Samopodesivi PID regulator zasnovan na adaptivnom Kalmanovom filtru i relejnom članu

Kučiš, Tomislav

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Tomislav Kučiš

ZAGREB, 2011.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

prof. dr. sc. Davor Zorc **Tomislav Kučiš**

ZAGREB, 2011.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **TOMISLAV KUČIŠ** Mat. br.: 0035155958

Naslov rada na hrvatskom jeziku: **SAMOPODESIVI PID REGULATOR ZASNOVAN NA ADAPTIVNOM KALMANOVOM FILTRU I RELEJNOM ČLANU**

Naslov rada na engleskom jeziku: **AUTOTUNING PID CONTROLLER BASED ON ADAPTIVE KALMAN FILTER AND RELAY TERM**

Opis zadatka:

U radu pristupnik mora uraditi sljedeće:

1. Proučiti releventnu literaturu vezanu uz sintezu PID regulatora dovođenjem sustava u stanje prinudnih oscilacija primjenom relejnog člana u povratnoj vezi. Također treba dokumentirati različite metode određivanja kritične frekvencije sustava u režimu prinudnih oscilacija, te metodu opisne funkcije (metodu harmoničke ravnoteže) za određivanje nadomjesnog pojačanja relejnog člana na frekvenciji prinudnih oscilacija (tzv. kritičnog pojačanja).

2. Postaviti pojednostavljeni model prinudnih oscilacija sustava s relejnim članom u povratnoj vezi, te temeljem tog modela projektirati adaptivni Kalmanov filtar za određivanje frekvencije prinudnih oscilacija u realnom vremenu.

3. Provesti sintezu PID regulatora prema Takahashiu dovođenjem sustava na rub stabilnosti pomoću relejnog člana. Također je potrebno analizirati vladanje regulacijskog sustava i adaptivnog Kalmanovog filtra simulacijama na računalu.

4. Ispitati vladanje samopodesivog PID regulatora eksperimentalnim putem za slučaj regulacije temperature grijalice s puhalom. Samopodesivi PID regulator treba implementirati na odgovarajućem upravljačkom računalu (npr. osobnom računalu opremljenom akvizicijskom karticom ili na programibilnom logičkom kontroleru opremljenom modulima s analognim ulazima i izlazima).

Također, pristupnik treba u radu navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: **11. ožujka 2010.**

Rok predaje rada: Ožujak 2011.

Zadatak zadao: Prof. dr. sc. Davor Zorc

Predsjednik Povjerenstva: Prof. dr. sc. Franjo Cajner

Zahvala

Zahvaljujem se prof. dr. sc. Davoru Zorcu
na mentorstvu u ovome radu.

Najviše se zahvaljujem asistentu dr. sc.Danijelu Pavkoviću
na suradnji, savjetima i pomoći prilikom izrade rada.

Također se zahvaljujem kolegama sa Zavoda za
strojarsku automatiku i svima koji su mi
na bilo koji način pomogli.

Posebnu zahvalu dugujem svojoj obitelji i prijateljima
na razumijevanju i podršci.

Sadržaj

Sažetak

Regulacija temperature je klasičan regulacijski problem, koji se vrlo često javlja u procesnoj tehnici. Ovakav projekt zahtijeva potrebna znanja iz područja kao što su modeliranje, simulacija i regulacija. Do parametara PI/PID regulatora dolazi se Takahashievom metodom dovođenja na rub stabilnosti, što se provodi pomoću dvopoložajnog regulatora (relejnog člana). Iz amplitude i frekvencije rubnih oscilacija, može se odrediti kritično pojačanje i period oscilacija, pomoću kojih se može provesti sinteza PI/PID regulatora prema Ziegler-Nicholsovim (Takahashievim) preporukama. Tijekom implementacije algoritma automatskog podešenja (auto-tuning) PI/PID regulatora posebna pažnja posvećuje se učincima šuma mjerenja (smetnji). Predloženi koncept auto-tuning regulatora ispitan je simulacijama na računalu i eksperimentalno na laboratorijskoj maketi grijalice s puhalom (toplinske staze).

Ključne riječi: *auto-tuning PID regulatora, dvopoložajni regulator, adaptivni Kalmanov filtar, estimator frekvencije, estimator amplitude*

Popis slika i tablica

Popis oznaka

Oznaka	Opis	Jedinica
x	Regulirana veličina (opći slučaj)	-
w	Referentna veličina (opći slučaj)	-
e	Regulacijsko odstupanje (opći slučaj)	-
y	Postavna veličina (opći slučaj)	-
z	Poremećajna veličina (opći slučaj)	-
K_u	Kritično pojačanje regulatora	-
T_u	Period oscilacija	s
T	Vrijeme uzrokovanja	s
u	Upravljačka veličina	V
$U_{\max}$	Maksimalna vrijednost upravljačkog signala	V
a	Amplituda	-
x_u	Ulaz nelinearnog člana	-
x_i	Odzivna funkcija	-
N	Opisna funkcija	-
$x(k)$	Vektor varijabli stanja	-
$y(k)$	Vektor izlaznih varijabli	-
$v(k-1)$	Vektor stohastičkih perturbacija	-
$e(k)$	Vektor šuma mjerenja	-
$u(k-1)$	Vektor ulaznih varijabli	-
$F(k-1)$	Matrica sustava	-
$H(k-1)$	Izlazna matrica	-
$G(k-1)$	Ulazna matrica	-
$\Omega(k-1)$	Matrica perturbacija stanja	-
$Q(k)$	Matrica kovarijanci pertubacija stanja	-
$R(k)$	Matrica kovarijanci šuma mjerenja	-
$x(k \mid k-1)$	Inicijalna procjena varijabli stanja na osnovi determinističkog dijela modela procesa	-
$x(k \mid k)$	Konačna procjena varijabli stanja korigirana na osnovi mjerenja	-

$\varepsilon(k \mid k-1)$	inicijalna procjena predikcijske pogreške Kalmanovog filtra	-
$K(k)$	matrica pojačanja Kalmanovog filtra	-
$P(k \mid k-1)$	inicijalna procjena matrica kovarijanci pogreški procjene stanja	-
$P(k \mid k)$	korigirana procjena matrica kovarijanci pogreški procjene stanja	-
Ω	Rezonantna frekvencija	Rad / s
Ω	Estimirana vrijednost rezonantne frekvencije	Rad / s
G_{PP}	Prijenosna funkcija pojasno propusnog filtra	-
y_{PP}	Izlaz pojasno propusnog filtra	-
G_{NPA}	Prijenosna funkcija anti-aliasing nisko propusnog filtra	-
y_{NPA}	Izlaz anti-aliasing nisko propusnog filtra	-
G_{NP}	Prijenosna funkcija nisko propusnog filtra	-
y_{NP}	Izlaz nisko propusnog filtra	-
A	Estimirana vrijednost amplitude	-
K_R	Koeficijent pojačanja regulatora	-
T_I	Integracijska vremenska konstanta	s
T_D	Derivacijska vremenska konstanta	s

Izjava

Izjavljujem da sam ovaj rad izradio samostalno služeći se stečenim znanjem i navedenom literaturom.

1. Uvod

Regulacija temperature je klasičan regulacijski problem, koji se vrlo često javlja u procesnoj tehnici, te zahtijeva potrebna znanja iz područja kao što su modeliranje, simulacija i regulacija. Kod mnogih industrijskih procesa može se provesti sinteza PI/PID regulatora prema Ziegler-Nicholsu (Takahashiu). Posebne pogodnosti takvog načina podešavanja parametara regulatora su vrlo jednostavni matematički izrazi za parametre, te značajna brzina odziva regulacijskog kruga na poremećajnu veličinu. Nedostaci takvog načina podešavanja parametara regulatora su potencijalno veliki iznos nadvišenja i moguća oscilatornost odziva regulacijskog kruga. Postoje dvije metode sinteze PI/PID regulatora prema Takahashievim preporukama:

1. metoda prijelazne karakteristike
2. metoda dovođenja na rub stabilnosti.

U ovom zadatku je korištena metoda dovođenja na rub stabilnosti. Za auto tuning dio korišten je adaptivni Kalmanov filtar za estimaciju frekvencije i amplitude prinudnih oscilacija. Simulacija je provedena koristeći programski paket Matlab/Simulink, dok je eksperimentalna provjera provedena na nastavnoj maketi toplinske staze sa regulacijom protoka i temperature. Pritom je realizacija regulacijskih i auto tuning algoritama provedena u programskom jeziku C, i to kao C-mex datoteka (Matlab/Simulink), odnosno source koda za Borland C++ 3.1 za potrebe eksperimentalne provjere.

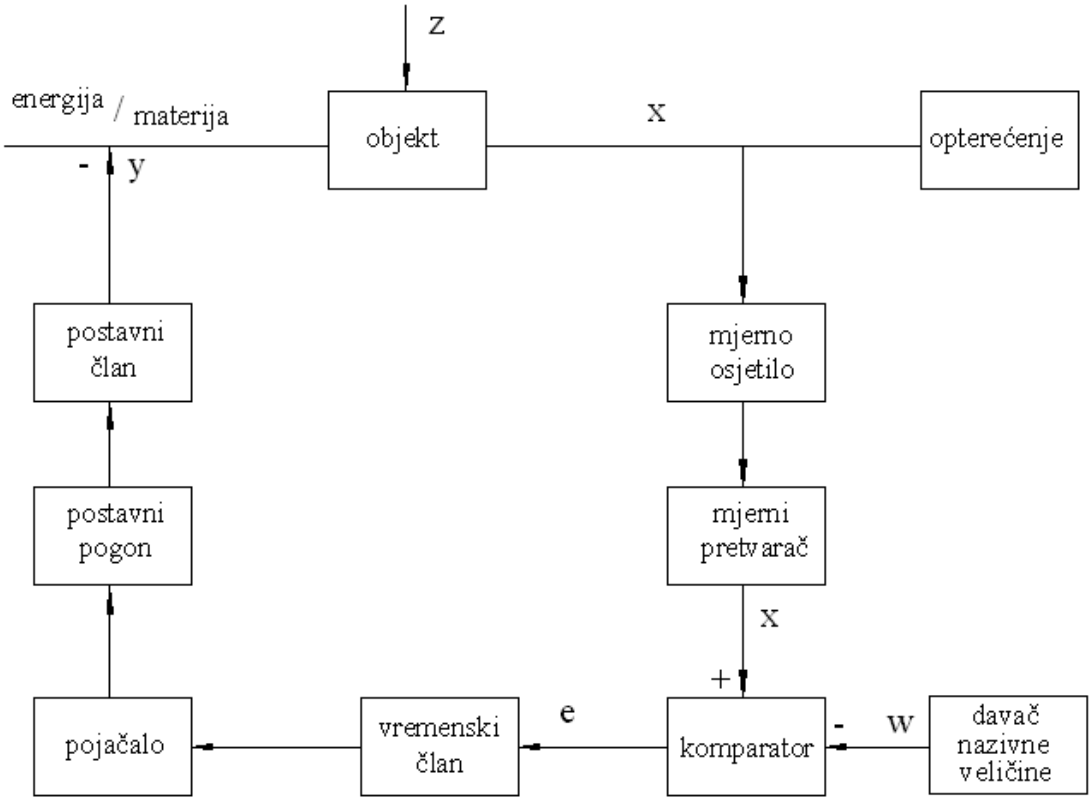
Struktura rada dana je kako slijedi. U drugom poglavlju se govori općenito o regulaciji i dijelovima regulacijskog kruga, te je prikazana maketa toplinske staze sa regulacijom protoka i temperature. Navedena je korištena programska oprema pri analizi podataka. U trećem poglavlju su prikazani PI i PID regulatori, te je objašnjen problem namatanja integratora. Objašnjena je metoda primjene dvopoložajnog regulatora, te određivanje kritičnog pojačanja (K_u) i perioda oscilacija (T_u). U četvrtom poglavlju je dan adaptivni Kalmanov filtar, te je objašnjen postupak kako se iz izlaznog signala može doći do rezonatne frekvencije i amplitude. Također su i dani grafički prikazi simulacije regulacijskog sustava sa dvopoložajnim regulatorom i određivanje rezonantne frekvencije i amplitude. U petom poglavlju su prikazani rezultati simulacije za auto tuning regulacijskog sustava sa PID regulatorom, odnosno za regulacijski sustav sa PI regulatorom. U ovom poglavlju je dan osvrt na problem mjernog šuma, te implementacija filtra koji taj problem smanjuje. U šestom poglavlju prikazani su eksperimentalni rezultati dobiveni testiranjem auto-tuning PI/PID regulatora na nastavnoj maketi toplinske staze s regulacijom protoka i temperature, te je uspoređeno njihovo vladanje. U sedmom poglavlju dana je diskusija dobivenih rezultata i moguće smjernice za daljnje istraživanje.

U prilogu su dane Matlab funkcije, C mex funkcije, Simulink blokovi korišteni u ovom radu, te izvorni (source) kod za eksperimentalnu provjeru predloženih algoritama.

2. Toplinska staza (objekt regulacije)

2.1. Regulacija

Automatizaciju možemo podijeliti u tri skupine: upravljanje, regulaciju i vođenje procesa. Upravljanje je proces pri kojem jedna ili više ulaznih veličina u ograničenom sustavu utječu na izlaznu veličinu prema zakonitostima koje su svojstvene tom sustavu. Informacija se prenosi u upravljačkom lancu ili "otvorenom krugu". Suprotno tome, pri regulaciji izlazna veličina u ograničenom sustavu djeluje povratno na ulaznu veličinu, održavajući željeno stanje. Ovdje se informacija prenosi u regulaciljskoj petlji ili "zatvorenom krugu". Vođenje procesa je kombinacija upravljanja i regulacije kod složenih sustava, uz korištenje računala. U regulaciji postoje problemi točnosti, povezani s pojavama u stacionarnom stanju. Međutim, zbog postojanja povratne veze regulacijski sustav može postati nestabilan, pa je od osnovne važnosti proučavanje stabilnosti u vezi s dinamičkim uvjetima. Pomoću reulacije se može postići visoka točnost i izlazne veličine, kao i neovisnost nekog procesa o poremećajima. Regulirana veličina (x) kojom želimo upravljati djeluje na ulaz regulacijskog uređaja, gdje se mjeri i uspoređuje s referentnom veličinom (w). Razlika između regulirane i referentne veličine jest regulacijsko odstupanje (e) koje se pojačava i na izlazu regulacijskog uređaja se naziva postavna veličina (y). Ona djeluje na ulaz procesa suprotstavljajući se djelovanju poremećajne veličine (z). Postavna veličina označena je predznakom " - " , što označava da postoji negativna povratna veza.



Slika 1. Blok - dijagram regulacijskog kruga

2.1.1. Mjerni član

Mjerenje nepoznate veličine znači brojčanu usporedbu s poznatom veličinom; mjerenjem se određuje koliko je jedinica poznate veličine sadržano u nepoznatoj veličini. Mjerenja su neophodna u regulaciji, jer ako se neke veličine ne mogu mjeriti, ne mogu se ni regulirati. Mjerni član se sastoji od tri člana:

1. Mjerno osjetilo - vrši funkciju pretvaranja jedne fizikalne veličine u drugu, npr. temperaturu pretvara u napon, tlak u otklon
2. Mjerni pretvarač - vrši funkciju pretvaranja iste fizikalne veličine u normirano područje vrijednosti kako bi se pojednostavila obrada u regulacijskom krugu
3. Mjerno pojačalo – pojačava mjerni signal tako da je iznad utjecaja smetnji

2.1.2. Regulacijski član

Regulacijski član predstavlja bitnu kariku u djelovanju automatske regulacije, jer su tu mehaniziraju logičke funkcije koje inače obavlja čovjek. Regulacijski član se sastoji od tri člana:

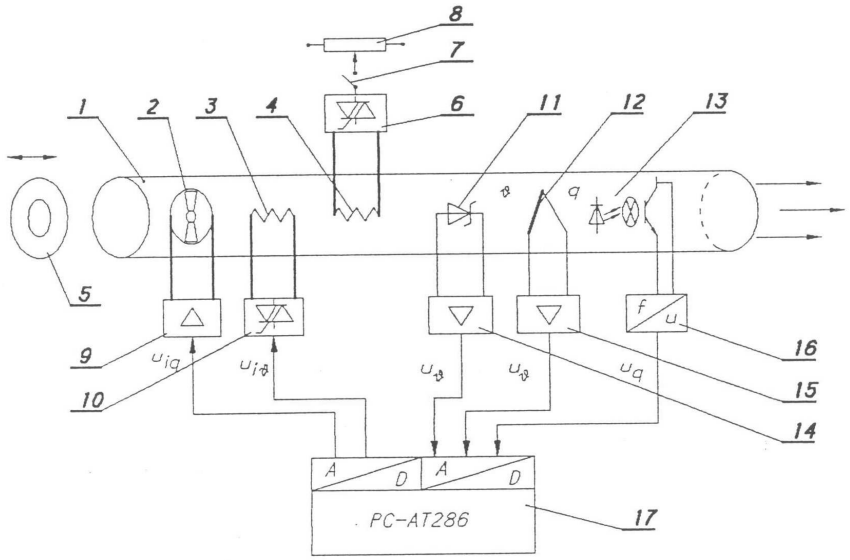
1. Komparator – obavlja logičku funkciju uspoređivanja dviju veličina: regulirane veličine i nazivne ili vodeće veličine
   - uspoređivati se mogu samo istovrsne fizikalne veličine, a to su u regulaciji položaj, sila, tlak, napon, struja i magnetski tok
   - s komparatorom je obično povezan i davač nazivne veličine
2. Regulator – pomoću regulatora se regulacijski signal vremenski preoblikuje, pa govorimo o proporcionalnom, integralnom i derivacijskom djelovanju
   - proporcionalni član reagira brzo, ali u mnogim slučajevima ne otklanja trajno regulacijsko odstupanje
   - integralni član djeluje sporo, ali u mnogim slučajevima potpuno otklanja trajno regulacijsko odstupanje
   - derivacijski član se dodjeljuje drugim regulacijskim djelovanjima jer djeluje stabilizirajuće
3. Regulacijsko pojačalo – u pojačalu se općenito pojačava snaga, međutim faktor pojačanja zapravo je omjer izlazne i ulazne veličine pojačala
   - od pojačala je taži da radi pouzdano, da ima što manju tromost i što veće pojačanje

2.1.3. Izvršni član

Na osnovi signala kojeg dobije od regulacijskog člana, izvršni član djeluje na ulaz staze, djelujući na tok energije ili materije. Izvršni član se sastoji od postavnog pogona, što je obično neki motor, i od postavnog člana, što je obično neki ventil. Postavni pogon i postavni član izvedeni su često kao jedan sklop,ali nailazimo i na postavni član izveden kao dio regulacijskog člana. Postavni pogon može raditi kontinuirano ili nekontinuirano, tj. samo s dva izlazna položaja (otvoren ili zatvoren). Prema vrsti energije razlikujemo električne, pneumatske i hidrauličke postavne pogone, ali mogu biti i kombinacija ovih pogona. U većini slučajeva postavni član je ventil koji upravlja tokom energije ili tokom materije. Pored mehaničkih ventila koji upravljaju protokom ulja, vode i zraka, postoje i električni ventili koji upravljaju tokom električne struje. Uz ventile nailazimo i na prigušene zaslone, zasune i lopatice.

2.2. Fizikalni model toplinske staze

Funkcionalna shema toplinske staze je prikazana na slici 2. Sastoji se od: cijevi puhala, ventilatora, glavnog grijača, pomočnog grijača, sustava upravljanja pomočnim grijačem, upravljačkog računala za ventilazor, pojačala snage za glavni grijač, PTC/NTC senzora temperature, senzora temperature tipa termopara, turbinskog senzora protoka, mjernog pojačala NTC/PTC senzora, mjernog pojačala za termopar, mjernog pretvornika za senzor protoka, AT računala opremljenog akvizicijskom karticom POL-812PG s A/D pretvornicima za prihvat mjernih signala i od D/A pretvornika za regulaciju.

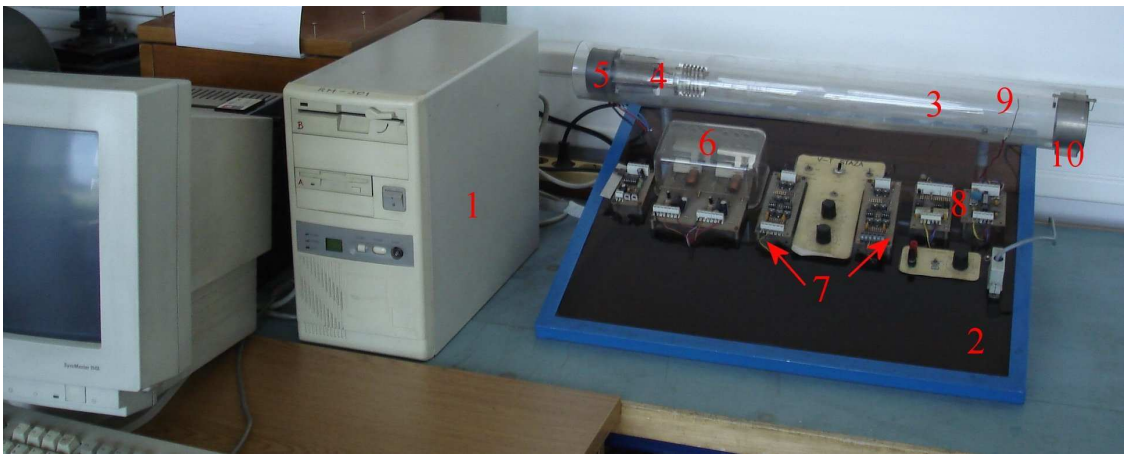


1 - cijev puhala, 2 - ventilator, 3 - glavni grijač, 4 - pomočni grijač (poremećaj), 5 - pokrov na ulazu u cijev,
6-8 - sustav upravljanja pomočnim grijačem, 9 - upravljačko pojačalo za ventilator,
10 - pojačalo snage za glavni grijač, 11 - PTC/NTC senzor temperature, 12 - senzor temperature tipa termopara,
13 - turbinski senzor protoka, 14 - mjerno pojačalo NTC/PTC senzora, 15 - mjerno pojačalo za termopar,
16 - mjerni pretvornik za senzor protoka, 17 - AT računalo opremljeno akvizicijskom katicom PCL-812PG
s A/D pretvornicima za prihvat mjernih signala i D/A pretvornicima za upravljanje/regulaciju

Slika 2. Funkcionalna shema toplinske staze

Nastavna maketa toplinske staze s regulacijom protoka i temperature prikazana je na slici 3. Sastoji se od upravljačkog računala (1) i makete grijala za zrak na postolju (2) u koje je ugrađen izvor napajanja makete. Zrak se pomoću ventilatora (5) s promjenjivom brzinom vrtnje upuhuje u cijev (3), te se prelaskom preko sustava grijača (4) zagrijava na određenu temperaturu. Grijači se napajaju iz tiristorskog pojačala (6). Mjerenje temperature i protoka zraka obavlja se pomoću senzora temprature i protoka s odgovarajućim mjernim pojačalima (8-10). Dosadašnji koncepti regulacije temperature i/ili protoka zasnivala su se na analognim regulatorima (7) (prvobitno rješenje), te digitalnim PID regulatorima implementiranim na PC računalu s odgovarajućim akvizicijskim karticama (novije rješenje). Studenti na kolegiju "Mikroprocesorsko upravljanje" izravno implementiraju digitalne PID regulatore temperature i protoka na PC računalu u programskom jeziku C.

.



1 – upravljačko PC računalo, 2 – postolje makete s izvorom napajanja, 3 – cijev za zrak, 4 – grijači, 5 – ventilator, 6 – tiristorsko pojačalo za napajanje grijača, 7 – alternativni analogni regulatori temperature i protoka, 8 – pojačala signala s pretvornika, 9 – mjerni pretvornik temperature, 10 –mjerni pretvornik protoka.

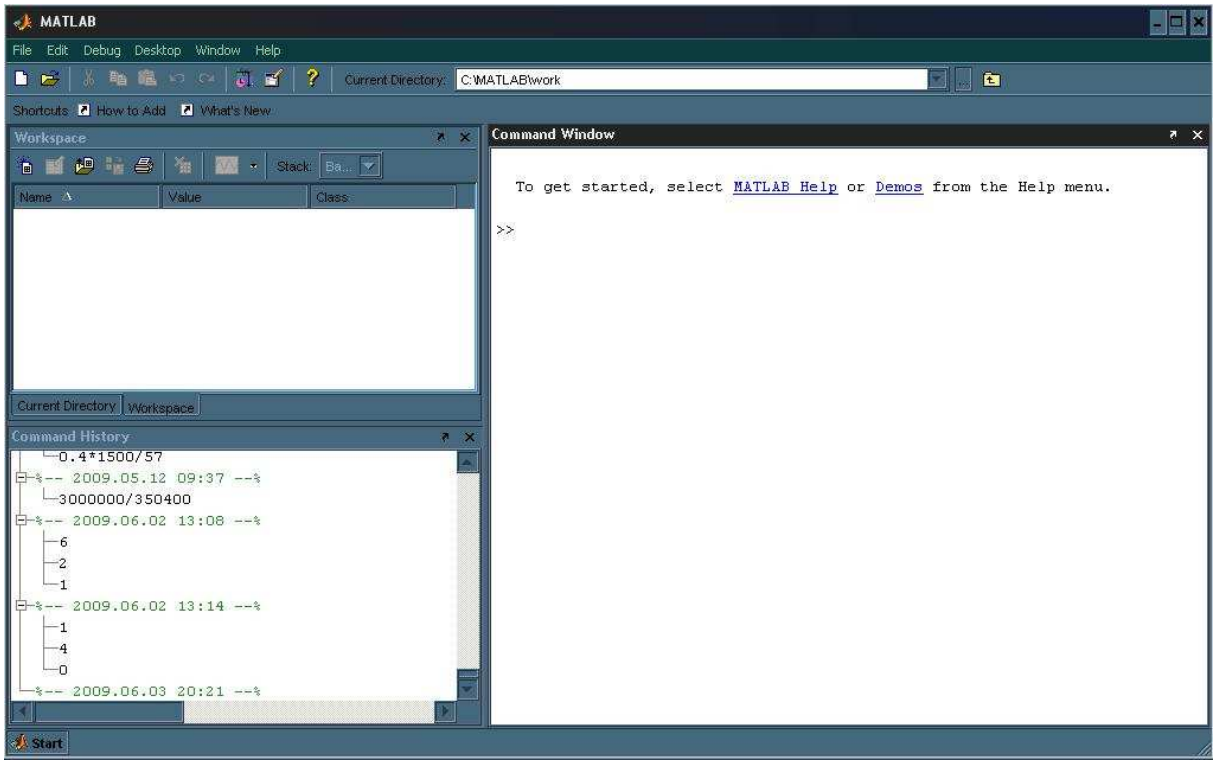
Slika 3. Nastavna maketa toplinske staze

2.3. Programska podrška korištena pri analizi podataka

2.3.1. Matlab

Matlab je matematički programski paket za znastveni i inženjerski numerički račun, a izrastao je iz desetljeća usavršavanih fortranskih paketa LINMPACKA-a i EISPACK-a. Stoga nije ni čudo da se smatra standardom sveučilišnog matematičkog alata, iako se intezivno koristi kako u industrijskom razvoju tako i u praktičnom inženjerstvu. Prva verzija Matlab-a napisana je krajem 1970. godine na sveučilištima University of New Mexico i Stanford University s ciljem primjene u matričnoj teoriji, linearnoj algebri i numeričkoj analizi. Budući da je zamišljen i razvijen kao interpretacijski programski jezik visoke razine, koji se u početku temeljio na kompleksnoj matrici kao osnovnom tipu podataka, imenovan je kraticom od **Mat**rični **lab**oratorij. Mogučnost povezivanja s programima pisanim u C jeziku ili Fontranu, čini ga otvorenim za složene projekte, a gotova (funkcijska) rješenja za različita područja primjene kontinuirano mu proširuje domet. Po svojoj formi blizak je načinu na koji inače zapisujemo matematičke formule, pa jedan redak u Matlab-u može zamijeniti na stotine redaka napisanih u nekom programskom jeziku opće namjene. Matlab je stoga jezik visoke učinkovitosti u tehničkom računanju. On objedinjuje računanje, vizualizaciju i programiranje u prozorskom okolišu vrlo ugodnom za korisnika, gdje su problemi i rješenja izražena uobičajenim matematičkim zapisom. Tipična upotreba matlaba uključuje:

- Matematiku i računanje
- Razvitak algoritama
- Modeliranje, simulaciju i izgradnju prototipova
- Analizu, obradu i vizualizaciju podataka
- Znastvenu i inženjersku grafiku
- Razvitak gotovih rješenja sa GUI alatima



Slika 4. Matlab

Danas svojstva Matlab-a daleko prelaze originalni "matrični laboratorij". Uz osnovni paket postoje i brojni programski alati (toolboxes) koji pokrivaju gotovo sva područja inženjerske djelatnosti. Paket SIMULINK je dodatak Matlab-u koji omogućuje simulaciju kontinuiranih i diskretnih sustava pomoću funkcijskih blok dijagrama i dijagrama stanja. Matlab je otvoren sustav u kojem korisnik može graditi svoje vlastite alate i biblioteke te modificirati postojeće, jer su dostupni u obliku izvornog koda. Razlog ovakve popularnosti leži u nekoliko činjenica:

- Matlab se odlikuje elegancijom, praktičnošću i preglednošću, pa se poput pseudokoda primjenjuje u mnogim knjigama kod opisivanja računskih postupaka
- Matlab posjeduje veliku fleksibilnost: od običnog stolnog računala za računanje brojeva i matrica do sredstva za rješavanje zahtjevnih zadataka
- Math Works nudi jako dobru "On - line" potporu
- Složeniji programi mogu biti postavljeni u kraćem vremenu, za razliku od vremena koje je potrebno kod "klasičnih" programskih jezika
- Matlab se brzo uči
- Matlab posjeduje jaku grafičku potporu, koja se sa svojim jednostavnim funkcijama može brzo naučiti, te nudi široke mogućnosti primjene.

Sva ta svojstva čine Matlab omiljenim među znanstvenicima i inženjerima, kao i među praktičarima različitih pravaca. Tome pridonosi Matlab-ova proširljivost kroz samoizgradnju ili dodavanje pripremljenih funkcija.

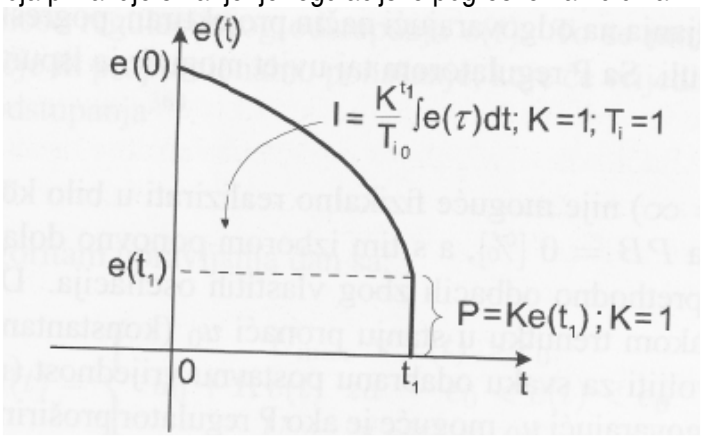
3. Sinteza regulatora temperature temeljem identificiranog modela toplinske staze

3.1. PI regulator

PI regulator se često koristi u praksi, a koristi dva osnovna ponašanja: P – proporcionalno, I – integracijsko. Algoritam upravljanja kod PI regulatora glasi:

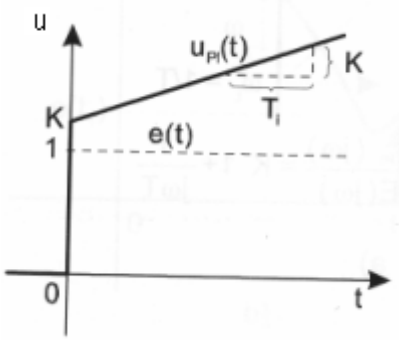
$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_o^t e(\tau)d\tau\right] \tag{1}$$

Na slici 5. je funkcija koja prikazuje smanjenje regulacijske pogreške na nulu za PI regulator.



Slika 5. Formiranje signala PI regulatora

PI regulator djeluje tako da smanjuje pogrešku sustava u ustaljenom stanju prema nuli. Prema tome, može se zaključiti da PI regulator može izbjeći pogrešku u ustaljenom stanju koja može proizaći iz primjene P regulatora. PI regulator je vrlo raširen u industriji, pogotovo za one primjere kad nema većih zahtjeva na brzinu odziva. Iako je integracijsko djelovanje eliminiralo potrebu za resetiranjem postavne vrijednosti, ono ima nepoželjan utjecaj na brzinu odziva i stabilnost sustava. Prijelazna karakteristika PI regulatora dana je na slici 6, gdje se vidi da je nagla promjena na izlazu rezultat proporcionalnog djelovanja, a ne integracijskog.



Slika 6. Prijelazna karakteristika PI regulatora

Ako je ubrzanje odziva naš prvenstveni interes, tada PI regulator treba proširiti derivacijskim djelovanjem koje može dodatno ubrzati odziv.

3.2. PID regulator

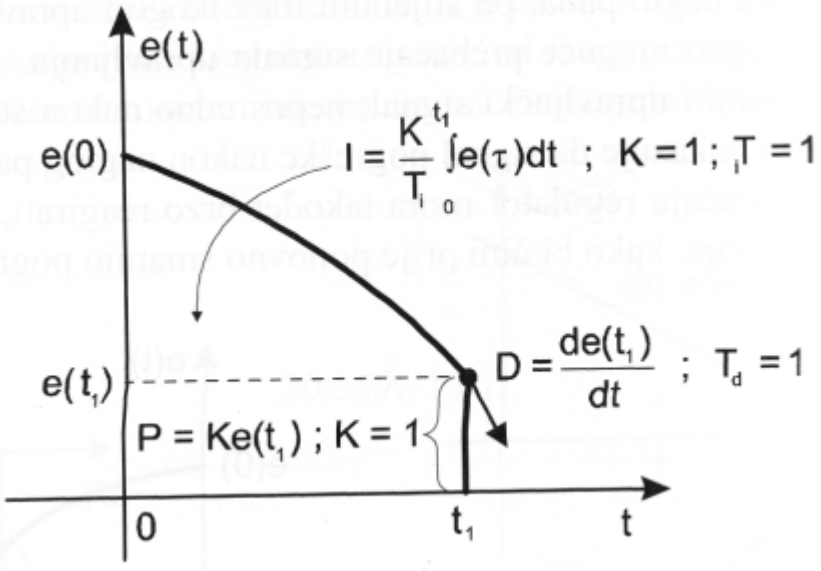
PID regulator ima sva neophodna dinamička ponašanja za potrebe kvalitetne regulacije: brzu reakciju na naglu promjenu pogreške (D dio), povećanje upravljačkog signala kako bi se pogreška natjerala prema nuli (I dio), te odgovarajući energetski sadržaj unutar određenog područja regulacijskog odstupanja, kako bi se eliminirale vlastite oscilacije (P dio). Algoritam upravljanja PID regulatora glasi:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_o^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \tag{2}$$

odnosno:

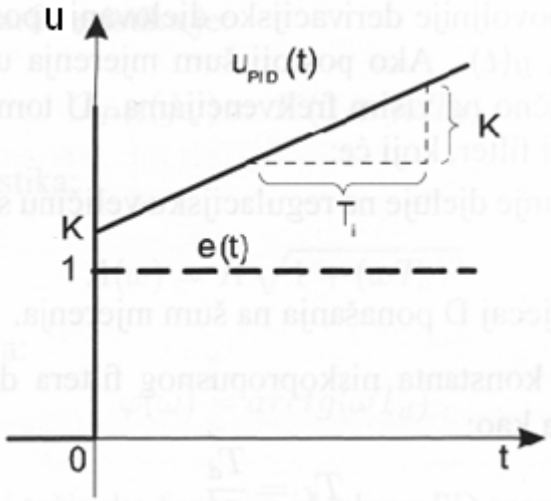
$$u(t) = Ke(t) + K_i\int_o^t e(\tau)d\tau + K_d\frac{de(t)}{dt} \tag{3}$$

Na slici 7. je funkcija koja prikazuje smanjenje regulacijske pogreške na nulu za PID regulator.



Slika 7. Formiranje signala PID regulatora

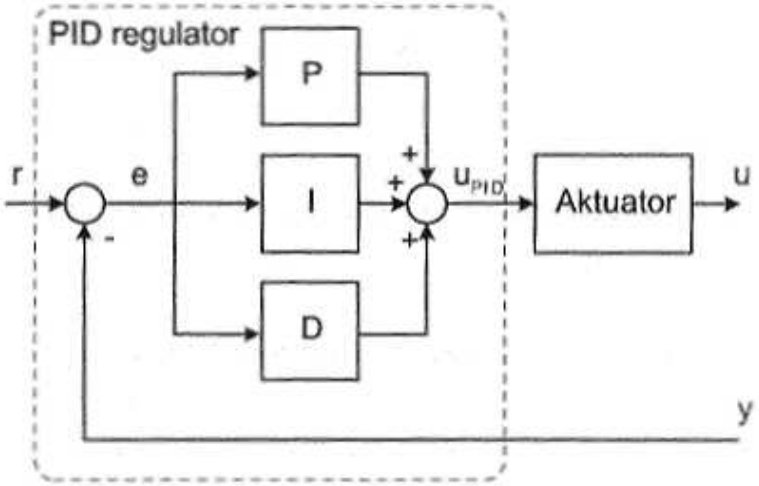
PID regulator ima prijelaznu karakteristiku koja se dobije kao zbroj individualnih P, I i D prijelaznih karakterisitika, a prikazana je na slici 8.



Slika 8. Prijelazna karakteristika PID regulatora

3.2.1. Direktna struktura PID regulatora

Kod ove strukture P, I i D ponašanje djeluje na signal regulacijske pogreške. Paralelni spoj proporcionalnog, derivacijskog i integracijskog ponašanja tvori neinteraktivnu strukturu PID regulatora, koja je prikazana na slici 9.



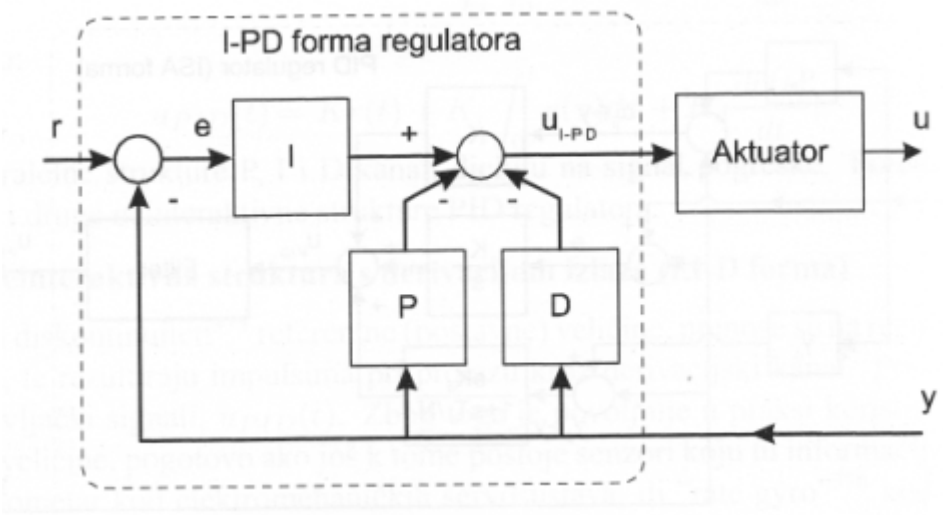
Slika 9. Direktna struktura PID regulatora

Kod ovakve strukture algoritam upravljanja je identičan algoritmu (3):

$$u_{PID}(t) = Ke(t) + K_i \int_o^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{4}$$

3.2.2. Modificirana struktura PID regulatora

Kod ove strukture samo integracijsko ponašanje djeluje na signal regulacijske pogreške, dok je proporcionalno i derivacijsko djelovanje u povratnoj vezi. Modificirana struktura PID regulatora je prikazana na slici:



Slika 10. Modificirana (I+PD) struktura PID regulatora

Zakon upravljanja za tu strukturu je:

$$u_{I-PD}(t) = K\left[-y(t) + \frac{1}{T_i}\int_o^t e(\tau)d\tau - T_d\frac{dy(t)}{dt}\right] \tag{5}$$

odnosno:

$$u_{I-PD}(t) = -Ky(t) + K_i\int_o^t e(\tau)d\tau - K_d\frac{dy(t)}{dt} \tag{6}$$

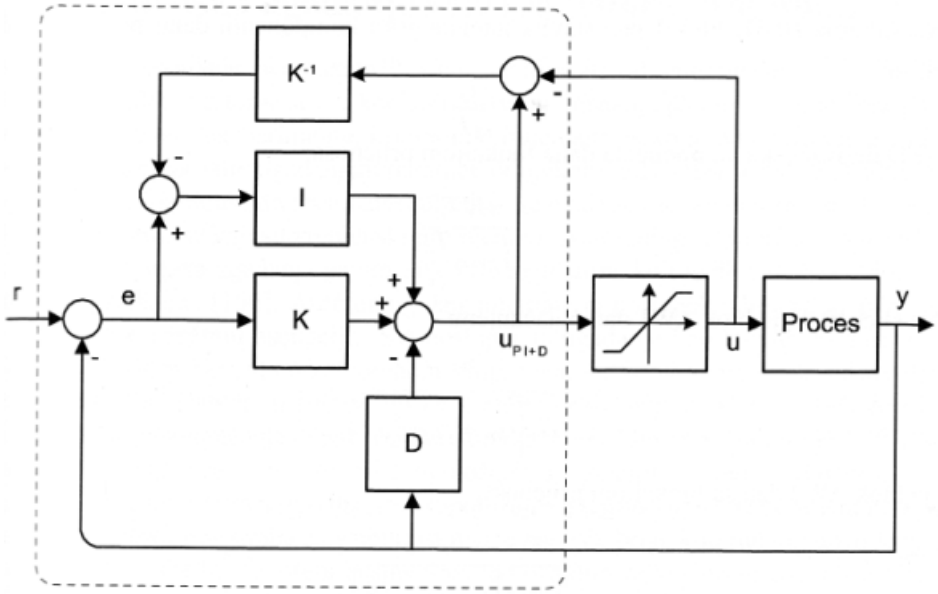
Prednost ovakve strukture regulatora je u tome da parametri PID regulatora utječu samo na položaje polova zatvorenog regulacijskog kruga (nule PID regulatora se ne pojavljuju u prijenosnoj funkciji zatvorenog regulacijskog kruga). Ovakva struktura jednako dobro potiskuje utjecaj poremećajne veličine procesa, uz bitno manje forsiranje upravljačke veličine na nagle promjene referentne vrijednosti.

3.3. Namatanje integratora (integrator windup)

Integrator ima obilježje perzistiranog djelovanja na izlaz regulatora jer stalno integrira signal na njegovu ulazu i ne staje s integracijom osim kada je pogreška nula. Ako izvršni element ima zasićenje, tada bi integrator trebao stati s forsiranjem upravljačke veličine jer je on ionako u zasićenju i ne može isporučiti procesu signal koji više nego što iznosi postignuta maksimalna vrijednost. Regulator šalje prema procesu signal koji stalno raste, dok proces (aktuator) vidi i osjeća signal koji je konstantan (aktuator u zasićenju). Ta pojava naziva se namatanje integratora odnosno "windup" i mora se spriječiti. Naime, mora se spriječiti da, kada aktuator uđe u zasićenje, signal iz regulatora mora ostati na zatečenoj razini i ne smije dalje rasti. Postoji cijeli niz različitih postupaka kojima je moguće otkloniti namatanje (windup), a susreću se pod nazivom "antiwindup" strategije. Neke od "antiwindup" strategija su:

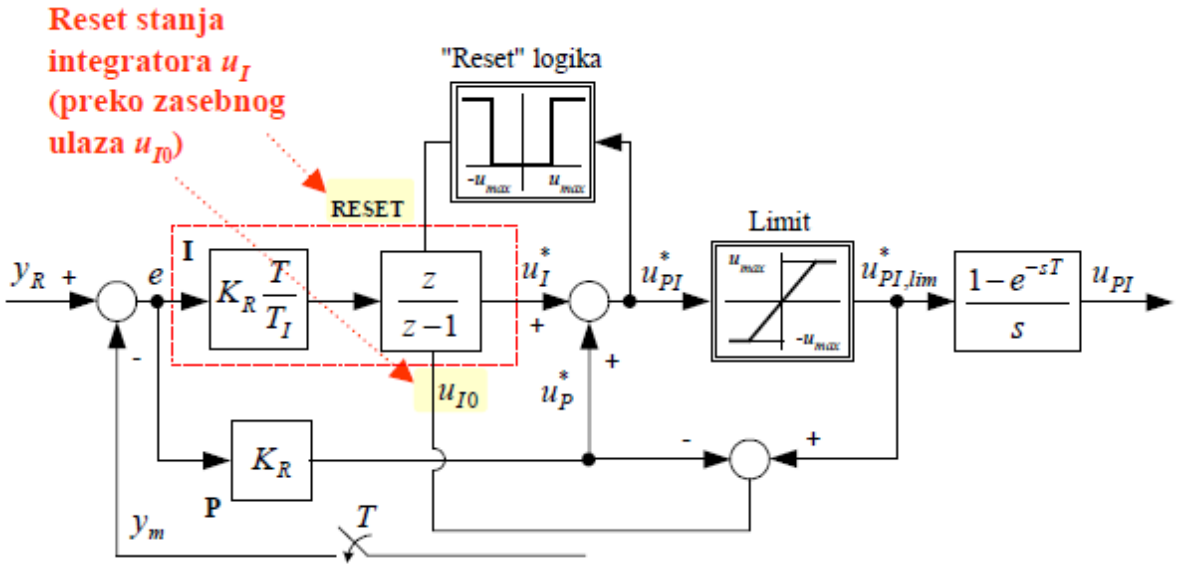
- Anti-windup s uvjetnom integracijom. Ako je izlaz aktuatora u zasićenju, a ulaz i izlaz regulatora su istog predznaka, tada postavi ulaz integratora na nulu.
- Anti-windup s praćenjem razlike zasićenog i nezasićenog signala. Ako je aktuator u zasičenju, smanji ulaz integratora za neku konstantu koja je proporcionalna razlici između nezasićenog i zasićenog signala regulatora.

Uvjetna "antiwindup" tehnika je prikazana za PID regulator na slici 11.



Slika 11. Antiwindup sa PI+D strukturu PID regulatora

Najčešće korišteno rješenje u vremenski-diskretnim regulatorima je antiwindup s praćenjem razlike i prikazano je na slici 12. Ukoliko je suma proporcionalnog i integrirajućeg djelovanja izvan izlaznog raspona regulatora, stanje integratora se resetira na vrijednost koja odgovara razlici pripadajućeg limita i proporcionalnog djelovanja.



Slika 12. Antiwindup za PI regulatora

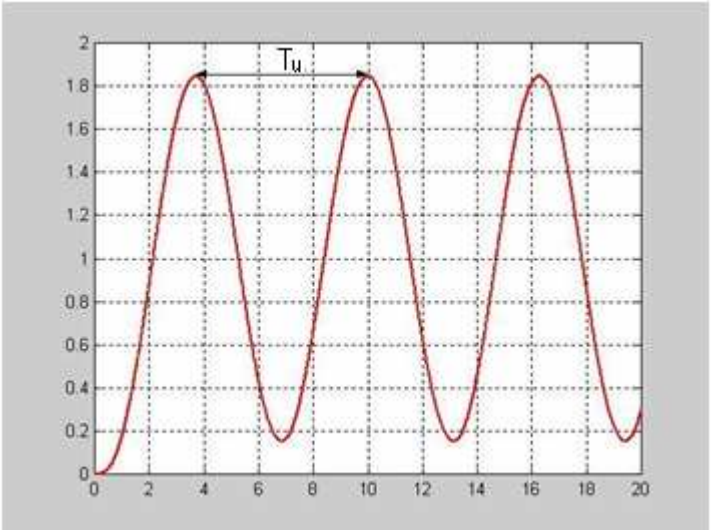
3.4. Podešavanje PI/PID regulatora Takahashi-evom metodom dovođenja na rub stabilnosti

Ova metoda koristi se tamo gdje je dozvoljeno dovesti regulacijski sustav do ruba stabilnosti ili ako prijelazna karakteristika objekta upravljanja nema aperiodski karakter. Važno je napomenuti da su regulatori ovim postupkom ugođeni za režim stabilizacije, a ne za režim sljeđenja postavne veličine. Ti regulatori dobro otklanjaju utjecaj poremećaja, ali nisu prikladni za zadatke praćenja (slijeđenja). Kod ove metode je vrlo bitno iskustvo.

Postupak namještanja je slijedeći:

- za regulator koji se koristi u regulacijskom sustavu odabere se proporcionalni regulator, bez integralnog i derivacijskog djelovanja,
- pojačanje regulatora se povećava sve dok se ne dobiju trajne oscilacije konstantne amplitude, pojačanje uz koje se dobiju trajne oscilacije označava se kritičnim (graničnim) pojačanjem regulatora (K_u), te se za taj slučaj odredi period kritičnih oscilacija (T_u).

Na temelju dobivenih vrijednosti parametara T_u i K_u podešava se PI/PID regulator prema izrazima navedenim u tablici 1 (vidi npr. [5] i tamo navedene reference).



Slika 13. Zatvoreni krug sa proporcionalnim regulatorom dovedenim na rub stabilnosti

Regulator	K_R	T_I	T_D
PI	$0{,}45 \cdot K_u - 0{,}27 \cdot K_u \frac{T}{T_u}$	$\frac{K_R \cdot T_u}{0{,}54 \cdot K_u}$	–
PID	$0{,}6 \cdot K_u - 0{,}6 \cdot K_u \frac{T}{T_u}$	$\frac{K_R \cdot T_u}{1{,}2 \cdot K_u}$	$\frac{3 \cdot K_u \cdot T_u}{40 \cdot K_R}$

Tablica 1. Ziegler - Nicholsove preporuke

Gdje je T vrijeme uzrokovanja, a uzeto je iz literature [5]. $T = (0{,}1 – 0{,}3)T_u$. Za PID regulatore se uzima manje vrijeme uzrokovanja ($T=0{,}1T_u$).

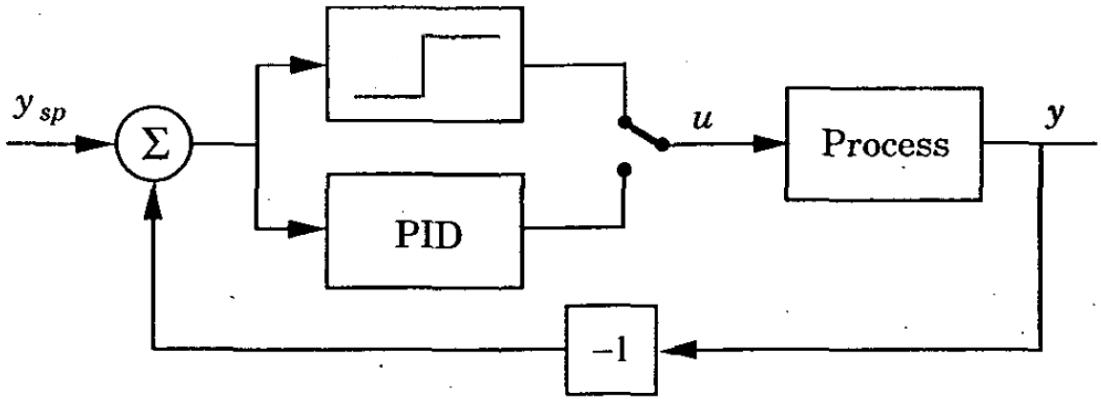
3.5. Dvopoložajni regulator u zatvorenom krugu

Problem kod sinteze regulatora po Ziegler-Nicholsovoj metodi ruba stabilnosti jest upravo dovođenje zatvorenog sustava na rub stabilnostijer je iznimno teško pogoditi stvarnu granicu stabilnosti i uvijek postoji opasnost od ulaska sustava u nestabilnost (raspirivanje). Stoga je osmišljen relejni postupak u kojem je sustav na rubu stabilnosti, ali sa kontroliranim amplitudama oscilacija.
Prednosti ove metode su:

1. Identifikacija procesa oko važnih frekvencija, kritičnih frekvencija (frekvencija gdje je fazni kut $-\pi$)
2. Eksperiment je u zatvorenoj petlji, stoga neće odstupati od nominalne radne točke
3. Za procese karakterizirane sporom dinamikom, postiže se bolja vremenska učinkovitost od ostalih metoda. Sustav se brzo dovede na rub stabilnosti primjenom relejnog člana

3.5.1. Automatsko ugađanje PID regulatora

Kod automatskog ugađanja parametri regulatora se parametri postave automatski temeljem eksperimenata i potom se ne mijenjaju tijekom rada sustava. Ovdje se primjenjuje postupak određivanja parametra K_u i T_u sustava u režimu oscilacija. Svi postupci za ugađanje koji koriste parametre K_u i T_u dobivene iz eksperimenta u zatvorenom krugu, temelje se na tome da se sustav dovede u režim oscilacija. Zatvoreni sustav biti će u režimu oscilacija ako ga do toga dovedemo ručnim mjenjanjem pojačanja ili pak koristimo dvopoložajni regulator. Oscilacije koje će se uspostaviti dvopoložajnim regulatorom zovu se vlastite oscilacije i svojstvene su nelinearnim sustavima. Uvjet postizanja stabilnih oscilacija je $\varphi = -180^\circ$, $L(\omega)[dB] = 0$ (za linearne sustave). Na slici 14. dana je načelna blok shema sustava upravljanja s dvopoložajnim regulatorom (y_{sp} je referentna (vodeća) vrijednost, u je upravljačka veličina, a y je izlaz procesa, tj. objekta upravljanja).



Slika 14. Blok shema sustava s dvopoložajnim regulatorom

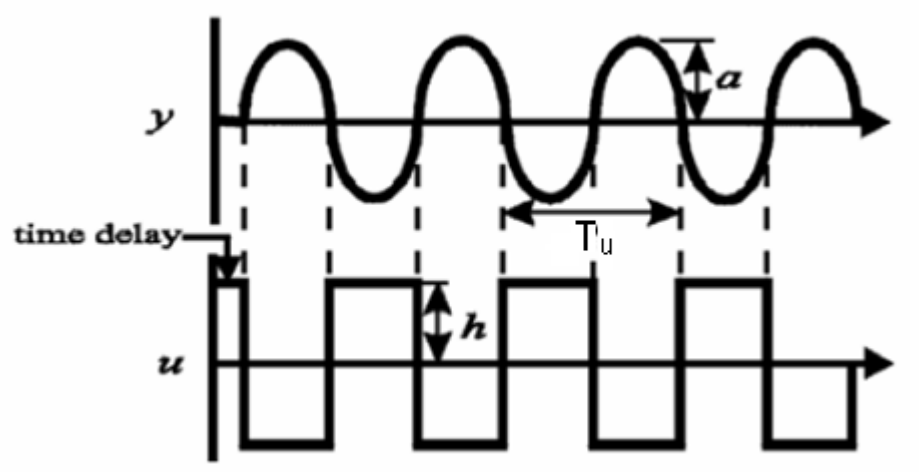
Postupak automatskog ugađanja se sastoji od sljedećih koraka:

1. Procesom se upravlja dvopoložajnim regulatorom kod kojeg se upravljački signal daje:

$$u(t) = \begin{cases} U_{\max} ; e(t) \rangle 0 \\ -U_{\max} ; e(t) \langle 0 \end{cases}$$

S dvopoložajnim regulatorom uspostavit će se nakon nekog vremena oscilatorni režim rada.

2. U oscilatornom režimu, izlazni signal će oscilirati određenom amplitudom i frekvencijom (Slika 15.)



Slika 15. Prikaz izlaznog i upravljačkog signala

3. Period oscilacija (T_u) i amplituda oscilacija (a) mogu se automatski očitati digitalnim regulatorom pod čijim se nadzorom vrši eksperiment
4. Pojačanje K_u određuje se iz formule:

$$K_u = \frac{4U_{\max}}{\pi a} \tag{7}$$

5. S poznatim pojačanjima Ku i periodom oscilacija Tu može se pristupiti izboru koeficijenata PID regulatora po nekoj od preporuka koje koriste te parametre (npr. Ziegler-Nicholsove preporuke, odnosno prema Takahashi-u za vremenski diskretne regulatore)

U slučaju da postoji šum, tada se dvopoložajnim regulatorom mogu dobiti rezultati eksperimenata koji nisu pouzdani i mogu rezultirati loše podešenim parametrima. To se može izbjeći ako se koristi dvopoložajni regulator sa histerezom.

3.5.2. Opisna funkcija

Opisna funkcija daje iznos ekvivalentnog kritičnog pojačanja nelinearnog člana (relejnog člana) u stanju oscilacija. Ako je na ulaz nelinearnog člana narinuta pobuda:

$$x_u(t) = x_u \sin \omega t\ , \tag{8}$$

odzivna funkcija je:

$$x_i(t) = A_o + \sum_{n=1}^{\infty} (A_n \cos n\omega t + B_n \sin n\omega t)\,. \tag{9}$$

Zbog niskopropusnog djelovanja procesa na izlazu praktički ostaju samo članovi osnovne frekvencije:

$$x_{i1}(t) = A_1 \cos \omega t + B_1 \sin \omega t \tag{10}$$

gdje su vrijednosti koeficijenata:

$$A_1 = \frac{2}{T}\int_0^T x_i(t)\cos\omega t\,dt = \frac{1}{\pi}\int_0^{2\pi} x_i(\alpha)\cos\alpha\,d\alpha \tag{11}$$

$$B_1 = \frac{2}{T}\int_0^T x_i(t)\sin\omega t\,dt = \frac{1}{\pi}\int_0^{2\pi} x_i(\alpha)\sin\alpha\,d\alpha \tag{12}$$

Napisani su integralni izrazi za varijablu t kao i za novu varijablu – fazni kut α = ωt. Ispuštena je istosmjerna komponenta A_0 koja se pojavljuje kod nesimetričnih nelinearnosti i koja prouzrokuje pomak radne točke.
Izraz (10) možemo preinačiti tako da dobijemo sažeti oblik:

$$x_{i1}(t) = X_{i1}\sin\varphi_1\cos\omega t + X_{i1}\cos\varphi_1\sin\omega t = X_{i1}\sin(\omega t + \varphi_1) \tag{13}$$

gdje je $A_1 = X_{i1}\sin\varphi_1$ i $B_1 = X_{i1}\cos\varphi_1$ ili amplituda i faza:

$$X_{i1} = \sqrt{A_1^2 + B_1^2} \tag{14}$$

$$\varphi_{i1} = arctg\frac{A_1}{B_1} \tag{15}$$

Radi jednostavnijeg izvoda opisne funkcije, izraze (8) i (13) pretvaramo u eksponencijalni oblik:

$$x_u(t) = X_u e^{j\omega t}, \tag{16}$$

$$x_{i1}(t) = X_{i1}e^{j\omega t}e^{j\varphi_1} \tag{17}$$

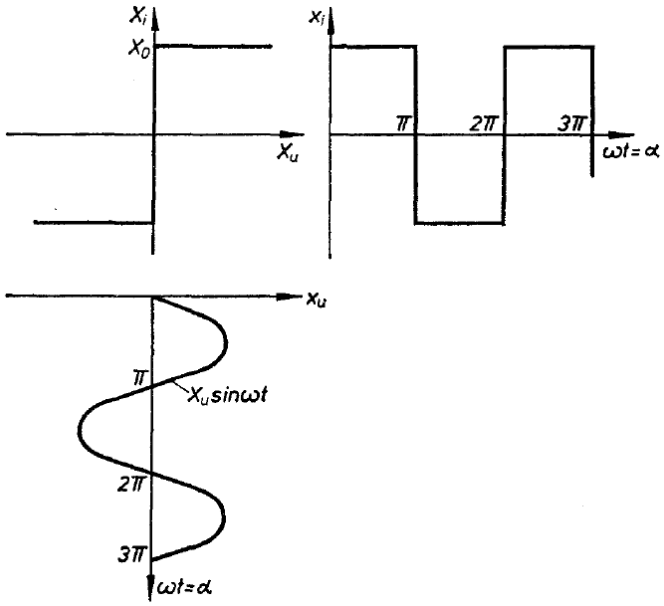
U izrazu za x_{i1} koristimo se najprije Eulerovom relacijom $e^{j\varphi_1} = \cos\varphi_1 + j\sin\varphi_1$ i zatim zamjenimo izraze za A_1 i B_1:

$$x_{i1}(t) = X_{i1}e^{j\omega t}\cos\varphi_1 + jX_{i1}e^{j\omega t}\sin\varphi_1 = B_1e^{j\omega t} + jA_1e^{j\omega t} \tag{18}$$

Dijeljenjem x_{i1} iz jednadžbe (18) sa xu iz jednadžbe (16) dobije se opisna funkcija u kompleksnom obliku:

$$N(X_u) = \frac{x_{i1}(t)}{x_u(t)} = \frac{B_1}{X_u} + j\frac{A_1}{X_u} \tag{19}$$

Ovaj izraz vrijedi kod dvoznačne nelinearnosti (histereza), odnosno kad nelinearni član diferencijalne jednadžbe sadrži i spremnik energije (derivacija). U slučaju simetrične i jednoznačne nelinearnosti (npr. signum funkcija), nema faznog pomaka φ_1, pa otpada i imaginarni član u jednadžbi.



Slika 16. Izvod opisne funkcije dvopoložajnog releja

Ulazna sinusna funkcija daje odrazom na statičkoj karakteristici relaja pravokutni impulsni niz. Kako je to liha funkcija, otpadaju kosinusni članovi, pa ostaje osnovni sinusni član. Uzimamo izraz (16) kao koeficijent B_1, s tim da zbog preglednosti uvodimo novu neovisnu varijablu – fazni kut $\alpha = \omega t$. Dobiva se:

$$B_1 = \frac{1}{\pi}\int_0^{2\pi} x_{i1}(t)\sin\alpha\, d\alpha = \frac{2}{\pi}\int_0^{\pi} X_0 \sin\alpha\, d\alpha = -\frac{2X_0}{\pi}\left[\cos\alpha\right]_0^{\pi} = \frac{4X_0}{\pi} \tag{20}$$

Iz toga slijedi da je opisna funkcija dvopoložajnog releja:

$$N(X_u) = \frac{x_{i1}(t)}{x_u(t)} = \frac{4X_0}{\pi}\sin\omega t \frac{1}{X_u \sin\omega t} = \frac{4X_0}{\pi X_u} \tag{21}$$

Odnosno, gore navedeni izraz definira ekvivanelntno kritično pojačanje K_u.

4. Adaptivni Kalmanov filtar

U ovom poglavlju opisuje se standardni oblik Kalmanovog filtra i adaptacijski mehanizam zasnovan na detekciji naglih promjena varijabli stanja procesa. Navedeni se algoritmi stapaju u cjeloviti adaptivni algoritam procjene varijabli stanja, koji se odlikuje visokom točnošću slijeđenja varijabli stanja i povoljnim odnosom signala i šuma. Sadržaj je većim djelom preuzet iz [6].

4.1. Klasični oblik Kalmanovog filtra

4.1.1. Model procesa

Razmatra se problem procjene varijabli stanja multivarijabilnog, stohastičkog i vremenski-diskretnog sustava opisanog sljedećim modelom:

$$x(k) = F(k-1)\,x(k-1) + G(k-1)\,u(k-1) + \Omega(k-1)\,v(k-1) \tag{10}$$

$$y(k) = H(k)\,x(k) + e(k) \tag{11}$$

gdje su:

x(k), y(k) – vektori varijabli stanja i izlaznih varijabli
v(k-1) – vektor stohastičkih perturbacija varijabli stanja x
e(k) – vektor šuma mjerenja
u(k-1) – vektor ulaznih varijabli
F(k-1) – matrica sustava
H(k-1) – izlazna matrica
G(k-1) – ulazna matrica
Ω(k-1) – matrica perturbacija stanja

Pritom se pretpostavlja da pojedine komponente stohastičkih perturbacija u varijablama stanja v(*k*) i šuma mjerenja e(*k*) imaju svojstva bijelog šuma s očekivanjima (srednjim vrijednostima) jednakim nuli i vjerojatnosnim gustoćama raspodjela *p*(e) i *p*(v) normalnog (Gaussovskog) tipa. Nadalje, pretpostavlja se da su vektori stohastičkih varijabli v(*k*) i e(*k*) međusobno neovisni (nekorelirani), te da su također pojedine komponente svakog od vektora v(*k*) i e(*k*) međusobno nekorelirane. Navedeni uvjeti mogu se iskazati sljedećim izrazima:

$$Ev(k) = \underline{0}$$
$$Ee(k) = \underline{0} \tag{12}$$

$$E\left\langle v(k)\,e^T(k)\right\rangle = \underline{0} \tag{13}$$

$$E\left\langle v(k)\,v^T(k)\right\rangle = Q(k) = \begin{bmatrix} q_{11}(k) & 0 & \cdots & 0 \\ 0 & q_{22}(k) & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & q_{ll}(k) \end{bmatrix} \tag{14}$$

$$E\left\langle e(k)e^T(k)\right\rangle = R(k) = \begin{bmatrix} r_{11}(k) & 0 & \cdots & 0 \\ 0 & r_{22}(k) & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & r_{mm}(k) \end{bmatrix} \tag{15}$$

gdje su:

0 – nul-matrica, odnosno nul-vektor

Q(k), R(k) – pozitivno definitne simetrične matrice kovarijanci stohastičkih perturbacija u varijablama stanja i šuma mjerenja

Dijagonalni elementi matrica kovarijanci Q(*k*) i R(*k*) odgovaraju varijancama pojedinih komponenti vektora v(*k*) i e(*k*). Kako je pretpostavljeno da su pojedine komponente vektora v(*k*) i e(*k*) međusobno nekorelirane, izvandijagonalni elementi (međukorelacije) su jednaki nuli.

Varijable stanja x(*k*) stohastičkog modela karakterizirane su očekivanjem *E*(x(*k*)) = x(*k*) i matricom kovarijanci odstupanja varijabli stanja x(*k*) od očekivanih vrijednosti x(*k*) definiranom sljedećim izrazom:

$$P(k) = E\left\langle \left[x(k) - \bar{x}(k)\right] \cdot \left[x(k) - \bar{x}(k)\right]^T \right\rangle \tag{16}$$

Može se pokazati da je uz poznatu matricu kovarijanci perturbacija stanja Q(*k*) dinamika matrice kovarijanci P(*k*) procesa opisanog izrazom (1) određena takozvanom vremenski diskretnom Riccatijevom jednadžbom:

$$P(k) = F(k-1)P(k-1)F^T(k-1) + \Omega(k-1)Q(k-1)\Omega^T(k-1) \tag{17}$$

4.1.2. Struktura Kalmanovog filtra

Kalmanov filtar je sustav procjene varijabli stanja (estimator stanja) stohastičkog sustava, koji je optimalan u smislu minimuma kovarijance pogreške procjene varijabli stanja. Za stohastički sustav Kalmanov filtar je definiran sljedećim dinamičkim jednadžbama:

$$x(k \mid k-1) = F(k-1)\,x(k-1 \mid k-1) + G(k-1)\,u(k-1) \tag{18}$$

$$\varepsilon(k \mid k-1) = y(k) - H(k)\,x(k \mid k-1) \tag{19}$$

$$P(k \mid k-1) = F(k-1)P(k-1 \mid k-1)F^T(k-1) + \Omega(k-1)\,Q(k-1)\,\Omega^T(k-1) \tag{20}$$

$$K(k) = P(k \mid k-1)\,H^T(k)\left[H(k)\,P(k \mid k-1)\,H^T(k) + R(k)\right]^{-1} \tag{21}$$

$$P(k \mid k) = P(k \mid k-1) - K(k)\,H(k)\,P(k \mid k-1) \tag{22}$$

$$x(k \mid k) = x(k \mid k-1) + K(k)\,\varepsilon(k \mid k-1) \tag{23}$$

gdje je:

$x(k \mid k-1)$ - inicijalna procjena varijabli stanja na osnovi determinističkog dijela modela procesa

$x(k \mid k)$ - konačna procjena varijabli stanja korigirana na osnovi mjerenja

$\varepsilon(k \mid k-1)$ - inicijalna procjena predikcijske pogreške Kalmanovog filtra

$K(k)$ - matrica pojačanja Kalmanovog filtra

$P(k \mid k-1)$ - inicijalna procjena matrica kovarijanci pogreški procjene stanja

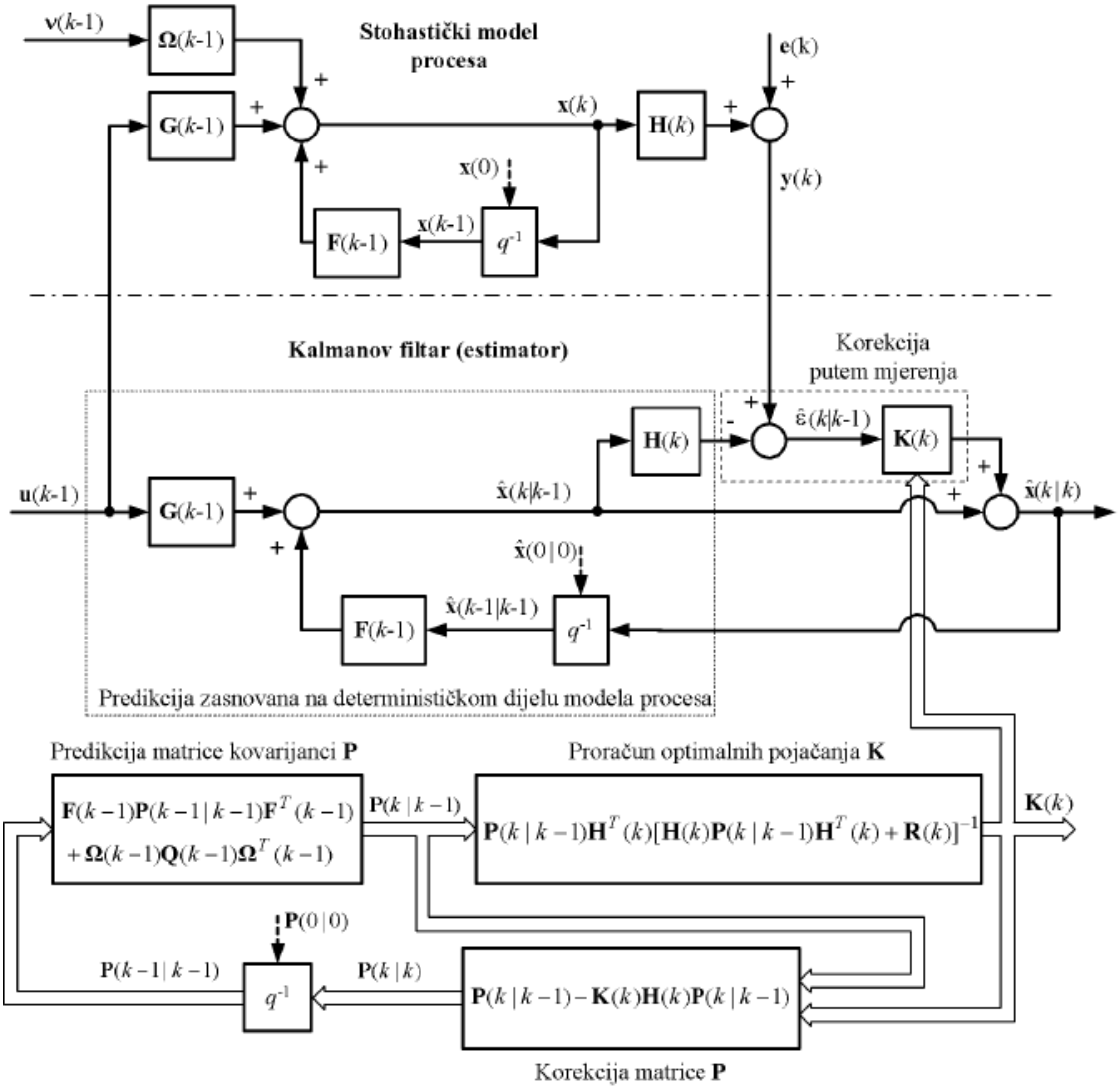
$P(k \mid k)$ - korigirana procjena matrica kovarijanci pogreški procjene stanja

A-posteriori procjena varijabli stanja $x(k \mid k)$ predstavlja procjenu očekivanja varijabli stanja x(*k*) na temelju inicijalne (a-priori) procjene $x(k \mid k-1)$ i korekcije putem mjerenja. Pojačanja Kalmanovog filtra K(*k*) za korekciju a-priori procjene stanja $x(k \mid k-1)$ određuju se na temelju procjene matrice a-priori kovarijanci pogreške P(*k*|*k*–1) i modela mjerenja (matrice H(*k*) i varijanci šuma mjerenja R(*k*)). Uz poznate vrijednosti varijanci šuma mjerenja R(*k*) i varijanci perturbacija stanja Q(*k*–1) iznosi pojačanja K(*k*) su takvi da je procjena stanja karakterizirana minimumom LQG kriterija optimalnosti:

$$\zeta = \left[x(k) - x(k \mid k)\right]^T \left[x(k) - x(k \mid k)\right] = \sum_{i=1}^{n} (x_i(k) - x_i(k \mid k))^2, \tag{24}$$

uz osigurano asimptotski stabilno vladanje Kalmanovog filtra.

Usporedni blokovski dijagrami stohastičkog modela procesa i Kalmanovog filtra prikazani su na slici 17. Kalmanov filtar može se podijeliti na algoritam procjene varijabli stanja (estimator) i proračun optimalnih pojačanja. Proračun pojačanja **K**(*k*) potpuno je neovisan o procjeni stanja, dok sam estimator predstavlja identičnu kopiju determinističkog dijela modela procesa proširenu odgovarajućom korekcijom po mjerenjima. Kalmanov filtar može se također promatrati kao takozvana prediktor-korektor struktura, jer se trenutni iznosi varijabli stanja i matrica kovarijanci inicijalno računaju predikcijom na temelju modela procesa iz vrijednosti dobivenih u prošlom koraku, te se naknadno korigiraju putem mjerenja, odnosno pojačanja **K**(*k*).



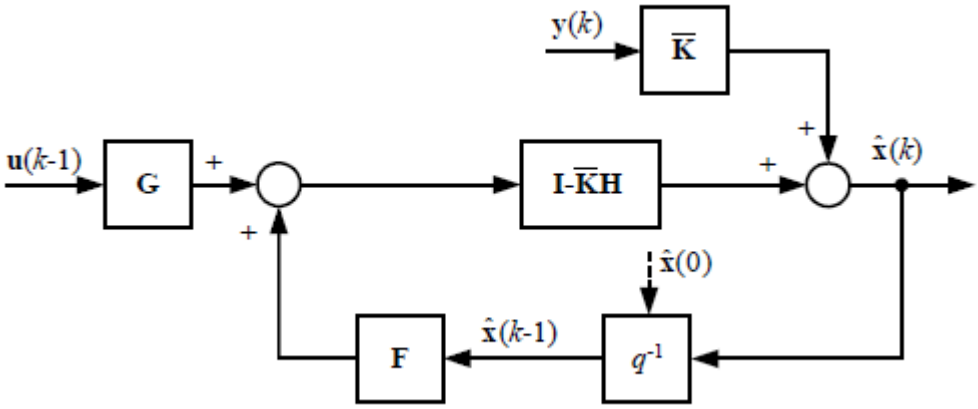
Slika 17. Usporedba blokovskih dijagrama stohastičkog modela procesa i odgovarajućeg Kalmanovog filtra kao sustava procjene varijabli stanja

4.1.3. Stacionarni Kalmanov filtar

Ukoliko stohastički proces ima konstantne parametre (vremenski neovisne matrice **F**, **G**, **H** i Ω), te ukoliko su stohastičke perturbacije v i šum mjerenja **e** stacionarnog karaktera (konstantne varijance **Q** i **R**), matrica kovarijanci **P**(*k*|*k*) i matrica pojačanja Kalmanovog filtra **K**(*k*) konvergiraju prema konstantnim stacionarnim vrijednostima $\overline{P}$ i $\overline{K}$ kada $k \to \infty$. Uvođenjem supstitucija $x(k) \equiv x(k \mid k)$ i $x(k-1) \equiv x(k-1 \mid k-1)$, i sređivanjem blokovskog dijagrama estimatora varijabli stanja sa slike 12 dobije se blokovski dijagram vremenski invarijantnog (stacionarnog) Kalmanovog filtra na slici 18 (**I** – jedinična matrica). Konačni izraz za procjenu varijabli stanja stacionarnog Kalmanovog filtra nakon sređivanja glasi:

$$x(k) = (I - \overline{K}H)Fx(k-1) + (I - \overline{K}H)Gu(k-1) + \overline{K}y(k) \tag{25}$$

odnosno, stacionarni Kalmanov filtar ima strukturu sličnu klasičnom (Luenbergerovom) estimatoru stanja.



Slika 18. Blokovski dijagram stacionarnog Kalmanovog filtra

Procjena izlaznih varijabli se računa prema sljedećem izrazu:

$$y(k) = Hx(k) \tag{26}$$

4.1.4. Podešavanje parametra Kalmanovog filtra

Izbor varijanci šuma mjerenja **R**(*k*) i varijanci perturbacija stanja **Q**(*k*-1) izravno utječe na matricu pojačanja **K**(*k*), odnosno na korekciju a-priori procjene stanja $x(k \mid k-1)$. Kako je u većini primjena varijance šuma mjerenja **R**(*k*) razmjerno lako ocijeniti iz samih mjerenja **y**(*k*), proizlazi da varijance stohastičkih perturbacija u stanjima (definirane matricom **Q**(*k*–1)) određuju podešenje Kalmanovog filtra. Ukoliko iznosi varijanci u matrici **Q**(*k*–1) točno odgovaraju varijancama pojedinih komponenti stohastičkih perturbacija u stanjima v(*k*–1), tada će a-posteriori procjena stanja $x(k \mid k)$ biti karakterizirana minimumom varijance odstupanja u odnosu na očekivane vrijednosti stanja procesa **x**(*k*). Međutim, u većini praktičnih primjena perturbacije u stanjima nisu mjerljive ili ih nije moguće procijeniti na jednostavan način, pa se elementi matrice **Q**(*k*–1) odabiru kao kompromis između većeg oslanjanja na mjerenja i bolje točnosti slijeđenja varijabli stanja (veći apsolutni iznosi članova matrice **Q**), i povoljnog odnosa signal/šum u procijenjenim stanjima (koji se dobiju uz manje iznose matrice **Q**).

Početne vrijednosti vektora procjene stanja $x(0\,|\,0)$ utječu na procjenu stanja na početku slijeđenja, dok pogreška početnog iznosa matrice kovarijanci **P**(0|0) utječe na proračun optimalnih pojačanja **K**(*k*) nakon pokretanja Kalmanovog filtra. Uslijed navedenog javlja se takozvana tranzijentna pogrešku slijeđenja, te je stoga potrebno početne vrijednosti $x(0\,|\,0)$ i **P**(0|0) postaviti čim bliže stvarnima. Početni iznosi varijabli stanja i matrice kovarijanci pogreške procjene stanja ne utječu na stabilnost Kalmanovog filtra.

4.2. Prošireni oblik Kalmanovog filtra

Klasični Kalmanov filtar nije pogodan za procjenu varijabli stanja ukoliko je stohastički model procesa nelinearan (odnosno ukoliko postoji međusobna nelinearna veza između varijabli stanja **x**, ulaznih veličina **u,** ili stohastičkih varijabli v i **e**). Pretpostavlja se da nelinearni model procesa ima sljedeću strukturu:

$$x(k) = f(x(k-1), u(k-1), v(k-1)) \tag{32}$$

$$y(k) = h(x(k), e(k)) \tag{33}$$

gdje su **f** i **h** vektorske funkcije koje su neprekinuto diferencijabilne na danim područjima definicije (**f** i **h** su "glatke" funkcije). Za procjenu varijabli stanja nelinearnog modela procesa opisanog izrazima (32) i (33) može se primijeniti prošireni oblik Kalmanovog filtra, čija se sinteza zasniva na sljedećoj linearnoj aproksimaciji nelinearnog modela procesa oko procijenjenih varijabli stanja:

$$x_0(k) = f(x(k-1\,|\,k-1), u(k-1), 0) \tag{34}$$

$$x(k) = x_0(k) + F(k-1)(x(k-1) - x(k-1\,|\,k-1)) + \Omega(k-1)v(k-1) \tag{35}$$

$$y(k) = h(x_0(k), \underline{0}) + H(k)(x(k) - x_0(k)) + \Psi(k)e(k) \tag{36}$$

gdje su matrice prvih parcijalnih derivacija **F**, **Ω**, **H** i **Ψ** (Jakobijan matrice) definirane na sljedeći način:

$$F(k-1) = \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x(k-1|k-1)\\ u(k-1)\\ v=\underline{0}}} \tag{37}$$

$$\Omega(k-1) = \left.\frac{\partial f}{\partial v}\right|_{\substack{x=x(k-1|k-1)\\ u(k-1)\\ v=\underline{0}}} \tag{38}$$

$$H(k) = \left.\frac{\partial h}{\partial x}\right|_{\substack{x=x_0(k)\\ e=\underline{0}}} \tag{39}$$

$$\Psi(k) = \left.\frac{\partial h}{\partial e}\right|_{\substack{x=x_0(k)\\ e=\underline{0}}} \tag{40}$$

Prošireni oblik Kalmanovog filtra zasnovan na lineariziranom stohastičkom modelu opisan je sljedećim izrazima:

$$x(k\,|\,k-1) = f(x(k-1\,|\,k-1), u(k-1), \underline{0}) \tag{41}$$

$$\varepsilon(k\,|\,k-1) = y(k) - h(x(k\,|\,k-1), \underline{0}) \tag{42}$$

$$P(k\,|\,k-1) = F(k-1)P(k-1\,|\,k-1)F^T(k-1) + \Omega(k-1)Q(k-1)\Omega^T(k-1) \tag{43}$$

$$K(k) = P(k\,|\,k-1)\,H^T(k)\left[H(k)\,P(k\,|\,k-1)\,H^T(k) + \Psi(k)R(k)\Psi^T(k)\right]^{-1} \tag{44}$$

$$P(k\,|\,k) = P(k\,|\,k-1) - K(k)\,H(k)\,P(k\,|\,k-1) \tag{45}$$

$$x(k\,|\,k) = x(k\,|\,k-1) + K(k)\,\varepsilon(k\,|\,k-1) \tag{46}$$

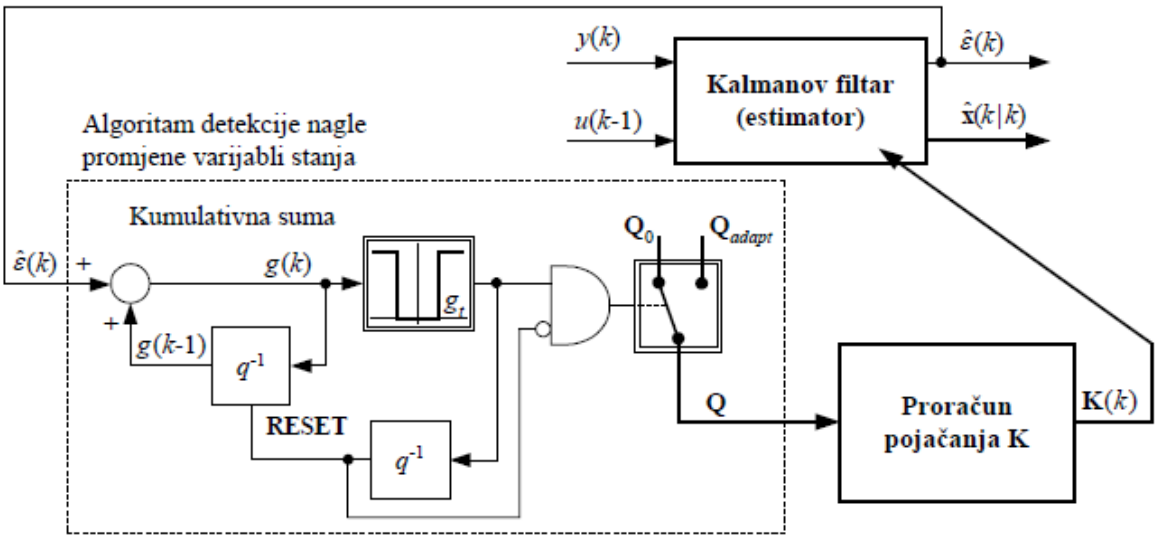
Izrazi (41) do (46) imaju sličan oblik kao i izrazi za klasični Kalmanov filtar s razlikom da su inicijalna (a-priori) procjena varijabli stanja $x(k \mid k-1)$ i procjena predikcijske pogreške $\varepsilon(k \mid k-1)$ dobivene iz determinističke aproksimacije nelinearnog modela procesa danog izrazima (32) i (33), uz v = 0 i **e** = 0. Propagacija matrice kovarijanci pogreške procjene stanja **P** i proračun optimalnog pojačanja **K** također se odvija na sličan način kao kod klasičnog Kalmanovog filtra. Osnovni problem kod primjene proširenog oblika Kalmanovog filtra jest da transformacije varijabli stanja **x** i perturbacijskih varijabli **e** i v nisu linearne, pa pripadajuće vjerojatnosne gustoće raspodjele više nemaju Gaussovski karakter. Uslijed toga, procjena varijabli stanja neće biti optimalna u LQG smislu (odnosno predstavljat će podoptimalno rješenje procjene stohastičkih varijabli stanja).

4.3. Adaptacijski mehanizam

Prema analizi provedenoj u potpoglavlju 4.1 kvaliteta slijeđenja varijabli stanja ovisi o izboru parametara matrice **Q** (pojačanja Kalmanovog filtra), odnosno u kojoj mjeri se procjena varijabli stanja oslanja na korekciju putem mjerenja. Podešenje Kalmanovog filtra najčešće predstavlja kompromis između dobrog slijeđenja varijabli stanja i povoljnog odnosa signala i šuma. Međutim, ukoliko bi bilo moguće detektirati nagle promjene varijabli stanja (na osnovi mjernih signala i procjene varijabli stanja), tada bi se privremeno mogli povećati iznosi pojačanja Kalmanovog filtra (tijekom trajanja prijelazne pojave) i ostvariti visoka točnost slijeđenja, dok bi u stacionarnom stanju Kalmanov filtar i dalje bio karakteriziran dobrim potiskivanjem šuma u procijenjenim stanjima.
U svrhu poboljšanja slijeđenja naglih promjena u varijablama stanja, Kalmanov filtar se može proširiti adaptacijskim mehanizmom zasnovanim na detekciji naglih promjena varijabli stanja. Pritom se adaptacija može zasnivati na proračunu kumulativne sume pogreške predikcije estimatora $\varepsilon(k)$, koji za skalarni slučaj (jedan izlazna varijabla), glasi:

$$g(0) = 0$$
$$g(k) = g(k-1) + \varepsilon(k) \tag{47}$$

U slučaju da je pogreška predikcije $\varepsilon(k)$ bliska bijelom šumu, iznos kumulativne sume $g(k)$ je blizak nuli. Međutim, ukoliko se dogodi iznenadna promjena jedne ili više varijabli stanja, predikcijska pogreška $\varepsilon(k)$ poprima razmjerno velike apsolutne iznose (Kalmanov filtar je podešen za dobro prigušenje šuma, odnosno razmjerno sporo slijeđenje varijabli stanja), što će zauzvrat rezultirati povećanjem iznosa kumulativne sume $g(k)$. Prema tome, ukoliko apsolutna vrijednost kumulativne sume $|g(k)|$ prijeđe neki prethodno definirani prag okidanja g_t, tada je vjerojatno došlo do nagle promjene u varijablama stanja i potrebno je provesti adaptaciju Kalmanovog filtra kako bi se poboljšala kvaliteta slijeđenja varijabli stanja.
Kao pogreška predikcije $\varepsilon(k)$ najčešće se koristi a-posteriori pogreška $\varepsilon(k \mid k) = y(k) - H(k)x(k \mid k)$ za koju se može uzeti da je bliska bijelom šumu (te bi trebala rezultirati malim iznosima perturbacija kumulativne sume $g(k)$ u stacionarnom stanju). Osnovni nedostatak primjene a-posteriori pogreške je inherentno kašnjenje adaptacije za jedan korak uzorkovanja jer se $\varepsilon(k \mid k)$ računa tek nakon korekcije stanja putem mjerenja. Kako bi se izbjeglo nepotrebno kašnjenje adaptacije može se umjesto a-posteriori pogreške koristiti a-priori pogreška $\varepsilon(k \mid k-1)$ (proračunata prije korekcije putem mjerenja), te tako adaptirati pojačanja Kalmanovog filtra u istom koraku kada je detektirana nagla promjena varijabli stanja.



Slika 19. Principni blokovski dijagram adaptacije Kalmanovog filtra

Principni blokovski dijagram mehanizma adaptacije Kalmanovog filtra za slučaj skalarne predikcijske pogreške prikazan je na slici 19 (q^{-1} – operator jediničnog kašnjenja). Adaptacija Kalmanovog filtra (povećavanje iznosa pojačanja **K)** nakon detekcije nagle promjene provodi se povećanjem matrice kovarijanci **Q** s početnog iznosa $\mathbf{Q}_0$ na iznos $\mathbf{Q}_{adapt}$. U koraku uzorkovanja nakon detekcije elementi matrice **Q** se vraćaju na početne vrijednosti (impulsna adaptacija) i prethodno stanje kumulativne sume $g(k\text{-}1)$ se postavlja na nulu (resetira se), te se nakon toga nastavlja proračun kumulativne sume. Valja uočiti da iako je adaptacija matrice **Q** provedena impulsno (samo u jednom koraku uzorkovanja), njena se promjena dinamički prenosi putem Riccatijeve jednadžbe (11) u iznose pojačanja Kalmanovog filtra **K** (izraz (12)). Stoga će pojačanja **K** zadržavati razmjerno visoke vrijednosti u odnosu na početne tijekom nešto duljeg vremenskog intervala zbog inherentne dinamike proračuna matrice kovarijanci **P**.

4.4. Estimator frekvencije i amplitude prinudnih oscilacija

Za rješavane problema u diplomskom radu potrebno je izraditi Kalmanov filter koji estimira frekvenciju i amplitudu prinudnih oscilacija regulacijskog keuga s relejnim članom. Polazni model procesa za estimaciju rezonantne frekvencije je model slobodnog oscilatora:

$$\frac{d^2x}{dt^2} + \Omega^2 x = 0 \tag{49}$$

gdje su:

x – diferencirani izlaz iz procesa
Ω – rezonantna frekvencija

Ovaj model se diskretizira u vremenu primjenom Eulerove aproksimacije $s \approx (z-1)/T$, te se dobije sljedeća jednadžba diferencija za izlaz modela:

$$x(k) = 2x(k-1) - x(k-2) - T^2\Omega^2 x(k-2) \tag{50}$$

$$\Delta_2 x(k) = x(k) - 2x(k-1) + x(k-2) = -T^2\Omega^2 x(k-2) \tag{51}$$

Na temelju dobivene jednadžbe diferencija implementira se adaptivni Kalmanov filtar u obliku sljedećih rekurzivnih jednadžbi:

$$P(k) = P(k-1) - K(k-1)H(k-1)P(k-1) + Q(k-1) \tag{52}$$

$$H(k) = \frac{\partial x(k)}{\partial \Omega^2(k)} = -T^2 x(k-2) \tag{53}$$

$$K(k) = \frac{P(k)H(k)}{P(k)H^2(k) + R(k)} \tag{54}$$

$$e(k) = \Delta_2 x(k) - H(k)\Omega^2(k-1) = \Delta_2 x(k) + T^2 x(k-2)\Omega^2 x(k-1) \tag{56}$$

$$\Omega^2(k) = \Omega^2(k-1) + K(k)e(k) \tag{57}$$

$$\Omega(k) = \sqrt{\Omega^2(k)} \tag{58}$$

gdje je *x*(*k*) = *y*(*k*) – *y*(*k*-1) (*y(k)* je izlazna veličina procesa)*. Adaptacija Kalmanovog filtra provodi se impulsno, to jest na početku auto-tuning eksperimenta se nakratko poveća iznos kovarijance Q (pojačanja K) da ubrza odziv estimirane veličine (za razliku od pristupa opisanog u potpoglavlju 4.3).

Do amplitude izlaznog signala može se doći na sljedeći način (slika 20):

1. Izlazni signal treba provući kroz pojasno propusni filtar kojemu je prijenosna funkcija:

$$G_{PP}(s) = \frac{2 \cdot \xi \cdot \Omega \cdot s}{s^2 + 2 \cdot \xi \cdot \Omega \cdot s + \Omega^2} \tag{59}$$

Vremenski diskretni filtar dobije se supstitucijom $s \approx (z-1)/T$ (Eulerova aproksimacija)

Izlaz iz vremenski diskretnog pojasno propusnog filtra je dan sljedećem jednadžbom diferencija:

$$y_{PP}(s) = -a_{PP1} \cdot y_{PP}(k-1) - a_{PP2} \cdot y_{PP}(k-2) + b_{PP1}(y(k) - y(k-1)) \tag{60}$$

gdje su:

$$a_{PP1} = (2 \cdot \xi \cdot \Omega \cdot T - 2) \tag{61}$$

$$a_{PP2} = (1 - 2 \cdot \xi \cdot \Omega \cdot T + \Omega^2 \cdot T^2) \tag{62}$$

$$b_{PP1} = 2 \cdot \xi \cdot \Omega \cdot T \tag{63}$$

$$\xi = \frac{\sqrt{2}}{2}$$

Ω - rezonantna frekvencija, koja je dobivena ranije objašnjenim postupkom

T – vrijeme uzrokovanja

y(k) – izlazni signal procesa

2. Sljedeći je korak dobiti signal sa samo pozitivnim vrijednostima, a izlaz je dan sljedećom jednadžbom:

$$y_{abs} = |y_{PP}| \tag{64}$$

3. Ako se sad taj signal propusti kroz niskopropusni filtar, dobije se srednja vrijednost signala, pomoću koje se može dobiti amplituda signala. Prijenosna funkcija niskopropusnog filtra je:

$$G_{NP}(s) = \frac{\Omega_1^2}{s^2 + 2 \cdot \xi \cdot \Omega_1 \cdot s + \Omega_1^2} \tag{65}$$

* Diferenciranje ulaza u Kalmanov filtar (izlazne veličine procesa) radi se u svrhu korektne interpretacije signala i izbjegavanja estimacije DC posmaka (engl. DC offset). Valja također uočiti da se estimira kvadrat frekvencije kako bi se izbjegla linearizacija modela procesa, odnosno primjena proširenog oblika Kalmanovog filtra.

Izlaz iz vremenski diskretnog niskopropusnog filtra dan je sljedećom jednadžbom diferencija:

$$y_{NP}(s) = -a_{NP1} \cdot y_{NP}(k-1) - a_{NP2} \cdot y_{NP}(k-2) + b_{NP1} \cdot y_{abs} \quad (66)$$

gdje su:

$$a_{NP1} = (2 \cdot \xi \cdot \Omega_1 \cdot T - 2) \quad (67)$$

$$a_{NP2} = (1 - 2 \cdot \xi \cdot \Omega_1 \cdot T + \Omega_1^2 \cdot T^2) \quad (68)$$

$$b_{NP1} = \Omega_1^2 \cdot T^2 \quad (69)$$

$$\xi = 1$$

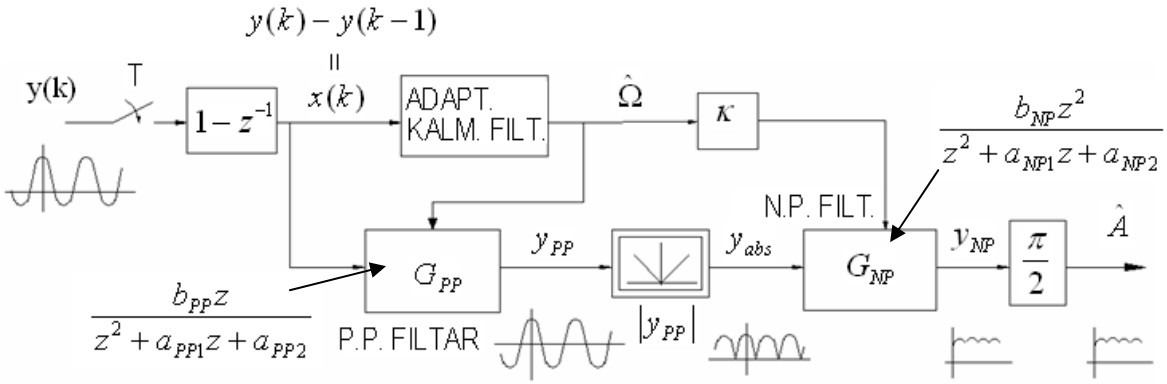
$$\Omega_1 = 0.25 \cdot \Omega$$

Ω - rezonantna frekvencija, koja je dobivena ranije objašnjenim postupkom
T – vrijeme uzrokovanja

4. Sad kad je poznata srednja vrijednost signala, amplituda se dobije iz sljedeće jednadžbe:

$$A = y_{NP} \cdot \frac{\pi}{2} \quad (70)$$

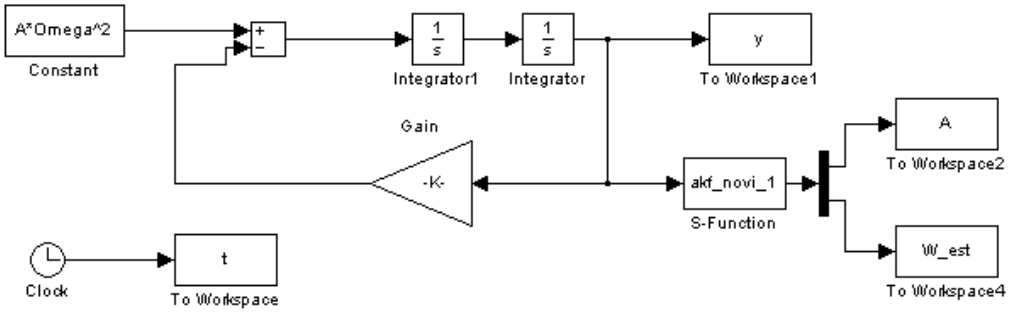
Na slici 20 prikazan je blokovski dijagram predloženog algoritma estimacije.



Slika 20. Blokovski dijagram estimatora frekvencije i amplitude

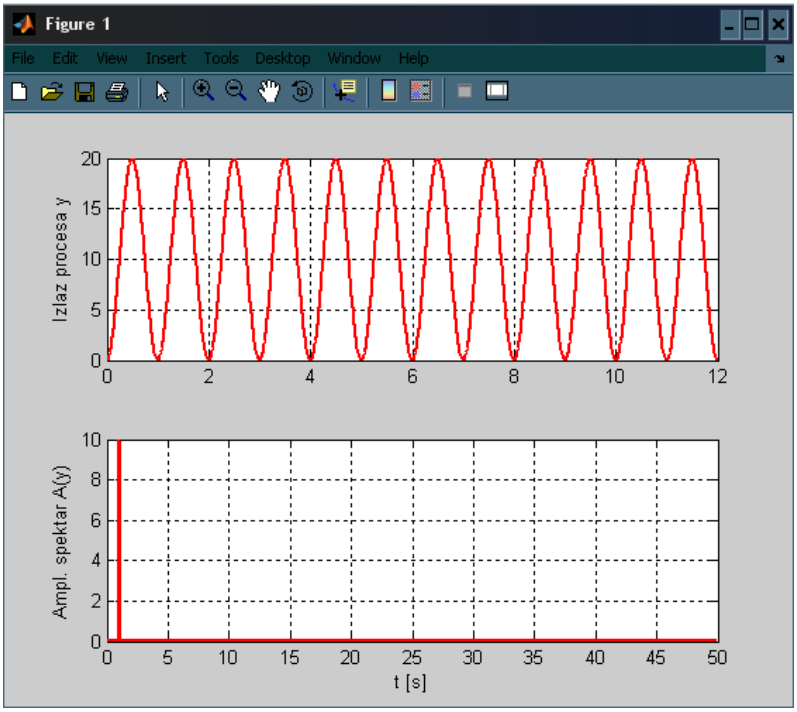
4.4.1. Simulacijski rezultati

Simulacijska ispitivanja su se provela za dva slučaja, prvi slučaj je sa sinusnim signalom, a za drugi slučaj je uzet relejni eksperiment s PT_n modelom procesa. Simulacijska shema za slučaj sa sinusnim signalom prikazana je na slici 21.



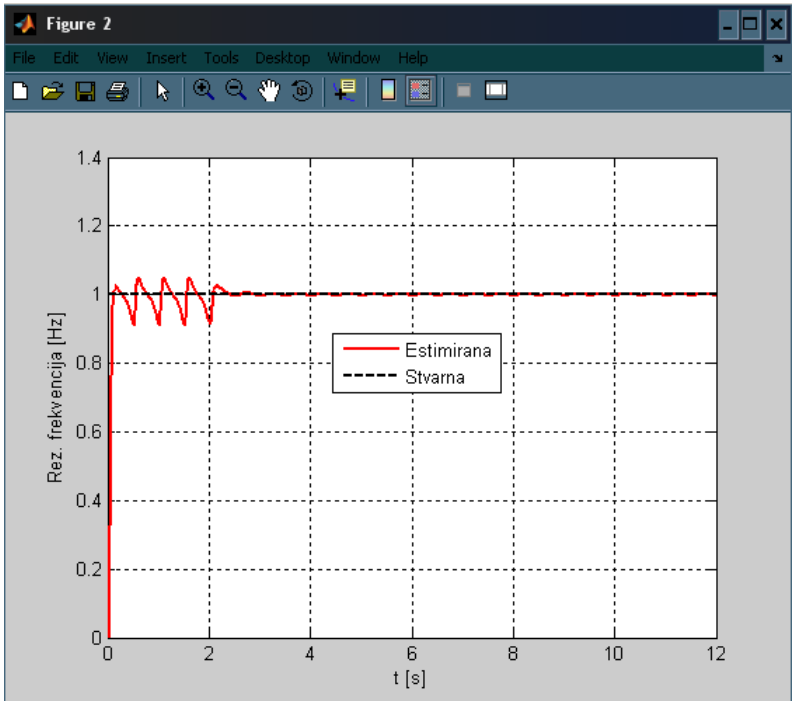
Slika 21. Simulacijski sustav sa sinusnim signalom i Kalmanovim filtrom kao estimatorom frekvencije i amplitude

Izlaz iz oscilatora je harmonička (sinusna) funkcija koja osim amplitude (A = 10) i frekvencije (Ω= 1 Hz), te posjeduje i posmak iznosa 10.

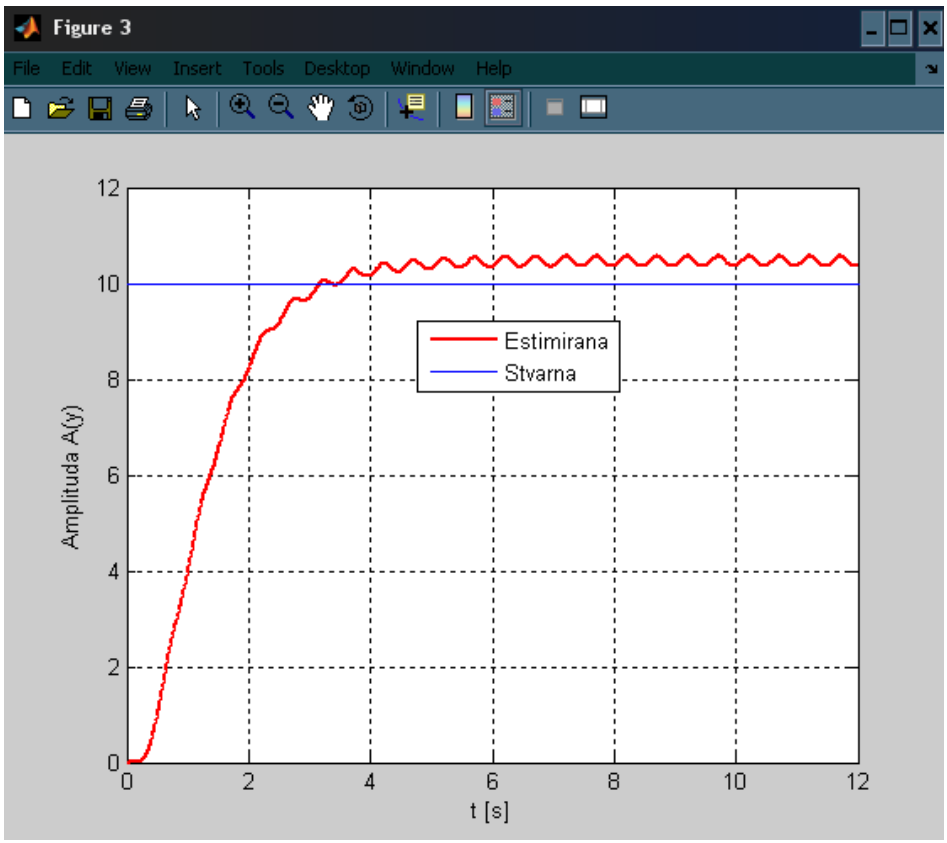


Slika 22. Izlaz procesa

Rezultat estimacije rezonantne frekvencije je prikazan na slici 23., a estimacija amplitude je prikazana na slici 24. Adaptivni Kalmanov filtar točno estimira rezonantnu frekvenciju Ω = 1 Hz, dok estimat amplitude A vrlo malo odstupa od stvarne vrijednosti (pogreška manja od 5%). Razlog odstupanja estimata amplitude može biti neidealna karakteristika nisko propusnog filtra u području propuštanja odnosno nedovoljno gušenje visoko frekvencijske komponente (eng. ripple) koja se uočava u iznosu estimirane amplitude (Slika 24.).



Slika 23. Rezonantna frekvencija



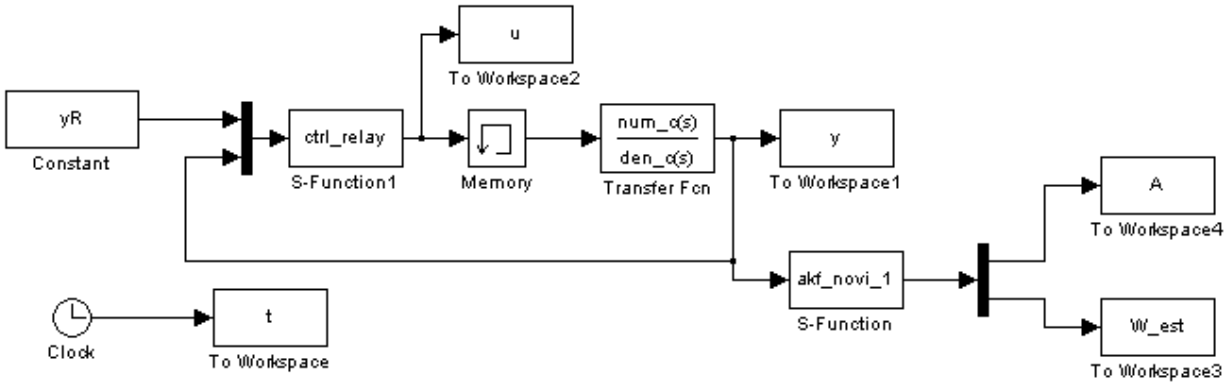
Slika 24. Amplituda

Simulacijska shema za slučaj relejnog eksperimenta i PT_n modela prikazana je na slici 25., pritom je korišten sljedeći model procesa:

$$G_P(s) = \frac{K_P}{(1+T_P s)^n} \tag{71}$$

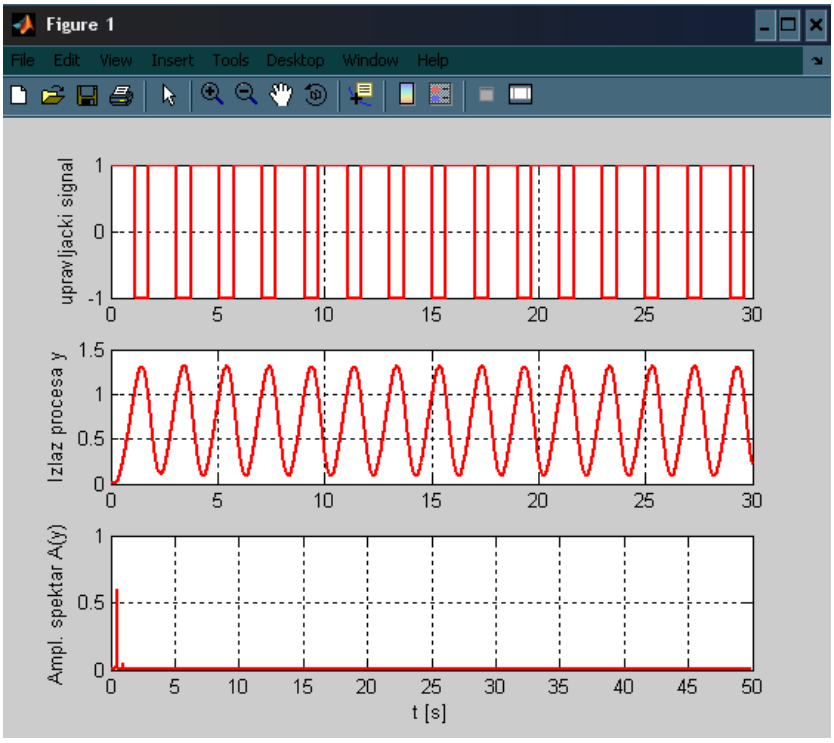
gdje su:

$K_P = 2.0$
$T_P = 0.3$
$n = 4$

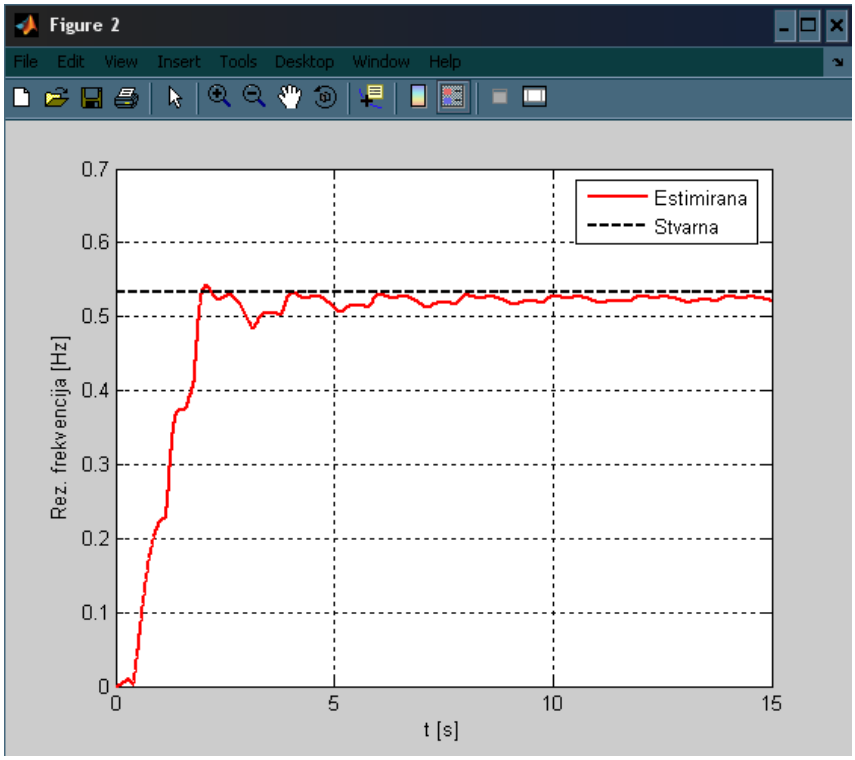


Slika 25. Simulacijski sustav PT_n modela procesa sa Kalmanovim filtrom kao estimatorom frekvencije i amplitude

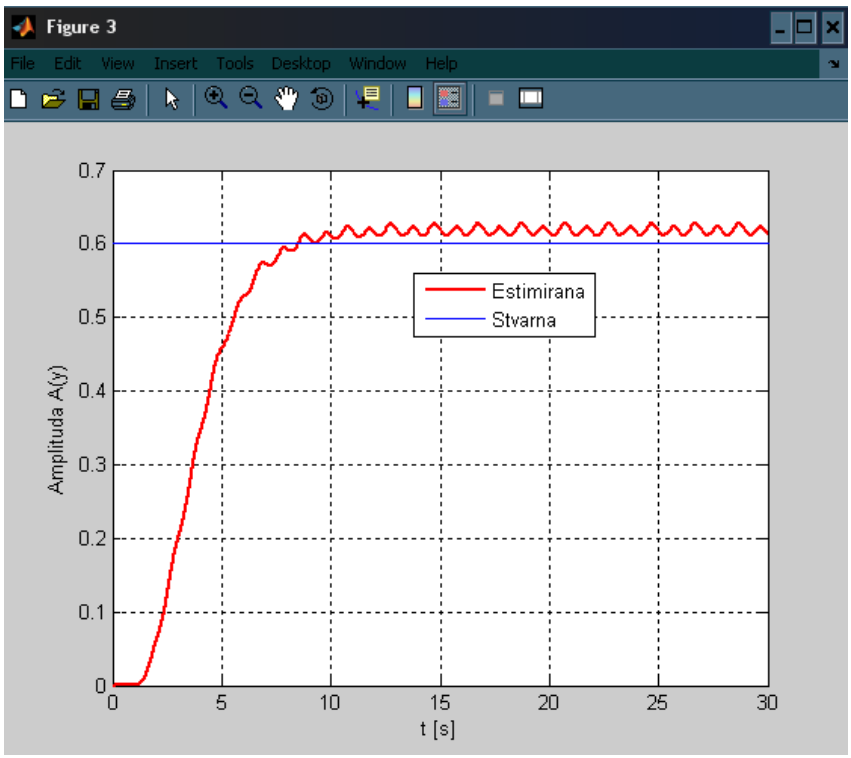
Iz slike 26. vidljivo je da regulacijski krug s relejnim regulatorom (dvopoložajni relej) izaziva pojavu vlastitih oscilacija ujednačene amplitude i frekvencije, odnosno dovodi sustav na rub stabilnosti. Adaptivni Kalmanov filtar korektno estimira frekvenciju i amplitudu osnovnog harmonika vlastitih oscilacija (Slike 27. i 28.), te se dobiveni iznosi parametara vlastitih oscilacija mogu primjeniti za sintezu PI ili PID regulatora prema Takahashi-evoj metodi dovođenja na rub stabilnosti (Poglavlje 3.4.).



Slika 26. Upravljački i izlazni signal



Slika 27. Rezonantna frekvencija



Slika 28. Amplituda

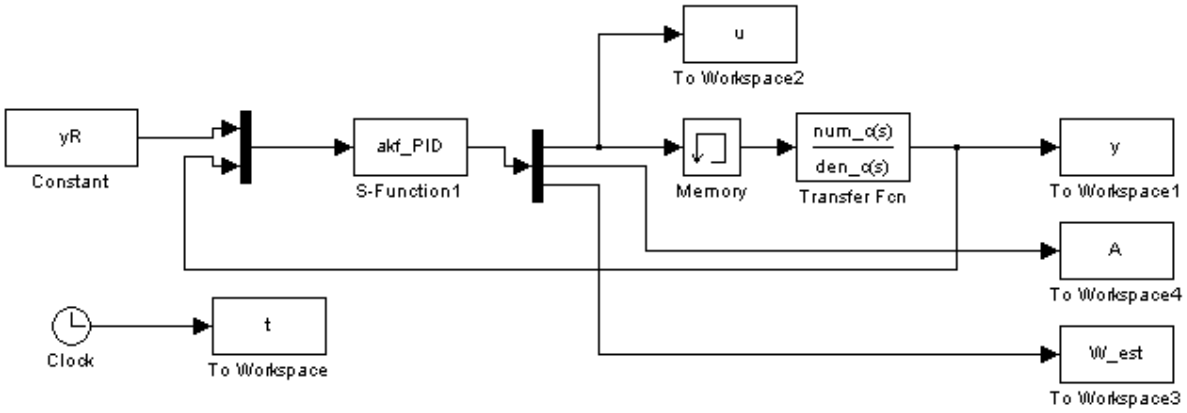
Stvarna vrijednost amplitude dobije se primjenom frekvencijske analize zasnovane na FFT algoritmu [7], algoritam je prikazan u prilogu B4.

5. Simulacijski rezultati

Potrebno je realizirati auto-tuning algoritam zasnovan na estimaciji frekvencije i amplitude vlastitih oscilacija pobuđenih relejnim članom te na temelju nadomjesnog pojačanja relejnog člana i dobivenih iznosa amplitude i frekvencije rubnih oscilacija provesti sintezu regulatora prema Ziegler-Nicholsu (Takahashiu).

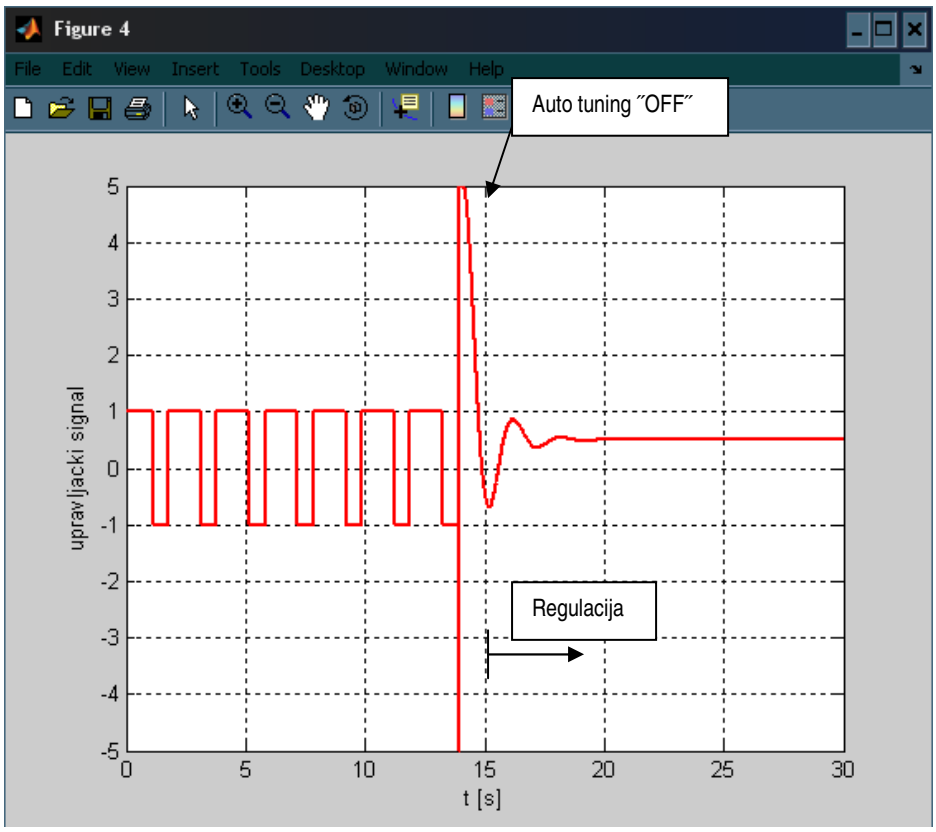
5.1. Ilustracija rada auto tuninga PID regulatora za PT_n model procesa

Auto tuning PID regulatora za PT_n model procesa dan je u prilogu B5, kao C-mex funkcija, a njen rad je provjeren u Simulink okruženju. Regulacijska shema prikazana je na slici 29.

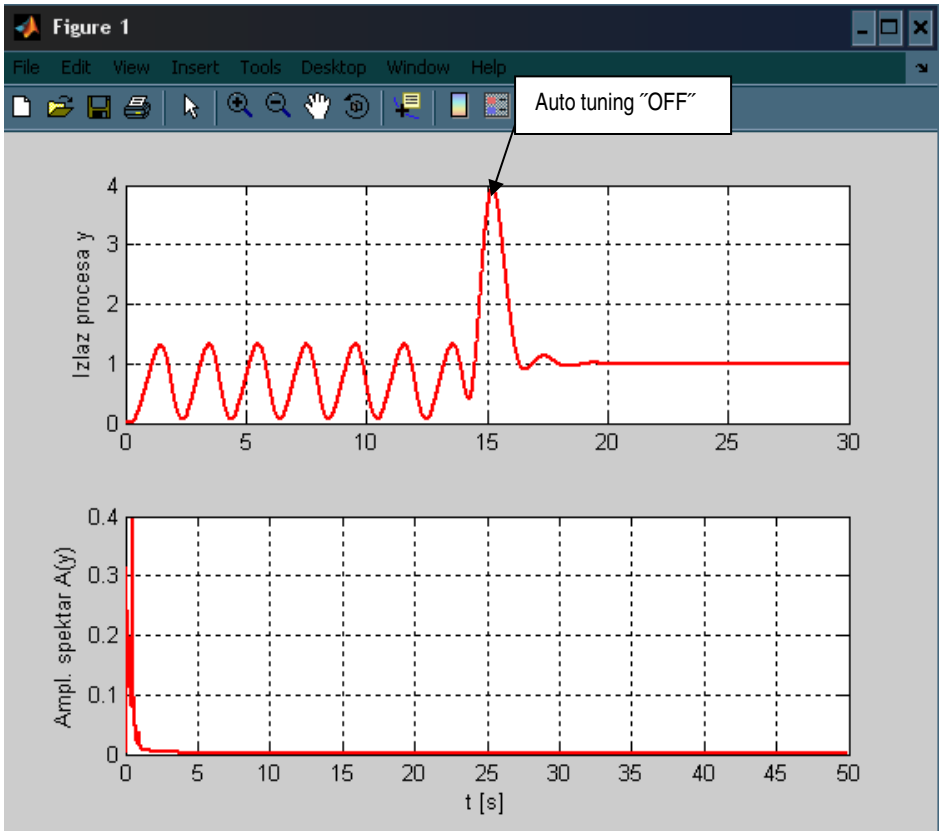


Slika 29. Regulacijska shema PT_n model procesa i auto tuning PID regulatora

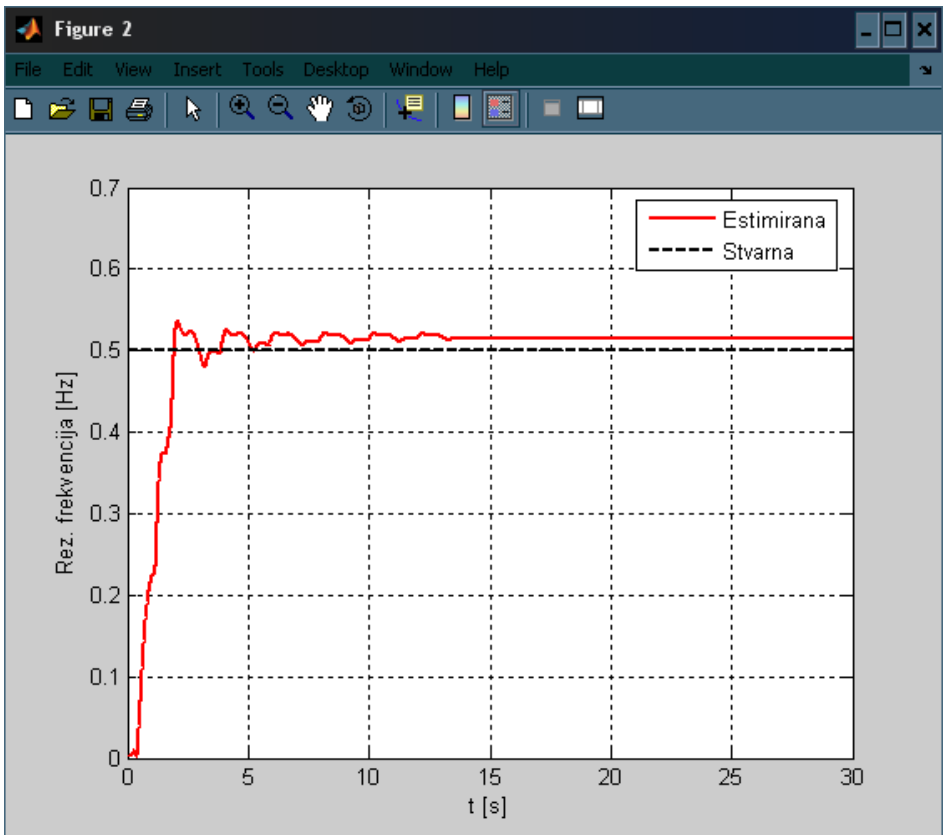
Auto tuning je realiziran tako da na se na početku sustav dovede na rub stabilnosti pomoću relejnog člana, te se iz rubnih oscilacija Kalmanovim filtrom estimiraju rezonantna frekvencija i amplituda. Nakon 15 sekundi isključuje se relejni član i uključuje PID regulator koji je podešen prema Ziegler-Nicholsovim (Takahashievim) preporukama.
Rezulatati simulacije su dani na slikama 30., 31., 32. i 33. Ponovno je vidljivo da adaptivni Kalmanov filtar točno estimira ključne parametre graničnih oscilacija sustava (amplitudu i vlastitu frekvenciju). Nakon prebacivanja na PID regulator postoji izražen tranzijent u upravljačkom signalu (u) i izlaznom signalu (y) zbog neprilagođenih početnih stanja integratora i derivatora u PID regulatoru. No PID regulator ipak brzo smiruje odziv bez značajnih istritavanja (odziv nije pretjerano oscilatoran).



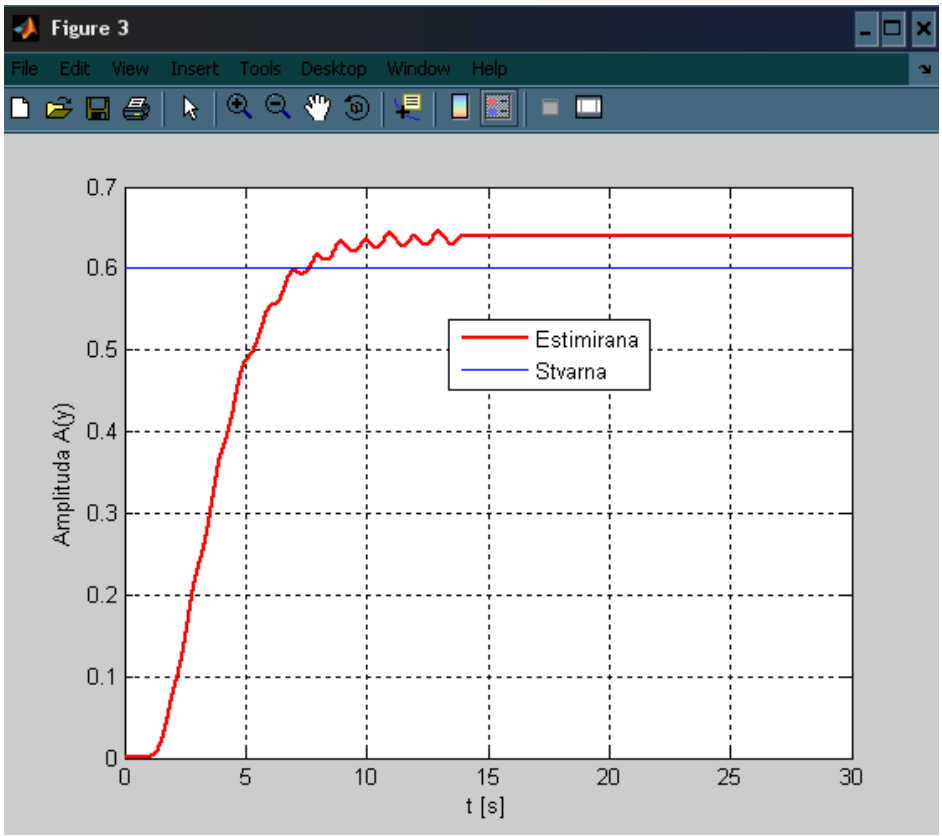
Slika 30. Upravljački signal



Slika 31. Izlazni signal



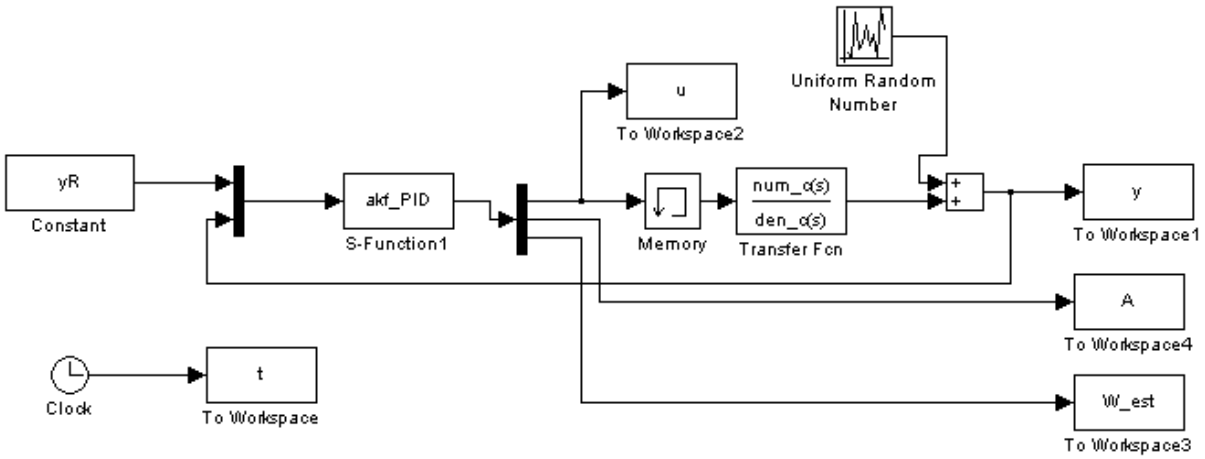
Slika 32. Rezonantna frekvencija



Slika 33. Amplituda

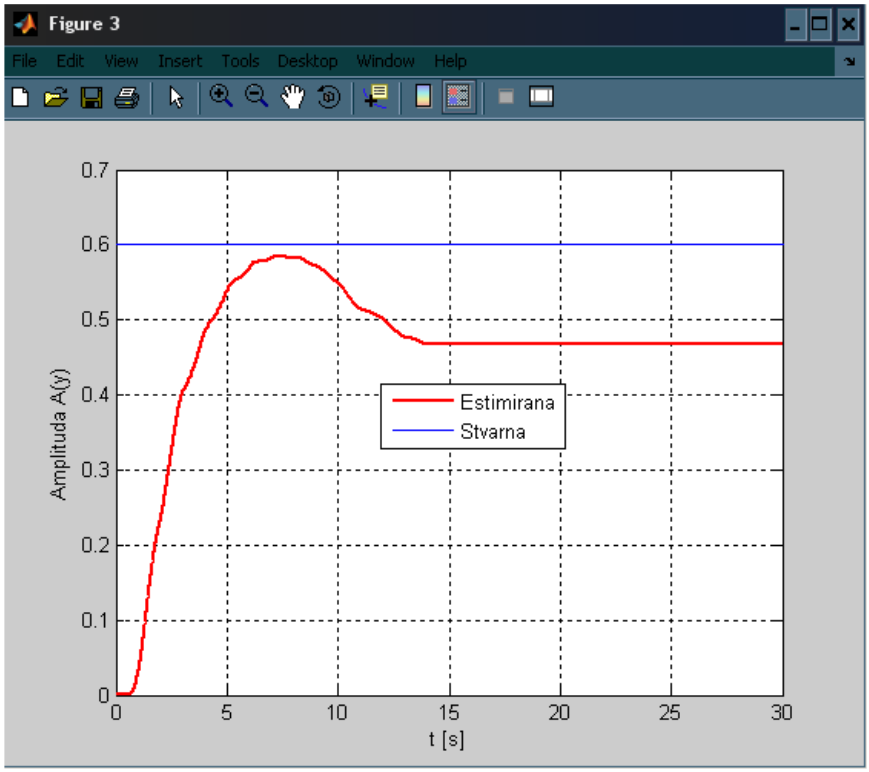
5.1.1. Utjecaj šuma mjerenja

Potrebno je provjeriti utjecaj šuma mjerenja na auto-tuning regulatora, te navesti rješenje kako se taj problem može otkloniti. Utjecaj možemo provjeriti u Simulink rješenju sa algoritmom auto tuninga PID regulatora prikazanog u poglavlju 5.1. Utjecaj šuma mjerenja možemo provjeriti tako da se doda stohastičko djelovanje u izlazni signal (u ovom slučaju vrijednost perturbacija je uzeta $\pm 0.1\%\ y_R$). Blokovski dijagram (simulacijski model) regulacijskog kruga s oscilatornim djelovanjem na izlazni signal prikazan je na slici 34.

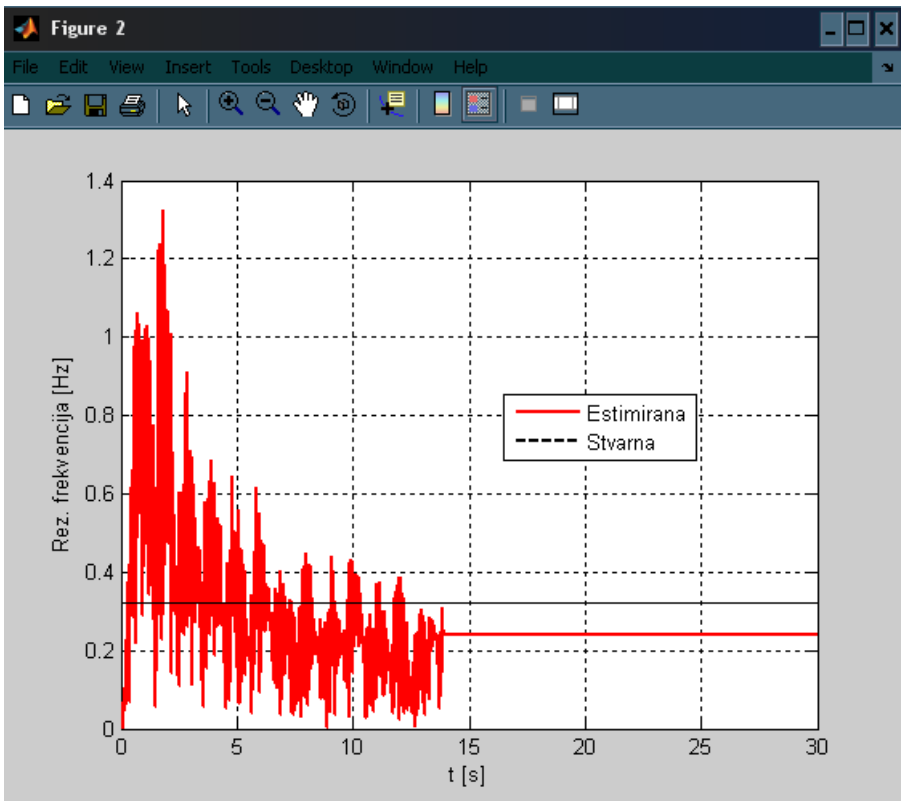


Slika 34. Regulacijska shema za provjeru utjecaja šuma mjerenja

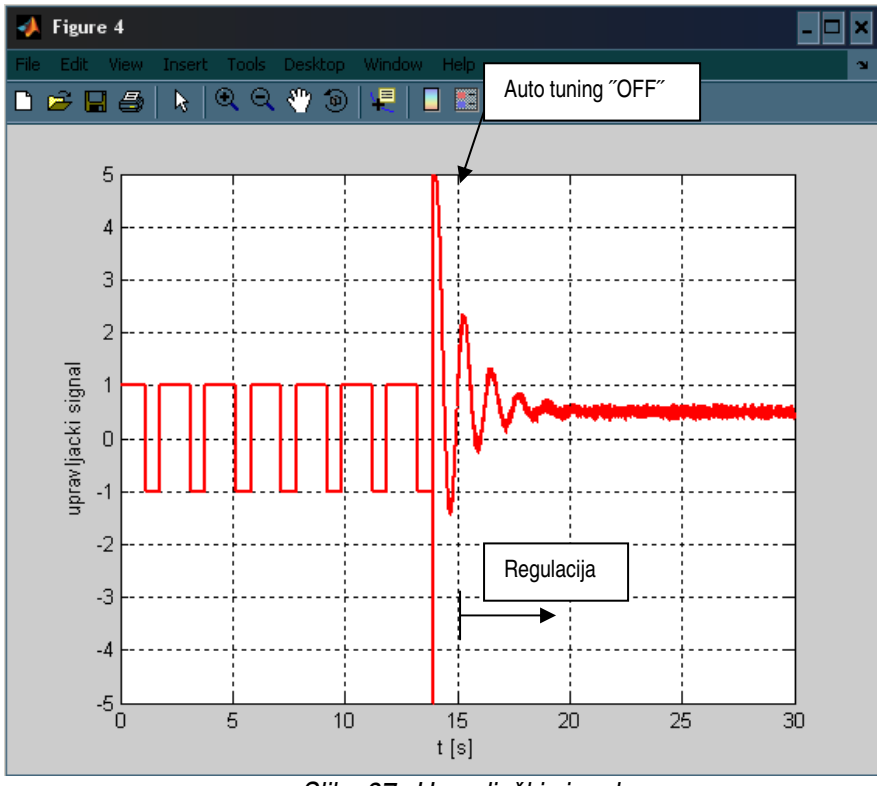
Rezultati simulacije dani su na slikama 35., 36., 37. i 38. U slučaju naglašenog šuma mjerenja, taj šum se izravno manifestira u pogrešci estimacije amplitude i vlastite frekvencije prinudnih oscilacija (Slike 35. i 36.), te se može dogoditi da na kraju auto-tuninga testa, podešenje PID regulator ne bude zadovoljavajuće kvalitete (pojava naglašenih oscilacija).



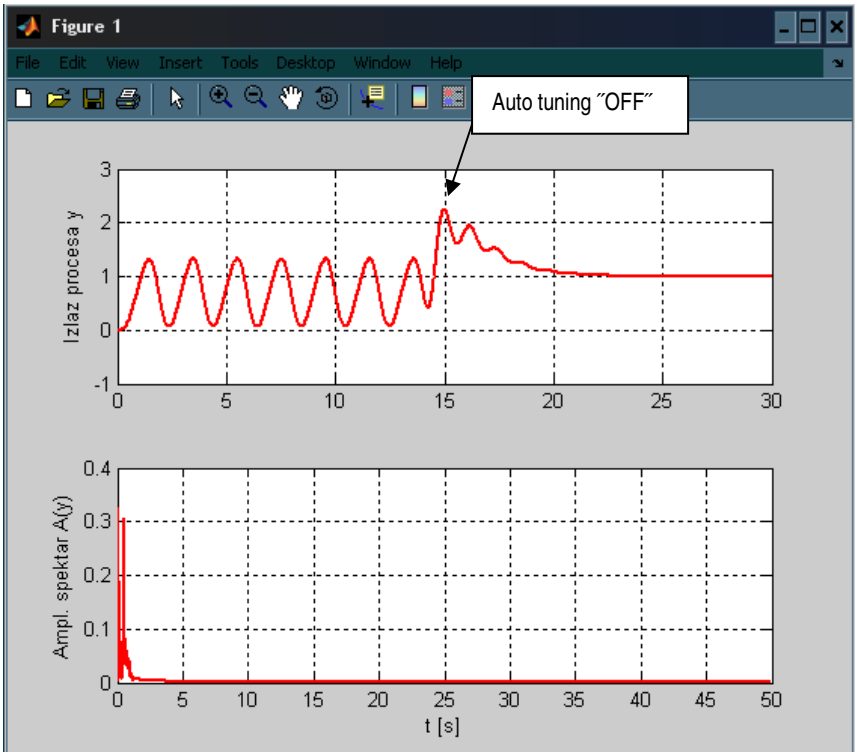
Slika 35. Amplituda



Slika 36. Rezonantna frekvencija



Slika 37. Upravljački signal



Slika 38. Izlazni signal

Uočljivo je da PID regulator značajno pojačava šum mjerenja, pretežno zbog derivacijskog djelovanja. Ovo može imati određene neželjene posljedice po izvršni član (pojačalo grijača). Naime, veliki šum na ulazu puhala utječe na visokofrekvencijske pojave u pojačalu snage (pojačalo se više zagrijava) bez značajnog efekta na izlaznu toplinsku snagu puhala.

Da bi se problem utjecaja šuma mjerenja riješio, potrebno je provući ulazni signal u Kalmanov filtar kroz niskopropusni filtar. Prijenosna funkcija niskopropusnog filtra je:

$$G_{NPA}(s) = \frac{\Omega_{NPA}^2}{s^2 + 2 \cdot \xi \cdot \Omega_{NPA} \cdot s + \Omega_{NPA}^2} \tag{72}$$

Izlaz iz niskopropusnog filtra dan je sljedećom jednadžbom:

$$y_{NPA}(s) = -a_{NPA1} \cdot y_{NPA}(k-1) - a_{NPA2} \cdot y_{NPA}(k-2) + b_{NPA1} \cdot y \tag{73}$$

gdje su:

$$a_{NPA1} = (2 \cdot \xi \cdot \Omega_{NPA} \cdot T - 2) \tag{74}$$

$$a_{NPA2} = (1 - 2 \cdot \xi \cdot \Omega_{NPA} \cdot T + \Omega_{NPA}^2 \cdot T^2) \tag{75}$$

$$b_{NPA1} = \Omega_{NPA}^2 \cdot T^2 \tag{76}$$

$$\xi = \frac{\sqrt{2}}{2}$$

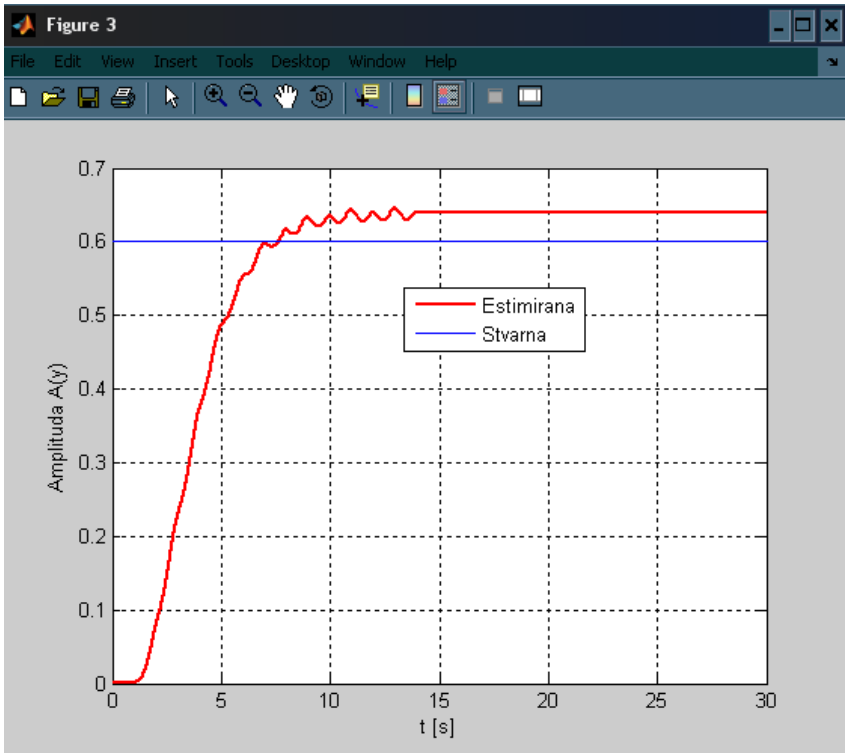
$$\Omega_{NPA} = \frac{1}{2\sqrt{2} \cdot T} \tag{77}$$

T – vrijeme uzrokovanja

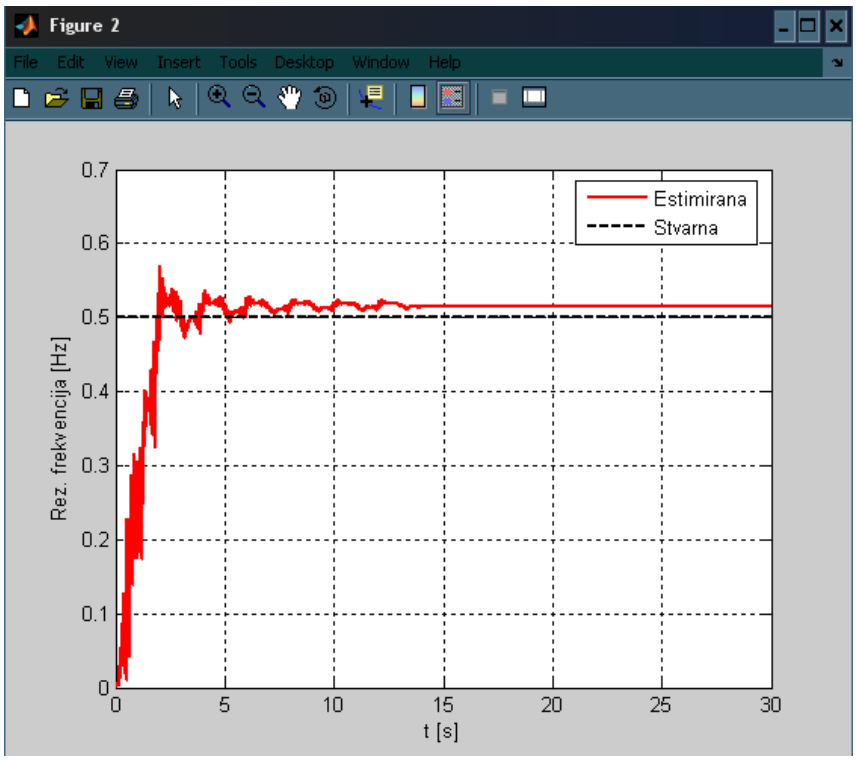
Ovim izborom se, između ostalog , dani filtar projektira i s obzirom na potiskivanje aliasing efekta (Ω_{NPA} se postavlja s obzirom na vrijeme uzorkovanja).

Simulacijski model je identičan modelu prokazanom na slici 34. Rezultati simulacije pokazuju da je sada utjecaj šuma na kvalitetu estimacije amplitude i frekvencije vlastitih oscilacija bitno smanjen, te je i podešenje PID regulatora bitno točnije (slično podešenju koje se dobilo kada nije bilo šuma u izlaznom

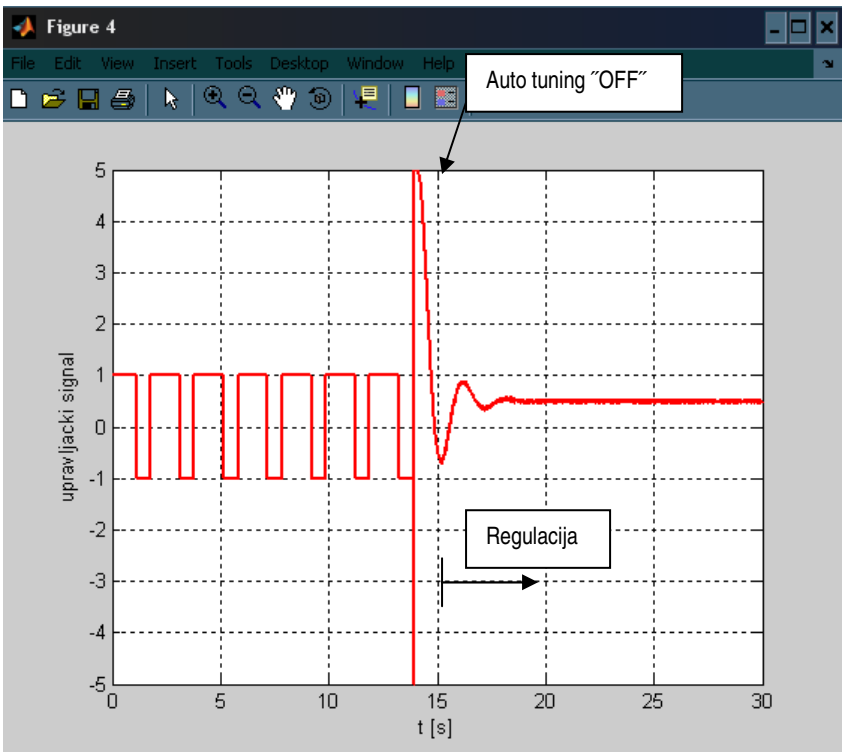
signalu (y) iz procesa). Sukladno tome, u nastavku rada će se koristiti gore navedeni anti-aliasing niskopropusni filtar.



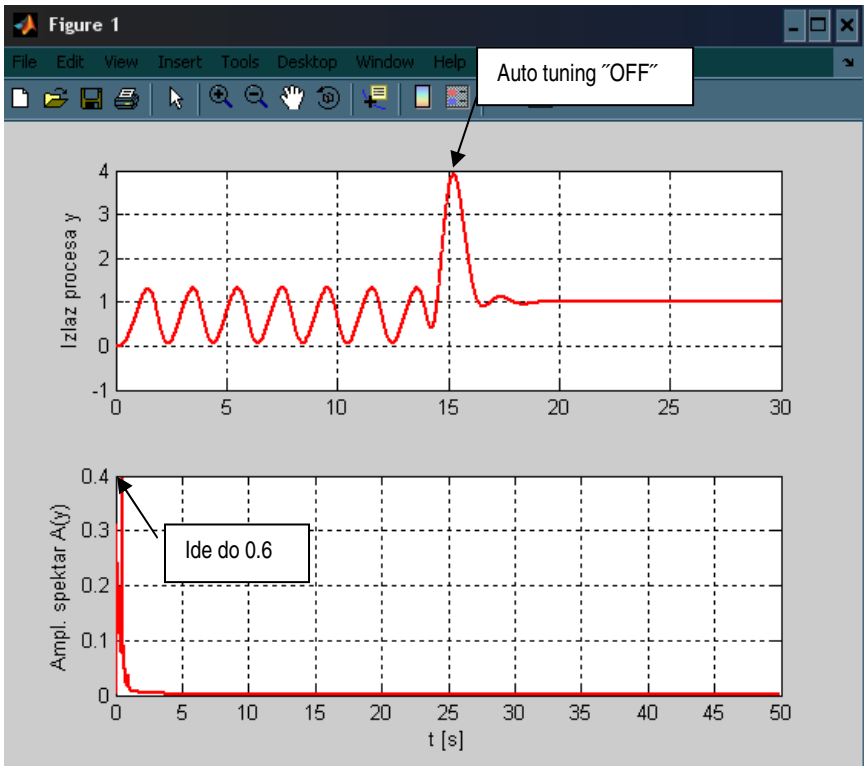
Slika 39. Amplituda



Slika 40. Rezonantna frekvencija



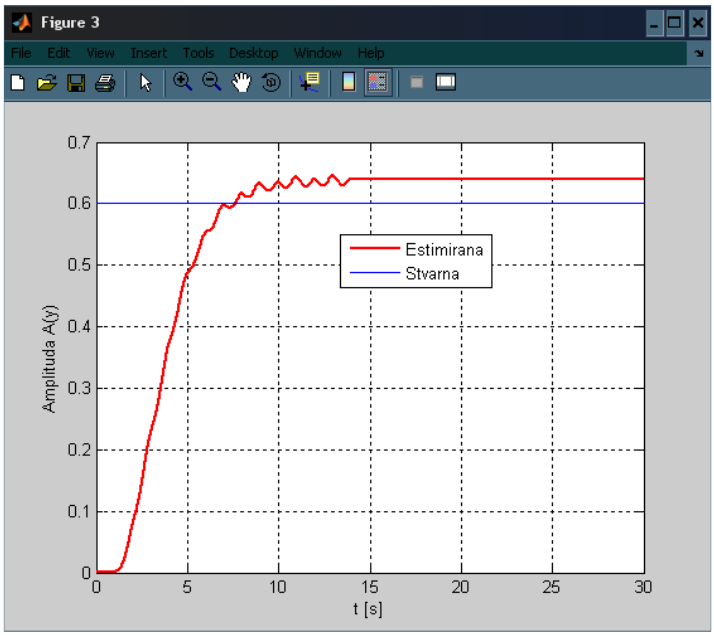
Slika 41. Upravljački signal



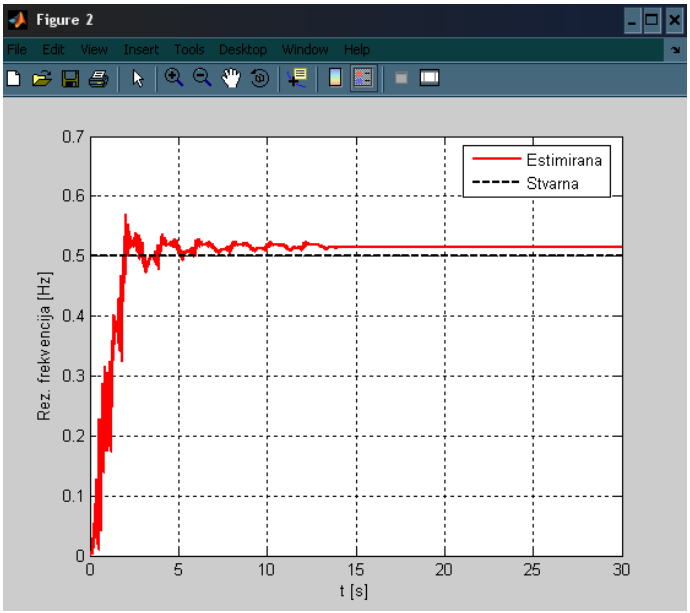
Slika 42. Izlazni signal

5.2. Ilustracija rada auto tuninga PI regulatora za PT_n model procesa

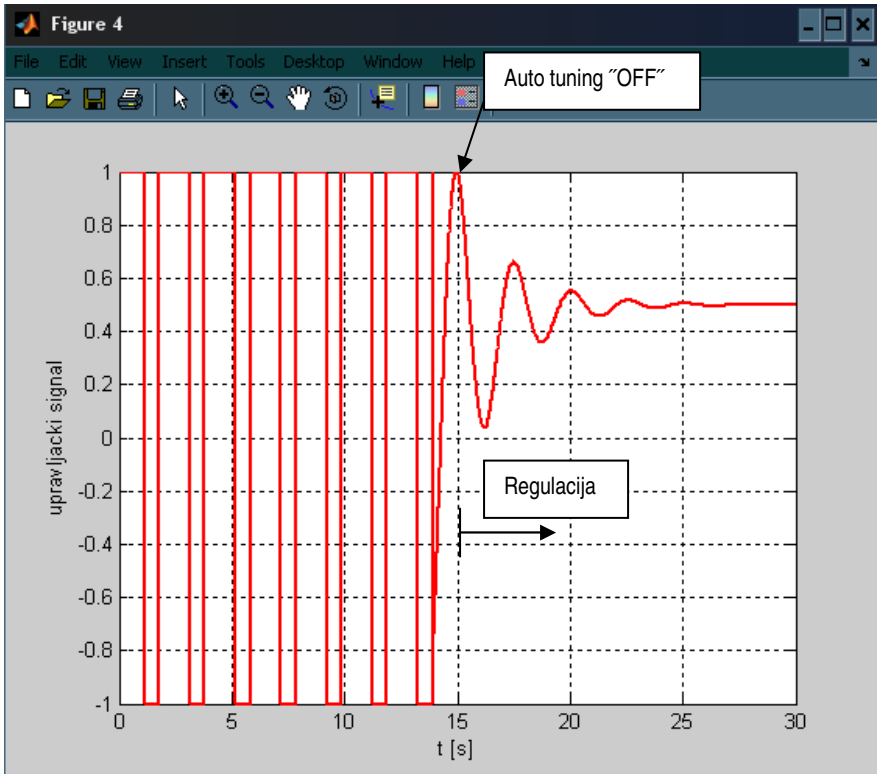
Postupak auto tuninga je isti kao i kod slučaja PID regulatora, samo je u ovom slučaju derivacijsko djelovanje regulatora jednako nuli. Rezultati simulacije s PI regulatorom i uključenim anti-aliasing filtrom prikazani su na slikama 43., 44., 45. i 46. Prema očekivanjima, estimirana amplituda i frekvencija prinudnih oscilacija ne odstupaju značajno od stvarnih vrijednosti (dobivenih frekvencijskom analizom). Međutim, PI regulator je karakteriziran nešto slabijim prigušenjem (većom oscilatornošću) odziva u usporedbi s PID regulatorom. Naime, PID regulator je karakteriziran dodatnim stupnjem slobode podešavanja dinamike odziva (preko derivacijskog člana), te ga je u osnovi moguće podesiti za povoljniju kvalitetu odziva (iako to ne mora uvijek biti slučaj kod Takahashi-eve metode).



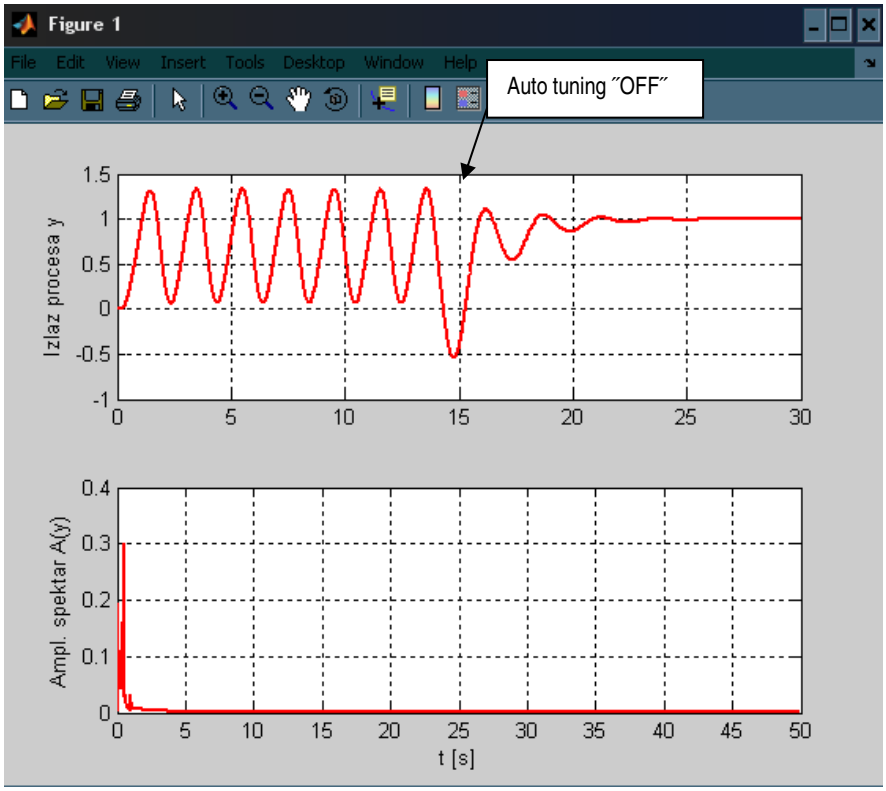
Slika 43. Amplituda



Slika 44. Rezonantna frekvencija



Slika 45. Upravljački signal



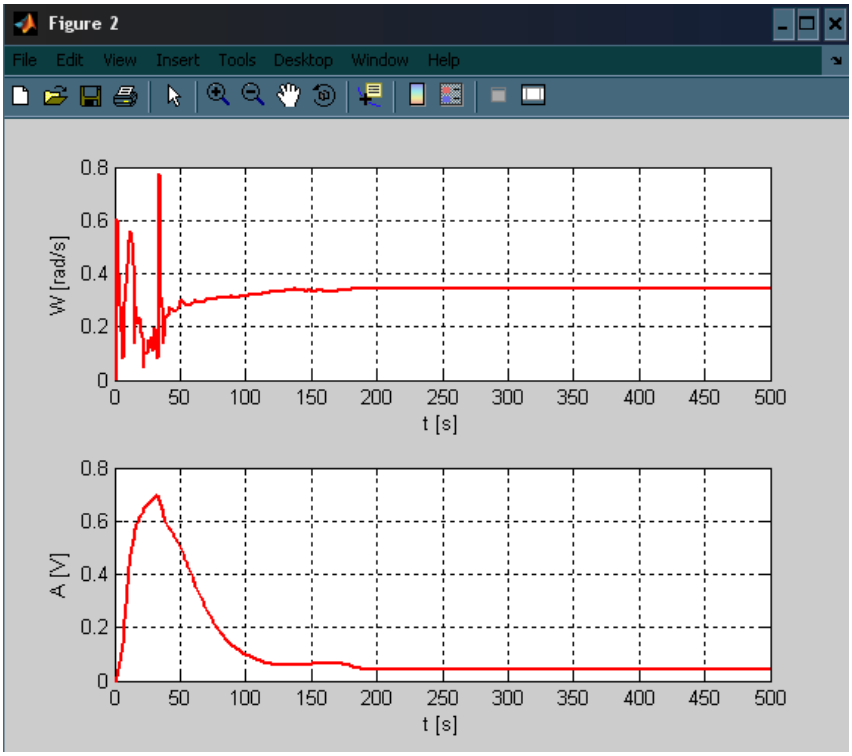
Slika 46. Izlazni signal

6. Eksperimentalna provjera

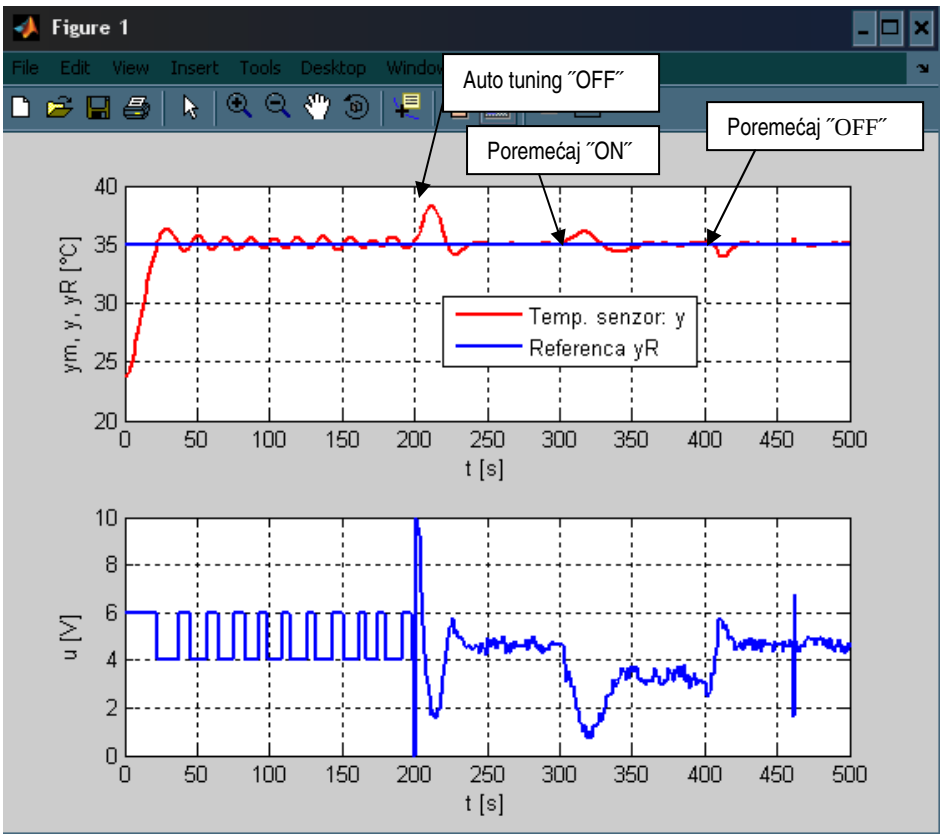
Eksperimentalna provjera regulacijskog sustava je provedena na nastavnoj maketi toplinske staze s regulacijom protoka i temperature koja je prikazana u poglavlju 2.2.

6.1. Eksperimentalna provjera PID regulatora

Eksperimentalna provjera regulacijskog sustava s PID regulatorom provedena je na nastavnoj maketi toplinske staze (Slika 3.). Referentna vrijednost temperature je iznosila 35 °C. Vrijeme uzrokovanja je uzeto, T_0 = 1 sekunde. Izlazni podaci su snimljeni u ASCII formatu i kao takvi su učitani u Matlab radi grafičkog prikaza.
Iz eksperimentalnih rezultata auto tuninga PID regulatora prikazanih na slikama 47. i 48. može se vidjeti da je PID regulator dobro podešen i da mu je potrebno vrlo kratko vrijeme smirivanja (oko 40 sekundi). Oscilacije izlaznog signala nisu velike, osim da dolazi do većeg nadvišenja kod prelaska regulacijskog sustava iz auto tuninga u rad sa PID regulatorom podešenim prema Ziegler-Nicholsovim (Takahashievim) preporukama. Također je dokazano da regulacijski sustav uspješno otklanja utjecaj vanjskog poremećaja (simuliranog uključivanjem pomoćnog grijača).



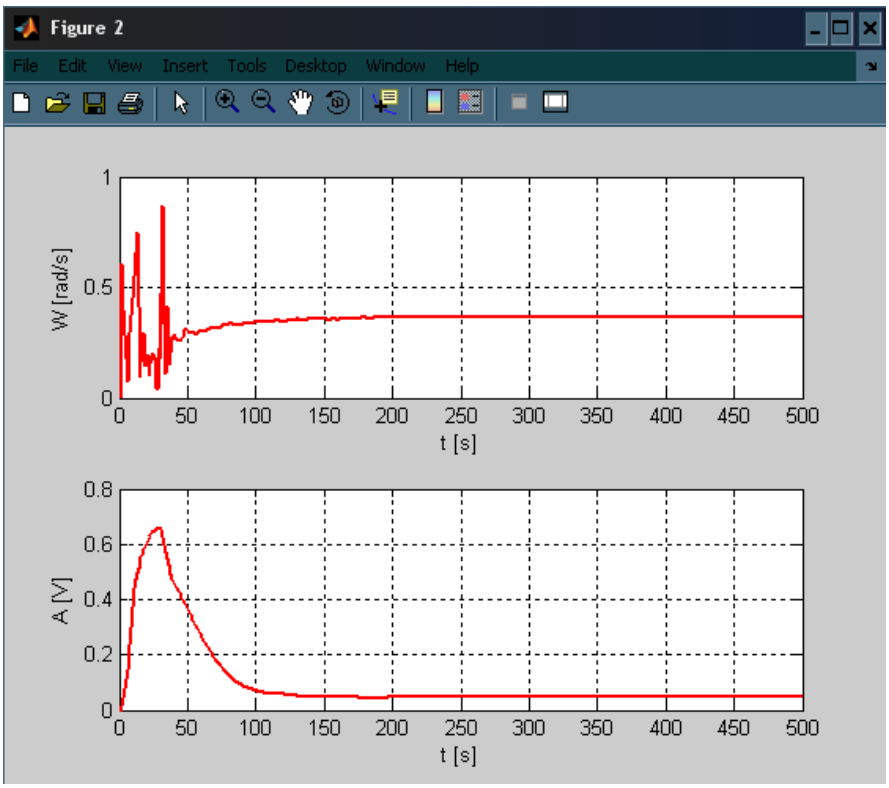
Slika 47. Rezonantna frekvencija i amplituda



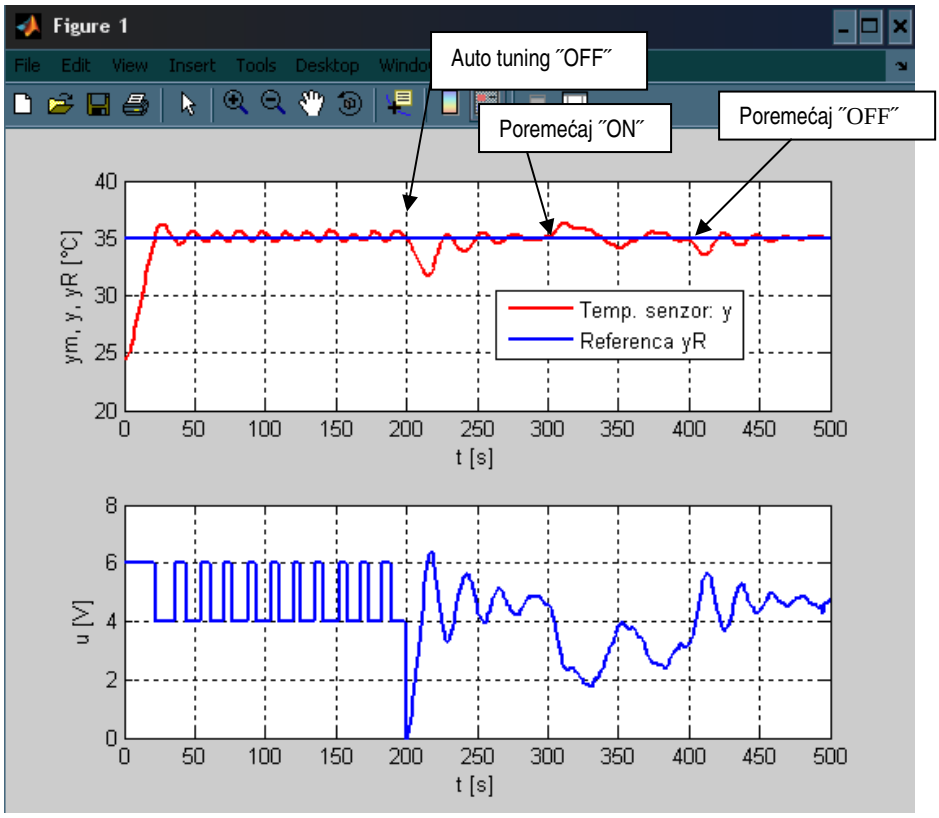
Slika 48. Izlazni i upravljački signal

6.2. Eksperimentalna provjera auto tuning PI regulatora

Eksperimentalna provjera regulacijskog sustava s PI regulatorom provedena je na jednak način kao u slučaju PID regulatora. Rezultati eksperimentalne provjere auto tuning PI regulatora prikazani su na slikama 49. i 50. Eksperiment je pokazao da regulacijski sustav sa PI regulatorom ima sporije i oscilatornije ponašanje od regulacijskog sustava sa PID regulatorom (usporedi odzive na slikama 48. i 50.). Regulacijski sustav sa PI regulatorom također uspješno otklanja vanjski poremećaj, ali mu za to treba nešto više vremena. Može se zaključiti da regulacijski sustav sa PID regulatorom isto kao i sa PI regulatorom uspješno dovodi temperaturu na željenu (referentnu) vrijednost, samo regulacijski sustav sa PID regulatorom ima brže ponašanje i bolju prigušenost odziva.



Slika 49. Rezonantna frekvencija i amplituda



Slika 50. Izlazni i upravljački signal

7. Zaključak

Regulacija temperature je čest problem u procesnoj tehnici. Ovakav projekt zahtijeva potrebna znanja iz područja kao što su: modeliranje, simulacija i regulacija.

Mnogi toplinski procesi imaju sporo aperiodsko ili čak naglašeno integracijsko vladanje. Za sintezu regulatora kod takvog vladanja ćesto se koristi Ziegler-Nicholsova (Takahashieva) metoda dovođenja na rub stabilnosti. Osnovna prednost tog postupka je jednostavan proračun parametara regulatora i značajna brzina odziva regulacijskog sustava, dok je glavni nedostatak taj da dolazi do većeg iznosa nadvišenja i vrlo često daje vrlo oscilatorno vladanje sustava regulacije.

Za sintezu parametara PI/PID regulatora potrebno je poznavanje kritičnog pojačanja (K_u) i perioda oscilacija (T_u). Zadatak auto tuninga je bio određivanje rezonatne frekvencije i amplitude izlaznog signala, što je pak moguće ako se regulacijski sustav regulira dvopoložajnim regulatorom, a kao estimator ključnih parametara stabilnih prinudnih oscilacija se koristi adaptivni Kalmanov filtar. Nakon toga se provede sinteza PI/PID regulatora prema Ziegler-Nicholsovim (Takahashievim) preporukama i počinje regulacija izlazne veličine procesa.

Iz simulacije se vidjelo da je odziv regulacijskog sustava vrlo brz, uz dosta veliko nadvišenje i oscilatorno ponašanje, što je bilo i za očekivati kod Ziegler-Nicholsovog (Takahashievog) postupka sinteze PI/PID regulatora. Iako i PI i PID regulator uspješno dovedu sustav na željenu (referentnu) vrijednost, vidljivo je da PID regulator to učini puno brže i sa manjim nadvišenjem i oscilacijama. Simulacije su pokazale da je regulacijski sustav sa PID regulatorom bolje rješenje od regulacijskog sustava sa PI regulatorom.

Eksperimentalna provjera na nastavnoj maketi toplinske staze je samo potvrdila rezultate dobivene simulacijom odabranog regulacijskog kruga. Iz toga se može zaključiti da je auto-tuning regulacijskog sustava sa PI/PID regulatorom temperature napravljen na zadovoljavajući način, te da dobiveni parametri regulatora daju dobro vladanje sustava. PID regulator opetovano daje bolje ponašanje sustava od PI regulatora u eksperimentalnim uvjetima.

Daljnje istraživanje može biti usmjereno prema mogučnosti primjene adaptivnog Kalmanovog filtra za estimaciju parametara drugih sustava karakteriziranih slabo prigušenim oscilacijama, te mogučnošću primjene naprednijih metoda podešavanja PI/PID regulatora.

8. Prikaz Matlab-ovih funkcija

Prilog B1. Adaptivni Kalmanov filtar za estimaciju frekvencije i amplitude

```
/*********************************/
/*                               */
/* Matlab ver. 5.3 (R11)         */
/* Matlab ver. 7.0 (R14)         */
/* C-mex S-Function              */
/*                               */
/*********************************/

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME akf_novi_1

#include "simstruc.h"    /* Defines SimStruct and corresponding macros */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Vrijeme uzorkovanja */
#define T   (double)(0.01)
/* Ludolfov broj */
#define     PI    (double)(3.1415926535898)

/* Parametri adaptivnog Kalmanovih filtara */
#define     Q1    (double)(1.0)
#define R1  (double)(1.0)
#define     MUL1  (double)(1.0e8)
#define     N1    (int)(200) // adaptacija traje 50 uzoraka
#define     THR   (double)(1.0e-6)
#define     E     (double)(0.707)
#define E1  (double)(1.0)

/*************************************************/

#define u(element) (*uPtrs[element])
#define NUM_OF_ARGS     0     /* Broj ulaznih argumenata - podaci
organizorani u redne vektore */
#define NUM_OF_IN 1     /* Broj ulaza:  1 - referenca, 2 - mjerenje */
#define NUM_OF_OUT      2     /* Broj izlaza: 1 - ur (izlaz regulatora)
*/
#define NUM_OF_REAL     0     /* Matlabova konvencija spremanja statickih
varijabli se !!ne koristi!! */
#define NUM_OF_PTR      0     /* Prije pocetka simulacije treba napraviti
novi mex-file!! */
#define NUM_OF_INT      0


/*********************************/
/* FUNCTION mdlInitializeSizes       */
/*********************************/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, NUM_OF_ARGS);  /* Ocekivani broj parametara
koji se prenosi preko dialog-boxa (0) */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
```

```
    }
    ssSetNumContStates(S, 0);   /* Broj vremenski-kontinuiranih stanja
(0) */
    ssSetNumDiscStates(S, 0);   /* Broj vremenski-diskretnih stanja (0)
*/

    /* Ne koristi se ... */
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, NUM_OF_IN);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, NUM_OF_OUT);
    ssSetNumSampleTimes(S, 1);   /* Jedinstveno vrijeme uzorkovanja 0.1
sek */

    ssSetNumRWork(S, NUM_OF_REAL);   /* Ne koriste se ... */
    ssSetNumIWork(S, NUM_OF_INT);
    ssSetNumPWork(S, NUM_OF_PTR);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
} /* end mdlInitializeSizes */

static void mdlInitializeSampleTimes(SimStruct *S)
{
      ssSetSampleTime(S, 0, T);
      ssSetOffsetTime(S, 0, 0);
}

/*====================================*/
/* FUNCTION mdlInitializeConditions  */
/*====================================*/
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
      if (ssGetSFcnParamsCount(S) != NUM_OF_ARGS)
      {
            #ifdef MATLAB_MEX_FILE
                  ssSetErrorStatus(S,"Wrong number of input args!");
            #else
                  printf("\nWrong number of input args!");
                  exit(0);
            #endif
      }

}
#endif

/*======================*/
/* FUNCTION mdlOutputs */
/*======================*/
/* Ovdje se obavlja svo racunanje */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T *y = ssGetOutputPortRealSignal(S,0);

      /* Definicije varijabli: referenca, mjerenje, naponska referenca,
proporcionalno djelovanje, derivativno djelovanje, regulacijsko
odstupanje */
      double x, a1, a2, b1, anp1, anp2, bnp1;
      // Varijable ad. Kalmanovog filtra
      double err1;
      static double H1 = 0.0;
```

```
static double S1 = 1.0;
static double x1e = 0.0;
static double K1 = 0.0;
  static double x1e_pred = 0.0;
  static double yf = 0.0;
  static double ynpf = 0.0;
  static double yf_pred1 = 0.0;
  static double yf_pred2 = 0.0;
  static double ynpf_pred1 = 0.0;
  static double ynpf_pred2 = 0.0;
  static double A = 0.0;
static double x1 = 0.0;

double D2y, den1;

static double dx0 = 0.0; /* diferencija x(k) - x(k-1) */
static double dx1 = 0.0; /* diferencija - prosla */
static double dx2 = 0.0; /* diferencija - pretprosla */
/* Proslo stanje izlazne velicine procesa */
static double x_ = 0.0;
/* Estimat frekvencije [rad/s] */
static double W_est = 0.0;
  static double W_est2 = 0.0;

/* Brojac koraka */
static unsigned int cnt = 0;

x = u(0); /* mjerenje */

/* Deriviranje ulazne velicine za uklanjanje DC posmaka */
  dx0 = x - x_;

/* spremanje proslog stanja x za novi korak */
x_ = x;

/*** Kalmanov filtar ***/
/* Izlaz "modela procesa" */
D2y = dx0 - 2.0*dx1 + dx2;

/* Adaptacija kovarijance */
if(cnt < N1){
     S1 = (1 - K1*H1)*S1 + MUL1*Q1;
  }
  else{
     S1 = (1 - K1*H1)*S1 + Q1;
}

/* Koeficijent izlazne jednadzbe */
H1 = -T * T * dx2;

/* Provjera da nazivnik nije blizak nuli */
  den1 = H1 * S1 * H1 + R1;

if(fabs(den1) < THR){
     den1 = THR;
  }

/* Adaptacija pojacanja */
K1 = S1 * H1 / den1;

/* Pogreska estimacije */
err1 = D2y - H1 * x1e_pred;

/* Osvjezavanje estimiranog stanja */
```

```
        x1e = x1e_pred + K1*err1;

      /* Apsolutna vrijednost (ne moze biti negativan) */
        x1e = fabs(x1e);
      x1e_pred = fabs(x1e_pred);
      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        dx2 = dx1;
        dx1 = dx0;
        x1e_pred = x1e;

        W_est = sqrt(x1e);

      /* Pojasno propusni filtar */
        a1 = 2*E*W_est*T - 2;
        a2 = 1 - 2*E*W_est*T + W_est*W_est*T*T;
        b1 = 2*E*W_est*T;

        yf = b1*dx0 - a1*yf_pred1 - a2*yf_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        yf_pred2 = yf_pred1;
        yf_pred1 = yf;

        x1 = fabs(yf);
        x = x1* PI/2;

        W_est2 = 0.25*W_est;

      /* Nisko propusni filtar */
        anp1 = 2*E1*W_est2*T - 2;
        anp2 = 1 - 2*E1*W_est2*T + W_est2*W_est2*T*T;
        bnp1 = W_est2*W_est2*T*T;

        ynpf = bnp1*x - anp1*ynpf_pred1 - anp2*ynpf_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ynpf_pred2 = ynpf_pred1;
        ynpf_pred1 = ynpf;

      /* Osvjezavanje brojaca */
          cnt++;

        A = ynpf;

        y[0] = A; /* Estimirana amplituda*/
      y[1] = W_est;     /* Estimirana kr. frekvencija */



}
/*********************/
/* FUNCTION mdlUpdate */
/*********************/
static void mdlUpdate(real_T *x, real_T *u, SimStruct *S, int_T tid)
{
}
/*************************/
/* FUNCTION mdlDerivatives */
/*************************/
static void mdlDerivatives(real_T *dx, real_T *x, real_T *u, SimStruct
*S, int_T tid)
{
```

```
}
/*************************/
/* FUNCTION mdlTerminate */
/*************************/
static void mdlTerminate(SimStruct *S)
{

      int_T i;
/* Oslobadja se memorija alocirana za radne vektore ... */
/* ... sto se ovdje ne koristi ...                      */
      for(i=0;i<ssGetNumPWork(S);i++) {
            if (ssGetPWorkValue(S,i) != NULL) {
                  free(ssGetPWorkValue(S,i));
            }
      }

}

#ifdef    MATLAB_MEX_FILE  /* Is file being compiled into the MEX-file ?
*/
#include "simulink.c"      /* Add functions for linking with Matlab */
#else
#include "cg_sfun.h"
#endif
```

Prilog B2. Pokretanje simulacije estimatora za sinusni signal i grafički ispis bitnih vrijednosti

```
clear;

T_p = 1;        % [s]
Omega = 2*pi/T_p;   % [rad/s]
tmax = 0.01*T_p;    % [s]
Ts = 1.0*tmax;      % [s]
t_end = 12*T_p;     % [s]
A = 10.0;       % [ ]

mex akf_novi_1.c
sim('sinus_cmex');

% frekvencijska analiza izlaznog signala
[am_sp,ph_sp,fr,Ts_est] = freq_analysis_dc_off(t,y);

figure(1),
subplot(211),plot(t,y,'r','LineWidth',2),grid on,hold on
ylabel('Izlaz procesa y')
subplot(212),plot(fr,am_sp,'r','LineWidth',2),grid on,hold on
ylabel('Ampl. spektar A(y)'), xlabel('t [s]');

figure(2),
plot(t,W_est/2/pi,'r','LineWidth',2),grid on,hold on
plot(t,Omega/2/pi*ones(size(t)),'k--','LineWidth',2),grid on,hold on
ylabel('Rez. frekvencija [Hz]'),legend('Estimirana','Stvarna'),xlabel('t
[s]')

figure(3),
plot(t,A,'r','LineWidth',2),grid on,hold on
plot(t,10,'b')
ylabel('Amplituda A(y)'), xlabel('t [s]'),legend('Estimirana','Stvarna');
```

Prilog B3. Pokretanje simulacije estimatora za PT_n model i grafički ispis bitnih vrijednosti

```
clear;

T_p = 1.0;      % [s]
tmax = 0.01*T_p;    % [s]
Ts = 1.0*tmax;      % [s]
t_end = 30*T_p;     % [s]

% Model procesa
Tp = 0.3; Kp = 2.0; N = 4;
num_c = Kp;
den_c = 1;
for(cnt = 1:N)
    den_c = conv([Tp 1],den_c);
end

yR = 1.0;
a_sw = 2.0*yR/Kp;

mex akf_novi_1.c
sim('proc_cmex');

% frekvencijska analiza izlaznog signala
[am_sp,ph_sp,fr,Ts_est] = freq_analysis_dc_off(t,y);
% Trazi se maksimum iznosa ampl. spektra
[A_max,idx_max] = max(am_sp);
% na tocki maksimuma je rez. frekvencija
W0 = 2*pi*fr(idx_max);

figure(1),
subplot(211),plot(t,y,'r','LineWidth',2),grid on,hold on
ylabel('Izlaz procesa y')
subplot(212),plot(fr,am_sp,'r','LineWidth',2),grid on,hold on
ylabel('Ampl. spektar A(y)'), xlabel('t [s]')

figure(2),
plot(t,W_est/2/pi,'r','LineWidth',2),grid on,hold on
plot(t,W0/2/pi*ones(size(t)),'k--','LineWidth',2),grid on,hold on
ylabel('Rez. frekvencija [Hz]'),legend('Estimirana','Stvarna'),xlabel('t
[s]')

figure(3),
plot(t,A,'r','LineWidth',2),grid on,hold on
plot(t,0.6,'b'),legend('Estimirana','Stvarna')
ylabel('Amplituda A(y)'), xlabel('t [s]');

figure(4),
plot(t,u,'r','LineWidth',2),grid on,hold on
ylabel('upravljacki signal'), xlabel('t [s]');
```

Prilog B4. Frekvencijka analiza zasnovana na FFT algoritmu

```
% freq_analysis.m is a m-script intended to simplify the
% spectral analysis of sampled (discrete-time) signals.
% It utilizes Discrete-time Fourier Transform
% [am_sp,ph_sp,fr,Ts] = freq_analysis_dc_off(t_res,wrr_res)
%
%   am_sp = resulting amplitude spectra data vector [rad/s]
%   ph_sp = resulting phase spectra data vector [deg]
%      fr = sampled frequency data vector [Hz]
%      Ts = sampling time
%
%   t_res = time data vector
% wrr_res = sampled wheel speed data
%

function [am_sp,ph_sp,fr,Ts] = freq_analysis_dc_off(t_res,wrr_res)

  N = 2*fix(length(t_res)/2);
  Ts = t_res(2)-t_res(1);
  fr = (1/Ts)*(0:N/2-1)/N;

  xrr = wrr_res - mean(wrr_res);

  % amplitude spectra
  yrr = abs(fft(xrr,N))/(N/2); am_sp = yrr(1:N/2);
  % phase spectra
  yrr = angle(fft(xrr,N)); ph_sp = yrr(1:N/2)*180/pi;
```

Prilog B5. Auto tuning + PID regulator

```
/*********************************/
/*                               */
/* Matlab ver. 5.3 (R11)         */
/* Matlab ver. 7.0 (R14)         */
/* C-mex S-Function              */
/*                               */
/*********************************/

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME akf_PID

#include "simstruc.h"    /* Defines SimStruct and corresponding macros */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Parametri PTn modela procesa */
/*                              */
/*        y(s)         Kp       */
/*G(s) = ------ = ------------  */
/*        u(s)     (1 + Tp)^n   */
/*                              */


/* Vrijeme uzorkovanja - ne treba biti vece od TP/N */
#define T   (double)(0.01)

#define     PI    (double)(3.1415926535898)
```

```
/* Parametri adaptivnog Kalmanovih filtara */
#define     Q1    (double)(1.0)
#define R1  (double)(1.0)
#define     MUL1  (double)(1.0e8)
#define     N1    (int)(200) // adaptacija traje 50 uzoraka
#define     THR   (double)(1.0e-6)
#define     E     (double)(0.707)
#define E1  (double)(1.0)
#define Y0  (double)(0.0)


/* Inicijalno stanje integratora */
//#define UI0     (double)(KR*Y0)

/* Limiti naponske reference procesa */
#define UMAX      (double)(5.0)     /* Gornji limit */
#define UMIN      (double)(-5.0)    /* Donji limit */

/*************************************************/

#define u(element) (*uPtrs[element])
#define NUM_OF_ARGS     0     /* Broj ulaznih argumenata - podaci
organizorani u redne vektore */
#define NUM_OF_IN 2     /* Broj ulaza:  1 - referenca, 2 - mjerenje */
#define NUM_OF_OUT      3     /* Broj izlaza: 1 - ur (izlaz regulatora) */
#define NUM_OF_REAL     0     /* Matlabova konvencija spremanja statickih
varijabli se !!ne koristi!! */
#define NUM_OF_PTR      0     /* Prije pocetka simulacije treba napraviti
novi mex-file!! */
#define NUM_OF_INT      0


/*************************************/
/* FUNCTION mdlInitializeSizes       */
/*************************************/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, NUM_OF_ARGS);  /* Ocekivani broj parametara koji
se prenosi preko dialog-boxa (0) */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);   /* Broj vremenski-kontinuiranih stanja (0)
*/
    ssSetNumDiscStates(S, 0);   /* Broj vremenski-diskretnih stanja (0) */

    /* Ne koristi se ... */
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, NUM_OF_IN);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, NUM_OF_OUT);
    ssSetNumSampleTimes(S, 1);   /* Jedinstveno vrijeme uzorkovanja 0.1
sek */

    ssSetNumRWork(S, NUM_OF_REAL);   /* Ne koriste se ... */
    ssSetNumIWork(S, NUM_OF_INT);
    ssSetNumPWork(S, NUM_OF_PTR);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
} /* end mdlInitializeSizes */

static void mdlInitializeSampleTimes(SimStruct *S)
```

```
{
      ssSetSampleTime(S, 0, T);
      ssSetOffsetTime(S, 0, 0);
}

/*************************************/
/* FUNCTION mdlInitializeConditions  */
/*************************************/
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
      if (ssGetSFcnParamsCount(S) != NUM_OF_ARGS)
      {
            #ifdef MATLAB_MEX_FILE
                  ssSetErrorStatus(S,"Wrong number of input args!");
            #else
                  printf("\nWrong number of input args!");
                  exit(0);
            #endif
      }

}
#endif

/***********************/
/* FUNCTION mdlOutputs */
/***********************/
/* Ovdje se obavlja svo racunanje */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T *y = ssGetOutputPortRealSignal(S,0);

   /* Definicije varijabli: referenca, mjerenje, naponska referenca,
proporcionalno djelovanje, derivativno djelovanje, regulacijsko odstupanje
*/
   double yR, ym, uR, uP, uD, err, y1, x, a1, a2, b1, anp1, anp2, bnp1,
err1;
   double KR, TI, TD, W, yR1, ym1, uR1, uP1, uD1, err_novi, To, anp11,
anp12, bnp11;

      static double H1 = 0.0;
      static double S1 = 1.0;
      static double x1e = 0.0;
      static double K1 = 0.0;
        static double x1e_pred = 0.0;
        static double yf = 0.0;
        static double ynpf = 0.0;
        static double yf_pred1 = 0.0;
        static double yf_pred2 = 0.0;
        static double ynpf_pred1 = 0.0;
        static double ynpf_pred2 = 0.0;
        static double A = 0.0;
      static double x1 = 0.0;
      static double x2 = 0.0;
        static double uR_prethodni = 0.0;
        static double err_prethodni = 1.0e12;
        static double ym_1 = 0.0;
        static double Tu = 0.0;
   static double uI1 = 0.0;
   static double Ku = 0.0;
        static double ym_pred1 = 0.0;
        static double ym_pred2 = 0.0;
```

```
        static double ym11 = 0.0;

      double D2y, den1;

      static double dx0 = 0.0; /* diferencija x(k) - x(k-1) */
      static double dx1 = 0.0; /* diferencija - prosla */
      static double dx2 = 0.0; /* diferencija - pretprosla */
      /* Proslo stanje izlazne velicine procesa */
      static double x_ = 0.0;
      /* Estimat frekvencije [rad/s] */
      static double W_est = 0.0;
        static double W_est2 = 0.0;

      /* Proslo stanje ulazne velicine procesa */
      static double y_ = 0.0;

   /* Proslo stanje izlazne velicine procesa */
   static double ym_ = Y0; /* Ocekivani iznos */

   static int ctrl_flag = 0;
   static int flag_init = 0;
   static int cnt_sw = 0;
      /* Brojac koraka */
      static unsigned int cnt = 0;

   yR = u(0); /* referenca */
   ym = u(1); /* mjerenje */

   /* Regulacijsko odstupanje */
   err = yR - ym;

if(cnt_sw<15){

   if(err > 0.0){
      uR = 1.0;
   }
   if(err < 0.0){
      uR = -1.0;
   }

   if(fabs(uR - uR_prethodni) > 1.0e-2){
    flag_init = 1;
      cnt_sw++;
   }
   else{
       flag_init = 0;
}
   uR_prethodni = uR;

      /* Deriviranje ulazne velicine za uklanjanje DC posmaka */
        dx0 = ym - x_;

      /* spremanje proslog stanja x za novi korak */
      x_ = ym;

      /*** Kalmanov filtar ***/
      /* Izlaz "modela procesa" */
      D2y = dx0 - 2.0*dx1 + dx2;

      /* Adaptacija kovarijance */
      if(cnt < N1){
           S1 = (1 - K1*H1)*S1 + MUL1*Q1;
        }
        else{
```

```
      S1 = (1 - K1*H1)*S1 + Q1;
}

/* Koeficijent izlazne jednadzbe */
H1 = -T * T * dx2;

/* Provjera da nazivnik nije blizak nuli */
  den1 = H1 * S1 * H1 + R1;

if(fabs(den1) < THR){
     den1 = THR;
  }

/* Adaptacija pojacanja */
K1 = S1 * H1 / den1;

/* Pogreska estimacije */
err1 = D2y - H1 * x1e_pred;

/* Osvjezavanje estimiranog stanja */
  x1e = x1e_pred + K1*err1;

/* Apsolutna vrijednost (ne moze biti negativan) */
  x1e = fabs(x1e);
x1e_pred = fabs(x1e_pred);
/* Osvjezavanje proslih stanja pojasnopropusnog filtra */
  dx2 = dx1;
  dx1 = dx0;
  x1e_pred = x1e;

  W_est = sqrt(x1e);

/* Pojasno propusni filtar */
  a1 = 2*E*W_est*T - 2;
  a2 = 1 - 2*E*W_est*T + W_est*W_est*T*T;
  b1 = 2*E*W_est*T;

  yf = b1*dx0 - a1*yf_pred1 - a2*yf_pred2;

/* Osvjezavanje proslih stanja pojasnopropusnog filtra */
  yf_pred2 = yf_pred1;
  yf_pred1 = yf;

  x1 = fabs(yf);
  x2 = x1* PI/2;

  W_est2 = 0.25*W_est;

/* Nisko propusni filtar */
  anp1 = 2*E1*W_est2*T - 2;
  anp2 = 1 - 2*E1*W_est2*T + W_est2*W_est2*T*T;
  bnp1 = W_est2*W_est2*T*T;

  ynpf = bnp1*x2 - anp1*ynpf_pred1 - anp2*ynpf_pred2;


/* Osvjezavanje proslih stanja pojasnopropusnog filtra */
  ynpf_pred2 = ynpf_pred1;
  ynpf_pred1 = ynpf;

/* Osvjezavanje brojaca */
    cnt++;

  A = ynpf;
```

```
 }

      /* Regulacija */
else{
     Ku=4/3.14/A;
     Tu=2*PI/W_est;
     To=0.1*Tu;

/* PID regulator */
      KR=0.6*Ku-0.6*Ku*To/Tu;
      TI=KR*Tu/1.2/Ku;
      TD=3*Ku*Tu/40/KR;

      yR1 = u(0); /* referenca */
      ym1 = u(1); /* mjerenje */

      /* Regulacijsko odstupanje */
      err_novi = yR1 - ym1;
      /* Proprcionalno djelovanje */
      uP1 = KR*ym1;
      /* Derivativno djelovanje */
      uD1 = KR*TD*(ym1 - ym_1)/T;
      /* Integralno djelovanje */
      uI1 += KR*T/TI*err_novi;
      /* Izlaz PI(D) regulatora */
      uR1 = uI1 - uP1 - uD1;

      /* Limitiranje izlaza i reset antiwindup integratora */
      if(uR1 > UMAX){
            uR1 = UMAX;
            uI1 = uR1 + uP1 + uD1;
      }

      if(uR1 < UMIN){
            uR1 = UMIN;
            uI1 = uR1 + uP1 + uD1;
      }
      ym_1 = ym1;
        uR=uR1;
}

   y[0] = uR;     /* Izlaz regulatora */
   y[1] = A;
   y[2] = W_est;

}
/*********************/
/* FUNCTION mdlUpdate */
/*********************/
static void mdlUpdate(real_T *x, real_T *u, SimStruct *S, int_T tid)
{
}
/*************************/
/* FUNCTION mdlDerivatives */
/*************************/
static void mdlDerivatives(real_T *dx, real_T *x, real_T *u, SimStruct *S,
int_T tid)
{
}
/************************/
/* FUNCTION mdlTerminate */
/************************/
static void mdlTerminate(SimStruct *S)
{
```

```
      int_T i;
/* Oslobadja se memorija alocirana za radne vektore ... */
/* ... sto se ovdje ne koristi ...                      */
      for(i=0;i<ssGetNumPWork(S);i++) {
            if (ssGetPWorkValue(S,i) != NULL) {
                  free(ssGetPWorkValue(S,i));
            }
      }

}

#ifdef    MATLAB_MEX_FILE  /* Is file being compiled into the MEX-file ?
*/
#include "simulink.c"      /* Add functions for linking with Matlab */
#else
#include "cg_sfun.h"
#endif
```

Prilog B6. Auto tuning + PID regulator + anti-aliasing filtar

```
/*********************************/
/*                               */
/* Matlab ver. 5.3 (R11)         */
/* Matlab ver. 7.0 (R14)         */
/* C-mex S-Function              */
/*                               */
/*********************************/

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME akf_PID_novi

#include "simstruc.h"    /* Defines SimStruct and corresponding macros */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Parametri PTn modela procesa */
/*                              */
/*        y(s)         Kp       */
/*G(s) = ------ = ------------  */
/*        u(s)     (1 + Tp)^n   */
/*                              */


/* Vrijeme uzorkovanja - ne treba biti vece od TP/N */
#define T   (double)(0.01)

#define     PI    (double)(3.1415926535898)

/* Parametri adaptivnog Kalmanovih filtara */
#define     Q1    (double)(1.0)
#define R1  (double)(1.0)
#define     MUL1  (double)(1.0e8)
#define     N1    (int)(200) // adaptacija traje 50 uzoraka
#define     THR   (double)(1.0e-6)
#define     E     (double)(0.707)
#define E1  (double)(1.0)
#define Y0  (double)(0.0)
```

```
/* Inicijalno stanje integratora */
//#define UI0     (double)(KR*Y0)

/* Limiti naponske reference procesa */
#define UMAX      (double)(5.0)     /* Gornji limit */
#define UMIN      (double)(-5.0)    /* Donji limit */

/*************************************************/

#define u(element) (*uPtrs[element])
#define NUM_OF_ARGS     0     /* Broj ulaznih argumenata - podaci
organizorani u redne vektore */
#define NUM_OF_IN 2     /* Broj ulaza:  1 - referenca, 2 - mjerenje */
#define NUM_OF_OUT      3     /* Broj izlaza: 1 - ur (izlaz regulatora) */
#define NUM_OF_REAL     0     /* Matlabova konvencija spremanja statickih
varijabli se !!ne koristi!! */
#define NUM_OF_PTR      0     /* Prije pocetka simulacije treba napraviti
novi mex-file!! */
#define NUM_OF_INT      0


/*************************************/
/* FUNCTION mdlInitializeSizes       */
/*************************************/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, NUM_OF_ARGS);  /* Ocekivani broj parametara koji
se prenosi preko dialog-boxa (0) */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);   /* Broj vremenski-kontinuiranih stanja (0)
*/
    ssSetNumDiscStates(S, 0);   /* Broj vremenski-diskretnih stanja (0) */

    /* Ne koristi se ... */
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, NUM_OF_IN);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, NUM_OF_OUT);
    ssSetNumSampleTimes(S, 1);   /* Jedinstveno vrijeme uzorkovanja 0.1
sek */

    ssSetNumRWork(S, NUM_OF_REAL);   /* Ne koriste se ... */
    ssSetNumIWork(S, NUM_OF_INT);
    ssSetNumPWork(S, NUM_OF_PTR);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
} /* end mdlInitializeSizes */

static void mdlInitializeSampleTimes(SimStruct *S)
{
      ssSetSampleTime(S, 0, T);
      ssSetOffsetTime(S, 0, 0);
}

/*************************************/
/* FUNCTION mdlInitializeConditions  */
/*************************************/
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
```

```
      if (ssGetSFcnParamsCount(S) != NUM_OF_ARGS)
      {
            #ifdef MATLAB_MEX_FILE
                  ssSetErrorStatus(S,"Wrong number of input args!");
            #else
                  printf("\nWrong number of input args!");
                  exit(0);
            #endif
      }

}
#endif

/***********************/
/* FUNCTION mdlOutputs */
/***********************/
/* Ovdje se obavlja svo racunanje */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T *y = ssGetOutputPortRealSignal(S,0);

   /* Definicije varijabli: referenca, mjerenje, naponska referenca,
proporcionalno djelovanje, derivativno djelovanje, regulacijsko odstupanje
*/
   double yR, ym, uR, uP, uD, err, y1, x, a1, a2, b1, anp1, anp2, bnp1,
err1;
   double KR, TI, TD, W, yR1, ym1, uR1, uP1, uD1, err_novi, To, anp11,
anp12, bnp11;

      static double H1 = 0.0;
      static double S1 = 1.0;
      static double x1e = 0.0;
      static double K1 = 0.0;
        static double x1e_pred = 0.0;
        static double yf = 0.0;
        static double ynpf = 0.0;
        static double yf_pred1 = 0.0;
        static double yf_pred2 = 0.0;
        static double ynpf_pred1 = 0.0;
        static double ynpf_pred2 = 0.0;
        static double A = 0.0;
      static double x1 = 0.0;
      static double x2 = 0.0;
        static double uR_prethodni = 0.0;
        static double err_prethodni = 1.0e12;
        static double ym_1 = 0.0;
        static double Tu = 0.0;
   static double uI1 = 0.0;
   static double Ku = 0.0;
        static double ym_pred1 = 0.0;
        static double ym_pred2 = 0.0;
        static double ym11 = 0.0;

      double D2y, den1;

      static double dx0 = 0.0; /* diferencija x(k) - x(k-1) */
      static double dx1 = 0.0; /* diferencija - prosla */
      static double dx2 = 0.0; /* diferencija - pretprosla */
      /* Proslo stanje izlazne velicine procesa */
      static double x_ = 0.0;
      /* Estimat frekvencije [rad/s] */
      static double W_est = 0.0;
        static double W_est2 = 0.0;
```

```
      /* Proslo stanje ulazne velicine procesa */
      static double y_ = 0.0;

   /* Proslo stanje izlazne velicine procesa */
   static double ym_ = Y0; /* Ocekivani iznos */

   static int ctrl_flag = 0;
   static int flag_init = 0;
   static int cnt_sw = 0;
      /* Brojac koraka */
      static unsigned int cnt = 0;

   yR = u(0); /* referenca */
   ym = u(1); /* mjerenje */

   /* Regulacijsko odstupanje */
   err = yR - ym;

if(cnt_sw<15){

   if(err > 0.0){
      uR = 1.0;
   }
   if(err < 0.0){
      uR = -1.0;
   }

   if(fabs(uR - uR_prethodni) > 1.0e-2){
    flag_init = 1;
      cnt_sw++;
   }
   else{
       flag_init = 0;
}
   uR_prethodni = uR;

        W=1/T/2/sqrt(2);
        anp11 = 2*E*W*T - 2;
        anp12 = 1 - 2*E*W*T + W*W*T*T;
        bnp11 = W*W*T*T;

        ym11 = bnp11*ym - anp11*ym_pred1 - anp12*ym_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ym_pred2 = ym_pred1;
        ym_pred1 = ym11;

      /* Deriviranje ulazne velicine za uklanjanje DC posmaka */
        dx0 = ym11 - x_;

      /* spremanje proslog stanja x za novi korak */
      x_ = ym11;

      /*** Kalmanov filtar ***/
      /* Izlaz "modela procesa" */
      D2y = dx0 - 2.0*dx1 + dx2;

      /* Adaptacija kovarijance */
      if(cnt < N1){
           S1 = (1 - K1*H1)*S1 + MUL1*Q1;
        }
        else{
```

```
            S1 = (1 - K1*H1)*S1 + Q1;
      }

      /* Koeficijent izlazne jednadzbe */
      H1 = -T * T * dx2;

      /* Provjera da nazivnik nije blizak nuli */
        den1 = H1 * S1 * H1 + R1;

      if(fabs(den1) < THR){
           den1 = THR;
        }

      /* Adaptacija pojacanja */
      K1 = S1 * H1 / den1;

      /* Pogreska estimacije */
      err1 = D2y - H1 * x1e_pred;

      /* Osvjezavanje estimiranog stanja */
        x1e = x1e_pred + K1*err1;

      /* Apsolutna vrijednost (ne moze biti negativan) */
        x1e = fabs(x1e);
      x1e_pred = fabs(x1e_pred);
      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        dx2 = dx1;
        dx1 = dx0;
        x1e_pred = x1e;

        W_est = sqrt(x1e);

        a1 = 2*E*W_est*T - 2;
        a2 = 1 - 2*E*W_est*T + W_est*W_est*T*T;
        b1 = 2*E*W_est*T;

        yf = b1*dx0 - a1*yf_pred1 - a2*yf_pred2;

      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        yf_pred2 = yf_pred1;
        yf_pred1 = yf;

        x1 = fabs(yf);
        x2 = x1* PI/2;

        W_est2 = 0.25*W_est;

        anp1 = 2*E1*W_est2*T - 2;
        anp2 = 1 - 2*E1*W_est2*T + W_est2*W_est2*T*T;
        bnp1 = W_est2*W_est2*T*T;

        ynpf = bnp1*x2 - anp1*ynpf_pred1 - anp2*ynpf_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ynpf_pred2 = ynpf_pred1;
        ynpf_pred1 = ynpf;

      /* Osvjezavanje brojaca */
          cnt++;

        A = ynpf;
 }
else{
```

```
     Ku=4/3.14/A;
     Tu=2*PI/W_est;
     To=0.1*Tu;

/* PID regulator */
      KR=0.6*Ku-0.6*Ku*To/Tu;
      TI=KR*Tu/1.2/Ku;
      TD=3*Ku*Tu/40/KR;

      yR1 = u(0); /* referenca */
      ym1 = u(1); /* mjerenje */

      /* Regulacijsko odstupanje */
      err_novi = yR1 - ym1;
      /* Proprcionalno djelovanje */
      uP1 = KR*ym1;
      /* Derivativno djelovanje */
      uD1 = KR*TD*(ym1 - ym_1)/T;
      /* Integralno djelovanje */
      uI1 += KR*T/TI*err_novi;
      /* Izlaz PI(D) regulatora */
      uR1 = uI1 - uP1 - uD1;

      /* Limitiranje izlaza i reset antiwindup integratora */
      if(uR1 > UMAX){
            uR1 = UMAX;
            uI1 = uR1 + uP1 + uD1;
      }

      if(uR1 < UMIN){
            uR1 = UMIN;
            uI1 = uR1 + uP1 + uD1;
      }
      ym_1 = ym1;
        uR=uR1;
}

   y[0] = uR;     /* Izlaz regulatora */
   y[1] = A;
   y[2] = W_est;

}
/*********************/
/* FUNCTION mdlUpdate */
/*********************/
static void mdlUpdate(real_T *x, real_T *u, SimStruct *S, int_T tid)
{
}
/*************************/
/* FUNCTION mdlDerivatives */
/*************************/
static void mdlDerivatives(real_T *dx, real_T *x, real_T *u, SimStruct *S,
int_T tid)
{
}
/************************/
/* FUNCTION mdlTerminate */
/************************/
static void mdlTerminate(SimStruct *S)
{

      int_T i;
/* Oslobadja se memorija alocirana za radne vektore ... */
/* ... sto se ovdje ne koristi ...                      */
```

```
      for(i=0;i<ssGetNumPWork(S);i++) {
            if (ssGetPWorkValue(S,i) != NULL) {
                  free(ssGetPWorkValue(S,i));
            }
      }

}

#ifdef    MATLAB_MEX_FILE  /* Is file being compiled into the MEX-file ?
*/
#include "simulink.c"      /* Add functions for linking with Matlab */
#else
#include "cg_sfun.h"
#endif
```

Prilog B7. Auto tuning + PI regulator + anti-aliasing filtar

```
/*********************************/
/*                               */
/* Matlab ver. 5.3 (R11)         */
/* Matlab ver. 7.0 (R14)         */
/* C-mex S-Function              */
/*                               */
/*********************************/

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME akf_PI_novi

#include "simstruc.h"    /* Defines SimStruct and corresponding macros */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Parametri PTn modela procesa */
/*                              */
/*        y(s)         Kp       */
/*G(s) = ------ = ------------  */
/*        u(s)     (1 + Tp)^n   */
/*                              */


/* Vrijeme uzorkovanja - ne treba biti vece od TP/N */
#define T   (double)(0.01)

#define     PI    (double)(3.1415926535898)

/* Parametri adaptivnog Kalmanovih filtara */
#define     Q1    (double)(1.0)
#define R1  (double)(1.0)
#define     MUL1  (double)(1.0e8)
#define     N1    (int)(200) // adaptacija traje 50 uzoraka
#define     THR   (double)(1.0e-6)
#define     E     (double)(0.707)
#define E1  (double)(1.0)
#define Y0  (double)(0.0)


/* Inicijalno stanje integratora */
//#define UI0     (double)(KR*Y0)

/* Limiti naponske reference procesa */
#define UMAX      (double)(5.0)     /* Gornji limit */
```

```
#define UMIN      (double)(-5.0)    /* Donji limit */

/*********************************************/

#define u(element) (*uPtrs[element])
#define NUM_OF_ARGS     0     /* Broj ulaznih argumenata - podaci
organizorani u redne vektore */
#define NUM_OF_IN 2     /* Broj ulaza:  1 - referenca, 2 - mjerenje */
#define NUM_OF_OUT      3     /* Broj izlaza: 1 - ur (izlaz regulatora) */
#define NUM_OF_REAL     0     /* Matlabova konvencija spremanja statickih
varijabli se !!ne koristi!! */
#define NUM_OF_PTR      0     /* Prije pocetka simulacije treba napraviti
novi mex-file!! */
#define NUM_OF_INT      0


/*************************************/
/* FUNCTION mdlInitializeSizes       */
/*************************************/
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, NUM_OF_ARGS);  /* Ocekivani broj parametara koji
se prenosi preko dialog-boxa (0) */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);   /* Broj vremenski-kontinuiranih stanja (0)
*/
    ssSetNumDiscStates(S, 0);   /* Broj vremenski-diskretnih stanja (0) */

    /* Ne koristi se ... */
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, NUM_OF_IN);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, NUM_OF_OUT);
    ssSetNumSampleTimes(S, 1);   /* Jedinstveno vrijeme uzorkovanja 0.1
sek */

    ssSetNumRWork(S, NUM_OF_REAL);   /* Ne koriste se ... */
    ssSetNumIWork(S, NUM_OF_INT);
    ssSetNumPWork(S, NUM_OF_PTR);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
} /* end mdlInitializeSizes */

static void mdlInitializeSampleTimes(SimStruct *S)
{
      ssSetSampleTime(S, 0, T);
      ssSetOffsetTime(S, 0, 0);
}

/*************************************/
/* FUNCTION mdlInitializeConditions  */
/*************************************/
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
      if (ssGetSFcnParamsCount(S) != NUM_OF_ARGS)
      {
            #ifdef MATLAB_MEX_FILE
                  ssSetErrorStatus(S,"Wrong number of input args!");
            #else
```

```
                printf("\nWrong number of input args!");
                exit(0);
            #endif
      }

}
#endif

/*********************/
/* FUNCTION mdlOutputs */
/*********************/
/* Ovdje se obavlja svo racunanje */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T *y = ssGetOutputPortRealSignal(S,0);

   /* Definicije varijabli: referenca, mjerenje, naponska referenca,
proporcionalno djelovanje, derivativno djelovanje, regulacijsko odstupanje
*/
   double yR, ym, uR, uP, uD, err, y1, x, a1, a2, b1, anp1, anp2, bnp1,
err1;
   double KR, TI, TD, W, yR1, ym1, uR1, uP1, uD1, err_novi, To, anp11,
anp12, bnp11;

      static double H1 = 0.0;
      static double S1 = 1.0;
      static double x1e = 0.0;
      static double K1 = 0.0;
        static double x1e_pred = 0.0;
        static double yf = 0.0;
        static double ynpf = 0.0;
        static double yf_pred1 = 0.0;
        static double yf_pred2 = 0.0;
        static double ynpf_pred1 = 0.0;
        static double ynpf_pred2 = 0.0;
        static double A = 0.0;
      static double x1 = 0.0;
      static double x2 = 0.0;
        static double uR_prethodni = 0.0;
        static double err_prethodni = 1.0e12;
        static double ym_1 = 0.0;
        static double Tu = 0.0;
   static double uI1 = 0.0;
   static double Ku = 0.0;
        static double ym_pred1 = 0.0;
        static double ym_pred2 = 0.0;
        static double ym11 = 0.0;

      double D2y, den1;

      static double dx0 = 0.0; /* diferencija x(k) - x(k-1) */
      static double dx1 = 0.0; /* diferencija - prosla */
      static double dx2 = 0.0; /* diferencija - pretprosla */
      /* Proslo stanje izlazne velicine procesa */
      static double x_ = 0.0;
      /* Estimat frekvencije [rad/s] */
      static double W_est = 0.0;
        static double W_est2 = 0.0;

      /* Proslo stanje ulazne velicine procesa */
      static double y_ = 0.0;

   /* Proslo stanje izlazne velicine procesa */
```

```
   static double ym_ = Y0; /* Ocekivani iznos */

   static int ctrl_flag = 0;
   static int flag_init = 0;
   static int cnt_sw = 0;
      /* Brojac koraka */
      static unsigned int cnt = 0;

   yR = u(0); /* referenca */
   ym = u(1); /* mjerenje */

   /* Regulacijsko odstupanje */
   err = yR - ym;

if(cnt_sw<15){

   if(err > 0.0){
      uR = 1.0;
   }
   if(err < 0.0){
      uR = -1.0;
   }

   if(fabs(uR - uR_prethodni) > 1.0e-2){
    flag_init = 1;
      cnt_sw++;
   }
   else{
       flag_init = 0;
}
   uR_prethodni = uR;

        W=1/T/2/sqrt(2);
        anp11 = 2*E*W*T - 2;
        anp12 = 1 - 2*E*W*T + W*W*T*T;
        bnp11 = W*W*T*T;

        ym11 = bnp11*ym - anp11*ym_pred1 - anp12*ym_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ym_pred2 = ym_pred1;
        ym_pred1 = ym11;

      /* Deriviranje ulazne velicine za uklanjanje DC posmaka */
        dx0 = ym11 - x_;

      /* spremanje proslog stanja x za novi korak */
      x_ = ym11;

      /*** Kalmanov filtar ***/
      /* Izlaz "modela procesa" */
      D2y = dx0 - 2.0*dx1 + dx2;

      /* Adaptacija kovarijance */
      if(cnt < N1){
           S1 = (1 - K1*H1)*S1 + MUL1*Q1;
        }
        else{
           S1 = (1 - K1*H1)*S1 + Q1;
      }

      /* Koeficijent izlazne jednadzbe */
      H1 = -T * T * dx2;
```

```
      /* Provjera da nazivnik nije blizak nuli */
        den1 = H1 * S1 * H1 + R1;

      if(fabs(den1) < THR){
           den1 = THR;
        }

      /* Adaptacija pojacanja */
      K1 = S1 * H1 / den1;

      /* Pogreska estimacije */
      err1 = D2y - H1 * x1e_pred;

      /* Osvjezavanje estimiranog stanja */
        x1e = x1e_pred + K1*err1;

      /* Apsolutna vrijednost (ne moze biti negativan) */
        x1e = fabs(x1e);
      x1e_pred = fabs(x1e_pred);
      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        dx2 = dx1;
        dx1 = dx0;
        x1e_pred = x1e;

        W_est = sqrt(x1e);

        a1 = 2*E*W_est*T - 2;
        a2 = 1 - 2*E*W_est*T + W_est*W_est*T*T;
        b1 = 2*E*W_est*T;

        yf = b1*dx0 - a1*yf_pred1 - a2*yf_pred2;

      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        yf_pred2 = yf_pred1;
        yf_pred1 = yf;

        x1 = fabs(yf);
        x2 = x1* PI/2;

        W_est2 = 0.25*W_est;

        anp1 = 2*E1*W_est2*T - 2;
        anp2 = 1 - 2*E1*W_est2*T + W_est2*W_est2*T*T;
        bnp1 = W_est2*W_est2*T*T;

        ynpf = bnp1*x2 - anp1*ynpf_pred1 - anp2*ynpf_pred2;


      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ynpf_pred2 = ynpf_pred1;
        ynpf_pred1 = ynpf;

      /* Osvjezavanje brojaca */
          cnt++;

        A = ynpf;
 }
else{
     Ku=4/3.14/A;
     Tu=2*PI/W_est;
     To=0.1*Tu;

/* PI regulator */
```

```
      KR=0.45*Ku-0.27*Ku*To/Tu;
      TI=KR*Tu/0.54/Ku;
        TD=0;

      yR1 = u(0); /* referenca */
      ym1 = u(1); /* mjerenje */

      /* Regulacijsko odstupanje */
      err_novi = yR1 - ym1;
      /* Proprcionalno djelovanje */
      uP1 = KR*ym1;
      /* Derivativno djelovanje */
      uD1 = KR*TD*(ym1 - ym_1)/T;
      /* Integralno djelovanje */
      uI1 += KR*T/TI*err_novi;
      /* Izlaz PI(D) regulatora */
      uR1 = uI1 - uP1 - uD1;

      /* Limitiranje izlaza i reset antiwindup integratora */
      if(uR1 > UMAX){
            uR1 = UMAX;
            uI1 = uR1 + uP1 + uD1;
      }

      if(uR1 < UMIN){
            uR1 = UMIN;
            uI1 = uR1 + uP1 + uD1;
      }
      ym_1 = ym1;
        uR=uR1;
}

   y[0] = uR;     /* Izlaz regulatora */
   y[1] = A;
   y[2] = W_est;

}
/*********************/
/* FUNCTION mdlUpdate */
/*********************/
static void mdlUpdate(real_T *x, real_T *u, SimStruct *S, int_T tid)
{
}
/***************************/
/* FUNCTION mdlDerivatives */
/***************************/
static void mdlDerivatives(real_T *dx, real_T *x, real_T *u, SimStruct *S,
int_T tid)
{
}
/************************/
/* FUNCTION mdlTerminate */
/************************/
static void mdlTerminate(SimStruct *S)
{

      int_T i;
/* Oslobadja se memorija alocirana za radne vektore ... */
/* ... sto se ovdje ne koristi ...                      */
      for(i=0;i<ssGetNumPWork(S);i++) {
            if (ssGetPWorkValue(S,i) != NULL) {
                  free(ssGetPWorkValue(S,i));
            }
      }
```

```
}

#ifdef    MATLAB_MEX_FILE  /* Is file being compiled into the MEX-file ?
*/
#include "simulink.c"      /* Add functions for linking with Matlab */
#else
#include "cg_sfun.h"
#endif
```

Prilog B8. Source kod za eksperimentalnu provjeru

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define OSNOVNA_ADR 0x220

#define UI_ADR(x) (OSNOVNA_ADR + x)
#define AD_NIZI_BAJT          4
#define AD_VISI_BAJT          5
#define DA1_NIZI_BAJT         4
#define DA1_VISI_BAJT         5
#define DA2_NIZI_BAJT         6 /* 6 */
#define DA2_VISI_BAJT         7 /* 7 */
#define POJACANJE             9
#define AD_KANAL        10
#define TRIGER_MOD            11
#define AD_PRETVORBA          12

#define T_ms            1000
#define T         (float)(T_ms/1000)

#define     PI    (float)(3.1415926535898)
#define CTRL      (int)(0) /* 1 -> PID, 0 -> PI */

/* Parametri adaptivnog Kalmanovih filtara */
#define     Q1    (float)(1.0)
#define R1  (float)(1.0)
#define     MUL1  (float)(1.0e8)
#define     N1    (int)(200) // adaptacija traje 50 uzoraka
#define     THR   (float)(1.0e-6)
#define     E     (float)(0.707)
#define E1  (float)(1.0)
#define Y0  (float)(0.0)

/* Limiti naponske reference procesa */
#define UMAX      (float)(10.0)     /* Gornji limit */
#define UMIN      (float)(0.0)      /* Donji limit */

main()
{
      float AD(void);
      void DA(float);
      // float PIalg(float,float,float, int, float,float,float,float);
      void Ispis(int, float, float, float, float, float);
      void FajlIspis(int);
      float AD_pod, DA_pod;
      float dT, di;
      static float dv = 3.5;
```

```
      float temp_ref;
      int korak = 0,flag = 0;

      float yR, ym, uR, uP, uD, err, y1, x, a1, a2, b1, anp1, anp2, bnp1,
err1;
      float KR, TI, TD, W, yR1, ym1, uR1, uP1, uD1, err_novi, To, anp11,
anp12, bnp11;

      static float H1 = 0.0;
      static float S1 = 1.0;
      static float x1e = 0.0;
      static float K1 = 0.0;
        static float x1e_pred = 0.0;
        static float yf = 0.0;
        static float ynpf = 0.0;
        static float yf_pred1 = 0.0;
        static float yf_pred2 = 0.0;
        static float ynpf_pred1 = 0.0;
        static float ynpf_pred2 = 0.0;
        static float A = 0.0;
      static float x1 = 0.0;
      static float x2 = 0.0;
        static float uR_prethodni = 0.0;
        static float err_prethodni = 1.0e12;
        static float ym_1 = 0.0;
        static float Tu = 0.0;
      static float uI1 = 0.0;
      static float Ku = 0.0;
        static float ym_pred1 = 0.0;
        static float ym_pred2 = 0.0;
        static float ym11 = 0.0;

      float D2y, den1;

      static float dx0 = 0.0; /* diferencija x(k) - x(k-1) */
      static float dx1 = 0.0; /* diferencija - prosla */
      static float dx2 = 0.0; /* diferencija - pretprosla */
      /* Proslo stanje izlazne velicine procesa */
      static float x_ = 0.0;
      /* Estimat frekvencije [rad/s] */
      static float W_est = 0.0;
        static float W_est2 = 0.0;

      /* Proslo stanje ulazne velicine procesa */
      static float y_ = 0.0;

      /* Proslo stanje izlazne velicine procesa */
      static float ym_ = Y0; /* Ocekivani iznos */

      static int ctrl_flag = 0;
      static int flag_init = 0;
      static int cnt_sw = 0;
      /* Brojac koraka */
      static unsigned int cnt = 0;

      clrscr();
      outportb(UI_ADR(AD_KANAL), 15);
      outportb(UI_ADR(POJACANJE), 0);
      outportb(UI_ADR(TRIGER_MOD), 1);
      // dv = 0.0;
      for (;;) {
            if (kbhit()) {
                  clrscr();
                  printf ("Nova referenca temperature:  ");
```

```
                  scanf ("%f", &temp_ref);
                  if (temp_ref < 0)
                        break;
                   //   korak = 0;
                  dv = temp_ref/10.0;
                  /* PI/PID regulator prima ovu referencu [V] */
                  if(dv > 6.0){
                     dv = 6.0;
                  }

                  printf("\n");
            }
      dT = AD();          /* mjerni signal temperature u [V] */
      yR = dv; /* referenca */
      ym = dT; /* mjerenje */
            /* Ovdje pocinje auto-tuning PI(D) program */
            /*****************************************/

      /* Regulacijsko odstupanje */
      err = yR - ym;

if(cnt<N1){

      if(err > 0.0){
            uR = 6.0;
      }
      if(err < 0.0){
            uR = 4.0;
      }

      /**/
      if(fabs(uR - uR_prethodni) > 1.0e-2){
         flag_init = 1;
            cnt_sw++;
      }
         else{
               flag_init = 0;
      }
      uR_prethodni = uR;
      /**/

        W=1/T/2/sqrt(2);
        anp11 = 2*E*W*T - 2;
        anp12 = 1 - 2*E*W*T + W*W*T*T;
        bnp11 = W*W*T*T;

        ym11 = bnp11*ym - anp11*ym_pred1 - anp12*ym_pred2;

      /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
        ym_pred2 = ym_pred1;
        ym_pred1 = ym11;

      /* Deriviranje ulazne velicine za uklanjanje DC posmaka */
        dx0 = ym11 - x_;

      /* spremanje proslog stanja x za novi korak */
      x_ = ym11;

      /*** Kalmanov filtar ***/
      /* Izlaz "modela procesa" */
      D2y = dx0 - 2.0*dx1 + dx2;

      /* Adaptacija kovarijance */
      if(cnt < N1/5){
```

```
        S1 = (1 - K1*H1)*S1 + MUL1*Q1;
      }
      else{
        S1 = (1 - K1*H1)*S1 + Q1;
    }

    /* Koeficijent izlazne jednadzbe */
    H1 = -T * T * dx2;

    /* Provjera da nazivnik nije blizak nuli */
      den1 = H1 * S1 * H1 + R1;

    if(fabs(den1) < THR){
        den1 = THR;
      }

    /* Adaptacija pojacanja */
    K1 = S1 * H1 / den1;

    /* Pogreska estimacije */
    err1 = D2y - H1 * x1e_pred;

    /* Osvjezavanje estimiranog stanja */
      x1e = x1e_pred + K1*err1;

    /* Apsolutna vrijednost (ne moze biti negativan) */
      x1e = fabs(x1e);
    x1e_pred = fabs(x1e_pred);
    /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
      dx2 = dx1;
      dx1 = dx0;
      x1e_pred = x1e;

      W_est = sqrt(x1e);

      a1 = 2*E*W_est*T - 2;
      a2 = 1 - 2*E*W_est*T + W_est*W_est*T*T;
      b1 = 2*E*W_est*T;

      yf = b1*dx0 - a1*yf_pred1 - a2*yf_pred2;

    /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
      yf_pred2 = yf_pred1;
      yf_pred1 = yf;

      x1 = fabs(yf);
      x2 = x1* PI/2;

      W_est2 = 0.25*W_est;

      anp1 = 2*E1*W_est2*T - 2;
      anp2 = 1 - 2*E1*W_est2*T + W_est2*W_est2*T*T;
      bnp1 = W_est2*W_est2*T*T;

      ynpf = bnp1*x2 - anp1*ynpf_pred1 - anp2*ynpf_pred2;


    /* Osvjezavanje proslih stanja pojasnopropusnog filtra */
      ynpf_pred2 = ynpf_pred1;
      ynpf_pred1 = ynpf;

    /* Osvjezavanje brojaca */
        cnt++;
```

```
        A = ynpf;
}
else{
     Ku=4/3.14/A;
     Tu=2*PI/W_est;
       To=T;

/* PID regulator */
      if(CTRL){
            KR=0.6*Ku-0.6*Ku*To/Tu;
            TI=KR*Tu/1.2/Ku;
            TD=3*Ku*Tu/40/KR;
      }
      else{
            KR=0.45*Ku-0.27*Ku*To/Tu;
            TI=KR*Tu/0.54/Ku;
            TD=0.0;
      }

      yR1 = yR; /* referenca */
      ym1 = ym; /* mjerenje */

      /* Regulacijsko odstupanje */
      err_novi = yR1 - ym1;
      if(CTRL){
            /* Proprcionalno djelovanje */
            uP1 = KR*ym1;
            /* Derivativno djelovanje */
            uD1 = KR*TD*(ym1 - ym_1)/T;
            /* Integralno djelovanje */
            uI1 += KR*T/TI*err_novi;
            /* Izlaz PI(D) regulatora */
            uR1 = uI1 - uP1 - uD1;
      }
      else{
            /* Proprcionalno djelovanje */
            uP1 = KR*ym1;
            /* Derivativno djelovanje */
            uD1 = 0.0;
            /* Integralno djelovanje */
            uI1 += KR*T/TI*err_novi;
            /* Izlaz PI(D) regulatora */
            uR1 = uI1 - uP1 - uD1;
      }

      /* Limitiranje izlaza i reset antiwindup integratora */
      if(uR1 > UMAX){
            uR1 = UMAX;
            uI1 = uR1 + uP1 + uD1;
      }

      if(uR1 < UMIN){
            uR1 = UMIN;
            uI1 = uR1 + uP1 + uD1;
      }
      ym_1 = ym1;
        uR=uR1;
}
            /*********************************************/
            /* Ovdje zavrsava auto-tuning PI(D) regulatora */
            DA(uR); /* slanje na DA konverter komande napona */

            Ispis(korak, yR, uR, ym, A, W_est);
            delay(T_ms);
```

```
            ++korak;

      }
      DA(0.0);
      FajlIspis(korak);
}

float AD(void)
{
      int nizi_bajt, visi_bajt, pod;
      float pod_float;

      outportb(UI_ADR(AD_PRETVORBA), 0xff);
      do
            visi_bajt = inportb(UI_ADR(AD_VISI_BAJT));
      while (visi_bajt & 0x10);
      nizi_bajt = inportb(UI_ADR(AD_NIZI_BAJT));
      pod = nizi_bajt | (visi_bajt << 8);
      pod_float = ((float)(pod - 2048))/204.8;

      return (pod_float);
}

void DA(float pod)
{
      int nizi_bajt, visi_bajt, pod_int;

      pod_int = (int)((10.0 - pod)*409.5);

      if(pod_int > 4095){
            pod_int = 4095;
      }

      if(pod_int < 0){
            pod_int = 0;
      }

      nizi_bajt = pod_int & 0xff;
      visi_bajt = pod_int >> 8;
      outportb(UI_ADR(DA2_NIZI_BAJT), nizi_bajt);
      outportb(UI_ADR(DA2_VISI_BAJT), visi_bajt);

      /* nizi_bajt = (2048 - 256) & 0xff;
      visi_bajt = (2048 - 256) >> 8;*/
      /* konstantna vrijednost protoka */
      outportb(UI_ADR(DA1_NIZI_BAJT),0xff);
      outportb(UI_ADR(DA1_VISI_BAJT),0x06);

}


//#define K (float)(6.75)
//#define TI (float)(19.6)

/*float PIalg(float dv, float dT,float dT_, int korak, float Tend,float
Ti,float Td,float Kr)
{
      static float yI_ = 0.0;
      float yp,yd;
      float e, dr, di;
      int first_in =1;
if (korak >=(Tend+1)){
      if (korak ==(Tend+1) && first_in==1){yI_=Kr*dT + 5;first_in=0; }
      e = dv - dT;
```

```
      yp = Kr*dT;
      yI_ = yI_ + Kr*T/Ti*e;
      yd = Kr*Td*(dT-dT_)/T;
      dr = yI_ - yp - yd;

      if (dr > 10.0) {
            dr = 10.0;
            yI_ = dr + yp + yd;
      }
      else if (dr < 0.0) {
            dr = 0.0;
            yI_ = dr + yp + yd;
      }

}
      if (korak < 80){dr=4.5;}
      if (korak >=80  && korak< (Tend+1) ){dr=5;}
//    if (korak < (Tend+50) && korak >= (Tend+10)){dr=5;}

      di = dr;
      return (di);
}*/

#define MAX_POD 1000

float pod[MAX_POD][6];

void Ispis(int korak, float dv, float di, float dT, float Tu, float Tg)
{
      static float uv0, ur0, uT0, uTu, uTg;
      static float ur=0., uv=0., uT=0., Tus=0., Tgs=0.;

      if (korak == 0) {
            uv0 = uv;
            ur0 = ur;
            uT0 = uT;
            uTu = Tus;
            uTg = Tgs;
      }
      uv = dv;
      ur = di;
      uT = dT;
      Tus = Tu;
      Tgs = Tg;

      if (korak < MAX_POD) {
            pod[korak][0] = korak * T;
            pod[korak][1] = uv /*- uv0 */;
            pod[korak][2] = ur /*- ur0 */;
            pod[korak][3] = uT /*- uT0*/;
            pod[korak][4] = Tus /*- */;
            pod[korak][5] = Tgs /*- */;
      }
      printf("k = %3d    uv = %6.2f   ur = %6.2f   uT = %6.2f A = %6.2f W
= %6.2f\n",
            korak, uv, ur, uT,Tus,Tgs);
}

void FajlIspis(korak)
int korak;
{
      FILE *fp;
      int i;
```

```
      fp = fopen("dipl.dat", "w+t");
      for (i = 0; i < korak; ++i)
            fprintf(fp, "%6.3f  %6.3f  %6.3f  %6.3f  %6.3f  %6.3f\n",
                  pod[i][0], pod[i][1], pod[i][2], pod[i][3], pod[i][4],
pod[i][5]);
      fclose(fp);

/*    if (!InitGraph())
            printf("Greska pri inicijalizaciji grafike!\n");
      else if (!PlotCurves(pod, i * 4, 4))
            printf("\n**Pogreska: X-varijabla (1.kolona) nije dana u
rastucem redoslijedu!!\n\n");
      DeInitGraph();*/
}
```

Prilog B9. Matlab kod za grafički ispis eksperimentalnih rezultata

```
load exp_data
 t = dipl(:,1);
yR = dipl(:,2);
 u = dipl(:,3);
 y = dipl(:,4);
 A = dipl(:,5);
 W = dipl(:,6);

Ts = t(2) - t(1);
figure(1),
subplot(211),plot(t,10*y,'r','LineWidth',2),hold on,
subplot(211),plot(t,10*yR,'b','LineWidth',2),grid on,
xlabel('t [s]'), ylabel('ym, y, yR [°C]'),
legend('Temp. senzor: y','Referenca yR',0),
subplot(212),plot(t,u,'b','LineWidth',2),grid on,
xlabel('t [s]'), ylabel('u [V]')

figure(2),
subplot(211),plot(t,W,'r','LineWidth',2),hold on,grid on,
xlabel('t [s]'), ylabel('W [rad/s]'),
subplot(212),plot(t,A,'r','LineWidth',2),grid on,
xlabel('t [s]'), ylabel('A [V]')
```

9. Literatura

[1] The Mathworks. Matlab-high level language and interactive environment that enables you to perform computationally intensive tasks. http://www.mathworks.com

[2] Z. Vukić, Lj. Kuljača, *Automatsko upravljanje.* Kigen, Zagreb, 2005.

[3] Tugomir Šurina, *Automatska regulacija.* Školska knjiga, Zagreb, 1987.

[4] Cheng-Ching Yu, *Autotuning of PID controllers*, Springer, London, 2006.

[5] K. J. Åström, B. Wittenmark: *Computer-Controlles Systems – Theory and Design*, 3rd edition, Prentice-Hall, 1997.

[6] D. Pavković: "*Procjena varijabli stanja automobilskog pogona s primjenama u regulaciji*", Doktorat, FSB, Zagreb, 2007.

[7] D. K. Lindner: *Introduction to signals and systems*, McGraw-Hill, 1999.

[8] T. Kučiš: "*Identifikacija vremenski kontinuiranih modela grijalice s puhalom i sinteza PID regulatora temperature*", Završni rad, FSB, Zagreb, 2009.

[9] J. Deur, *Predavanja iz kolegija "Elektromotorni servo pogoni"*.

[10] J. Deur, *Predavanja iz kolegija* "*Neizrazito i digitalno upravljanje*".