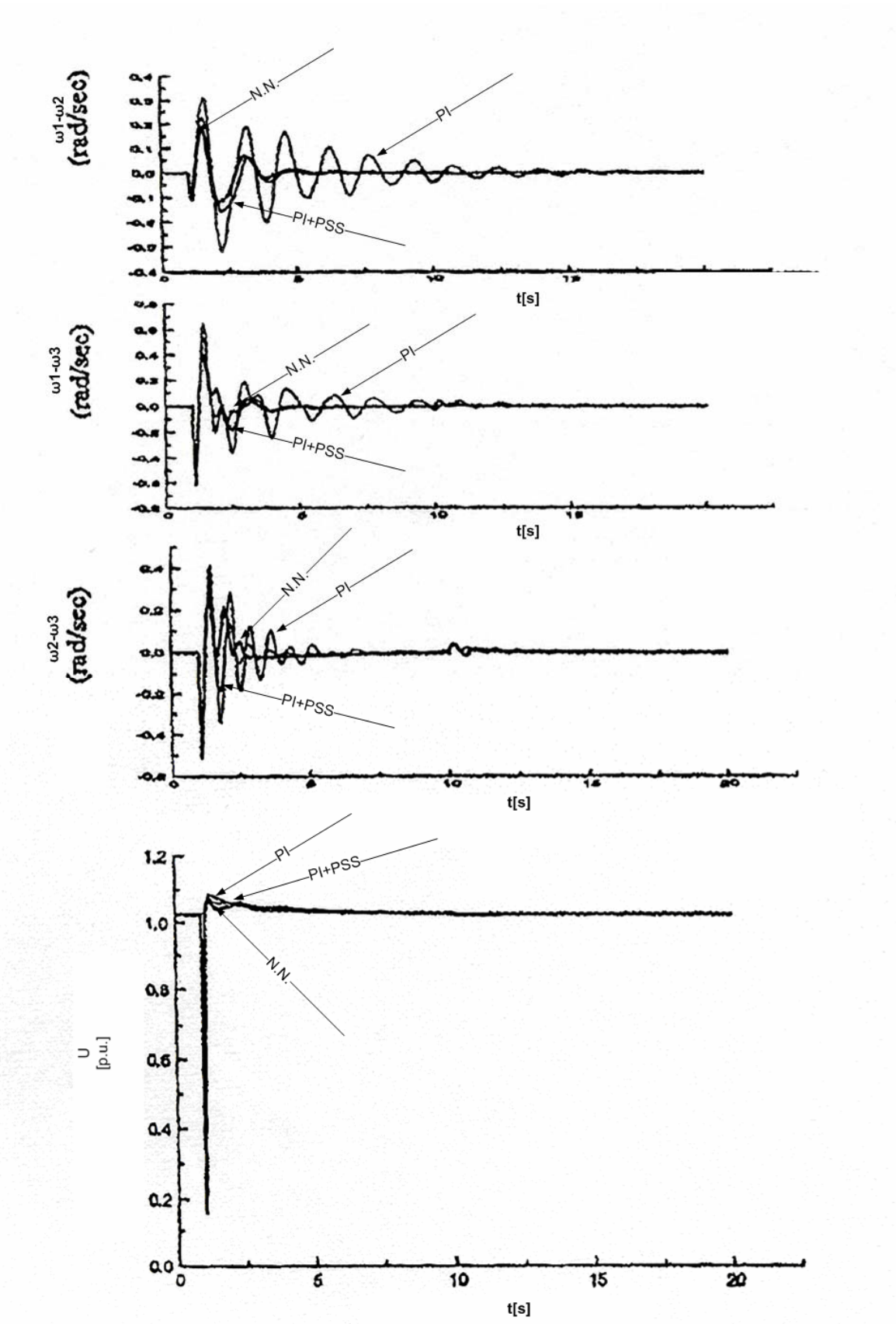
Slika 4.8. Modelirani sustav sinkronih generatora u [5]

Simulirani sustav sinkronih generatora može se razdvojiti na dio koji čine generatori 2, 3 i 5 te na dio koji čine generatori 1 i 4. Ova dva dijela povezana su jednim dugačkim vodom. U ovom slučaju istraživane su pojave lokalnih oscilacija te su prikazane za slučaj tropolnog kratkog spoja između sabirnica 3 i 6 na slici 4.8**.**

Potrebno je naglasiti da samo generator G3 ima sustav uzbude zasnovan na neuronskoj mreži, dok ostali generatori imaju klasični sustav uzbude.

Simulirani rezultati u prezentiranom sustavu generatora pokazuju opravdanost upotrebe sustava regulacije uzbude sinkronog generatora zasnovanog na neuronskoj mreži.

Ustanovljeno je da neuronska mreža ostvaruje značajnije prigušenje lokalnih oscilacija brzine vrtnje u simuliranim uvjetima te da regulira napon generatora što je vidljivo na slici 4.9.

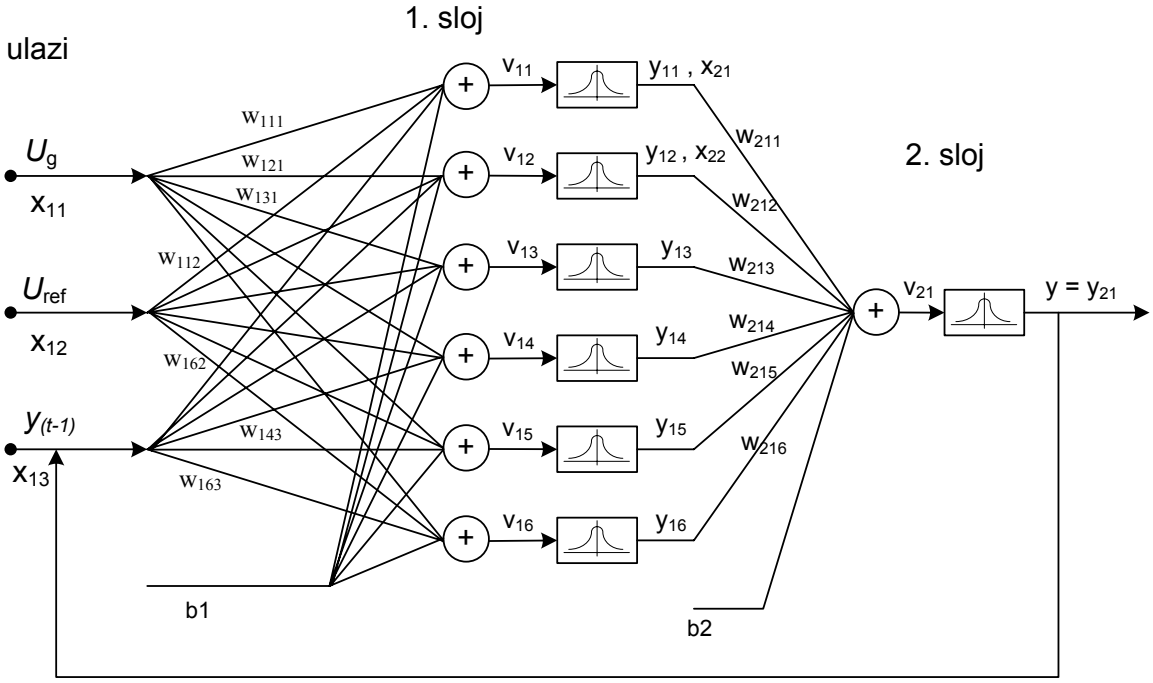


Slika 4.9. Simulacijski odzivi brzina vrtnje generatora za slučaj kratkog spoja na sabirnicama 3 i 6 (slika 4.8.) te odziv napona generatora G3 koji je najbliže kratkom spoju prema [5]

5 STRUKTURA SUSTAVA NEURONSKOG UPRAVLJNJA UZBUDOM SINKRONOG GENERATORA

5.1 Struktura regulatora napona sinkronog generatora zasnovanog na neuronskoj mreži

Regulator napona sinkronog generatora zasnovan na unaprijednoj neuronskoj mreži (MLP mreža) prikazan je na slici 5.1**.**



Slika 5.1. Regulator napona sinkronog generatora zasnovan na neuronskoj mreži

Neuronska mreža ima dva sloja, šest neurona u unutarnjem, skrivenom sloju, te jedan neuron u izlaznom sloju. Koriste se tri ulazna signala: referentna vrijednost napona generatora (U_{ref}), trenutna vrijednost napona generatora (U_g) i povratni signal s izlaza neuronske mreže ($y(t\text{-}1)$). Povratnim signalom izvana je statičkoj neuronskoj mreži dodan dinamički karakter. Ova jednostavna struktura pojednostavljuje algoritam učenja u smislu vremena izračunavanja. Kao aktivacijska funkcija pojedinih neurona u skrivenom i izlaznom sloju korištena je *tansig* nelinearna funkcija.

Izlaz iz pojedinog neurona dobije se prikupljanjem otežanih ulaznih signala i usporedbom njihovog zbroja s pragom osjetljivosti (pomakom), te propuštanjem tako dobivenog signala kroz aktivacijsku funkciju. Prema tome, izlaz *i*-og neurona u *k*-tom sloju iznose:

$$y_{ki} = \psi\left(\sum_{k} w_{kij} \cdot x_{kj} + b_1\right), \qquad (5.1)$$

gdje *j* predstavlja redni broj neurona od kojeg signal ide (ili u slučaju 1. sloja redni broj ulaznog signala), *x* je oznaka ulaznog signala, a *w* težinski koeficijent kojim se otežava pojedini ulazni signal .ψ *je* aktivacijska *tansig* funkcija opisana izrazom:

$$\tan sig(v) = \frac{2}{1 + e^{-2g_a v}} \ . \qquad (5.2)$$

Logika označavanja težinskih koeficijenata je sljedeća:

- Značenje pojedinog indeksa u težinskom koeficijentu na slici 5.1. je

  W sloj u kojem se nalazi neuron, neuron prema kojem ide signal, neuron od kojeg ide signal

- Primjerice, w_{162} označava težinski koeficijent kojim se množi signal koji ide od drugog ulaza prema četvrtom neuronu u prvom sloju. Indeksi uz oznaku izlaza ($y_{k,j}$) pojedinog neurona označavaju sloj u kojem se neuron nalazi (k) i redni broj neurona u tom sloju (j). Izlazi neurona u prvom, sloju odgovaraju ulazima u neuron u drugom sloju($y_{k,1} = x_{k+1,1}$)

5.2 Algoritam učenja

Parametri mreže koji se podešavaju su težinski koeficijenti pojedinih neurona u mreži (w_{kij}) i pragovi osjetljivosti neurona u pojedinom sloju (b_1 i b_2).

Korišteni algoritam podešavanja parametara neuronske mreže je algoritam povratnog prostiranja izlazne pogreške kroz mrežu, odnosno BP algoritam. BP algoritam učenja je iterativni, gradijentni algoritam učenja, projektiran sa ciljem da minimizira kriterijsku funkciju koja je jednaka kvadratu razlike između stvarnog i željenog izlaza iz neuronske mreže. Algoritam koristi rekurzivnu tehniku izračunavanja parcijalnih derivacija kriterijske funkcije po parametrima mreže od izlaznog prema ulaznom sloju mreže.

Težinski koeficijenti se podešavaju na temelju izraza:

$$w_{kij}(t+1) = w_{kij}(t) + \Delta w_{kij}(t) \ . \qquad (5.3)$$

Promjena vrijednosti težinskog koeficijenta u nekom trenutku $\Delta w_{kij}(t)$ izračunava se prema izrazu:

$$\Delta w_{kij}(t) = \eta \nabla \Im = \eta \frac{d\Im}{dw_{kij}}, \tag{5.4}$$

gdje je η korak učenja, a $\nabla \Im$ gradijent kriterijske funkcije po težinskom koeficijentu w_{kij}

Uobičajeno korištena kriterijska funkcija u BP algoritmu ima oblik:

$$\Im = \frac{1}{2}\left(t_{ki} - y_{ki}\right)^2, \tag{5.5}$$

gdje t_{ki} predstavlja željeni izlaz iz *i*-tog neurona u *k*-tom sloju.

BP algoritam izračunava parcijalne derivacije kriterijske funkcije po težinskim koeficijentima (gradijent kriterijske funkcije) prema izrazu:

$$\frac{\partial \Im}{\partial w_{kij}} = \frac{\partial \Im}{\partial y_{ki}} \cdot \frac{dy_{ki}}{dv_{ki}} \cdot \frac{\partial v_{ki}}{\partial w_{kij}} = \frac{\partial \Im}{\partial y_{ki}} \cdot \psi'_{ki} \cdot x_{kj}. \tag{5.6}$$

Izračunavanje parcijalne derivacije $\frac{\partial \Im}{\partial y_{ki}}$ izvršava se s obzirom na to o kojem se sloju neurona u neuronskoj mreži radi.

U slučaju izlaznog sloja mreže može se pisati:

$$\frac{\partial \Im}{\partial y_{ki}} = t_{ki} - y_{ki}. \tag{5.7}$$

Za unutarnji sloj mreže parcijalna derivacija izračunava se prema rekurzivnom izrazu:

$$\frac{\partial \Im}{\partial y_{ki}} = \sum_{p=1}^{n(k+1)} \frac{\partial \Im}{\partial y_{k+1,p}} \cdot \frac{dy_{k+1,p}}{dv_{k+1,p}} \cdot \frac{\partial v_{k+1,p}}{\partial y_{kp}} = \sum_{p=1}^{n(k+1)} \frac{\partial \Im}{\partial y_{k+1,p}} \cdot \psi'_{k+1,p} \cdot w_{k+1,1,i} \quad . \tag{5.8}$$

5.3 Modificirana kriterijska funkcije

Neuronski regulator napona sinkronog generatora, umjesto kriterijske funkcije zadane izrazom (5.5) koristi modificiranu kriterijsku funkciju koja osim razlike između trenutačnog i željenog izraza u sebi uključuje i promjenu te razlike. Izraz za parcijalnu derivaciju tako modificirane kriterijske funkcije po izlazu izlaznog sloja glasi:

$$\frac{\partial \Im}{\partial y_{ki}} = \left(t_{ki} - y_{ki}\right) - \frac{dy_{ki}}{dt}. \tag{5.9}$$

Ovakva modifikacija kriterijske funkcije ubrzava BP algoritam, tako da on postaje pogodan za implementacije koje se temelje na *on-line* učenju, tj. učenju koje se odvija tijekom normalnog rada sinkronog generatora.

Gradijent kriterijske funkcije koja u neuronskom regulatoru ostvaruje funkciju regulatora napona sinkronog generatora glasi:

$$\frac{\partial \Im}{\partial y_{ki}} = K(U_{ref} - U_g) - k_u \frac{dU_g}{dt}, \qquad (5.10)$$

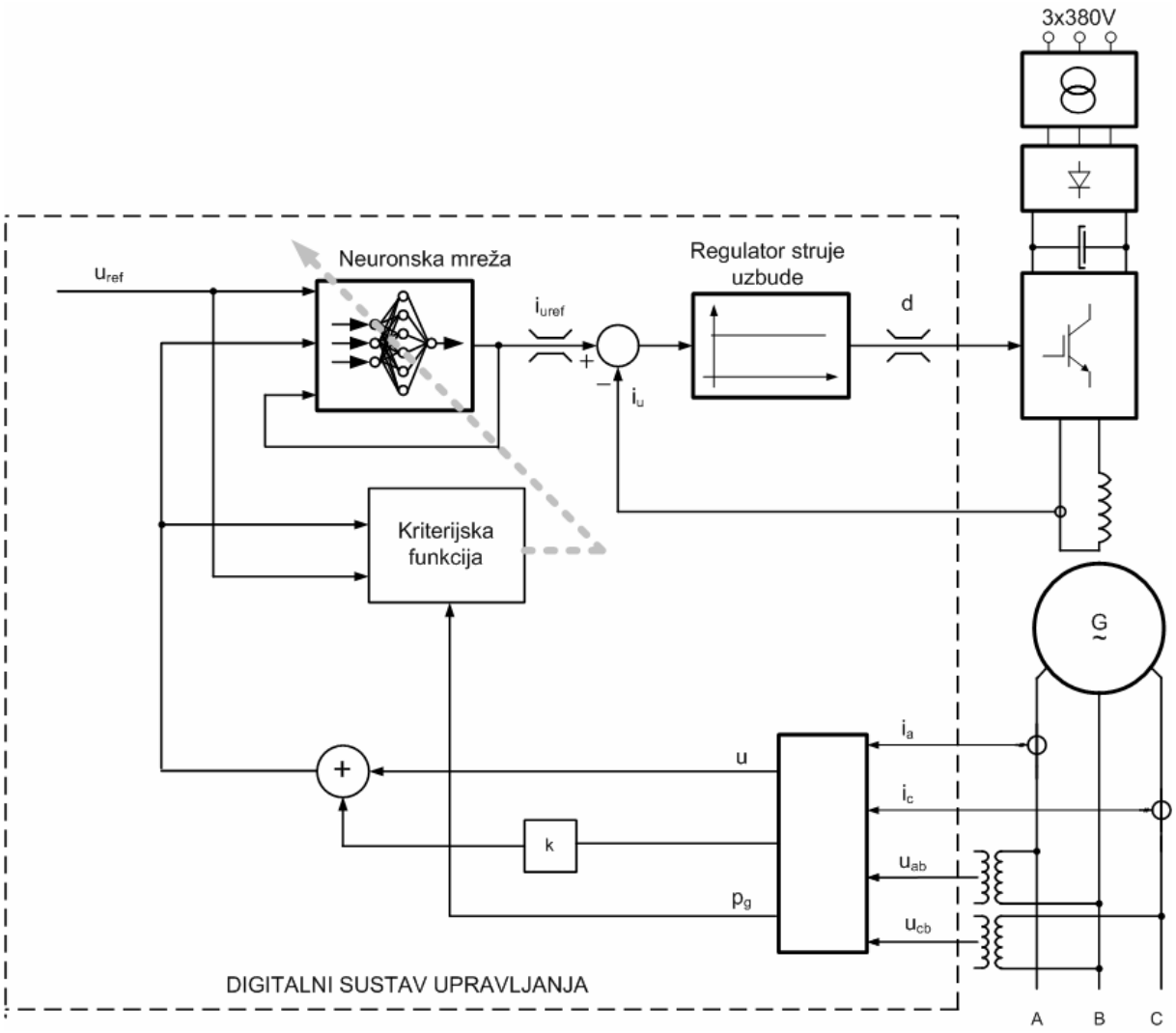
gdje U_{ref} predstavlja zadanu referentnu vrijednost napona generatora, U_g trenutačni iznos napona generatora (dq) te k_u faktor skaliranja.

Uvođenjem signala radne snage sinkronog generatora i daljnjim modificiranjem izraza (5.11) neuronski regulator uz funkciju regulatora napona ostvaruje i funkciju stabilizatora elektroenergetskog sustava. Krajnji modificirani izraz gradijenta kriterijske funkcije glasi:

$$\frac{\partial \Im}{\partial y_{ki}} = \left[K(U_{ref} - U_g) - k_1 \frac{dU_g}{dt} \right] - \left[k_3 \left(\Delta P_{el} \right) + k_2 \frac{dP_{el}}{dt} \right], \qquad (5.11)$$

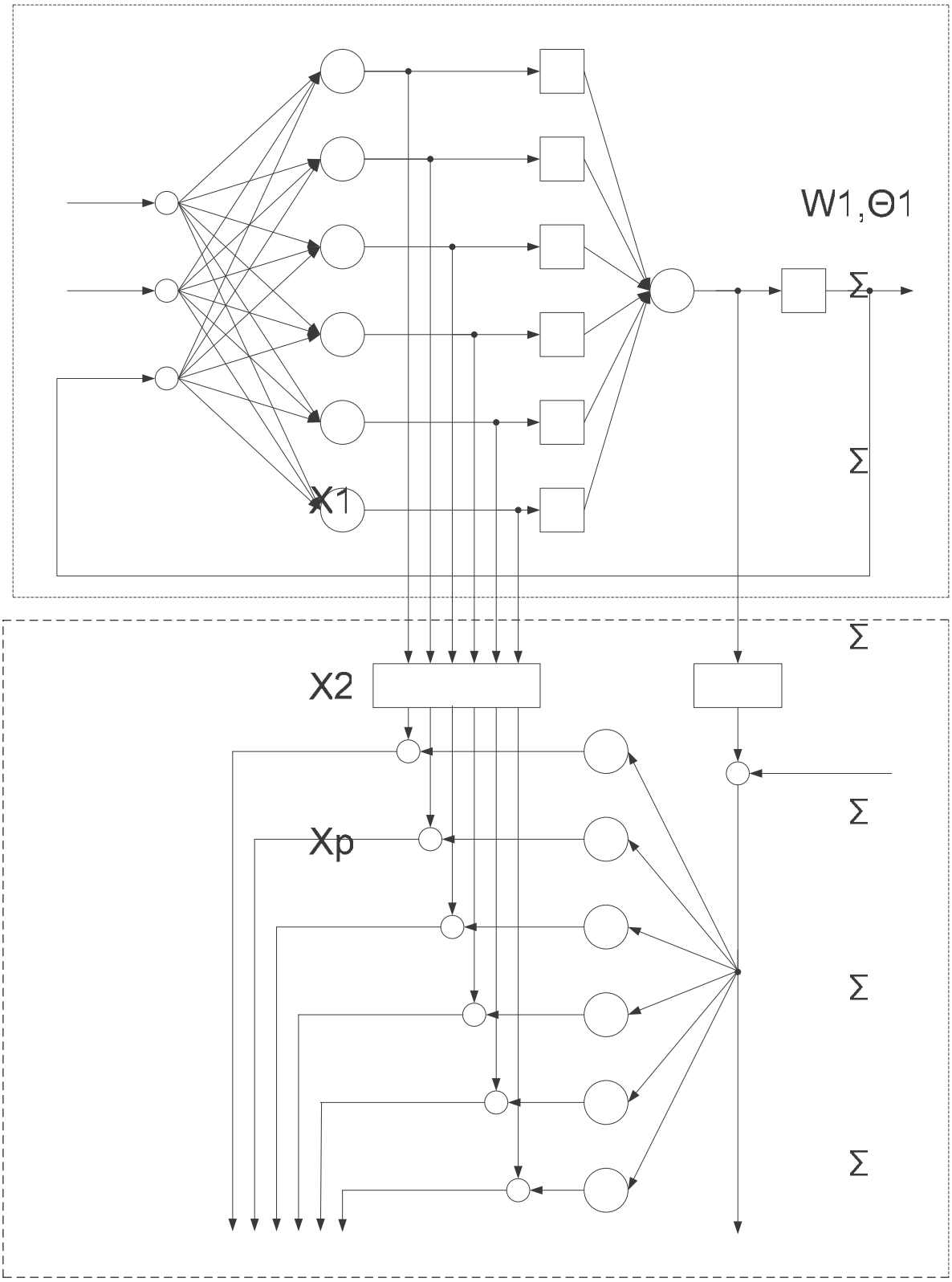
gdje P_{el} označava radnu snagu sinkronog generatora, a K, k_1, k_2 i k_3 su konstante korištene za skaliranje.

Izraz (5.11) te izrazi (5.4) i (5.6) koji ga koriste predstavljaju konačne izraze pomoću kojih se odvija učenje neuronske mreže, tj. podešavanje vrijednosti težinskih koeficijenata i pomaka pojedinih neurona. Na slici 5.2. prikazan je implementirani regulator napona u postojeći sustav regulacije napona sinkronog generatora. Vidljivo je da se regulator zasnovan ne neuronskoj mreži nalazi na mjestu regulatora napona te da on formira referentnu vrijednost struje uzbude. Ulazni signali u neuronsku mrežu su referentna vrijednost napona generatora i trenutna vrijednost napona generatora te zakašnjeli signal izlaza iz neuronske mreže. U kriterijsku funkciju (5.11) potrebno je dovesti osim stvarne i referentne vrijednosti napona generatora i signal devijacije i derivacije radne snage.



Slika 5.2. Strukturna blokovska shema neuronskog sustava regulacije uzbude sinkronog generatora

Implementirana struktura neuronske mreže s cjelokupnim povratnim prostiranjem prikazana je na slici 5.3. Neuronske mreža sastoji se od dva sloja s šest neurona u skrivenom i jednim neuronom u izlaznom sloju. U prvoj fazi signali kroz mrežu prolaze od ulaza prema izlazima. Na temelju kriterijske funkcije generira se signal pogreške kojim neuronska mreža podešava svoje težinske koeficijente. Za podešavanje težinskih koeficijenata neuronske mreže koristi se algoritam povratnog prostiranja. Podešavanje koeficijenata mreže ide smjerom suprotnim od ulaznih signala tako da se prvo podešavaju koeficijenti u izlaznom sloju pa zatim u skrivenom sloju. Budući da je mreža dinamička i radi u *on-line* režimu učenje se neprestano odvija.



Slika 5.3. Implementirana neuronska mreža s algoritmom povratnog prostiranja

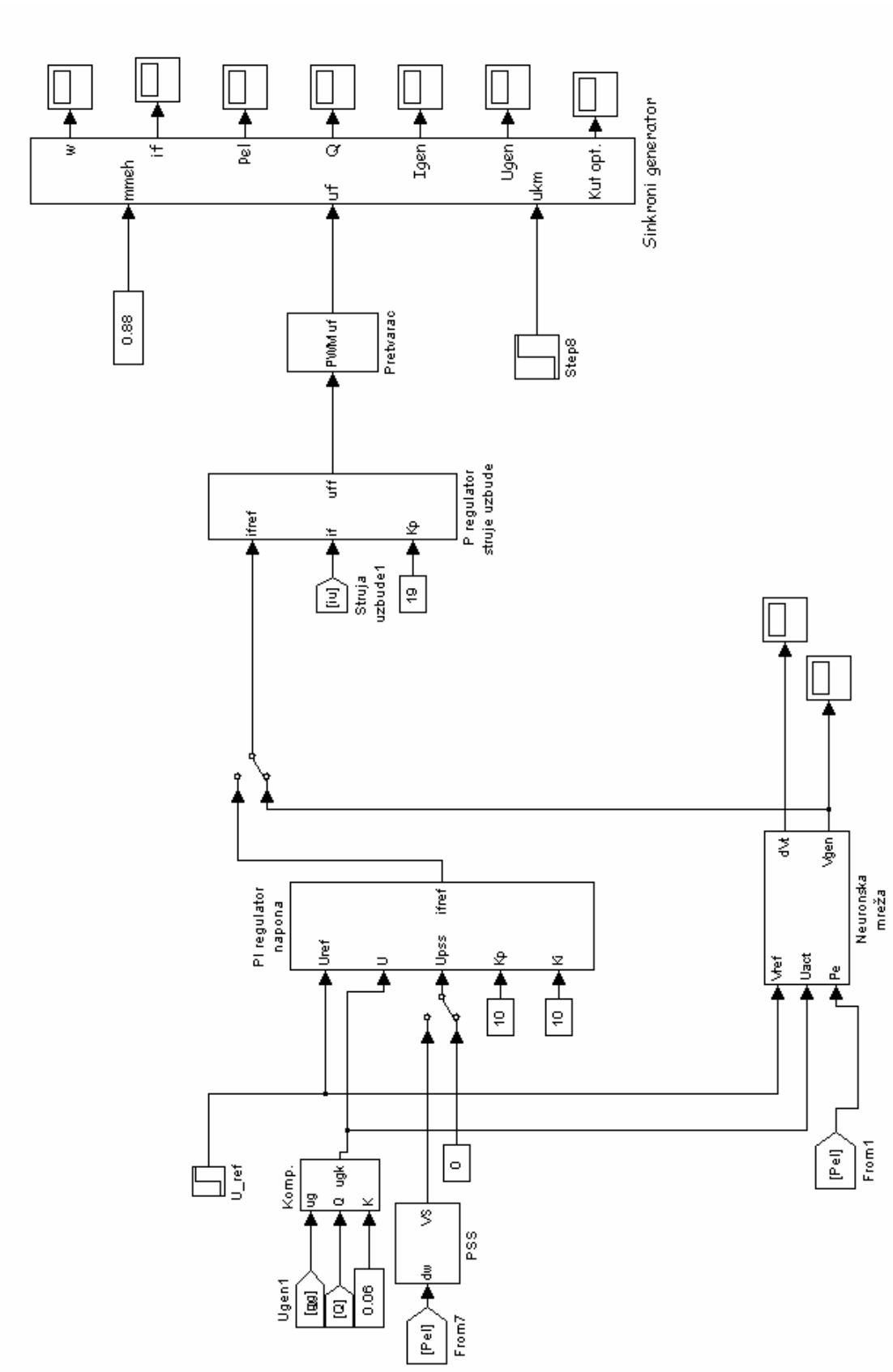
6 SIMULACIJSKI MODEL KOMPONENATA I SUSTAVA REGULACIJE UZBUDE GENERATORA

Simulacijski model sustava za regulaciju napona sinkronog generatora s klasičnom regulacijskom strukturom zasnovanim na PI regulatorima načinjen je na Zavodu za Elektrostrojarstvo i automatizaciju, te je preuzet iz [6],[7]. Usporedba simuliranog sustava s realnim sustavom regulacije napona sinkronog generatora prikazana je u [6]. Osim klasične regulacijske strukture, modeliran je i sustav upravljanja uzbudom sinkronog generatora s dinamičkom neuronskom mrežom s tri ulaza, šest neurona u skrivenom sloju i jednim neuronom u izlaznom sloju [8].

Simulacija sustava za regulaciju napona sinkronog generatora izvedena je u programskom paketu *Matlab Simulink*. Simulacija se sastoji od sljedećih osnovnih dijelova:

- čoper,
- pogon sinkronog generatora,
- sinkroni generator,
- regulator struje uzbude generatora,
- regulator napona generatora,
- dinamička neuronska mreža.

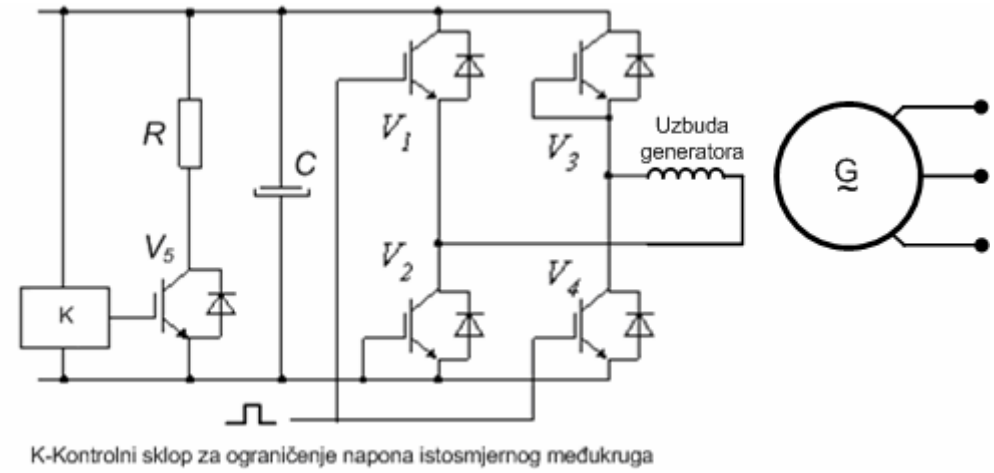
Simulacijski model sustava za regulaciju napona sinkronog generatora u programskom paketu *Matlab Simulink* prikazan je na slici 6.1**.** Referencu za regulator struje uzbude formira PI regulator napona ili dinamička neuronska mreža ovisno o položaju preklopke koja se nalazi iza regulatora napona odnosno neuronske mreže.



Slika 6.1. Model sustava za regulaciju napona sinkronog generatora u programskom paketu *Matlab Simulink*

6.1 Simulacijski model čopera

Digitalni sustav pulsno širinski moduliranim signalom upravlja IGBT sklopkama. U trenutku kada se sklopke V_1 i V_4 sa slike 6.2 uključe, napon uzbude je jednak naponu istosmjernog međukruga. Kada se sklopke isključe, struja nastavlja teći kroz diode sklopki V_2 i V_3, i tada je napon uzbude jednak negativnom naponu međukruga.



Slika 6.2. Blok shema IGBT čopera sustava uzbude sinkronog generatora

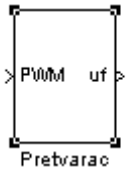
Napon uzbude je negativan sve dok struja uzbude ne padne na nulu. Iz toga slijedi da sve dok širina impulsa ne prijeđe 50%, srednja vrijednost napona uzbude će biti nula. Izlazni teret čopera je uzbudni namot induktivnog karaktera, vremenske konstante 550 ms. Frekvencija pulsno širinski moduliranog signala je 450 Hz. Iz toga slijedi da je struja uzbude praktički kontinuirana pa se čoper može modelirati pojednostavljenim kvazistacionarnim modelom koji napon uzbude opisuje istosmjernim naponom jednakom srednjoj vrijednosti stvarnog pulsno širinski moduliranog napona uzbude. Stanovito odstupanje postoji budući da je napon istosmjernog međukruga nešto manji kada se kondenzator prazni u odnosu na trenutak kada se puni. Eksperimentalno dobiveni odnos srednje vrijednosti izlaznog napona čopera i širine impulsa upravljanja tranzistora je prikazan na slici 6.3.

Pokus izveden u [6] pokazuje da se promjenjivost napona istosmjernog međukruga može zanemariti, te da simulacija s konstantnim naponom međukruga u iznosu od 150V u potpunosti zadovoljava.

Simulacijski model čopera opisuje vezu između širine impulsa upravljanja tranzistora čopera i izlaznog napona iz čopera.

Slika 6.3 Odnos izlaznog napona čopera i perioda vođenja tranzistora

Simulacijski modela čopera u programskom paketu *Matlab Simulink* je prikazan na slici 6.4.



Slika 6.4. Simbolički prikaz čopera u programskom paketu *Matlab Simulink*

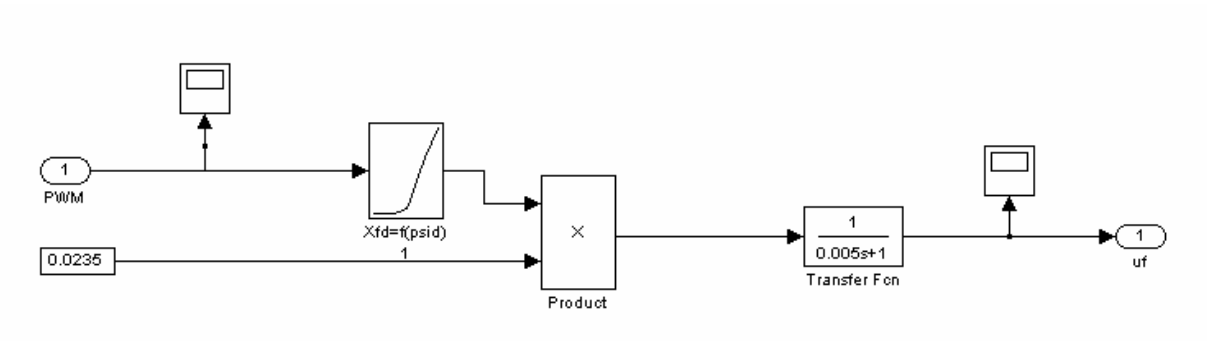
Unutarnja struktura modela čopera prikazana je na slici 6.5. Blok čopera ima jedan ulazni signal:

- Iznos širine impulsa upravljanja tranzistorima čopera *(PWM)*

Izlaz iz bloka modela čopera je napon uzbude *(uf)* koji je uveden u model sinkronog generatora.

U model čopera je uključena nelinearnost odnosa između širine impulsa upravljanja tranzistora čopera i napona uzbude u bloku *u=f(D)*, a napon istosmjernog međukruga je konstantan. Model čopera simulira jednokvadrantni čoper. Član prvog reda na izlazu iz bloka opisuje kašnjenje izlaznog napona od jednog koraka diskretizacije. To kašnjenje nastaje jer postoji kašnjenje između trenutka kada regulator u digitalnom sustavu odredi referentnu vrijednost perioda vođenja do trenutka kada se odgovarajući napon pojavi na uzbudi generatora. To vrijeme je određeno frekvencijom pulsno širinske modulacije jer se ona izvodi na nižoj frekvenciji nego algoritmi regulacije.

Frekvencija pulsno širinske modulacije je 450Hz pa bi izlazna vremenska konstanta trebala biti 2ms. Zbog problema pojavljivanja algebarske petlja pri određenim uvjetima simulacije, vremenska konstanta je povećana na 5ms.



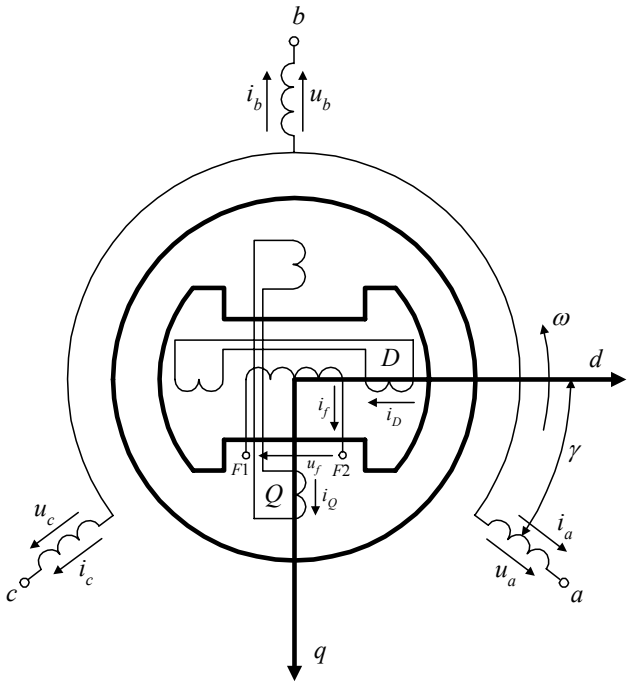
Slika 6.5. Simulacijski model čopera u programskom paketu *Matlab Simulink*

6.2 Simulacijski model pogona sinkronog generatora

U laboratorijskom modelu pogonski stroj sinkronog generatora je par istosmjernih nezavisno uzbuđenih motora koji se napajaju iz jednosmjernog tiristorskog usmjerivača . Tiristorski usmjerivač posjeduje regulacijski sustav izveden u analognoj tehnici. Regulacijska struktura realizirana u tiristorskom usmjerivaču sastoji se od unutarnjeg kruga regulacije struje, te vanjskog nadređenog kruga regulacije brzine. Ovakav sustav regulacije posjeduje mogućnost da se isključi djelovanje kruga regulacije brzine. To svojstvo se koristi kod rada sinkronog generatora na mreži kada se u trenutku sinkronizacije generatora mijenja regulacija struktura. Prilikom rada na mreži referenca struje armature istosmjernih motora određuje predanu radnu snagu sinkronog generatora u mrežu. Krug regulacije struje osigurava konstantan moment istosmjernih motora. Moment pogonskog stroja izravno se unosi u simulaciju kao konstantna veličina.

6.3 Simulacijski model sinkronog generatora

Sinkroni stroj se u općoj teoriji strojeva predstavlja s tri fazna namota na statoru i tri namota na rotoru. Namote na rotoru čine uzbudni namot i dva ekvivalentna prigušna namota (Slika 6.6.).



Slika 6.6. Namoti u sinkronom stroju i rotirajuće d-q osi

Matematički model sinkronog generatora je temeljen na Parkovim transformacijama. To su naponske jednadžbe zamišljenih armaturnih krugova u uzdužnoj *d* i poprečnog *q* osi. Te dvije osi se dobivaju linearnom transformacijom faznih armaturnih fizikalnih veličina *a*, *b* i *c*.

Takav način modeliranja generatora je u širokoj upotrebi. Složeniji modeli zahtijevaju poznavanje dodatnih parametara stroja i prikladniji su za detaljna istraživanja stroja. Budući da su u ovom radu od ključnog interesa pitanja vezana za regulaciju uzbude generatora, ovakav model zadovoljava po složenosti. Zasićenje u željezu je opisano u *d* osi.

Do matematičkog modela se dolazi transformacijom sustava naponskih jednadžbi iz faznih *abc* koordinata u *dq* koordinate. Pretpostavlja se da su naponi simetrično raspoređeni, da imaju jednake parametre i da u zračnom rasporu ima samo jedan harmonik polja. Početne jednadžbe u ovom modelu su diferencijalne jednadžbe u *dq* koordinatama koje su izražene u relativnim veličinama (vrijeme je izraženo apsolutno)[9]:

$$-u_d = r \cdot i_d + \frac{1}{\omega_s} \cdot \frac{d\psi_d}{dt} + \omega \cdot \psi_q \,, \qquad (6.1)$$

$$-u_q = r \cdot i_q + \frac{1}{\omega_s} \cdot \frac{d\psi_q}{dt} - \omega \cdot \psi_d \,, \qquad (6.2)$$

$$u_u = r_u \cdot i_u + \frac{1}{\omega_s} \cdot \frac{d\psi_u}{dt} \,, \qquad (6.3)$$

$$0 = r_D \cdot i_D + \frac{1}{\omega_s} \cdot \frac{d\psi_D}{dt}, \qquad (6.4)$$

$$0 = r_Q \cdot i_Q + \frac{1}{\omega_s} \cdot \frac{d\psi_Q}{dt}, \qquad (6.5)$$

Jednadžbe koje definiraju odnos ulančanih tokova i struja su:

$$\psi_d = x_d \cdot i_d + x_{ud} \cdot i_u + x_{dD} \cdot i_D, \qquad (6.6)$$

$$\psi_q = x_q \cdot i_q + x_{qQ} \cdot i_Q, \qquad (6.7)$$

$$\psi_u = x_{ud} \cdot i_d + x_u \cdot i_u + x_{uD} \cdot i_D, \qquad (6.8)$$

$$\psi_D = x_{dD} \cdot i_d + x_{uD} \cdot i_u + x_D \cdot i_D, \qquad (6.9)$$

$$\psi_Q = x_{qQ} \cdot i_q + x_Q \cdot i_Q. \qquad (6.10)$$

Jednadžbe (6.6) do (6.10) se uvrštavaju u jednadžbe (6.1) do (6.5). Sređivanjem jednadžbi dobiva se matrični oblik [15]:

$$\begin{vmatrix} \frac{di_d}{dt} \\ \frac{di_1}{dt} \\ \frac{di_D}{dt} \end{vmatrix} = \omega_s \cdot \begin{vmatrix} x_d & x_{ud} & x_{dD} \\ x_{ud} & x_u & x_{uD} \\ x_{dD} & x_{uD} & x_D \end{vmatrix}^{-1} \cdot \begin{vmatrix} A_d \\ B_d \\ C_d \end{vmatrix}, \qquad (6.11)$$

$$\begin{vmatrix} \frac{di_q}{dt} \\ \frac{di_Q}{dt} \end{vmatrix} = \omega_s \cdot \begin{vmatrix} x_q & x_{qQ} \\ x_{qQ} & x_Q \end{vmatrix}^{-1} \cdot \begin{vmatrix} A_q \\ B_q \end{vmatrix}, \qquad (6.12)$$

gdje su:

$$A_d = -u_d - \omega \cdot \psi_q - r \cdot i_d, \qquad (6.13)$$

$$B_d = u_u - r_u \cdot i_u, \qquad (6.14)$$

$$C_d = -r_D \cdot i_D, \qquad (6.15)$$

$$A_q = -u_q + \omega \cdot \psi_d - r \cdot i_q, \qquad (6.16)$$

$$B_q = -r_Q \cdot i_Q. \qquad (6.17)$$

Proračun parametara A_d, B_d, C_d, A_q, B_q u simulacijskom modelu je prikazan u dodatku B.

Za simulaciju potrebno je izračunati inverzne matrice iz jednadžbi (6.11) i (6.12). Dobivaju se sljedeći članovi matrice reaktancija *d* osi kao inverzna matrica matrice iz jednadžbe (6.11):

$$Y_d(3,3) = \frac{K}{L}, \tag{6.18}$$

$$Y_d(3,2) = -\frac{M}{L}, \tag{6.19}$$

$$Y_d(3,1) = \frac{-\frac{x_{dD}}{x_d} \cdot K + \frac{x_{ud}}{x_d} \cdot M}{L}, \tag{6.20}$$

$$Y_d(2,2) = \frac{x_d}{I} - N \cdot Y_d(3,2), \tag{6.21}$$

$$Y_d(2,1) = -\frac{x_{ud}}{I} - N \cdot Y_d(3,1), \tag{6.22}$$

$$Y_d(1,1) = \frac{1}{x_d} - \frac{x_{dD}}{x_d} \cdot Y_d(3,1) - \frac{x_{uD}}{x_d} \cdot Y_d(2,1), \tag{6.23}$$

$$Y_d(2,3) = Y_d(3,2), \tag{6.24}$$

$$Y_d(1,3) = Y_d(3,1), \tag{6.25}$$

$$Y_d(1,2) = Y_d(2,1). \tag{6.26}$$

Proračun parametara Y_d u simulacijskom modelu je prikazan u dodatku B .

gdje su:

$$K = x_u - \frac{x_{ud}^{\ 2}}{x_d}, \tag{6.27}$$

$$L = (x_D - \frac{x_{dD}^{\ 2}}{x_d}) \cdot (x_u - \frac{x_{ud}^{\ 2}}{x_d}) - (x_{uD} - \frac{x_{ud} \cdot x_{dD}}{x_d})^2, \tag{6.28}$$

$$M = x_{uD} - \frac{x_{ud} \cdot x_{dD}}{x_d}, \tag{6.29}$$

$$N = \frac{M}{K}, \tag{6.30}$$

$$I = x_u \cdot x_d - x_{ud}^{\ 2}. \tag{6.31}$$

Proračun parametara *K*, *L*, *M*, *N*, *I*, u simulacijskom modelu je prikazan u dodatku B9.

Dobivaju se sljedeći članovi matrice reaktancija *q* osi kao inverzna matrica matrice iz jednadžbe (6.12):

$$Y_q(1,1) = \frac{1}{x_q} \cdot (1 + \frac{x_{qQ}^{\ 2}}{P}), \tag{6.32}$$

$$Y_q(1,2) = -\frac{x_{qQ}}{P}, \tag{6.33}$$

$$Y_q(2,1) = Y_q(1,2), \tag{6.34}$$

$$Y_q(2,2) = \frac{x_q}{P}. \tag{6.35}$$

Proračun parametara Y_q u simulacijskom modelu je prikazan u dodatku B 11.

gdje je:

$$P = x_q \cdot x_Q - x_{qQ}^{\ 2}. \tag{6.36}$$

Proračun parametara *P* u simulacijskom modelu je prikazan u dodatku B9.

Sustav jednadžbi je napisan u relativnim, jediničnim vrijednostima. Prednost takvog zapisa je lakša usporedba ponašanja sinkronih strojeva svih raspona snaga i brzina vrtnje. Bazne vrijednosti pojedinih veličina su dane u dodatku B1.

Za simulaciju potrebno je poznavati sljedeće parametre generatora: r, r_u, r_D, r_Q, x_d, x_{ud}, x_{dD}, x_u, x_{uD}, x_D, x_q, x_{qQ}, x_Q. Standardni parametri strojeva koji su najčešće na raspolaganju su: x_d, x_q, x_l, r, T_m, x_d', x_d'', x_q'', T_d'', T_q''. Veza između ta dvije skupine parametara je dana formulama[11]:

$$x_{ud} = x_d - x_l, \tag{6.37}$$

$$x_{uD} = x_{ud}, \tag{6.38}$$

$$x_u = \frac{(x_d - x_l)^2}{x_d - x_d'}, \tag{6.39}$$

$$x_D = x_{ud} + \frac{(x_d' - x_l) \cdot (x_d'' - x_l)}{x_d' - x_d''}, \tag{6.40}$$

$$x_{qQ} = x_q - x_l, \tag{6.41}$$

$$x_Q = \frac{(x_q - x_l)^2}{x_q - x_q''}, \tag{6.42}$$

$$r_u = \frac{x_u}{T_{d0} \cdot \omega_s}, \tag{6.43}$$

$$r_D = \frac{(x_d' - x_l)^2}{x_d' - x_d''} \cdot \frac{x_d''}{x_d'} \cdot \frac{1}{T_d'' \cdot \omega_s}, \tag{6.44}$$

$$r_Q = \frac{(x_q - x_l)^2}{x_q - x_q''} \cdot \frac{x_q''}{x_q} \cdot \frac{1}{T_q'' \cdot \omega_s}, \tag{6.45}$$

Jednadžbe gibanja agregata su:

$$\frac{d\varphi}{dt} \cdot \frac{1}{\omega_s} = \omega, \tag{6.46}$$

$$\frac{d\vartheta}{dt} \cdot \frac{1}{\omega_s} = 1 - \omega, \tag{6.47}$$

$$\frac{d\omega}{dt} = \frac{1}{T_m} \cdot (m_{meh} + m_{elm}), \tag{6.48}$$

$$m_{elm} = \psi_q \cdot i_d - \psi_d \cdot i_q. \tag{6.49}$$

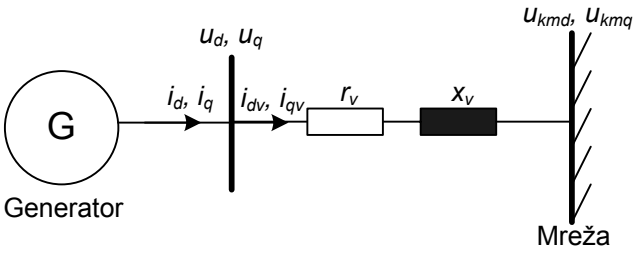
Simuliran je generator spojen na krutu mrežu preko blok transformatora i spojnog voda. Otpor r_v i reaktancija x_v predstavljaju ukupan otpor i ukupnu reaktanciju između generatora i krute mreže kao što je prikazano na slici 6.7. Vrijede sljedeće jednadžbe[10]:

$$u_{dv} = i_{dv} \cdot r_v + \frac{x_v}{\omega_s} \cdot \frac{di_{dv}}{dt} + \omega \cdot x_v \cdot i_{qv} + u_{kmd}, \tag{6.50}$$

$$u_{qv} = i_{qv} \cdot r_v + \frac{x_v}{\omega_s} \cdot \frac{di_{qv}}{dt} - \omega \cdot x_v \cdot i_{dv} + u_{kmq}, \tag{6.51}$$

$$u_{kmd} = u_{km} \cdot \sin\vartheta, \tag{6.52}$$

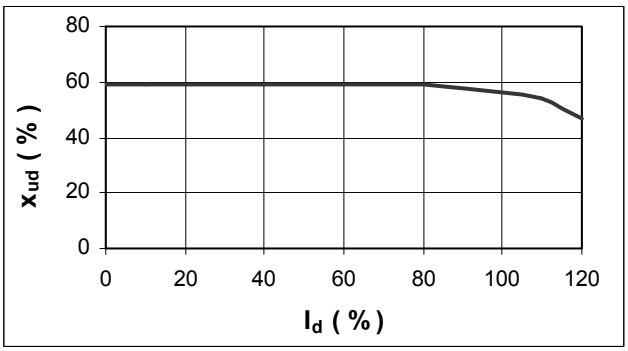
$$u_{kmq} = u_{km} \cdot \cos\vartheta. \tag{6.53}$$



Slika 6.7. Generator spojen na krutu mrežu preko otpora r_v i reaktancije x_v

Kut ϑ je kut opterećenja generatora, a u_{km} je iznos napona mreže.

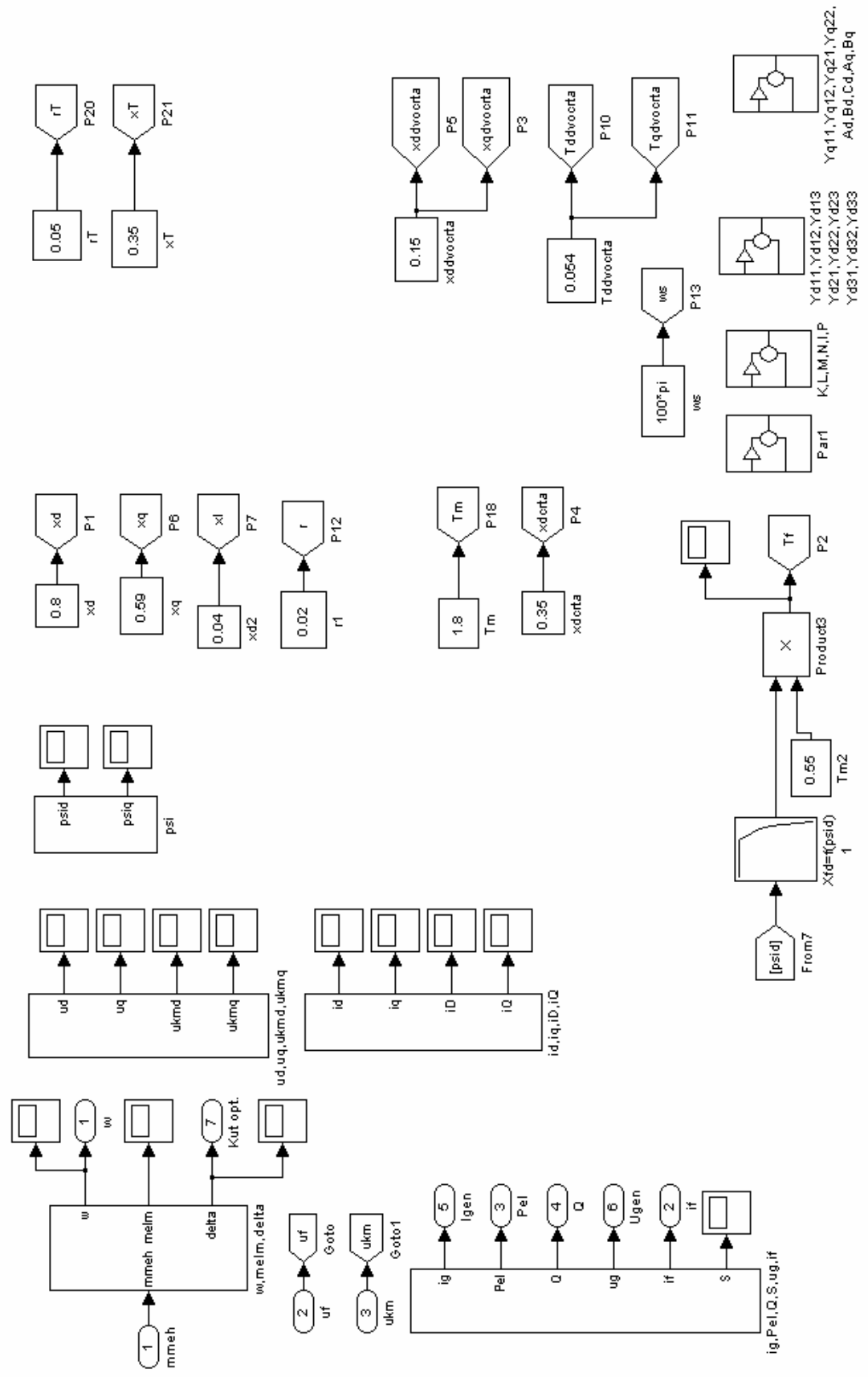
Zasićenje generatora ima posljedicu promjene parametara generatora što izravno utječe na dinamičko i stacionarno stanje generatora. Zasićenje se uvodi preko snimljene karakteristike praznog hoda, preko parametra x_{ud}, jer uz konstantnu brzinu vrtnje taj parametar predstavlja izravnu vezu između uzbudne struje i induciranog napona generatora [12]. Parametar x_{ud} mora biti promjenjiv tako da se dobije ista funkcijska povezanost između struje uzbude i induciranog napona (ulančanog toka) dobivena pokusom praznog hoda. Zasićenje se uvodi samo u *d* os generatora, a parametri *q* osi ostaju konstantni. Karakteristika promjene x_{ud} dobivena iz pokusa praznog hoda prikazana je na slici 6.8.



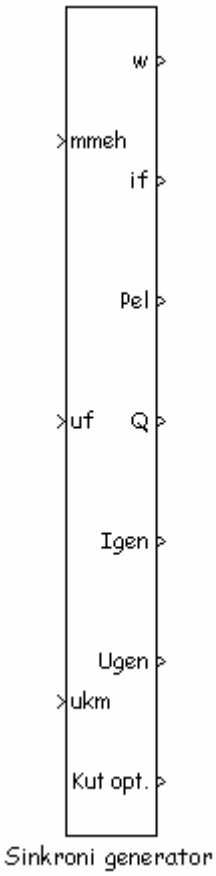
Slika 6.8. Karakteristika promjene reaktancije x_{ud} s promjenom *d* komponente struje generatora

Pretpostavlja se da je rasipna reaktancija x_l generatora sa zasićenjem jednaka nezasićenoj rasipnoj reaktanciji [12].

Na osnovu gore navedenih jednadžbi i razmatranja realiziran je simulacijski model sinkronog generatora u programskom paketu "MATLAB-SIMULINK" koji je prikazan na slici 6.9. Blokovski prikaz generatora dan je na slici 6.10.



Slika 6.9. Simulacijski model sinkronog generatora u programskom paketu *Matlab Simulink*u

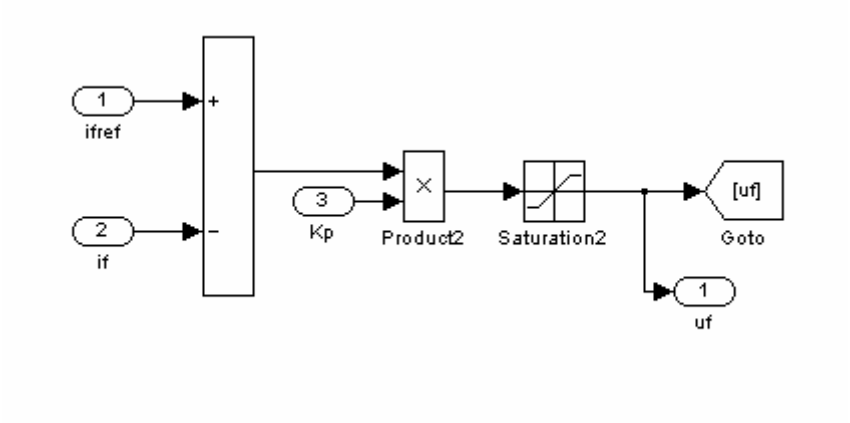


Slika 6.10. Blokovski prikaz sinkronog generatora u programskom paketu *Matlab Simulink*

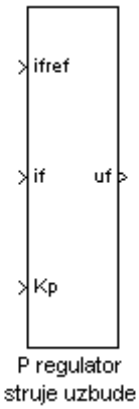
Ulaz u blok sinkronog generatora su mehanički moment *(mmeh)*, napon uzbude *(uf)* i napon krute mreže *(ukm)*, a izlazi su brzina vrtnje *(w)*, struja uzbude *(if)*, radna snaga *(Pel)*, jalova snaga *(Q)*, struja *(Igen)*, napon *(Ugen)* i kut opterećenja generatora *(Kut opt.)*.

6.4 Simulacijski model kruga regulacije struje uzbude sinkronog generatora

Simulacijski model regulatora struje uzbude, u programskom paketu *Matlab Simulink* prikazan na slici 6.11. Regulator struje uzbude ima P karakteristiku. Blokovski je prikazan regulator struje uzbude na slici 6.12.



Slika 6.11. Simulacijski model regulatora struje uzbude u programskom paketu M*atlab Simulink*



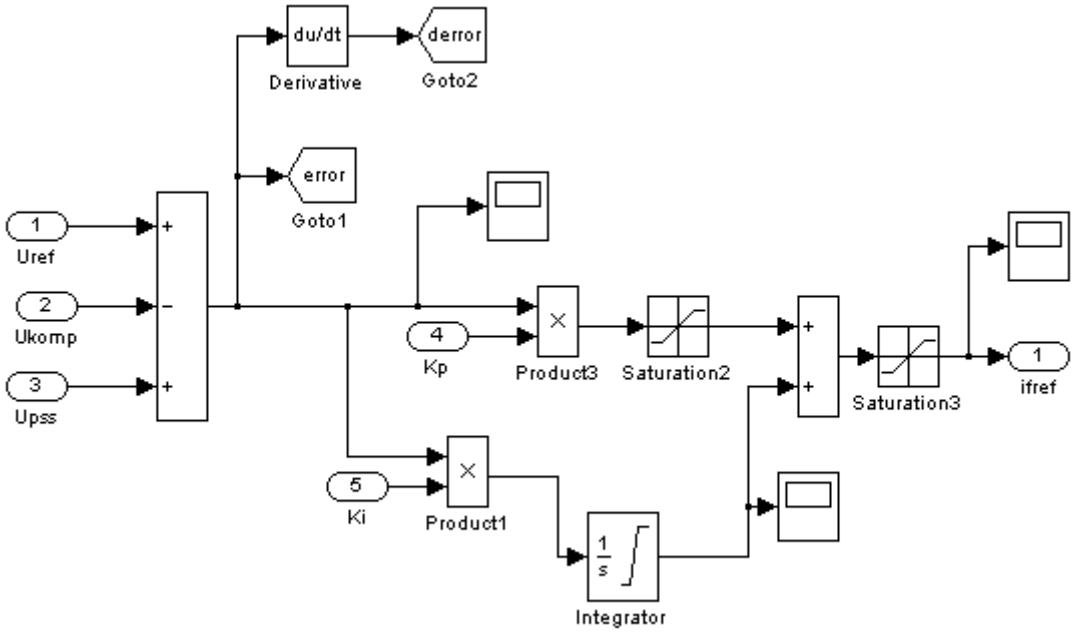
Slika 6.12. Blokovski prikaz regulatora struje u programskom paketu M*atlab Simulink*

Ulazni signali u blok regulatora struje uzbude su referenca struje uzbude *(ifref)* i struja uzbude *(if)*, a izlaz je širina impulsa upravljanja tranzistora čopera *(uf)*. Maksimalna vrijednost izlaza iz regulatora je ograničena na 100%, a minimalna vrijednost izlaza iz regulatora je podešena na 0%. Pojačanje regulatora K_p je podešeno na 3.

6.5 Simulacijski model kruga regulacije napona sinkronog generatora

Simulacijski model regulatora napona, u programskom paketu *Matlab Simulink* prikazan na slici 6.13. Regulator napona ima PI karakteristiku. Blokovski je prikazan na slici 6.14. Ulaz u blok regulatora napona su referenca napona generatora *(Uref)*, kompenzirani napon generatora po jalovoj snazi *(Ukomp)*, stabilizacijski signal (*Upss*), proporcionalno pojačanje *(Kp)* i integralno pojačanje (*Ki)* koje predstavlja inverznu vrijednost vremenske konstante integratora. Izlaz iz bloka je referenca struje uzbude generatora (*ifref*). Pojačanja proporcionalnog K_p i integralnog člana K_i regulatora napona su podešena na 10. Signal

proporcionalnog dijela ograničen je na (0 %, 200 %), signal integralnog dijela na (-200 %, 200 %), a izlazni signal iz regulatora na (0 %, 200%).



Slika 6.13. Simulacijski model regulatora napona u programskom paketu *Matlab Simulink*

Slika 6.14. Simbolički prikaz regulatora napona u programskom paketu *Matlab Simulink*

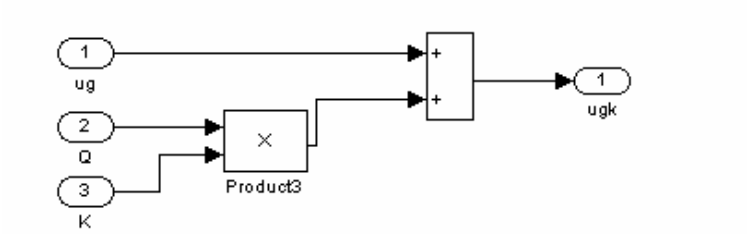
6.6 Kompenzacija napona po jalovoj snazi

Kompenzacija napona generatora prema jalovoj snazi uključuje se kada je generator spojen u elektroenergetski sustav i osigurava potrebnu karakteristiku napona generatora s porastom jalove snage. Kompenzirana vrijednost napona generatora računa se prema izrazu:

$$u_{gk} = u_g + q \cdot k \ . \tag{6.54}$$

gdje je u_g napon generatora, q jalova snaga generatora, te k konstanta kojom se osigurava potrebna karakteristika napona generatora s porastom jalove snage.

Simulacijski model kompenzacije jalove snage u programskom paketu *Matlab Simulink* prikazan je na slici 6.15. Blokovski ova struktura je prikazana na slici 6.16. Konstanta kojom se osigurava nagib karakteristike napona generatora k iznosi 0.06.

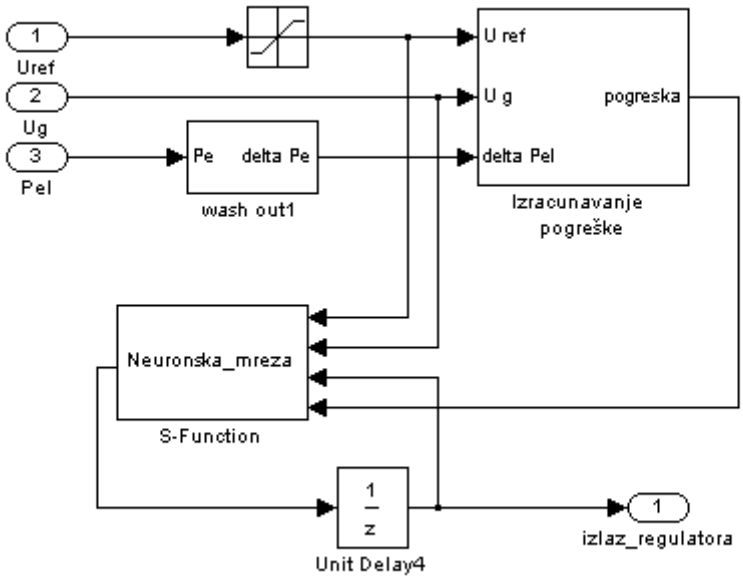


Slika 6.15. Simulacijski model kompenzacije jalove snage u programskom paketu *Matlab Simulink*

Slika 6.16. Blok kompenzacije jalove snage u programskom paketu *Matlab Simulink*

6.7 Simulacijski model regulatora napona sinkronog generatora zasnovanog na neuronskoj mreži

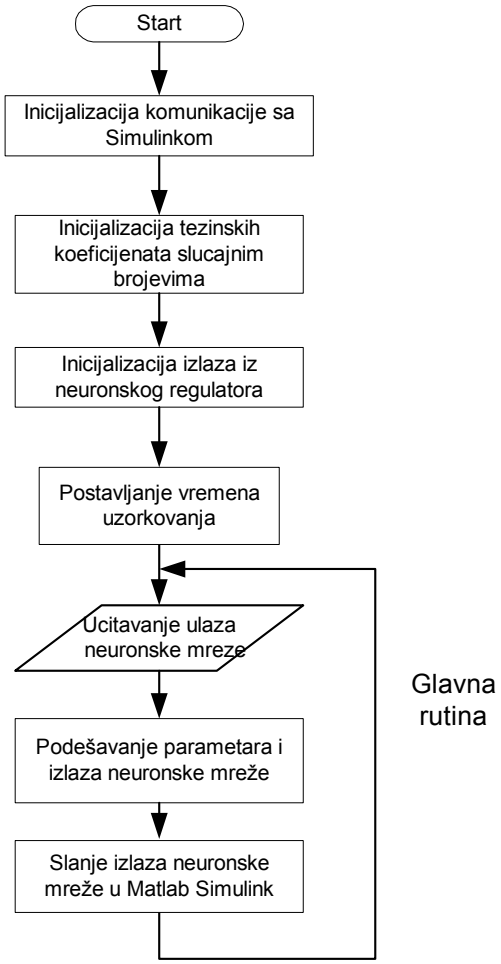
Regulator napon sinkronog generatora zasnovan na neuronskoj mreži (detaljno opisan u poglavlju 5) sastoji se od dva dijela neuronske mreže i dijela za *on-line* učenje kako je to prikazano na slici 6.17.



Slika 6.17. Simulacijski model regulatora napona zasnovan na neuronskoj mreži realiziran u Programskom paketu Matlab Simulink

Neuronska mreža zbog svoje kompleksnosti realizirana je pomoću S-funkcije. Ovako realizirana neuronska mreža je u programskom paketu *Matlab Simulink* ugrađena kao korisnički definirana funkcija. Blok S-funkcije ima četiri ulaza: referentnu vrijednost napona generatora U_{ref}, trenutnu vrijednost napona generatora U_g, izlaz iz mreže u prethodnom trenutku i pogrešku (gradijent kriterijske funkcije). Izlaz iz S-funkcije ujedno je i izlaz iz neuronskog regulatora koji pojačan ulazi u regulator struje uzbude sinkronog generatora .

Programski paket *Matlab* nudi gotov predložak kôda S-funkcije pisane u programskom jeziku *C* koji je potrebno prilagoditi vlastitim zahtjevima. Struktura S-funkcije kojom je opisana neuronska mreža pisana je u programskom jeziku *C* te je prikazana na slici 6.18.



Slika 6.18. Struktura S-funkcije u programskom jeziku C kojom se opisuje neuronska mreža

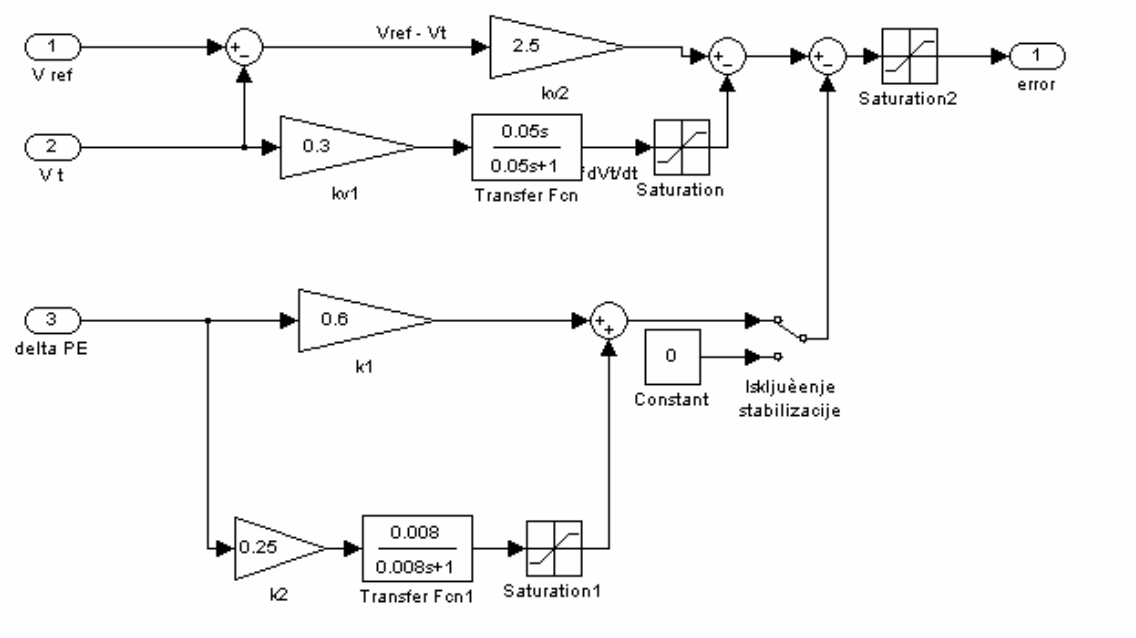
Za uspostavljanje komunikacije sa *Matlab Simulink*-om potrebno je definirati broj ulaznih i izlaznih signala. Definirana su četiri ulazna signala i jedan izlazni signal. Definirano je 40 radnih varijabli koje se koriste kao parametri neuronske mreže (težinski koeficijenti i pragovi osjetljivosti neurona u skrivenom i izlaznom sloju). Početne vrijednosti parametara neuronske mreže postavljene su na neke male slučajne vrijednosti u rasponu (-0.5, 0.5), što predstavlja standardni postupak inicijalizacije neuronske mreže pazeći prilikom toga na mogućnost zasićenja pojedinih neurona.

Izlaz neuronskog regulatora potrebno je početno postaviti na određenu stabilnu vrijednost. U tu svrhu provedeno je inicijalno *off-line* učenje neuronske mreže uz konstantno postavljene očekivane vrijednosti napona generatora i reference napona generatora. Vrijednost izlaza pri tome je početno podešena na malu slučajnu vrijednost (-0.5, 0.5). Nakon nekoliko iteracija neuronska mreža generira stabilan izlazni signal koji se koristi kao početna vrijednost izlaza iz neuronske mreže.

Nakon postavljanja željenog vremena uzorkovanja počinje se izvršavati glavna rutina S-funkcije koja podešava parametre neuronske mreže na temelju izraza navedenih u poglavlju (5). Glavna rutina S-funkcije izvršava se tijekom simulacije u programskom paketu *Matlab Simulink* sa zadanim vremenom uzorkovanja.

S obzirom da je neuronska mreža realizirana pisanjem programskog kôda, podešavanje parametara neuronske mreže nemoguće je izvesti paralelno tj. istovremeno, već se koristi sekvencijalna metoda podešavanja parametara [4],[5]. Ova metoda podešava neuronske parametre jedan iza drugoga. Podešavanje počinje s prvim neuronom u prvom sloju, a završava s zadnjim neuronom u zadnjem sloju. Izlaz pojedinog neurona drži se konstantnim sve dok se ne počne novi krug podešavanja parametara. Ovaj način podešavanja parametara daleko je sporiji od onog zamišljenog u teoriji, no postignuti rezultati su zadovoljavajući.

Gradijenta kriterijske funkcije, tj. pogreške računa se prema izrazima navedenim u 5. poglavlju, a na slici 6.17. to je predstavljeno simulacijskim blokom *Izračunavanje pogreška*, čija je strukturna u programskom paketu *Matlab Simulink* prikazana na slici 6.19. Kao ulaz u blok *Izračunavanje pogreške* koriste se trenutna vrijednost iznosa napona, referenca napona i promjena vrijednosti radne snage. Sklopka *Isključenje stabilizacije* u položaju 0 isključuje funkciju stabilizatora elektroenergetskog sustava, tj. neuronski regulator napona tada sadrži samo funkciju regulatora napona sinkronog generatora.



Slika 6.19. Simulacijski model bloka *Izračunavanje pogreške*
u programskom paketu *Matlab Simulink*

Promjene vrijednosti radne snage dobivena je propuštanjem signala radne snage kroz realni derivacijski član koji ima prijenosnu funkciju oblika:

$$F(s) = k_p \frac{T_p \cdot s}{T_p \cdot s + 1}. \tag{6.55}$$

Pri tome je vremenska konstanta T_p odabrana tako da ne utječe na frekvenciju njihanja u području od 0.1 do 5 Hz.

Za dobivanje derivacija napona i promjene radne snage generatora korišteni su također realni derivacijski članovi:

$$F(s) = k_1 \frac{T_1 \cdot s}{T_1 \cdot s + 1} \qquad \text{za derivaciju napona,} \tag{6.56}$$

$$F(s) = k_2 \frac{T_2 \cdot s}{T_2 \cdot s + 1} \qquad \text{za derivaciju promjene radne snage.} \tag{6.57}$$

7 IZVEDBE I IMPLEMENTACIJE ALGORITAMA NEURONSKOG UPRAVLJANJA UZBUDOM GENERATORA

Cjelokupni algoritam regulacije uzbude sinkronog generatora realiziran je u programskom jeziku C te je implementiran u procesor za obradu signala. Detalji o realizaciji pojedinih sklopova sustava regulacije te o funkcionalnim dijelovima procesora za obradu signala nalaze se u dodatku C i D.

7.1 PID regulator

PID regulator u kontinuiranom području možemo matematički možemo opisati na sljedeći način:

$$u(t) = K_p e(t) + K_i \int_{\tau=0}^{t} e(\tau) d\tau + K_d \frac{d}{dt}\left[e(t)\right]. \tag{7.2}$$

Gdje su :

- *u(t)* upravljački signal,
- *e(t)* signal greške,
- *t* vrijeme iz kontinuirane domene,
- τ vremenska konstanta integracije,
- *Kp* proporcionalno pojačanje,
- *Ki* integralno pojačanje,
- *Kd* derivacijsko pojačanje.

Budući da je procesor za obradu signala diskretni sustav potrebno je ovaj algoritam transformirati u oblik prikladan za implementaciju u procesor. Postoji više načina da bi se to obavilo ovdje ćemo prikazati način iz literature [26]. Dakle možemo pisati:

- P dio: $$K_p e(t) = K_p e(k) \tag{7.2}$$
- I dio: $$K_i \int_{\tau=0}^{t} e(\tau) d\tau \cong K_i \sum_{i=0}^{k} \frac{h}{2}\left[e(i) + e(i-1)\right] \tag{7.3}$$
- D dio: $$K_d \frac{d}{dt}\left[e(t)\right] \cong K_d \left[\frac{e(k) - e(k-1)}{h}\right] \tag{7.4}$$

Vremenska relacija je predstavljena sljedećom relacijom:

$$t = k \cdot h .$$

Gdje je :

- *h* period uzorkovanja
- *k* indeks diskretizacije: k = 0, 1, 2…..

Iz razloga jednostavnijeg računanja poželjno je redefinirati pojačanja regulatora. Pa uvodimo sljedeće zamjene:

$$K_i' = K_i \frac{h}{2}, \tag{7.5}$$

$$K_d' = \frac{K_d}{h}, \tag{7.6}$$

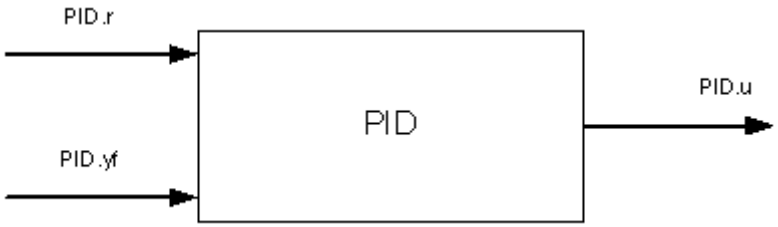
pa uvrštavanjem u izraze (7.2), (7.3) i (7.4) dobijemo slijedeći izraz:

$$u(k) = K_p e(k) + K_i' \sum_{i=0}^{k} \left[e(i) + e(i-1) \right] + K_d' \left[e(k) - e(k-1) \right] . \tag{7.7}$$

Da bi se izbjeglo računanje sume u gornjem izrazu u svakom koraku izvođenja algoritma što bi rezultiralo povećanjem vremena potrebnim za izračunavanje s odmicanjem vremena uvodi se pojam rastuće sume (*running sum*). Ovo ima za direktnu posljedicu smanjenje vremena potrebno za izračunavanje PID regulatora. U konačnici PID regulator je opisan sljedećim jednadžbama koje su i implementirane u sustav a blokovski prikazane na slici 7.1.

$$sum(k) = sum(k-1) + \left[e(k) - e(k-1) \right] \tag{7.8}$$

$$u(k) = K_p e(k) + K_i' sum(k) + K_d' \left[e(k) - e(k-1) \right] \tag{7.9}$$



Slika 7.1. Blokovski prikaz PID regulatora

Alocirana struktura u C kodu :

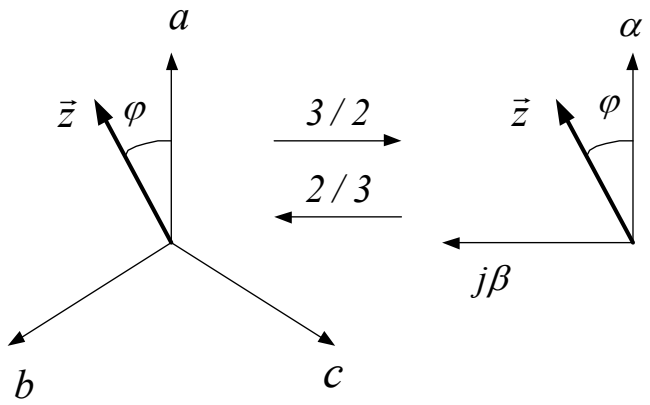
```
typedef struct {
      float32 r;
      float32 u;
      float32 e1;
      float32 yf;
      float32 ui1;
      float32 u_max;
      float32 u_min;
      float32 ui_max;
      float32 Kp;
      float32 Ki;
      float32 Kd;
      int16 y;
      int16 y1;
      int16 y2;
      int16 y3;
      int16 y4;
  } PID;
```

Funkcija za izračun parametara PID regulatora realizirana u C kodu:

```
void PidCtrl1(PID *x)
{
        float32 e, up, ui, ud;
        e = x->r - x->yf;
        up = x->Kp * e;
        ui = x->ui1 + (x->Ki * (e + x->e1));
        if(ui > x->ui_max) ui = x->ui_max;
        else if(ui < -(x->ui_max)) ui = -(x->ui_max);
        ud = x->Kd * (e - x->e1);
        x->e1 = e;
        x->ui1 = ui;

        x->u = up + ui + ud;
        if(x->u > x->u_max) x->u = x->u_max;
        else if(x->u < x->u_min) x->u = x->u_min;

}
```

7.2 Clarkeove transformacije

Za prelaze iz abc koordinatnog sustava u αβ sustav koriste se Clarkeove transformacije (Slika 7.2.). Da bi se odredila amplituda napona ili struje, potrebno je izračunati apsolutnu vrijednost vektora napona ili struje od komponenti iz d i q osi.[11]



Slika 7.2. Clarkova transformacija

Clarkeove transformacije za napone i struje glase, dok je blokovska struktura prikazana na slici 7.3.:

$$u_d = u_A, \tag{7.10}$$

$$u_q = \frac{1}{\sqrt{3}} \cdot \left(u_C - u_B\right), \tag{7.11}$$

$$i_d = i_A, \tag{7.12}$$

$$i_q = \frac{1}{\sqrt{3}} \cdot \left(i_C - i_B\right), \tag{7.13}$$

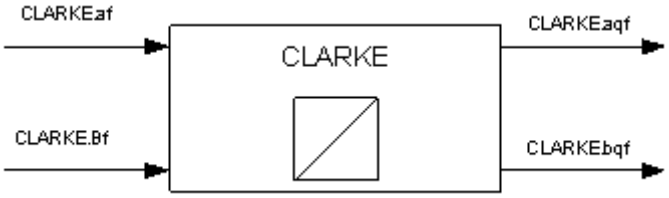
$$u_{gact} = \sqrt{{u_d}^2 + {u_q}^2}, \tag{7.14}$$

$$i_{gact} = \sqrt{i_d{}^2 + i_q{}^2} \ . \tag{7.15}$$

Na temelju trenutnih vrijednosti napona i struja dobivenih Clarkeovom transformacijama računaju se trenutne vrijednosti amplituda radne i jalove snage prema sljedećim izrazima:

$$p_g = \frac{1}{2} \cdot (u_\alpha \cdot i_\alpha + u_\beta \cdot i_\beta), \tag{7.16}$$

$$q_g = \frac{1}{2} \cdot (u_\beta \cdot i_\alpha - u_\alpha \cdot i_\beta). \tag{7.17}$$



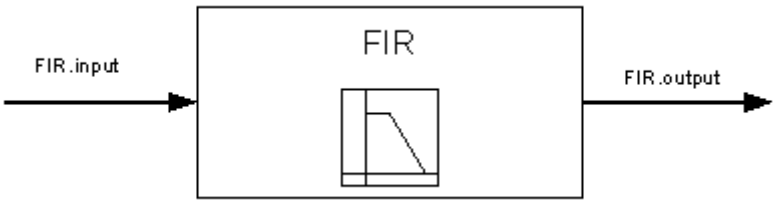
Slika 7.3. Blokovski prikaz CLARKE-ovih transformacija

Napisano u C programskom jeziku:

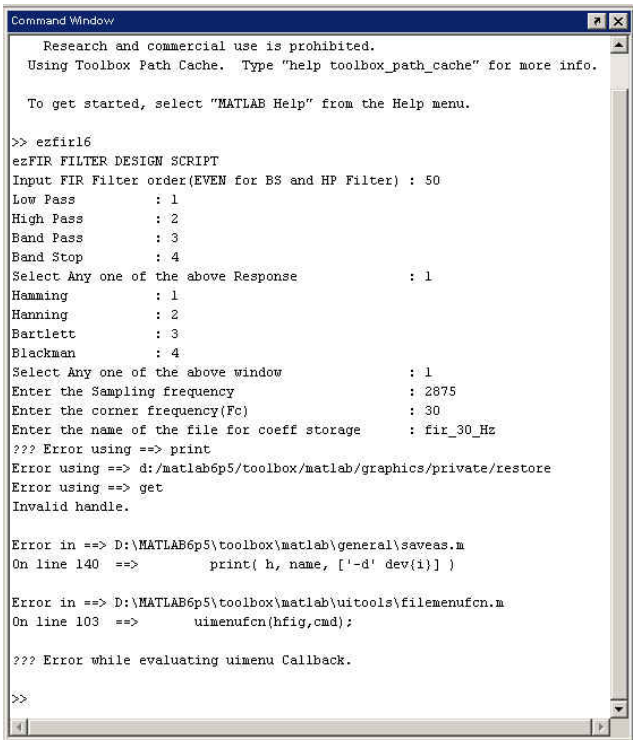
```
UAB.aqf=UAB.af;
UAB.bqf=0.57735*(2*UAB.bf-UAB.af);
UAB.modsqf=UAB.aqf*UAB.aqf+UAB.bqf*UAB.bqf;

IAB.aqf=IAB.af;
IAB.bqf=0.57735*(2*IAB.bf-IAB.af);
IAB.modsqf=(IAB.aqf*IAB.aqf+IAB.bqf*IAB.bqf);

UAB.ulsqrtf=UAB.modsqf;
UAB.ulsqrti=UAB.ulsqrtf*65536;
UAB.izsqrti=qsqrt(UAB.ulsqrti);
UAB.izsqrtf=UAB.izsqrti*0.00390625;
IAB.ulsqrtf=IAB.modsqf;
IAB.ulsqrti=IAB.ulsqrtf*65536;
IAB.izsqrti=qsqrt(IAB.ulsqrti);
IAB.izsqrtf=IAB.izsqrti*0.00390625;
```

7.3 FIR filtar

FIR filtar je preuzet kao gotova funkcijska cjelina iz literature [27]. Više informacija o strukturi koda implementiranog filtra može se naći u pripadajućoj literaturi. Ovdje je potrebno naglasiti da je napravljen makro za MATLAB od strane *Texas Instruments*-a koji proračunava parametre filtra u ovisnosti o tipu filtra, redu filtra, vremenu uzorkovanja (Slika 7.5.). Blokovski filter je prikazan na slici 7.4.



Slika 7.4. Blokovski prikaz FIR filtera



Slika 7.5. Sučelje u MATLAB-u za realizaciju filtra

7.4 Neuronska mreža

Implementirana je neuronska mreža po modelu koji je dan u poglavlju 5. ovog rada. Implementacija mreže napravljena je tako da je prvo alocirano šest struktura ulaznih neurona slijedećeg oblika:

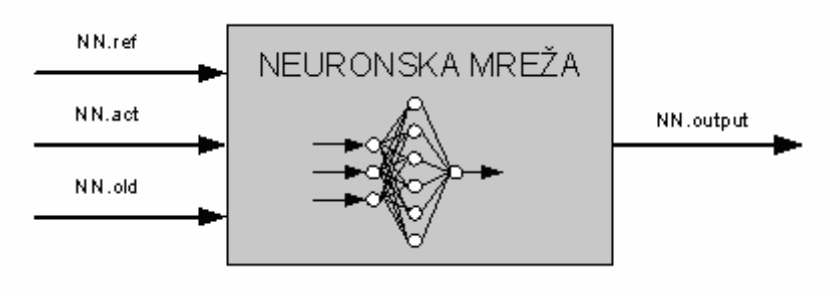
```
typedef struct {
        float   IN1[3];
        float   W1[3];
        float   dW1[3];
        float      OUT_SUMA1;
        TANH    TANHN1;
        float   OUT;
         float   dfi_v_e1;
        } NULAZNI;
```

Zatim je alocirana struktura izlaznog neurona:

```
typedef struct {
```

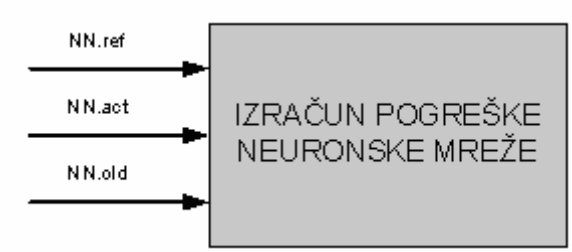
```
        float   IN2[6];
        float   W2[6];
        float   dW2[6];
        float   b2;
        float      OUT_SUMA;
        float   SCAL_SUM;
        TANH    TANHN2;
        float   OUT;
        float   OUT_1;
        float   dfi_v_e;
        float   izlaz;
        float   scalval_izlaza_prema_procesu;

        } NIZLAZNI;
```

Na slici 7.6. blokovski je prikazana implementirana neuronska mreža. U blokovskom prikazu izostavljeni su pojedini parametri da bi se pojednostavio prikaz.



Slika 7.6. Blokovski prikaz neuronske mreže

Te je na kraju alocirana struktura koja računa grešku. Blokovski struktura računanja greške prikazana je na slici 7.7**.** samo sa ulaznim signalima dok parametri nisu prikazani.



Slika 7.7. Blokovski prikaz dijela programa za izračun pogreške neuronske mreže

Alocirana struktura za računanje pogreške je slijedeća:

```
typedef struct {
        float   UREF;
        float      UACT;
        float   UACT_old;
        float   Kv;
        float      RATE;
        float   ERROR;
        TANH    TAN_ERR2;
        TANH    TAN_ERR1[6];
        float   scalval;
        float   KC;
        float   d_v_1_ulaz;
        float   d_v_2_ulaz;
```

```
float  d_v_3_ulaz;
} ERROR;
```

Neuronska mreža implementirana je na način da se prilikom pokretanja sustava parametri mreže postave u neke inicijalne vrijednosti čiji je izbor objašnjen u 2. poglavlju. Zatim se propuste signali referentne vrijednosti napona, stvarne vrijednosti napona generatora i izlaza iz mreže u prošlom trenutku. Nakon što su signali propušteni kroz mrežu računa se greška i lokalni gradijenti za svaki pojedini neuron u neuronskoj mreži. Nakon izračuna dobiju se potrebne promjene težina pojedinih neurona neuronske mreže koje se sekvencijalno dodaju inicijalnim vrijednostima težina. Sljedeći implementirani algoritam to pokazuje:

```
ERROR1.UREF=CP1.Ur;
ERROR1.UACT=CP1.Uact;
ERROR1.ERROR=ERROR1.KC*(ERROR1.UREF-ERROR1.UACT)-
        ERROR1.Kv*(ERROR1.UACT-ERROR1.UACT_old);
        for(i=0;i<6;i++){
                NUL[i].IN1[0]=CP1.Ur;
                NUL[i].IN1[1]=CP1.Uact;
                NUL[i].IN1[2]=NIZ2.OUT;
                                        };
ERROR1.TAN_ERR2.x=NIZ2.OUT_SUMA*NIZ2.SCAL_SUM;
        tansig(&ERROR1.TAN_ERR2);

NIZ2.dfi_v_e=(1-((ERROR1.TAN_ERR2.tanh))*((ERROR1.TAN_ERR2.tanh)));
        for(i=0;i<6;i++){
                NIZ2.dW2[i]=ERROR1.RATE*NIZ2.dfi_v_e*ERROR1.ERROR*NIZ2.IN2[i];
                        };
ERROR1.d_v_1_ulaz=NIZ2.dfi_v_e*CP1.Ur*ERROR1.ERROR*ERROR1.RATE;
ERROR1.d_v_2_ulaz=NIZ2.dfi_v_e*CP1.Uact*ERROR1.ERROR*ERROR1.RATE;
ERROR1.d_v_3_ulaz=NIZ2.dfi_v_e*NIZ2.OUT*ERROR1.ERROR*ERROR1.RATE;

        for(i=0;i<6;i++){
        ERROR1.TAN_ERR1[i].x=NUL[i].OUT_SUMA1;
        tansig(&ERROR1.TAN_ERR1[i]);
        NUL[i].dfi_v_e1=(1-((ERROR1.TAN_ERR1[i].tanh) )*((ERROR1.TAN_ERR1[i].tanh)));

                        };
        for(i=0;i<6;i++){

                NUL[i].dW1[0]=ERROR1.d_v_1_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;
                NUL[i].dW1[1]=ERROR1.d_v_2_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;
                NUL[i].dW1[2]=ERROR1.d_v_3_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;
                        };

        for(i=0;i<6;i++){
                NUL[i].OUT_SUMA1=   NUL[i].IN1[0]*(NUL[i].W1[0]+NUL[i].dW1[0])+
                                    NUL[i].IN1[1]*(NUL[i].W1[1]+NUL[i].dW1[1])+
                                    NUL[i].IN1[2]*(NUL[i].W1[2]+NUL[i].dW1[2]);
                NUL[i].TANHN1.x=NUL[i].OUT_SUMA1;
                tansig(&NUL[i].TANHN1);
                NUL[i].OUT=NUL[i].TANHN1.tanh;
                        };

        for (i=0;i<6;i++){
                NIZ2.IN2[i]= NUL[i].OUT;
                                        };
                NIZ2.OUT_SUMA=NIZ2.IN2[0]*(NIZ2.W2[0]+NIZ2.dW2[0])+
```

```
                                        NIZ2.IN2[1]*(NIZ2.W2[1]+NIZ2.dW2[1])+
                                        NIZ2.IN2[2]*(NIZ2.W2[2]+NIZ2.dW2[2])+
                                        NIZ2.IN2[3]*(NIZ2.W2[3]+NIZ2.dW2[3])+
                                        NIZ2.IN2[4]*(NIZ2.W2[4]+NIZ2.dW2[4])+
                                        NIZ2.IN2[5]*(NIZ2.W2[5]+NIZ2.dW2[5])+
                                        NIZ2.b2;
                NIZ2.TANHN2.x=NIZ2.OUT_SUMA*NIZ2.SCAL_SUM;
                tansig(&NIZ2.TANHN2);
                NIZ2.OUT=ERROR1.scalval*NIZ2.TANHN2.tanh;
                NIZ2.izlaz=(NIZ2.OUT-duty)*NIZ2.scalval_izlaza_prema_procesu;
                ERROR1.UACT_old=ERROR1.UACT;
```

7.5 Nelinearna aktivacijska funkcija

Neuronska mreža implementirana u ovom sustavu koristi *tansig* aktivacijsku funkciju. Ta je funkcija detaljno opisana u 5 poglavlju. Budući da ona nije implementirana u DSP od strane proizvođača potrebno ju je implementirati. Matematički model implementirane funkcije glasi:

$$\varphi_j(n) = \frac{1}{1+\exp(-v_j(n))} - 1 \quad . \tag{7.18}$$

Aktivacijska funkcija u sebi sadrži eksponencijalnu (*exp()*) funkciju koju je potrebno zasebno implementirati u sustav. To je učinjeno tako da se funkcija *exp()* razvije u Taylor-ov red te je kao red implementirana u sustav [28].

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ... \tag{7.19}$$

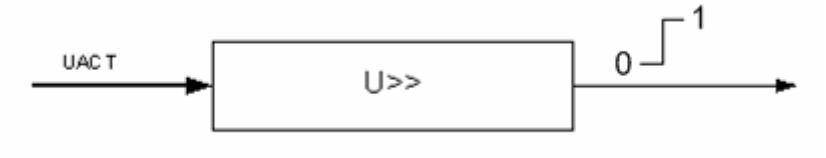
Iz praktičnih razloga u sustav je implementirano prvih sedam članova gore navedenog reda što se kroz eksperiment pokazalo sasvim zadovoljavajućim. Izgled *exp()* funkcije razvijene u red i napisane u C kodu :

```
void tansig (TANH *TANH1){

 (TANH1->in)=(TANH1->x)*(-2);
(TANH1->ex)=1+(TANH1->in)+(0.5*(TANH1->in)*(TANH1->in))+
                        (0.16666667*(TANH1->in)*(TANH1->in)*(TANH1->in))+
                        (0.04166667*(TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1->in))+
                         (0.00833333*(TANH1->in)*(TANH1->in)*(TANH1->in)*
                                        (TANH1->in)*(TANH1->in))+
                         (0.00138889*(TANH1->in)*(TANH1->in)*(TANH1->in)*
                                        (TANH1->in)*(TANH1->in)*(TANH1->in))+
                         (0.00019841*(TANH1->in)*(TANH1->in)*(TANH1->in)*
                                (TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1->in));

TANH1->tanh=(2/(1+(TANH1->ex)))-1;
                        };
```

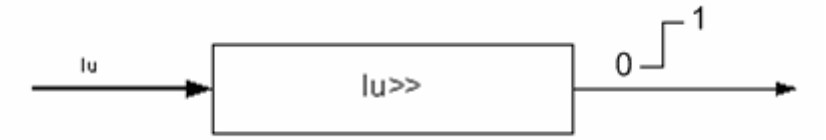
7.6 Zaštitne funkcije

U sustavu su implementirane sljedeće zaštitne funkcije :

- zaštita od prenapona (Slika 7.8.),
- zaštita od prevelike struje uzbude (Slika 7.9.).



Slika 7.8. Blokovski prikaz zaštite od previsokog napona



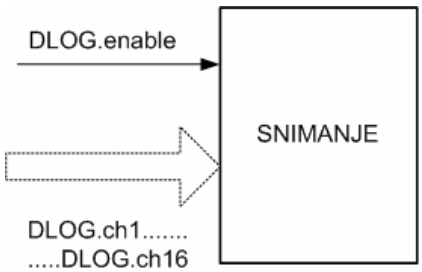
Slika 7.9. Blokovski prikaz zaštite od previsoke uzbudne struje

Zaštita od prenapona inicijalno je postavljena na vrijednost 1.2*Un tako da ukoliko se postigne vrijednost napona veća ili jednaka od 1.2*Un dolazi do gašenja čopera. Isto vrijedi i za preveliku struju uzbude te je ovdje vrijednost za proradu zaštite postavljena na 1.1*Iun.

Izlazi iz gore navedena dva zaštitna bloka su binarnog oblika. U slučaju da se prekorače parametarski zadane vrijednosti izlaz iz zaštitnih funkcija prelazi iz logičke nule u logičku jedinicu. Ukoliko su promatrane vrijednosti ispod postavljenih gornjih vrijednosti izlaz iz zaštitnog bloka je logička nula.

7.7 Funkcije snimanja podataka u stvarnom vremenu

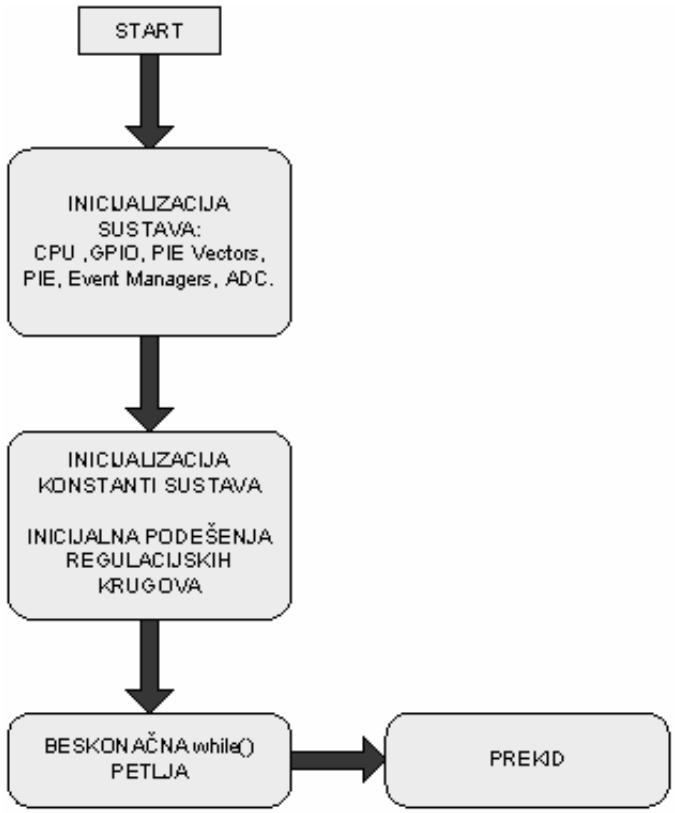
Za potrebe laboratorijskih istraživanja i praktičnih potreba razvijena je funkcija koja omogućava snimanje 16 različitih varijabli u rezoluciji od 1024 točke 16 bitne preciznosti. Ova funkcija snimljene podatke pohranjuje u vanjski SARAM koji se nalazi na eZdsp pločici i veličine je 64 k. Ovim snimljenim vrijednostima pristupa se preko JTAG emulatora i pohranjuje ih se na osobno računalo u obliku slijedne tekstualne datoteke koja se dalje može obrađivati različitim programskim paketima (primjerice EXCEL).
Blokovski prikazanu funkciju snimanja u stvarnom vremenu vidimo na slici 7.10.



Slika 7.10.. Blokovski prikaz snimanja podataka u SARAM

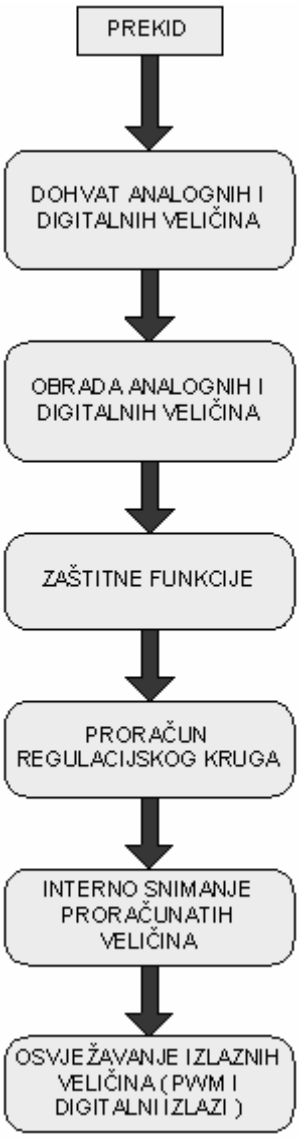
7.8 Dijagram toka izvođenja programa u procesoru

Implementirani program u procesoru radi na slijedeći način. Prilikom pokretanja procesora prvo se inicijaliziraju osnovni sklopovi procesora, CPU, GPIO, PIE *Vectors*, PIE, *Event Managers*, ADC. Nakon inicijalizacije sustava postavljaju se inicijalna vrijednosti konstanti u sustavu i početna podešenja regulacijskih krugova. Program zatim ulazi u beskonačnu *while()* petlju čekajući neki od hardverskih prekida (Slika 7.11.). Cjelokupna programska rutina sustava regulacije dana je u dodatku E, dok je glavna *main()* rutina i definicija funkcija dana u dodatku F.



Slika 7.11. Dijagram toka izvođenja programa

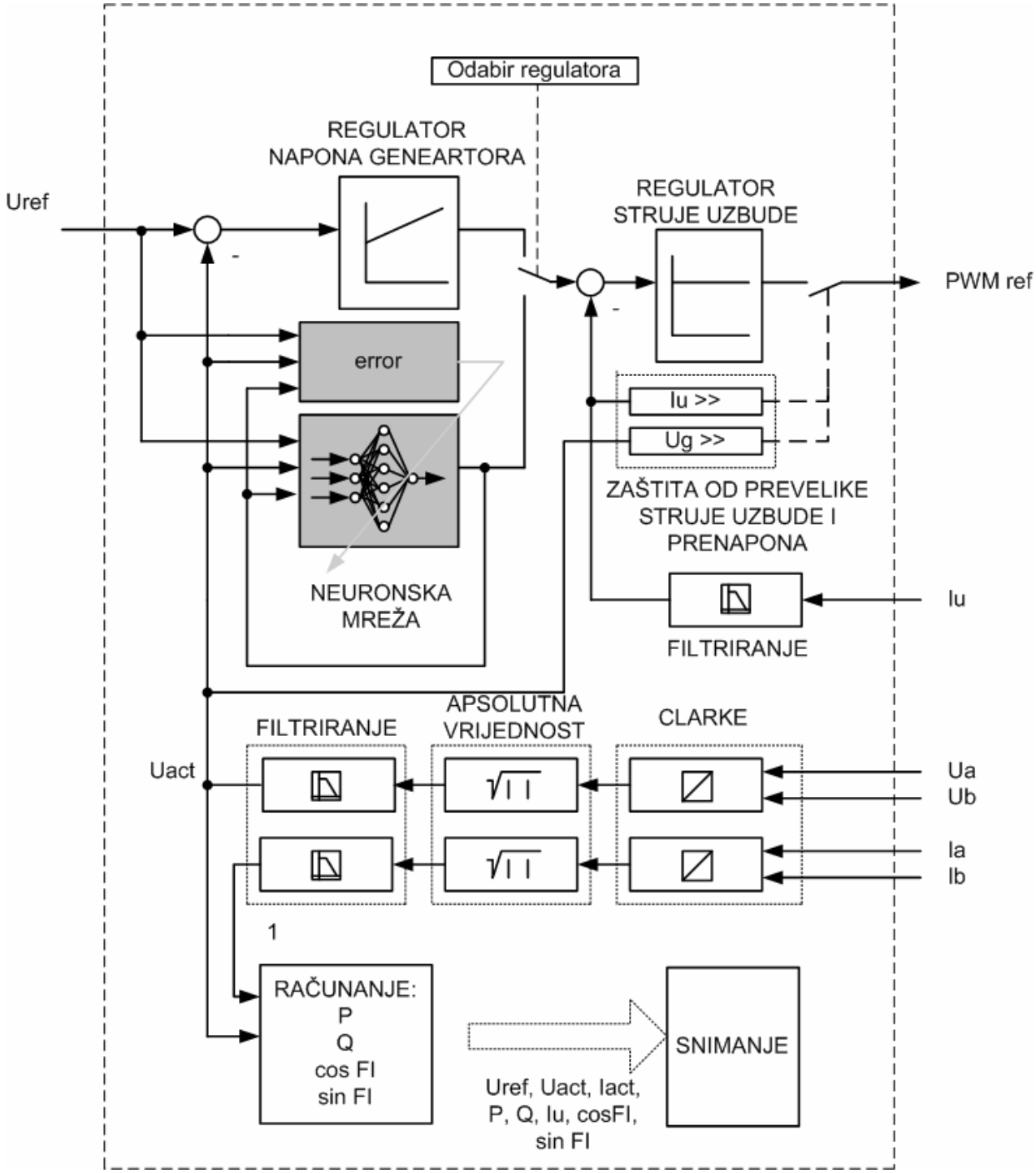
U implementiranom programu cijeli regulacijski algoritam postavljen je da se izvrši u prekidu koji generira A/D konvertor. Prekid počinje svaki put kad je završena jedna konverzija A/D pretvarača. Nakon što se izvrše sve rutine koje su zadane u prekidu, nastavlja se izvršavati glavna beskonačna *while()* petlja do ponovnog prekida. Prilikom izvođenja prekida prvo se uzimaju sve analogne i digitalne veličine koje ulaze u sustav. Nakon što su analogne i digitalne veličine uzete pristupa se njihovoj obradi i filtriranju. Zatim se izvršavaju zaštitne funkcije te nakon njih izvršavaju se regulacijske funkcije. Pošto se završi s računanjem regulacijskih veličina pristupa se izvršavanju bloka za snimanje podataka u internu memoriju. Na kraju prekida osvježavaju se izlazne veličine to jest referenca za PWM i digitalni izlazi (Slika 7.12.)



Slika 7.12. Dijagram toka izvođenja prekida

7.9 IMPLEMENTIRANA REGULACIJSKA STRUKTURA

Funkcije realizirane u C programskom kodu implementirane su u regulacijsku strukturu koja je prikazana na slici 7.13.



Slika 7.13. Implementirana regulacijska struktura

Naponi U_a i U_b su linijski naponi generatora, a struje I_a i I_b su fazne struje generatora. Pomoću funkcija Clarkeovih transformacija te se veličine pretvaraju iz troosnog sustava u dvoosni te se izračunavaju njihove apsolutne vrijednosti. Te se veličine zatim filtriraju. Struja uzbude I_u generatora također se uvodi u sustav. Ulazi u regulator napona su napon generatora i zadana referenca. Isto tako parametarski se odabire regulator napona PI tipa ili neuronska mreža.

Sinkroni generator štiti se od prevelike vrijednosti induciranog napona i od pojave prevelike struje uzbude.

U sustav je implementirano i računanje radne i jalove snage generatora na osnovi mjerenja struja i napona generatora.

Sve karakteristične veličine koje se izračunavaju u sustavu snimaju se u stvarnom vremenu te se pohranjuju u memoriju koja se nalazi na eZdsp pločici. Nadalje te se veličine preko *Code Composer Studio* programskog paketa mogu prebaciti na osobno računalo te se tamo dodatno obraditi u nekom od programa (EXCEL...). Slike realiziranog sustava uzbude i sinkronog generatora s parametrima nalaze se u dodatku G.

8 PODEŠAVANJE PARAMETARA SUSTAVA UZBUDE GENERATORA

Parametre regulatora napona zasnovanog na neuronskoj mreži potrebno je podesiti. Parametri regulatora napona zasnovanog na neuronskoj mreži podešeni su pri radu generatora u praznom hodu. Da bi se ostvarilo stabilizirajuće djelovanje neuronske mreže na njihanje generatora, potrebno je dodatno podesiti parametre neuronske mreže prilikom rada generatora na elektroenergetskoj mreži.

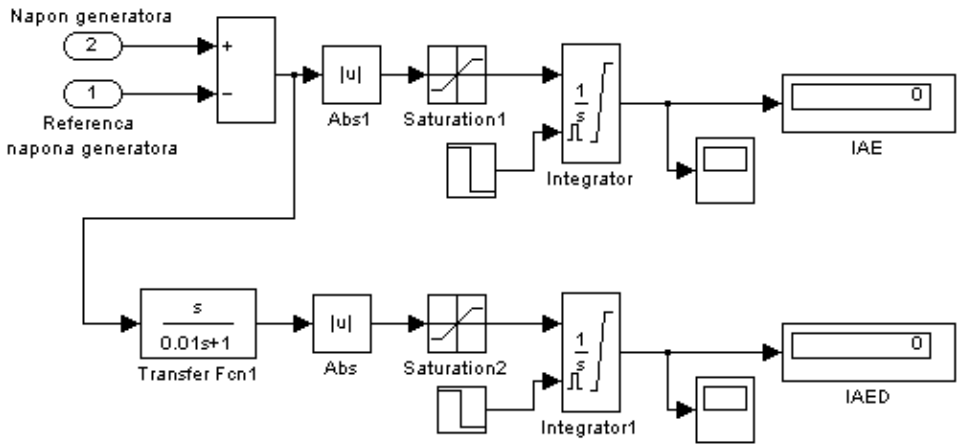
Za analizu kvalitete odziva regulirane veličine (napona ili radne snage sinkronog generatora) korištena su dva kriterija kvalitete :

- Integral apsolutnog odstupanja (eng. *integral absolute error*, akronim *IAE*) – integral apsolutne razlike između napona sinkronog generatora i reference napona sinkronog generatora
- Integral apsolutnog deriviranog odstupanja *(IAED)* – integral derivirane razlike između napona i reference napona. Iznos ovog kriterija raste u slučaju oscilatornog odziva.

Za ove kriterije vrijedi da je odziv to bolji što je iznos kriterija manji.

8.1 Podešavanje parametara regulatora napona zasnovanog na neuronskoj mreži

Kriteriji kvalitete odziva napona generatora računaju se simulacijom na računalu (Slika 8.1.).



Slika 8.1. Simulacijski algoritmi određivanja kriterija kvalitete *IAE* i *IAED* (utjecaj veličina k1 i K)

Funkcija za određivanje kriterija IAE I IAED realizirana u programskom jeziku C izgleda:

```
void iae(IAE *IAE){
        float32 der,sum_1,int1,int2,sum_2;
                sum_1=IAE->in_1-IAE->in_2;
                        if(sum_1<0){sum_1=-sum_1;}

                int1=IAE->IAE1_1_old+IAE->K1_sum_1*(sum_1+IAE->sum_1_old);
                IAE->sum_1_old=sum_1;
                IAE->IAE1_1_old=int1;
                IAE->IAE=int1;
                sum_2=IAE->in_1-IAE->in_2;

                der=IAE->Kd_sum_2*(sum_2-IAE->sum_2_old);
                IAE->sum_2_old=sum_2;

                        if(der<0){der=-der;}

                int2=IAE->IAE2_2_old +IAE->K2_sum_2*(der+IAE->sum_deriv_2);
                IAE->sum_deriv_2=der;
                IAE->IAE2_2_old=int2;

                IAE->IAED=int2;
                }
```

Za rad regulatora napona zasnovanog na neuronskoj mreži mjerodavna je modificira funkcija greške (5.11) i jednadžba (5.4) za izračun težina pojedinih neurona metodom povratnog prostiranja kao i ukupno pojačanje neuronske mreže.

U modificiranoj funkciji greške osim signala koji se računaju postoje i faktori skaliranja :

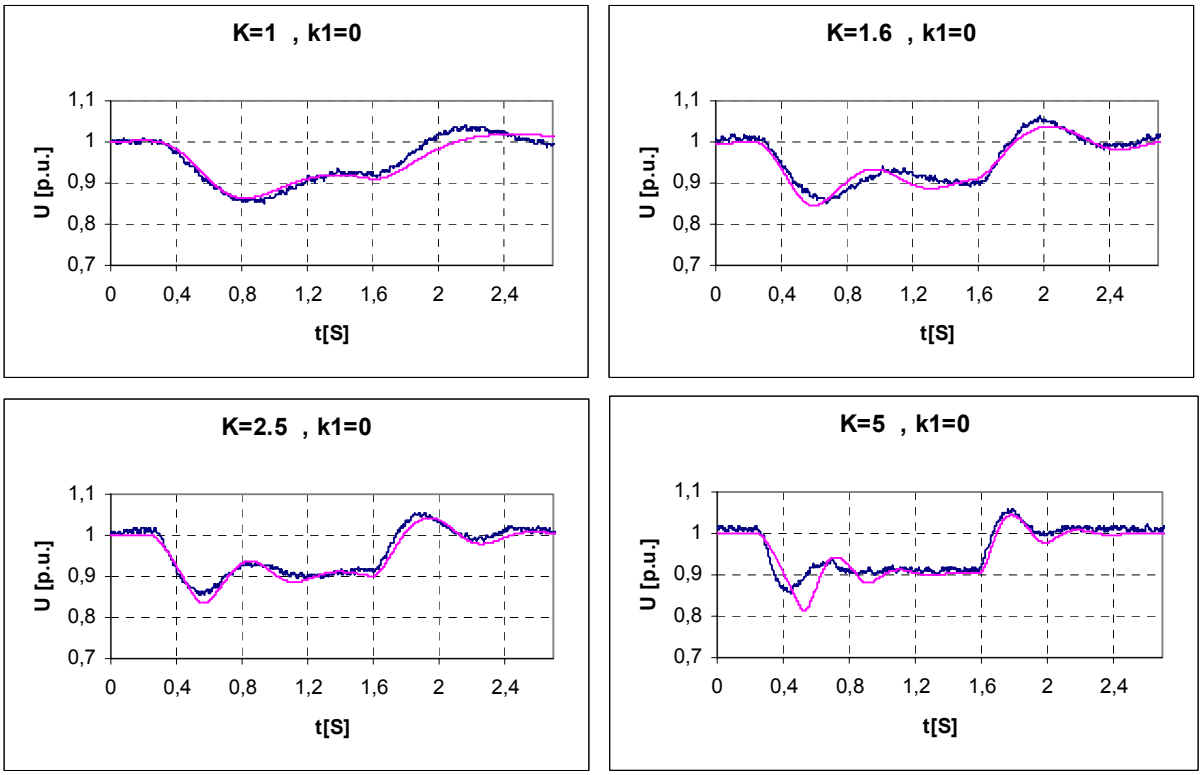
- *K* – skaliranje razlike referentne i stvarne vrijednosti napona
- *k1* – skaliranje derivacije napona generatora
- *k2* – skaliranje devijacije radne snage.
- *k3*- skaliranje derivacije radne snage

Faktori skaliranja *K* i *k1* su mjerodavni za ponašanje regulatora kod regulacije napona dok su veličine *k2* i *k3* mjerodavne za stabilizaciju njihanja radne snage. Početno se podešavaju parametri K i k1 u praznom hodu na skokovitu promjenu postavne veličine napona generatora te se promatra ponašanje regulatora. Veličine *k2* i *k3* se podešavaju kad je generator spojen na mrežu preko reaktancije opterećen s približno 50% radne snage te mu se skokovito promjeni postavna veličina reference napona u iznosu 10% od nazivnog napona [7].

IAE i *IAED* kriteriji kvalitete prigušenja oscilacija radne snage identični su kriterijima za određivanje kvalitete odziva napona, osim što za usporedbu koriste stacionarnu i trenutnu vrijednost radne snage.

Ponašanje regulatora napona zasnovanog na neuronskoj mreži u prvom trenutku prijelazne pojave određuje faktor skaliranja razlike (k) stvarne i referentne vrijednosti napona i skaliranje derivacija (k1) napona generatora u modificiranoj kriterijskoj funkciji (5.11).

Nadalje brzinu postizanja stacionarnog stanja zadanog postavom veličinom napona određuje brzina učenja mreže (η) . Vrijednost skaliranja brzine učenja ne može se postaviti u vrijednost nula da bi se samo promatrao utjecaj ukupnog pojačanja regulatora (tu vrijednost možemo smatrati da je ekvivalent integralnoj konstanti kod klasičnog PI regulatora). U tom se slučaju ne bi mogao mijenjati iznos težina pojedinih neurona u neuronskoj mreži u procesu učenja koji se odvija cijelo vrijeme rada regulatora. Početna vrijednost faktora brzine učenja η preuzeta je iz literature [8] te iznosi 0.04. Faktor skaliranja K je za početak izabran da bude 1, te je povećavan do iznosa 5 što je prikazano na Slici 8.2.



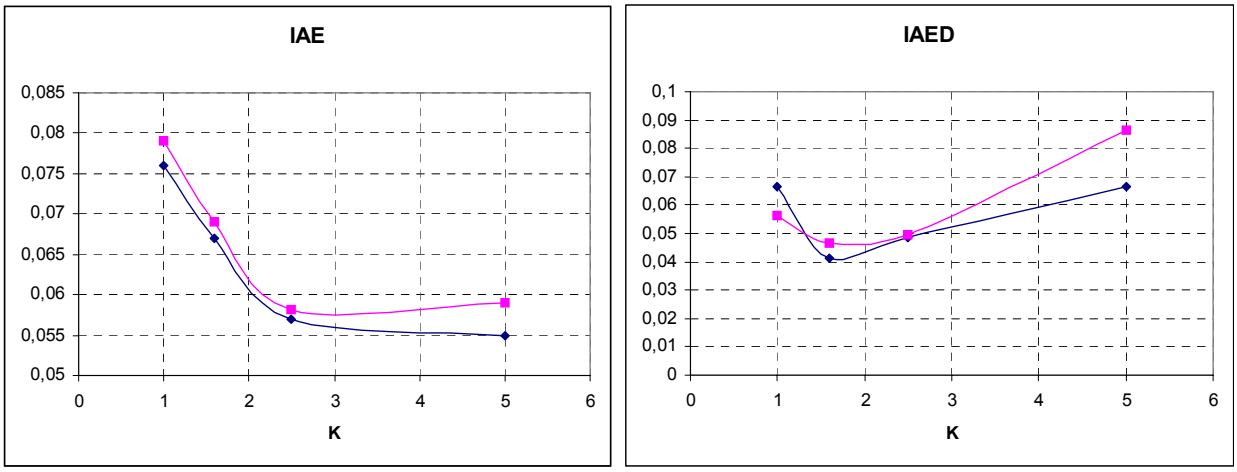
Slika 8.2. Eksperimentalni (plavo) i simulacijski (rozo) odzivi napona generatora za različite iznose parametra K pri skokovitoj promjeni postavne veličine napona

Iz prikazanih odziva računaju se kriteriji kvalitete odziva te su prikazani u tablici 8.1.

K	IAE (SIMULACIJA)	IAED (SIMULACIJA)	IAE (EKSPERIMENT)	IAED (EKSPERIMENT)
1	0,0791	0,0565	0,076	0,0665
1,6	0,0691	0,0465	0,067	0,0415
2,5	0,0581	0,04958	0,057	0,0485
5	0,059	0,08658	0,055	0,06658

Tablica 8.1. Ovisnost kriterija kvalitete odziva napona generatora o ukupnom pojačanju regulatora K

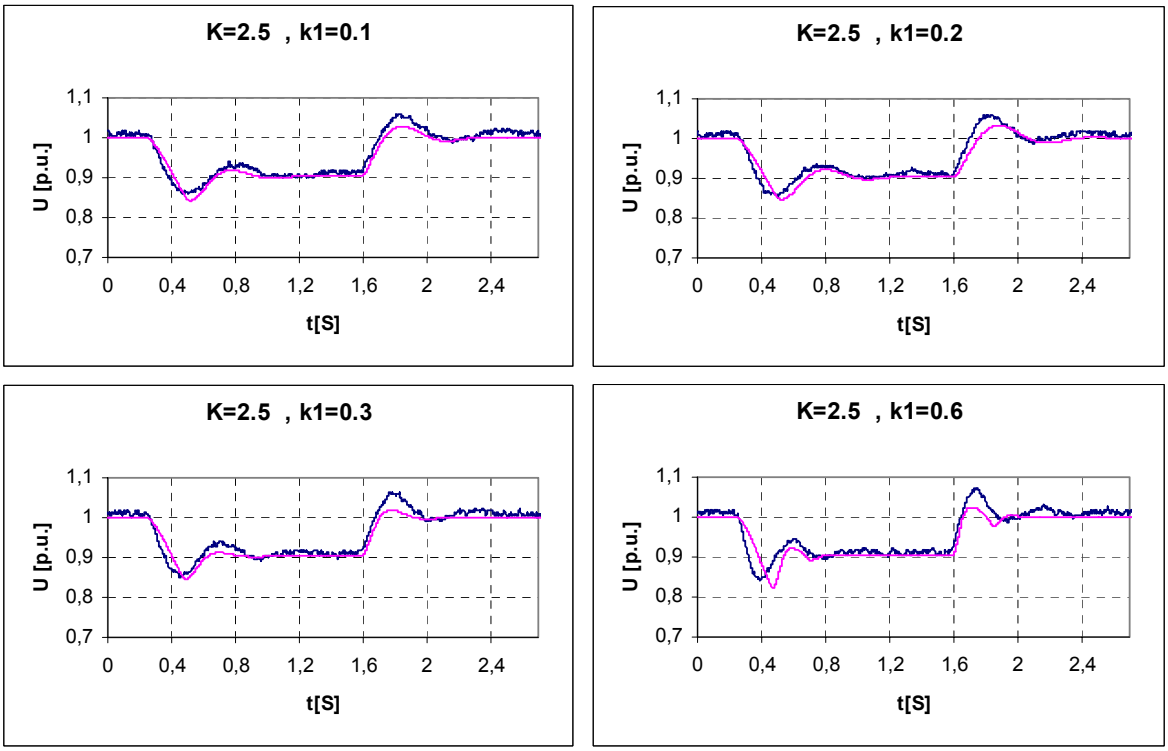
Ovisnost kriterija kvalitete odziva o parametru K regulatora zasnovanog na neuronskoj mreži grafički je prikazana na slici 8.3.



Slika 8.3 Ovisnost eksperimentalno (plavo) i simulacijski (rozo) dobivenih kriterija kvalitete o iznosu parametra *K* regulatora napona generatora zasnovanog na neuronskoj mreži

S gornjih slika vidljivo je da povećanjem parametra K regulator postiže brže stacionarnu vrijednost međutim pojačava se oscilatornost odziva tako da je izabrana vrijednost *K*=2.5.

Osim ukupnog pojačanja regulatora potrebno je podesiti i faktor skaliranja derivacije napona generatora k1 u kriterijskoj funkciji. Za početnu vrijednost izabrana je vrijednost *k1*=0.1 (Slika 8.4.)



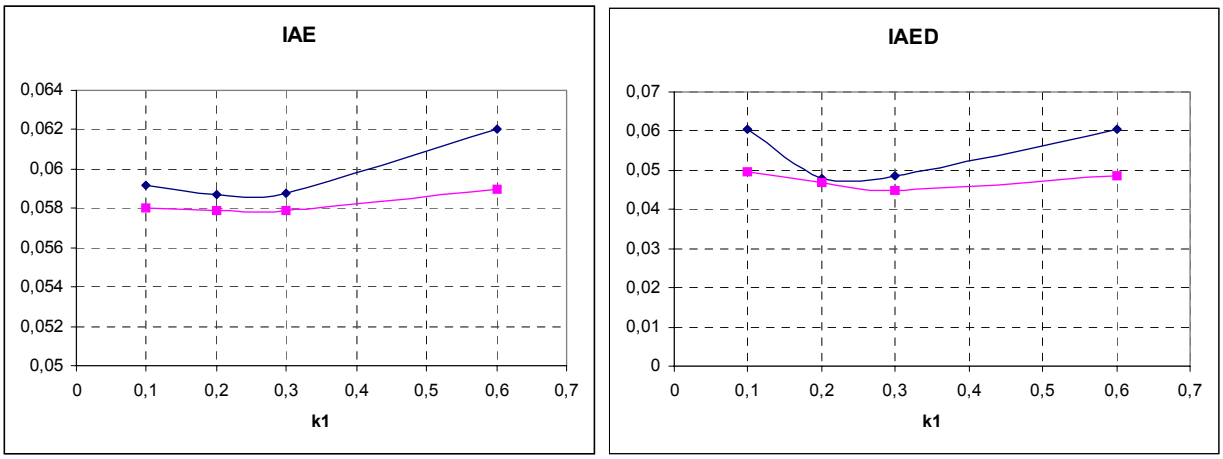
Slika 8.4. Eksperimentalni (plavo) i simulacijski (rozo) odzivi napona generatora za različite iznose parametra *k1* pri skokovitoj promjeni postavne veličine napona konstantnoj vrijednosti parametra K=2.5

Iz prikazanih odziva računaju se kriteriji kvalitete odziva te su prikazani u tablici 8.2**.**

k1	IAE (SIMULACIJA)	IAED (SIMULACIJA)	IAE (EKSPERIMENT)	IAED (EKSPERIMENT)
0,1	0,058	0,0495	0,0592	0,06065
0,2	0,0579	0,047	0,0587	0,0479
0,3	0,0579	0,0448	0,0588	0,0485
0,6	0,059	0,04858	0,062	0,06058

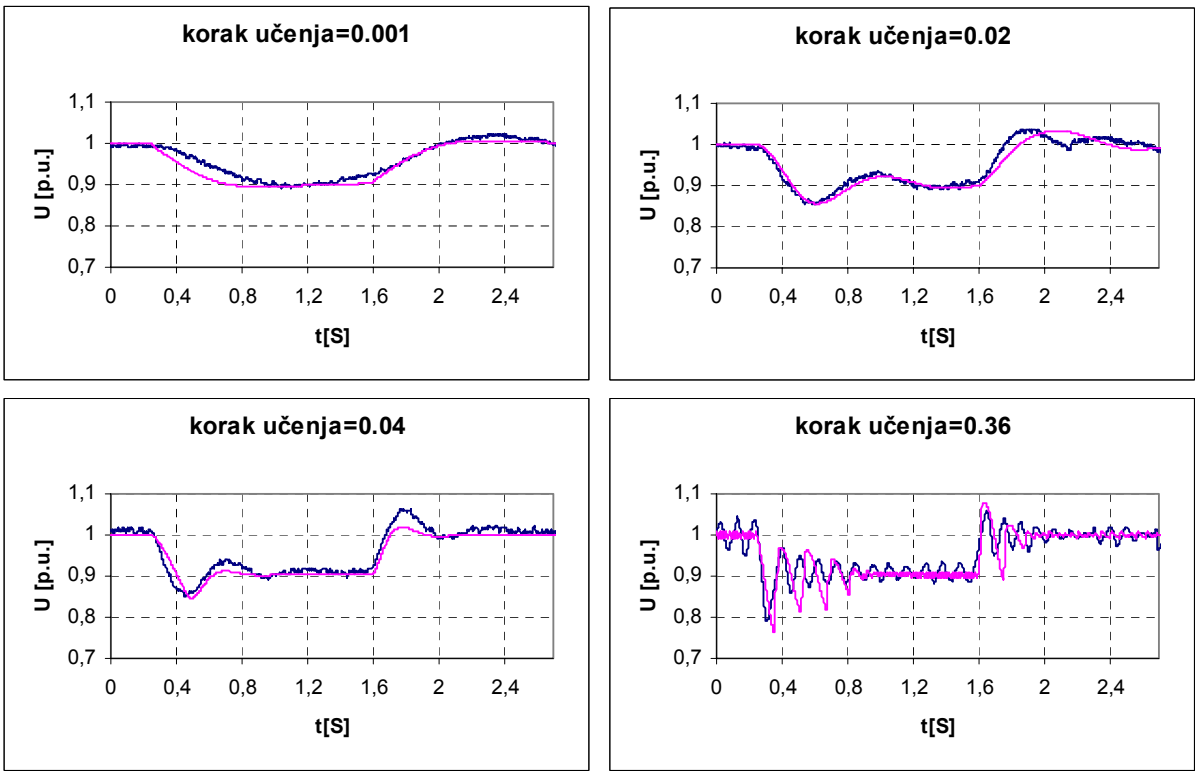
Tablica 8.2. Ovisnost kriterija kvalitete odziva napona generatora o parametru k1 uz konstantnu vrijednost parametra K=2.5

Ovisnost kriterija kvalitete odziva o parametru K regulatora zasnovanog na neuronskoj mreži grafički je prikazana na slici 8.5.



Slika 8.5. Ovisnost eksperimentalno (plavo) i simulacijski (rozo) dobivenih kriterija kvalitete o iznosu parametra *k1* regulatora napona generatora zasnovanog na neuronskoj mreži uz konstantnu vrijednost parametra K=2.5

Korak učenja η neuronske mreže preuzet je iz literature [8] te mu je početna vrijednost iznosila 0.04. Za konstantne parametre K=2.5 i k1=0.3 modificirane kriterijske funkcije mijenjani su iznosi koraka učenja. Početna vrijednost je postavljena na 0.001 (Slika 8.6).



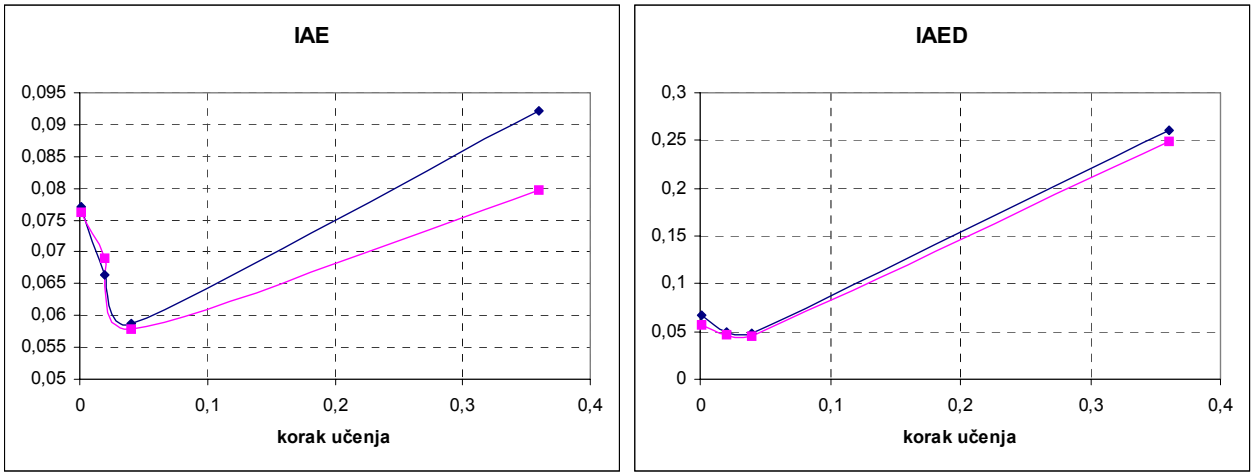
Slika 8.6. Eksperimentalni (plavo) i simulacijski (rozo) odzivi napona generatora za različite iznose koraka učenja pri skokovitoj promjeni postavne veličine napona i konstantnoj vrijednosti parametra K=2.5 i k1=0.3

Iz prikazanih odziva računaju se kriteriji kvalitete odziva te su prikazani u tablici **8.3.**

Korak učenja	IAE (SIMULACIJA)	IAED (SIMULACIJA)	IAE (EKSPERIMENT)	IAED (EKSPERIMENT)
0,001	0,0762	0,0565	0,0771	0,0665
0,02	0,0691	0,0465	0,0664	0,0497
0,04	0,0579	0,0448	0,0588	0,0485
0,36	0,0798	0,24858	0,0922	0,26058

Tablica 8.3. Ovisnost kriterija kvalitete odziva napona generatora o koraku učenja uz konstantnu vrijednost parametara K=2.5 i k1=0.3

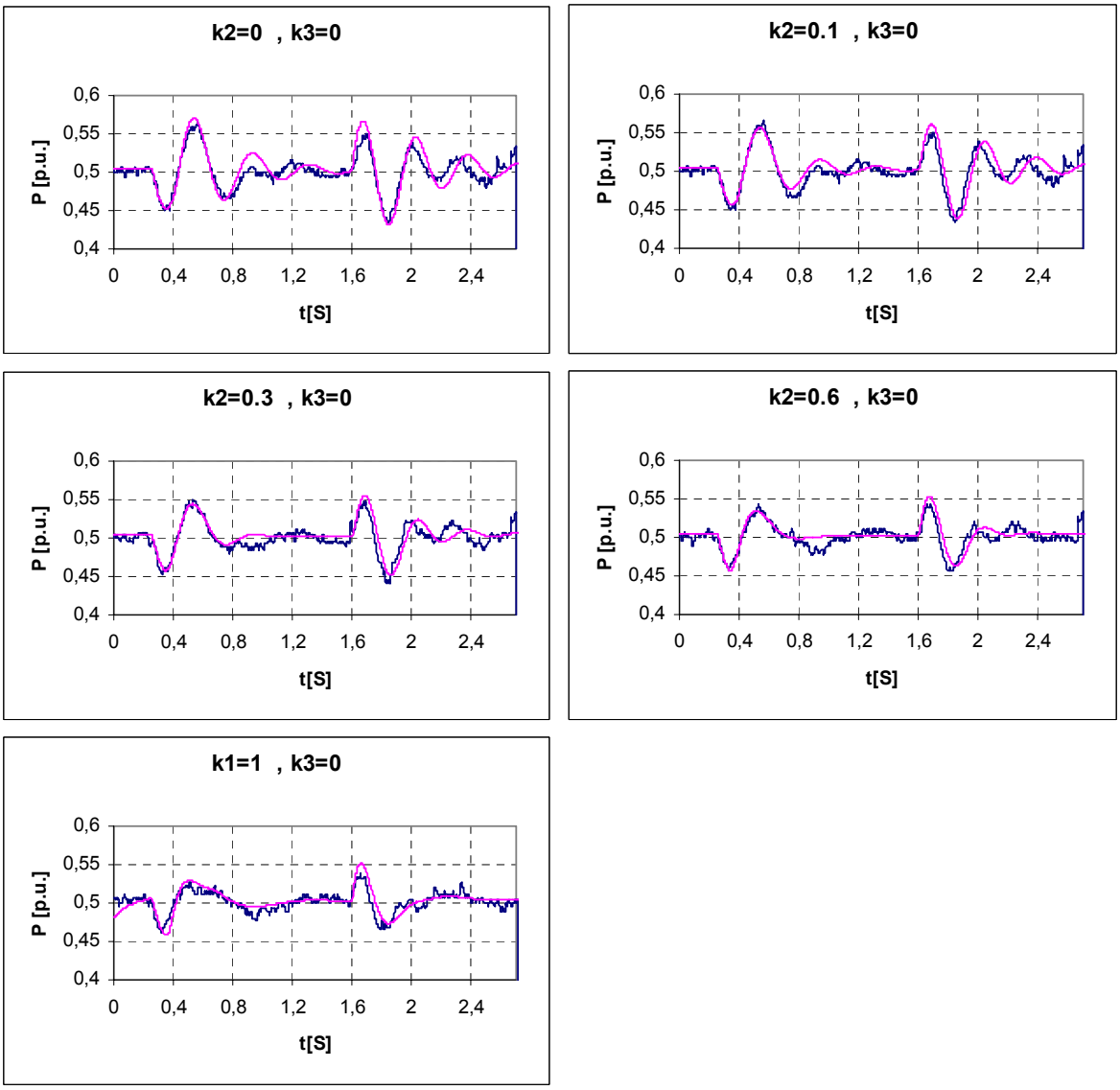
Ovisnost kriterija kvalitete odziva o koraku učenja neuronske mreže grafički je prikazana na slici 8.7.



Slika 8.7. Ovisnost eksperimentalno (plavo) i simulacijski (rozo) dobivenih kriterija kvalitete o koraku učenja neuronske mreže uz konstantnu vrijednost parametra K=2.5 i k1=0.3

8.2 Podešavanje parametara kriterijske funkcije za ostvarivanje stabilizirajućeg efekta

Osim podešenja parametara skaliranja *K* i k1 potrebno je podesiti i faktore skaliranja k2 i k3 kriterijske funkcije koji se odnose na skaliranje devijacije radne snage i derivacije radne snage koji se uvode kao signali u kriterijsku funkciju neuronske mreže. Algoritam za određivanje kvalitete odziva identičan je kao onaj prikazan na slici 8.1. uz razliku da se signali koji se uvode u algoritam su radna snaga i stacionarna vrijednost radne snage. Za početak su parametri k2 i k3 podešeni na vrijednost 0, te se je prvo povećavao faktor skaliranja devijacije radne snage a zatim faktor skaliranja derivacije radne snage.(Slika 8.8)



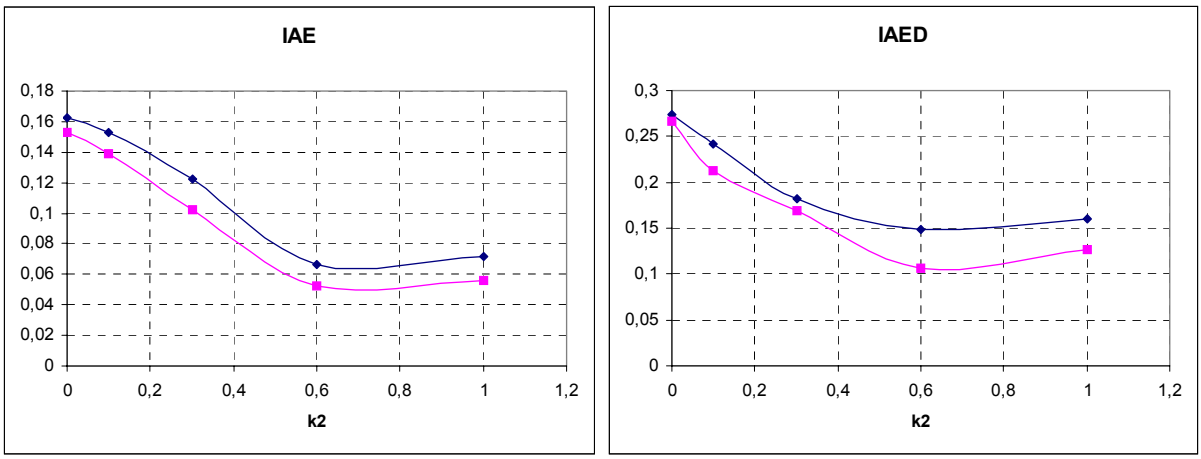
Slika 8.8. Eksperimentalni (plavo) i simulacijski (rozo) odzivi napona generatora za različite iznose parametra *k2* pri skokovitoj promjeni postane veličine napona konstantnoj vrijednosti parametra K=2.5, k1=0.3

Iz prikazanih odziva računaju se kriteriji kvalitete odziva te su prikazani u tablici **8.4.**

k2	IAE (SIMULACIJA)	IAED (SIMULACIJA)	IAE (EKSPERIMENT)	IAED (EKSPERIMENT)
0	0,1532	0,26625	0,1627	0,2742
0,1	0,1385	0,21256	0,1532	0,2421
0,3	0,1022	0,16875	0,1225	0,1824
0,6	0,0527	0,106	0,066	0,1481
1	0,05581	0,1265	0,0713	0,16058

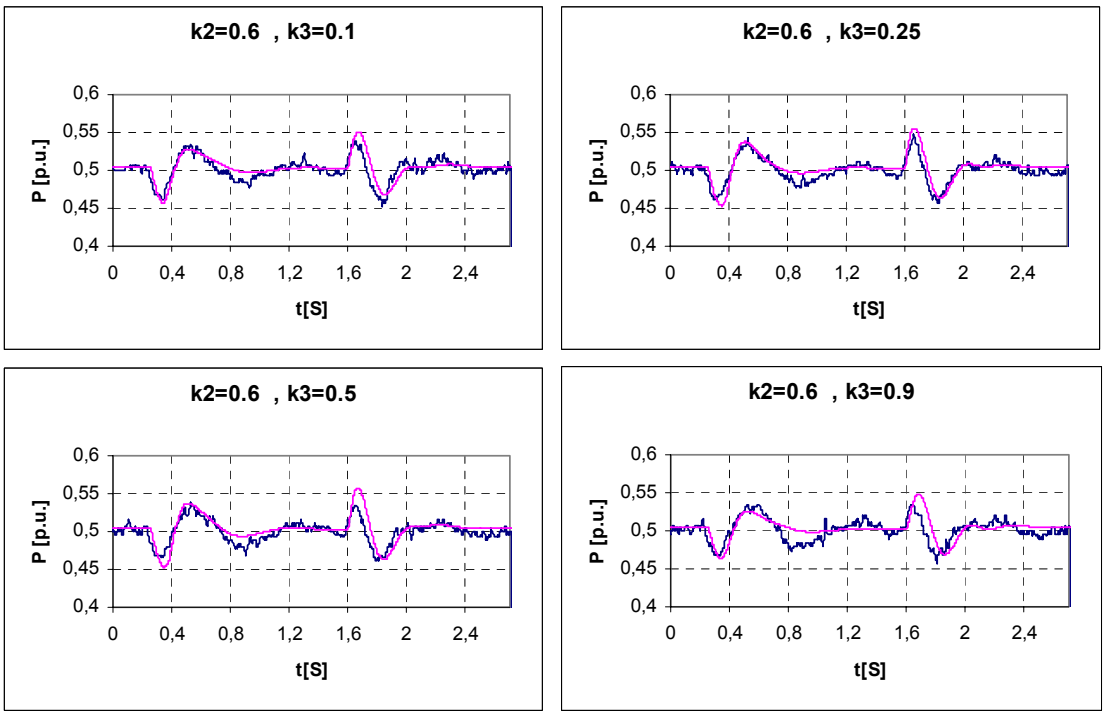
Tablica 8.4. Ovisnost kriterija kvalitete odziva napona generatora izračunati o parametru k2 uz konstantnu vrijednost parametra K=2.5 i k1=0.3

Ovisnost kriterija kvalitete odziva o parametru k2 regulatora zasnovanog na neuronskoj mreži grafički je prikazana na slici 8.9.



Slika 8.9. Ovisnost eksperimentalno (plavo) i simulacijski (rozo) dobivenih kriterija kvalitete o iznosu parametra *k2* regulatora napona generatora zasnovanog na neuronskoj mreži uz konstantnu vrijednost parametra K=2.5, k1=0.3

Slijedeći korak u podešavanju kriterijske funkcije je podešavanje parametra k3 što je prikazano na slici 8.10.



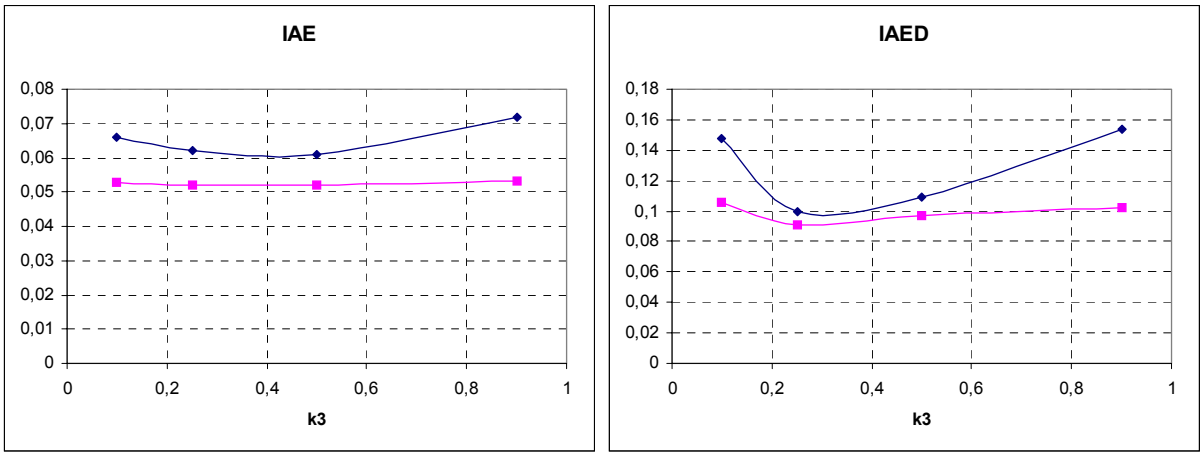
Slika 8.10. Eksperimentalni (plavo) i simulacijski (rozo) odzivi napona generatora za različite iznose parametra *k3* pri skokovitoj promjeni postane veličine napona konstantnoj vrijednosti parametra K=2.5, k1=0.3, k2=0.6

Iz prikazanih odziva računaju se kriteriji kvalitete odziva te su prikazani u tablici 8.5.

K3	IAE (SIMULACIJA)	IAED (SIMULACIJA)	IAE (EKSPERIMENT)	IAED (EKSPERIMENT)
0,1	0,0527	0,106	0,066	0,1481
0,25	0,052	0,091	0,062	0,0993
0,5	0,052	0,0972	0,061	0,1096
0,9	0,0533	0,1022	0,072	0,1534

Tablica 8.5. Ovisnost kriterija kvalitete odziva napona generatora o parametru k3 uz konstantnu vrijednost parametra K=2.5 i k1=0.3, k2=0.6

Ovisnost kriterija kvalitete odziva o parametru k3 regulatora zasnovanog na neuronskoj mreži grafički je prikazana na slici 8.11.



Slika 8.11. Ovisnost eksperimentalno (plavo) i simulacijski (rozo) dobivenih kriterija kvalitete o iznosu parametra *k3* regulatora napona generatora zasnovanog na neuronskoj mreži uz konstantnu vrijednost parametra K=2.5, k1=0.3, k2=0.6

Izabrana podešenja kriterijske funkcije na osnovu razmatranja u ovom poglavlju dana su u tablici 8.6.

FAKTOR SKALIRANJA	IZABRANA VRIJEDNOST
K	2.5
k1	0.3
KORAK UČENJA (η)	0.04
k2	0.6
k3	0.25

Tablica 8.6. Izabrana podešenja kriterijske funkcije

9 SIMULACIJSKI I EKSPERIMENTALNI REZULTATI DJELOVANJA SUSTAVA REGULACIJE UZBUDE SINKRONOG GENERATORA

Sustav regulacije uzbude ispitan je eksperimentalno te je načinjen simulacijski model u uvjetima skokovite promjena postavne veličine napona generatora (podaci generatora u dodatku B) .

Ispitano je djelovanje sustava regulacije uzbude za sljedeće regulatore napona:

- proporcionalno integralni regulator napona
- neuronsku mrežu bez stabilizirajućeg efekta u kriterijskoj funkciji
- neuronsku mrežu sa stabilizirajućim efektom u kriterijskoj funkciji

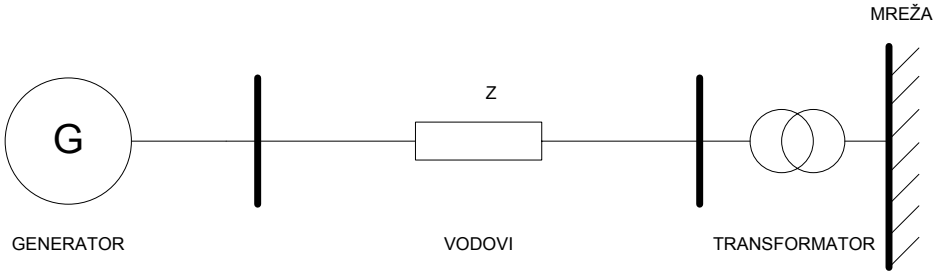
U ispitivanju djelovanja sustava uzbude obuhvaćene su različite radne točke sinkronog generatora s obzirom na radnu i jalovu snagu. U svakom eksperimentu uspoređeni su simulacijski i eksperimentalni odzivi sustava regulacije uzbude za slučaj djelovanja PI regulatora napona, neuronske mreže bez stabilizirajućeg efekta u kriterijskoj funkciji i s neuronskom mrežom sa stabilizirajućim efektom u kriterijskoj funkciji.

Nakon toga u istom dijagramu uspoređen je eksperimentalni odziv radne snage generatora za sva tri slučaja. Učinak regulatora na njihanje radne snage je bolji ako su oscilacije radne snage manje po amplitudom iznosu.

Analiza djelovanja sustava regulacije uzbude generatora provedena je pomoću izravnih pokazatelja kvalitete odziva radne snage za eksperimentalne odzive. To su minimalna vrijednost (*MIN*), maksimalna vrijednost (*MAX*), integral apsolutnog odstupanja (*IAE*) i integral apsolutnog deriviranog odstupanja (*IAED*). Za kvalitetniji odziv radne snage iznos minimalne vrijednosti treba biti što veći jer to znači da je djelovanje stabilizatora smanjilo iznos amplituda oscilacije snage. Za maksimalnu vrijednost, integral apsolutnog odstupanja i integral apsolutnog deriviranog odstupanja poželjno je da budu što manji. Maksimalna vrijednost je manja ako se amplitude oscilacija snage smanjuju. Integral apsolutnog odstupanja je manji, ako iznos radne snage manje odstupa od stacionarne vrijednosti. Integral apsolutnog deriviranog odstupanja je manji, ako je odziv radne snage manje oscilatoran. Na osnovu tih kriterija i dijagrama u kojem su prikazani odzivi radne snage moguće je ocijeniti djelovanje PI regulatora napona i djelovanje neuronske mreže bez i sa stabilizirajućim efektom u kriterijskoj funkciji. Ta je ocjena donesena za pojedine grupe eksperimenata uz skokovitu promjenu postavne veličine napona.

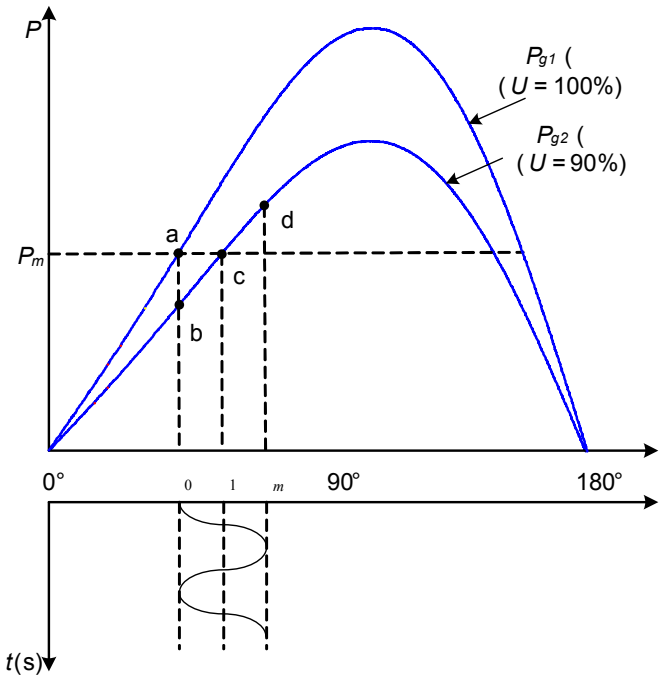
U eksperimentima bit će prikazani odzivi napona generatore, struje generatora, radne snage , jalove snage i struje uzbude generatora. Sve veličine izražene su relativnim jedinicama (p.u.) baznih veličina. Vrijeme trajanja pokusa je 2,7 sekundi s tim da se promjene postavne veličine napona događaju u 0.3 i 1.6 sekundi.

Eksperimenti su obavljeni za slučaj prikazan na slici 9.1. Generator je preko dva paralelna prijenosna voda i transformatora spojen na mrežu.



Slika 9.1 Konfiguracija spoja generatora na mrežu

Skokovitom promjenom referentne veličine napona generator zauzima novu radnu točku uz pojavu oscilacije radne snage kako je to prikazano na slijedećoj slici.



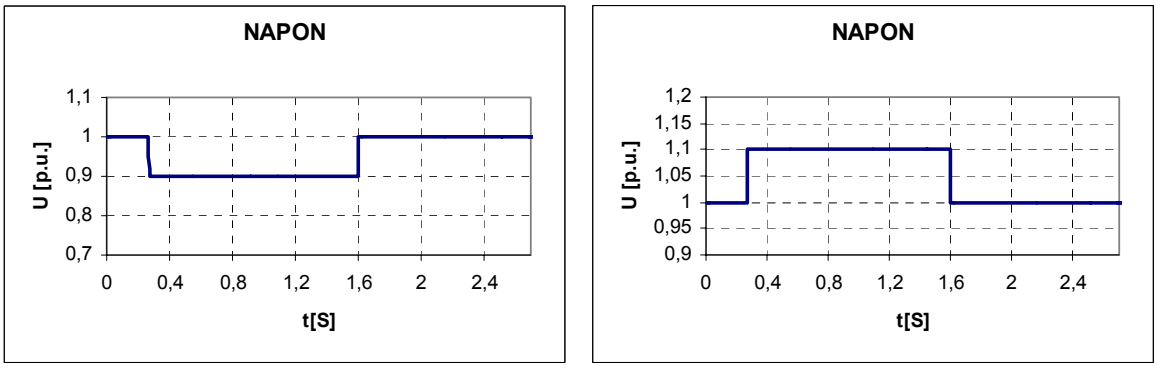
Slika 9.2 Oscilacije rotora sinkronog generatora pri skokovitoj promjeni referentne vrijednosti napona

Prije promjene napon generatora iznosi 1 p.u., mehanička ulazna snaga kojom je opterećen generator je P_m, a kut opterećenja iznosi δ_0. Generator se nalazi u točki a. Pri promjeni napona sa 1 p.u. na 0.9 p.u. generator prelazi s karakteristike $P_{g1}(\delta)$ na karakteristiku $P_{g2}(\delta)$, odnosno iz točke a u točku b. Kut opterećenja se na može trenutno promijeniti zbog tromosti mase, dok električna snaga generatora pada na iznos koji odgovara točki b. Razlika mehaničke i električne snage je snaga akceleracije P_a. U točki b snaga akceleracije je pozitivna što ubrzava rotor prema točki c. Time se povećava električna snaga, a smanjuje

snaga akceleracije. Kut opterećenja raste s iznosa δ_0 na iznos δ_1. Dakle, promjenom napona sa 1 p.u. na 0.9 p.u. u prvoj oscilaciji radna snaga generatora opada, a kut opterećenja raste. U točki c snaga akceleracije jednaka je nuli. Kut opterećenja je δ_1, a to je upravo onaj kut koji treba biti u stacionarnom stanju. Rotor ima maksimalnu brzinu te se nastavlja gibati dalje i predaje veću električnu snagu od ulazne mehaničke snage. Snaga akceleracije je negativna. U točki d rotor ima sinkronu brzinu ($\Delta\omega = 0$) i najveću negativnu akceleraciju, pa negativna snaga akceleracije $-P_a$ vraća generator prema točki c. Na putu od točke d do točke c generator daje veću električnu radnu snagu od ulazne mehaničke snage i pri tome se rotor usporava. U točki c električna i mehanička snaga su izjednačene, ali rotor je usporen i ne može više davati tu električnu snagu te se ne zadržava u točki c, nego ubrzava prema točki b. Cijeli proces se ponavlja, a zbog trenja i električnih gubitaka se s vremenom prigušuje te će promjena brzine i akceleracija postati približno jednake nuli [7].

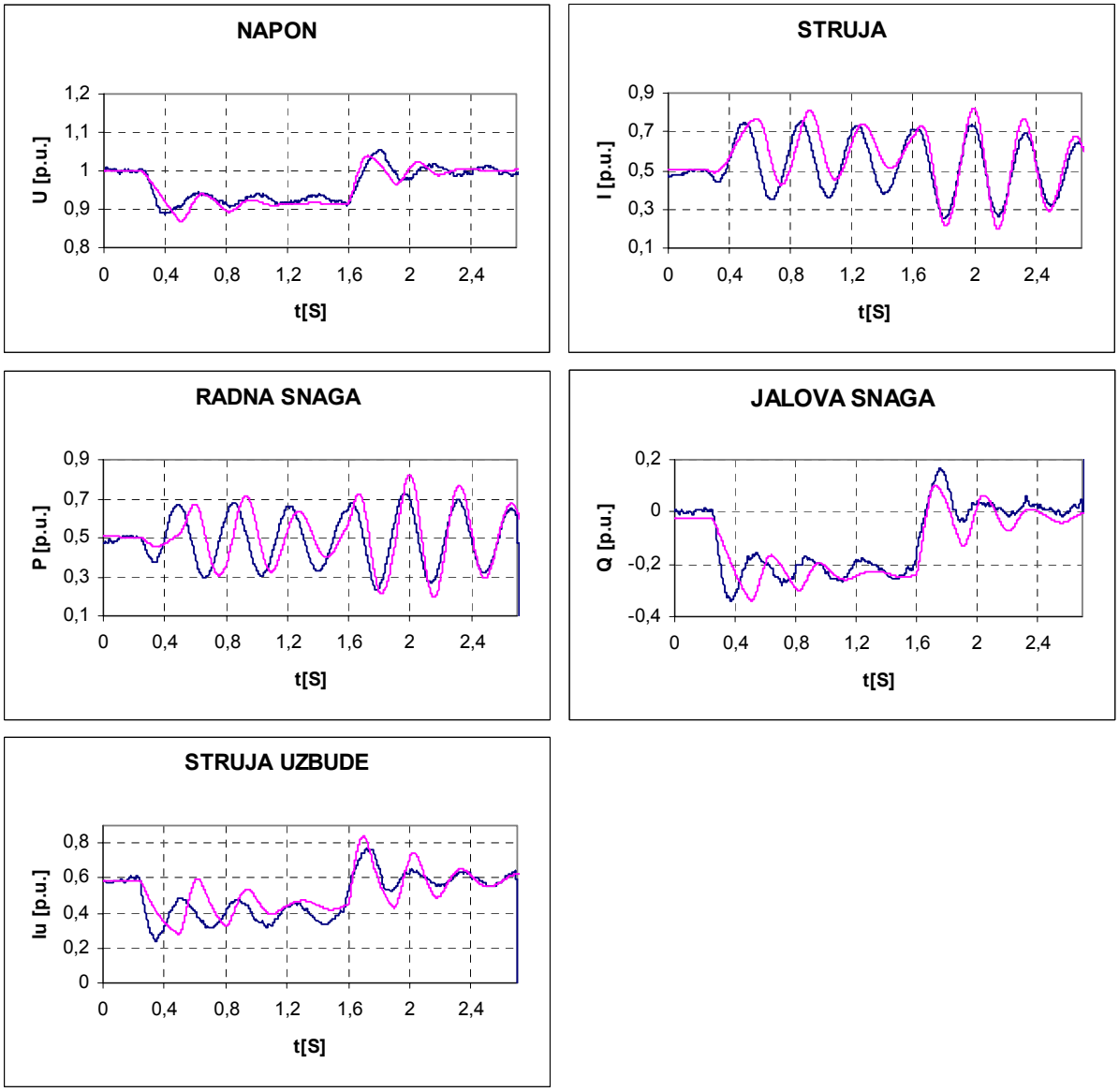
Gornje razmatranje nam pokazuje da se prilikom udarne promjene referentne vrijednosti napona generatora pojavljuju oscilacije radne snage.

Početni iznos referentne veličine napona generatora je 1 p.u. ako se izražavamo u relativnim jedinicama. Promjena referentne veličine napona prikazana je na slici 9.3. U tim uvjetima nijedna veličina ne postiže iznos ograničenja, a osigurava se prijelazni proces u kojem se može ispitati djelovanje proporcionalno integralnog regulatora napona te regulatora napona zasnovanog na neuronskoj mreži sa i bez stabilizirajućeg efekta u kriterijskoj funkciji. Eksperimenti su izvršeni za obje promjene referentne veličine napona, a pri tom generator predaje mreži približno 0.5 p.u. odnosno 0.8 p.u. radne snage.

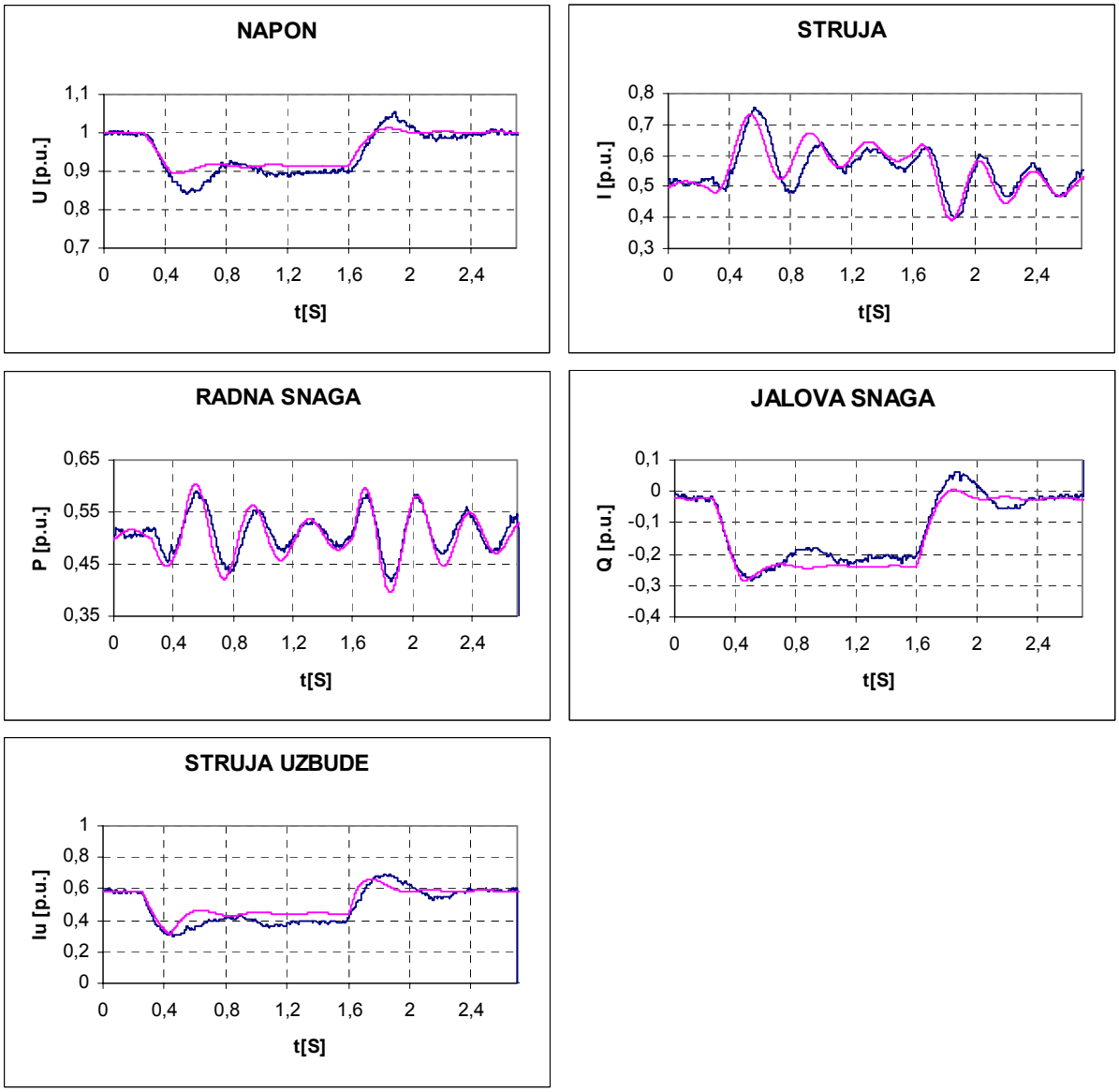


Slika 9.3 Promjene postavne veličine napona generatora

Na slikama 9.4, 9.5 i 9.6 prikazani su odzivi uz promjenu postavne veličine napona 1-0.9-1 p.u. te radnu snagu generatora $P_g \approx 0.5$ p.u.

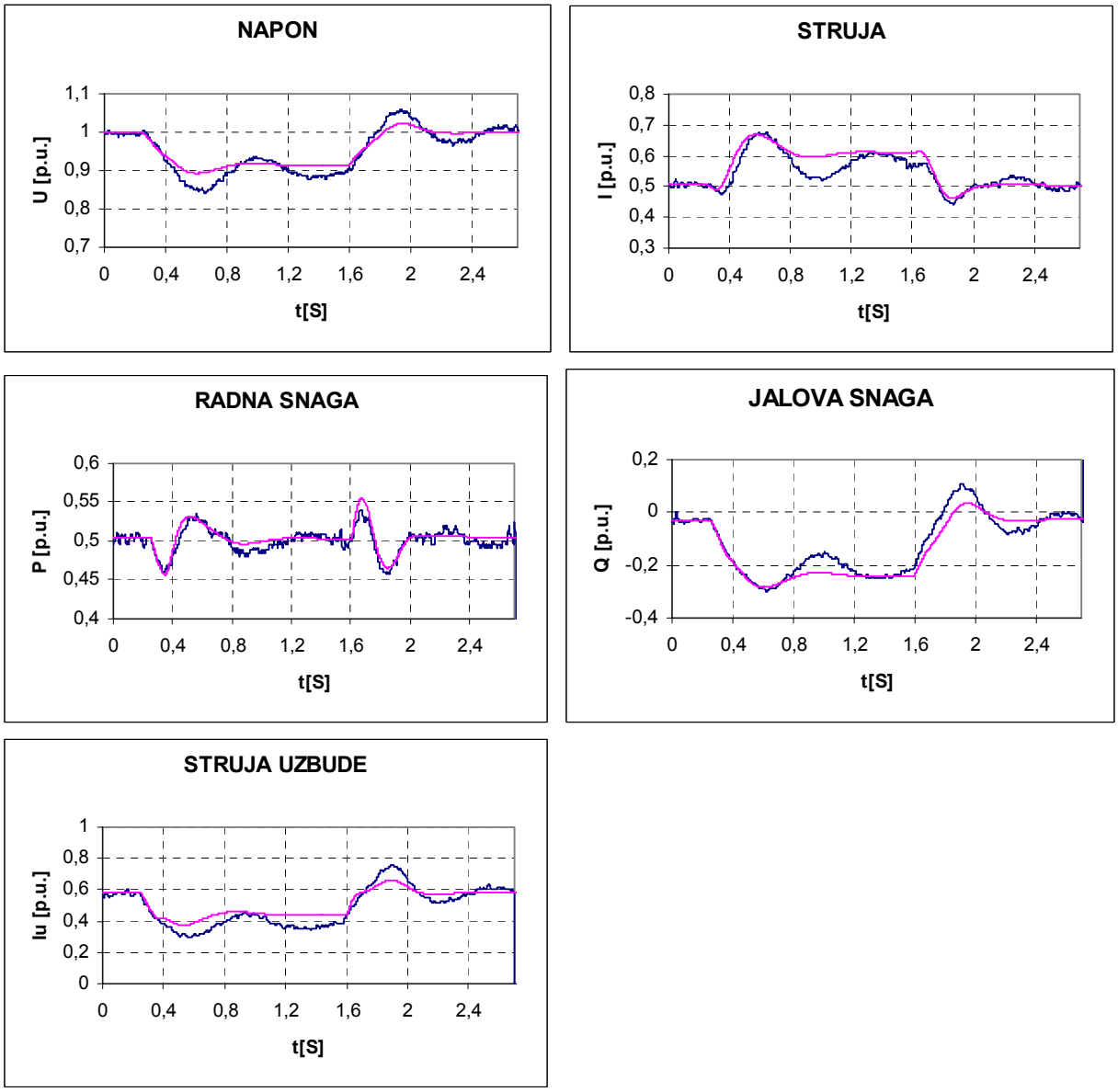


Slika 9.4 Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje PI regulatora napona uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u.



Slika 9.5. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži bez stabilizirajućeg efekta u kriterijskoj funkciji uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora

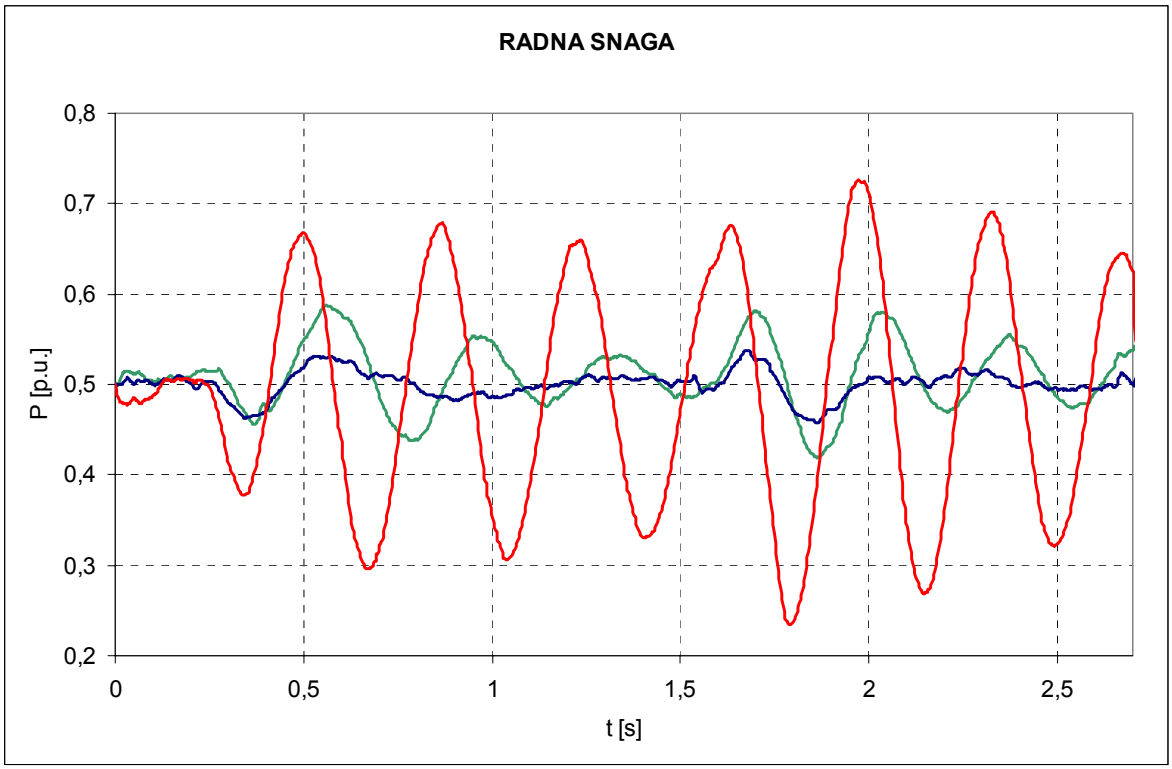
$$P_g \approx 0.5 \text{ p.u.}$$



Slika 9.6. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži sa stabilizirajućim efektom u kriterijskoj funkciji uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora

$$P_g \approx 0.5 \text{ p.u.}$$

Na slici 9.7 prikazani su eksperimentalni odzivi radne snage generatora sa slika 9.4, 9.5 i 9.6.

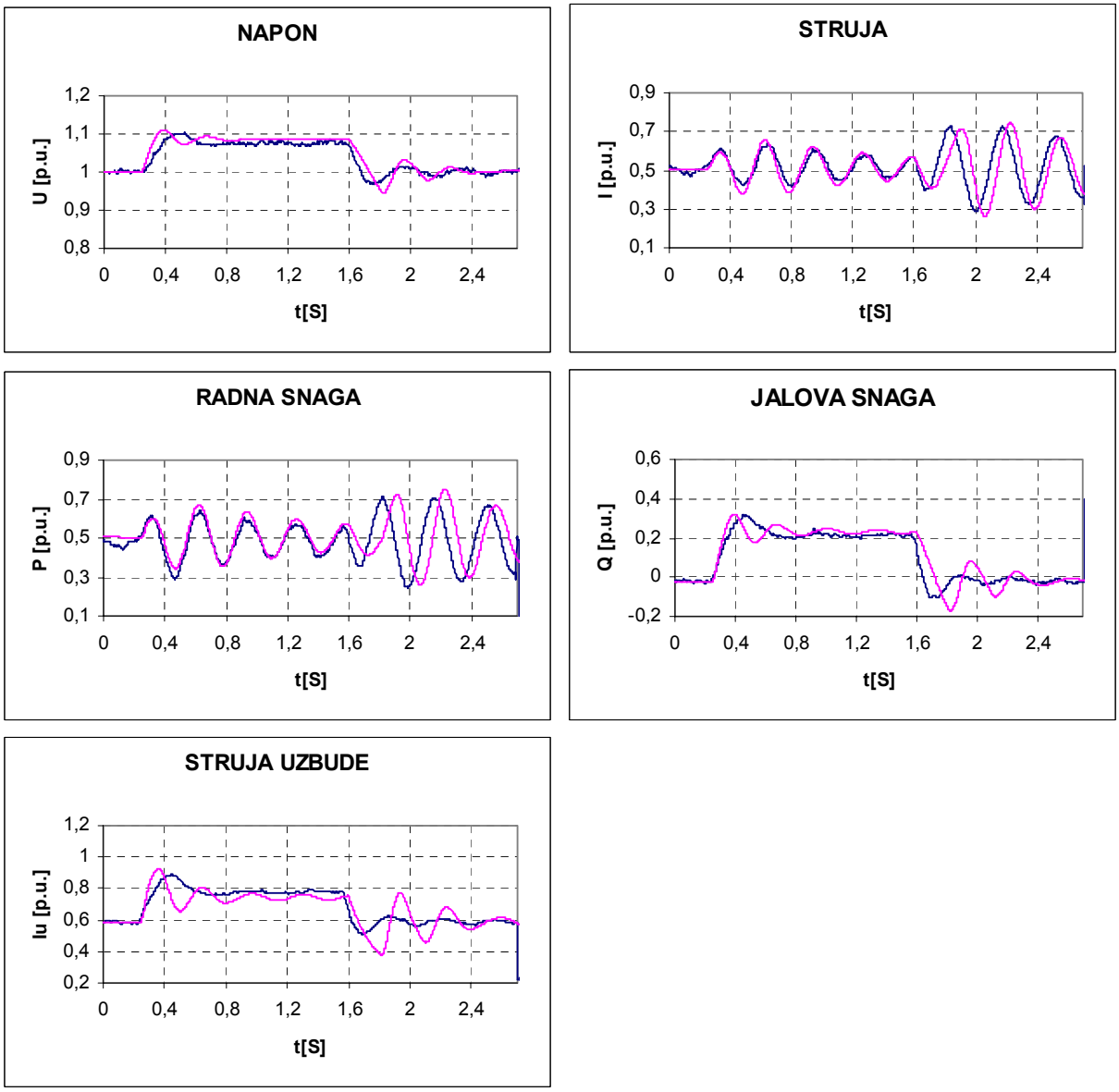


Slika 9.7 Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u. (PI regulator crveno, neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

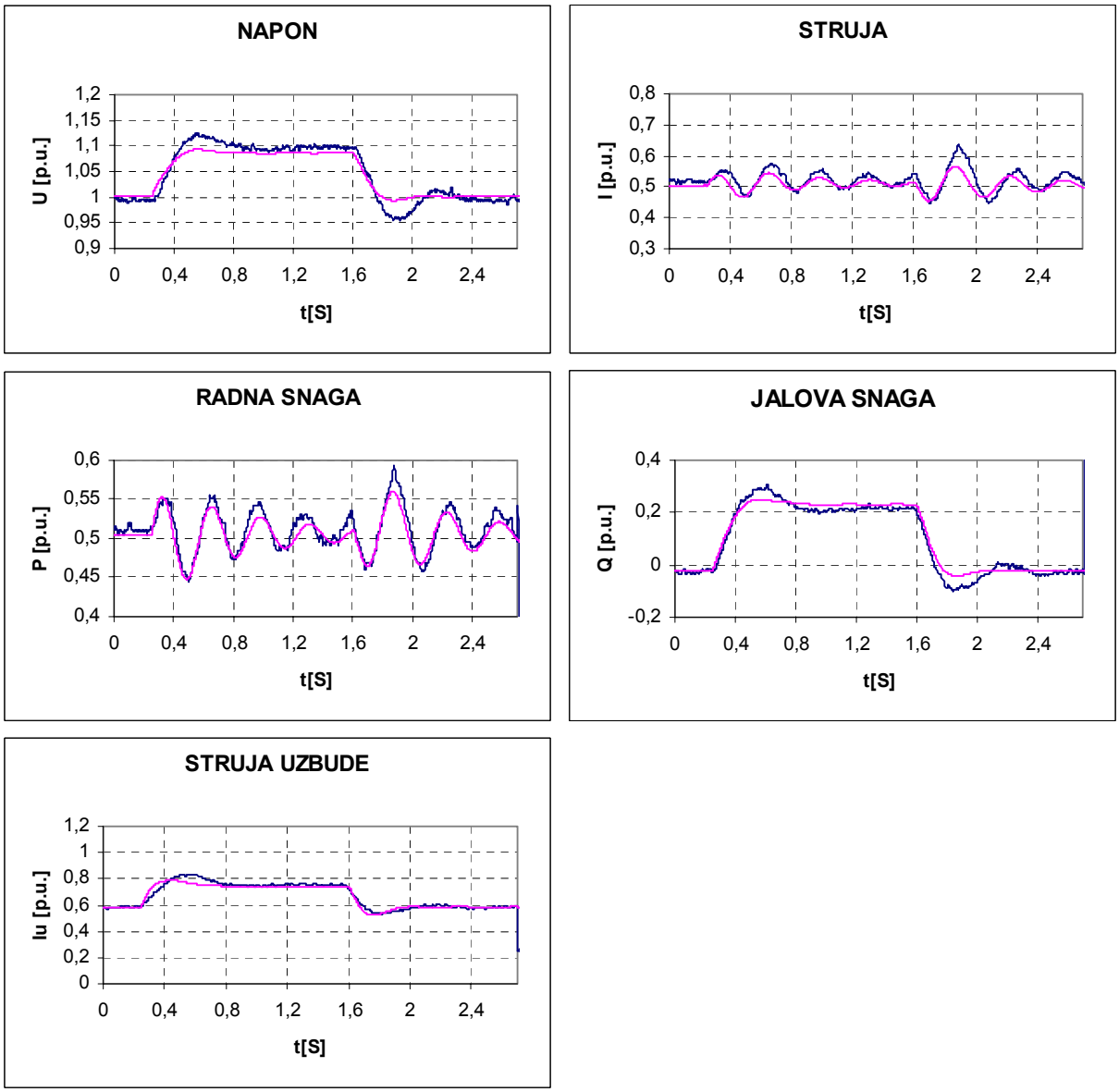
Promjena postavne veličine napona 1-0.9-1 p.u. $P_g \approx 0.5$ P.U.	Maksimalna vrijednost *MAX* (p.u.)	Minimalna vrijednost *MIN* (p.u.)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.7	0.25	0.3012	0.5358
Neuronska mreža	0.6	0.4	0.1543	0.2645
Neuronska mreža s stabilizirajućim efektom	0.54	0.45	0.07143	0.1019

Tablica 9.1 Kriteriji kvalitete odziva radne snage generatora

Na slikama 9.8, 9.9 i 9.10 prikazani su odzivi uz promjenu postavne veličine napona 1-1.1-1 p.u. te radnu snagu generatora $P_g \approx 0.5$ p.u..

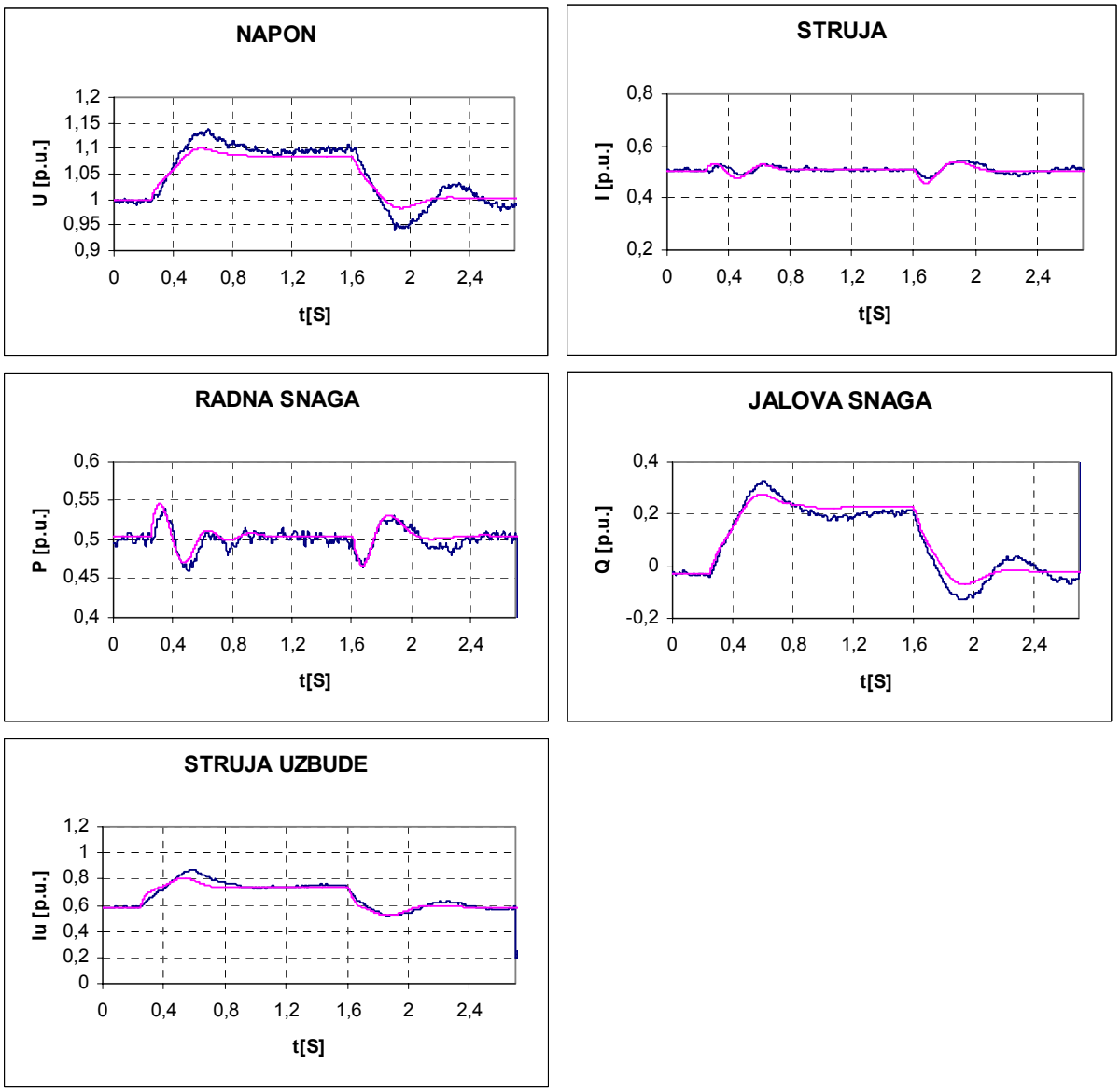


Slika 9.8. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje PI regulatora napona uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u.



Slika 9.9. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži bez stabilizirajućeg efekta u kriterijskoj funkciji uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora

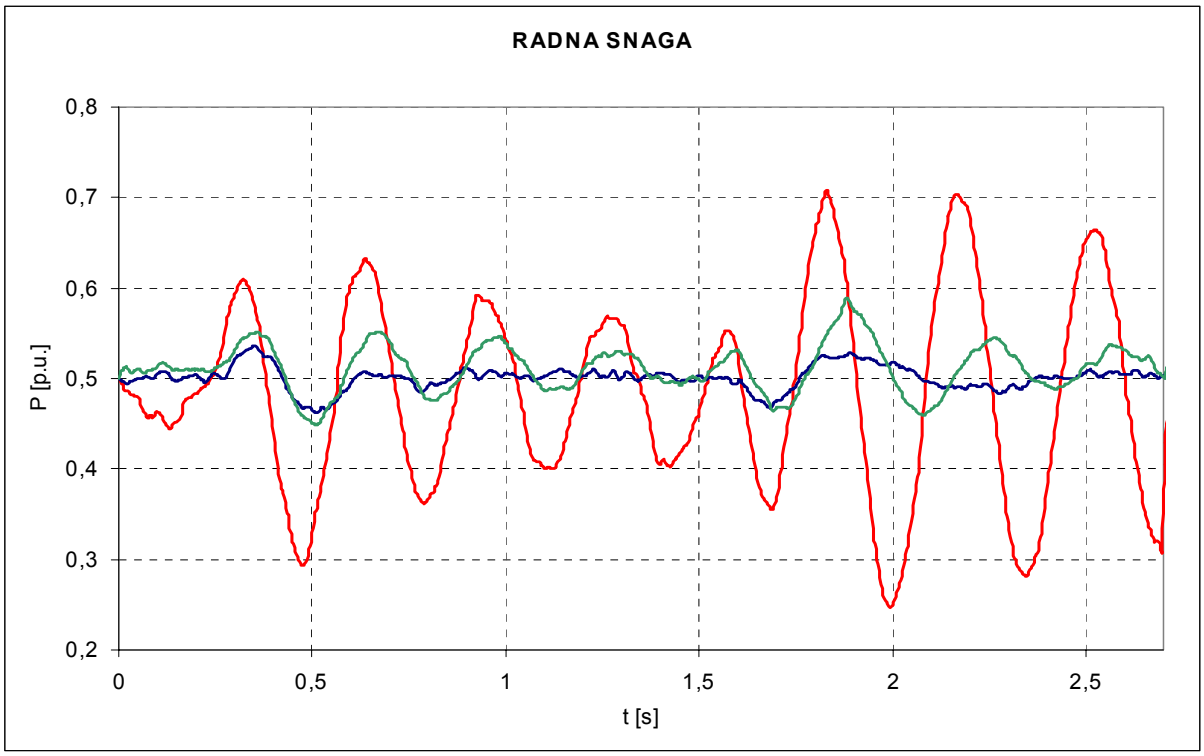
$P_g \approx 0.5$ p.u.



Slika 9.10. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži sa stabilizirajućim efektom u kriterijskoj funkciji uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora

$$P_g \approx 0.5 \text{ p.u.}$$

Na slici 9.11 prikazani su eksperimentalni odzivi radne snage generatora sa slika 9.8, 9.9 i 9.10.

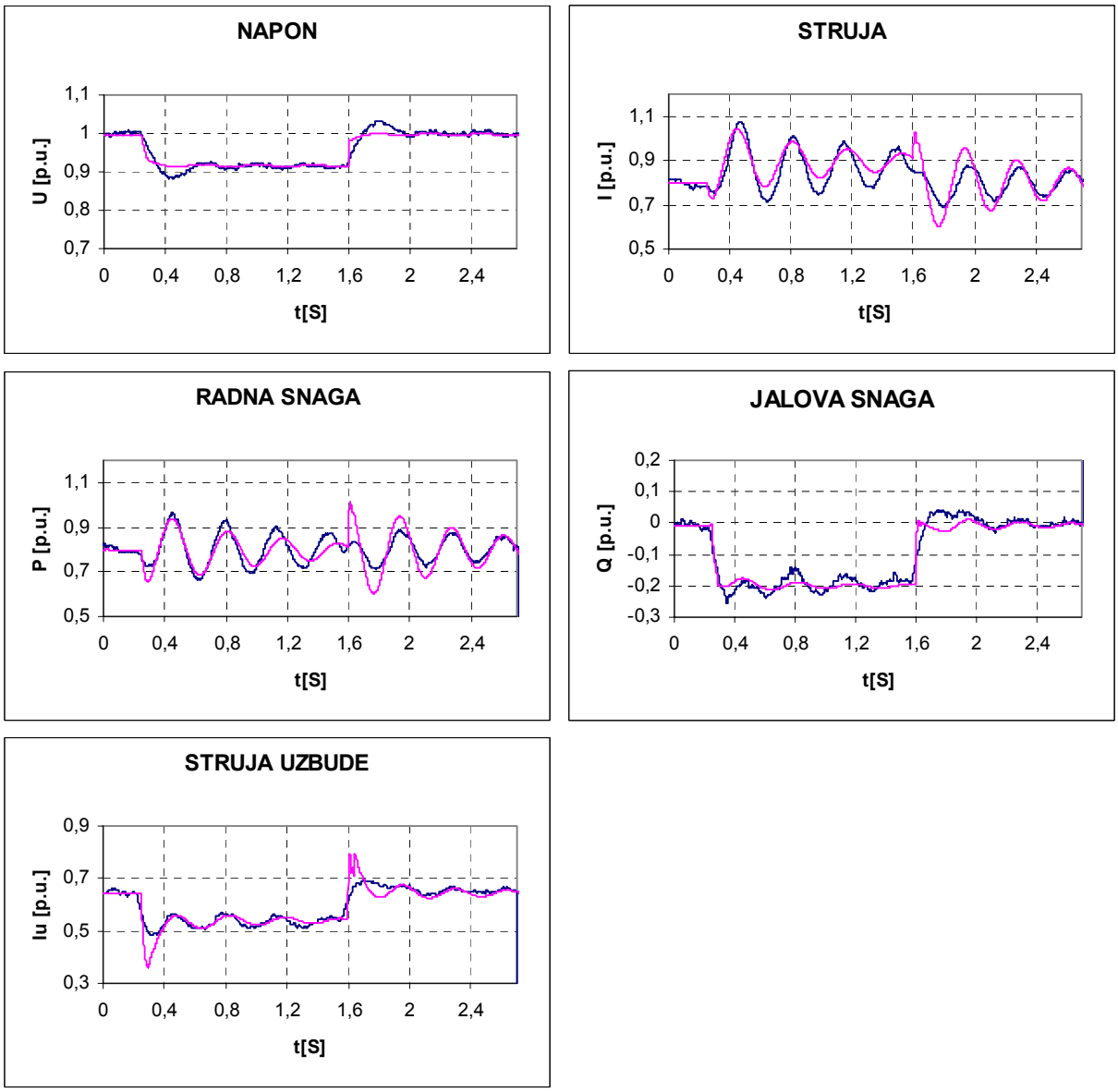


Slika 9.11. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u. (PI regulator crveno, neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

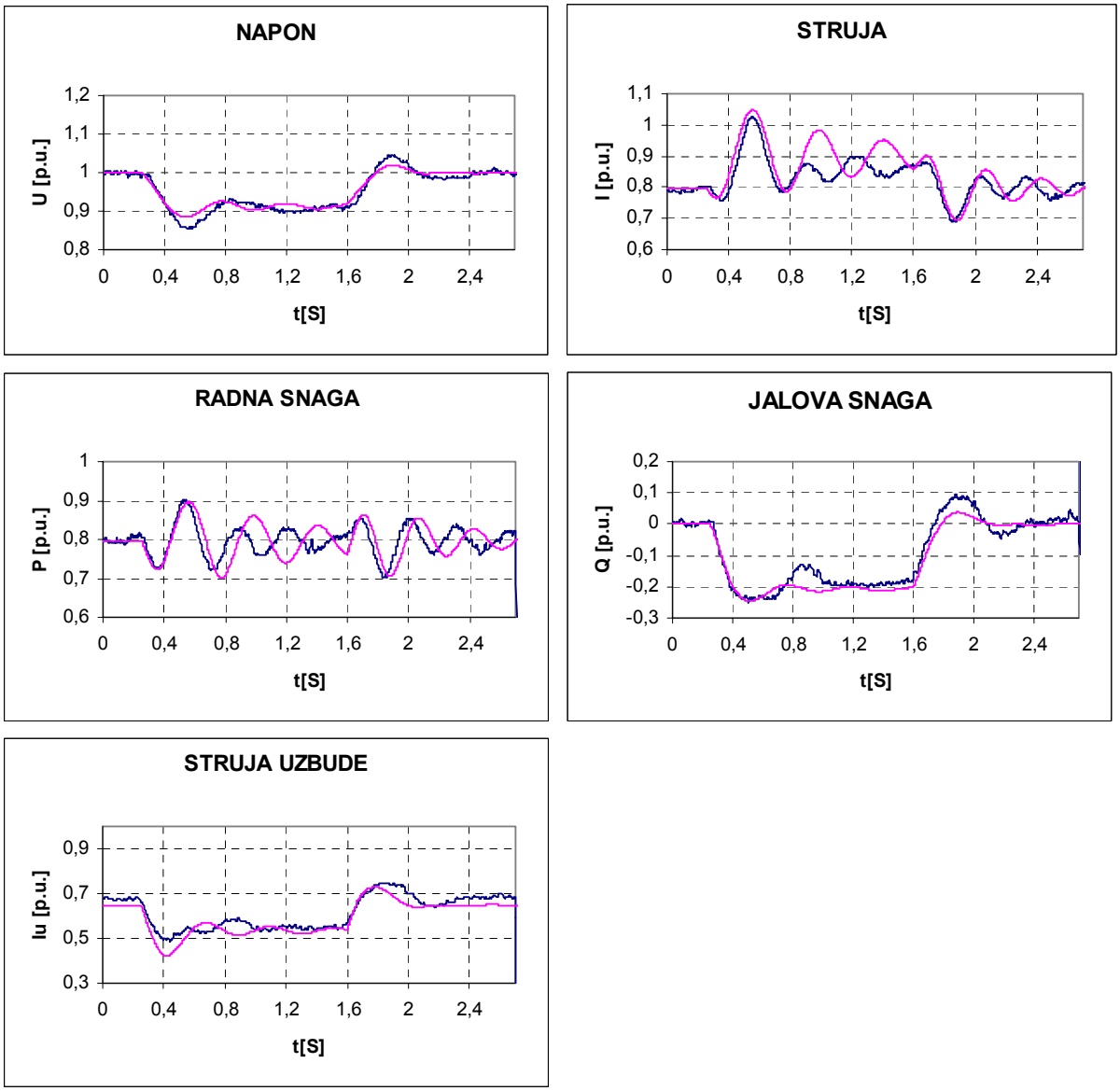
Promjena postavne veličine napona 1-09-1p.u. $P_g \approx 0.5$ P.U.	Maksimalna vrijednost *MAX* (p.u.)	Minimalna vrijednost *MIN* (p.u.)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.6	0.25	0.1898	0.4254
Neuronska mreža	0.57	0.45	0.08238	0.1496
Neuronska mreža s stabilizirajućim efektom	0.53	0.44	0.03893	0.06011

Tablica 9.2. Kriteriji kvalitete odziva radne snage generatora

Na slikama 9.12, 9.13 i 9.14 prikazani su odzivi uz promjenu postavne veličine napona 1-0.9-1 p.u. te radnu snagu generatora $P_g \approx 0.8$ p.u..

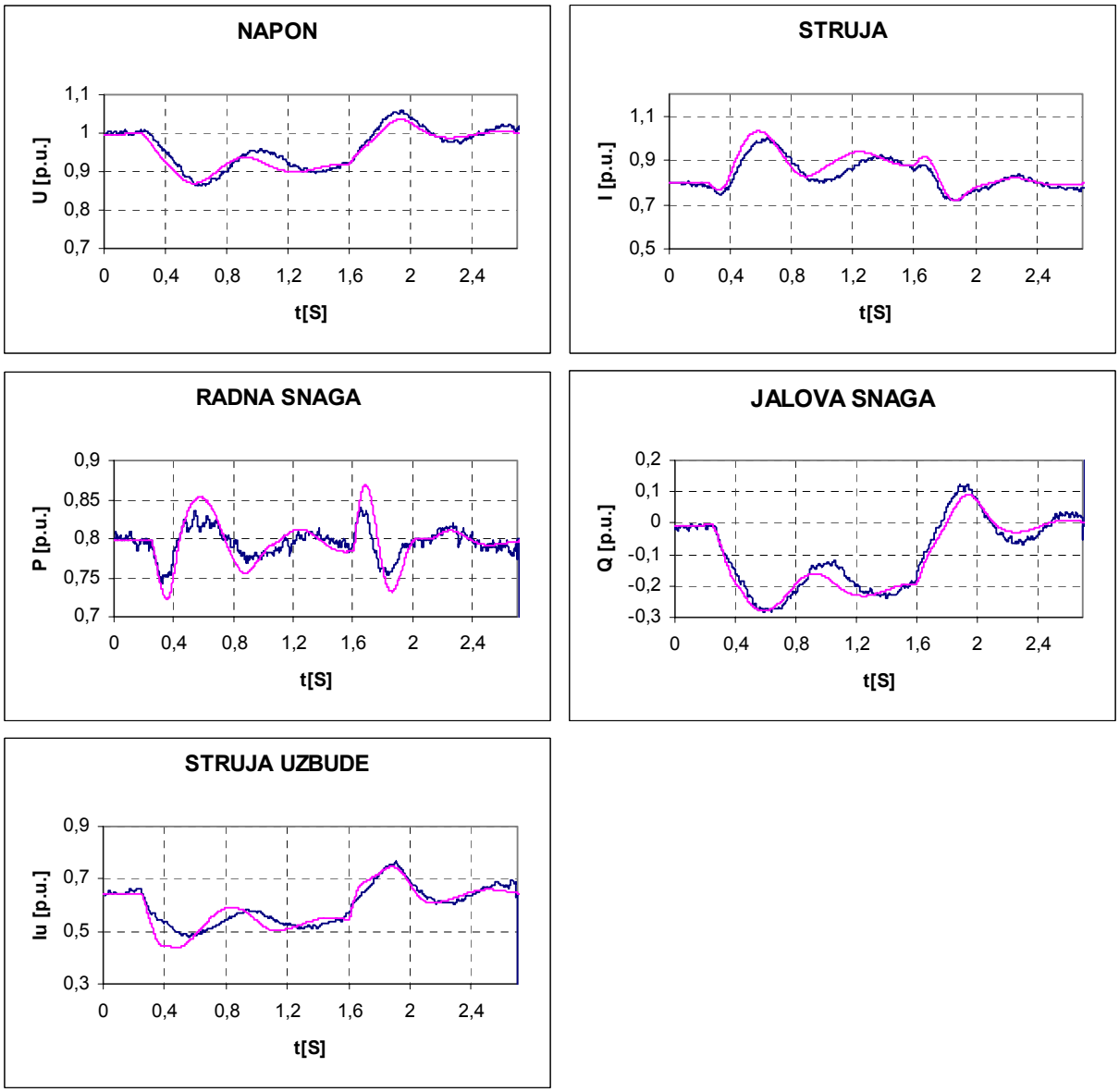


Slika 9.12. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje PI regulatora napona uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u.



Slika 9.13. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži bez stabilizirajućeg efekta u kriterijskoj funkciji uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora

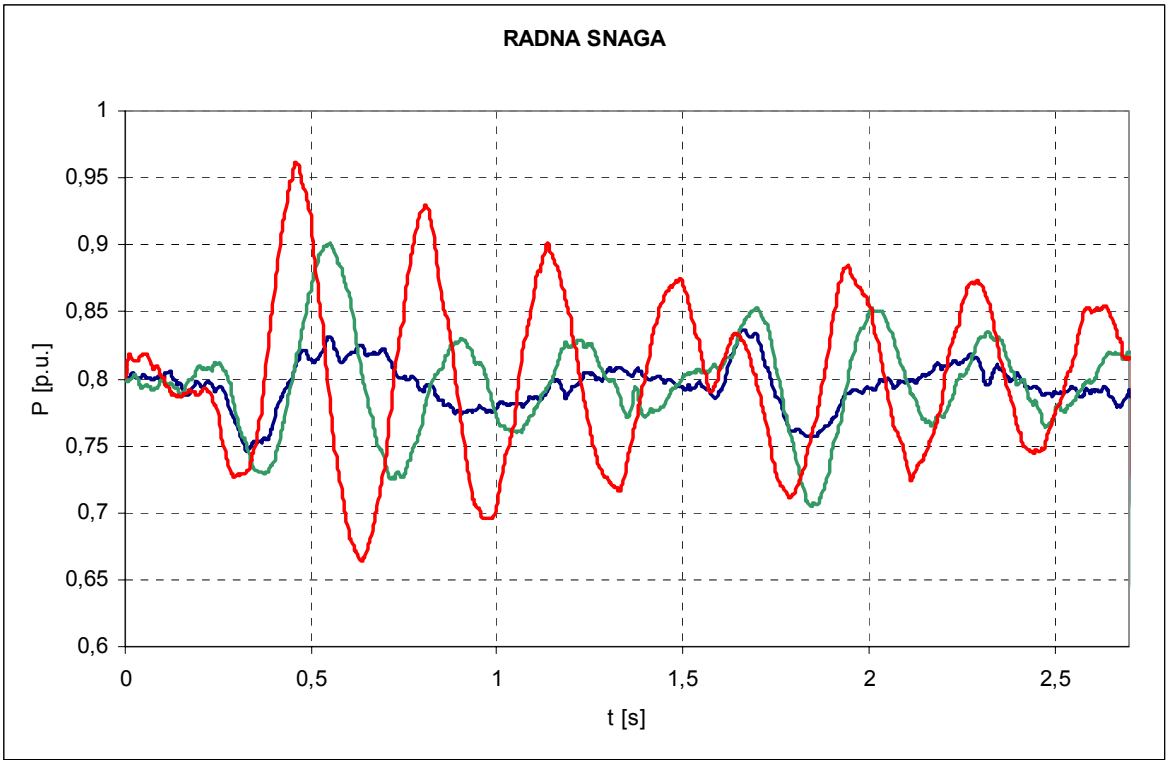
$$P_g \approx 0.8 \text{ p.u.}$$



Slika 9.14. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži sa stabilizirajućim efektom u kriterijskoj funkciji uz promjenu referentne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora

$$P_g \approx 0.8 \text{ p.u.}$$

Na slici 9.15 prikazani su eksperimentalni odzivi radne snage generatora sa slika 9.12, 9.13 i 9.14.

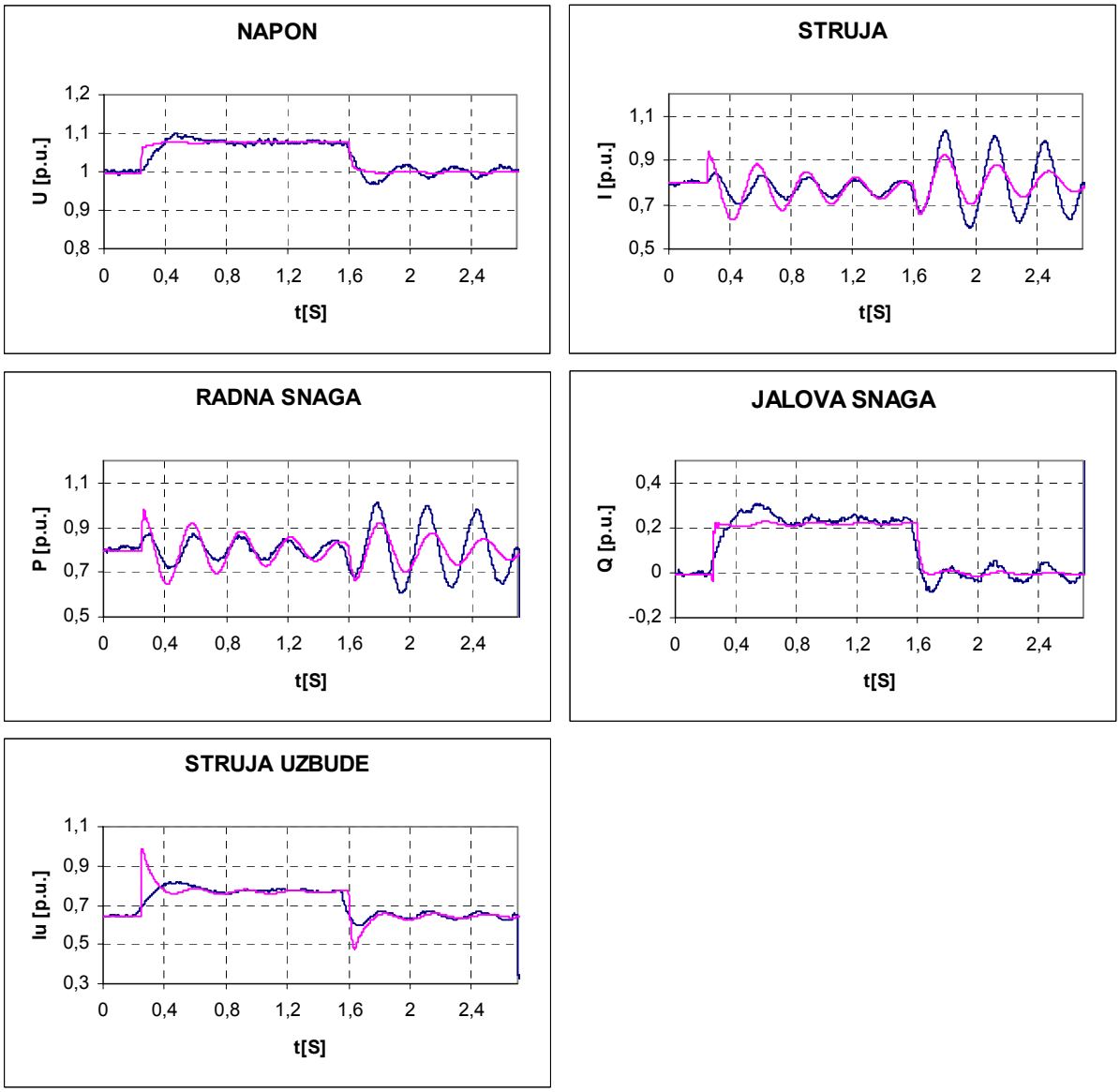


Slika 9.15. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u. (PI regulator crveno, neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

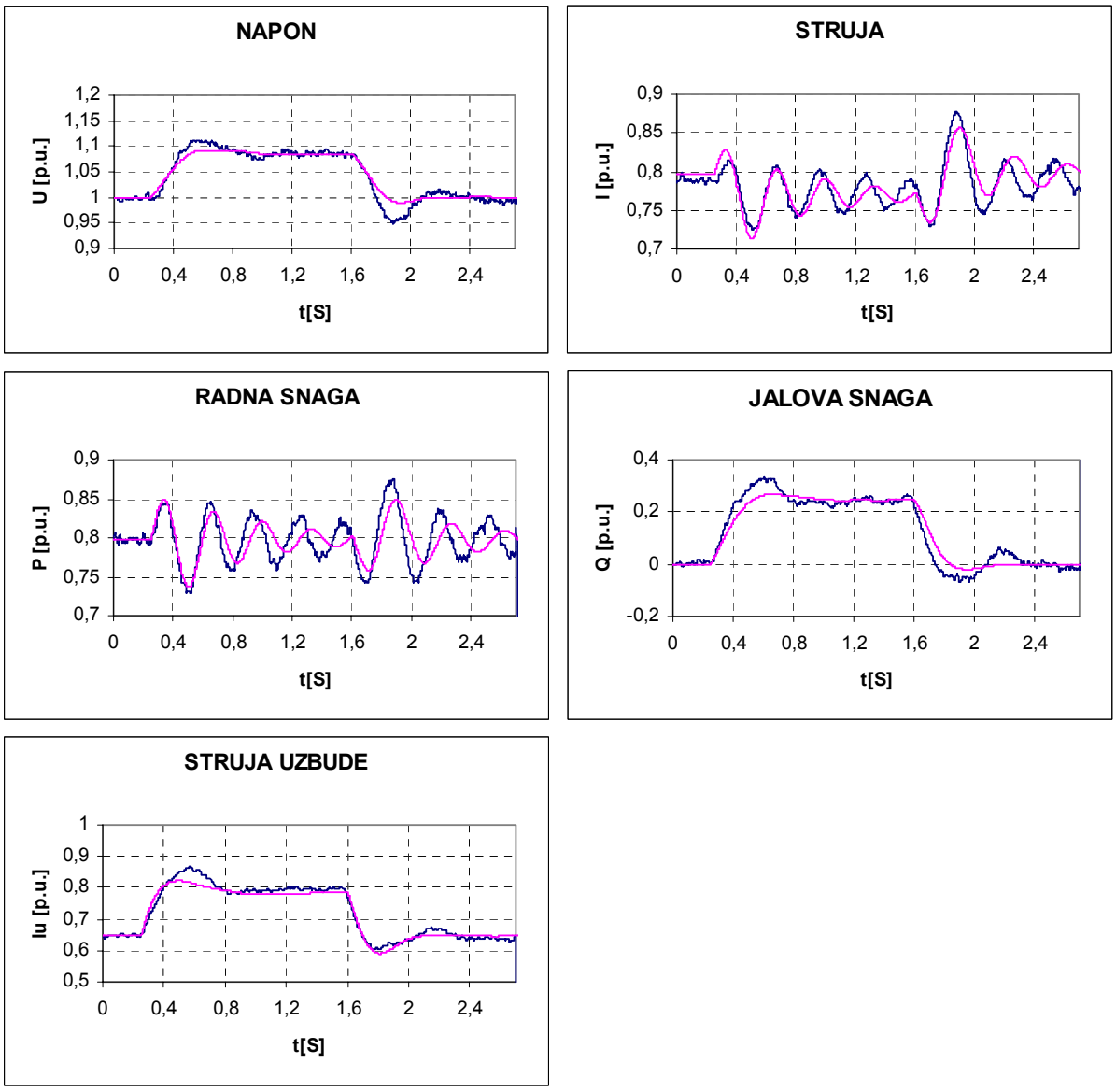
Promjena postavne veličine napona 1-0.9-1 p.u. $P_g \approx 0.8$ P.U.	Maksimalna vrijednost *MAX* (p.u.)	Minimalna vrijednost *MIN* (p.u.)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.97	0.68	0.3142	0.5541
Neuronska mreža	0.9	0.71	0.1582	0.2563
Neuronska mreža s stabilizirajućim efektom	0.85	0.75	0.07143	0.1019

Tablica 9.3. Kriteriji kvalitete odziva radne snage generatora

Na slikama 9.16, 9.17 i 9.18 prikazani su odzivi uz promjenu postavne veličine napona 1-1.1-1 p.u. te radnu snagu generatora $P_g \approx 0.8$ p.u.

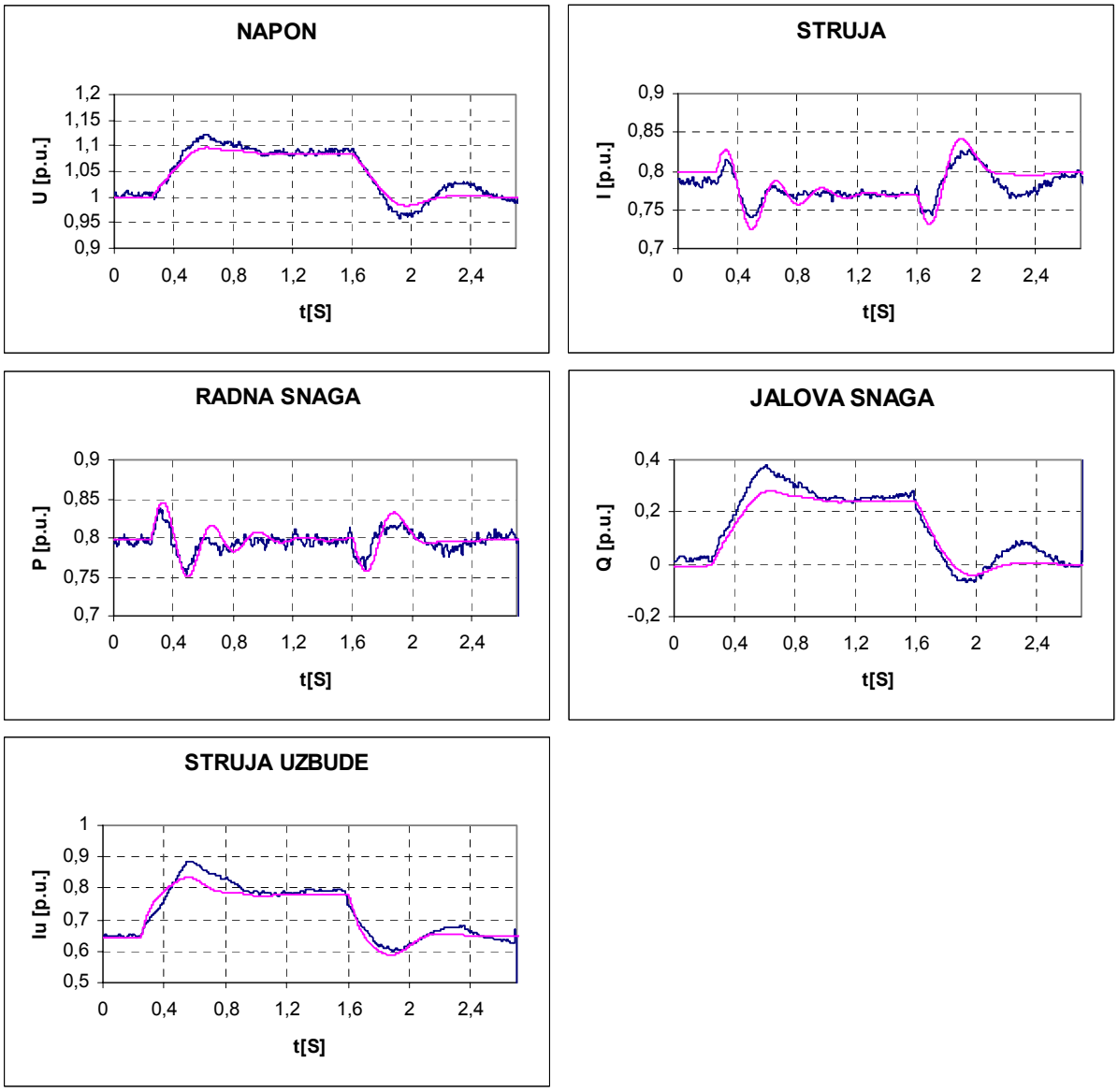


Slika 9.16. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje PI regulatora napona uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u.



Slika 9.17. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži bez stabilizirajućeg efekta u kriterijskoj funkciji uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora

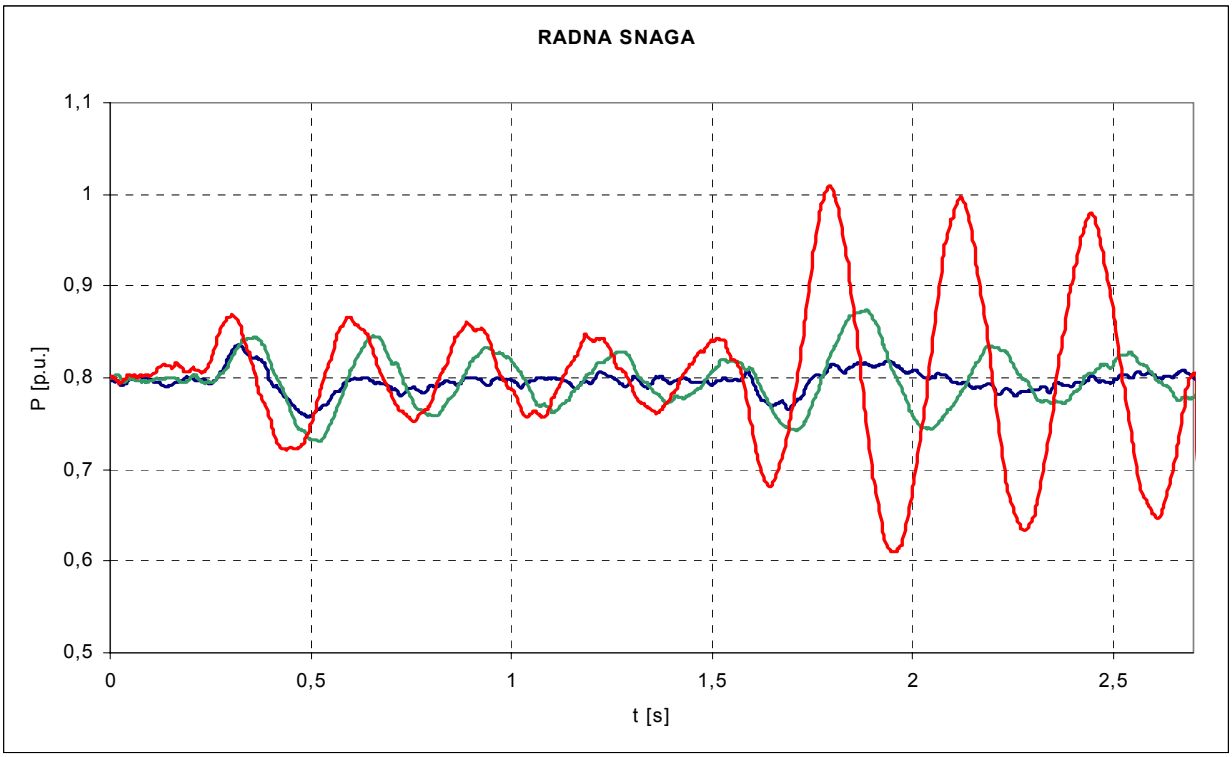
$$P_g \approx 0.8 \text{ p.u.}$$



Slika 9.18. Eksperimentalni (plavo) i simulacijski (rozo) odzivi za djelovanje regulatora napona zasnovanog na neuronskoj mreži sa stabilizirajućim efektom u kriterijskoj funkciji uz promjenu referentne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora

$$P_g \approx 0.8 \text{ p.u.}$$

Na slici 9.19 prikazani su eksperimentalni odzivi radne snage generatora sa slika 9.16, 9.17 i 9.18.



Slika 9.19. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u. (PI regulator crveno, neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

Promjena postavne veličine napona 1-1.1-1 p.u. $P_g \approx 0.8$ P.U.	Maksimalna vrijednost *MAX* (p.u.)	Minimalna vrijednost *MIN* (p.u.)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	1	0.6	0.1967	0.3691
Neuronska mreža	0.88	0.72	0.142	0.2527
Neuronska mreža s stabilizirajućim efektom	0.83	0.75	0.04245	0.07226

Tablica 9.4. Kriteriji kvalitete odziva radne snage generatora

Eksperimentalni i simulacijski odzivi uz skokovitu promjenu postavne veličine napona pokazuju pozitivno djelovanje neuronske mreže bez stabilizirajućeg efekta implementiranog u kriterijskoj funkciji a posebno kad je implementiran stabilizirajući efekt. Kod promjene postavne veličine napona 1-0.9-1 p.u. i 1-1.1-1 p.u. nastaju oscilacije radne snage. Uz djelovanje proporcionalno integralnog regulatora napona oscilacije iznose i do 0.2 p.u. radne snage od stacionarne vrijednosti. Ovisno o vrsti eksperimenta. *IAE* kriterij pokazuje značajno odstupanje odziva radne snage od stacionarne vrijednosti a vrijednost mu se kreće od 0.1898 pa sve do 0.3142. *IAED* kriterij također pokazuje oscilatornost odziva radne snage a vrijednost mu je u rasponu od 0.3691 pa do 0.5541 ovisno o vrsti eksperimenta . Kada imamo djelovanje neuronske mreže bez stabilizirajućeg efekta , oscilacije radne snage su nešto prigušenije te se kreću u rasponu do 0.1 p.u. od stacionarne vrijednosti radne snage. *IAE* kriterij daje vrijednosti od 0.0823 pa do 0.1582 i *IAED* kriterij se kreću u rasponu od 0.1496 pa do 0.2645 što je oko 2 puta manje u odnosu u odnosu na PI regulator (ovisno o radnoj točki generatora). Još su više smanjeni ukoliko se koristi neuronska mreža s stabilizirajućim efektom u kriterijskoj funkciji IAE kriterij se u ovom slučaju kreće od iznosa 0.03893 pa do 0.07143 dok IAED kriterij daje vrijednosti u rasponu od 0.06011 pa do 0.1019 (za oko 4 puta ukoliko se gleda u odnosu na iznose kriterija dobivenih s PI regulatorom) što znači da ovako realiziran regulator napona pozitivno djeluje na prigušenje oscilacija radne snage generatora. Osim po numeričkim kriterijima kvalitete odziva vidimo da su i maksimalne vrijednosti amplituda oscilacije snage prigušene tako da se maksimum amplituda za PI regulator kreću od 0.6 do 1 p.u. dok se istovremeno za neuronsku mrežu te vrijednosti kreću od 0.5 do 0.85 p.u.

Neuronske mreža povoljno uteče na prigušenje oscilacija radne snage, to naročito dolazi do izražaja kad se uključi stabilizirajući efekt u kriterijskoj funkciji.

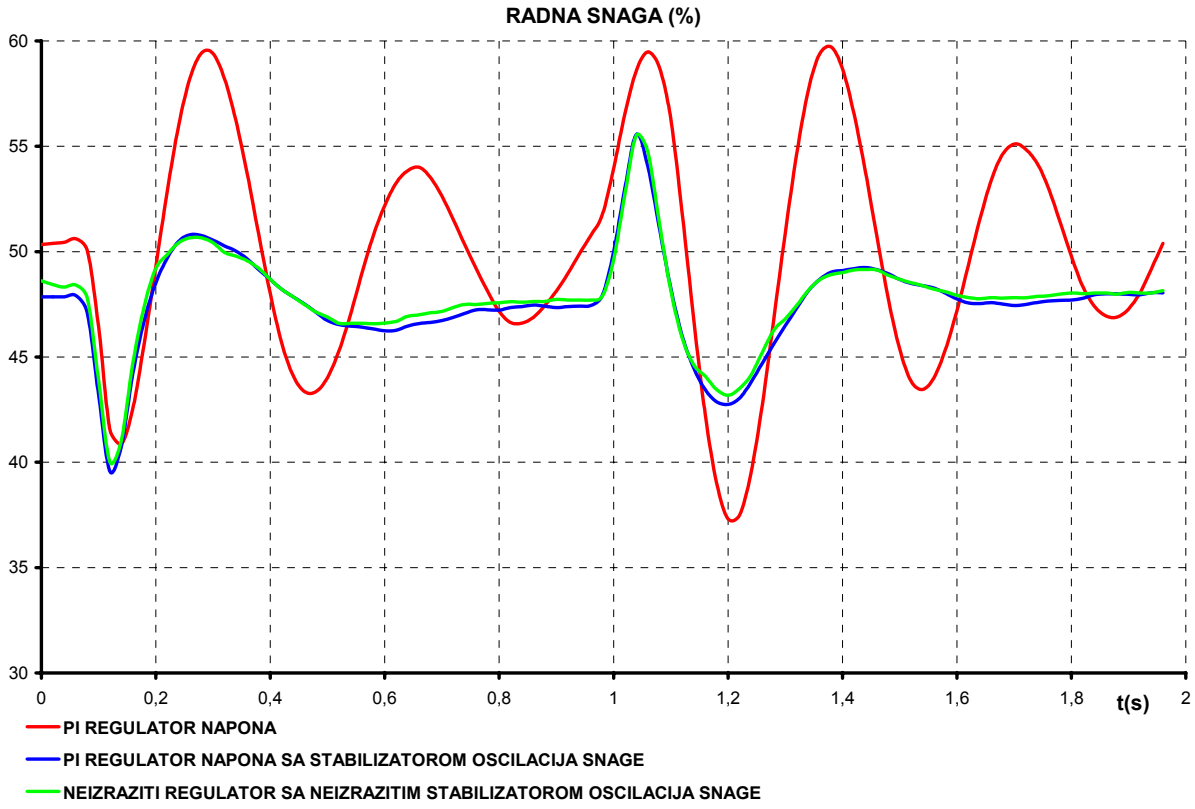
10 USPOREBE DJELOVANJA SUSTAVA NEURONSKE I NEIZRAZITE REGULACIJE UZBUDE GENERATORA

Na istom sinkronom generatoru implementirane su različite vrste upravljanje. Posebno je zanimljiv neizraziti regulator čija je realizacija prikazana u literaturi [7]. Ovaj algoritam upravljanja realiziran je u drugom digitalnom sustavu upravljanja uzbudom generatora ali pokusi napravljeni u tom radu istovjetni su eksperimentima izvedenim u ovom radu tako da su pogodni za usporedbu. Usporedit će se slučajevi skokovite promjene reference napona te će biti prikazani odzivi radne snage generatora. Isto tako navest će se tablično rezultati IAE i IAED kriterija, te MIN i MAX kriterija.

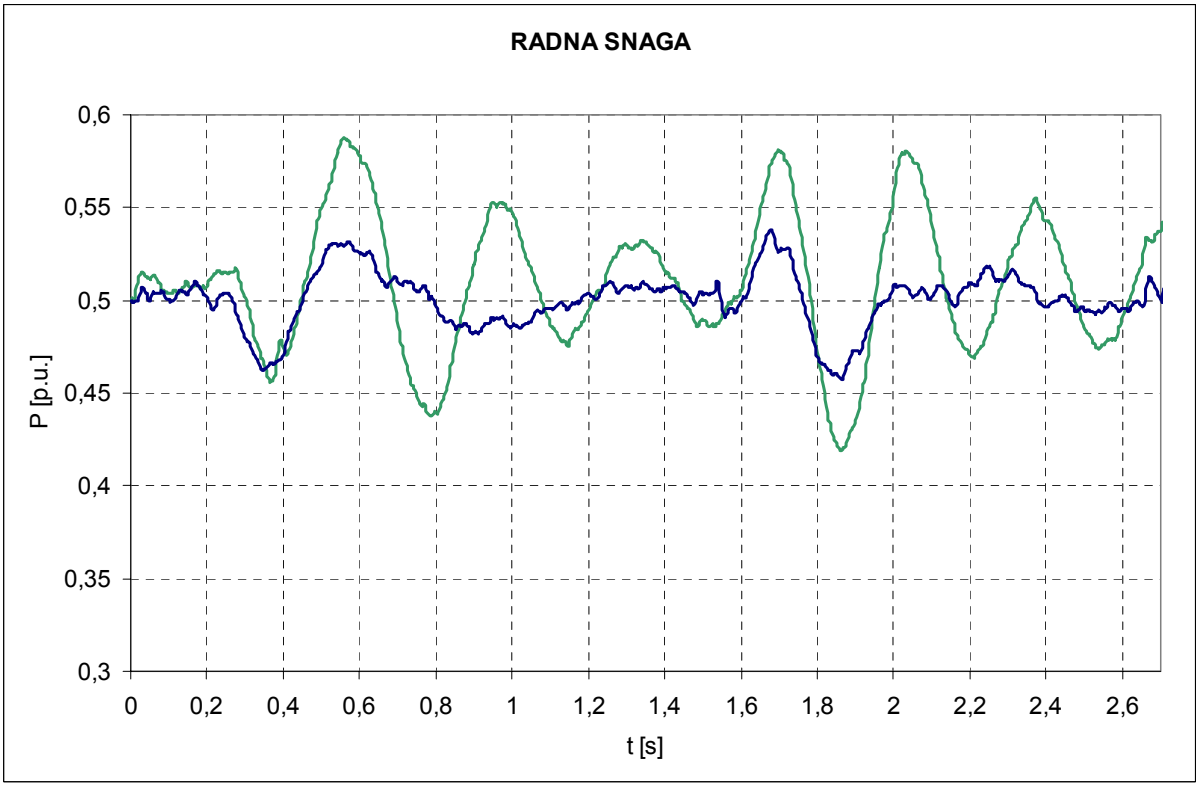
Rad neizrazitog regulatora napona odvija se u polarnom koordinatnom sustavu faznog prostora napona. Ulazi u neizraziti regulator napona su greška napona i derivacija greške napona i te dvije veličine čine fazni prostor napona. Središte faznog prostora napona je željena točka ravnoteže u kojoj su i greška napona i derivacija greške napona jednake nuli. Na osnovu stanja greška napona/derivacija greške napona generatora i skupine jednostavnih neizrazitih upravljačkih pravila određen je izlazni signal koji će pomaknuti trenutno stanje generatora prema središtu faznog prostora napona. Izlazni signal neizrazitog regulatora napona je postava veličina struje uzbude, jer je neizraziti regulator napona nadređen proporcionalnom regulatoru struje uzbude[7].

Rezultati dobiveni u ovom radu izraženi su u postotnoj veličini od nazivnih veličina tako da će se odmah moći uspoređivati s rezultatima dobivenim u ovom radu koji su izraženi u relativnim jedinicama u odnosu na nazivne veličine (p.u.).

Na slici 10.1**.** i slici 10.2. prikazani su odzivi za slučaj skokovite promjene reference napona i to 1-0.9-1 p.u. uz $P_g \approx 0.5$ p.u, dok su tablično ti rezultati prikazani u tablici 10.1.



Slika 10.1. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 100-90-100 % pri radnoj snazi generatora $P_g \approx 50$ %

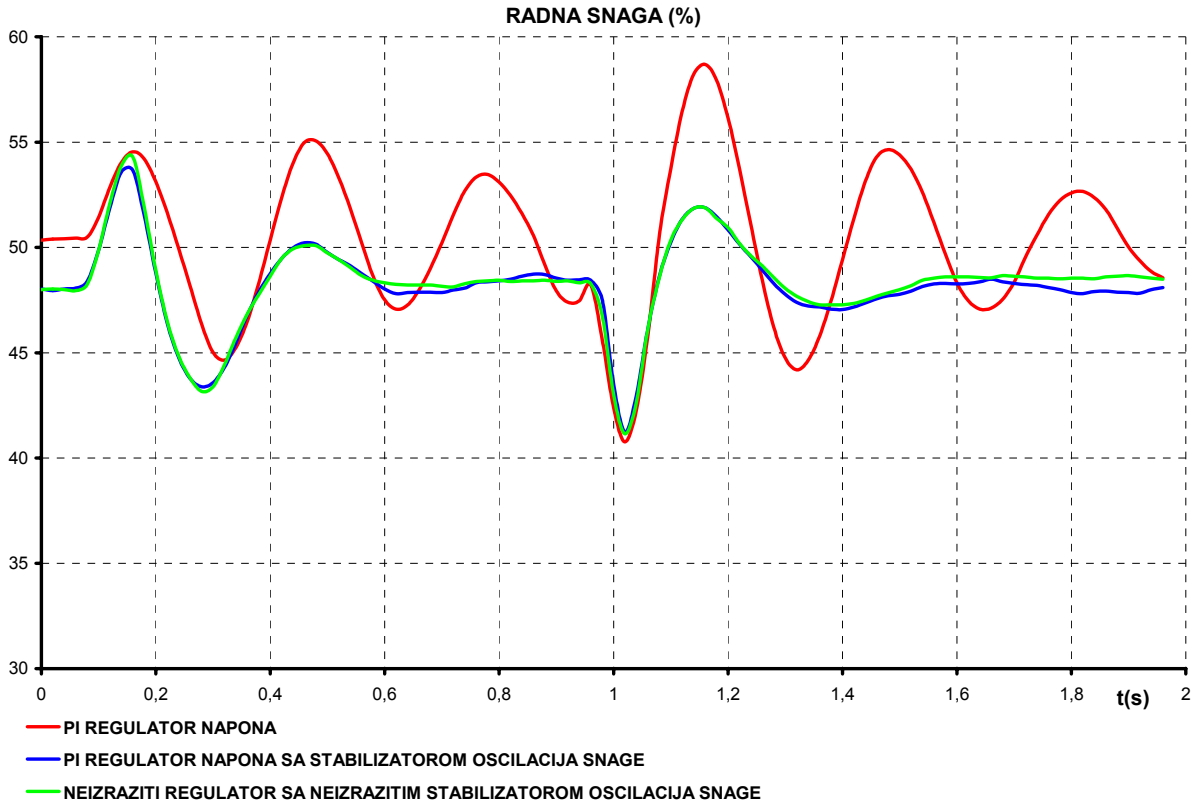


Slika 10.2. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u. (neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

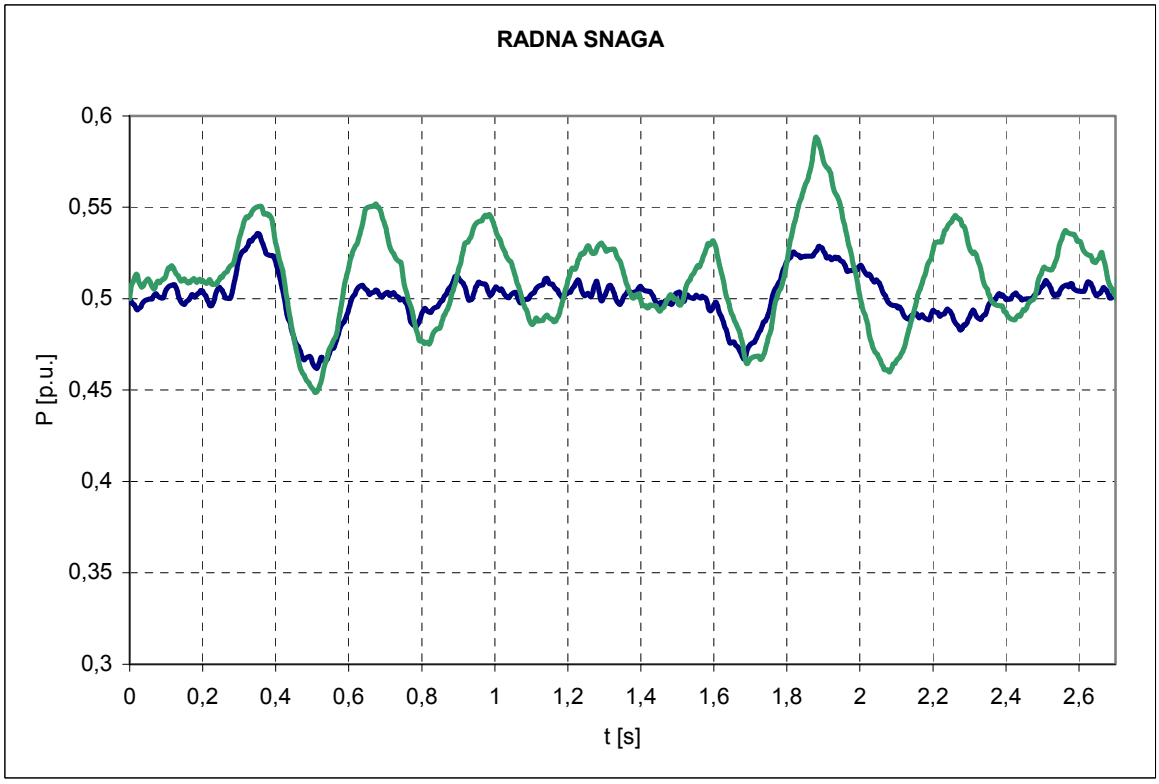
Promjena postavne veličine napona 100-90-100 % $P_g \approx 50$ %	Maksimalna vrijednost *MAX* (%)	Minimalna vrijednost *MIN* (%)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.59	0.37	0.389	0.279
PI regulator napona sa stabilizatorom	0.55	0.39	0.255	0.097
Neizraziti regulator napona s neizrazitim stabilizatorom	0.55	0.40	0.235	0.09
Neuronska mreža	0.6	0.4	0.1543	0.2645
Neuronska mreža s stabilizirajućim efektom	0.54	0.45	0.07143	0.1019

Tablica 10.1 Kriteriji kvalitete odziva za različite tipove regulatora za promjenu reference 1-0.9-1 p.u. napona i $P_g \approx 0.5$ p.u.

Na slici 10.3. i slici 10.4. prikazani su odzivi za slučaj skokovite promjene reference napona i to 1-1.1-1 p.u. uz $P_g \approx 0.5$ p.u, dok su tablično ti rezultati prikazani u tablici 10.2



Slika 10.3 Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 100-110-100 % pri radnoj snazi generatora $P_g \approx 50$ %

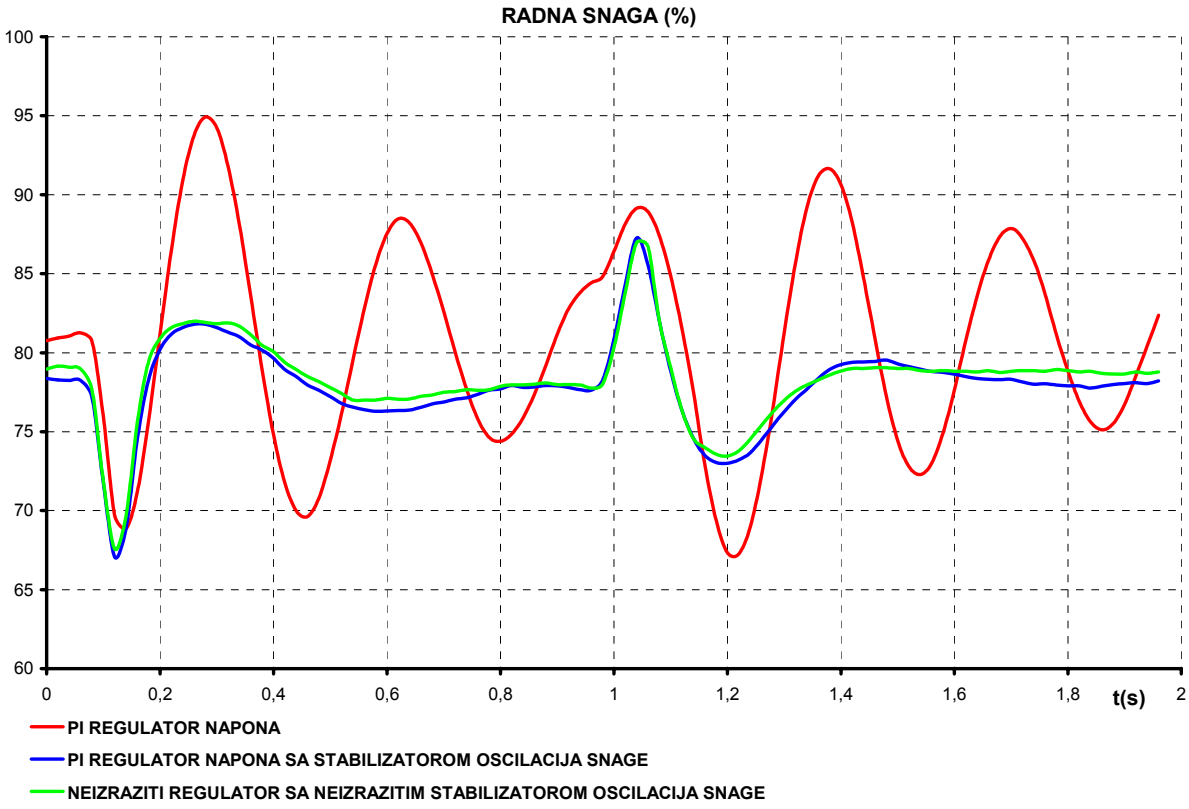


Slika 10.4. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.5$ p.u. (neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

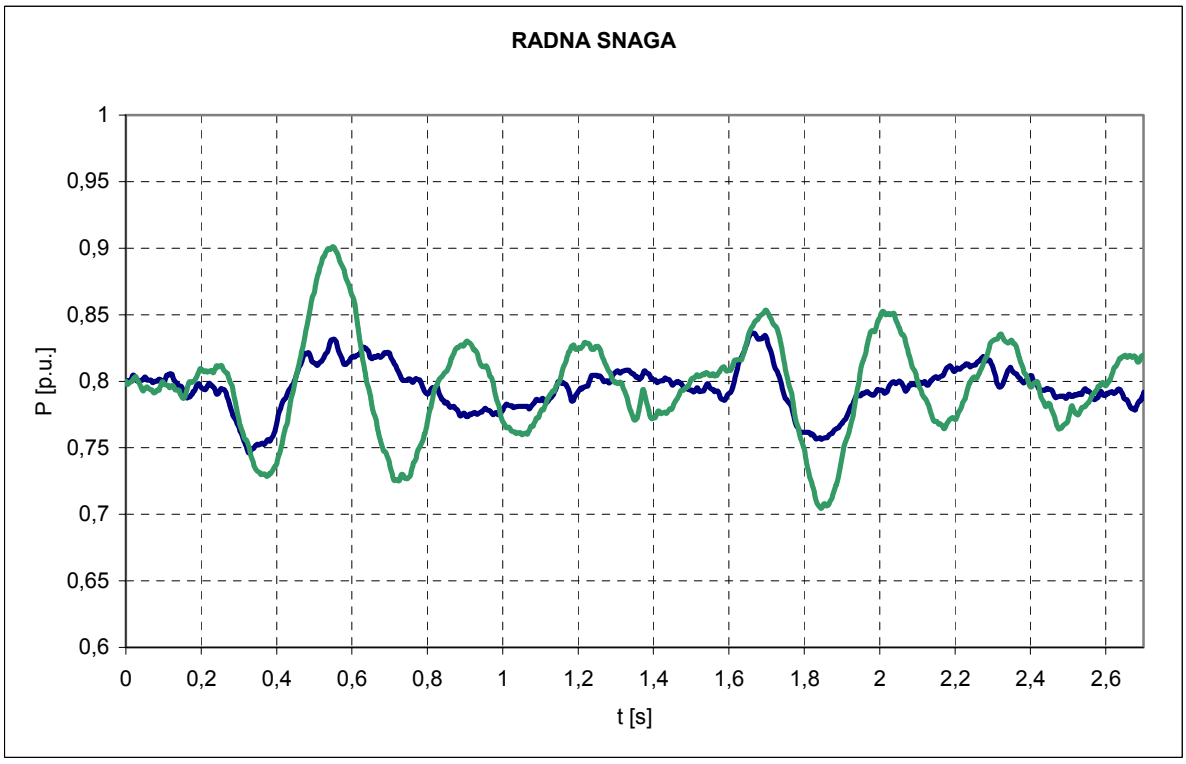
Promjena postavne veličine napona 100-110-100 % $P_g \approx 50$ %	Maksimalna vrijednost *MAX* (%)	Minimalna vrijednost *MIN* (%)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.58	0.40	0.264	0.196
PI regulator napona sa stabilizatorom	0.53	0.41	0.202	0.092
Neizraziti regulator napona s neizrazitim stabilizatorom	0.542	0.41	0.192	0.091
Neuronska mreža	0.57	0.45	0.08238	0.1496
Neuronska mreža s stabilizirajućim efektom	0.53	0.44	0.03893	0.06011

Tablica 10.2 Kriteriji kvalitete odziva za različite tipove regulatora za promjenu reference 1-1.1-1 p.u. napona i $P_g \approx 0.5$ p.u.

Na slici 10.5. i slici 10.6. prikazani su odzivi za slučaj skokovite promjene reference napona i to 1-0.9-1 p.u. uz $P_g \approx 0.8$ p.u, dok su tablično ti rezultati prikazani u tablici 10.3.



Slika 10.5. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 100-90-100 % pri radnoj snazi generatora $P_g \approx 80$ %

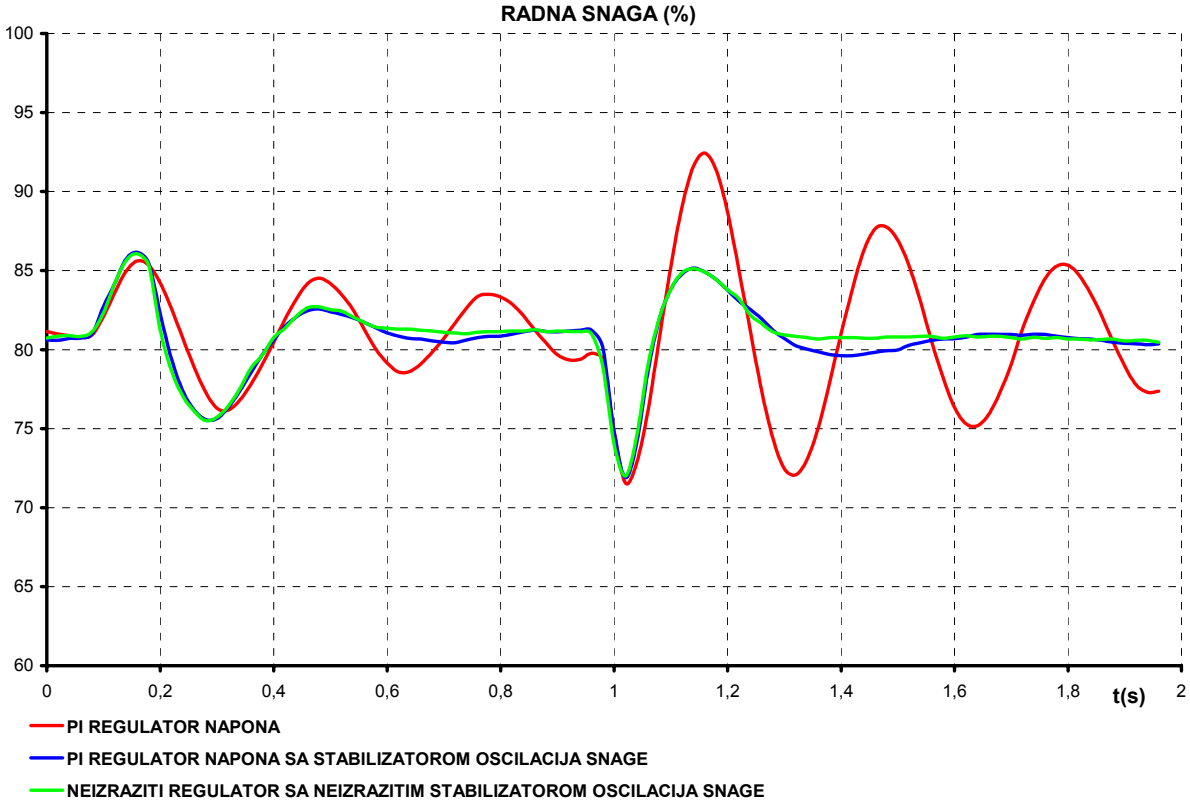


Slika 10.6. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-0.9-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u. (neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

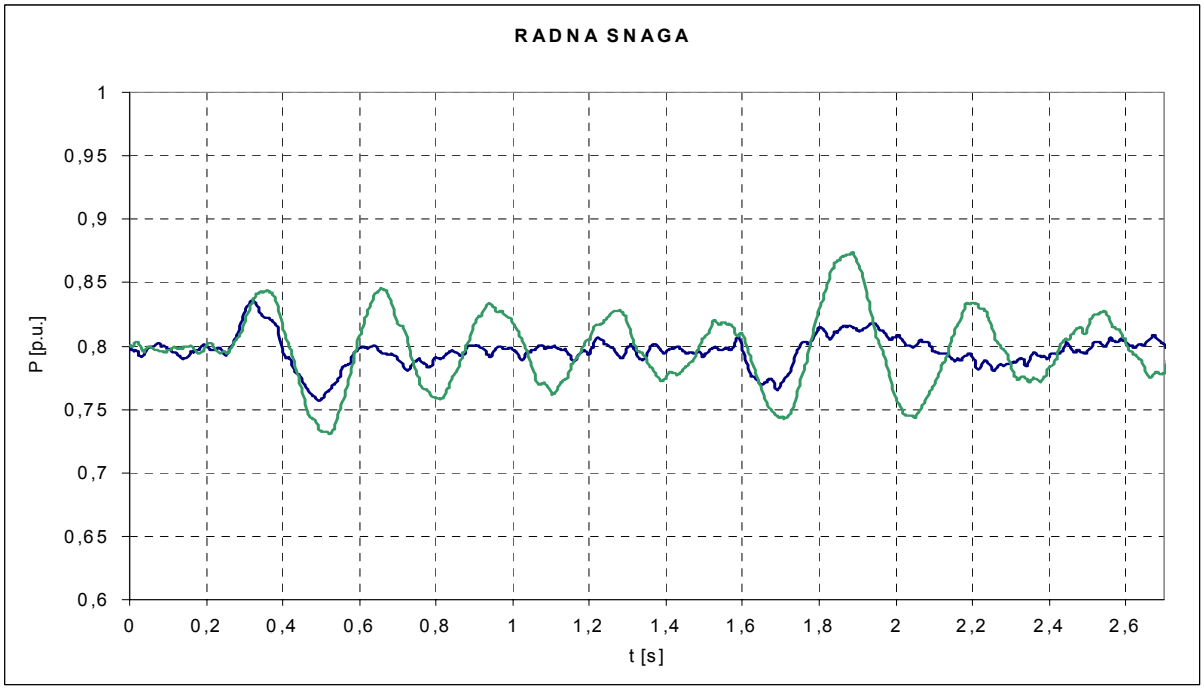
Promjena postavne veličine napona 100-90-100 % $P_g \approx 80$ %	Maksimalna vrijednost *MAX* (%)	Minimalna vrijednost *MIN* (%)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.95	0.67	0.52	0.373
PI regulator napona sa stabilizatorom	0.87	0.67	0.248	0.114
Neizraziti regulator napona s neizrazitim stabilizatorom	0.87	0.67	0.219	0.106
Neuronska mreža	0.9	0.71	0.1582	0.2563
Neuronska mreža s stabilizirajućim efektom	0.85	0.75	0.07143	0.1019

Tablica 10.3 Kriteriji kvalitete odziva za različite tipove regulatora za promjenu reference 1-0.9-1 p.u. napona i $P_g \approx 0.8$ p.u

Na slici 10.7. i slici 10.8**.** prikazani su odzivi za slučaj skokovite promjene reference napona i to 1-1.1-1 p.u. uz $P_g \approx 0.8$ p.u, dok su tablično ti rezultati prikazani u tablici 10.4.



Slika 10.7. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 100-110-100 % pri radnoj snazi generatora $P_g \approx 80$ %



Slika 10.8. Eksperimentalni odzivi radne snage generatora uz promjenu postavne veličine napona 1-1.1-1 p.u. pri radnoj snazi generatora $P_g \approx 0.8$ p.u. (neuronska mreža zeleno, neuronska mreža s stabilizirajućim efektom plavo)

Promjena postavne veličine napona 100-110-100 % $P_g \approx 80$ %	Maksimalna vrijednost *MAX* (%)	Minimalna vrijednost *MIN* (%)	Integral apsolutnog odstupanja *IAE*	Integral deriviranog apsolutnog odstupanja *IAED*
PI regulator napona	0.92	0.71	0.312	0.234
PI regulator napona sa stabilizatorom	0.86	0.72	0.130	0.097
Neizraziti regulator napona s neizrazitim stabilizatorom	0.86	0.72	0.119	0.090
Neuronska mreža	0.88	0.72	0.142	0.2527
Neuronska mreža sa stabilizirajućim efektom	0.83	0.75	0.04245	0.07226

Tablica 10.4 Kriteriji kvalitete odziva za različite tipove regulatora za promjenu reference 1-1.1-1 p.u. napona i $P_g \approx 0.8$ p.u

Iz gornjih slika je vidljivo da neuronska mreža s implementiranim stabilizirajućim efektom u kriterijskoj funkciji postiže nešto bolje stabilizirajuće djelovanje od neizrazitog regulatora s stabilizatorom oscilacija radne snage u prosjeku oko 2-3 % kriterijima MIN i MAX. Kriteriji IAE i IAED se ne mogu primijeniti za ocjenu kriterija kvalitete odziva jer su eksperimenti rađeni na dva različita sustava i u nešto drugačijim uvjetima (vrijeme trajanja prijelazne pojave nije isto) pa stoga postoje tolike razlike u iznosima. Naime trajanje pojave (odnosno trenutka u kojem nastupi promjena referentne vrijednosti napona i trenutka u kojem se vrati na prvobitni iznos) u literaturi [7] je oko 0.9 sekundi dok trajanje prijelazne pojave ovom radu iznosi 1.3 sekunde.

11 ZAKLJUČAK

Istraživanja u ovom radu odnose se na primjenu neuronske mreže u regulacijskoj strukturi sustava uzbude sinkronog generatora. Umjesto proporcionalno integralnog regulatora napona u sustav uzbude uključena je dinamička unaprijedna neuronska mreža sa stabilizirajućim efektom u modificiranoj kriterijskoj funkciji.

Dinamička neuronska mreža implementirana u ovom radu sastoji se od dva sloja sa po šest neurona u skrivenom sloju te jednim neuronom u izlaznom sloju. Učenje mreže odvija se *on-line* po algoritmu povratnog prostiranja.

Sustav regulacije uzbude je matematički modeliran i simuliran na računalu u *Matlab/Simulink* programu. Simulacijski model sustava omogućio je dobivanje početnih podešenja parametara regulatora napona zasnovanog na neuronskoj mreži, što je iskorišteno prilikom implementacije algoritama te je u konačnici i eksperimentalno provjereno.

Za eksperimentalnu provjeru djelovanja sustava regulacije napona napravljen je sustav zasnovan na procesoru za digitalnu obradu signala. Djelovanja navedenih algoritama su ispitana pri radu sinkronog generatora koji je preko prijenosna voda i transformatora spojen na mrežu. Eksperimentima na laboratorijskom modelu i simulacijom na računalu ispitano je djelovanje sustava u uvjetima skokovite promjene postavne veličine napona iznosa ±0.1 p.u. od nazivne vrijednosti napona.

Za ocjenjivanje djelovanja proporcionalno integralnog regulatora napona i neuronske mreže korišteni su numerički kriteriji kvalitete odziva radne snage generatora i to : integral apsolutnog odstupanja radne snage od stacionarne vrijednosti (*IAE*) te integral apsolutnog deriviranog odstupanja (*IAED*).

Eksperimentalni i simulacijski rezultati pokazuju da u trenutku promjene postavne veličine napona generatora dolazi do oscilacija radne snage koje iznose i do ±0.2 p.u. uz djelovanje regulator napona s PI karakteristikom. IAE i IAED kriteriji za taj slučaj pokazuju značajna odstupanja uz pojavu oscilacija. Kada je neuronska mreža u sustavu upravljanja (bez stabilizirajućeg efekta) oscilacije radne snage po amplitudi iznose 0.1 p.u ., a IAE i IAED kriteriji su približno dva puta manji. Kad je uključen stabilizacijski efekt u kriterijsku funkciju oscilacije snage su do 0.05 p.u. radne snage a IAE i IAED kriterij su približno četiri puta manji u odnosu na djelovanje PI regulatora te oko dva puta manji u odnosu na djelovanje neuronske mreže bez stabilizirajućeg efekta.

Simulacijski i eksperimentalni rezultati pokazuju prihvatljivost implementacije dinamičke neuronske mreže u sustav radi smanjivanja oscilacija radne snage. Kompleksnost implementacije neuronske mreže opravdana je jednostavnošću podešavanja parametara modificirane kriterijske funkcije i kvalitetom odziva koju daje ovakva struktura mreže.

Nadalje, usporedbom djelovanja regulatora napona zasnovanog na neuronskoj mreži s neizrazitim regulatorom napona sa stabilizatorom oscilacija radne snage implementiranog u dugi sustav pri čemu se je koristio isti sinkroni generator na kojem su se vršili pokusi, pokazuje da neuronska mreža s stabilizirajućim efektom u kriterijskoj funkciji pokazuje 2-3 % bolje prigušenje oscilacija od neizrazitog regulatora. Rezultati kriterija dobiveni eksperimentalnim putem nešto su veći nego što bi stvarno trebali biti zbog postojanja visokofrekventnog šuma koji je posljedica ove verzije procesora (sva snimanja su napravljena s preliminarnom verzijom procesora koja u potpunosti ne zadovoljava tvornički deklarirane osobine procesora).

Eksperimentalni i simulacijski rezultati pokazuju opravdanost primjene dinamičke neuronske mreže u sustavu za regulaciju napona sinkronog generatora. Osim toga eksperimentalni rezultati pokazuju opravdanost izgradnje sustava regulacije napona sinkronog generatora na platformi procesora za digitalnu obradu signala TMS320F2812 zbog komfornih razvojnih alata koji su industrijski standard te su popraćeni s mnogim primjerima i primjerenom podrškom od strane proizvođača.

.

12 POPIS OZNAKA

φ	kut između napona i struje generatora
ϑ	kut opterećenja generatora
$\Im$	kriterijska funkcija
ψ	aktivacijska funkcija neurona
ψ_d, ψ_d	ulančani tok armaturnog namota sinkronog generatora u *d* i *q* osi
ψ_D , ψ_Q	ulančani tok prigušnog namota sinkronog generatora u *d* i *q* osi
ω	kutna brzina (električna)
ω_{meh}	kutna brzina (mehanička)
E_0	inducirani napon generatora
I	amplituda struje generatora
i_a, i_b, i_c	fazna struje armature generatora
i_{ab}, i_{bc}, i_{ca}	linijske struje armature generatora
i_d, i_q	komponente struje statora u koordinatnom sustav rotora generatora
i_u	struja uzbude generatora
i_α, i_β	komponente struje statora u koordinatnom sustav statora generatora
m_{elm}	elektromagnetski moment generatora
m_{meh}	mehanički moment na osovini generatora
P, P_{el}	radna snaga generatora
P_{max}	prekretna snaga generatora
Q	jalova snaga generatora
R	fazni otpor statora generatora
R_D, R_Q	otpor prigušnog napona u *d* i *q* osi generatora
R_u	otpor prigušnog napona u *d* i *q* osi generatora
r_v	spojnih vodova do mreže i blok transformatora generatora
S	prividna snaga generatora
T_d	vremenska konstanta derivacijskog člana regulatora kuta opterećenja
T_d''	vremenska konstanta za subtranzijentno stanje u *d* osi
T_q''	vremenska konstanta za subtranzijentno stanje u *q* osi
T_m	mehanička vremenska konstanta agregata
T_u	vremenska konstanta uzbude
U	amplituda napona generatora

U_{PSS}	izlaz iz stabilizatora elektroenergetskog sustava
U_{komp}	kompenzirana vrijednost amplitude napona generatora
U_{ref}	referentna vrijednost napona generatora
u_a, u_b, u_c	fazni naponi armature generatora
u_{ab}, u_{bc}, u_{ca}	fazni naponi armature generatora
u_d, u_q	komponente napona statora u koordinatnom sustav rotora generatora
u_{km}	napon krute mreže
u_{kmd}, u_{kmq}	napon krute mreže u *d* i *q* osi
u_u	napon uzbude generatora
u_α, u_β	komponente napona statora u koordinatnom sustav statora generatora
w_{ijk}	težinski koeficijent neurona
x	ulaz u neuron
x_d, x_q	sinkrona reaktancija u *d* i *q* osi generatora
x_D, x_Q	reaktancija prigušnog namota u *d* i *q* osi generatora
x_{dD}	reaktancija armature prigušnog namota u *d* osi generatora
x_d'	tranzijentna uzdužna reaktancija generatora
x_d''	subtranzijentna uzdužna reaktancija generatora
x_q''	subtranzijentna poprečna reaktancija generatora
x_l	rasipna reaktancija generatora
x_u	reaktancija uzbudnog namota generatora
x_{ud}	reaktancija uzbude armature u *d* osi generatora
x_{uD}	reaktancija uzbude prigušnog namota u *d* osi generatora
x_{qQ}	reaktancija armature prigušnog namota u *q* osi generatora
x_v	reaktancija spojnih vodova do mreže i blok transformatora generatora
y_{ki}	izlaz iz neurona nakon aktivacijske funkcije
v_{ki}	izlaz iz neurona prije aktivacijske funkcije

13 LITERATURA

[1] Petrović I., *"Inteligentno upravljanje sustavima 2. dio: Neuronsko upravljanje"*, FER Zagreb, 2003/2004.

[2] Symon Haykin , "*Neural Networks: A Comprehensive Foundation",* IEEE Press 1994

[3] O.P.Malik,.M.M.Salem, A.M. Zaki, O.A. Mahgoub and E. Abu El-Zahab, *"Experimental studies with simple neuro-controller based excitation controller",* IEEE Proc.-Gener. Transm. Distrib. Vol. 149. No I. January 2002

[4] M.M.Selem, A.M. Zaki, O.P. Mahgoub, E. Abu El-Zahab, O.P. Malik, *"Experimental Veification of Generating Unit Excitation Neuro-Controller"* Proc. IEEE Power Engineering Society, Winter Meeting 2000, Singapure

[5] M.M.Selem, A.M. Zaki, O.P. Mahgoub, E. Abu El-Zahab, O.P. Malik, *"Studies on Multi-Machine Power System With a Neural Network Based Excitation Controller"* Proc. IEEE Power Engineering Society, 2000

[6] Idžotić T., *Proširenje područja stabilnog rada sinkronog generatora regulacijom uzbude,* Doktorska disertacija, Zagreb, 2003.

[7] Sumina D., *"Neizrazito upravljanje sustavom uzbude"*, Magistarski rad, Zagreb 2005

[8] Tečec Z., *"Primjena neuronske mreže u sustavu za regulaciju napona sinkronog generatora*", Diplomski rad, Zagreb 2004

[9] Sirotić Z., Maljković Z.,*"Sinkroni strojev",* CIP, Zagreb, 1996.

[10] Glavan B. , *"ZES490 Matematički model i simulacija samouzbudnog sustava za regulaciju sinkronog generatora"*, Elaborat, Zagreb, 1993.

[11] Ferega D., *"Digitalna regulacija napona sinkronog generatora s tranzistorskim pretvaračem u uzbudnom krugu",* Magistarski rad, Zagreb, 2000.

[12] G. Erceg, *"Regulacija napona beskontaktnog samouzbudnog sinkronog generatora u autonomnom radu"*, Doktorska disertacija, Zagreb, 1996.

[13] Texas Instruments:"*TMS320F2810, TMS320F2811, TMS320F2812,TMS320C2810, TMS320C2811,TMS320C2812,Digital Signal Processors,Data Manual",*Literature Number: SPRS174L April 2001 − Revised December 2004

[14] www.ti.com

[15] Texas Instruments: "*TMS320C28x DSP CPU and Instruction Set Reference Guide*", Literature Number: SPRU430D,August 2001 − Revised March 2004

[16] Texas Instruments:"*TMS320x281x DSP Analog-to-Digital Converter(ADC) Reference Guide*",Literature Number: SPRU060D,June 2002 − Revised July 2005

[17] Texas Instruments:"*TMS320x281x DSP System Control and Interrupts Reference Guide*",Literature Number: SPRU078C,April 2002 − Revised March 2005

[18] Texas Instruments:"*TMS320x281x DSP Event Manager (EV) Reference Guide*",Literature Number: SPRU065C, November 2004

[19] www.spectrumdigital.com

[20] Spectrum Digital:"*eZdspTM F2812 Technical Reference*", 506265-0001 Rev. B, July 2002

[21] Texas Instrumants: "*Code Composer Studio White Paper*",SPRA520 ,Digital Signal Processing Solutions May 1999

[22] Zhenyu Yu:" *3.3V DSP for Digital Motor Control*", Application Report, SPRA550, Digital Signal Processing Solutions June 1999,C2000 Applications

[23] Spectrum Digital:"*DMC1500 Technical Reference*",504915-0001 Rev. A, September 2000

[24] Semikron: "*SKHI10/12*",22-08-2003, Driver electronic-PCB drivers

[25] Damir Sumina:"*PRIMJENA SUSTAVA ASTAT ZA ISTOSMJERNE PRETVARAČE*",DIPLOMSKI RAD br. 1603,Zagreb, rujan 2001.

[26] David M. Alter, Ph.D.:"*Thermoelectric Cooler Control Using a TMS320F2812 DSP and a DRV592 Power Amplifier*", DSP Applications - Semiconductor Group, SPRA873 - February 2003

[27] Texas Instruments:" *FIR Filter Design package user's Guide*", Filter Library, C28x Foundation Software, Texas Instruments Inc., May 2002

[28] Končar, "*Tehnički priručnik peto izdanje*", Zagreb 1991

[29] Bulic N., Erceg G., Idzotic T, *Comparison of different methods of excitation control for a synchronous generator*, EPE-PEMC, Riga, 2004.

DODATAK A

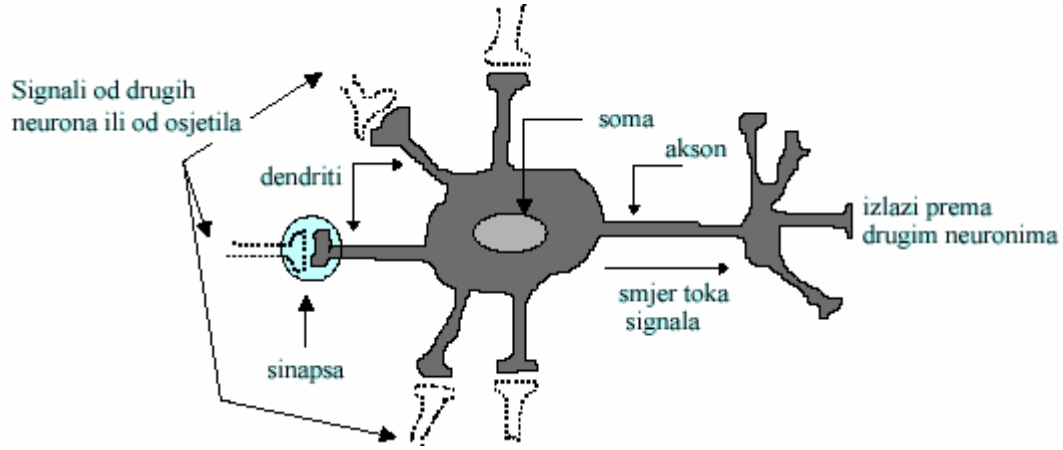
Biološka osnova razvoja neuronskih mreža

Istraživanja i razvoj umjetnih neuronskih mreža motivirana su spoznajama o građi i načinu funkcioniranja ljudskog mozga te njegovim velikim sposobnostima u rješavanju složenih problema. Ljudski je mozak previše složen da bi se mogao potpuno opisati i razumjeti način njegova djelovanja, no zna se da se sastoji od oko 10^{11} osnovnih živčanih stanica (neurona) (Slika A.1.) organiziranih u module (slojeve) koji su međusobno povezani u složenu mrežu s oko 10^{15} međusobnih veza. Za mozak se može reći da je kompleksno, nelinearno i paralelno računalo. Osim toga ljudski mozak je energetski gledano efikasan uređaj te mu je za jednu operaciju potrebno približno 10^{-16} (J/ operacija u sekundi), dok je današnjim računalima potrebno oko 10^{-6} (J/ po operacija u sekundi).

Slika A.1 Osnovni neuron

Biološki neuron, kao osnovna gradivna jedinica ove biološke neuronske mreže, prima i obrađuje informacije od drugih neurona i/ili osjetilnih organa. Sa stajališta obrade signala biološki se neuron može pojednostavljeno prikazati kao stanica sastavljena od tijela (SOMA), DENDRITA

mnoštva dendrita i aksona (Slika A.2). Akson se može zamisliti kao tanka cjevčica čiji je jedan kraj povezan na tijelo neurona, a drugi je razdijeljen na mnoštvo grana. Krajevi ovih grana završavaju malim zadebljanjem koja najčešće dodiruju dendrite, a rjeđe tijelo drugog neurona. Mali razmak između završetka aksona prethodnog neurona i dendrita sljedećeg neurona naziva se sinapsa. Impulsi koji se generiraju u tijelu neurona, putuju kroz akson do sinapsi. Ovisno o učinkovitosti svakog pojedinačnog sinaptičkog prijenosa, signali različitih intenziteta dolaze do dendrita. Učinkovitost sinaptičkog prijenosa kroz neku sinapsu ovisi o njezinom elektrokemijskom stanju, koje je rezultat prethodnih sinaptičkih prijenosa kroz nju. Sinapse, dakle, predstavljaju memorijske članove biološke neuronske mreže. Signali se od sinapsi dendrita prosljeđuju do tijela neurona, gdje se prikupljaju i obrađuju. Ovi signali za tijelo neurona mogu biti pobuđujući ili smirujući. Matematički gledano, pobuđujući i smirujući signali imaju suprotan predznak. Ako je njihova kumulativna vrijednost tijekom kratkog vremenskog intervala veća od praga osjetljivosti neurona, tijelo neurona generira impulse koji se šalju duž aksona prema drugim neuronima, a ako je manja neuron ostaje nepobuđen i ne generira impuls [1].



Slika A.2 Shematski prikaz biološkog neurona

Unatoč velikom broju vrsta, sve neuronske mreže imaju neka zajednička svojstva koja posjeduje biološki sustav, a koja se ne mogu naći kod konvencionalnih računskih tehnika. To je prije svega paralelno raspodijeljena obrada informacija, što kod neuronskih mreža povećava otpornost na kvar u odnosu na klasične računske tehnike. Svojstvo učenja i adaptacije čini neuronske mreže sposobnima prilagođavati svoje ponašanje novonastalim prilikama. Svojstvo aproksimacije proizvoljne nelinearne funkcije do željene točnosti najvažnije je njihovo svojstvo sa stajališta modeliranja, identifikacije i upravljanja nelinearnim procesima. Sva ova svojstva poboljšavaju se razvojem specijaliziranog sklopovlja za implementaciju neuronskih mreža.

DODATAK B

BAZNE VELIČINE……..…….. B1

PARAMETRI GENERATORA KORIŠTENI U SIMULACIJI…..……..……. B2

MODELI U PROGRAMSKOM PAKETU *Matlab Simulink*

proračun momenata i kuta opterećenja u modelu sinkronog generatora…..... B3

proračun radne, jalove snage, napona i struja generatora u model sinkronog generatora……. B4

proračun d i q komponenata napona u modelu sinkronog generatora…….…. B5

proračun d i q komponenata struje u modelu sinkronog generatora…….…… B6

proračun tokova u modelu sinkronog generatora…… B7

proračun parametara 1 u modelu sinkronog generatora…… B8

proračun parametara 2 u modelu sinkronog generatora……... B9

proračun parametara 3 u modelu sinkronog generatora……..…… B10

proračun parametara 4 u modelu sinkronog generatora……..… B11

C KÔD S-FUNKCIJE NEURONSKE MREŽE..... B12

B1 BAZNE VELIČINE

Napon armature $U_B = \sqrt{2} \cdot U_{fn}$

Struja armature $I_B = \sqrt{2} \cdot I_{fn}$

Impedancija u armaturnom krugu $z_B = \frac{U_B}{I_B}$

Snaga $S_B = \frac{3}{2} \cdot U_B \cdot I_B$

Električna kutna brzina $\omega_B = \omega_s = 2 \cdot \pi \cdot f$

Ulančeni tok $\psi_B = \frac{U_B}{\omega_B}$

Moment $M_B = p \cdot \frac{S_B}{\omega_B}$

Induktivitet u armaturnom krugu $L_B = \frac{z_B}{\omega_B}$

Kapacitet u armaturnom krugu $C_B = \frac{1}{z_B \cdot \omega_B}$

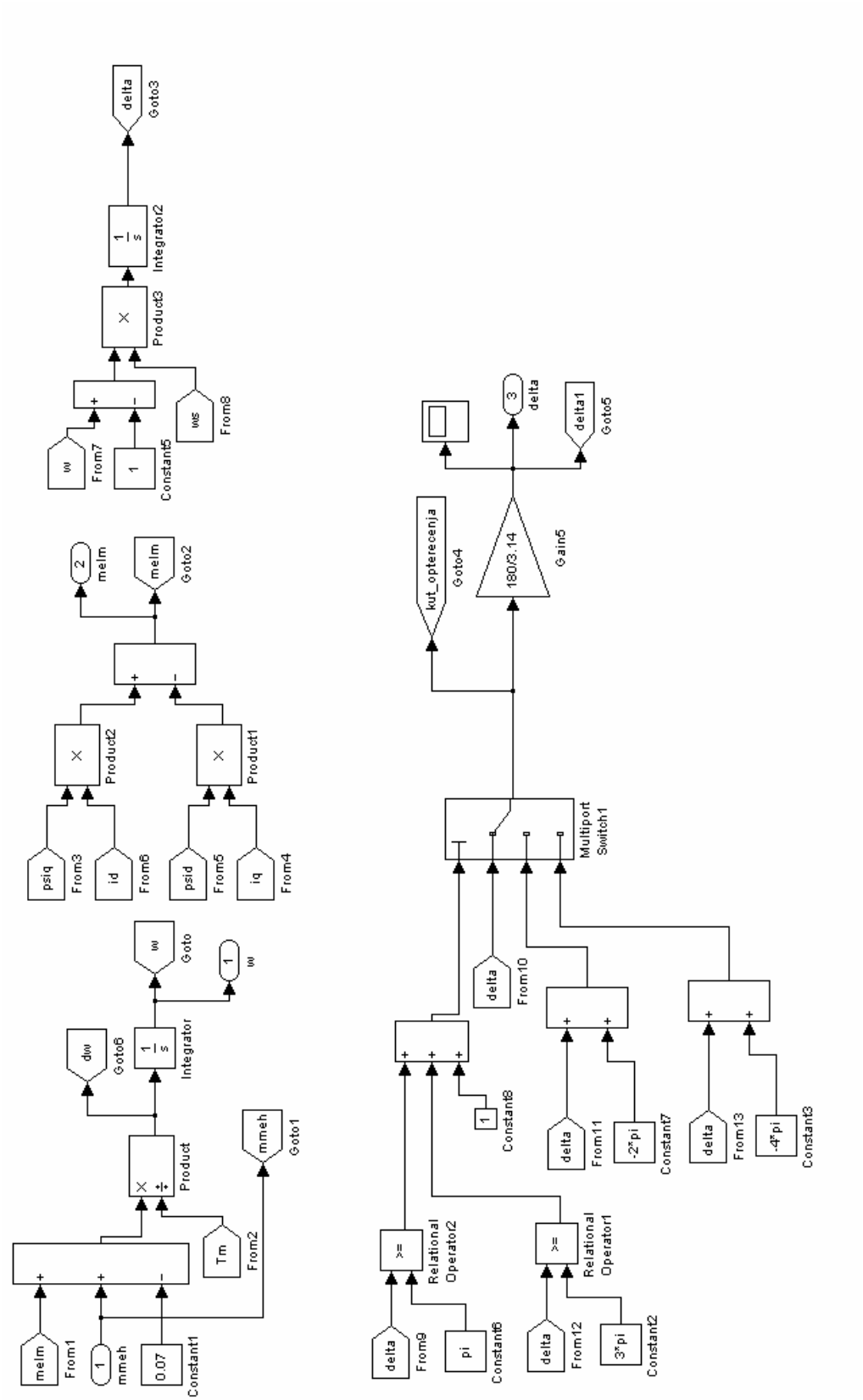
Struja uzbude $I_{uB} = I_{u0} \cdot x_{ud}$

Napon uzbude $U_{uB} = U_B \cdot \frac{3}{2} \cdot \frac{I_B}{I_{uB}}$

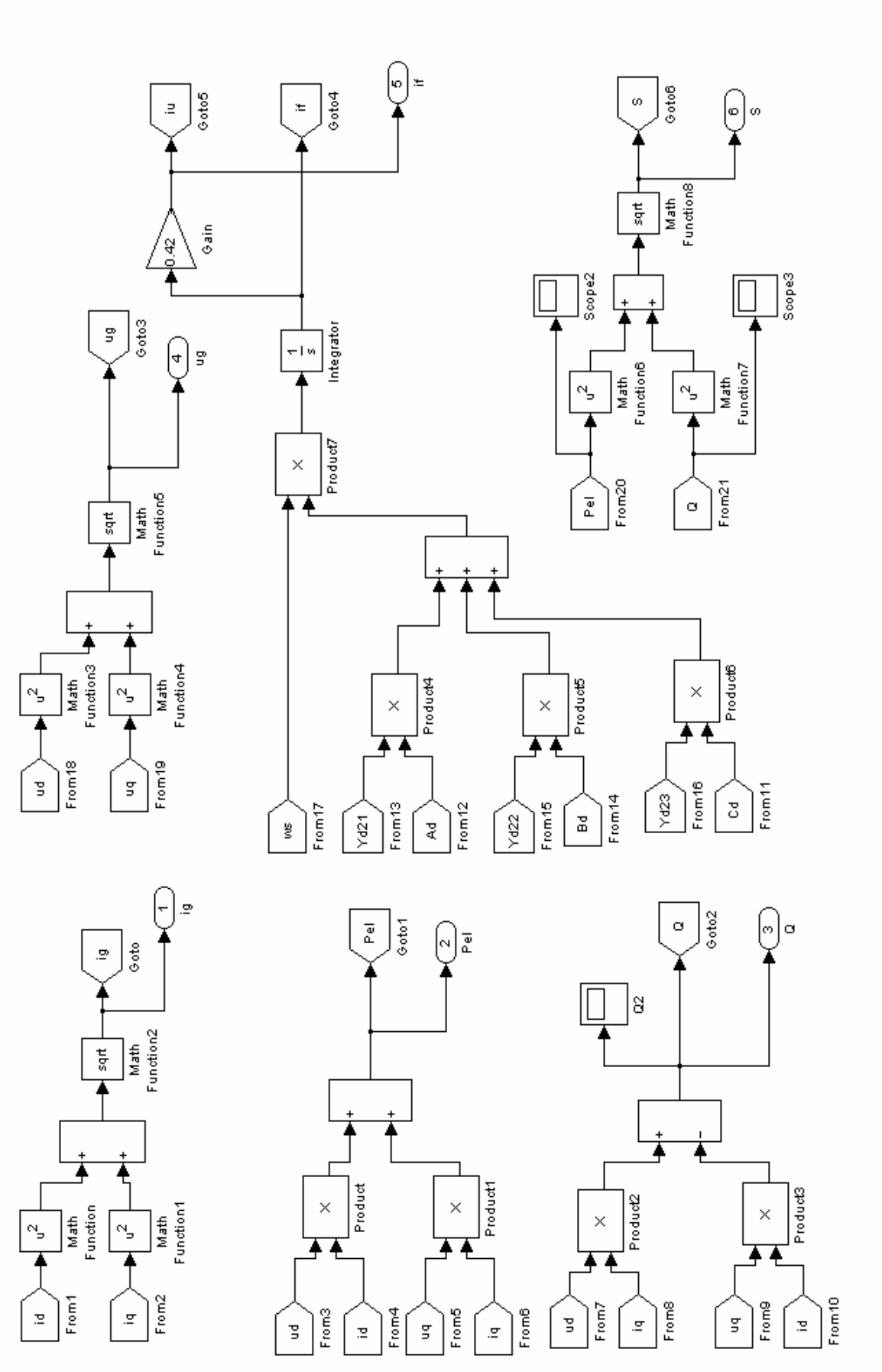
Impedancija uzbude $z_{uB} = z_B \cdot \frac{3}{2} \cdot \left(\frac{I_B}{I_{uB}} \right)^2$

B2 PARAMETRI GENERATORA KORIŠTENI U SIMULACIJI

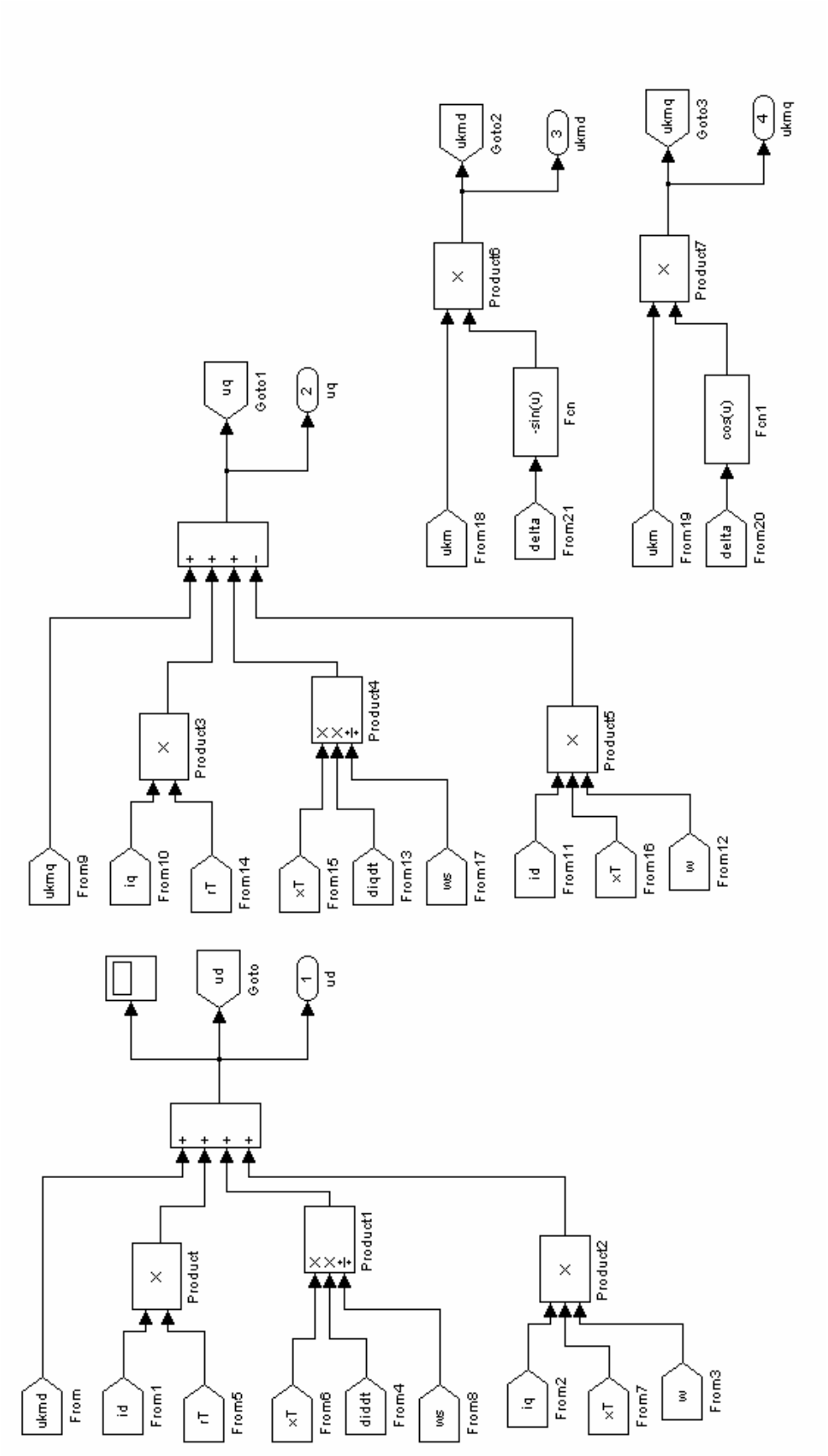
X_d	0,8 (p.u.)
X_q	0,51 (p.u.)
X_l	0,04 (p.u.)
R	0,04 (p.u.)
T_m	2,6 (s)
T_f	0,55 (s)
X_d'	0,35 (p.u.)
X_d''	0,15 (p.u.)
X_q''	0,15 (p.u.)
T_d''	0,054 (s)
T_q''	0,054 (s)
r_v	0,05 (p.u.)
x_v	0,35 (p.u.)



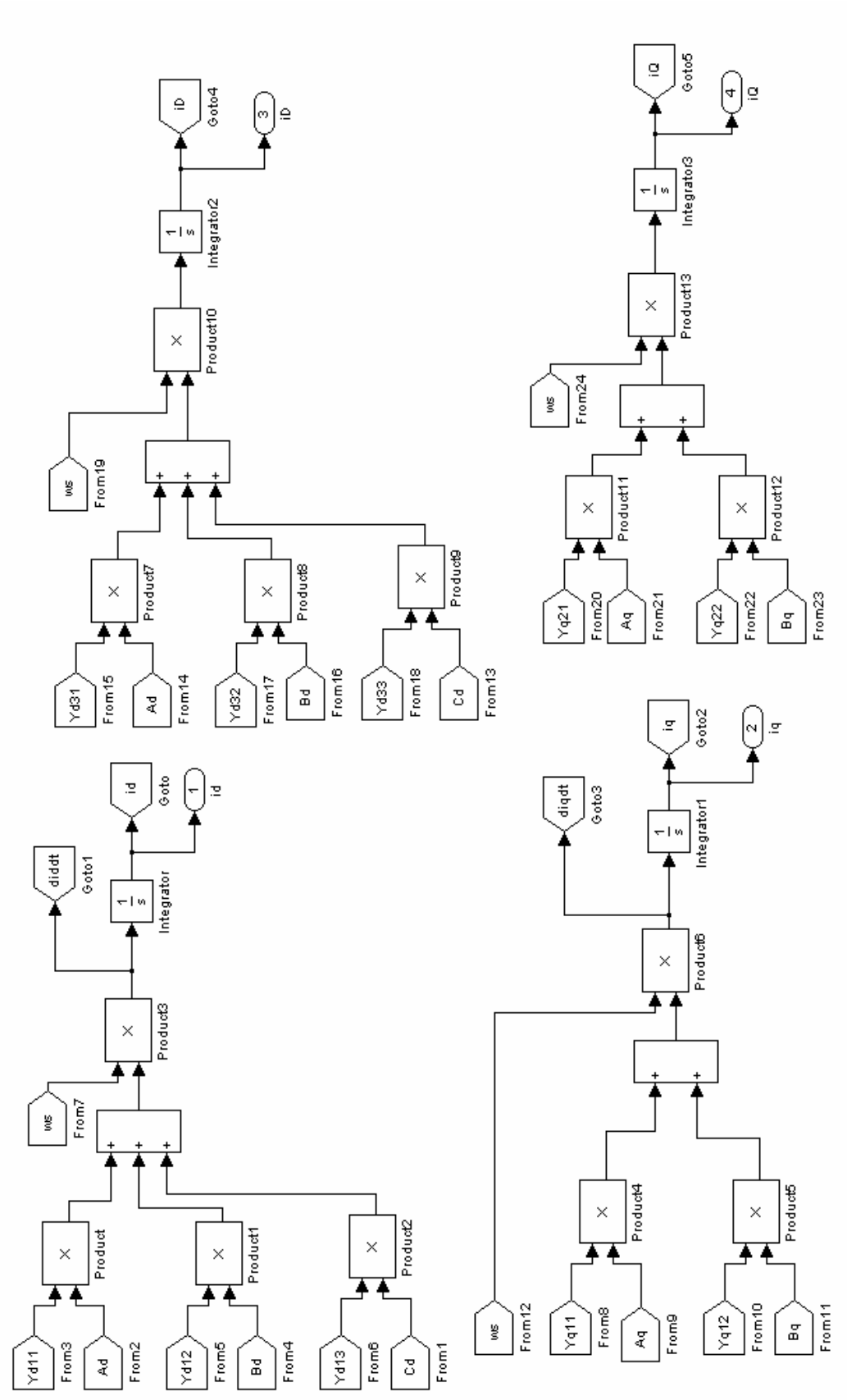
B3 Proračun momenata i kuta opterećenja u modelu sinkronog generatora



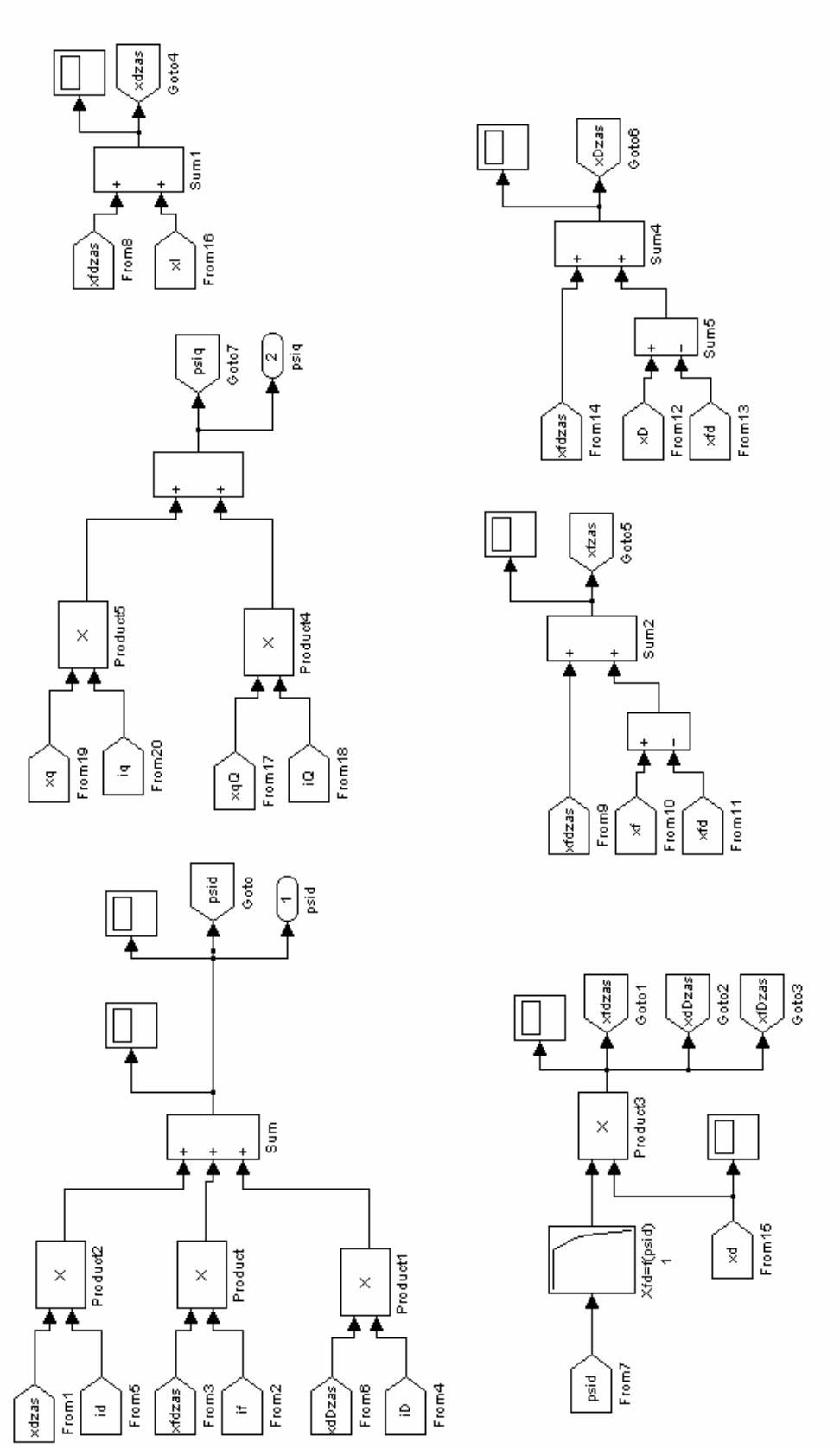
B4 Proračun radne, jalove snage, napona i struja generatora u modelu sinkronog generatora



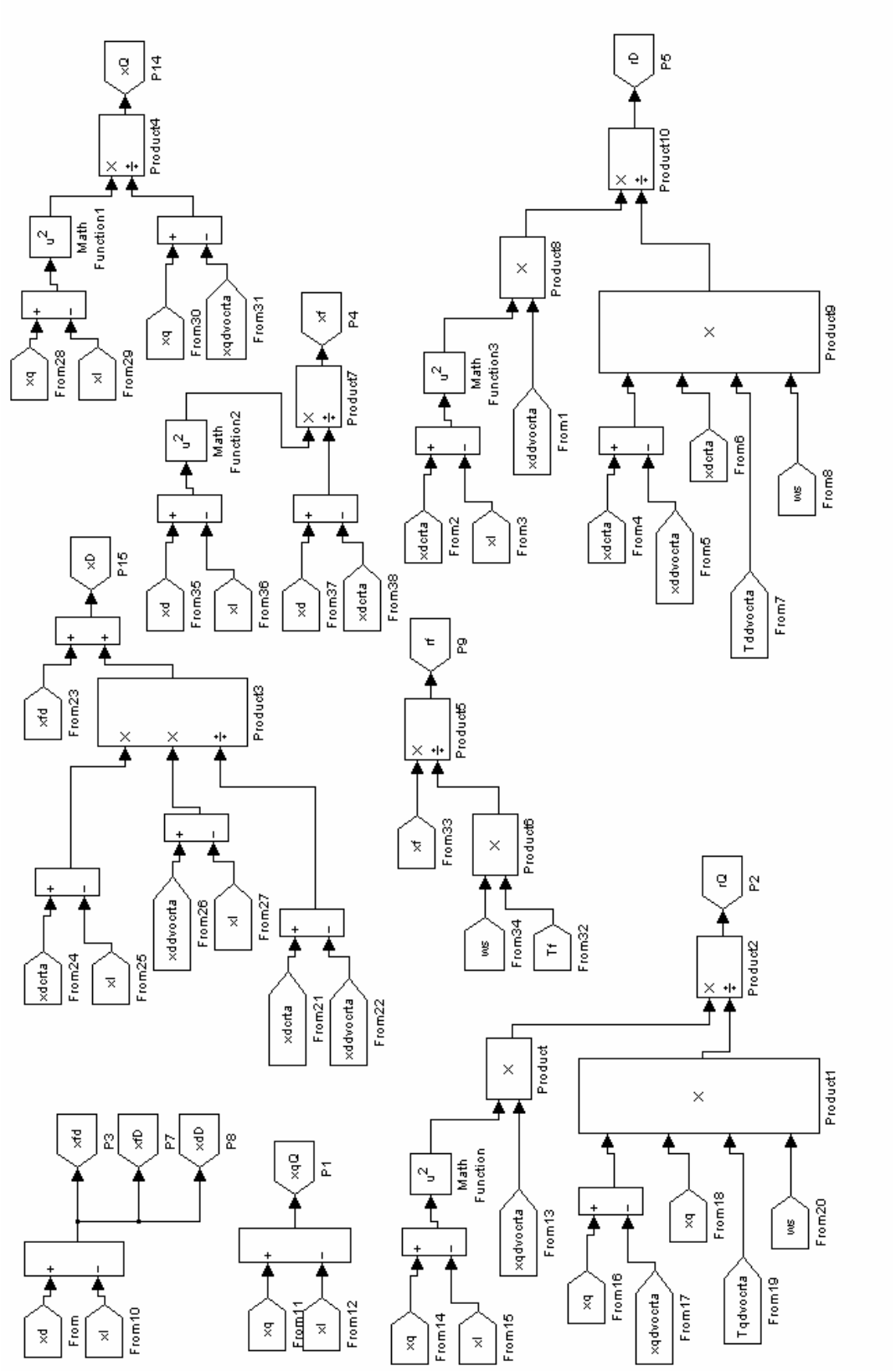
B5 Proračun d i q komponenata napona u modelu sinkronog generatora



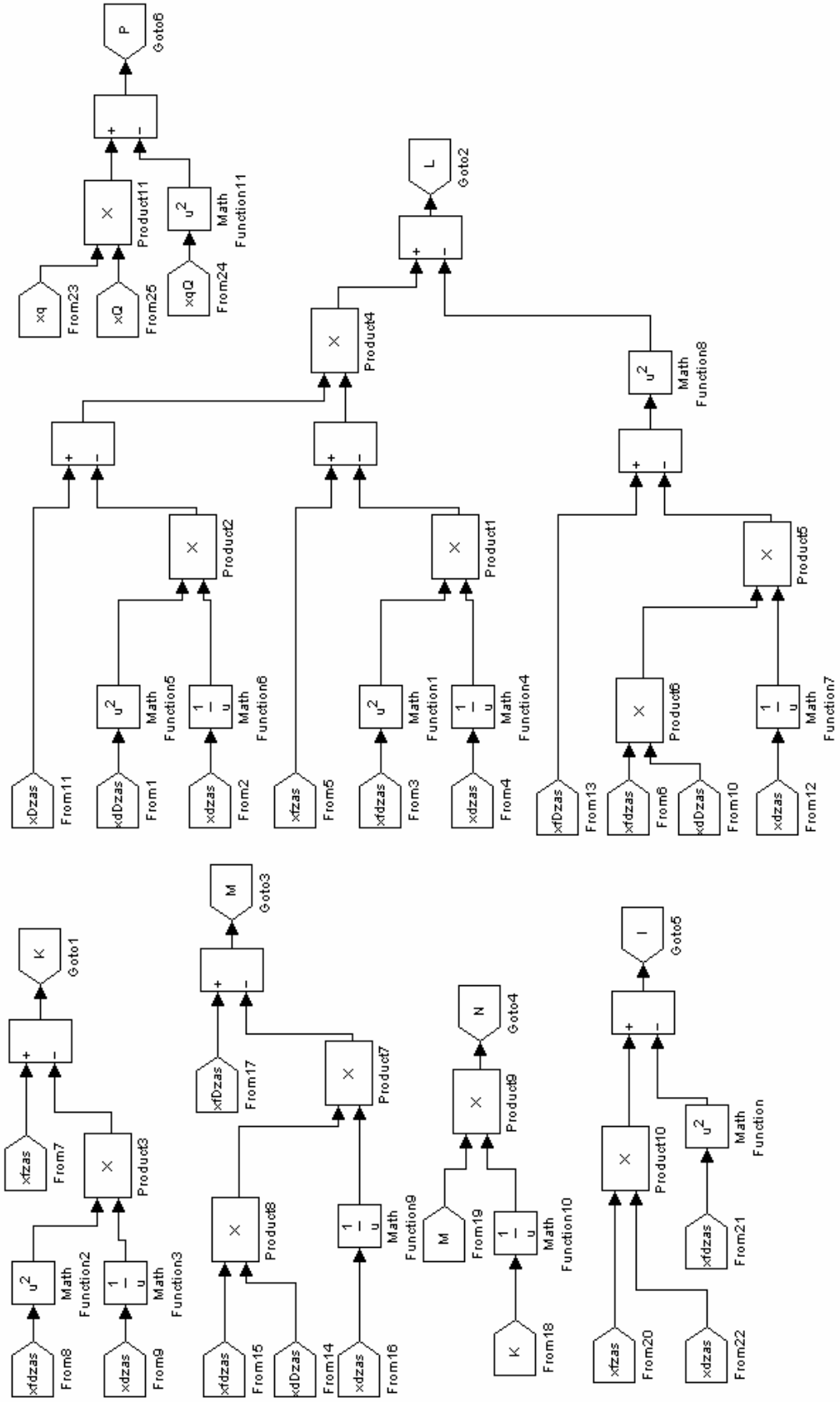
B6 Proračun d i q komponenata struje u modelu sinkronog generatora



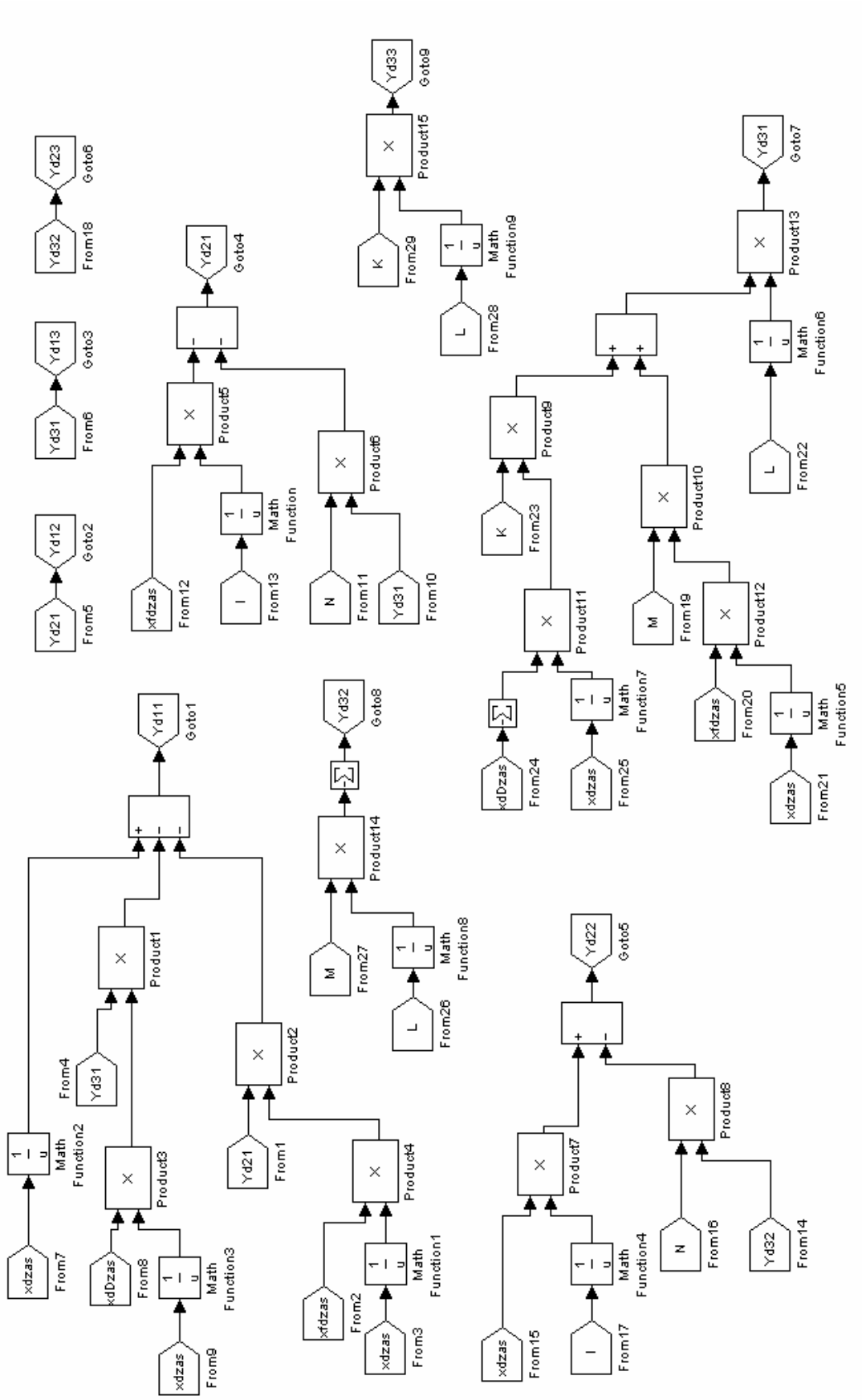
B7 Proračun tokova u modelu sinkronog generatora



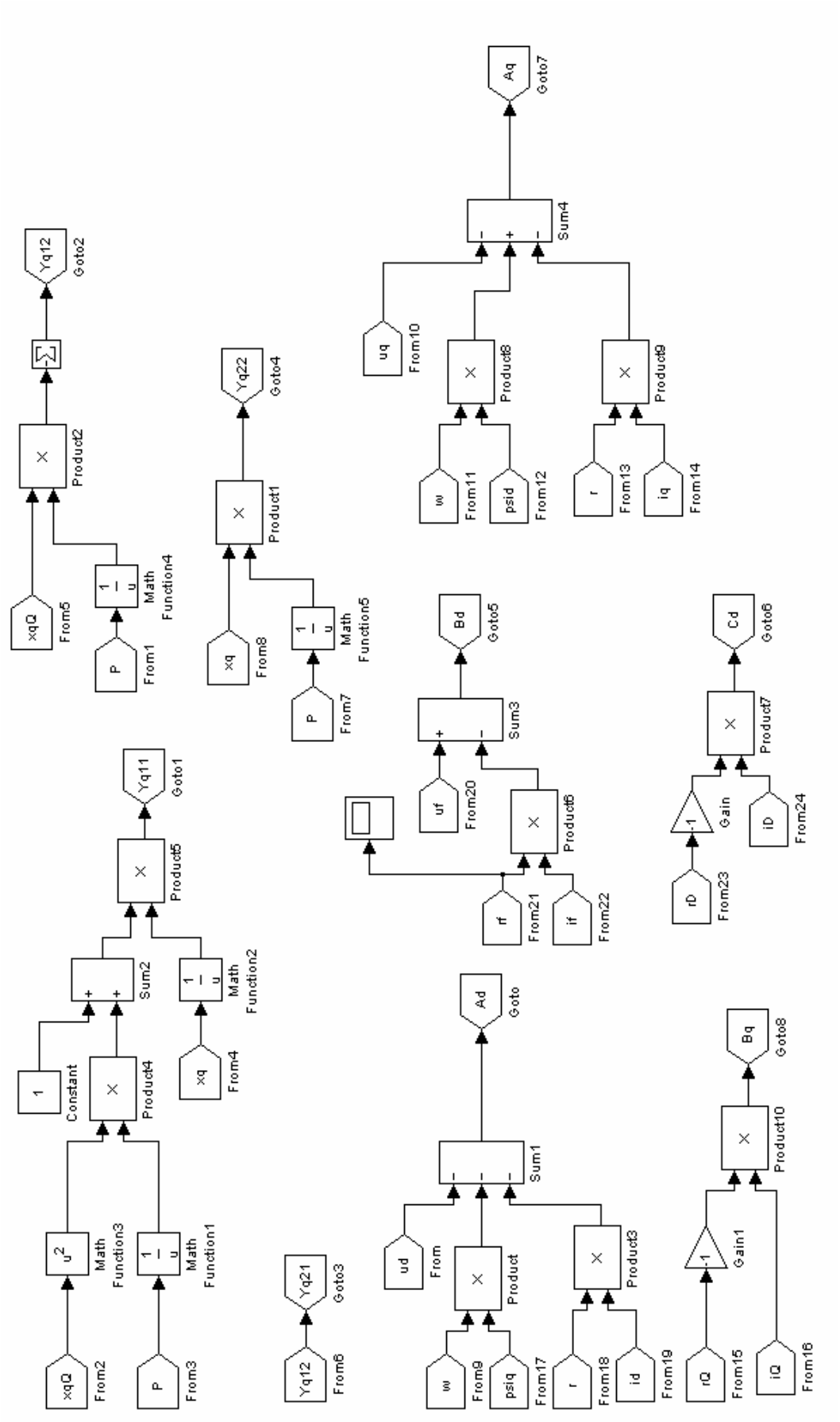
B8 Proračun parametara 1 u modelu sinkronog generatora



B9 Proračun parametara 2 u modelu sinkronog generatora



B10 Proračun parametara 3 u modelu sinkronog generatora



B11 Proračun parametara 4 u modelu sinkronog generatora

B12 C KÔD S-FUNKCIJE NEURONSKE MREŽE

```
                                        /*
 * sfuntmpl_basic.c: Basic 'C' template for a level 2 S-function.
 *  -------------------------------------------------------------------------
 *  | See matlabroot/simulink/src/sfuntmpl_doc.c for a more detailed template |
 *  -------------------------------------------------------------------------
 * Copyright 1990-2002 The MathWorks, Inc.
 * $Revision: 1.27 $
 */
#define S_FUNCTION_NAME  mreza1_real
#define S_FUNCTION_LEVEL 2
#define       TANSIG(x)                 ( 2.0 / (1 + exp(-2.0*x)) - 1.0 )    /*
Aktivacijske funkcije*/
#define     LOGSIG(x)            ( 1.0 / (1 + exp(-x))           )
#define     PURELIN(x)           ( x                             )
#define     TANSIGder(x)         ( (1.0 - (TANSIG(x)*TANSIG(x))) )
#include "simstruc.h"
#include<stdlib.h>
#include<math.h>
#include<stdio.h>
#include<time.h>
static void mdlInitializeSizes(SimStruct *S)
{
    /* See sfuntmpl_doc.c for more details on the macros below */
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return; }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);
    if (!ssSetNumInputPorts(S, 4)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortWidth(S, 1, 1);
       ssSetInputPortWidth(S, 2, 1);
       ssSetInputPortWidth(S, 3, 1);
    ssSetInputPortRequiredContiguous(S, 0, true); /*direct input signal access*/
    ssSetInputPortRequiredContiguous(S, 1, true);
       ssSetInputPortRequiredContiguous(S, 2, true);
       ssSetInputPortRequiredContiguous(S, 3, true);
        ssSetInputPortDirectFeedThrough(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);
       ssSetInputPortDirectFeedThrough(S, 2, 1);
       ssSetInputPortDirectFeedThrough(S, 3, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 40);    /* Broj radnih varijabli sa kojima radi tijekom ucenja
*/
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    ssSetOptions(S, 0);
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, 0.01);  /* Vrijeme uzorkovanja */
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  static void mdlStart(SimStruct *S)
  {real_T *y = ssGetOutputPortSignal(S,0);
   /* postavljanje pocetnih vrijednosti težina, biasa i izlaza iz neurona */
    ssSetRWorkValue(S,0,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,1,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,2,((double)((rand()%(int)(201))-100)/200));
```

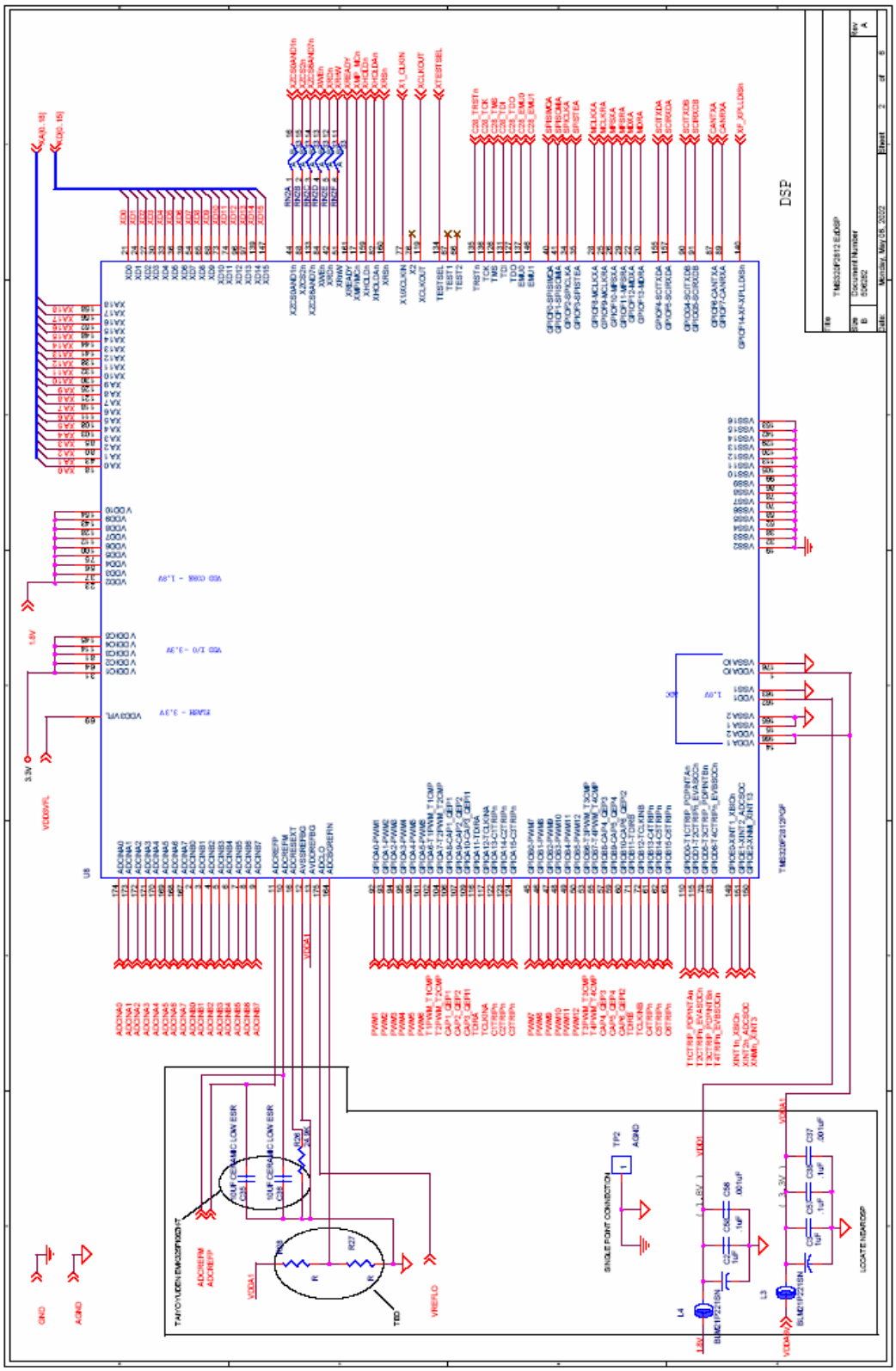
```
    ssSetRWorkValue(S,3,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,4,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,5,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,6,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,7,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,8,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,9,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,10,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,11,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,12,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,13,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,14,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,15,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,16,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,17,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,18,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,19,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,20,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,21,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,22,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,23,((double)((rand()%(int)(201))-100)/200));
    ssSetRWorkValue(S,24,-0.1);
    ssSetRWorkValue(S,25,0);
    ssSetRWorkValue(S,26,0.58);
    ssSetRWorkValue(S,27,-0.462);
    ssSetRWorkValue(S,28,0.48);
    ssSetRWorkValue(S,29,0.19);
    ssSetRWorkValue(S,30,0.14);
    ssSetRWorkValue(S,31,-0.15);
    ssSetRWorkValue(S,32,0.707);
    ssSetRWorkValue(S,33,0.67);
    ssSetRWorkValue(S,34,-0.49);
    ssSetRWorkValue(S,35,0.53);
    ssSetRWorkValue(S,36,0.2);
    ssSetRWorkValue(S,37,0.14);
    ssSetRWorkValue(S,38,-0.15);
    ssSetRWorkValue(S,39,0.7);
  }
#endif /*  MDL_START */
/* Function: mdlOutputs =======================================================
 * Abstract:
 *    In this function, you compute the outputs of your S-function
 *    block. Generally outputs are placed in the output vector, ssGetY(S).
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T i;
    real_T   rate, bias1;
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);   /* referenca
napona */
    const real_T *u_1= (const real_T*) ssGetInputPortSignal(S,1);    /* trenutna
vrijednost napona iz generatora */
    const real_T *u_2= (const real_T*) ssGetInputPortSignal(S,2);    /*Izlaz iz
mreže u prošlom trenutku */
     const real_T *u_3= (const real_T*) ssGetInputPortSignal(S,3);    /* Pogreška
(error dobiven blokovski u simulinku) */
    real_T *y = ssGetOutputPortSignal(S,0);
    real_T  w111=ssGetRWorkValue(S,0);
    real_T  w112=ssGetRWorkValue(S,1);
    real_T  w113=ssGetRWorkValue(S,2);
    real_T  w121=ssGetRWorkValue(S,3);
    real_T  w122=ssGetRWorkValue(S,4);
    real_T  w123=ssGetRWorkValue(S,5);
    real_T  w131=ssGetRWorkValue(S,6);
    real_T  w132=ssGetRWorkValue(S,7);
    real_T  w133=ssGetRWorkValue(S,8);
    real_T  w141=ssGetRWorkValue(S,9);
    real_T  w142=ssGetRWorkValue(S,10);
    real_T  w143=ssGetRWorkValue(S,11);
```

```
    real_T  w151=ssGetRWorkValue(S,12);
    real_T  w152=ssGetRWorkValue(S,13);
    real_T  w153=ssGetRWorkValue(S,14);
    real_T  w161=ssGetRWorkValue(S,15);
    real_T  w162=ssGetRWorkValue(S,16);
    real_T  w163=ssGetRWorkValue(S,17);
    real_T  w211=ssGetRWorkValue(S,18);
    real_T  w212=ssGetRWorkValue(S,19);
    real_T  w213=ssGetRWorkValue(S,20);
    real_T  w214=ssGetRWorkValue(S,21);
    real_T  w215=ssGetRWorkValue(S,22);
    real_T  w216=ssGetRWorkValue(S,23);
    real_T  bias2=ssGetRWorkValue(S,24);
    real_T  brojac=ssGetRWorkValue(S,25);
    real_T  y111=ssGetRWorkValue(S,26);
    real_T  y112=ssGetRWorkValue(S,27);
    real_T  y113=ssGetRWorkValue(S,28);
    real_T  y114=ssGetRWorkValue(S,29);
    real_T  y115=ssGetRWorkValue(S,30);
    real_T  y116=ssGetRWorkValue(S,31);
    real_T  y211=ssGetRWorkValue(S,32);
    real_T  v111=ssGetRWorkValue(S,33);
    real_T  v112=ssGetRWorkValue(S,34);
    real_T  v113=ssGetRWorkValue(S,35);
    real_T  v114=ssGetRWorkValue(S,36);
    real_T  v115=ssGetRWorkValue(S,37);
    real_T  v116=ssGetRWorkValue(S,38);
    real_T  v211=ssGetRWorkValue(S,39);
    rate=-0.04; bias1=0.0 ;  /*Korak ucenja i početna vrijednost pomaka*/
  /********Podešavanje težina********/
  /* Podešavanje težina - vanjski sloj */
   for (i=0;i<4;i++){       /* definiranje broja iteracija po sampling periodu  */
     ssSetRWorkValue(S,18,w211-
rate*u_3[0]*y111*TANSIGder(v211));w211=ssGetRWorkValue(S,18);
     ssSetRWorkValue(S,19,w212-
rate*u_3[0]*y112*TANSIGder(v211));w212=ssGetRWorkValue(S,19);
     ssSetRWorkValue(S,20,w213-
rate*u_3[0]*y113*TANSIGder(v211));w213=ssGetRWorkValue(S,20);
     ssSetRWorkValue(S,21,w214-
rate*u_3[0]*y114*TANSIGder(v211));w214=ssGetRWorkValue(S,21);
     ssSetRWorkValue(S,22,w215-
rate*u_3[0]*y115*TANSIGder(v211));w215=ssGetRWorkValue(S,22);
     ssSetRWorkValue(S,23,w216-
rate*u_3[0]*y116*TANSIGder(v211));w216=ssGetRWorkValue(S,23);
 /* Podešavanje težina - unutarnji sloj */
      ssSetRWorkValue(S,0,w111-
rate*u_3[0]*u[0]*TANSIGder(v111)*w211*TANSIGder(v211));w111=ssGetRWorkValue(S,0);
    ssSetRWorkValue(S,3,w121-
rate*u_3[0]*u[0]*TANSIGder(v112)*w212*TANSIGder(v211));w121=ssGetRWorkValue(S,3);
    ssSetRWorkValue(S,6,w131-
rate*u_3[0]*u[0]*TANSIGder(v113)*w213*TANSIGder(v211));w131=ssGetRWorkValue(S,6);
    ssSetRWorkValue(S,9,w141-
rate*u_3[0]*u[0]*TANSIGder(v114)*w214*TANSIGder(v211));w151=ssGetRWorkValue(S,9);
    ssSetRWorkValue(S,12,w151-
rate*u_3[0]*u[0]*TANSIGder(v115)*w215*TANSIGder(v211));w151=ssGetRWorkValue(S,12);
    ssSetRWorkValue(S,15,w161-
rate*u_3[0]*u[0]*TANSIGder(v116)*w216*TANSIGder(v211));w161=ssGetRWorkValue(S,15);
    ssSetRWorkValue(S,1,w112-
rate*u_3[0]*u_1[0]*TANSIGder(v111)*w211*TANSIGder(v211));w112=ssGetRWorkValue(S,1);
    ssSetRWorkValue(S,4,w122-
rate*u_3[0]*u_1[0]*TANSIGder(v112)*w212*TANSIGder(v211));
w122=ssGetRWorkValue(S,4);
    ssSetRWorkValue(S,7,w132-
rate*u_3[0]*u_1[0]*TANSIGder(v113)*w213*TANSIGder(v211));w132=ssGetRWorkValue(S,7);
    ssSetRWorkValue(S,10,w142-
rate*u_3[0]*u_1[0]*TANSIGder(v114)*w214*TANSIGder(v211));w142=ssGetRWorkValue(S,10)
;
```

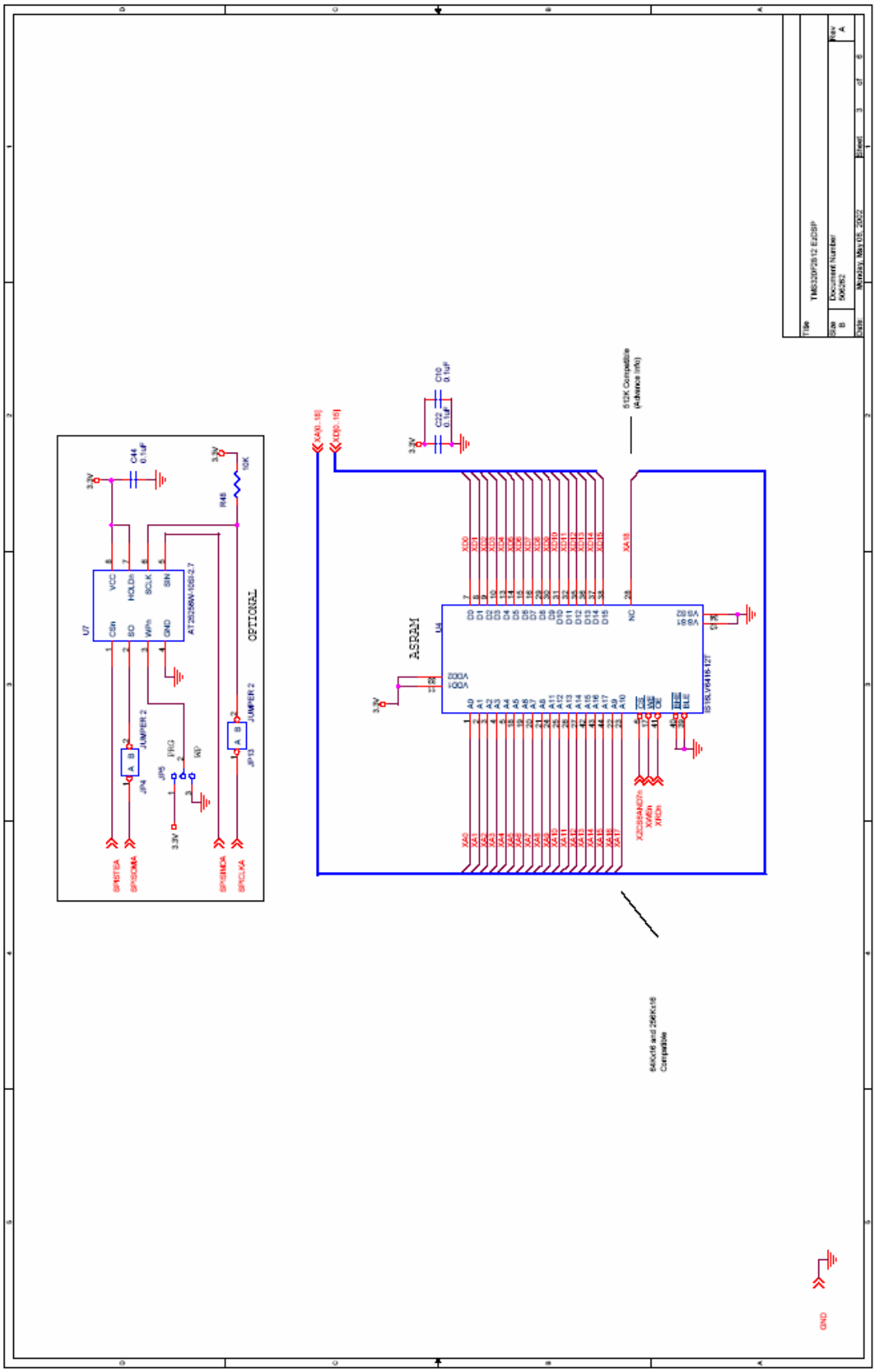
```
    ssSetRWorkValue(S,13,w152-
rate*u_3[0]*u_1[0]*TANSIGder(v115)*w215*TANSIGder(v211));w152=ssGetRWorkValue(S,13)
;
    ssSetRWorkValue(S,16,w162-
rate*u_3[0]*u_1[0]*TANSIGder(v116)*w216*TANSIGder(v211));w162=ssGetRWorkValue(S,16)
;
   ssSetRWorkValue(S,2,w113-
rate*u_3[0]*u_2[0]*TANSIGder(v111)*w211*TANSIGder(v211));w113=ssGetRWorkValue(S,2);
    ssSetRWorkValue(S,5,w123-
rate*u_3[0]*u_2[0]*TANSIGder(v112)*w212*TANSIGder(v211));w123=ssGetRWorkValue(S,5);
    ssSetRWorkValue(S,8,w133-
rate*u_3[0]*u_2[0]*TANSIGder(v113)*w213*TANSIGder(v211));w133=ssGetRWorkValue(S,8);
    ssSetRWorkValue(S,11,w143-
rate*u_3[0]*u_2[0]*TANSIGder(v114)*w214*TANSIGder(v211));w143=ssGetRWorkValue(S,11)
;
    ssSetRWorkValue(S,14,w153-
rate*u_3[0]*u_2[0]*TANSIGder(v115)*w215*TANSIGder(v211));w153=ssGetRWorkValue(S,14)
;
    ssSetRWorkValue(S,17,w163-
rate*u_3[0]*u_2[0]*TANSIGder(v116)*w216*TANSIGder(v211));w163=ssGetRWorkValue(S,17)
;
  /*Podešavanje biasa*/
    ssSetRWorkValue(S,24,bias2+rate*u_3[0]);  bias2=ssGetRWorkValue(S,24);
 /******Izlaz iz neuronske mreže******/
    /* Izlaz - Unutarnji sloj */

ssSetRWorkValue(S,33,u[0]*w111+u_1[0]*w112+u_2[0]*w113+bias1);v111=ssGetRWorkValue(
S,33); ssSetRWorkValue(S,26,TANSIG(v111));y111=ssGetRWorkValue(S,26);

ssSetRWorkValue(S,34,u[0]*w121+u_1[0]*w122+u_2[0]*w123+bias1);v112=ssGetRWorkValue(
S,34); ssSetRWorkValue(S,27,TANSIG(v112));y112=ssGetRWorkValue(S,27);

ssSetRWorkValue(S,35,u[0]*w131+u_1[0]*w132+u_2[0]*w133+bias1);v113=ssGetRWorkValue(
S,35); ssSetRWorkValue(S,28,TANSIG(v113));y113=ssGetRWorkValue(S,28);

ssSetRWorkValue(S,36,u[0]*w141+u_1[0]*w142+u_2[0]*w143+bias1);v114=ssGetRWorkValue(
S,36); ssSetRWorkValue(S,29,TANSIG(v114));y114=ssGetRWorkValue(S,29);

ssSetRWorkValue(S,37,u[0]*w151+u_1[0]*w152+u_2[0]*w153+bias1);v115=ssGetRWorkValue(
S,37); ssSetRWorkValue(S,30,TANSIG(v115));y115=ssGetRWorkValue(S,30);

ssSetRWorkValue(S,38,u[0]*w161+u_1[0]*w162+u_2[0]*w163+bias1);v116=ssGetRWorkValue(
S,38); ssSetRWorkValue(S,31,TANSIG(v116));y116 =ssGetRWorkValue(S,31);
     /* Izlaz - Vanjski sloj */
     v211=y111*w211+y112*w212+y113*w213+y114*w214+y115*w215+y116*w216-bias2;
ssSetRWorkValue(S,39,v211);
     y211=TANSIG(v211); ssSetRWorkValue(S,32,y211);
     }
    y[0] =y211;   /*****IZLAZ IZ MREŽE ****/
    ssSetRWorkValue(S,25,brojac+1);
 }
/*=============================================================*
 * See sfuntmpl_doc.c for the optional S-function methods *
 *=============================================================*/
/*=============================*
 * Required S-function trailer *
 *=============================*/
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

DODATAK C

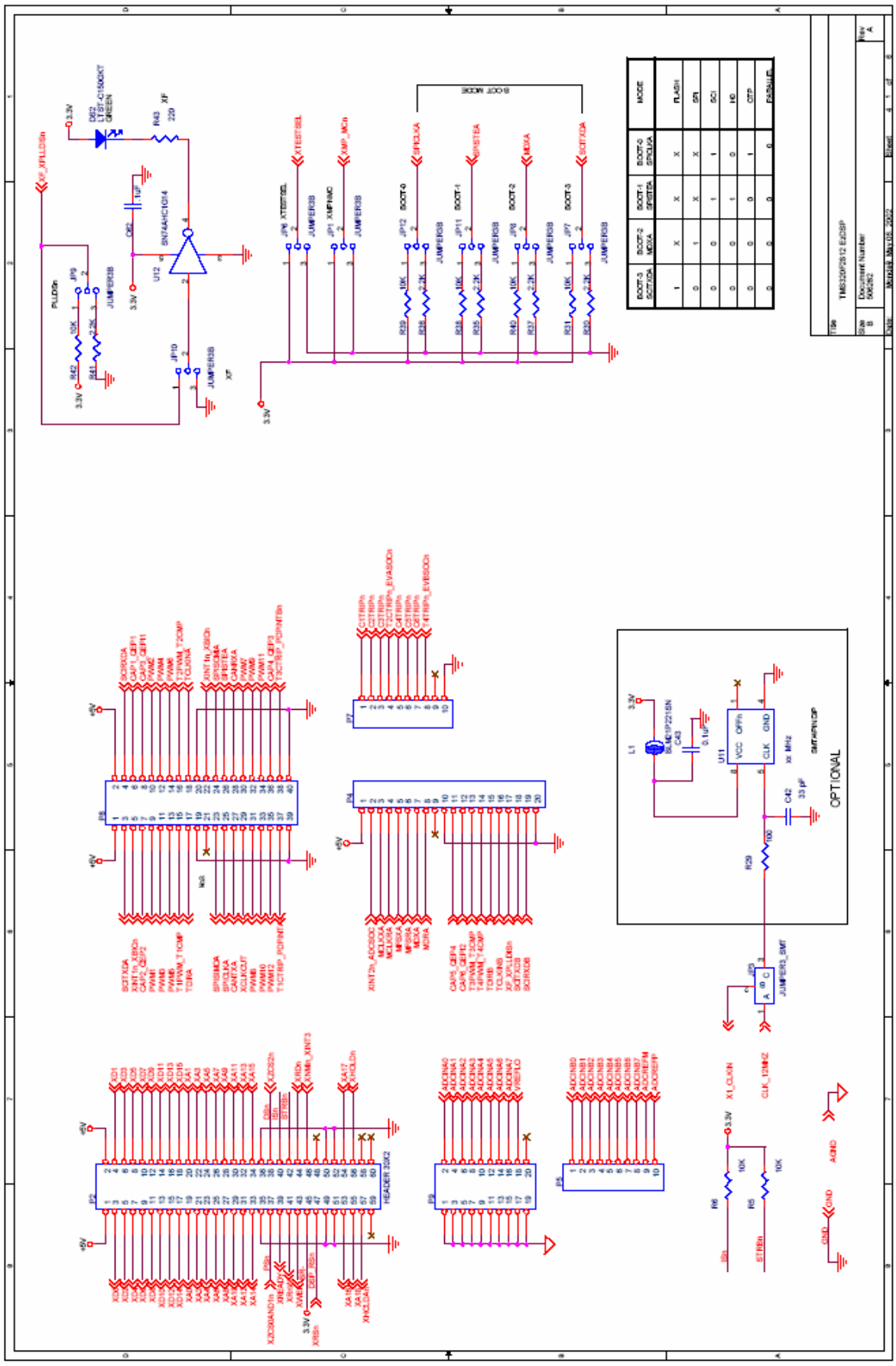
Sheme procesorske pločice



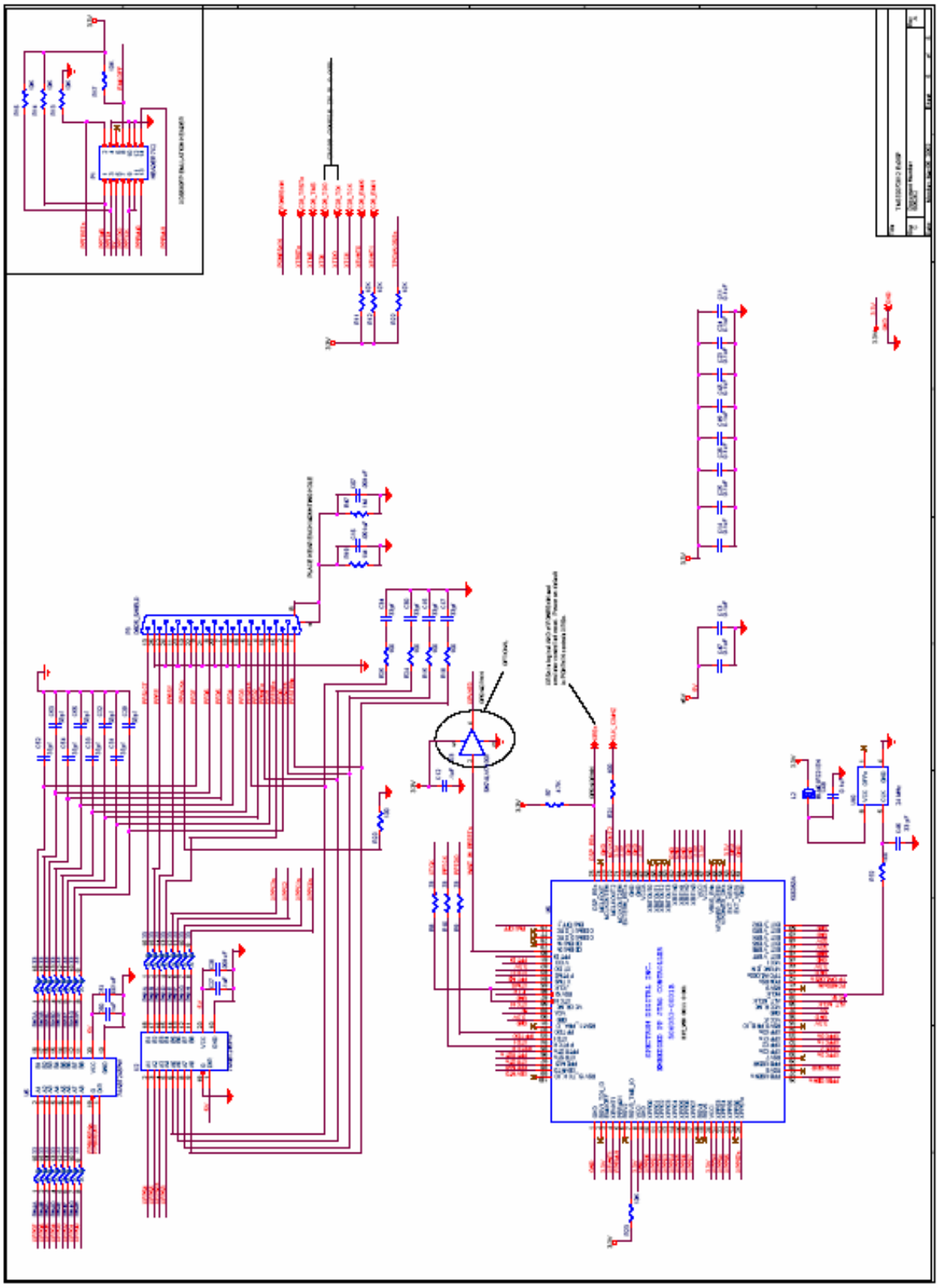
Slika C.1. DSP



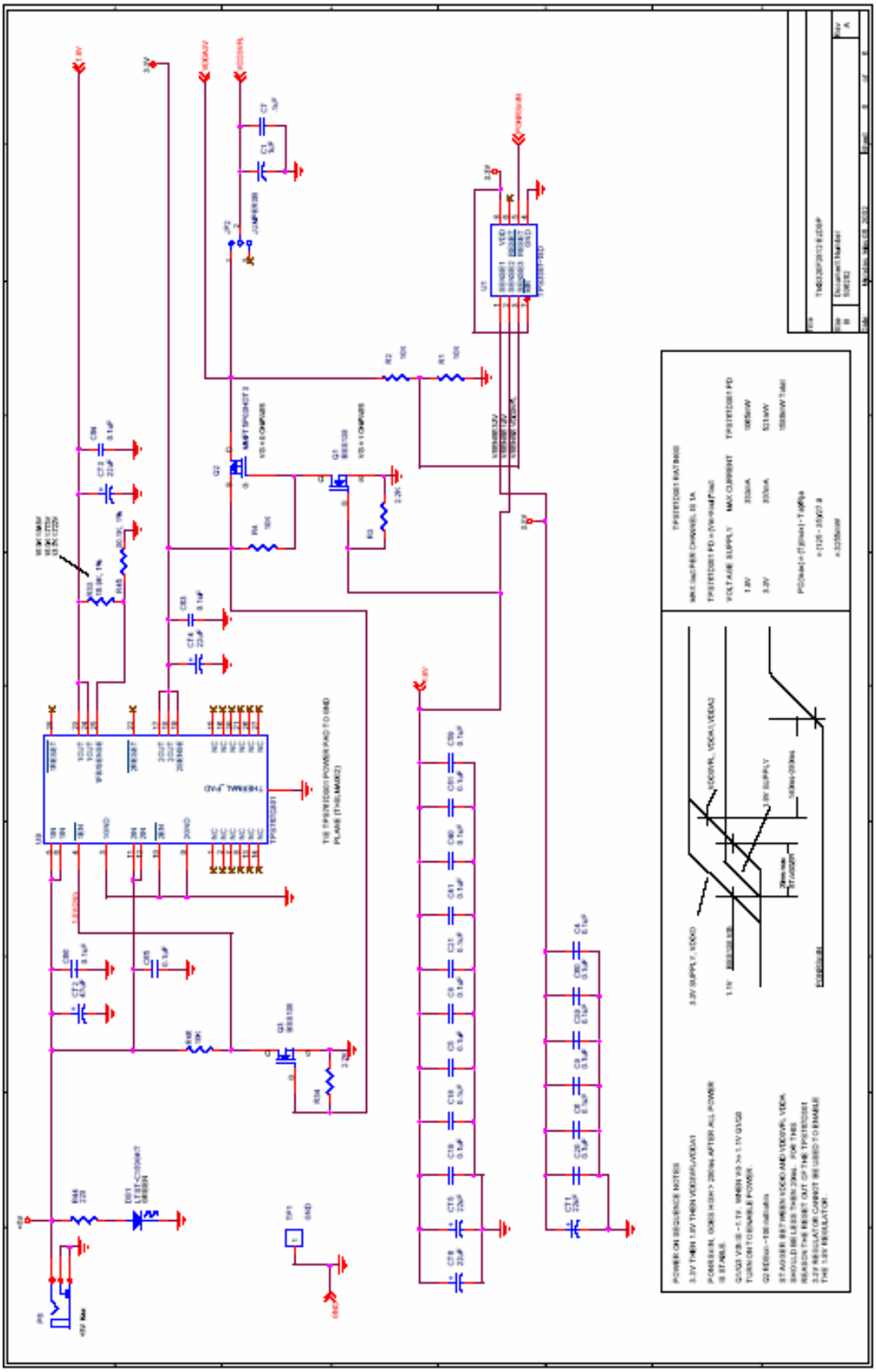
Slika C.2. Vanjska (SARAM) memorija



Slika C.3. Priključci na pločici



Slika C.4. JTAG emulator



Slika C.5. Napajanje

DODATAK D

SUSTAV ZA REGULACIJU UZBUDE SINKRONOG GENERATORA ZASNOVAN NA PROCESORU TMS320F2812

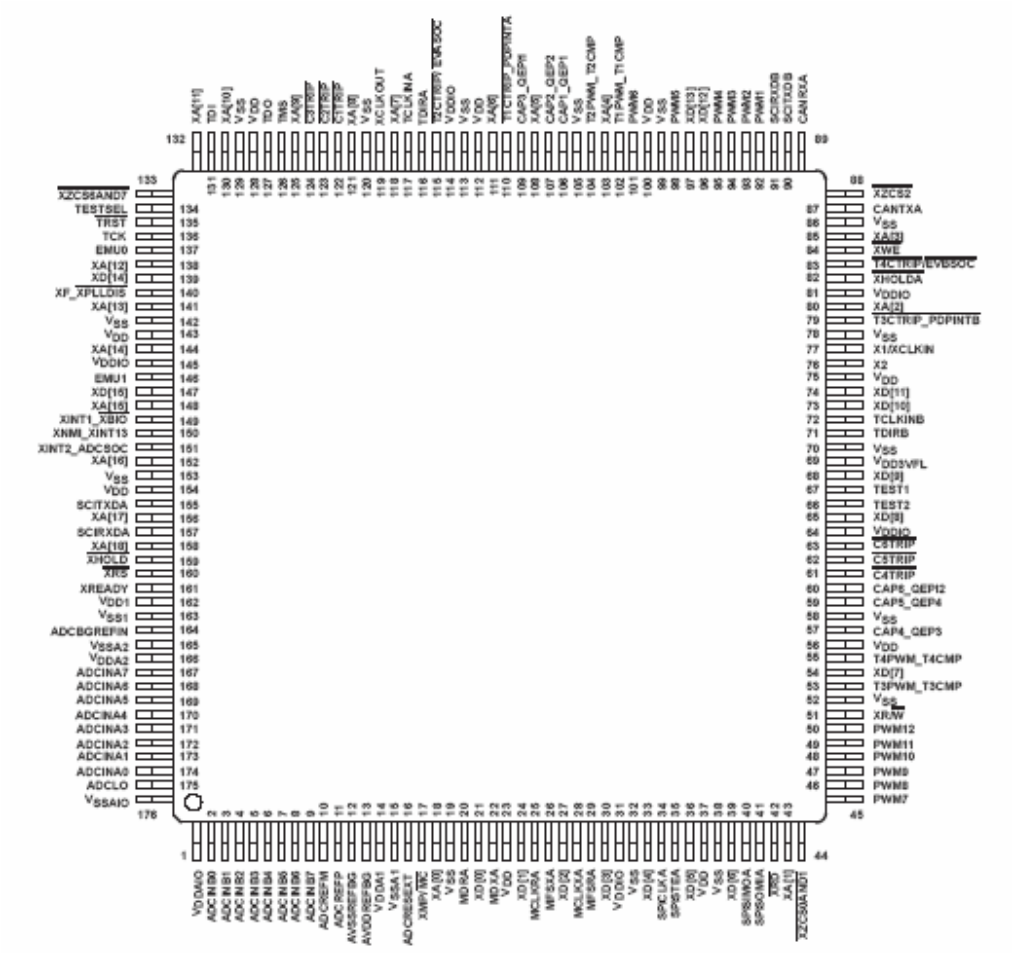
Sustav za upravljanje sustavom uzbude realiziran u ovom radu baziran je na procesoru za obradu signala TMS320F2812 razvijenim od tvrtke *Texas Instruments*.

Karakteristike TMS320F2812 digitalnog procesora za obradu signala su:

- statička CMOS tehnologija koju karakterizira
  - frekvencija 150 MHz (vrijeme ciklusa 6.67ns)
  - nisko napajanje (1.8V jezgra, 3.3V U/I)
  - 3.3V *Flash Programming Voltage* (napon pri programiranju *flash* memorije)
- JTAG podrška
- 32-bitna CPU
- ugrađena (*on-chip*) memorija
  - 128K x 16 Flash memorije
  - 1K x 16 OTP ROM
  - 18K x 16 SARAM
- Boot ROM (4K x 16)
- vanjsko memorijsko sučelje
- tri vanjska prekida
- PIE blok koji podržava 45 vanjska prekida od strane periferije
- tri 32-bitna CPU *timer-a*
- periferni uređaji za upravljanje elektromotornim pogonima (*Motor Control Peripherals*)
  - dva *Event Manager*-a (EVA, EVB)
- serijski kanali
  - serijsko periferno sučelje (SPI)
  - dva serijska komunikacijska sučelja (SCI-a), standardni UART
  - napredna mrežna kontrola (eCAN)
  - višekanalni serijski kanal (McBSP) sa SPI režimom rada
- 12-bitni ADC, 16 kanala
  - 2 x 8 kanalni ulazni multipleksor
  - dva *Sample-and-Hold*
  - odvojena/istovremena *(Single/Simultaneous)* konverzija

- brzina konverzije: 80 ns/12.5 MSPS
- 16-kanalni PWM blok
- razvojni alat uključuje
  - ANSI C/C++ kompajler/asembler/linker
  - podržava TMS320C24x™/240x instrukcije
  - Code Composer Studio™ IDE
  - DSP/BIOS™
- stanja niskog napajanja i ušteda energije
  - podržava IDLE, STANDBY i HALT načine rada
- temperaturne mogućnosti
  - A: –40°C to 85°C (GHH, PGF, PBK)
  - S: –40°C to 125°C (GHH, PGF,PBK)

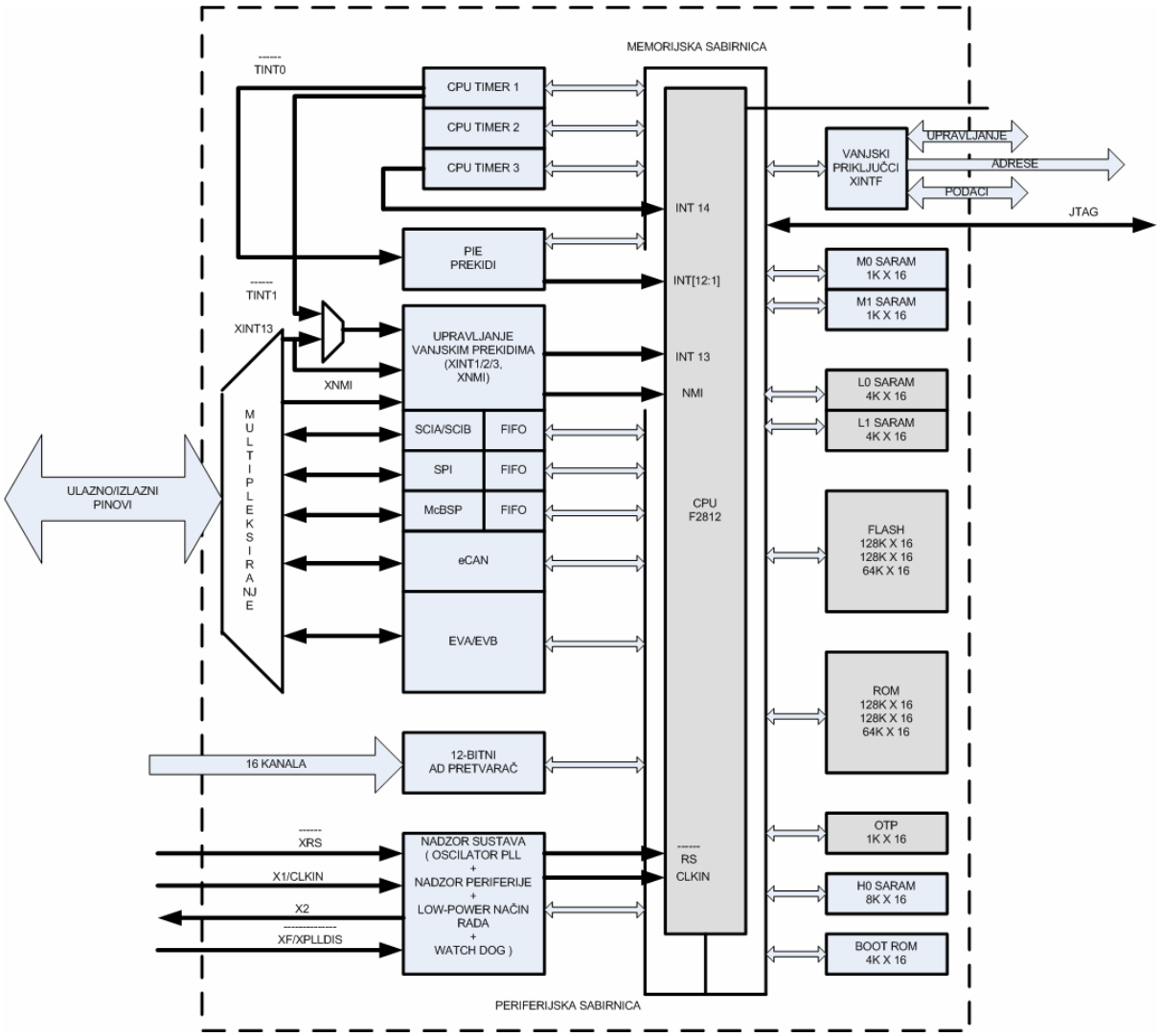
Procesor se nalazi u standardnom kućištu za montažu na štampanoj elektroničkoj pločici za površinsku montažu (SMD). Izgled i raspored pinova na kućištu prikazan je na slici D.1.



Slika D.1. Prikaz rasporeda pinova na procesoru TMS320F2812

Detaljan funkcijski opis pinova nalazi se u literaturi [13],[14].Puna oznaka verzije procesora upotrijebljenog u ovom radu je TMS320F2812 PGF LQFP, što znači da je ovaj procesor iz generacije 320, posjeduje integriranu *flash* memoriju (oznaka F), 2812 označava seriju. Osim navedenog posjeduje mogućnost priključenja vanjske memorije (PGF) te je izveden u 176 pinskom kućištu za površinsku montažu (LQFP).

Funkcionalni opis procesora dan je na slici D.2. Procesor se sastoji od jezgre na koju se nadovezuju periferalni uređaji integrirani u procesor. CPU samog procesora je povezan sabirnicom s ostalim funkcijskim dijelovima u procesoru. Direktno u CPU se dvosmjerno vode signali za JTAG emulator te signali prekida (*interrupt*) .



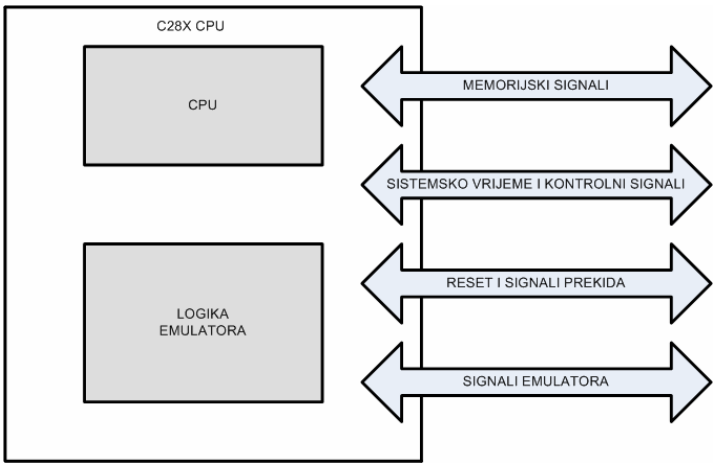
Slika D.2. Funkcionalne cjeline TMS320F2812 procesora

D1 CPU

CPU procesora TMS320F2812 sastoji se od [15](Slika D3) :

- CPU –a za obradu podataka
- Emulacijsku logiku za kontrolu i nadzor pojedinih sklopova i ispravan rad procesora

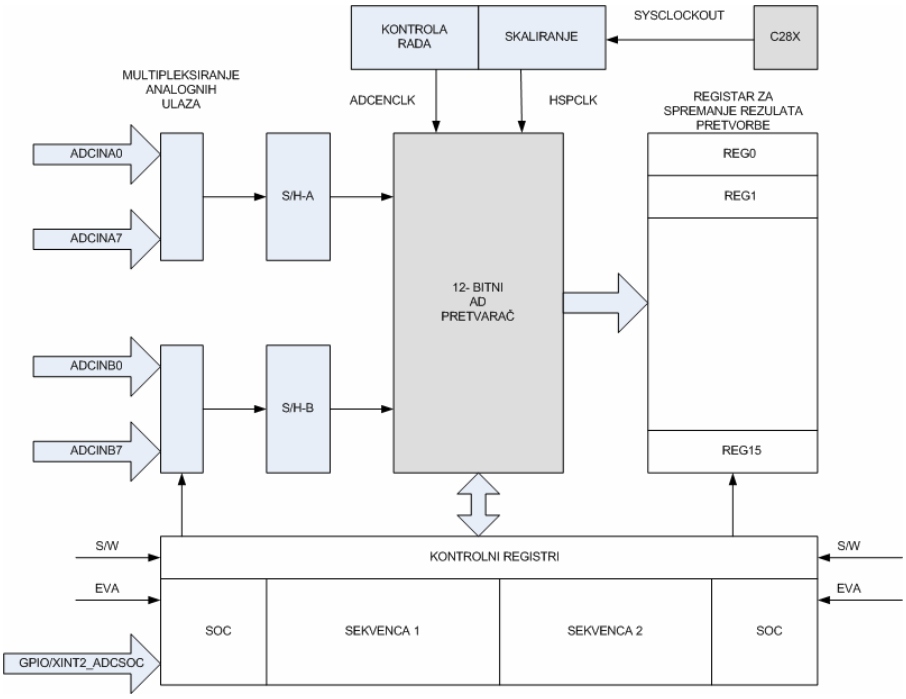
- Signale za spajanje sa memorijom i periferijom



Slika D.3. Konceptualni dijagram CPU-a procesora TMS320F2812

D2 ANALOGNO DIGITALNI PRETVORNIK ADC

Analogno digitalni konvertor realiziran u procesoru ima jedan A/D konvertor na koji se vode 16 kanala za konverziju (Slika D.4.). Analogno digitalna konverzija ostvaruje se u 12 bitnoj rezoluciji, čija se vrijednost kasnije pohranjuju u registre.



Slika D.4. ADC modul

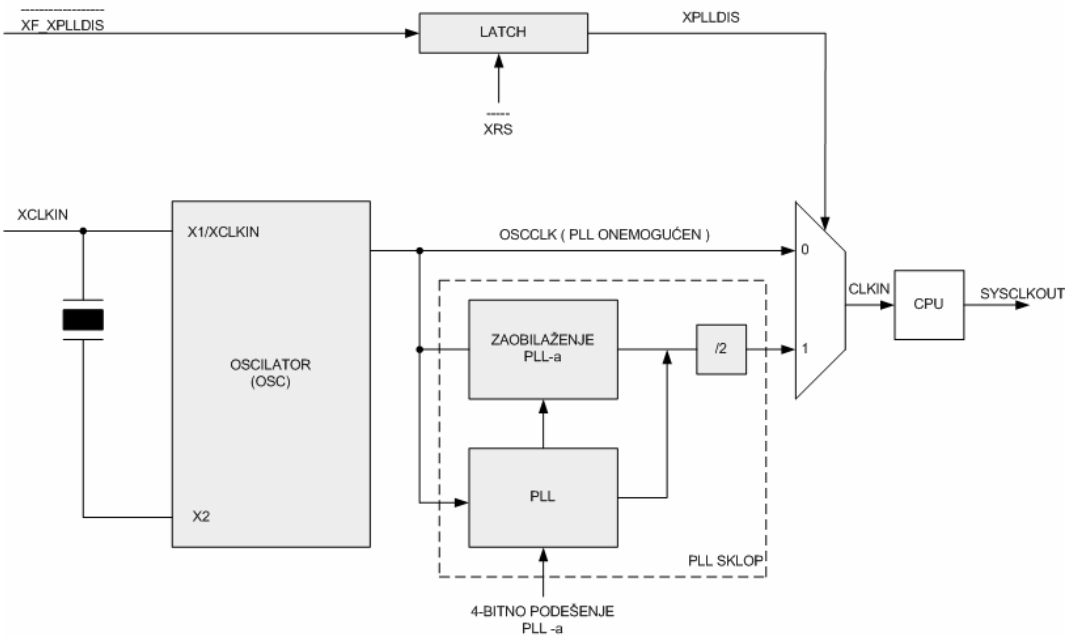
Brzina A/D konverzije određuje se skaliranjem sistemskog vremena te je na taj način moguće ostvarivati različita vremena konverzije. Vrijednost koja se dobije konverzijom izračunata je na slijedeći način:

$$Iznos = 4095 \cdot \frac{ulazni\_napon - ADCLO}{3} \qquad \text{(D.1)}$$

A/D konvertor može raditi u dva moda rada simultani ili sekvencijalni režim rada. Kod simultanog režima rada uzmu se vrijednosti analognih ulaza te se spreme u *Sample and hold buffer*-e. Zatim se slijedno u A/D konvertoru pretvore u digitalni format i spreme u izlazne registre. Kod sekvencijalnog režima rada slijedno se konvertiraju vrijednosti jedna za drugom. Početak pojedine konverzije može se sinkronizirati sa sistemskim vremenom, vanjskim signalom (GPIO/XINT2_ADCSOC) ili se konverzije odvijaju jedna za drugom bez sinkronizacije s prekidom ("*free run*"). U radu je A/D konvertor podešen tako da se sinkronizira s sistemskim vremenom te je period uzorkovanja 1/2875 sekundi. Trenutak konverzije u praksi se gotovo uvijek sinkronizira s nekim od prekida. Te se na taj način izbjegava generiranje šumova.[16]

D3 SISTEMSKO VRIJEME (PLL CLOCKING)

Sklop za podešavanje sistemskog vremena služi za određivanje radnog takta procesora. Na ovaj način je određena radna frekvencija procesora. Na vanjski pin procesora spaja se kristalni oscilator frekvencije 30 MHz te se taj signal uvodi u ustav kao x1/xclkin signal. Zatim se pomoću *PLL* podiže na zadanu vrijednost (Slika D.5.).

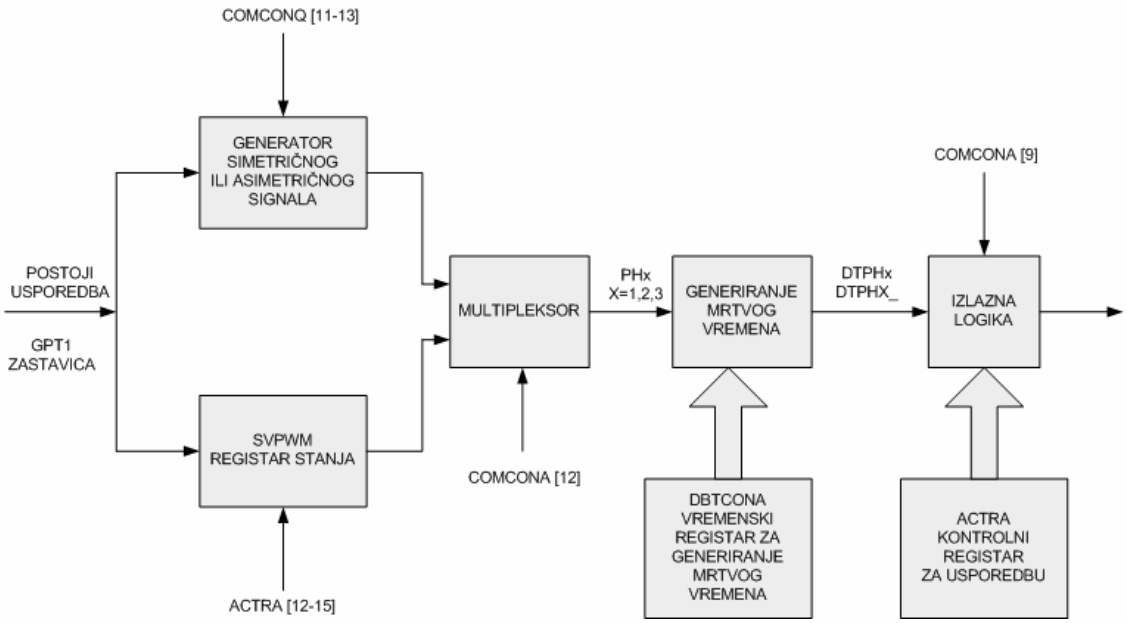


Slika D.5. Sistemsko vrijeme *PLL*

Vrijednosti s kojima se množi ulazni signal su 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5. U radu *PLL* je podešen na 5 tako da kad se pomnoži s ulaznim signalom od 30 MHz dobijemo da procesor radi na 150 MHz [17].

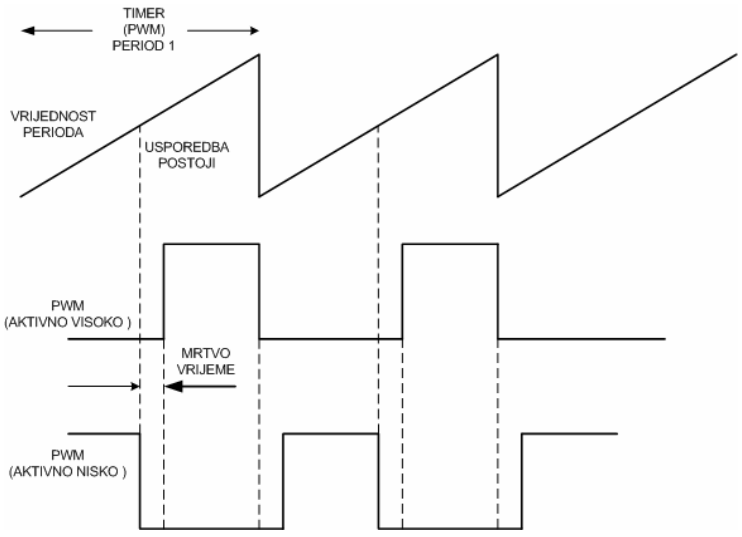
D4 PULSNO-ŠIRINSKO SKLOPOVLJE

Sklopovlje za generiranje pulsno-širinskog signala prikazano je na slici D.6. a sastoji se od sklopa za generiranje simetričnog ili asimetričnog valnog oblika, sklopa za određivanje mrtvog vremena i izlazne logike. Signali koji izlaze iz ovog sklopa su pravokutnog oblika s promjenjivom širinom impulsa.



Slika D.6. Pulsno širinski sklop

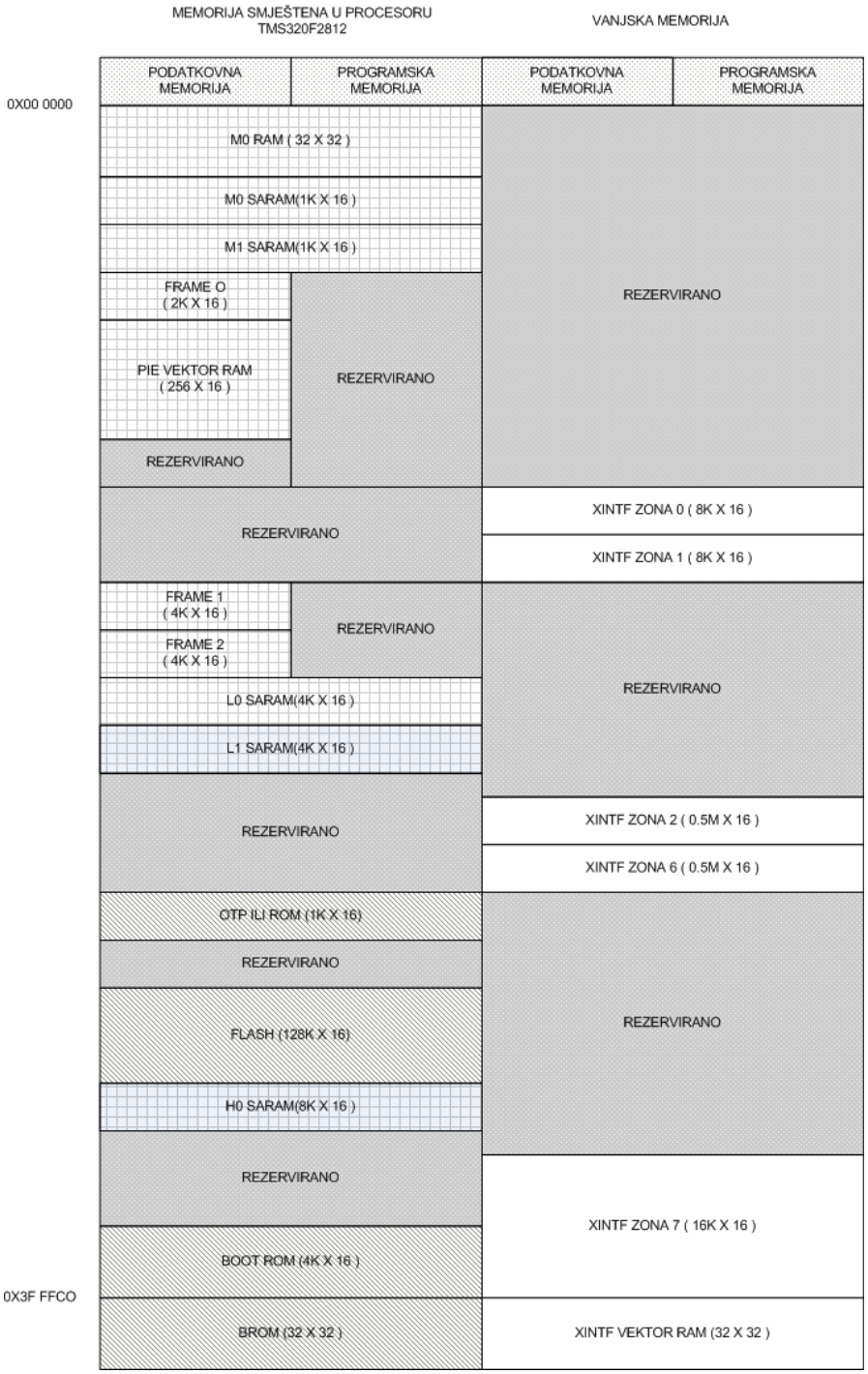
Ulazni signal u sklop je komparacijski signal koji se uspoređuje s trenutnom vrijednošću *Timera*, s tim da je period *Timera* je predefiniran kod inicijalizacije sustava (Slika D.7.). U trenutku kad su komparacijski signal i vrijednost *Timer-a* jednake po iznosu generira se pulsno-širinski signal. Tako generiran signal vodi se u sklop za generiranje mrtvog vremena gdje se uvodi mrtvo vrijeme između PWM signala i njegovog komplementa. Tako se generiran signal vodi preko izlazne logike na pinove procesora. [18]



Slika D.7. Asimetrični pulsno širinski signal

D5 MEMORIJSKA MAPA

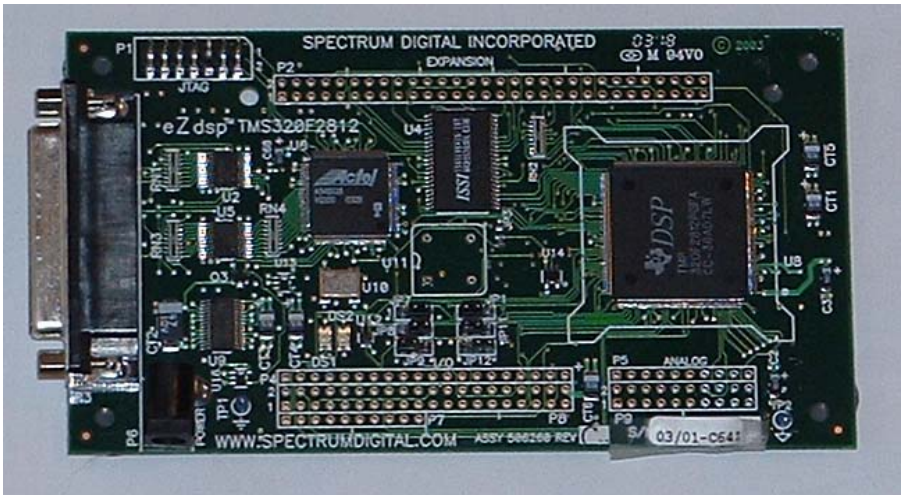
Kod procesora TMS320F2812 je moguće adresirati i prostor van samog procesora koji se nalazi na sabirnica. Tako da se memorijska mapa dijeli na dva dijela memorijski prostor na procesoru i van njega što je prikazano na slici 7.8. Vidljivo je da se može adresirati 1M x 16 bita prostora izvan procesora. Sama memorija unutar procesora podijeljena je na gornjih i donjih 64 k. Nadalje memorija na procesoru je podijeljena od strane korisnika na programsku i podatkovnu (*Page0 i Page1*) [15].



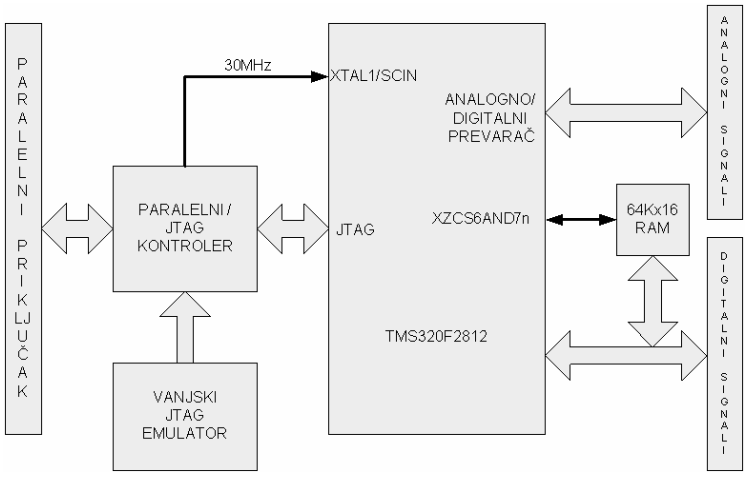
Slika D.8. Memorijska mapa

D6 eZdsp PLOČICA

Procesor TMS320F2812 je smješten na četveroslojnu štampanu pločicu koju proizvodi tvrtka *SpectrumDigital* (Slika D.9.) [19] Na elektroničkoj pločici je osim samog procesora smještena i dodatna memorije od 64k x 16, JTAG kontroler za izmjenu podataka s računalom i razvojnim alatima (Slika D.10.).



Slika D.9. eZdsp pločica

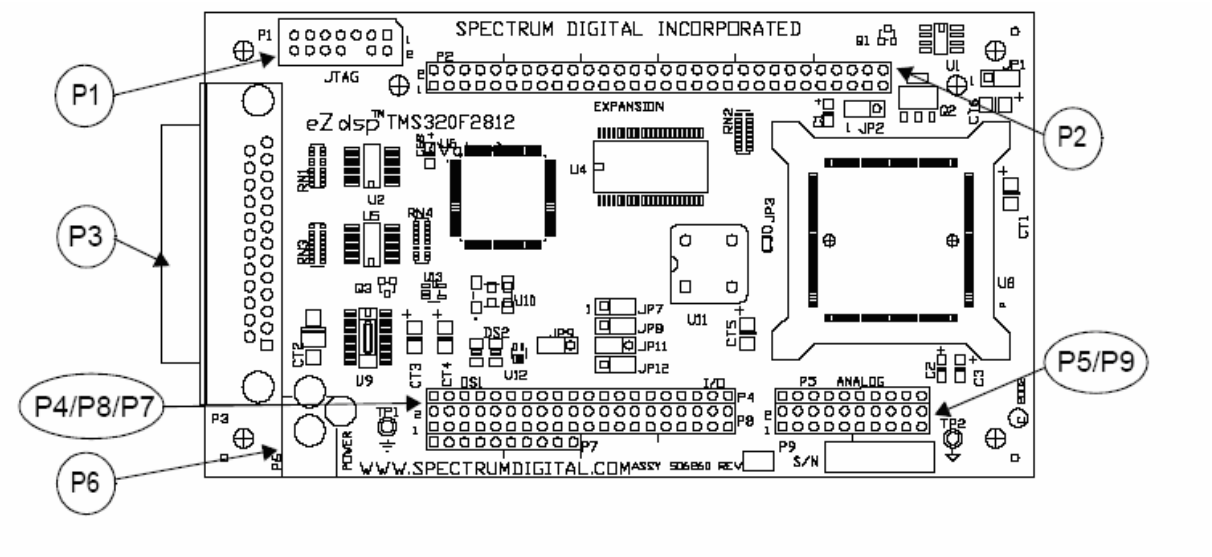


Slika D.10. Shematski prikaz eZdsp pločice

Signali s procesora su izvedeni na konektore koji su grupirani po funkcijama tako da imamo tri funkcijske cjeline konektora, digitalne signale, analogne signale te paralelni port za JTAG komunikaciju. Tabelarno popis konektora je prikazan u tablici D.1., dok je fizički raspored signala prikazan na slici D.11.

PRIKLJUČAK	FUNKCIJA
P1	JTAG priključak
P2	Proširenje
P3	Paralelni priključak/ JTAG priključak
P4/P8/P7	Ulazno/izlazni priključci
P5/P9	Analogni priključci
P6	Napajanje

Tablica D.1. Popis priključaka na eZdsp pločici



Slika D.11. Fizički raspored konektora na pločici

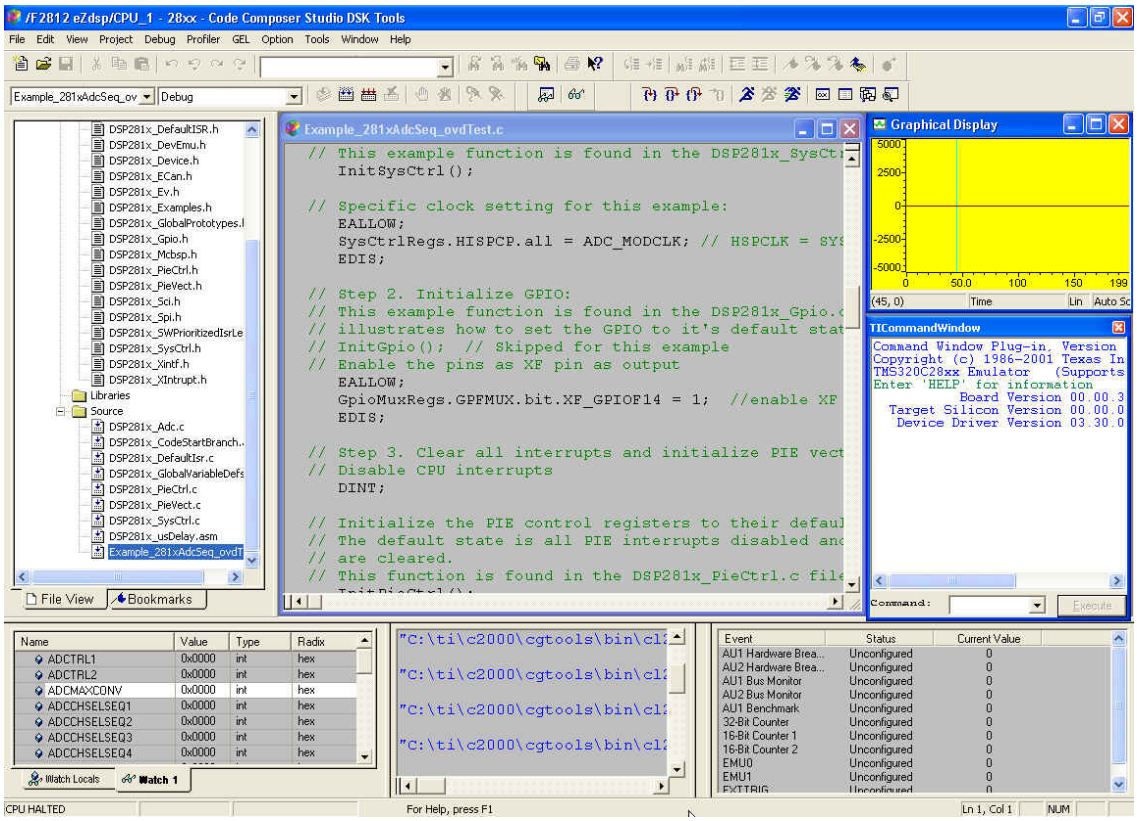
Detaljniji opis kao i funkcijske cjeline eZdsp pločice moguće je naći u literaturi [20] .

D7 PROGRAMSKA PODRŠKA

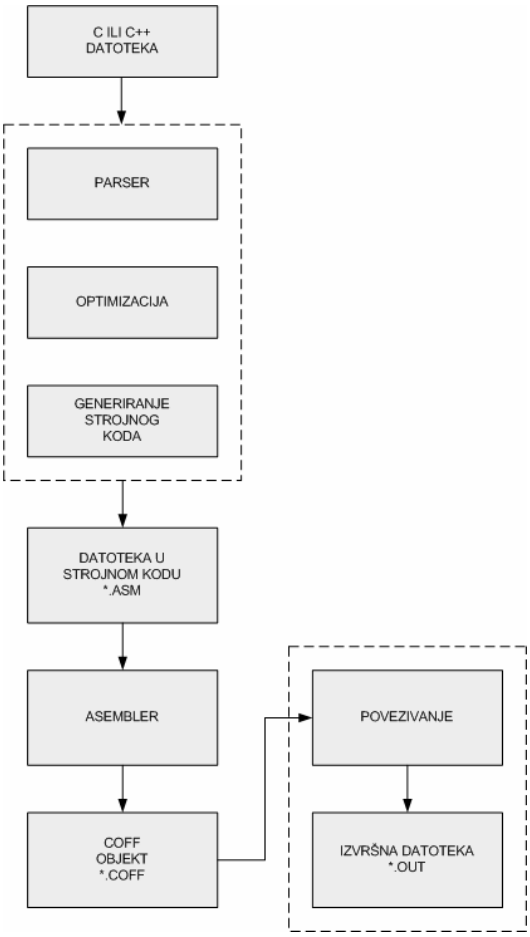
Za izradu programa za sustav regulacije uzbude sinkronog generatora odnosno za programiranje procesora TMS320F2812 korišten je programski razvojni sustav *Code Composer Studio*. Programski paket je integrirano razvojno sučelje s editorom, *debugerr*-om, upravljanjem projektima, grafičkim i parametarskim prikazom varijabli sustava, linijskim naredbenim editorom (Slika D.12.). Osim ovih vizualnih elemenata posjeduje C/C++ kompajler, linker te optimiranje generiranog koda u asembleru. Nadalje omogućava izmjenu podataka u stvarnom vremenu između procesora i računala. Razvoj programa u *Code Composer Studiu* odvija se na taj način da se napišu rutine u C ili C++ programskom jeziku. Od strane *Texas Instruments*-a razvijena je velika biblioteka predefiniranih funkcija (filtriranje, PID regulatori, PWM i razni signalni generatori, razne matematičke funkcije)

tako da se te funkcije samo pozivaju u programu, odnosno nije ih potrebno posebno razvijati. Nakon što se program napiše u programskom jeziku C ili C++ u tekstualnom editoru vrši se generiranje strojnog koda koji se spušta u procesor. Proces prevođenja u strojni kod prikazan je na Slici D.13. Proces prevođenja u strojni kod počinje tako da nakon što je napisan C ili C++ kod kompajler izvrši prevođenje u asemblerski kod.

U kompajleru je moguće uključiti opciju da se kod optimira. Zatim se asemblerski kod prevodi u *Common Object File Format* (COFF format s ekstenzijom *.out) koji se spušta u procesor. Isto tako je moguće podešavati opcije kod prevođenja u COFF datoteku tako je moguće izvesti i optimizaciju koda [21].



Slika D.12. Grafičko sučelje *Code Composer Studio*



Slika D.13. Postupak prevođenja u strojni kod

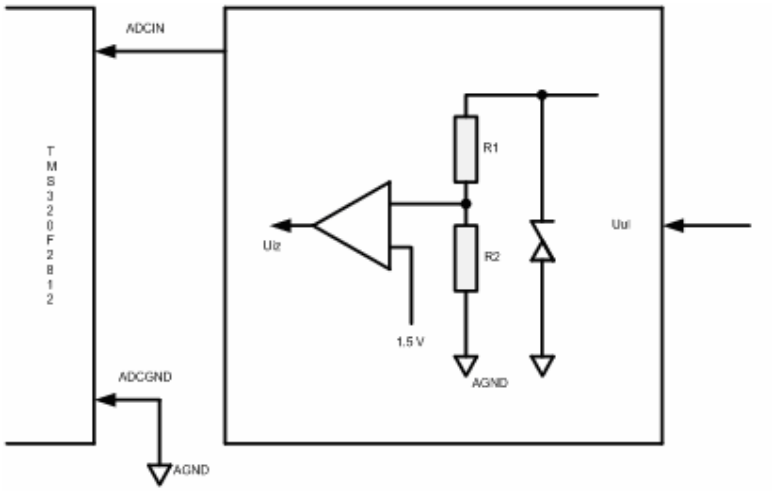
D8 PRILAGODNI SKLOPOVI

Na eZdsp pločici razvedeni su signali s procesora tako da su dovedeni na konektore. Radni napon procesora TMS320F2812 je 3,3 V , a samim tim i svi ulazni i izlazni signali iz procesora su te vrijednosti. Analogni ulazi u procesor su izvedeni tako da se na njih može dovesti napon u granicama 0 do 3 V. Iz tog razloga signale je potrebno prilagoditi po naponskoj i strujnoj razini za spajanje na pojedine komponente u sustavu. Za tu potrebu izrađeni su prilagodni sklopovi na elektroničkoj pločici.

D8.1 SKLOPOVI ZA PRILAGODBU ANALOGNIH ULAZA

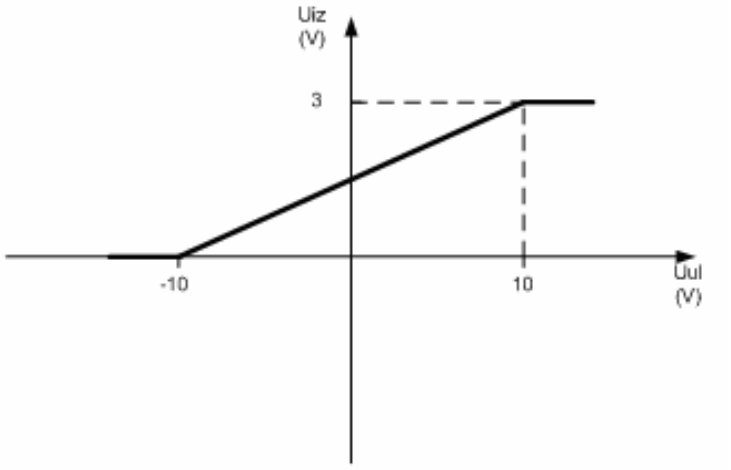
Analogni ulazi u procesor rade na naponu od 0 do 3 V budući da za potrebe ovog rada ta naponska razina ne zadovoljava iz razloga što se mjere naponi i struje sinkronog generatora koji su izmjenične veličine, načinjen je sklop koji omogućuje da se mjere naponi u rasponu +/- 10 V.

Osim za podešavanje naponskih razina na slici D.14. je vidljivo da su postavljene i zaštitne diode koje sprečavaju da se izlazni napon iz prilagodnog sklopa poveća iznad 3 volta što je najveća dopuštena vrijednost ulaznog napona u analogno-digitalni pretvarač.



Slika D.14. Blokovski prikaz prilagodnog sklopa za analogne ulaze

Nadalje prilagodni sklop prvo napon naponske razine -10 do + 10V pretvori u naponsku razinu -1,5 do 1,5 volti. Zatim se tome naponu dodaje istosmjerna komponenta od 1,5 volti tako da se dobije raspon signala od 0 do 3 volta tako da 0 ulaznog napona u prilagodni sklop odgovara 1,5 volti izlaznog napona iz sklopa. Ovisnost izlaznog o ulaznom signalu u prilagodni sklop prikazana je na slici D.15**.**



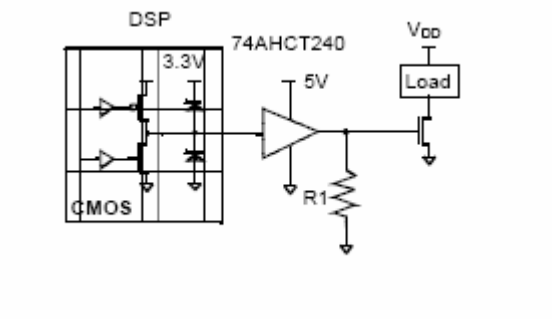
Slika D.15. Ovisnost izlaznog napona prilagodnog sklopa o ulaznom naponu

D8.2 DIGITALNI ULAZI I IZLAZI

Iz razloga strujne i naponske kompatibilnosti signale direktno s DSP-a nije moguće koristiti za digitalne ulaze ili izlaze. Stoga je potrebno prilagoditi te signale za daljnju upotrebu. Osim prilagodbe po iznosu potrebno je štititi i sam procesor od nedopustivo visokih napona (većih od 3,3 V) koji bi mogli dovesti do uništenja procesora.

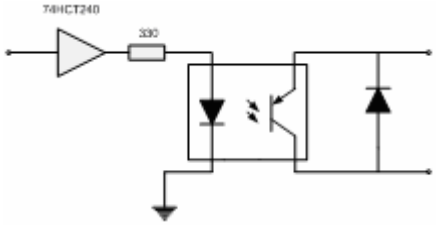
D8.3 DIGITALNI IZLAZI

Preporuka za realizaciju izlaza preuzeta je iz literature [22]. Te na slici D.16. vidimo prijedlog rješenja izlaza. Osim samo dodavanja sklopa 74HCT 240 dodan je još i optokapler za galvansko odvajanje i zaštitna dioda na izlazu iz optokaplera.



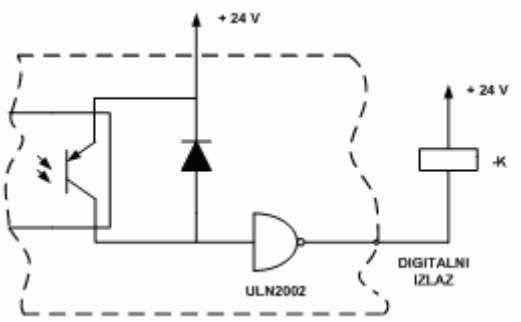
Slika D.16. Preporuka za realizaciju digitalnih izlaza

Galvansko odvajanje signala realizirano je prema literaturi [23], te prikazano na slici D.17.



Slika D.17. Galvansko odvajanje izlaznih signala

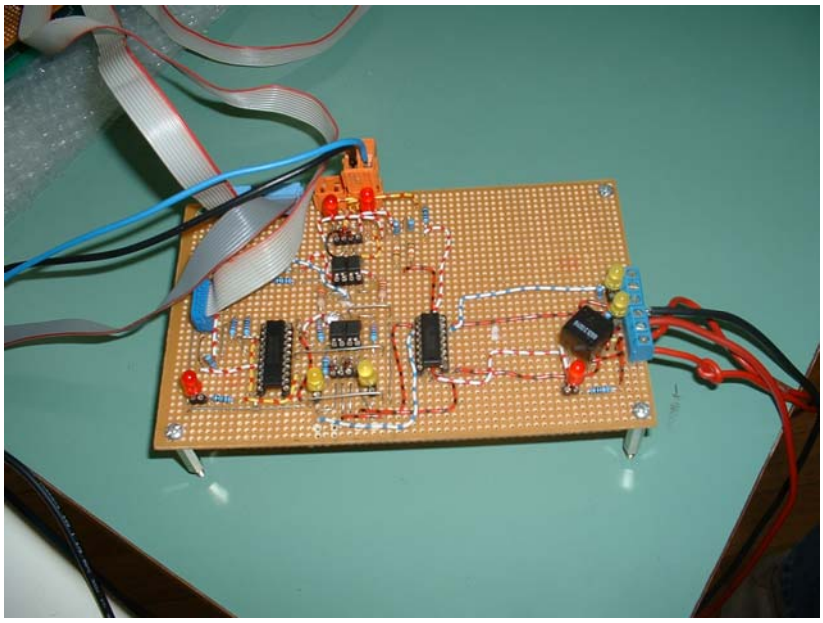
Digitalni izlaz realiziran prema slici D.17. nedostatan je da bi upravljao zahtjevnijim potrošačima (u smislu strujne opteretivosti) stoga je dodan još i element ULN2002 da bi se postigla mogućnost upravljanja sa svitcima releja jer element ULN2002 dopušta strujnu opteretivost od 0,5 A. Takvo rješenje je prikazano na slici D.18. te je u konačnici realizirano na univerzalnoj štampanoj pločici prikazanoj na slici D.19.



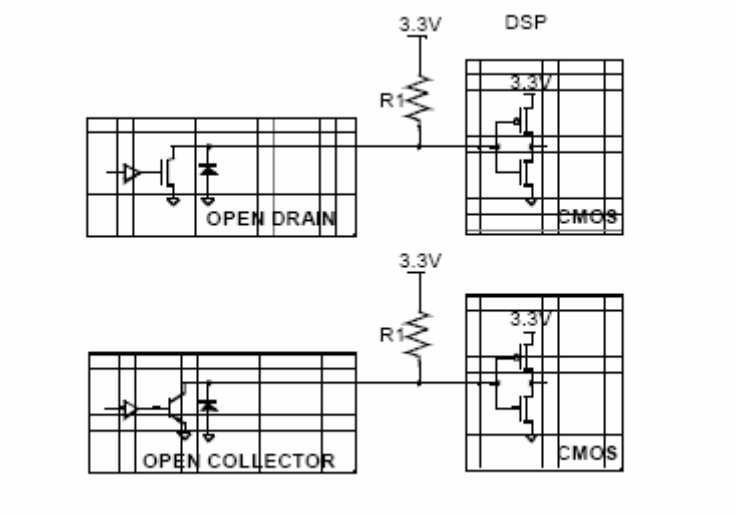
Slika D.18. Detalj realiziranog digitalnog izlaza

D8.4 DIGITALNI ULAZI

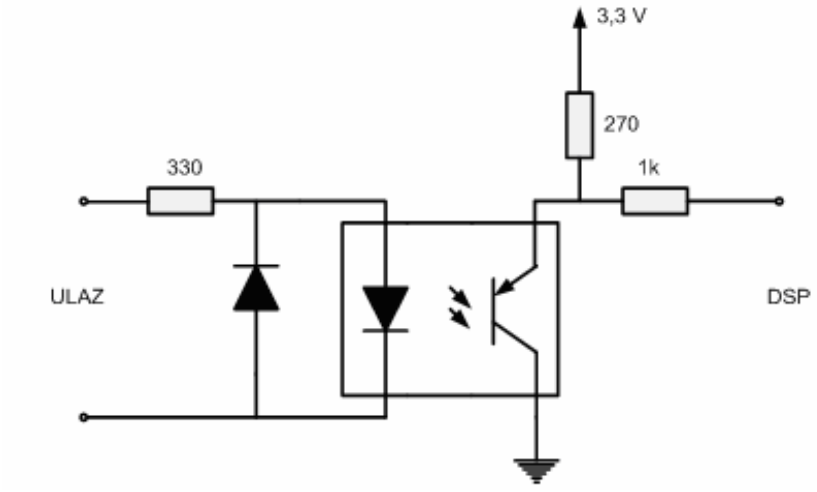
Preporuke za realizaciju digitalnih ulaza korištene su iz iste literature [23] kao i u prethodnom poglavlju te su prikazani na slici D.20. Osim otpornika prema 3,3 V strani izvedeno je galvansko odvajanje u svrhu zaštite DSP-a od previsokog napona. Realizirano rješenje prikazano je na slici D.21.



Slika D.19. Realizacija digitalnih ulaza i izlaza na univerzalnoj tiskanoj pločici



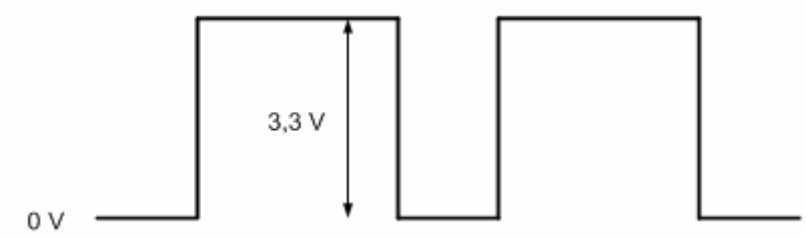
Slika D.20. Preporuka za izvedbu digitalnih ulaza



Slika D.21. Digitalni ulazi

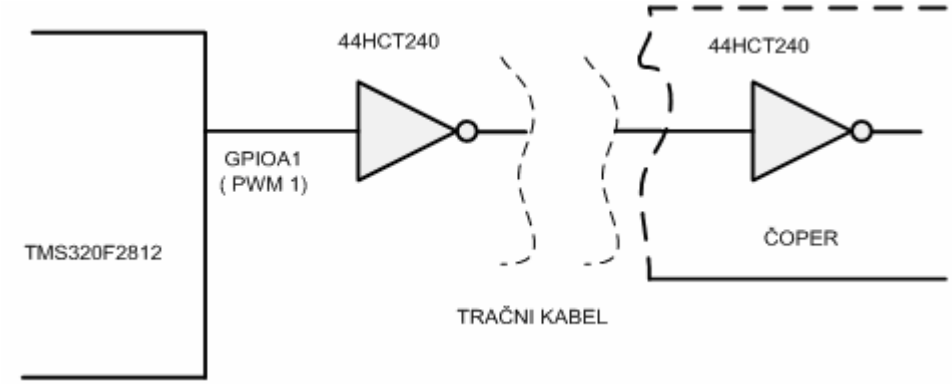
D8.5 PWM SKLOPVI

Signal PWM u 3,3 V razini (Slika D.22.) što znači da niskoj razini odgovara razina od 0 V dok je visoka razina predočena naponom 3,3 V.



Slika D.22. Razine PWM signala

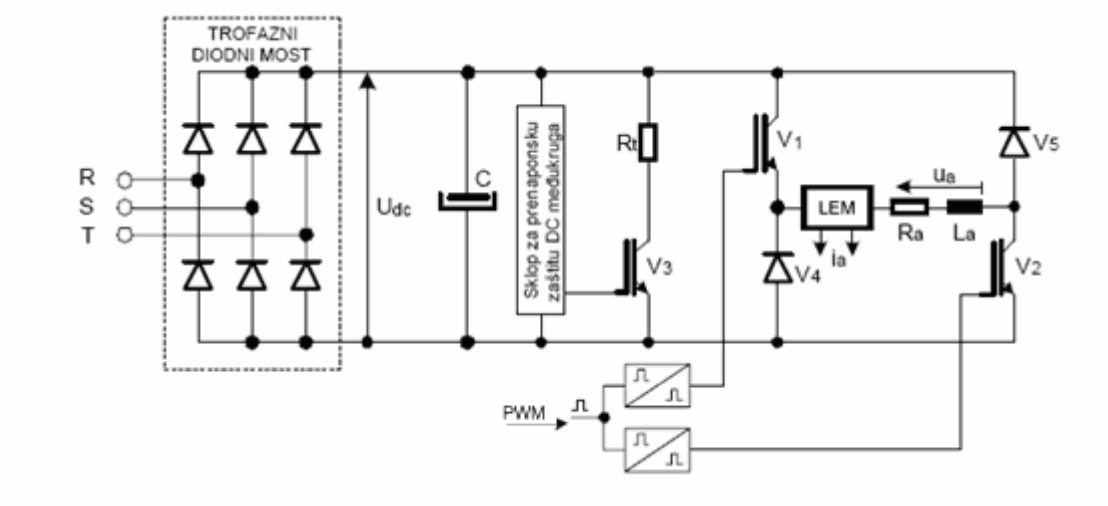
Ulaz u čoper zahtijeva naponsku razinu od 0 do 5 V iz tog razloga signal se vodi preko invertirajućeg *buffera* što je vidljivo iz slike D.23..



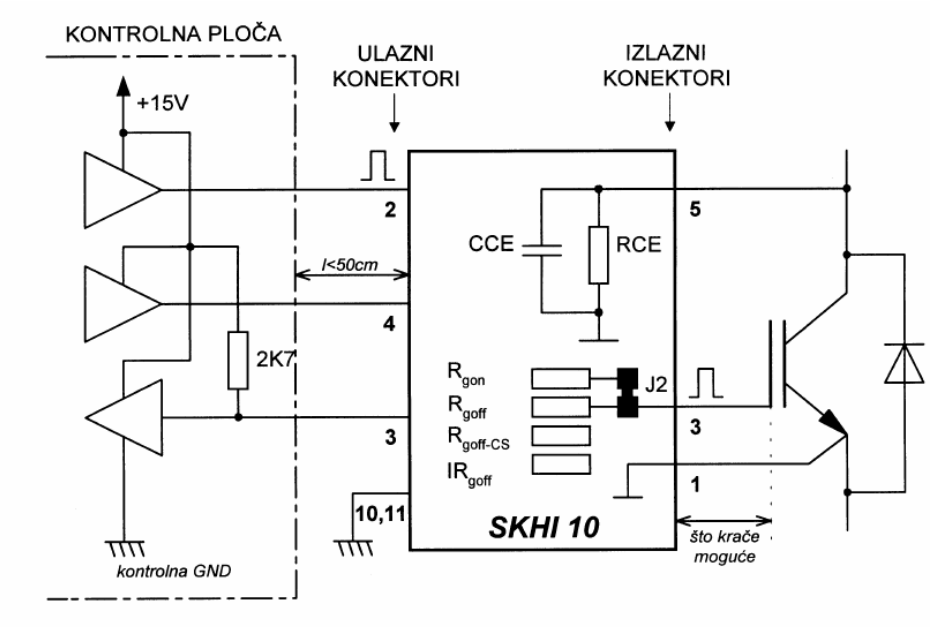
Slika D.23. Prilagodba PWM signala za 5 voltnu razinu

D9 ČOPER

U ovom slučaju korišten je tranzistorski pretvarač za 1. i 4. kvadrant rada prikazan na slici D.24 . Svojstvo pretvarača za 1. i 4. kvadrant rada je da je struja uzbude uvijek istog smjera dok napon na uzbudi može mijenjati polaritet. Za realizaciju takvog pretvarača potrebne su dvije tranzistorske sklopke i dvije diode. Tranzistorske sklopke čopera su realizirane sa bipolarnim tranzistorima upravljanim poljem (IGBT-ima). Oba IGBT-a (na slici su označeni sa V1 i V2) moraju voditi uvijek u isto vrijeme. To je osnovni uvjet koji mora biti ispunjen za pravilan rad ovog pretvarača. Za upravljanje čoperskih IGBT sklopki (V1 i V2) koriste se dva pojačala impulsa SEMIDRIVER-a SEMIKRON SKHI10 (Slika D.25.) [24]. Svako pojačalo impulsa upravlja jednom IGBT sklopkom čopera. Impulsi za upravljanje IGBT sklopki dovode se preko kontrolne pločice iz širinsko-impulsnog modulatora procesora TMS320F2812. Kontrolna pločica za tranzistorski pretvarač služi za povezivanje PWM signala sa pojačalom impulsa za upravljanje IGBT-om. Na nju se dovodi PWM signal i koji se direktno prosljeđuje na pojačala impulsa. Na tom konektoru također se može dobiti informacija greške, a mogu se i resetirati pojačala impulsa u slučaju greške.



Slika D.24. Realizirani čoper za napajanje uzbude



Slika D.25. Izvedba spoja jednog od IGBT-a u pretvaraču na izlaz pojačala impulsa SKHI10

Na kontrolnu pločicu moramo dovesti napajanje 24V. Na njoj se ujedno nalazi i logika za uključivanje sklopa za prenaponsku zaštitu kondenzatora. Naime, elektronički sklop realiziran na kontrolnoj pločici cijelo vrijeme rada mjeri napon istosmjernog međukruga te ovisno o podešenju, uključuje i isključuje IGBT na koji je spojen kočni otpornik. Uključivanje sklopa za prenaponsku zaštitu kondenzatora podešeno je tako da logika proradi pri naponu na kondenzatoru od 250 V [25].

DODATAK E

Organizacija programa u razvojnom okruženju *Code Composer Studio*

Izrada programske aplikacije za procesor TMS320F2812 vrši se u programskom alatu *Code Composer Studio* tvrtke *Texas Instruments.* Programiranje se izvodi u programskom jeziku C a strukturalno je podijeljeno na zaglavlja s deklaracijama, biblioteke ugrađenih funkcija i izvorni kod. Unutar ovih cjelina napravljena je podjela na datoteka koje su funkcionalno vezane uz pojedine hardverske cjeline ili uz programske rutine.

Podjela s kratkim opisom datoteka

Zaglavlja:

- `Device.h` – definicije procesora TMS320F2812
- `DSP28_Adc.h` – definicija A/D konvertora
- `DSP28_CpuTimers.h` – definicija tajmera procesora
- `DSP28_DefaultIsr.h` – definicije korisničkih prekida
- `DSP28_DevEmu.h` – definicija emulatora
- `DSP28_ECan.h` – definicija komunikacije CAN
- `DSP28_Gpio.h` – definicija ulazno izlaznih pinova
- `DSP28_Mcbsp.h` – definicija mcbsp komunikacije
- `DSP28_PieCtrl.h` – definicija pie statusnih registara
- `DSP28_PieVect.h` – definicija pie prekidnog vektora
- `DSP28_Sci.h` – definicija sci komunikacije
- `DSP28_Spi.h` – definicija spi komunikacije
- `DSP28_SWPrioritizedIsrLevels.h` – definicija programskih prekidnih rutina
- `DSP28_SysCtrl.h` – definicija kontrole sustava
- `DSP28_XIntf.h` – definicija vanjskog proširenja
- `DSP28_XInterupt.h` – definicija vanjskih prekida
- `DSP28_GlobalPrototypes.h` – definicija globalnih funkcija
- `qmath.h` – definicija matematičkih funkcija
- `Function.h` – definicija struktura i funkcija unutar programa

Biblioteke ugrađenih funkcija:

- `qmath.lib` – biblioteka matematičkih funkcija
- `rts2800_ml.lib` – biblioteka ugrađenih funkcija procesora TMS320F2812

Izvorni kod:

- `Adc.c` – konfiguriranje i parametriranje A/D konvertora
- `DefaultIsr.c` – korisnički prekidi
- `DLOG4CHC.asm` – funkcije snimanje u stvarnom vremenu
- `DSP28_CodeStartBranch.asm` – konfiguracija za određivanja mjesta pokretanja programa
- `Ev.c` – upravljanje događajima unutar sustava
- `Fir16.asm` – funkcije filtriranja
- `GlobalVariablesdef.c` – definicija globalnih varijabli

- `Gpio.c` – upravljanje pinovima procesora
- `Main.c` – glavna programska rutina
- `PieCtrl.c` – upravljanje Pie registrima
- `PieVect.c` – inicijalizacija i upravljanje Pie vektorima
- `qsqrt.asm` – računanje drugog korijena
- `SysCtrl.c` – inicijalizacija sistemskih registara
- `Function.c` – korisničke funkcije

Programska rutina sustava regulacije u datoteci DefaultIsr.c

```
interrupt void ADCINT_ISR(void)
{
  int16 u;
       int i;
              asm(" SPM  0");
              asm(" CLRC AMODE");
              PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
              AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;
              AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
              brojac++;
              if(brojac<500)      {
              GpioDataRegs.GPFSET.bit.GPIOF14=1;
                                                    }
              if(brojac>500)      {
              GpioDataRegs.GPFCLEAR.bit.GPIOF14=1;
                                                    }
                            if(brojac==1000){brojac=0;}
if(DIG_IO.OUT_1==1){
                                  GpioDataRegs.GPFCLEAR.bit.GPIOF8=1;
                               }
if(DIG_IO.OUT_2==1){
                                  GpioDataRegs.GPFCLEAR.bit.GPIOF9=1;
                               }
if(DIG_IO.OUT_1==0){
                                  GpioDataRegs.GPFSET.bit.GPIOF8=1;
                               }
if(DIG_IO.OUT_2==0){
                                  GpioDataRegs.GPFSET.bit.GPIOF9=1;
                               }
if(GpioDataRegs.GPFDAT.bit.GPIOF10==1){
                                                                  DIG_IO.IN_1=0;
                               }

if(GpioDataRegs.GPFDAT.bit.GPIOF10==0){
                                                                  DIG_IO.IN_1=1;
                               }
if(GpioDataRegs.GPFDAT.bit.GPIOF12==1){
                                                                  DIG_IO.IN_2=0;
                               }
if(GpioDataRegs.GPFDAT.bit.GPIOF12==0){
                                                                  DIG_IO.IN_2=1;
                               }
              UAB.af=(3.0/4096)*(float32)(AdcRegs.ADCRESULT0 >> 3);
              UAB.bf=(3.0/4096)*(float32)(AdcRegs.ADCRESULT1 >> 3);
              UAB.af=UAB.af-UAB.OFFSET;
              UAB.bf=UAB.bf-UAB.OFFSET;
              IAB.af=(3.0/4096)*(float32)(AdcRegs.ADCRESULT8 >> 3);
              IAB.bf=(3.0/4096)*(float32)(AdcRegs.ADCRESULT9 >> 3);
              IAB.af=IAB.af-IAB.OFFSET;
              IAB.bf=IAB.bf-IAB.OFFSET;
              UAB.aqf=UAB.af;
              UAB.bqf=0.577350269189625764509148780501960*(-2*UAB.bf+UAB.af);
              UAB.modsqf=UAB.aqf*UAB.aqf+UAB.bqf*UAB.bqf;
              IAB.aqf=(K.ka1*IAB.bf-K.ka3*IAB.af);
```

```
              IAB.bqf=K.ka2*IAB.af;
              IAB.modsqf=(IAB.aqf*IAB.aqf+IAB.bqf*IAB.bqf);
              UAB.ulsqrtf=UAB.modsqf;
              UAB.ulsqrti=UAB.ulsqrtf*65536;
              UAB.izsqrti=qsqrt(UAB.ulsqrti);
              UAB.izsqrtf=UAB.izsqrti*0.00390625;
              IAB.ulsqrtf=IAB.modsqf;
              IAB.ulsqrti=IAB.ulsqrtf*65536;
              IAB.izsqrti=qsqrt(IAB.ulsqrti);
              IAB.izsqrtf=IAB.izsqrti*0.00390625;
if(DIG_IO.IN_2==0){CP1.syncro=1;}
if(DIG_IO.IN_2==1){CP1.syncro=0;}
if(CP1.enable_syncro==0){DIG_IO.OUT_2=0;}
if(CP1.enable_syncro==1){DIG_IO.OUT_2=1;}
              filtulaz=UAB.izsqrti;
         fir.input=filtulaz;
         fir.calc(&fir);
         UAB.output_filt=fir.output*UAB.scalval*0.00390625;
         fir2.input=IAB.izsqrti;
         fir2.calc(&fir2);
         IAB.output_filt=fir2.output*IAB.scalval*0.00390625*K.scal_iact;
         FILT_POM=(-((3.0/4096)*(float32)(AdcRegs.ADCRESULT2 >> 3)-3.2))*8192;
              fir1.input=FILT_POM;
              fir1.calc(&fir1);
              CP1.Uacti=fir.output;
if(CP1.kompenzacija==0)    {CP1.Uact=UAB.output_filt;}
if(CP1.kompenzacija==1)    {CP1.Uact=UAB.output_filt
              +CP1.faktor_kompenzacije*CP1.POW_act_Q_filt;}
              CP1.Iacti=fir2.output;
              CP1.Iact=IAB.output_filt;
              CP1.POW_act_P=0.5*(UAB.aqf*IAB.aqf-UAB.bqf*IAB.bqf);
              CP1.POW_act_Q=0.5*(UAB.bqf*IAB.aqf+UAB.aqf*IAB.bqf);
              fir3.input=CP1.POW_act_Q*K.Q_scal_int;
              fir3.calc(&fir3);
              CP1.POW_act_Q_filt=fir3.output*K.Q_scal_fl*K.scal_Q;
              fir4.input=CP1.POW_act_P*K.P_scal_int;
              fir4.calc(&fir4);
              CP1.POW_act_P_filt=fir4.output*K.P_scal_fl*K.scal_P;
              CP1.POW_acti_Q=fir3.output;
              fir5.input=K.P_mreza*65536;
              fir7.input=fir5.output;
              fir7.calc(&fir7);
CP1.delta_Pe=fir4.output*0.00390625-K.offest_filt;
CP1.dPe=CP1.POW_act_P_filt-CP1.Pe_old;
              fir5.input=CP1.dPe*65536;
              fir5.calc(&fir5);
              CP1.dPe_filt=fir5.output*0.00390625;
              PID3.y=CP1.POW_acti_Q;
              PID3.yf=CP1.POW_act_Q_filt;
              PID3.r=CP1.POW_ref_Q;
         PidCtrl1(&PID3);
       PID1.y = CP1.Uacti;
              PID1.yf = CP1.Uact;
              PID1.r=CP1.Ur;
       PidCtrl1(&PID1);
              NN1.in=CP1.Ur;
              NN1.Uact=CP1.Uact;
if(CP1.stabilization==0){NN1.ERROR=(NN1.in-NN1.Uact)-NN1.Kv*(NN1.Uact-
NN1.Uact1);}//in ulaz u mrežu
if(CP1.stabilization==1){NN1.ERROR=((NN1.in-NN1.Uact)-NN1.Kv*(NN1.Uact-NN1.Uact1))
              ERROR1.UREF=CP1.Ur;
              ERROR1.UACT=CP1.Uact;
              ERROR1.dUact=ERROR1.UREF-ERROR1.UACT_old;
              fir6.input=ERROR1.dUact*65536;
              fir6.calc(&fir6);
              ERROR1.dUact_filt=fir6.output*0.00390625;
       if(CP1.stabilization==0){ERROR1.ERROR=ERROR1.KC*(ERROR1.UREF-ERROR1.UACT)
                                        -ERROR1.Kv*(ERROR1.dUact_filt);}
if(CP1.stabilization==1){ERROR1.ERROR=(ERROR1.KC*(ERROR1.UREF-ERROR1.UACT)
```

```
                                    -ERROR1.Kv*(ERROR1.dUact_filt))
                                    -
((ERROR1.K1*CP1.delta_Pe)+(ERROR1.K2*CP1.dPe_filt));}
                    for(i=0;i<6;i++){
                    NUL[i].IN1[0]=CP1.Ur;
                    NUL[i].IN1[1]=CP1.Uact;
                    NUL[i].IN1[2]=NIZ2.OUT;
                                    };
             ERROR1.TAN_ERR2.x=NIZ2.OUT_SUMA*NIZ2.SCAL_SUM;
             tansig(&ERROR1.TAN_ERR2);
             NIZ2.dfi_v_e=(1-((ERROR1.TAN_ERR2.tanh))*((ERROR1.TAN_ERR2.tanh)));
                    for(i=0;i<6;i++){
                    NIZ2.dW2[i]=ERROR1.RATE*NIZ2.dfi_v_e*ERROR1.ERROR*NIZ2.IN2[i];
                    ERROR1.d_v_1_ulaz=NIZ2.dfi_v_e*CP1.Ur*ERROR1.ERROR*ERROR1.RATE;
             ERROR1.d_v_2_ulaz=NIZ2.dfi_v_e*CP1.Uact*ERROR1.ERROR*ERROR1.RATE;
             ERROR1.d_v_3_ulaz=NIZ2.dfi_v_e*NIZ2.OUT*ERROR1.ERROR*ERROR1.RATE;
             for(i=0;i<6;i++){
             ERROR1.TAN_ERR1[i].x=NUL[i].OUT_SUMA1;
             tansig(&ERROR1.TAN_ERR1[i]);
             NUL[i].dfi_v_e1=(1-((ERROR1.TAN_ERR1[i].tanh)
)*((ERROR1.TAN_ERR1[i].tanh)));
                                    };
             for(i=0;i<6;i++){
                    NUL[i].dW1[0]=ERROR1.d_v_1_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;;
                    NUL[i].dW1[1]=ERROR1.d_v_2_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;;
                    NUL[i].dW1[2]=ERROR1.d_v_3_ulaz*NUL[i].dfi_v_e1*ERROR1.RATE;;
                                                                    };
             for(i=0;i<6;i++){
                    NUL[i].OUT_SUMA1=NUL[i].IN1[0]*(NUL[i].W1[0]+NUL[i].dW1[0])+
                    NUL[i].IN1[1]*(NUL[i].W1[1]+NUL[i].dW1[1])+
                    NUL[i].IN1[2]*(NUL[i].W1[2]+NUL[i].dW1[2]);
                    NUL[i].TANHN1.x=NUL[i].OUT_SUMA1;
                    tansig(&NUL[i].TANHN1);
                    NUL[i].OUT=NUL[i].TANHN1.tanh;
                    };
             for (i=0;i<6;i++){
                    NIZ2.IN2[i]= NUL[i].OUT;
                                    };
                    NIZ2.OUT_SUMA=      NIZ2.IN2[0]*(NIZ2.W2[0]+NIZ2.dW2[0])+
                                           NIZ2.IN2[1]*(NIZ2.W2[1]+NIZ2.dW2[1])+
                                           NIZ2.IN2[2]*(NIZ2.W2[2]+NIZ2.dW2[2])+
                                           NIZ2.IN2[3]*(NIZ2.W2[3]+NIZ2.dW2[3])+
                                           NIZ2.IN2[4]*(NIZ2.W2[4]+NIZ2.dW2[4])+
                                           NIZ2.IN2[5]*(NIZ2.W2[5]+NIZ2.dW2[5])+
                                           NIZ2.b2;
                    NIZ2.TANHN2.x=NIZ2.OUT_SUMA*NIZ2.SCAL_SUM;
                    tansig(&NIZ2.TANHN2);
                    NIZ2.OUT=ERROR1.scalval*NIZ2.TANHN2.tanh;
                    NIZ2.izlaz=(NIZ2.OUT-duty)*NIZ2.scalval_izlaza_prema_procesu;
                           ERROR1.UACT_old=ERROR1.UACT;
                           CP1.Pe_old=CP1.POW_act_P_filt;
if(CP1.CONTROLL_TYPE==0){PID2.r=PID1.u;}
if(CP1.CONTROLL_TYPE==2){PID2.r=NIZ2.izlaz;}
  CP1.Iu_act=fir1.output*0.00390625*2*0.1;
       PID2.yf=fir1.output*0.00390625+PID2.OFFSET;
       PID2.y=fir1.output;
       PidCtrl1(&PID2);
       if(CP1.ENABLE_SYSTEM==1){
if(CP1.AUTO_HAND==1){
       if(CP1.Iu_act < CP1.over_Iu_limit)      {
              if(CP1.Uact < CP1.over_U_limit){       u = (int16)PID2.u-
(int16)(duty*0.01*5000);   }
              if(CP1.Uact > CP1.over_U_limit){       u = 0; }
                                                     }
       if(CP1.Iu_act > CP1.over_Iu_limit)      {u = 0;       }
                                    }
                                  }
if(CP1.AUTO_HAND==0){u=CP1.DUTY_MANUAL*50;}
if(CP1.ENABLE_SYSTEM==0){u = 0;}
```

```
CP1.D=u*0.02;
if(u > 0) EvaRegs.ACTRA.all = ACTRA_pos_mask;
         else      EvaRegs.ACTRA.all = ACTRA_neg_mask;
               EvaRegs.CMPR1 = abs(u);
        SNIMANJE1.s1=SNIMANJE1.i*SNIMANJE1.s_scal_val[0];
        SNIMANJE1.s2=CP1.Ur*SNIMANJE1.s_scal_val[1];
        SNIMANJE1.s3=CP1.Uact*SNIMANJE1.s_scal_val[2];
               SNIMANJE1.s4=IAB.output_filt*SNIMANJE1.s_scal_val[3];
               SNIMANJE1.s5=CP1.delta_Pe*SNIMANJE1.s_scal_val[4];
               SNIMANJE1.s6=CP1.dPe_filt*SNIMANJE1.s_scal_val[5];
               SNIMANJE1.s7=PID2.r*SNIMANJE1.s_scal_val[6];
               SNIMANJE1.s8=PID2.u*SNIMANJE1.s_scal_val[7];
               SNIMANJE1.s9=ERROR1.dUact_filt*SNIMANJE1.s_scal_val[8];
               SNIMANJE1.s10=u*SNIMANJE1.s_scal_val[9];
               SNIMANJE1.s11=CP1.POW_act_P_filt*SNIMANJE1.s_scal_val[10];
               SNIMANJE1.s12=CP1.POW_act_Q_filt*SNIMANJE1.s_scal_val[11];
               SNIMANJE1.s13=IAB.output_filt*SNIMANJE1.s_scal_val[12];
               SNIMANJE1.s14=PID2.yf*SNIMANJE1.s_scal_val[13];
               SNIMANJE1.s15=ERROR1.ERROR*SNIMANJE1.s_scal_val[14];
               SNIMANJE1.s16=FILT_POM*SNIMANJE1.s_scal_val[15];
               IAE1.in_1=CP1.POW_act_P_filt;
               IAE1.in_2=K.p;
               if(IAE1.ENABLE==1){
               IAE1.sum_1=0;
               IAE1.sum_2=0;
               IAE1.K1_sum_1=1;
               IAE1.sum_1_old=0;
               IAE1.IAE1_1_old=0;
               IAE1.sum_deriv_2=0;
               IAE1.Kd_sum_2=1;
               IAE1.sum_deriv_2_old=0;
               IAE1.K2_sum_2=1;
               IAE1.sum_2_old=0;
               IAE1.IAE2_2_old=0;
               IAE1.IAE=0;
               IAE1.IAED=0;
               IAE1.ENABLE=0;
               };
SNIMANJE1.j++;
if(SNIMANJE1.pocetak==1 && SNIMANJE1.i<1023 &&
SNIMANJE1.j==SNIMANJE1.preskakanje_tocaka) {
       if(CP1.RECORDING==1){dlog1.update(&dlog1);}
       SNIMANJE1.s1=SNIMANJE1.i;
       SNIMANJE1.i++;
       SNIMANJE1.j=0;
       IAE1.ENABLE=0;
       iae(&IAE1);
       if(SNIMANJE1.i<50)  {SNIMANJE1.REF_OLD=CP1.Ur;}
       if(SNIMANJE1.i>100 && SNIMANJE1.i<600)        {
       CP1.Ur=SNIMANJE1.REF;
                           }
       if(SNIMANJE1.i>600)       {
       CP1.Ur=SNIMANJE1.REF_OLD;
                                 }
                                 }
if(SNIMANJE1.j==SNIMANJE1.preskakanje_tocaka) {      SNIMANJE1.j=0;}
if(SNIMANJE1.i==1023){SNIMANJE1.i=0;
SNIMANJE1.pocetak=0;
                                 }
       if(CP1.RECORDING==0){dlog1.update(&dlog1);}

} // end ADCINT_ISR()
```

DODATAK F

Deklaracija varijabli (Function.h), glavna rutina (Main.c) i definicija funkcija (Function.c)

F1 Function.h

```
#ifndef PID_H
#define PID_H
#define               sample_period5999
#define               ACTRA_pos_mask      0x0001
#define               ACTRA_neg_mask      0x0004
typedef struct {
   float32 r;
   float32 u;
   float32 e1;
   float32 yf;
   float32 ui1;
   float32 u_max;
   float32 u_min;
   float32 ui_max;
   float32 Kp;
   float32 Ki;
   float32 Kd;
   float32 scalval_yf;
   float32 OFFSET;
   int16 y;
   int16 y1;
   int16 y2;
   int16 y3;
   int16 y4;
  } PID;
typedef struct {
   float32 in_1;
   float32 in_2;
   float32 sum_1;
   float32 sum_2;
   float32 K1_sum_1;
   float32 sum_1_old;
   float32 IAE1_1_old;
   float32 sum_deriv_2;
   float32 Kd_sum_2;
   float32 sum_deriv_2_old;
   float32 K2_sum_2;
   float32 sum_2_old;
   float32 IAE2_2_old;
   float32 IAE;
   float32 IAED;
   int     ENABLE;
  } IAE;
typedef struct {
   float32      af;
   float32             bf;
   float32             aqf;
   float32             bqf;
   float32             modsqf;
   float32             OFFSET;
   float32             ulsqrtf;
   long         ulsqrti;
   int                 izsqrti;
   float32          izsqrtf;
   float32     scalval;
   float32             output_filt;
   } Uab;
 typedef struct {
    long *coeff_ptr;
    long * dbuffer_ptr;
    int  cbindex;
```

```
        int order;
        int input;
        int output;
        void (*init)(void *);
        void (*calc)(void *);
        }FIR16;
    typedef FIR16   *FIR16_handle;
    #define FIR16_DEFAULTS { (long *)NULL, \
                 (long *)NULL,   \
                 0,            \
                 50,             \
                 0,                    \
                 0,                    \
                 (void (*)(void *))FIR16_init,\
                 (void (*)(void *))FIR16_calc}
    void FIR16_calc(void *);
    void FIR16_init(void *);
    #define FIR16_lp_30_Hz {\
      1210,6292662,6620329,7472275,8848501,10814544,13239331,16188400,19530679,2320
0634,\
      27263801,31523573,35914416,40501866,45089316,49611233,54067616,58327394,62325
034,66060535,\
                    69402826,72351909,74776711,76742770,78184549,79036512}
    typedef struct {
      float   x;
      float        in;
      float   ex;
      float   tanh;

      } TANH;
    typedef struct {
      float   IN2[6];
      float   W2[6];
      float   dW2[6];
      float   b2;
      float        OUT_SUMA;
      float   SCAL_SUM;
      TANH    TANHN2;
      float   OUT;
      float   OUT_1;
      float   dfi_v_e;
      float   izlaz;
      float   scalval_izlaza_prema_procesu;
      } NIZLAZNI;
    typedef struct {
      float   IN1[3];
      float   W1[3];
      float   dW1[3];
      float        OUT_SUMA1;
      TANH    TANHN1;
      float   OUT;
       float   dfi_v_e1;
      } NULAZNI;
    typedef struct {
      float   UREF;
      float        UACT;
      float  dUact;
      float  dUact_filt;
      float   UACT_old;
      float   Kv;
      float   K1;
      float   K2;
      float   RATE;
      float   ERROR;
      TANH    TAN_ERR2;
      TANH    TAN_ERR1[6];
      float   scalval;
      float   KC;
      float   d_v_1_ulaz;
```

```
   float   d_v_2_ulaz;
   float   d_v_3_ulaz;
   } ERROR;
   typedef struct {
   float   ka1;
   float  ka2;
   float  ka3;
   float  P_scal_int;
   float  P_scal_fl;
   float  Q_scal_int;
   float  Q_scal_fl;
   float  scal_iact;
   float  scal_P;
   float  scal_Q;
   float  Iu_scal;
   float  Iu_scal_OFFSET;
   float  delta;
   float  Kd;
   float  delta_filter;
   float  offest_filt;
   float  p;
          } prek;
typedef struct {
   float   in;
   float        in_old;
   float   KD;
   float   out;
   float  out_filt;
          } DER;
typedef struct{
   float  Ur;
   float  Uact;
   float  Iact;
   int      Uacti;
   int      Iacti;
   float  Iu_act;
   float  POW_ref_Q;
   float  POW_act_P;
   float  dPe;
   float  dPe_filt;
   float  Kd;
   float  Pe_old;
   float  delta_Pe;
   float  POW_act_Q;
   float  act_COS_FI;
   float  act_SIN_FI;
   float  act_COS_FI_filt;
   float  act_SIN_FI_filt;
   float  POW_act_P_filt;
   float  POW_act_Q_filt;
   float  over_U_limit;
   float  over_Iu_limit;
   float  DUTY_MANUAL;
   int     syncro;
   int     enable_syncro;
   int          AUTO_HAND;
   int    ENABLE_SYSTEM;
   int          POW_acti_Q;
   int          CONTROLL_TYPE;
   int     RECORDING;
   int     stabilization;
   int     kompenzacija;
   float  faktor_kompenzacije;
   float  D;
   } CONTROL;
#define NUMBER_SAMPLES  0x400
typedef int type_data;
typedef struct { type_data  *dlog_iptr1;
                     type_data  *dlog_iptr2;
```

```
                        type_data  *dlog_iptr3;
                        type_data  *dlog_iptr4;
                        type_data  *dlog_iptr5;
                        type_data  *dlog_iptr6;
                        type_data  *dlog_iptr7;
                        type_data  *dlog_iptr8;
                        type_data  *dlog_iptr9;
                        type_data  *dlog_iptr10;
                        type_data  *dlog_iptr11;
                        type_data  *dlog_iptr12;
                        type_data  *dlog_iptr13;
                        type_data  *dlog_iptr14;
                        type_data  *dlog_iptr15;
                        type_data  *dlog_iptr16;
                        type_data  *graph_ptr1;
                        type_data  *graph_ptr2;
                        type_data  *graph_ptr3;
                        type_data  *graph_ptr4;
                        type_data  *graph_ptr5;
                        type_data  *graph_ptr6;
                        type_data  *graph_ptr7;
                        type_data  *graph_ptr8;
                        type_data  *graph_ptr9;
                        type_data  *graph_ptr10;
                        type_data  *graph_ptr11;
                        type_data  *graph_ptr12;
                        type_data  *graph_ptr13;
                        type_data  *graph_ptr14;
                        type_data  *graph_ptr15;
                        type_data  *graph_ptr16;
                        int  dlog_cntr_max;
                        long  dl_buffer1_adr;
                        long  dl_buffer2_adr;
                        long  dl_buffer3_adr;
                        long  dl_buffer4_adr;
                        long  dl_buffer5_adr;
                        long  dl_buffer6_adr;
                        long  dl_buffer7_adr;
                        long  dl_buffer8_adr;
                        long  dl_buffer9_adr;
                        long  dl_buffer10_adr;
                        long  dl_buffer11_adr;
                        long  dl_buffer12_adr;
                        long  dl_buffer13_adr;
                        long  dl_buffer14_adr;
                        long  dl_buffer15_adr;
                        long  dl_buffer16_adr;
                        void  (*update)();
                        } DATALOG;
#define DATALOG_DEFAULTS {  (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                             (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                             (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                            (type_data *)0x00000300, \
                             (type_data *)0x00100000, \
                             (type_data *)0x00100400, \
                             (type_data *)0x00100800, \
                             (type_data *)0x00100C00, \
```

```
                                (type_data *)0x00101000, \
                                (type_data *)0x00101400, \
                                (type_data *)0x00101800, \
                                (type_data *)0x00101C00, \
                                (type_data *)0x00102000, \
                                (type_data *)0x00102400, \
                                (type_data *)0x00102800, \
                                (type_data *)0x00102C00, \
                                (type_data *)0x00103000, \
                                (type_data *)0x00103400, \
                                (type_data *)0x00103800, \
                                (type_data *)0x00103C00, \
                               NUMBER_SAMPLES, \
                               0x00100000, \
                               0x00100400, \
                               0x00100800, \
                               0x00100C00, \
                               0x00101000, \
                               0x00101400, \
                               0x00101800, \
                               0x00101C00, \
                               0x00102000, \
                               0x00102400, \
                               0x00102800, \
                               0x00102C00, \
                               0x00103000, \
                               0x00103400, \
                               0x00103800, \
                               0x00103C00, \
                              (void (*)(long))data_log_update }
void data_log_update(DATALOG *);
typedef struct{
   int i;
   int pocetak;
   int preskakanje_tocaka;
   int j;
   float REF;
   float REF_OLD;
   int s1;
   int s2;
   int s3;
   int s4;
   int s5;
   int s6;
   int s7;
   int s8;
   int s9;
   int s10;
   int s11;
   int s12;
   int s13;
   int s14;
   int s15;
   int s16;
   int s_scal_val[16];
   } SNIMANJE;
typedef struct{
   int IN_1;
   int IN_2;
   int OUT_1;
   int OUT_2;
   } DIG_In_Out;
extern PID            PID1, PID2;
extern Uab    UAB,IAB;
extern float duty,fil_pow, Ia,Ib,Ic,Id,Ie;
extern SNIMANJE SNIMANJE1;
extern CONTROL CP1;
extern TANH TANH1;
extern int brojac, filtulaz,filtulaz_IU,FILT_POM;
```

```
extern FIR16 fir,fir1,fir2,fir3,fir4,fir5,fir6,fir7;
extern int16 D;
extern DIG_In_Out DIG_IO;
extern IAE IAE1;
extern prek K;
extern DATALOG dlog1;
extern NIZLAZNI NIZ2;
extern NULAZNI  NUL[6];
extern ERROR ERROR1;
#endif
```

F2 Main.c

```
#include "Device.h"
         DATALOG dlog1 = DATALOG_DEFAULTS;
         #define FIR_ORDER   50
         #pragma DATA_SECTION(fir, "firfilt");
         FIR16  fir= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir1, "firfilt1");
         FIR16  fir1= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir2, "firfilt2");
         FIR16  fir2= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir3, "firfilt3");
         FIR16  fir3= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir4, "firfilt4");
         FIR16  fir4= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir5, "firfilt5");
         FIR16  fir5= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir6, "firfilt6");
         FIR16  fir6= FIR16_DEFAULTS;
         #pragma DATA_SECTION(fir7, "firfilt7");
         FIR16  fir7= FIR16_DEFAULTS;
         #pragma DATA_SECTION(dbuffer,"firldb");
         #pragma DATA_SECTION(dbuffer1,"firldb1");
         #pragma DATA_SECTION(dbuffer2,"firldb2");
         #pragma DATA_SECTION(dbuffer3,"firldb3");
         #pragma DATA_SECTION(dbuffer4,"firldb4");
         #pragma DATA_SECTION(dbuffer5,"firldb5");
         #pragma DATA_SECTION(dbuffer6,"firldb6");
         #pragma DATA_SECTION(dbuffer7,"firldb7");
         long dbuffer[(FIR_ORDER+2)/2];
         long dbuffer1[(FIR_ORDER+2)/2];
         long dbuffer2[(FIR_ORDER+2)/2];
         long dbuffer3[(FIR_ORDER+2)/2];
         long dbuffer4[(FIR_ORDER+2)/2];
         long dbuffer5[(FIR_ORDER+2)/2];
         long dbuffer6[(FIR_ORDER+2)/2];
         long dbuffer7[(FIR_ORDER+2)/2];
         long const coeff[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff1[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff2[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff3[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff4[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff5[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff6[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         long const coeff7[(FIR_ORDER+2)/2]= FIR16_lp_10_Hz;
         void main(void){
         int i;
         DIG_IO.IN_1=0;
         DIG_IO.IN_2=0;
         DIG_IO.OUT_1=0;
         DIG_IO.OUT_2=0;
                InitSysCtrl();
                InitGpio();
                InitPieVectTable();
                InitPieCtrl();
                       InitEv();
```

```
                        InitAdc();
            EALLOW;
      GpioMuxRegs.GPFMUX.all=0x0000;
GpioMuxRegs.GPFDIR.bit.GPIOF14=1;
      GpioMuxRegs.GPFDIR.bit.GPIOF8=1;
      GpioMuxRegs.GPFDIR.bit.GPIOF9=1;
      GpioMuxRegs.GPFDIR.bit.GPIOF12=0;
      GpioMuxRegs.GPFDIR.bit.GPIOF10 =0;
      EDIS;
            PID1.r = 0;
            PID1.ui1 = 0.0;
            PID1.e1 = 0.0;
            PID1.y1 = 0;
            PID1.y2 = 0;
            PID1.y3 = 0;
            PID1.y4 = 0;
            PID1.u_max = 5000;
            PID1.u_min=-5000;
            PID1.ui_max = 5000;
          PID1.Kp = 0;
          PID1.Ki = 1;
          PID1.Kd = 0;
            PID1.scalval_yf=1;
IAE1.in_1=0;
IAE1.in_2=0;
IAE1.sum_1=0;
IAE1.sum_2=0;
IAE1.K1_sum_1=1;
IAE1.sum_1_old=0;
IAE1.IAE1_1_old=0;
IAE1.sum_deriv_2=0;
IAE1.Kd_sum_2=1;
IAE1.sum_deriv_2_old=0;
IAE1.K2_sum_2=1;
IAE1.sum_2_old=0;
IAE1.IAE2_2_old=0;
IAE1.IAE=0;
IAE1.IAED=0;
IAE1.ENABLE=0;
            PID2.r = 0;
            PID2.ui1 = 0.0;
            PID2.e1 = 0.0;
            PID2.y1 = 0;
            PID2.y2 = 0;
            PID2.y3 = 0;
            PID2.y4 = 0;
            PID2.u_max = PWM_period;
            PID2.u_min=0;
            PID2.ui_max = PWM_period;
          PID2.Kp = 2;
          PID2.Ki = 0.01;
          PID2.Kd = 0;
          PID2.OFFSET=0;
            PID2.scalval_yf=1;
      duty=0;
      brojac=0;
            UAB.OFFSET=3.200684;
            IAB.OFFSET=3.200684;
            UAB.scalval=0.8233333;
            IAB.scalval=0.3891051;
K.ka1=-2;
K.ka2=0.57735;
K.ka3=1;
K.P_scal_int=327.67;
K.P_scal_fl=0.00390625;
K.Q_scal_int=327.67;
K.Q_scal_fl=-0.00390625;
K.scal_iact=1.5;
K.scal_P=1;
```

```
   K.scal_Q=1;
   K.Iu_scal=1;
   K.Iu_scal_OFFSET=0;
   K.delta=1;
   K.Kd=30;
   K.offest_filt=0;
   K.p=0;
   K.delta_filter=1;
              InitGptimer2(sample_period);
                 InitPwm(PWM_period);
              asm(" PUSH IER");
              asm(" POP  DBGIER");
              asm(" CLRC INTM, DBGM");
           fir.order=FIR_ORDER;
           fir.dbuffer_ptr=dbuffer;
           fir.coeff_ptr=(long *)coeff;
           fir.init(&fir);
           fir1.order=FIR_ORDER;
           fir1.dbuffer_ptr=dbuffer1;
           fir1.coeff_ptr=(long *)coeff1;
           fir1.init(&fir1);
           fir2.order=FIR_ORDER;
           fir2.dbuffer_ptr=dbuffer2;
           fir2.coeff_ptr=(long *)coeff2;
           fir2.init(&fir2);
           fir3.order=FIR_ORDER;
           fir3.dbuffer_ptr=dbuffer3;
           fir3.coeff_ptr=(long *)coeff3;
           fir3.init(&fir3);
           fir4.order=FIR_ORDER;
           fir4.dbuffer_ptr=dbuffer4;
           fir4.coeff_ptr=(long *)coeff4;
           fir4.init(&fir4);
           fir5.order=FIR_ORDER;
           fir5.dbuffer_ptr=dbuffer5;
           fir5.coeff_ptr=(long *)coeff5;
           fir5.init(&fir5);
           fir6.order=FIR_ORDER;
           fir6.dbuffer_ptr=dbuffer6;
           fir6.coeff_ptr=(long *)coeff6;
           fir6.init(&fir6);
           fir7.order=FIR_ORDER;
           fir7.dbuffer_ptr=dbuffer7;
           fir7.coeff_ptr=(long *)coeff7;
           fir7.init(&fir7);
         dlog1.dlog_iptr1 = &SNIMANJE1.s1;
         dlog1.dlog_iptr2 = &SNIMANJE1.s2;
         dlog1.dlog_iptr3 = &SNIMANJE1.s3;
         dlog1.dlog_iptr4 = &SNIMANJE1.s4;
         dlog1.dlog_iptr5 = &SNIMANJE1.s5;
         dlog1.dlog_iptr6 = &SNIMANJE1.s6;
         dlog1.dlog_iptr7 = &SNIMANJE1.s7;
         dlog1.dlog_iptr8 = &SNIMANJE1.s8;
         dlog1.dlog_iptr9 = &SNIMANJE1.s9;
         dlog1.dlog_iptr10 = &SNIMANJE1.s10;
         dlog1.dlog_iptr11 = &SNIMANJE1.s11;
         dlog1.dlog_iptr12 = &SNIMANJE1.s12;
         dlog1.dlog_iptr13 = &SNIMANJE1.s13;
         dlog1.dlog_iptr14 = &SNIMANJE1.s14;
         dlog1.dlog_iptr15 = &SNIMANJE1.s15;
         dlog1.dlog_iptr16 = &SNIMANJE1.s16;
for( i=0;i<6;i++){
          NUL[i].W1[0]=0.5;
          NUL[i].W1[1]=0.5;
          NUL[i].W1[2]=0.5;
          NUL[i].TANHN1.x=0;
          };
          ERROR1.Kv=-0.1;
          ERROR1.RATE=100;
```

```
          ERROR1.UACT_old=0;
          ERROR1.scalval=1;
          ERROR1.KC=100;
          ERROR1.K1=-90;
          ERROR1.K2=-10;
          NIZ2.W2[0]=0.5;
          NIZ2.W2[1]=0.5;
          NIZ2.W2[2]=0.5;
          NIZ2.W2[3]=0.5;
          NIZ2.W2[4]=0.5;
          NIZ2.W2[5]=0.5;
          NIZ2.SCAL_SUM=0.00001;
          NIZ2.b2=0;
          NIZ2.scalval_izlaza_prema_procesu=-5000;
          SNIMANJE1.pocetak=0;
          SNIMANJE1.i=0;
          SNIMANJE1.j=0;
          SNIMANJE1.REF=0.9;
          SNIMANJE1.preskakanje_tocaka=5;
          SNIMANJE1.s_scal_val[0]=1;
          SNIMANJE1.s_scal_val[1]=10000;
          SNIMANJE1.s_scal_val[2]=10000;
          SNIMANJE1.s_scal_val[3]=10000;
          SNIMANJE1.s_scal_val[4]=10000;
          SNIMANJE1.s_scal_val[5]=10000;
          SNIMANJE1.s_scal_val[6]=10;
          SNIMANJE1.s_scal_val[7]=1;
          SNIMANJE1.s_scal_val[8]=10;
          SNIMANJE1.s_scal_val[9]=1;
          SNIMANJE1.s_scal_val[10]=-10000;
          SNIMANJE1.s_scal_val[11]=10000;
          SNIMANJE1.s_scal_val[12]=10000;
          SNIMANJE1.s_scal_val[13]=100;
          SNIMANJE1.s_scal_val[14]=100;
          SNIMANJE1.s_scal_val[15]=1;
          CP1.Ur=0;
          CP1.POW_ref_Q=0;
          CP1.CONTROLL_TYPE=2;
          CP1.over_U_limit=1.2;
          CP1.ENABLE_SYSTEM=0;
          CP1.over_Iu_limit=5;
          CP1.DUTY_MANUAL=0;
          CP1.AUTO_HAND=0;
          CP1.syncro=0;
          CP1.enable_syncro=0;
          CP1.RECORDING=1;
          CP1.stabilization=0;
          CP1.Kd=30;
          CP1.kompenzacija=0;
          CP1.faktor_kompenzacije=0;
                    while(1)
                    {
                              asm(" NOP");
                    }
}
```

F3 Function.c

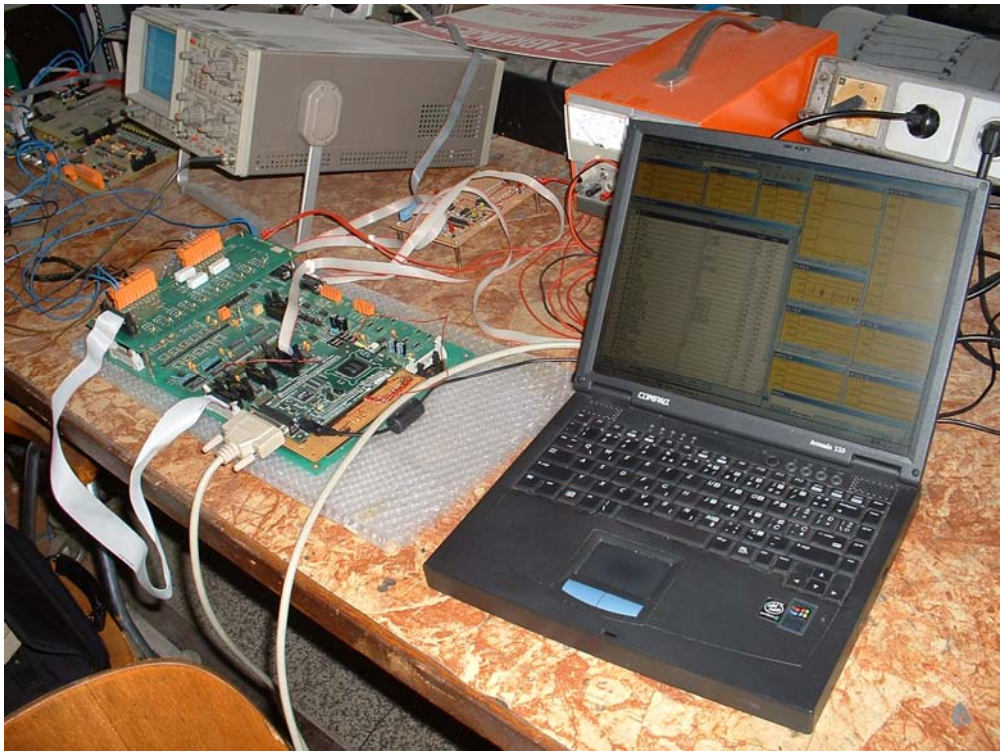
```
#include "Device.h"
void InitGptimer2(Uint16 period)
{
   EvaRegs.T2CON.all = 0x0000;
    EvaRegs.T2CNT = 0x0000;
    EvaRegs.T2PR = period;
    EvaRegs.T2CMPR = 0x0006;
   EvaRegs.GPTCONA.all = (EvaRegs.GPTCONA.all | 0x0620) & 0x0FF3;
   EvaRegs.T2CON.all = 0xD382;
}
void InitPwm(Uint16 period)
```

```
    {  EvaRegs.T1CON.all = 0x0000;
       EvaRegs.T1CNT = 0x0000;
       EvaRegs.T1PR = period;
       EvaRegs.GPTCONA.all = (EvaRegs.GPTCONA.all | 0x0000) & 0xF66C;
       EvaRegs.DBTCONA.all = 0x0000;
       EvaRegs.CMPR1 = 0x0000;
       EvaRegs.ACTRA.all = 0x0000;
       EvaRegs.COMCONA.all = 0x8220;
       EvaRegs.T1CON.all = 0xD640;
    }
    void PidCtrl1(PID *x)
    {  float32 e, up, ui, ud;
       e = x->r - (x->yf*x->scalval_yf);
       up = x->Kp * e;
       ui = x->ui1 + (x->Ki * (e + x->e1));
       if(ui > x->ui_max) ui = x->ui_max;
       else if(ui < -(x->ui_max)) ui = -(x->ui_max);
       ud = x->Kd * (e - x->e1);
       x->e1 = e;
       x->ui1 = ui;
       x->u = up + ui + ud;
       if(x->u > x->u_max) x->u = x->u_max;
       else if(x->u < x->u_min) x->u = x->u_min;
    }
    void iae(IAE *IAE){
             float32 der, sum_1, int1 int2, sum_2;
             sum_1=IAE->in_1-IAE->in_2;
             if(sum_1<0){sum_1=-sum_1;}
             int1=IAE->IAE1_1_old+IAE->K1_sum_1*(sum_1+IAE->sum_1_old);
             IAE->sum_1_old=sum_1;
             IAE->IAE1_1_old=int1;
             IAE->IAE=int1;
             sum_2=IAE->in_1-IAE->in_2;
             der=IAE->Kd_sum_2*(sum_2-IAE->sum_2_old);
             IAE->sum_2_old=sum_2;
             if(der<0){der=-der;}
             int2=IAE->IAE2_2_old +IAE->K2_sum_2*(der+IAE->sum_deriv_2);
             IAE->sum_deriv_2=der;
             IAE->IAE2_2_old=int2;
             IAE->IAED=int2;}
    void tansig (TANH *TANH1){
     (TANH1->in)=(TANH1->x)*(-2);
    (TANH1->ex)=1+(TANH1->in)+(0.5*(TANH1->in)*(TANH1-
>in))+(0.16666666666667*(TANH1->in)*(TANH1->in)*(TANH1->in))+
       (0.041666666666666666667*(TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1->in))+
         (0.008333333333333333333*(TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1-
>in)*(TANH1->in))+
         (0.0013888888888888888889*(TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1-
>in)*(TANH1->in)*(TANH1->in))+
         (0.00019841269841269841269*(TANH1->in)*(TANH1->in)*(TANH1->in)*(TANH1-
>in)*(TANH1->in)*(TANH1->in)*(TANH1->in))
       TANH1->tanh=(2/(1+(TANH1->ex)))-1;
    }
    void data_log_update(DATALOG *p)
    {
       *p->graph_ptr1 = *p->dlog_iptr1;
       p->graph_ptr1 = p->graph_ptr1+1;
         if   (p->graph_ptr1   ==   (type_data   *)(p->dl_buffer1_adr   +   sizeof(*p-
>dlog_iptr1)*p->dlog_cntr_max))
       p->graph_ptr1 = (type_data *)p->dl_buffer1_adr;
       *p->graph_ptr2 = *p->dlog_iptr2;
       p->graph_ptr2 = p->graph_ptr2+1;
         if   (p->graph_ptr2   ==   (type_data   *)(p->dl_buffer2_adr   +   sizeof(*p-
>dlog_iptr2)*p->dlog_cntr_max))
       p->graph_ptr2 = (type_data *)p->dl_buffer2_adr;
       *p->graph_ptr3 = *p->dlog_iptr3;
       p->graph_ptr3 = p->graph_ptr3+1;
         if   (p->graph_ptr3   ==   (type_data   *)(p->dl_buffer3_adr   +   sizeof(*p-
>dlog_iptr3)*p->dlog_cntr_max))
```

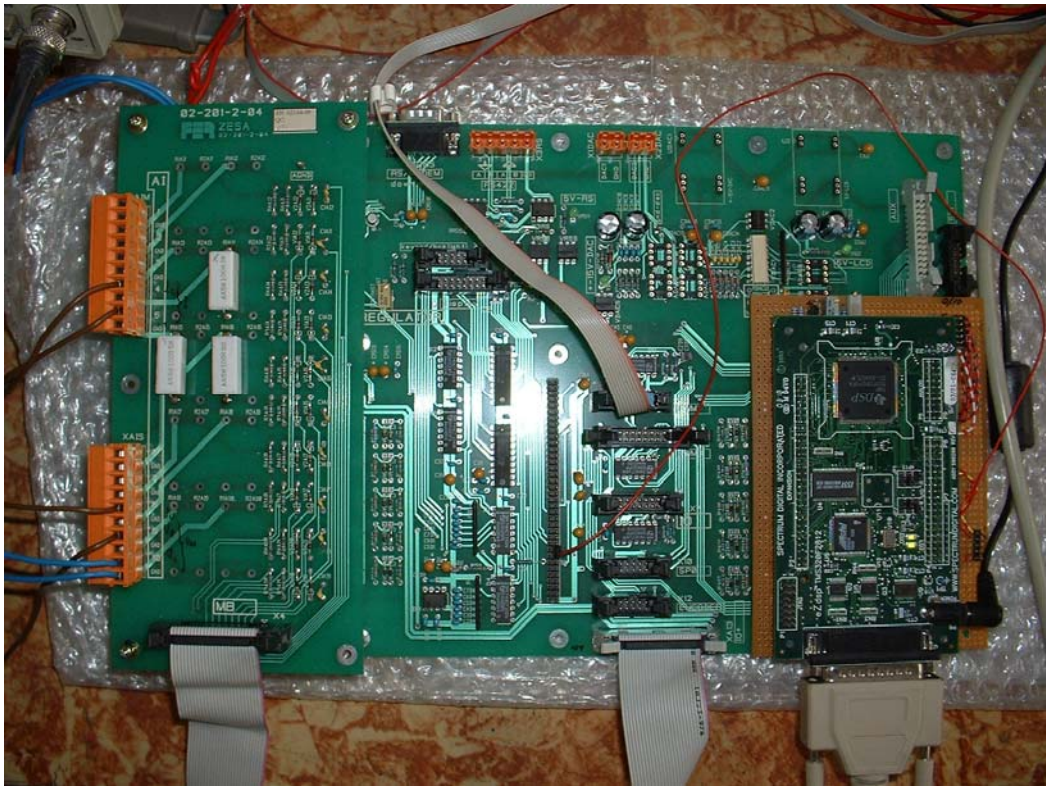
```
      p->graph_ptr3 = (type_data *)p->dl_buffer3_adr;
      *p->graph_ptr4 = *p->dlog_iptr4;
      p->graph_ptr4 = p->graph_ptr4+1;
       if   (p->graph_ptr4   ==   (type_data   *)(p->dl_buffer4_adr   +   sizeof(*p-
>dlog_iptr4)*p->dlog_cntr_max))
      p->graph_ptr4 = (type_data *)p->dl_buffer4_adr;
      *p->graph_ptr5 = *p->dlog_iptr5;
      p->graph_ptr5 = p->graph_ptr5+1;
       if   (p->graph_ptr5   ==   (type_data   *)(p->dl_buffer5_adr   +   sizeof(*p-
>dlog_iptr5)*p->dlog_cntr_max))
      p->graph_ptr5 = (type_data *)p->dl_buffer5_adr;
      *p->graph_ptr6 = *p->dlog_iptr6;
      p->graph_ptr6 = p->graph_ptr6+1;
       if   (p->graph_ptr6   ==   (type_data   *)(p->dl_buffer6_adr   +   sizeof(*p-
>dlog_iptr6)*p->dlog_cntr_max))
      p->graph_ptr6 = (type_data *)p->dl_buffer6_adr;
      *p->graph_ptr7 = *p->dlog_iptr7;
      p->graph_ptr7 = p->graph_ptr7+1;
       if   (p->graph_ptr7   ==   (type_data   *)(p->dl_buffer7_adr   +   sizeof(*p-
>dlog_iptr7)*p->dlog_cntr_max))
      p->graph_ptr7 = (type_data *)p->dl_buffer7_adr;
      *p->graph_ptr8 = *p->dlog_iptr8;
      p->graph_ptr8 = p->graph_ptr8+1;
       if   (p->graph_ptr8   ==   (type_data   *)(p->dl_buffer8_adr   +   sizeof(*p-
>dlog_iptr8)*p->dlog_cntr_max))
      p->graph_ptr8 = (type_data *)p->dl_buffer8_adr;
      *p->graph_ptr9 = *p->dlog_iptr9;
      p->graph_ptr9 = p->graph_ptr9+1;
       if   (p->graph_ptr9   ==   (type_data   *)(p->dl_buffer9_adr   +   sizeof(*p-
>dlog_iptr9)*p->dlog_cntr_max))
      p->graph_ptr9 = (type_data *)p->dl_buffer9_adr;
      *p->graph_ptr10 = *p->dlog_iptr10;
      p->graph_ptr10 = p->graph_ptr10+1;
       if   (p->graph_ptr10   ==   (type_data   *)(p->dl_buffer10_adr   +   sizeof(*p-
>dlog_iptr10)*p->dlog_cntr_max))
      p->graph_ptr10 = (type_data *)p->dl_buffer10_adr;
      *p->graph_ptr11 = *p->dlog_iptr11;
      p->graph_ptr11 = p->graph_ptr11+1;
       if   (p->graph_ptr11   ==   (type_data   *)(p->dl_buffer11_adr   +   sizeof(*p-
>dlog_iptr11)*p->dlog_cntr_max))
      p->graph_ptr11 = (type_data *)p->dl_buffer11_adr;
      *p->graph_ptr12 = *p->dlog_iptr12;
      p->graph_ptr12 = p->graph_ptr12+1;
       if   (p->graph_ptr12   ==   (type_data   *)(p->dl_buffer12_adr   +   sizeof(*p-
>dlog_iptr12)*p->dlog_cntr_max))
      p->graph_ptr12 = (type_data *)p->dl_buffer12_adr;
      *p->graph_ptr13 = *p->dlog_iptr13;
      p->graph_ptr13 = p->graph_ptr13+1;
       if   (p->graph_ptr13   ==   (type_data   *)(p->dl_buffer13_adr   +   sizeof(*p-
>dlog_iptr13)*p->dlog_cntr_max))
      p->graph_ptr13 = (type_data *)p->dl_buffer13_adr;
      *p->graph_ptr14 = *p->dlog_iptr14;
      p->graph_ptr14 = p->graph_ptr14+1;
       if   (p->graph_ptr14   ==   (type_data   *)(p->dl_buffer14_adr   +   sizeof(*p-
>dlog_iptr14)*p->dlog_cntr_max))
      p->graph_ptr14 = (type_data *)p->dl_buffer14_adr;
      *p->graph_ptr15 = *p->dlog_iptr15;
      p->graph_ptr15 = p->graph_ptr15+1;
       if   (p->graph_ptr15   ==   (type_data   *)(p->dl_buffer15_adr   +   sizeof(*p-
>dlog_iptr15)*p->dlog_cntr_max))
      p->graph_ptr15 = (type_data *)p->dl_buffer15_adr;
      *p->graph_ptr16 = *p->dlog_iptr16;
      p->graph_ptr16 = p->graph_ptr16+1;
       if   (p->graph_ptr16   ==   (type_data   *)(p->dl_buffer16_adr   +   sizeof(*p-
>dlog_iptr16)*p->dlog_cntr_max))
      p->graph_ptr16 = (type_data *)p->dl_buffer16_adr;}
```

DODATAK G

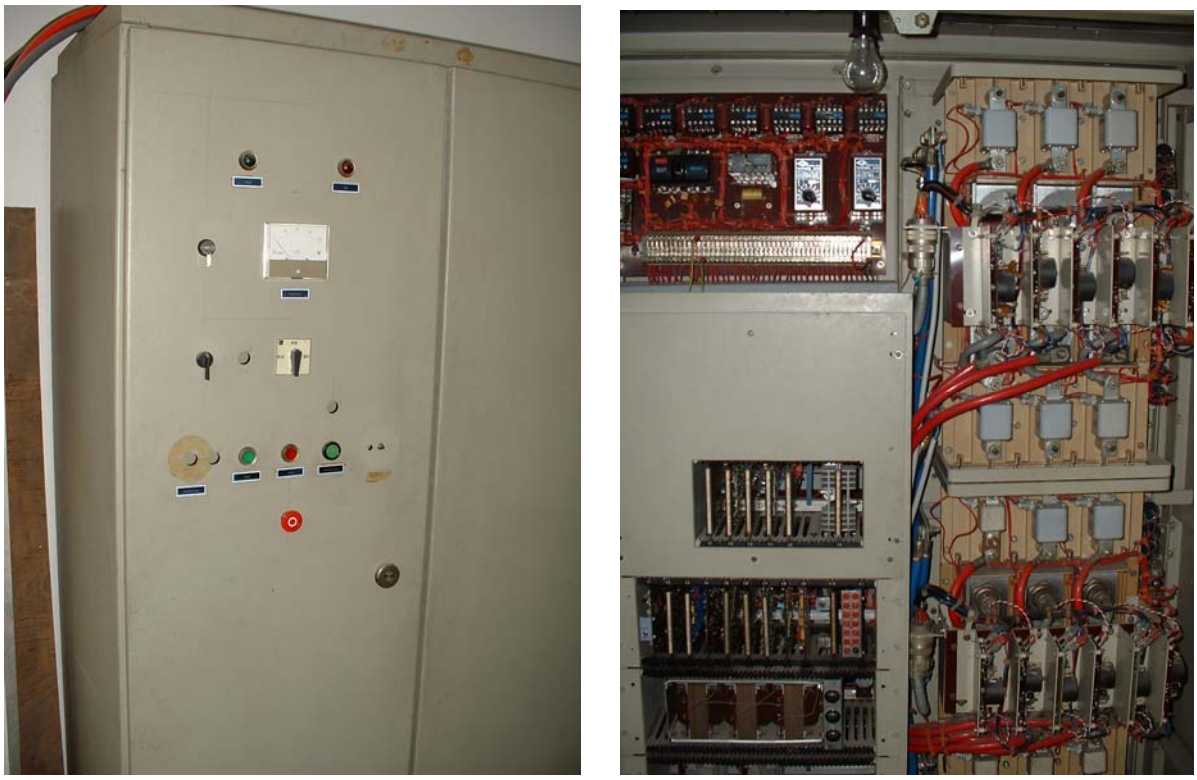
Parametri i slike sustava



Slika G1. Maketa sustava

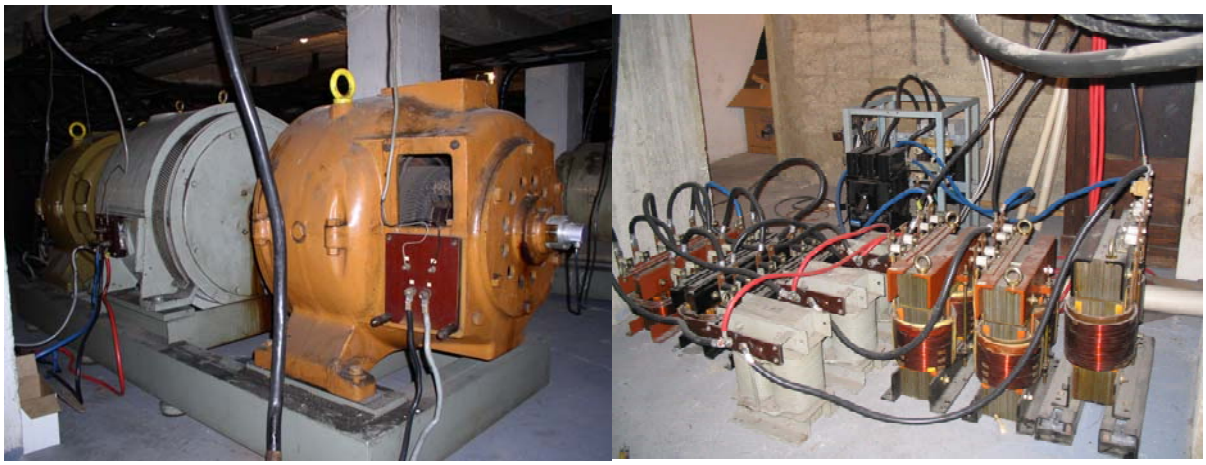


Slika G.2. DSP s prilagodnim pločicama



Slika G.3. Tiristorski usmjerivač

Slika G.4. Spojno polje generatora



Slika G.5. Generator i prigušnice kao spojni vodovi

Tablica G.1. Nazivni podaci sinkronog generatora

Nazivni napon	400 V
Nazivna struja	120 A
Nazivna snaga	83 kVA
Nazivna frekvencija	50 Hz
Nazivna brzina	600 r/min
Faktor snage	0,8
Napon uzbude	100 V
Struja uzbude	11.8 A

Tablica G.2. Nazivni podaci istosmjernih motora

Nazivni napon	220 V
Nazivna struja	192 A
Nazivna snaga	42,24 kW
Nazivna brzina	600 r/min

Tablica G.3. Podaci IGBT pretvarača

Proizvođač	FER-ZESA
Ulazni napon napajanja	3*380 V
Prijenosni omjer transformatora	380/125 V
Nazivna struja ispravljača	50 A
Nazivna izlazna struja pretvarača	50 A
Kapacitet elektrolitskih kondenzatora	4700 μF
Limit zaštite od prenapona	200 V
Otpor u krugu zaštite od prenapona	10 Ω

Tablica G.4. Nazivni podaci prigušnica L1

Induktivitet	3.5 mH
Nazivna struja	106 A

Tablica G.5. Podešenja mjernih članova

Mjerenje napona generatora	Mjerni opseg mjernog člana u odnosu na nominalni napon generatora	±150 %
Mjerenje struje generatora	Mjerni opseg mjernog člana u odnosu na nominalnu struju generatora	±400 %
Mjerenje struje uzbude	Mjerni opseg mjernog člana u odnosu na nominalnu struju uzbude	±400 %

SAŽETAK

UPRAVLJANJE SUSTAVOM UZBUDE SINKRONOG GENERATORA UPOTREBOM NEURONSKE MREŽE

U klasičnoj regulacijskoj strukturi sustav uzbude sinkronog generatora uključuje regulator struje uzbude, regulator napona, regulator jalove snage i stabilizator oscilacija radne snage. Upotrijebljeni regulatori imaju proporcionalno ili proporcionalno integralno djelovanje. Stabilizator oscilacija radne snage obično je realiziran metodom fazne kompenzacije.

Primijenjena je dinamička neuronska mreža s modificiranom kriterijskom funkcijom u sustavu uzbude sinkronog generatora. Umjesto proporcionalno integralnog regulatora napona i stabilizatora realiziranog po metodi fazne kompenzacije, u sustav uzbude uključena je regulator napona zasnovan na dinamičkoj neuronskoj mreži s stabilizirajućim efektom u kriterijskoj funkciji. Realizirana neuronska mreže ima tri ulaza i jedan izlaz. Ulazi su referentna vrijednost napona, stvarna vrijednost napona generatora i prošlo stanje izlaza iz mreže, dok je izlaz iz mreže referentna vrijednost struje uzbude koja se vodi u regulator struje uzbude. Modificirana kriterijska funkcija koristi četiri signala za izračun greške neuronske mreže. Ulazi u kriterijsku funkciju su razlika stvarne i referentne vrijednosti napona generatora, derivacija stvarne vrijednosti napona, devijacija radne snage derivacija radne snage.

Sustav uzbude generatora se sastoji od generatora spojenog na mrežu preko jednog prijenosna voda i transformatora, energetskog kruga uzbude i digitalnog sustava s pripadnim sklopovima povratne veze. Za svaki dio sustava napravljen je matematički model. Pojedini dijelovi su spojeni u jedinstveni model sustava uzbude. Načinjen je simulacijski model sustava regulacije uzbude generatora u programskom paketu *Matlab/Simulink*.

Neuronska mreža s stabilizirajućim efektom u kriterijskoj funkciji implementirana je u sustav s digitalnim procesorom za obradu signala. Ispitivanje je obavljeno u laboratorijskim uvjetima rada sa sinkronim generatorom snage 83 kVA koji je preko jednog voda i transformatora spojen na mrežu. Parametri modificirane kriterijske funkcije podešeni su numeričkim pokazateljima kvalitete odziva napona i radne snage generatora te simulacijom.

Uspoređeno je djelovanje neuronske mreže sa i bez stabilizirajućeg efekta u kriterijskoj funkciji s djelovanjem klasičnog proporcionalno integralnog regulatora napona te je u

konačnici uspoređeno i s djelovanjem neizrazitog regulatora s neizrazitim stabilizatorom oscilacija radne snage. Usporedba je obavljena u dinamičkim uvjetima kod udarne promjene postavne veličine napona.. Ocijenjene su regulacijske karakteristike dinamičke neuronske mreže s stabilizirajućim efektom u kriterijskoj funkciji koje pokazuju poboljšanje u odnosu na proporcionalno integralni regulator i neizraziti regulator napona s neizrazitim stabilizatorom oscilacija radne snage.

Ključne riječi: sinkroni generator, uzbuda generatora, neuronske mreže, stabilizacija oscilacija snage.

SUMMARY

NEURAL NETWORK BASED EXCITATION CONTROL OF SYNCHRONOUS GENERATOR

Classical excitation control structure includes excitation current controller, voltage controller, reactive power controller and power system stabilizer. These controllers have proportional or proportional and integral behaviour. The power system stabilizer based on phase compensation method is classical solution.

In this work neural network based excitation control of synchronous generator is presented. Instead of PI voltage controller and power system stabilizer (based on the phase compensation method) dynamic neural network with modified error function is used. Implemented neural network have tree inputs and one output. Input in neural network is referent value of generator voltage, terminal voltage and previous output of neural network Modified error function uses following signal in error calculation difference between terminal voltage and referent generator voltage, derivation of terminal voltage, active power deviation and derivation of active power.

Synchronous generator excitation system includes synchronous generator connected to the AC grid via one transmission lines and transformer, excitation fed IGBT converter and digital control system based on digital signal processor. The mathematical model of synchronous generator excitation system is given. Based on this mathematical model an appropriated simulation model in software package *Matlab/Simulink* was developed.

Neural network with modified error function is implemented in system based on digital signal processor. The experimental verification is made on the laboratory setup at the Faculty of electrical engineering and computing, Department of electric machines, drives and automation. The laboratory setup consist of synchronous generator (83 kVA) connected to the AC grid via one parallel transmission lines and transformer and digital control system. Parameters of modified error function are tuned based on numerical criteria quality of voltage and active power response.

Control characteristics neural network with modified error function is compared with PI voltage controller. The comparison is made for voltage reference step change. Measured dynamic and static system performances match the calculated and simulated ones. At the end

behaviour of neural network with modified error function is compared with fuzzy voltage stabilizing controller. Control characteristics of are evaluated and show an improvement against PI voltage controller and fuzzy voltage control.

Keywords: synchronous generator, generator excitation, neural network, power system stabilization

ŽIVOTOPIS

Rođen sam u Rijeci 29. srpnja 1977. godine gdje sam završio srednju elektrotehničku školu "Rijeka". Godine 1996. upisao sam Fakultet elektrotehnike i računarstva u Zagrebu. Diplomirao sam na smjeru Elektrostrojarstvo i automatizacija 2001. godine.

Godine 2001. upisao sam poslijediplomski studij Fakulteta elektrotehnike i računarstva, smjer Elektrostrojarstvo na području digitalne regulacije sinkronog generatora. Autor sam četiri rada na znanstvenostručnim skupovima s međunarodnom recenzijom. Sudjelujem na projektima u zemlji i inozemstvu na području regulacije, upravljanja i automatizacije industrijskih postrojenja.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Neven Bulić

UPRAVLJANJE SUSTAVOM UZBUDE SINKRONOG GENERATORA UPOTREBOM NEURONSKE MREŽE

Magistarski rad

Zagreb, 2005.

Magistarski rad je izrađen na Zavodu za elektrostrojarstvo i automatizaciju
FAKULTETA ELEKTROTEHNIKE I RAČUNARSTVA Zagreb.

Mentor: Prof.dr.sc. Gorislav Erceg

Magistarski rad ima 174 lista.

Povjerenstvo za ocjenu magistarskog rada:

1. Prof.dr.sc. Ivan Petrović - predsjednik
2. Prof.dr.sc. Gorislav Erceg - mentor
3. Doc.dr.sc. Livio Šušnjić – Tehnički fakultet Sveučilišta u Rijeci

Povjerenstvo za obranu magistarskog rada:

1. Prof.dr.sc. Ivan Petrović - predsjednik
2. Prof.dr.sc. Gorislav Erceg - mentor
3. Doc.dr.sc. Livio Šušnjić – Tehnički fakultet Sveučilišta u Rijeci
4. Prof.dr.sc. Zlatko Maljković - zamjenik

Datum obrane: 25.11.2005.

Zahvaljujem prof.dr.sc. Gorislavu Ercegu na savjetima i pomoći oko izrade magistarskog rada.

Zahvaljujem mr.sc. Marinku Miletiću na suradnji prilikom izrade ovog rada.

Također se zahvaljujem kolegama sa Zavoda za elektrostrojarstvo i automatizaciju, posebno mr.sc. Damiru Sumini i mr.sc. Alenu Poljuganu, na pomoći pri izradi ovog rada.

Posebnu zahvalu dugujem svojoj obitelji na razumijevanju prilikom izrade ovog rada.

Sadržaj:

1 UVOD 1

2 KLASIČNI SUSTAV REGULACIJE UZBUDE SINKRONOG GENERATORA ... 2

3 UMJETNE NEURONSKE MREŽE 4

3.1 Osnovni modeli umjetnih neurona 4
3.2 Tipovi aktivacijskih funkcija 5
3.3 Arhitektura neuronskih mreža 7
3.3.1 Jednoslojne unaprijedne mreže 7
3.3.2 Višeslojne unaprijedne mreže 7
3.3.3 Dinamičke mreže 8
3.3.4 Rešetkasta struktura (*Lattice*) 9
3.4 Algoritmi učenja neuronskih mreža 9
3.4.1 MLP neuronske mreže i algoritam povratnog prostiranja 10
3.4.2 Osnovni pojmovi i definicije 10
3.4.3 Algoritam povratnog prostiranja 11
3.4.4 Nelinearna aktivacijska funkcija 16
3.4.5 Korak učenja 17
3.4.6 Kriteriji zaustavljanja 17
3.4.7 Inicijalizacija mreže 17

4 PREGLED SUSTAVA REGULACIJU UZBUDE S VIŠESLOJNIM UNAPRIJEDNIM NEURONSKIM MREŽAMA 19

4.1 Regulator napona zasnovan na jednom perceptronu [3],[29] 19
4.2 Regulator napona zasnovan na višeslojnoj neuronskoj mreži 22
4.3 Simulacija ponašanja paralelnog rada generatora s regulatorima uzbude zasnovanim na neuronskoj mreži 24

5 STRUKTURA SUSTAVA NEURONSKOG UPRAVLJNJA UZBUDOM SINKRONOG GENERATORA 26

5.1 Struktura regulatora napona sinkronog generatora zasnovanog na neuronskoj mreži 26
5.2 Algoritam učenja 27
5.3 Modificirana kriterijska funkcije 28

6 SIMULACIJSKI MODEL KOMPONENATA I SUSTAVA REGULACIJE UZBUDE GENERATORA 32

6.1 Simulacijski model čopera 34
6.2 Simulacijski model pogona sinkronog generatora 36
6.3 Simulacijski model sinkronog generatora 36
6.4 Simulacijski model kruga regulacije struje uzbude sinkronog generatora 44
6.5 Simulacijski model kruga regulacije napona sinkronog generatora 45
6.6 Kompenzacija napona po jalovoj snazi 47
6.7 Simulacijski model regulatora napona sinkronog generatora zasnovanog na neuronskoj mreži 47

7 IZVEDBE I IMPLEMENTACIJE ALGORITAMA NEURONSKOG UPRAVLJANJA UZBUDOM GENERATORA 52

7.1 PID regulator 52
7.2 Clarkeove transformacije 54

7.3 FIR filtar 55
7.4 Neuronska mreža 56
7.5 Nelinearna aktivacijska funkcija 59
7.6 Zaštitne funkcije 60
7.7 Funkcije snimanja podataka u stvarnom vremenu 60
7.8 Dijagram toka izvođenja programa u procesoru 61
7.9 IMPLEMENTIRANA REGULACIJSKA STRUKTURA 62
8 PODEŠAVANJE PARAMETARA SUSTAVA UZBUDE GENERATORA 65
8.1 Podešavanje parametara regulatora napona zasnovanog na neuronskoj mreži 65
8.2 Podešavanje parametara kriterijske funkcije za ostvarivanje stabilizirajućeg efekta 72
9 SIMULACIJSKI I EKSPERIMENTALNI REZULTATI DJELOVANJA SUSTAVA REGULACIJE UZBUDE SINKRONOG GENERATORA 76
10 USPOREBE DJELOVANJA SUSTAVA NEURONSKE I NEIZRAZITE REGULACIJE UZBUDE GENERATORA 96
11 ZAKLJUČAK 105
12 POPIS OZNAKA 107
13 LITERATURA 109
DODATAK A 111
DODATAK B 113
DODATAK C 129
DODATAK D 134
DODATAK E 150
DODATAK F 155
DODATAK G 166
SAŽETAK 170
SUMMARY 172
ŽIVOTOPIS 174