Elektrotehnički fakultet u Sarajevu
Odsjek za Automatiku i elektroniku
Predmet: Projektovanje sistema automatskog upravljanja
Nastavnik: Prof. dr Adnan Tahirović

SEMINARSKI RAD

Student: Mirza Balta

Naslov rada**: Rekurzivne metode identifikacije sistema i primjeri primjene u adaptivnim sistemima upravljanja**

Sarajevo, maj 2021. godine

Sadržaj

Uvod

U ovom radu su izloženi i implementirani neki od rekurzivnih algoritama parametarske identifikacije sistema, koji su primjenjivi i u sistemima upravljanja koji u sebe uključuju identifikaciju objekta upravljanja. Algoritmi su implementirani u C++ programskom jeziku kao blok S-funkcije u Matlab Simulink-u, i ovaj blok je korišten u simulacijama adaptivnog upravljanja nelinearnim procesom.

Kod rekurzivnih metoda identifikacije, koje se još zovu i on-line metode identifikacije, estimati parametara se računaju rekurzivno u vremenu. Ovo znači da ako postoji estimat $\hat{\boldsymbol{\theta}}(k-1)$ koji je izračunat na osnovu izmjerenih podataka o sistemu do trenutka $k-1$, tada je $\hat{\boldsymbol{\theta}}(k)$ izračunat određenom modifikacijom $\hat{\boldsymbol{\theta}}(k-1)$. Rekurzivne metode identifikacije imaju sljedeće opće osobine:

- One su dio mnogih adaptivnih sistema upravljanja gdje je upravljačko djelovanje zasnovano na najrecentnijem modelu objekta upravljanja.
- Zahtjevi za računarskom memorijom su mali.
- Lako se mogu modifikovati u real-time algoritme, ciljane na praćenje vremenski promjenjivih parametara.

Nelinearni procesi mogu biti linearizovani u okolini radne tačke. Pri promjeni radne tačke mijenja se i linerizovana dinamika, što se odražava tako da su parametri linearnog modela vremenski promjenjivi. Za sporu promjenu radne tačke, mogu se postići dobri rezultati ako za nelinearni proces pretpostavimo linearni model sa vremenski promjenjivim parametrima.

Mnogi adaptivni sistemi upravljanja su (implicitno ili eksplicitno) zasnovani na rekurzivnoj identifikaciji, gdje parametri modela procesa se osvežavaju u svakom trenutku odabiranja. Ovi vremesko-promjenjivi modeli su upotrijebljeni za određivanje (vremensko-promjenjivih) parametara regulatora, tako da regulator zavisi od prethodnog ponašanja objekta upravljanja (kroz tok informacija: objekat $\rightarrow$ model $\rightarrow$ regulator). Ako je upotrijebljen adekvatan princip podešavanja regulatora, tada će se regulator adaptirati dinamici objekta.

Postoje različite varijante rekurzivnih algoritama identifikacije, pogodnih za sisteme sa vremenski promjenjivim parametrima, i ovi algoritmi su konstruisani tako da odgovaraju na promjene u procesu i da u određenom smislu zanemaruju stare podatke o procesu.

U praktičnom dijelu rada su navedeni primjeri sistema automatskog upravljanja, odnosno njihove simulacije, u kojima se koristi rekurzivna estimacija parametara modela objekta upravljanja (u zatvorenoj upravljačkoj konturi). Objekat upravljanja u primjerima je izrazito nelinearan hemijski reaktor. Prvi primjer je PI adaptivno upravljanje u kojem se estimacija parametara linearnog modela objekta vrši samo pri promjeni radne tačke objekta. Da bi se obezbjedila dosljedna pobuda procesa u intervalu vremena u kojem se vrši estimacija, na referentni ulaz se injektira odgovarajući dodatni testni signal. Zbog nelinearnosti objekta, za različite radne tačke imamo različite linearne modele, odnosno prenosne funkcije koje karakterišu estimirani parametri. Ovi parametri su upotrijebljeni za podešavanje pojačanja PI regulatora.

Drugi primjer je adaptivno MPC upravljanje, koje koristi linearni model procesa u prostoru stanja. Kako je objekat nelinearan, to je linearni model objekta vremenski promjenjiv. Prema tome ovaj model se mora ažurirati u svakom trenutku odabiranja, za šta nam je služi implemetirani blok rekurzivnog estimatora parametara.

1. Identifikacija sistema metodom najmanjih kvadrata

To je metoda kojom se pod određenim uslovima i na osnovu mjerenja N parova ulaza i izlaza sistema kojeg identificiramo, mogu identificirati parametri tog sistema sa određenom tačnošću. Ova metoda spada u grupu off-line ili batch metoda, koje za estimaciju parametara istovremeno koriste sve izmjerene podatke o sistemu. Pretpostavimo da je proces kojeg identificiramo određen prenosnom funkcijom:

$$G\left(z^{-1}\right)=\frac{y(z)}{u(z)}=\frac{\sum_{i=0}^{n_b} b_i z^{-i}}{\sum_{i=0}^{n_a} a_i z^{-i}} z^{-d} = \frac{B(z^{-1})}{A(z^{-1})} z^{-d} \tag{1.1}$$

gdje su a_i, b_i parametri koje identificiramo. Bez umanjenja opštosti može se uzeti da je $n_a = n_b = m$. Uzimamo da su ulaz u i izlaz y zapravo odstupanja od stacionarnih vrijednosti, koje su poznate. Takođe parametar b_0 može biti zanemaren zbog fenomena direktnog preslikavanja skokovitih promjena sa izlaza na ulaz. Pretpostavlja se da je na stvarni izlaz $y_u(k)$ superponirana stohastička smetnja $n(k)$, što daje mjerljivi signal $y(k)$, tj. :

$$y(k) = y_u(k) + n(k) \tag{1.2}$$

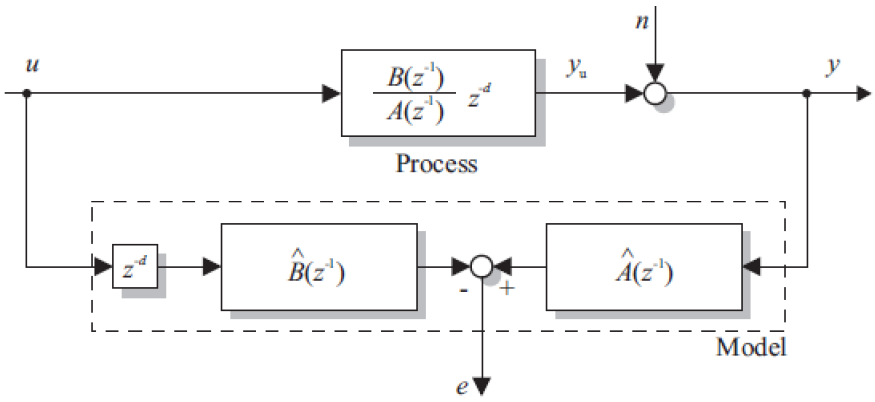
Smetnja mora bit stacionarna sa nultom očekivanom vrijednosti. Relacija između ulaza i izlaza može se napisa u formi diferentne jednačine:

$$y_u\left(k\right)+a_1 y_u\left(k-1\right)+\ldots+a_m y_u\left(k-m\right)=b_1 u\left(k-d-1\right)+\ldots+b_m u\left(k-m-d\right) \tag{1.3}$$

Sada, u prethodnoj jednačini umjesto stvarnog izlaza uzmimo izmjereni izlaz:

$$y\left(k\right)+\sum_{i=1}^{m}\hat{a}_i(k-1)y\left(k-i\right)=\sum_{i=1}^{m}\hat{b}_1(k-1)u\left(k-i-d\right)+e(k) \tag{1.4}$$

Zbog prirode greške $e(k)$, (1.4) predstavlja stohastički sistem. Greška $e(k)$ se naziva rezidual, tipa je generalisane greške i nastaje zbog korištenja izmjerenih vrijednosti $y(k)$ umjesto stvarnih vrijednosti $y_u(k)$.



Slika 1.1. *Blok dijagram metode najmanjih kvadrata*

Prethodna relacija može biti napisana u kompaktnoj vektorskoj formi:

$$y(k) = \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k-1) + e(k) \tag{1.5}$$

gdje se:

$$\boldsymbol{\varphi}^T(k) = \left[-y(k-1)\ \ldots\ -y(k-m)\ \mid\ u(k-1-d)\ \ldots\ \mathrm{u}(k-m-d)\right] \tag{1.6}$$

naziva vektor podataka ili regresor. I vektor parametara:

$$\boldsymbol{\theta}^T(k) = \left[\hat{a}_1\ \ldots\ \hat{a}_m \mid\ \hat{b}_1\ \ldots\ \hat{\mathrm{b}}_m\right] \tag{1.7}$$

Ako se uvede oznaka: $\hat{y}(k \mid k-1) = \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k-1)$ (1.8)

što prestavlja izraz za predikciju modela za jedan korak u naprijed, greška se može tumačiti kao razlika novog mjerenja i predikcije modela. Da bi odredili vektor parametara na osnovu N parova izmjerenih vrijednosti ulaznog i izlaznog signala, uvode se sljedeće pretpostavke: Proces je stacionaran za $k < 0$, red modela m i kašnjenje d su poznati, ulaz $u(k)$ i njegova DC vrijednost U_{00} , DC vrijednost izlaza Y_{00} je poznata. Sada posmatrajmo uzorke mjerenih vrijednosti ulaza i izlaza u trenucima:

$$k = m+d,\ m+d+1, \ldots, m+d+N$$

za koje se može formirati sistem jednačina sa nepoznatih $2m$ parametara. Da bi odredili ove parametre potrebno je da bude : $N \geq 2m-1$, a da bi se otklonio uticaj šuma, N se uzima mnogo više. Pomenutih N+1 jednačina se može napisati u matričnom obliku:

$$\boldsymbol{y}(m+d+N) = \boldsymbol{\Psi}(m+d+N)\hat{\boldsymbol{\theta}}(m+d+N) + \boldsymbol{e}(m+d+N) \tag{1.9}$$

gdje je:

$$\boldsymbol{\Psi}(m+d+N) = \begin{bmatrix} -y(m+d-1) & \cdots & -y(d) & u(m-1) & \cdots & u(0) \\ -y(m+d) & \cdots & -y(d+1) & u(m) & \cdots & u(1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -y(m+d+N-1) & \cdots & -y(d+N) & u(m+N-1) & \cdots & u(N) \end{bmatrix} \tag{1.10}$$

matrica podataka. Sada, na osnovu iznjerenih vrijednosti ulaza i izlaza sistema, koji figurišu u (1.10), potrebno je odrediti vektor parametara (1.7) tako da je vektor greške u (1.9), minimalan. Obzirom da je prema (1.5), greška linearna po vektoru parametara, ovaj problem je rješiv u zatvorenoj formi. Na ovaj problem možemo gledati kao problem optimizacije, gdje se traži minimum kvadratne forme:

$$V = \boldsymbol{e}^T(m+d+N)\boldsymbol{e}(m+d+N) = \sum_{k=m+d}^{m+d+N} e^2(k) \tag{1.11}$$

koja je pod određenim uslovima (koji su detaljno razrađeni u naslovu 2.5) pozitivno definitna kvadratna forma i konveksna. Prema tome, njena stacionarna tačka je i njen miminum, pa diferencirajući funkciju kriterija po vektoru parametara i izjednačavajući sa nulom dobijamo:

$$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{y} \tag{1.12}$$

što je rješenje datog problema, a time i rješenje sistema jednačina (1.9), i predstavlja izraz za estimaciju parametara nerekurzivnom metodom najmanjih kvadrata. Na osnovu gornjih razmatranja vidimo da su

podaci (parovi mjerenih ulaza i izlaza) potrebni za identifikaciju prvo pohranjeni, a zatim procesirani u jednom koraku, što znači da su estimirani parametri sistema na raspolaganju tek na kraju mjerenja, pa je ova metoda povoljna za off-line identifikaciju. Ako bi se ova metoda primjenjivala za on-line identifikaciju, matrica $\boldsymbol{\Psi}$ bi konstantno rasla, što znaći da bi se u svakom koraku procesirali svi podaci do tada pohranjeni, što bi zahtijevalo mnogo procesorskog vremena, pa ova metoda nije efikasna za on-line identifikaciju. Nerekurzivna LS metoda se za dinamičke sisteme može formulisati i preko korelacionih funkcija. Estimacija vektora parametara (1.12) se može izraziti u obliku:

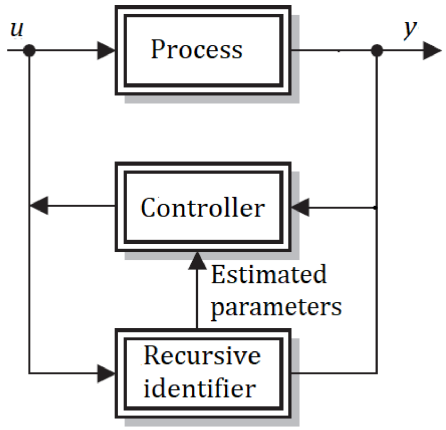
$$\hat{\boldsymbol{\theta}} = \left(\frac{1}{N+1} \boldsymbol{\Psi}^T \boldsymbol{\Psi} \right)^{-1} \frac{1}{N+1} \boldsymbol{\Psi}^T \boldsymbol{y} \tag{1.13}$$

$\frac{1}{N+1}\boldsymbol{\Psi}^T\boldsymbol{\Psi}$ i $\frac{1}{N+1}\boldsymbol{\Psi}^T\boldsymbol{y}$ predstavljaju korelacionu matricu $(2m \times 2m)$ i korelacioni vektor $(1 \times 2m)$, respektivno. U slučaju konvergencije njihovi elementi će težiti konstantnim vrijednostima.

Na rješavanje gornjeg problema optimizacije mogu se nametnuti ograničenja u obliku jednakosti i/ili nejednakosti koji određuju skup dozvoljenih vrijednosti vektora $\boldsymbol{\theta}$. Za ograničenja tipa jednakosti rješenje ovog problema je dato u [1]. Problem sa ograničenjima tipa nejednakosti može se riješiti metodom aktivnog skupa, ovaj metod je izložen u [5].

2. Rekurzivne metode identifikacije

Kod rekurzivnih metoda identifikacije model se ažurira sa svakim raspoloživim mjerenjem, što implicira "poboljšanje" modela s obzirom na prethodni korak. Ovaj pristup omogućuje da se stara mjerenja ne pohranjuju i ovi metodi se obično vežu se za on-line identifikaciju. Mnogi adaptivni sistemi upravljanja su (implicitno ili eksplicitno) zasnovani na rekurzivnoj identifikaciji, gdje parametri modela procesa se osvežavaju u svakom trenutku odabiranja. Ovi vremesko-promjenjivi modeli su upotrijebljeni za određivanje (vremensko-promjenjivih) parametara regulatora, tako da regulator zavisi od prethodnog ponašanja objekta upravljanja (kroz tok informacija: objekat $\rightarrow$ model $\rightarrow$ regulator). Ako je upotrijebljen adekvatan princip podešavanja regulatora, tada će se regulator adaptirati dinamici objekta.



Slika 2.1. *Opšta šema adaptivnog upravljanja*

2.1. *Rekurzivna metoda najmanjih kvadrata*

Rekurzivne jednačine LS metode (eng. least squares) mogu se izvesti polazeći od jednačine (1.12), prema kojoj je estimacija parametara u trenutku k data sa:

$$\hat{\boldsymbol{\theta}}(k) = \boldsymbol{P}(k)\boldsymbol{\Psi}^T(k)\boldsymbol{y}(k) \tag{2.1}$$

gdje je:

$$\boldsymbol{P}(k) = \left(\boldsymbol{\Psi}^T(k)\boldsymbol{\Psi}(k)\right)^{-1} \tag{2.2}$$

$$\boldsymbol{y}(k) = \left[y(1)\ y(2)\ \ldots\ y(k)\right]^T \tag{2.3}$$

$$\boldsymbol{\Psi}^T(k) = \left[\boldsymbol{\varphi}^T(1)\ \boldsymbol{\varphi}^T(2)\ \ldots\ \boldsymbol{\varphi}^T(k)\right] \tag{2.4}$$

Za naredni trenutak $k+1$ analogno vrijedi:

$$\hat{\boldsymbol{\theta}}(k+1) = \boldsymbol{P}(k+1)\boldsymbol{\Psi}^T(k+1)\boldsymbol{y}(k+1) \tag{2.5}$$

što se može napisati u obliku:

$$\hat{\boldsymbol{\theta}}(k+1) = \boldsymbol{P}(k+1)\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}^T\begin{bmatrix}\boldsymbol{y}(k)\\ y(k+1)\end{bmatrix} = \boldsymbol{P}(k+1)\left(\boldsymbol{\Psi}^T(k)\boldsymbol{y}(k) + \boldsymbol{\varphi}(k+1)y(k+1)\right) \tag{2.6}$$

Dalje koristeći (2.2): $\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \left(\boldsymbol{P}(k+1)\boldsymbol{P}^{-1}(k) - \boldsymbol{I}\right)\hat{\boldsymbol{\theta}}(k) + \boldsymbol{P}(k+1)\boldsymbol{\varphi}(k+1)y(k+1)$ (2.7)

Takođe, može se pisati:

$$\boldsymbol{P}(k+1) = \left(\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}\right)^{-1} = \left(\boldsymbol{P}^{-1}(k) + \boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^T(k+1)\right)^{-1} \tag{2.8}$$

pa slijedi:

$$\boldsymbol{P}^{-1}(k) = \boldsymbol{P}^{-1}(k+1) - \boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^T(k+1) \tag{2.9}$$

Nakon uvrštavanja (2.9) u (2.7), imamo:

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \boldsymbol{P}(k+1)\boldsymbol{\varphi}(k+1)\left[y(k+1) - \boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k)\right] \tag{2.10a}$$

Na osnovu (1.8) član $\boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k) = \hat{y}(k+1\,|\,k)$ se može interpretirati kao predikcija izlaza modela na osnovu trenutnih parametara modela i mjerenja do trenutka k. Stoga, faktor u srednjoj zagradi jednačine (2.10), tj.:

$$e(k+1) = y(k+1) - \boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k) \tag{2.10b}$$

predstavlja grešku predikcije za jedan korak naprijed, odnosno grešku modela. Na osnovu (2.9) matricu $\boldsymbol{P}$ možemo računati po obrascu:

$$\boldsymbol{P}(k+1)=\left[\boldsymbol{P}^{-1}(k)+\boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^{T}(k+1)\right]^{-1} \tag{2.11}$$

Ovaj izraz možemo prevesti u rekurzivnu formu, time izbjeći računanje inverzne matrice. To možemo postići pod uslovom nesingularnosti odgovarajućih matrica koristeći lemu o inverziji matrice:

$$\left(A^{-1}+BC^{-1}D\right)^{-1}=A-AB\left(DAB+C\right)^{-1}DA \tag{2.12}$$

Pa na osnovu leme (2.12) i izraza (2.11), slijedi:

$$\boldsymbol{P}(k+1)=\boldsymbol{P}(k)-\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)\left[\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)+1\right]^{-1}\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{P}(k) \tag{2.13}$$

pri tome je izraz u srednoj zagradi skalar. Nakon množenja obje strane (2.13) sa $\boldsymbol{\varphi}(k+1)$ i sređivanja, imamo:

$$\boldsymbol{P}(k+1)\boldsymbol{\varphi}(k+1)=\frac{1}{1+\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)}\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)=\boldsymbol{\gamma}(k) \tag{2.14}$$

Prema tome, dobijamo rekurzivnu formu izraza (2.11) za ažuriranje matrice P :

$$\boldsymbol{P}(k+1)=\left[\boldsymbol{I}-\boldsymbol{\gamma}(k)\boldsymbol{\varphi}^{T}(k+1)\right]\boldsymbol{P}(k) \tag{2.15}$$

Uvrštanjem (2.14) u (2.10), imamo:

$$\hat{\boldsymbol{\theta}}(k+1)=\hat{\boldsymbol{\theta}}(k)+\boldsymbol{\gamma}(k)\left[y(k+1)-\boldsymbol{\varphi}^{T}(k+1)\hat{\boldsymbol{\theta}}(k)\right] \tag{2.16}$$

Dakle, ažuriranu estimaciju parametara računamo kao zbir stare estimacije sa proizvodom korekcionog vektora $\boldsymbol{\gamma}$ i greške modela $\boldsymbol{e}$. Jednačine (2.16), (2.15) i (2.14) predstavljaju jednačine rekurzivne metode najmanih kvadrata za estimaciju parametara sistema na osnovu mjerenja njegovog ulaza i izlaza. Početne vrijednosti $\hat{\boldsymbol{\theta}}(0)$ i $\boldsymbol{P}(0)$ mogu se odabrati na više načina: kao izlaz iz algoritma nerekurzivne LS metode na osnovu najmanje $2m$ jednačina; koristeći poznate apriorne vrijednosti, ako su poznate; ili u pogodnom obliku kao:

$$\boldsymbol{P}(0)=\alpha^{2}\boldsymbol{I}\,,\quad \hat{\boldsymbol{\theta}}(0)=\boldsymbol{0} \tag{2.17}$$

Ovaj izbor je opravdan sljedećim razmatranjem: iterirajući relaciju (2.9) od 0 do k dobijamo:

$$\boldsymbol{P}^{-1}(k)=\boldsymbol{P}^{-1}(0)+\boldsymbol{\Psi}^{T}(k)\boldsymbol{\Psi}(k) \tag{2.18}$$

Ako se izabere da je $\boldsymbol{P}(0)=\alpha^{2}\boldsymbol{I}$, za dovoljno veliko α (2.18) prelazi u (2.2), tj. u slučaj definisan za nerekurzivni LS metod. Analogno, iterirajući (2.10), dobijamo:

$$\hat{\boldsymbol{\theta}}(k)=\boldsymbol{P}(k)\left[\boldsymbol{\Psi}(k)y(k)+\boldsymbol{P}^{-1}(0)\hat{\boldsymbol{\theta}}(0)\right] \tag{2.19}$$

Ako je $\hat{\boldsymbol{\theta}}(0) = \boldsymbol{0}$ (2.19) prelazi u relaciju za računanje $\hat{\boldsymbol{\theta}}(k)$ prema nerekurzivnoj LS metodi, tj. u (1.12).

2.2. Real-time identifikacija

U ovom naslovu su izložene određene modifikacije rekurzivnog LS algoritma koje se koriste za estimaciju vremenski promjenjivih parametara, što se naziva i 'real-time identifikacija' [2]. Kada se osobine procesa mijenjaju u vremenu, rekurzivni algoritam u kojem su izvedene određene modifikacije je u mogučnosti da prati vremeski promjenjive parametre modela koji opisuje taj proces. Postoji više pristupa u modifikaciji rekurzivnog LS algoritma za real-time identifikaciju. Neki od njih su:

- Upotreba težinskog faktora koji daje težinu izmjerenim podacima o procesu tako da, kako podaci zastarjevaju tako njihovi težinski kojeficijenti eksponencijalno opadaju. Ovaj faktor se naziva eksponencijalni faktor iščezavanja (exponential forgetting factor).
- Upotreba Kalmanovog filtra kao estimatora parametara.
- Upotreba rekurzivnog LS algoritma sa vremenskim okvirom (eng. windowed recursive LS method), gdje se zanemaruju podaci van vremenskog okvira sa ishodištem u sadašnjem trenutku.
- Upotreba varijabinog težinskog faktora i varijabilne matrice kovarijanse.

2.2.1. Rekurzivna metoda najmanjih kvadrata sa ekponencijalnim faktorom iščezavanja

U izvođenju ove metode krenimo od funkcije kriterija:

$$V = \boldsymbol{e}^T(m+d+N)W(m+d+N)\boldsymbol{e}(m+d+N) = \sum_{k=m+d}^{m+d+N} w(k)e^2(k) \tag{2.20}$$

gdje je W simetrična pozitivno definitna matrica:

$$W = \begin{bmatrix} w(m+d) & 0 & \dots & 0 \\ 0 & w(m+d+1) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & w(m+d+N) \end{bmatrix} \tag{2.21}$$

Za ovu funkciju kriterija, primjenjujući analogan postupak kao u naslovu 1, imamo:

$$\hat{\theta} = \left(\boldsymbol{\Psi}^T W \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^T W \boldsymbol{y} \tag{2.22}$$

Izborom funkcije kriterija (2.20) smo različitim $e(k)$ dodijelili različite težinske faktore. Izborom :

$$w(k) = \lambda^{m+d+N-k}, \quad 0<\lambda<1 \tag{2.23}$$

ovi težinski faktori opadaju ekponencijalno polazeći od sadašnjeg trenutka ka prošlim. λ se zove faktor iščezavanja (eng. forgetting factor). Pristizanjem novih mjerenja težinska matrica se, prema tome, ažurira kao:

$$W(k+1)=\begin{bmatrix}\lambda W(k) & \boldsymbol{0}\\ \boldsymbol{0}^T & 1\end{bmatrix} \tag{2.24}$$

Ako označimo:

$$\boldsymbol{P}_W(k)=\left[\boldsymbol{\Psi}^T(k)W(k)\boldsymbol{\Psi}(k)\right]^{-1} \tag{2.25}$$

na osnovu (2.22) $\hat{\boldsymbol{\theta}}$ i $\boldsymbol{P}$ se ažuriraju prema:

$$\begin{aligned}\hat{\boldsymbol{\theta}}(k+1)&=\boldsymbol{P}_W(k+1)\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}\begin{bmatrix}\lambda W(k) & \boldsymbol{0}\\ \boldsymbol{0}^T & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{y}(k)\\ y(k+1)\end{bmatrix}\\ &=\boldsymbol{P}_W(k+1)\left(\lambda\boldsymbol{\varphi}^T(k)W(k)\boldsymbol{y}(k)+\boldsymbol{\varphi}(k+1)y(k+1)\right)\\ &=\boldsymbol{P}_W(k+1)\left(\lambda\boldsymbol{P}_W^{-1}(k)\hat{\boldsymbol{\theta}}(k)+\boldsymbol{\varphi}(k+1)y(k+1)\right)\end{aligned} \tag{2.26}$$

$$\begin{aligned}\boldsymbol{P}_W(k+1)&=\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}\begin{bmatrix}\lambda W(k) & \boldsymbol{0}\\ \boldsymbol{0}^T & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\Psi}(k)\\ \boldsymbol{\varphi}^T(k+1)\end{bmatrix}^{-1}\\ &=\left(\lambda\boldsymbol{\Psi}^T(k)W(k)\boldsymbol{\Psi}(k)+\boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^T(k+1)\right)^{-1}\\ &=\left[\lambda\boldsymbol{P}_W^{-1}(k)+\boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^T(k+1)\right]^{-1}\end{aligned} \tag{2.27}$$

Pa je:

$$\boldsymbol{P}_W^{-1}(k+1)=\lambda\boldsymbol{P}_W^{-1}(k)+\boldsymbol{\varphi}(k+1)\boldsymbol{\varphi}^T(k+1) \tag{2.28}$$

Na osnovu (2.26) i (2.28), imamo:

$$\begin{aligned}\hat{\boldsymbol{\theta}}(k+1)&=\hat{\boldsymbol{\theta}}(k)+\left(\lambda\boldsymbol{P}_W(k+1)\boldsymbol{P}_W^{-1}(k)-\boldsymbol{I}\right)\hat{\boldsymbol{\theta}}(k)+\boldsymbol{P}_W(k+1)\boldsymbol{\varphi}(k+1)y(k+1)\\ &=\hat{\boldsymbol{\theta}}(k)+\boldsymbol{P}_W(k+1)\boldsymbol{\varphi}(k+1)\left[y(k+1)-\boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k)\right]\end{aligned} \tag{2.29}$$

Primjenom leme o inverziji matrice na (2.28), imamo:

$$\boldsymbol{P}_W(k+1)=\frac{1}{\lambda}\boldsymbol{P}_W(k)-\frac{1}{\lambda}\boldsymbol{P}_W(k)\boldsymbol{\varphi}(k+1)\left[\boldsymbol{\varphi}^T(k+1)\boldsymbol{P}_W(k)\boldsymbol{\varphi}(k+1)+1\right]^{-1}\boldsymbol{\varphi}^T(k+1)\frac{1}{\lambda}\boldsymbol{P}_W(k) \tag{2.30}$$

$$\boldsymbol{P}_W(k+1)\boldsymbol{\varphi}(k+1)=\frac{1}{\lambda+\boldsymbol{\varphi}^T(k+1)\boldsymbol{P}_W(k)\boldsymbol{\varphi}(k+1)}\boldsymbol{P}_W(k)\boldsymbol{\varphi}(k+1)=\boldsymbol{\gamma}_W(k) \tag{2.31}$$

Prema tome, dobijamo rekurzivnu formu izraza (2.28) :

$$\boldsymbol{P}_W(k+1)=\left[\boldsymbol{I}-\boldsymbol{\gamma}_W(k)\boldsymbol{\varphi}^T(k+1)\right]\boldsymbol{P}_W(k)\frac{1}{\lambda} \tag{2.32}$$

Uvrštanjem (2.31) u (2.29), imamo:

$$\hat{\boldsymbol{\theta}}(k+1)=\hat{\boldsymbol{\theta}}(k)+\boldsymbol{\gamma}_W(k)\left[y(k+1)-\boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k)\right] \tag{2.33}$$

Prema tome, rekurzivne jednačine metode najmanjih kvadrata sa ekponencijalnim faktorom iščezavanja imaju oblik:

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \boldsymbol{\gamma}(k)\left[y(k+1) - \boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k) \right] \tag{2.34a}$$

$$\boldsymbol{\gamma}(k) = \frac{1}{\lambda + \boldsymbol{\varphi}^T(k+1)\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)} \boldsymbol{P}(k)\boldsymbol{\varphi}(k+1) \tag{2.34b}$$

$$\boldsymbol{P}(k+1) = \left[\boldsymbol{I} - \boldsymbol{\gamma}(k)\boldsymbol{\varphi}^T(k+1) \right] \boldsymbol{P}(k)\frac{1}{\lambda} \tag{2.34c}$$

Na osnovu (2.34) slijedi da smanjenjem vrijednosti λ povećavaju se vrijednosti matrice $\boldsymbol{P}$, što se odražava na povećanje kovarijanse estimiranih parametara time se povećava uticaj šuma na estimaciju parametara se. S druge strane, na osnovu (2.34b) i (2.34a) slijedi da smanjenjem λ povećava se vektor korekcije $\boldsymbol{\gamma}(k)$, i to se odražava na povećan uticaj novog mjerenja na korekciju vektora parametara prema (2.34a), što je poželjno kod identifikacije sistema sa vremenski promjenjivim parametrima, tj. za real-time identifikaciju. Stoga se λ određuje kao kompromis između bolje prigušivanja smetnji - manje kovarijanse $(\lambda \to 1)$ i boljeg praćenja promjena kod estimacije vremenski promjenjivih parametara $(\lambda < 1)$.

2.2.2. Rekurzivna LS metoda sa konačnim vremenskim okvirom

Ova varijanta rekurzivnog LS algoritma uzima u obzir mjerene podatke samo na određenom vremenskom razmaku kojeg ćemo ovdje nazvati vremenski okvir (eng. sliding window). Drugim riječima, za estimaciju parametara u trenutku k uzimaju se u obzir mjereni podaci o sistemu kojeg identificiramo u tom trenutku i konačan slijed prethodnih mjerenja. Označimo sa N ukupan broj mjerenja koji se uzimaju u obzir, i taj broj nazivamo širina vremenskog okvira. Kod ovog metoda za funkciju kriterija se uzima funkcija:

$$V = \sum_{j=k-N+1}^{k} \left[y(j) - y(j \mid j-1) \right]^2 = \sum_{j=k-N+1}^{k} e^2(j) \tag{2.35}$$

Ovaj algoritam starta u trenutku $k = N$. Neka je $k \geq N$ tekući korak iteracije ovog algoritma. Iteracija u ovom trenutku se sastoji od dva koraka. Ovo se može pokazati rješavajući problem optimizacije za kriterij (2.35), gdje je $y(j)$ mjereni izlaz sistema i gdje je $y(j \mid j-1)$ određeno sa (1.8), za mjerene ulaze i izlaze . U prvom koraku (updating) se ažuriraju vrijednosti $\hat{\boldsymbol{\theta}}(k)$ i $\boldsymbol{P}(k)$ na osnovu novopristiglih mjerenja u trenutku k , na onovu relacija:

$$\boldsymbol{\gamma}(k) = \frac{1}{1 + \boldsymbol{\varphi}^T(k+1)\underline{\boldsymbol{P}}(k)\boldsymbol{\varphi}(k+1)} \underline{\boldsymbol{P}}(k)\boldsymbol{\varphi}(k+1)\boldsymbol{\gamma}(k) \tag{2.36a}$$

$$\boldsymbol{P}(k+1) = \left[\boldsymbol{I} - \underline{\boldsymbol{\gamma}}(k)\boldsymbol{\varphi}^T(k+1) \right] \underline{\boldsymbol{P}}(k) \tag{2.36b}$$

$$\hat{\boldsymbol{\theta}}(k+1) = \underline{\hat{\boldsymbol{\theta}}}(k) + \underline{\boldsymbol{\gamma}}(k)\left[\, y(k+1) - \boldsymbol{\varphi}^T(k+1)\underline{\hat{\boldsymbol{\theta}}}(k)\right] \tag{2.36c}$$

U drugom koraku (downdating) se otklanja uticaj mjerenih podataka koji su u trenutku k izašli van vremenskog okvira širine N. Ovaj korak se odvija prema relacijama:

$$\underline{\boldsymbol{\gamma}}(k) = \frac{1}{1 - \boldsymbol{\varphi}^T(k-N)\boldsymbol{P}(k)\boldsymbol{\varphi}(k-N)}\boldsymbol{P}(k)\boldsymbol{\varphi}(k-N) \tag{2.37a}$$

$$\underline{\boldsymbol{P}}(k+1) = \left[\boldsymbol{I} + \underline{\boldsymbol{\gamma}}(k)\boldsymbol{\varphi}^T(k-N)\right]\boldsymbol{P}(k) \tag{2.37b}$$

$$\underline{\hat{\boldsymbol{\theta}}}(k+1) = \hat{\boldsymbol{\theta}}(k) - \underline{\boldsymbol{\gamma}}(k)\left[\, y(k-N) - \boldsymbol{\varphi}^T(k-N)\hat{\boldsymbol{\theta}}(k)\right] \tag{2.37c}$$

Na osnovu (2.37) vidimo da se u drugom koraku može pojaviti singularno rješenje kada je:

$$1 - \boldsymbol{\varphi}^T(k-N)\boldsymbol{P}(k)\boldsymbol{\varphi}(k-N) = 0 \tag{2.38}$$

Ovo dalje implicira da je matrica: $$\boldsymbol{P}(k) = \sum_{j=k-N+1}^{k} \boldsymbol{\varphi}(j)\boldsymbol{\varphi}^T(j) \tag{2.39}$$

singularna, što je, prema onome što je izloženo u naslovu 2.5, indikator da sistem kojeg identificiramo nije konzistentno pobuđen u na vremenskom okviru (prozoru) $\left[k-N+1,\ k\right]$. Za iteraciju u kojoj vrijedi (2.38) preskaće se ažuriranje $\boldsymbol{\theta}(k)$ i $\boldsymbol{P}(k)$.

Umjesto računanja inicijalnih vrijednosti prema relacijama (1.12) i (2.2), inicijalne vrijednosti za ovaj metod se mogu računati prema relacijama (2.17), s tim da u inicijalnom vremenskom okviru vrijedi:

$$\boldsymbol{\varphi}(-N) = \boldsymbol{\varphi}(-N+1) = \cdots = \boldsymbol{\varphi}(-n-1) = \left[0\ 0\ \ldots\ 0\right]^T \tag{2.40a}$$

$$\boldsymbol{\varphi}(-n) = \begin{bmatrix} \alpha^{-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix},\ \boldsymbol{\varphi}(-n+1) = \begin{bmatrix} 0 \\ \alpha^{-1} \\ \vdots \\ 0 \end{bmatrix},\ \ldots,\ \boldsymbol{\varphi}(-1) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \alpha^{-1} \end{bmatrix} \tag{2.40b}$$

$$y(-N) = y(-N+1) = \cdots = y(-1) = 0 \tag{2.40c}$$

Ovim pretpostavljenim inicijalnim vrijednostima se u trenutku $k = N$ u potpunosti otklanja uticaj početnih vrijednosti (2.17), što je pokazano u [7]. Za ove početne vrijednosti algoritam starta u trenutku $k = 0$, čime je postignuta i homogenost prelaza između inicijalizacije i iteracije algoritma.

2.2.3. Kalmanov filter

Sa ciljem izvođenja rekurzivnih jednačina Kalmanovog filtra za estimaciju parametara, prvo navedimo jednačine Kalmanovog filtra za estimaciju stanja linearnog procesa. Posmatrajmo linearni diskretni dinamički MIMO (Multi Input Multi Output) sistem opisan sistemom dvije matrične jednačine-jednačinom stanja sistema i jednačinom mjerenja:

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k\text{-}1} + \boldsymbol{B}_k \boldsymbol{u}_{k\text{-}1} + \boldsymbol{v}_{k\text{-}1} \tag{2.41a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_k \boldsymbol{x}_k + \boldsymbol{e}_k \tag{2.41b}$$

Ovdje zbog preglednosti koristimo indeksnu notaciju. Vektori v_k i $\boldsymbol{e}_k$ predstavljaju vektore procesnog šuma i šuma mjerenja, respektivno, i oni su modelirani statistički nezavisnim Gaussovim bijelim šumom. Koristeći operator matematičkog očekivanja E , to bilježimo:

$$E\left[\boldsymbol{v}_k \boldsymbol{v}_j^T\right] = \begin{cases} \boldsymbol{Q}_k, & j \neq k \\ 0, & j = k \end{cases}, \quad \text{tj. } \boldsymbol{v}_k \sim \mathcal{N}(0,\ \boldsymbol{Q}_k) \tag{2.42a}$$

$$E\left[\boldsymbol{e}_k \boldsymbol{e}_j^T\right] = \begin{cases} \boldsymbol{R}_k, & j \neq k \\ 0, & j = k \end{cases}, \quad \text{tj. } \boldsymbol{e}_k \sim \mathcal{N}(0,\ \boldsymbol{R}_k) \tag{2.42b}$$

gdje su: $\boldsymbol{A}_k$, $\boldsymbol{x}_k$, $\boldsymbol{B}_k$, $\boldsymbol{u}_k$, $\boldsymbol{v}_k$, $\boldsymbol{e}_k$, $\boldsymbol{y}_k$, $\boldsymbol{C}_k$ – matrica stanja sistema u diskretnom trenutku vremena k, vektor stanja, matrica ulaza, vektor ulaza, vektor stohastičkih perturbacija stanja, vektor šuma mjerenja, vektor izlaza (mjerenja) i matrica izlaza, respektivno. Matrice $\boldsymbol{Q}_k$ i $\boldsymbol{R}_k$ se zovu matrica kovarijanse stohastičkih perturbacija stanja i matrica kovarijanse greške mjerenja. Kalmanov filter za ovaj sistem je formulisan sa pet rekurzivnih jednačina:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k \boldsymbol{u}_k \tag{2.43a}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{A}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{A}_k^T + \boldsymbol{Q}_k \tag{2.43b}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{C}_k^T (\boldsymbol{C}_k \boldsymbol{P}_{k|k-1} \boldsymbol{C}_k^T + \boldsymbol{R}_k)^{-1} \tag{2.43c}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (\boldsymbol{y}_k - \boldsymbol{C}_k \hat{\boldsymbol{x}}_{k|k-1}) \tag{2.43d}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{C}_k \boldsymbol{R}_k^{-1} \boldsymbol{C}_k^T)^{-1} = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_k\right) \boldsymbol{P}_{k|k-1} \tag{2.43e}$$

gdje (2.43a) predstavlja (a priori) predikciju (ekstrapolaciju, propagaciju) stanja sistema i izvodi se iz jednačine (2.41a), pri čemu je estimirana vrijednos jednaka očekivanoj vrijednosti. (2.43b) predstavlja predikciju matrice kovarijanse estimacije stanja i dobije se takođe iz (2.41a). (2.43c) je Kalmanovo pojačanje i izvodi se iz uslova da je matrica kovarijanse greške korekcije stanja minimalna, primjenjenog na pretpostavljeni oblik jednačine (2.43d). (2.43d) relacija predstavlja korekciju (a posteriori) estimacije stanja. (2.43e) je korekciju matrice kovarijanse i dobije se iz istog uslova kao i za Kalmanovo pojačanje.

- $\hat{\boldsymbol{x}}_{k|k-1}$ -predstavlja estimiranu vrijednost vektora stanja u diskretnom trenutku k na osnovu modela (tj. jednačine stanja) i k-1 prethodnih mjerenja.

- $\boldsymbol{P}_{k|k-1} = \mathrm{cov}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})$ -matrica kovarijanse greške predikcije stanja na osnovu modela i k-1 prethodnih mjerenja.
- $\hat{\boldsymbol{x}}_{k|k}$ -korekcija estimacije stanja sistema na osnovu mjerenja u prethodnim diskretnim trenucima i mjerenja u trenutku k.
- $\boldsymbol{P}_{k|k} = \mathrm{cov}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k})$ - matricu kovarijanse greške korekcije stanja na osnovu mjerenja u trenutku k i k-1 prethodnih mjerenja.

Ovim jednačinama su pridodati početni uslovi za $\boldsymbol{x}$ i $\boldsymbol{P}$. jednačine (2.43) važe i za SISO (Single Input Single Output) i MISO (Multi Inputs Single Output) sisteme, pri čemu treba obratiti pažnju na dimenzionalnost pojedinih članova u jednačinama. Kalmanov filter se formuliše i za sisteme sa ograničenjima na vektor stanja [13].

Ako u estimaciji parametara SISO sistema, vektor stvarnih parametara sistema $\boldsymbol{\theta}$ posmatramo kao vektor stanja sistema, tj.:

$$\boldsymbol{\theta}_k = \boldsymbol{I}\boldsymbol{\theta}_{k-1} + \boldsymbol{v}_{k-1} \tag{2.44a}$$

posmatramo kao jednačinu stanja (2.41a), a jednačinu (1.5), tj.:

$$y(k) = \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k-1) + e(k) \tag{2.44b}$$

posmatramo kao jednačinu mjerenja (2.41b) (s tim da ovdje imamo skalarnu jednačinu), tada jednačinu (2.14) možemo interpretirati kao Kalmanovo pojačanje , tj.:

$$\boldsymbol{K}_k = \frac{1}{R_k + \boldsymbol{\varphi}_k^T \boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k}\boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k \tag{2.45a}$$

pri čemu je kovarijansa greške mjerenja $R_k = 1$. Poredeći (2.14) i (2.43c), imamo:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{P}_{k-1} \tag{2.46}$$

Dalje, poredeći jednačine (2.16) i (2.43d), te (2.15) i (2.43e) može se pisati jednačina korekcije estimacije parametara i jednačinu matrice kovarijanse:

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{K}_k\left(y_k - \boldsymbol{\varphi}_k^T\hat{\boldsymbol{\theta}}_{k-1}\right) \tag{2.45b}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{P}_{k-1|k-1} + \boldsymbol{Q}_k \quad \Rightarrow \quad \boldsymbol{P}_{k|k} = \boldsymbol{P}_{k+1|k} + \boldsymbol{Q}_{k+1} = \boldsymbol{P}_k + \boldsymbol{Q}_{k+1} \tag{2.47}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{\varphi}_k^T\boldsymbol{P}_{k|k-1} = \boldsymbol{P}_{k|k-1} - \frac{\boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T\boldsymbol{P}_{k|k-1}}{R_k + \boldsymbol{\varphi}_k^T\boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k} \tag{2.48}$$

Uvrštavajući (2.47) i (2.46) u (2.48), dobijamo izraz za matricu kovarijanse estimacije parametara:

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} + \boldsymbol{Q}_{k+1} - \frac{\boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T\boldsymbol{P}_{k-1}}{R_k + \boldsymbol{\varphi}_k^T\boldsymbol{P}_{k-1}\boldsymbol{\varphi}_k} \tag{2.45c}$$

Jednačine (2.45) predstavljaju jednačine Kalmanovog filtra za estimaciju parametara modela procesa kojeg identificiramo. Matrica $\boldsymbol{Q}_k$ se obično unaprijed bira kao konstantna dijagonalna matrica. Njeni elementi se biraju kao veći od nule u slučaju estimacije vremenski promjenjivih parametara. Za brzo promjenjive parametre biraju se elementi sa velikim pozitivnim vrijednostima, što rezultira i povečanjem kovarianse estimacije parametara.

2.3. Razmatranje poopštenja rekurzivnih metoda na identifikaciju MISO linearnih sistema i pitanje izbora strukture modela

Prethodno navedene metode su izvedene za identifikaciju linearnog stohastičkog sitema:

$$A(z^{-1})y(k) = B(z^{-1})u(k) + e(k)\ , \qquad e(k) \sim \mathcal{N}(0,\ \sigma^2) \tag{2.49}$$

koji je zapravo ARX (Auto-Regresive with eXogenous input) linearni model sa jednim ulazom i jednom izlazom (SISO model). Ove metode mogu se poopštiti na AR (Auto Regresive), ARMA (Auto Regresive Moving Average), ARMAX (Auto Regresive Moving Average with eXogenous input), BJ (Box Jenkins) i OE(Output Error) linearne modele sa jednim i više ulaza (MISO modeli) . Opšti SISO model ima oblik:

$$A(z^{-1})y(k) = \frac{B(z^{-1})}{F(z^{-1})}u(k) + \frac{C(z^{-1})}{D(z^{-1})}e(k) \tag{2.50}$$

gdje su $A(z^{-1}),\ B(z^{-1}),\ F(z^{-1}),\ C(z^{-1}),\ D(z^{-1})$ polinomi reda $na,\ nb,\ nf,$ nc, nd respektivno, s tim da polinom $B(z^{-1})$ figuriše i kasnjenje koje iznosi nk. $e(k)$ je Gaussov bijeli šum sa nultom očekivanom vrijednosti. Da bi sistem određen ovim modelom bio stabilan polinomi $F(z^{-1}),\ C(z^{-1}),\ D(z^{-1})$ (po nekim autorima $F(z^{-1}),\ C(z^{-1})$) moraju imati nule koje leže unutar jediničnog kruga u ravni $\{z\}$.

Poopštenje rekurzivnih metoda za MISO linearne modele izvodimo polazeći od opšteg MISO linearnog modela:

$$A(z^{-1})y(k) = \frac{\sum_{i=1}^{nu} B_i(z^{-1})u_i(k)}{F(z^{-1})} + \frac{C(z^{-1})}{D(z^{-1})}e(k) = \frac{\boldsymbol{B}^T(z^{-1})}{F(z^{-1})}\boldsymbol{u}(k) + \frac{C(z^{-1})}{D(z^{-1})}e(k) \tag{2.51a}$$

gdje je:

vektor od nu ulaza $\qquad \boldsymbol{u}(k) = \left[u_1(k)\ u_2(k)\ \ldots u_{nu}(k)\right]^T$ (2.51b)

$$\boldsymbol{B}(z^{-1}) = \left[B_1(z^{-1})\ B_2(z^{-1})\ \ldots B_{nu}(z^{-1})\right]^T \tag{2.51c}$$

$$A(z^{-1}) = \sum_{j=0}^{na} a_j z^{-j} \ , \quad a_0 = 1$$

$$B_i(z^{-1}) = \sum_{j=1}^{nb_i} b_j^i z^{-j} \cdot \mathrm{z}^{-nk_j} \ , \quad i = 1,\ 2,\ ...,nu$$

$$F(z^{-1}) = \sum_{j=0}^{nf} f_j z^{-j} \ , \quad f_0 = 1 \qquad (2.51d)$$

$$C(z^{-1}) = \sum_{j=0}^{nc} c_j z^{-j} \ , \quad c_0 = 1$$

$$D(z^{-1}) = \sum_{j=0}^{nd} d_j z^{-j} \ , \quad d_0 = 1$$

. Jednačinu (2.51a) možemo napisati u obliku:

$$y(k) = \left[1 - \frac{A(z^{-1})D(z^{-1})}{C(z^{-1})} \right] y(k) + \frac{D(z^{-1})\boldsymbol{B}^T(z^{-1})}{C(z^{-1})F(z^{-1})} \boldsymbol{u}(k) + e(k) \qquad (2.52)$$

Poredeći (2.52) , (2.49) i imalući u vidu (1.8) -izraz za predikciju modela za jedan korak unaprijed, može se pisati da je:

$$\hat{y}(k \mid k-1) = \left[1 - \frac{A(z^{-1})D(z^{-1})}{C(z^{-1})} \right] y(k) + \frac{D(z^{-1})\boldsymbol{B}^T(z^{-1})}{C(z^{-1})F(z^{-1})} \boldsymbol{u}(k) \qquad (2.53)$$

Za grešku predikcije, koju ovdje obilježavamo sa ε , vrijedi:

$$\hat{\varepsilon}(k) = y(k) - \hat{y}(k \mid k-1) = \frac{D(z^{-1})}{C(z^{-1})} \left[A(z^{-1}) y(k) - \frac{\boldsymbol{B}^T(z^{-1})}{F(z^{-1})} \boldsymbol{u}(k) \right] \qquad (2.54)$$

Da bi ovu jednačinu preveli u rekurzivnu formu, formalno uvodimo smjene:

$$\hat{w}(k) = \frac{\boldsymbol{B}^T(z^{-1})}{F(z^{-1})} \boldsymbol{u}(k) = \frac{\sum_{j=1}^{nu} B_j(z^{-1}) u_j(k)}{F(z^{-1})} \ , \qquad \hat{v}(k) = A(z^{-1}) y(k) - \hat{w}(z^{-1}) \qquad (2.55)$$

Na osnovu smjena (2.55) i izraza (2.54), imamo:

$$\hat{\varepsilon}(k) = \frac{D(z^{-1})}{C(z^{-1})} \hat{v}(k) \qquad (2.56)$$

$$\hat{w}(k) = \sum_{i=1}^{nu} \sum_{j=1}^{nb_i} b_j^i u_i(k-j) - \sum_{j=1}^{n_f} f_j \hat{w}(k-j) \qquad (2.57a)$$

$$\hat{\varepsilon}(k) = \hat{v}(k) + \sum_{j=1}^{nd} d_j \hat{v}(k-j) - \sum_{j=1}^{nc} c_j \hat{\varepsilon}(k-j) \tag{2.57b}$$

$$\hat{v}(k) = y(k) + \sum_{j=1}^{na} a_j y(k-j) - \hat{w}(k) \tag{2.57c}$$

Uvrštavajući (2.57c) u (2.57b) i zamjenom $\hat{w}(k)$ sa (2.57a), imamo da je:

$$\hat{\varepsilon}(k) = y(k) - \boldsymbol{\varphi}(k)\boldsymbol{\theta}^T(k-1) \tag{2.58}$$

gdje je:

$$\boldsymbol{\theta}^T(k) = \left[\hat{a}_1 \ldots \hat{a}_{na} | \, \hat{b}_1 \ldots \hat{b}_{nb_1} | \quad \ldots | \, \hat{b}_1 \ldots \hat{b}_{nb_{nu}} | \, \hat{f}_1 \ldots \hat{f}_{nf} | \, \hat{c}_1 \ldots \hat{c}_{nc} | \, \hat{d}_1 \ldots \hat{d}_{nd} \right] \tag{2.59}$$

vektor parametara sistema kojeg identificiramo i $\boldsymbol{\varphi}$ vektor podataka (regresor):

$$\begin{aligned} \boldsymbol{\varphi}^T(k) = \big[& -y(k-1) \ldots -y(k-na) \,|\, u_1(k-1) \ldots u_1(k-nb_1) \,| \ldots | \, u_{nu}(k-1) \ldots u_{nu}(k-nb_{nu}) \,| \\ & | -\hat{w}(k-1) \ldots -\hat{w}(k-nf) \,|\, \hat{\varepsilon}(k-1) \ldots \hat{\varepsilon}(k-nc) \,| -\hat{v}(k-1) \ldots -\hat{v}(k-nd) \big] \end{aligned} \tag{2.60}$$

Obzirom da je prema (2.58) greška predikcije linearna po vektoru parametara, za estimaciju vektora parametara (2.59) modela sistema čiji je regresor određen sa (2.60) , gdje elemente regresora određujemo prema izvedenim relacijama (2.57), mogu se primijeniti gore navedene metode identifikacije. Opšti linearni MISO stohastički model u sebe uključuje različite linearne modele. Ovi modeli slijede iz (2.51), kao posebni slučajevi:

- Za $nu = 1$ imamo opšti linearni SISO model.
- Za $F(z^{-1}) = 1,\ C(z^{-1}) = 1,\ D(z^{-1}) = 1$
  imamo model sa ARX strukturom , tj. $A(z^{-1})y(k) = B(z^{-1})u(k) + e(k)$. (2.61)
- Za $F(z^{-1}) = 1,\ D(z^{-1}) = 1$
  imamo model sa ARMAX strukturom, tj. $A(z^{-1})y(k) = B(z^{-1})u(k) + C(z^{-1})e(k)$ (2.62)
- Za $A(z^{-1}) = 1$
  imamo model sa BJ, tj. $y(k) = \dfrac{B(z^{-1})}{F(z^{-1})}u(k) + \dfrac{C(z^{-1})}{D(z^{-1})}e(k)$ (2.63)
- Za $A(z^{-1}) = 1,\ C(z^{-1}) = 1,\ D(z^{-1}) = 1$
  imamo model sa OE strukturom, tj. $y(k) = \dfrac{B(z^{-1})}{F(z^{-1})}u(k) + e(k)$ (2.64)

Na pitanje izbora strukture ne postoji generalni odgovor. Može se reći da različite strukture modela imaju sebi svojstven model dinamike šuma. Kod ARX modela, $A(z^{-1})$ ujedno određuje i dinamiku šuma, što može dovesti do pristrasne estimacije parametara. Ova loša strana ARX modela se umanjuje povoljnim odnosom signala i šuma (SNR-Signal to Nouse Ratio). ARMAX model zbog prisustva polinoma

C daje dodatnu fleksibilnost u modelu dinamike šuma. Kod BJ modela dinamika šuma je modelirana odvojeno od dinamike sistema. OE model ima istu prednost kao i BJ, stim da nemamo modeliranu dinamiku šuma.

Da bi sistem određen modelom (2.51) bio stabilan polinomi $F(z^{-1}),\ C(z^{-1}),\ D(z^{-1})$ moraju imati nule koje leže unutar jediničnog kruga u ravni $\{z\}$, s tim u slučaju da nula jednog od polinoma leži van pomenutog kruga potrebno je izvršiti projekciju u jedinični krug. U mnogim praktičnim slučajevima pojava nule van jediničnog kruga je relativno ridak slučaj, pa je prema literaturi [3] dovoljno izvršiti jednostavnu projekciju, čija se umanjena preciznost ne odražava na estimaciju. Ovaj algoritam projekcije se sastoji iz sljedećih koraka:

1. Izabrati faktor $0 \le \mu < 1$.
2. Izračunati $\tilde{\boldsymbol{\theta}}(k) = \boldsymbol{\gamma}(k)\left(y(k+1) - \boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k) \right)$ (za Kalmanov filter je $\boldsymbol{\gamma}(k) = \boldsymbol{K}(k)$)
3. Ažurirati vektor estimacije parametara prema $\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k+1) + \tilde{\boldsymbol{\theta}}(k)$
4. Ispitati da li vektor $\hat{\boldsymbol{\theta}}(k+1)$ leži u oblasti stabilnosti, tj. dali polinomi $F(z^{-1}),\ C(z^{-1}),\ D(z^{-1})$ čiji su kojeficijenti jednaki odgovarajućim elementima vektora $\hat{\boldsymbol{\theta}}(k+1)$ imaju nule oblasti stabilnosti. Ako imaju, preći na korak 6. Ako ne, preći na korak 5.
5. Staviti $\tilde{\boldsymbol{\theta}}(k) \sim \mu\tilde{\boldsymbol{\theta}}(k)$ i preći na korak 3.
6. Stop.

2.4. Konvergencija

Ako jednačinu (1.9) , tj $\boldsymbol{y} = \boldsymbol{\Psi\theta} + \boldsymbol{e}$, gdje je $\boldsymbol{\theta}$ vektor stvarnih parametara, uvrstimo u (1.12), tada za očekivanu vrijednost vektora estimiranih parametara za nerekurzivnu LS metodu imamo:

$$\begin{aligned} E\left\{\hat{\boldsymbol{\theta}}\right\} &= E\left\{\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Psi\theta} + \left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\right\} \\ &= \boldsymbol{\theta} + E\left\{\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\right\} = \boldsymbol{\theta} + \boldsymbol{b} \end{aligned} \tag{2.65}$$

gdje se vektor $\boldsymbol{b} = E\left\{\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\right\}$ naziva bias ili sistemska greška. Ako bias postaje ravan nul-vektoru tada je vektor estimiranih parametara jednak vektoru stvarnih parametara, tj. $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$. U tom slučaju kažemo da je estimacija parametara nepristrasna.

Postavlja se pitanje koji uslovi moraju biti zadovoljeni da bi se dobila nepristrasna estimacija. Sljedeća teorema daje potrebne uslove za nepristrasnu estimaciju ($\boldsymbol{b} = \boldsymbol{0}$).

Teorema 1: *Ako su parametri procesa (1.3) estimirani nepristrasno LS metodom, tada su* $\boldsymbol{\Psi}^T$ *i* $\boldsymbol{e}$ *nekorelisani i vrijedi* $E\{\boldsymbol{e}\} = 0$ *, i za konačan interval mjerenja N vrijedi:* $\boldsymbol{b} = \boldsymbol{0}$ *.*

Može se pokazati [1] da je bias iščezava, tj. $\lim_{N\to\infty} \boldsymbol{b} = \boldsymbol{0}$ ako je zadovoljena jednakost:

$$\sum_{j=0}^{m} a_j R_{nn}(\tau - j) = 0\ , \quad 1 \le \tau \le m,\ a_0 = 1 \tag{2.66}$$

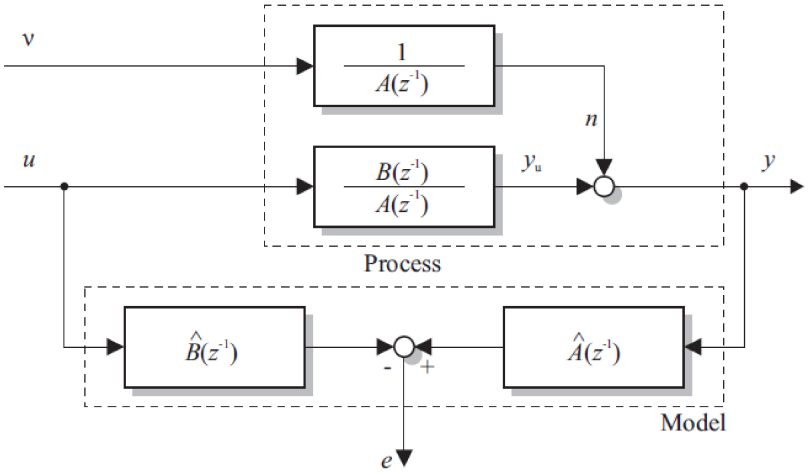
što predstavlja Yule-Walker jednačinu za autoregresivni stohastički proces:

$$\sum_{j=0}^{m} a_j n(k-j) = v(k)\ , \quad v(k) \sim \mathcal{N}(0,\ 1)\ \text{, tj. } A(z^{-1})n(z) = v(z) \tag{2.67}$$

Ovo znači da se smetnja $n(k)$ na izlazu generiše na osnovu bijelog šuma $v(k)$ putem filtra sa prenosnom funkcijom koja određuje dinamiku šuma:

$$G_v(z) = \frac{n(z)}{v(z)} = \frac{1}{A(z^{-1})} \tag{2.68}$$

što predstavlja potreban uslov da bi se garantovala nepristrasna estimacija, tj. $\boldsymbol{b} = \boldsymbol{0}$. Zahtjevana blok struktura procesa za nepristrasnu estimaciju nerekurzivnom LS metodom prestavljena je na narednoj slici.



Slika 2.2. *Struktura procesa koja se zahtjeva da bi parametri bili nepristrasno estimirani nerekurzivnom LS metodom*

Na osnovu slike 2.2 , imamo:

$$y(z) = \frac{1}{A}v(z) + \frac{B}{A}u(z) \tag{2.69}$$

$$e(z) = -\hat{B}u(z) + \hat{A}y(z) = -\hat{B}u(z) + \frac{\hat{A}}{A}v(z) + \hat{A}\frac{B}{A}u(z) \tag{2.70}$$

Ako je proces tačno identificiran, tj. $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$ ($\hat{A} = A,\ \hat{B} = B$ i $\boldsymbol{b} = \boldsymbol{0}$), tada na osnovu (2.70) vrijedi:

$$e(z) = v(z) \tag{2.71}$$

Naredna teorema daje dovoljne uslove za asimptotski nepristrasnu ($\lim_{N\to\infty} \boldsymbol{b} = \boldsymbol{0}$)estimaciju:

Teorema 2: *Parametri procesa (1.3) se mogu estimirati asimptotski nepristrasno metodom najmanjih kvadrata, ako vrijedi:*

$$R_{ee}(\tau) = \sigma_e^2 \delta(\tau) \tag{2.72a}$$

$$E\{e(k)\} = 0 \tag{2.72b}$$

Ovdje je sa $R_{ee}(\tau)$ označena autokorelaciona funkcija signala šuma $e(k)$. Ovo znači da e predstavlja statistički nekolerisan Gaussov bijeli šum sa nultom očekuvanom vrijednosti, tj. $e(k) \sim \mathcal{N}(0,\ \sigma_{ee}^2)$.
Teorema prepostavlja strukturu filtra vezanog za šum koji djeluje na izlazu sistema i sa ovom pretpostavkom estimacija je nepristrasna i za konačan period mjerenja. Pretpostavljeni filter je:

$$G_v(z) = \frac{D(z^{-1})}{C(z^{-1})} = \frac{1}{A(z^{-1})} \tag{2.73}$$

što znači da $A(z^{-1})$ ujedno određuje i dinamiku šuma. Ova struktura modela data na slici (2.2) jeste ARX struktura i vrlo često daje zadovoljavajuće rezultate. Za procese reda većeg od 1, brojnik $D(z^{-1})$ ima oblik polinoma, stoga estimacija parametara sa ARX strukturom modela daje pristrasne rezultate. Može se pokazati da bias raste povečanjem relativne amplitude šuma koji djeluje na izlaz, u odnosu na stvarni signal na izlazu.
Ako su uslovi teoreme 2 zadovoljeni ($\Leftrightarrow E\{\boldsymbol{e}\boldsymbol{e}^T\} = \sigma_e^2 \boldsymbol{I}$) i ako su $\boldsymbol{\Psi}$ i $\boldsymbol{e}$ statistički nezavisni, vrijedi:

$$\begin{aligned}
\operatorname{cov}\Delta\boldsymbol{\theta} &= E\left\{(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})^T\right\} = E\left\{\left((\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\right)\left((\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\right)^T\right\} \\
&= E\left\{(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{\Psi}(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\right\} = E\left\{(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\right\} E\left\{\boldsymbol{e}\boldsymbol{e}^T\right\} E\left\{\boldsymbol{\Psi}(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\right\} \\
&= \sigma_e^2 E\left\{(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\right\} = \sigma_e^2 E\left\{(N+1)(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\right\}\frac{1}{N+1} \\
&= \sigma_e^2 \frac{1}{N+1} E\left\{\hat{\boldsymbol{R}}^{-1}(N+1)\right\}
\end{aligned} \tag{2.74}$$

Prema tome vrijedi:

$$\lim_{N\to\infty} \operatorname{cov}\Delta\boldsymbol{\theta} = \boldsymbol{R}^{-1} \lim_{N\to\infty} \frac{\sigma_e^2}{N+1} = \boldsymbol{\theta} \tag{2.75}$$

Dakle, ako su uslovi teoreme 2 ispunjeni tada je estimacija parametara konzistentna u smislu srednje kvadratne greške.
Za identifikaciju procesa metodom najmanjih kvadrata sa eksponencijalnim faktorom važi teorema u nastavku.

Definicija 1: *Kažemo da je regresor* $\boldsymbol{\varphi}(k) = \left[u(k-1)\ u(k-2)\ \ldots\ u(k-n)\right]^T$ *dosljedno pobuđen sa pobudom reda* n *ako za cjelobrojnu konstantu S i* $\forall j \in \mathbb{N}$*, postoje konstante* $\alpha,\ \beta \in \mathbb{R}$ *tako da vrijedi:*

$$0 < \alpha \boldsymbol{I} \le \sum_{i=j}^{j+S} \boldsymbol{\varphi}(i)\boldsymbol{\varphi}^T(i) \le \beta \boldsymbol{I} < \infty\,. \tag{2.76}$$

Teorema 3: *Ako je niz regresora* $\{\boldsymbol{\varphi}(k)\}$ *dosljedno pobuđen, tada za proizvoljni početni vektor parametara* $\boldsymbol{\theta}_0$*, rekurzivni metod najmanjih kvadrata sa eksponencijalnim faktrom iščezavanja* λ

određen jednačinama (2.34) eksponencijalno konvergira, tj.za proizvoljno $\boldsymbol{\theta}_0$ *postoji* $\gamma > 0$ *tako da za* $\forall k \geq S$ *vrijedi:*

$$\left\| \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(k) \right\|^2 \leq \gamma \lambda^k \left\| \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_0 \right\| \tag{2.77}$$

gdje je $\boldsymbol{\theta}$ *vektor stvarnih parametara.*

Dokaz ove teoreme za SISO sisteme se nalazi u referenci [8], za MIMO dokaz je u [10]. Za $\lambda = 1$ ovaj metod ne konvergira eksponencijalno, nego ima sporiju konvergenciju. Problem spore konvergencije javlja se kod estimacije parametara korištenjem Kalmanovog filtra.

2.5. Identifikabilnost i uslovi dosljedne pobude procesa

Pojam identifikabilnost se odnosi na pitanje da li je stvarni sistem moguće opisati modelom koji se identificira korištenjem neke od metoda identifikacije. Identifikabilnost zavisi od:

- sistema S,
- postavki eksperimenta X,
- strukture modela M,
- metode identifikacije I.

Postoje različite definicije pojma identifikabilnosti, i najčešće se veže za konvergenciju u srednjem i konzistenciju estimacije.

Definicija 2: *Vektor parametara* $\boldsymbol{\theta}$ *modela kaže se da je identifikabilan ako estimirane vrijednosti* $\hat{\boldsymbol{\theta}}$ *konvergiraju u srednjem ka stvarnim vrijednostima parametara* $\boldsymbol{\theta}$ *i ako je estimacija parametara konzistentna u smislu srednje kvadratne greške, tj. ako vrijedi:*

$$\lim_{k \to \infty} E\left\{ \hat{\boldsymbol{\theta}}(k) \right\} = \boldsymbol{\theta} \tag{2.78a}$$

$$\lim_{k \to \infty} cov \Delta \hat{\boldsymbol{\theta}}(k) = \lim_{k \to \infty} E\left\{ \left(\hat{\boldsymbol{\theta}}(k) - \boldsymbol{\theta} \right) \left(\hat{\boldsymbol{\theta}}(k) - \boldsymbol{\theta} \right)^T \right\} = \boldsymbol{0} \tag{2.78b}$$

Definicija 3: *Za signal* $u(k)$ *se kaže da je dosljedno pobuđen sa pobudom reda* n *, ako vrijedi:*

(i) *postoji granična vrijednost:* $r_u(\tau) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} u(i+\tau) u(i)$ (2.79a)

(i) *matrica* $R_u(n) = \begin{bmatrix} r_u(0) & r_u(1) & \cdots & r_u(n-1) \\ r_u(-1) & r_u(0) & \cdots & r_u(n-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_u(1-n) & r_u(2-n) & \cdots & r_u(0) \end{bmatrix}$ (2.79b)

je pozitivno definitna.

Ova definicija je ekvivalentna definiciji 1 jer se gornja matrica može napisati u formi:

$$R_u(n) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\varphi}(i) \boldsymbol{\varphi}^T(i) \tag{2.80}$$

Pretpostavimo da struktura modela M odabrana tako da su ispunjeni uslovi teoreme 2. Da bi u (1.12) postojala inverzna matrica, potrebno je da vrijedi uslov $\det\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)\neq 0$. Da bi $\hat{\boldsymbol{\theta}}$ bio jedinstven globalni minimum kriterija (1.11), potrebno je da je matrica:

$$\frac{d^2V}{d\boldsymbol{\theta}d\boldsymbol{\theta}^T}=\boldsymbol{\Psi}^T\boldsymbol{\Psi} \tag{2.81}$$

pozitivno definitna. Oba navedena uslova su zadovoljena ako vrijedi:

$$\det\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)=\det \boldsymbol{P}^{-1}>0 \tag{2.82}$$

U terminima korelacione matrice u (1.13), ovaj uslov možemo napisati u obliku:

$$\det\left(\frac{1}{N+1}\boldsymbol{\Psi}^T\boldsymbol{\Psi}\right)=\det \hat{\boldsymbol{R}}(N)>0 \tag{2.83}$$

Uslov (2.83) između ostalog implicira da se proces koji se identificira mora dovoljno pobuditi ulaznim signalom. U skladu sa zaključkom (2.75), $\operatorname{cov}\Delta\boldsymbol{\theta}\to 0$ za $N\to\infty$, tako da je estimacija konzistentna.

Sljedeća teorema objedinjuje sve uslove (na sistem, model, metod) koji su potrebni da bi parametri sistema bili identifikabilni:

Teorema 4: *Parametri linearnog, vremenski invarijantnog modela se mogu estimirati konzistentno u smislu srednje kvadratne greške ako su zadovoljeni sljedeći potrebni uslovi:*

- *Red sistema m i kašnjenje d su poznati.*
- *Ulazni signal $u(k)=U(k)-U_{00}$ je tačno mjerljiv, i U_{00} poznato.*
- *Korelaciona matrica $\boldsymbol{R}=(N+1)^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Psi}$ je pozitivno definitna (što povlači uslove dosljedne pobude, stabilnosti, upravljivosti, osmotrivosti procesa).*
- *Stohastička smetnja superponirana na $y(k)=Y(k)-Y_{00}$ je stacionarna, a Y_{00} poznato.*
- *$E\{e(k)\}=0$ i greška $e(k)$ ne smije niti korelisana.*

Kod identifikacije rekurzivnom LS metodom sa faktorom iščezavanja, vektor stvarnih parametara sistema $\boldsymbol{\theta}$ možemo interpretirati kao vektor stanja, pa možemo pisati:

$$\boldsymbol{\theta}(k+1)=\boldsymbol{I}\boldsymbol{\theta}(k)+\boldsymbol{v}(k) \tag{2.84a}$$

$$y(k+1)=\boldsymbol{\varphi}^T(k+1)\hat{\boldsymbol{\theta}}(k)+e(k+1) \tag{2.84b}$$

gdje je $\boldsymbol{v}(k)$ vektor perturbacija parametara. U tom kontekstu, poredeći ove relacije i jednačinu (2.34a), sa jednačinama observera stanja [6]:

$$\hat{\boldsymbol{x}}(k+1)=\boldsymbol{A}\hat{\boldsymbol{x}}(k)+bu(k)+\boldsymbol{h}\left(y(k)-\boldsymbol{c}^T\hat{\boldsymbol{x}}(k)\right) \tag{2.85}$$

slijedi da je:

$$\boldsymbol{A}=\boldsymbol{I},\quad \boldsymbol{h}=\boldsymbol{\gamma}(k),\quad \boldsymbol{c}=\boldsymbol{\varphi}^T(k+1) \tag{2.86}$$

Observer stanja je stabilan ako nule njegove karakteristične jednačine:

$$\det\left(z\boldsymbol{I}-\boldsymbol{A}-\boldsymbol{h}\boldsymbol{c}^{T}\right)=0 \tag{2.87}$$

leže unutar jediničnog kruga u $\{z\}$ ravni. Analogno tome, karakteristična jednačina našeg rekurzivnog estimatora je:

$$\det\left(z\boldsymbol{I}-\boldsymbol{I}+\boldsymbol{\gamma}(k)\boldsymbol{\varphi}^{T}(k)\right)=0 \tag{2.88}$$

Nakon primjene relacije $\det\left(A+uv^{T}\right)=\det A\left(1+v^{T}A^{-1}u\right)$ i niza transformacija, dobijamo:

$$\det\left(z\boldsymbol{I}-\boldsymbol{I}+\boldsymbol{\gamma}(k)\boldsymbol{\varphi}^{T}(k)\right)=(z-1)^{n-1}\left(z-1+\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{\gamma}(k+1)\right) \tag{2.89}$$

Izjednačavanjem posljednje relacije sa nulom dobijamo svojstvene vrijednosti:

$$z_i=1\,,\qquad i=1,\ 2,\ldots,n-1 \tag{2.90a}$$

$$z_n=1-\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{\gamma}(k+1)=\frac{1}{\lambda+\boldsymbol{\varphi}^{T}(k+1)\boldsymbol{P}(k)\boldsymbol{\varphi}(k+1)} \tag{2.90b}$$

Odavde vidimo da LS metod sa ekponencijalnim faktorom λ ima $n-1$ konstantnih svojstvenih vrijednosti i jednu vremenski promjenjivu svojstvenu vrijednost z_n . Pokazuje se da za dovoljno pobuđen sistem za vremenski promjenjivu svojstvenu vrijednost važi:

$$z_n\to 0 \tag{2.91a}$$

Ako je sistem nedovoljno pobuđen: $z_n\to\lambda$ (2.91b)

Iz tog razloga vremenski promjenjiva svojstvena vrijednost z_n se može koristiti kao mjera pobude i za superviziju estimacije vremenski promjenjivih parametara, na primjer kod primjena adaptivnog upravljanja.

2.6. Validacija modela

Da bi model procesa kojeg identificiramo bio validan, potrebno je provjeriti da li je ponašanje dobijenog modela u skladu sa ponašanjem procesa. Ova provjera zavisi od metode identifikacije i tipa evaluacije (rekurzivna, nerekurzivna). U svim slučajevima provjeravaju se:

- Apriorne pretpostavke o metodi identifikacije.
- Slaganje ulaz/izlaz ponašanja identificiranog modela sa izmjerenim vrijednostima ulaza/izlaza procesa kojeg identificiramo.

Kod parametarske identifikcije provjera apriornih pretpostavki se izvodi prema sljedećem:

- Za signal greške: Provjera da li je statistički neovisan, tj. dali je $R_{ee}(\tau)=0$ za $\tau\neq 0$, i da li je statistički neovisan od ulaznog signala, tj. $R_{ue}(\tau)=0$ i $E\{e(k)\}=0$.

- Za matricu kovarijanse greške parametara: Provjera da li se varijanse greške estimacije parametara smanjuju sa porastom vremena i da li su dovoljno male.

Dodatna evaluacija modela bazira se na:

1. poređenju izlaza modela $\hat{y}(k)$ i mjerenog izlaza procesa $y(k)$ i to za: ulazni signal korišten za identifikaciju, druge ulazne signale. Ovo poređenje vršimo kada imamo povoljan SNR.
2. Poređenju kroskorelacionih funkcija $R_{uy}(\tau)$ modela i sistema.

Za proces sa povoljnim SNR-om, mogu se direktno porediti izlaz procesa i izlaz modela, čija razlika se zove greška modela ili greška simulacije. Za ARX model ona je:

$$\Delta y(k) = y(k) - \hat{y}(k) = \left(\frac{B(z^{-1})}{A(z^{-1})} - \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} \right) u(k) + n(k) \tag{2.92}$$

odakle se vidi da za veliki iznos šuma $n(k)$, greška $\Delta y(k)$ je izdominirana šumom i ta greška ne predstavlja dosljedno grešku modela. I u tom slučaju treba upotrijebiti korelacione funkcije. Ako je dinamika šuma dobro modelirana (npr. BJ model), imamo:

$$\Delta y(k) = y(k) - \hat{y}(k) = \left(\frac{B(z^{-1})}{A(z^{-1})} - \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} \right) u(k) + \left(n(k) - \frac{\hat{D}(z^{-1})}{\hat{C}(z^{-1})} \right) \hat{v}(k) \tag{2.93}$$

pri čemu se može odrediti predikcija jedan korak naprijed na onovu estimata $\hat{v}(k-1)$.

Prikladan kriterij za poređenje različitih modela jeste:

$$V = \frac{1}{N} \sum_{k=1}^{N} \Delta y^2(k) \tag{2.94}$$

Ostali testovi uključuju statističke pokazatelje kao što su srednja vrijednost greški, varijansa i momenti višeg reda. Takođe, kod off-line identifikacije, može se izvršiti validacija dobijenog modela koristeći podatke o procesu koji nisu upotrijebljeni u procesu identifikacije. Ovo nazivamo kros-validacijom. Obično, podaci o procesu se podjele u dvije cjeline. Jedan dio koji obično iznosi 2/3 od ukupne količine podataka, se upotrebljava u identifikaciji procesa, preostali dio otpada na validaciju.

2.7. Bayesovski pristup problemu rekurzivne identifikacije

U prethodnim izlaganjima θ i mjerene veličine ulaza i izlaza procesa su posmatrani kao determinističke veličine. U Bayesovskom pristupu rješavanja problema rekurzivne estimacije, vektor parametara sistema se posmatra kao vektor slučajnih varijabli. Na osnovu posmatranja drugih slučajnih varijabli koreliranih sa vektorom parametara donosimo određene zaključke o vrijednosti vektora parametara. Moglo bi se reći da je Kalmanov filter formulisan koristeći Bayesovski pristup, gdje je pretpostavljeno da vektor stanja koji je neopservabilan, je korelisan sa izlazom sistema, tako da na osnovu mjerenja izlaza, može se estimirati vektor stanja.

Po Bayesovskom pristupu θ posmatramo kao slučajni vektor sa određenom apriornom funkcijom gustine raspodjele. Izmjereni izlazi y^k i ulazi u^k su korelirani sa θ. Ovdje su sa y^k označene sve

izmjerene vrijednosti izlaza do diskretnog trenutka k . U trenutku k potrebno je odrediti aposteriornu funkciju gustine raspodjele $p(\boldsymbol{\theta} \mid y^k, u^k)$.

$\hat{\boldsymbol{\theta}}(k)$ može biti određeno kao uslovna očekivana vrijednost:

$$\hat{\boldsymbol{\theta}}(k) = E\left\{\boldsymbol{\theta} \mid y^k, u^k\right\} \tag{2.95}$$

Ovako određen estimat vektora parametara procesa minimizira kovarijansu $E\left\{\left(\hat{\boldsymbol{\theta}}(k)-\boldsymbol{\theta}\right)\left(\hat{\boldsymbol{\theta}}(k)-\boldsymbol{\theta}\right)^T\right\}$

Zadatak je odrediti zakonitost vremenske promjene (2.95) ili funkcije gustine vjerovatnoće, što je složen problem pa se pribjegava aproksimativnim rješenjima.

Bayesovski pristup se najbolje reflektira u uslovima i pretpostavci naredne teoreme.

Teorema: *Neka je $\boldsymbol{\varphi}(k)$ vektor podataka procesa koji je funkcija od y^{k-1} i u^{k-1} , gdje je $\{e(k)\}$ niz nezavisnih Gaussovih varijabli $e(k) \sim \mathcal{N}\left(0,\ r(k)\right)$. Neka su apriorne funkcije raspodjele od $\boldsymbol{\theta}$ takođe Gaussove sa očekivanom vrijednsti $\boldsymbol{\theta}_0$ i matricom kovarijanse $\boldsymbol{P}_0$,tj. $\mathcal{N}\left(\boldsymbol{\theta}_0,\ \boldsymbol{P}_0\right)$. Tada je aposteriorna funkcija gustine raspodjele $p(\boldsymbol{\theta} \mid y^k, u^k)$ takođe Gaussova sa očekivanom vrijednosti $\hat{\boldsymbol{\theta}}(k)$ i matricom kovarijanse $\boldsymbol{P}(k)$, tj. $\mathcal{N}\left(\hat{\boldsymbol{\theta}}(k),\ \boldsymbol{P}(k)\right)$, tako da su vrijedi*:

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{\gamma}(k-1)\left[\, y(k) - \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k-1)\right] \tag{2.96a}$$

$$\boldsymbol{\gamma}(k-1) = \frac{1}{r(k) + \boldsymbol{\varphi}^T(k)\boldsymbol{P}(k-1)\boldsymbol{\varphi}(k)}\boldsymbol{P}(k-1)\boldsymbol{\varphi}(k) \tag{2.96b}$$

$$\boldsymbol{P}(k) = \left[\boldsymbol{I} - \boldsymbol{\gamma}(k-1)\boldsymbol{\varphi}^T(k)\right]\boldsymbol{P}(k-1)\frac{1}{\lambda} \tag{2.96c}$$

$$\hat{\boldsymbol{\theta}}(0) = \boldsymbol{\theta}_0\ , \quad \boldsymbol{P}(0) = \boldsymbol{P}_0 \tag{2.97}$$

Vidimo da algoritam određen sa (2.96) ima isti oblik kao i (2.34) rekurzivni LS algoritam sa faktorom iščezavanja, pri čemu je $r(k) = 1/\lambda$. Ovo pokazuje da početne vrijednosti $\hat{\boldsymbol{\theta}}(0)$ i $\boldsymbol{P}(0)$ mogu biti interpretirane kao apriorno znanje o vektoru parametara $\boldsymbol{\theta}$, i da je optimalna vrijednost faktora λ jednaka inverznoj vrijednosti varijanse šuma. Ova teorema se dokazuje indukcijom polazeći od Bayesovog pravila za uslovnu vjerovatnoću:

$$P(A \mid B,\ C) = \frac{P(B \mid A,\ \mathrm{C})P(A \mid C)}{P(B \mid C)} \tag{2.98}$$

primjenjenog na posteriornu funkciju gustine vjerovatnoće: $p(\boldsymbol{\theta} \mid y^k) = p(\boldsymbol{\theta} \mid y(k),\ y^{k-1})$. Dokaz je dat u [3].

Bayesovski pristup je teorijski najdosljedniji. Njegov glavniji nedostatak jeste potreba poznavanja funkcije gustine vjerovatnoće parametara koja je generalno nepoznata. Zbog toga ovaj koncept ima teorijski značaj i služi kao polazna tačka za razvoj drugih algoritama i njihovu teorijsku razradu.

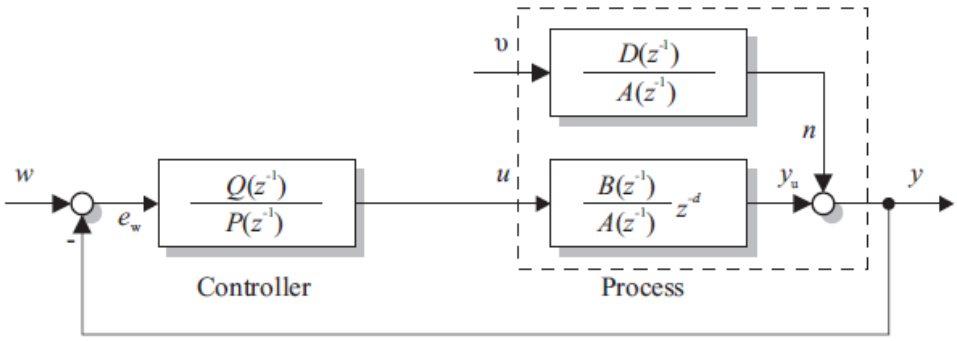
3. Identifikacija sistema u zatvorenoj upravljačkoj konturi

U mnogim aplikacijama, proces se može identificirati isključivo u okviru upravljačke konture. U oblasti adaptivnih sistema upravljanja, model procesa mora biti ažuriran dok se upravljanje odvija u okviru povratne sprege. Bitno je provjeriti da li je određenu metodu identifikacije preporučljivo primijeniti za sistem koji funkcioniše u okviru povratne sprege. Za primjenu metode najmanjih kvadrata, prema teoremi 1, bilo bi potrebno da greška $e(k)$ ne bude korelisana sa komponentama regresora $\varphi(k)$. Ovdje je potrebno provjeriti da li će povratna veza prouzrokovati korelaciju ovih veličina.
Metode identifikacije procesa u okviru povratne sprege se uglavnom dijele na:

- Indirektna identifikacija procesa: Identificira se model ukupnog sistema u zatvorenom, a zatim se izvodi model procesa. Funkcija kontrolera mora biti poznata.
- Direktna identifikacija procesa: Proces se identificira direktno, tj. bez međukoraka koji podrazumijeva nalaženje modela cijele petlje. Kontroler u ovom slučaju ne mora biti poznat.
- Združeni ulaz-izlaz (eng. Joint Input-Output) identifikacija procesa: Izmjereni podaci o sistemu ($u(k),\ \mathrm{y(k)}$) se posmatraju kao izlaz multivarijabilmog sistema ((nu+ny) – dimenzionalni niz) na čijem ulazu djeluje bijeli šum.

3.1. Uslovi identifikabilnosti procesa u zatvorenoj upravljačkoj konturi

U razmatranju uslova identifikabilnosti u zatvorenom pođimo od slučaja identifikacije bez testnog signala, čija je blok struktura predstavljena na slici 3.1.



Slika 3.1. *Blok dijagram identifikacije procesa u zatvorenom bez dodatnog testnog signala*

Teba napomenuti da u slučaju identifikacije sa dodatnim testnim signalom, testni signal se dodaje na upravljački signal kontrolera, tako da vrijedi:

$$u(k) = u_C(k) + s(k) \cdot G_S(z^{-1}) \tag{3.1}$$

gdje je $s(k)$ dodatni testni signal. Neka je prenosna funkcija procesa data sa (1.1). Prenosna funkcija filtra koji definiše dinamiku šuma je:

$$G_v = \frac{n(z)}{v(z)} = \frac{D(z^{-1})}{A(z^{-1})} = \frac{1 + d_1 z^{-1} + \ldots + d_{n_d} z^{-n_d}}{1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a}} \tag{3.2}$$

Neka je kontroler opisan prenosnom funkcijom:

$$G_v = \frac{u(z)}{e_w(z)} = \frac{Q(z^{-1})}{P(z^{-1})} = \frac{q_0 + q_1 z^{-1} + \ldots + q_\nu z^{-\nu}}{1 + p_1 z^{-1} + \ldots + p_\mu z^{-\mu}} \tag{3.3}$$

Bez umanjenja opštosti ovdje možemo pretpostaviti da je $w(z) = 0$, pa je prema slici 3.1. $e_w(z) = -y(z)$. Pretpostavimo da je $v(z)$ nemjerljiv statistički nezavisan šum, tako da je $v \sim \mathcal{N}(0,\ \sigma_v^2)$. Prema slici 3.1. prenosna funkcija sistema u zatvorenom, obzirom na smetnju je :

$$\frac{y(z)}{v(z)} = \frac{G_v(z)}{1 + G_C(z)G_P(z)} = \frac{DP}{AP + BQ\, z^{-d}} \tag{3.4}$$

što se može nepisati i u obliku:

$$\frac{y(z)}{v(z)} = \frac{1 + \beta_1 z^{-1} + \ldots + \beta_r z^{-r}}{1 + \alpha_1 z^{-1} + \ldots + \alpha_l z^{-l}} = \frac{\mathcal{B}(z^{-1})}{\mathcal{A}(z^{-1})} \tag{3.5}$$

Parametri α i β u (3.5) se mogu estimirati, a na osnovu njih se mogu odrediti parametri modela procesa $\hat{a}_i,\ \hat{b}_i,\ \hat{d}_i$. Ovaj postupak je u stvari indirektna metoda identifikacije u zatvorenom. Prije toga treba provjeriti uslove identifikabilnosti, koje izvodimo u daljem tekstu.

Ako upravljačku konturu sa slike 3.1. posmatramo kao odgovarajući filter, možemo smatrati da se signal $y(k)$ generiše na osnovu šuma $v(k)$, kao izlaz iz ARMA procesa, što (3.5) to i jeste.

Relacija (3.4) se za $d = 0$ može napisati u obliku:

$$\left(A + B\frac{Q}{P} \right) y = Dv \tag{3.6}$$

Dodavanjem i oduzimanjem polinoma $S(z^{-1})$ u posljednju relaciju, relacija (3.6) je ekvivalentna sa:

$$\left(Q\left(A + S\right) + \left(QB - PS\right)\frac{Q}{P} \right) y = QDv \tag{3.7}$$

što se može napisati u obliku:

$$\left(A^* + B^*\frac{Q}{P} \right) y = D^* v \tag{3.8}$$

Poređenjem (3.8) i (3.4) može se zaključiti da upravljačka petlja prema slici 3.1, definisana za:

$$\frac{B^*}{A^*} = \frac{BQ - PS}{AQ + SQ}, \quad \frac{D^*}{A^*} = \frac{DQ}{AQ + SQ} \tag{3.9}$$

sa istim regulatorom Q / P , ima isto ponašanje (I/O) kao i originalna petlja. Ovo znači da se proces ne može jednoznačno identificirati (S je proizvojan polinom) iz ponašanja y / v, čak i ako je kontroler Q / P poznat, osim u slučaju da su redovi polimoma A, B i iznos kašnjenja poznati. Na osnovu toga se može furmulisati:

Prvi uslov identifikabilnosti*: Red modela mora biti apriori poznat.*

Redovi polinoma koji figurišu u filtru, tj. relaciji (3.5) , na osnovu (3.4) su :

$$l = \max\left(n_a + \mu,\ \mathrm{n}_b + \nu + d\right) \tag{3.10a}$$

$$r = n_d + \mu \tag{3.10b}$$

Na osnovu (3.4) slijedi da se $n_a + n_b$ nepoznatih parametara $\hat{a}_i$, $\hat{b}_i$ mora odrediti na osnovu l parametara $\hat{\alpha}_i$. Ako u (3.5) polinomi D i $\mathcal{A}$ nemaju zajedničkih korijena, tada da bi parametre procesa jednoznačno odredili, potrebno je da bude:

$$l = n_a + n_b \tag{3.11}$$

ili, na osnovu (3.10a):

$$\max\left(n_a + \mu,\ \mathrm{n}_b + \nu + d\right) \geq n_a + n_b \tag{3.12}$$

tj.:

$$\max\left(\mu - n_b,\ \nu + d - n_a\right) \geq 0 \tag{3.13}$$

Na osnovu ovih razmatranja možemo formulisati:

Drugi uslov identifikabilnosti: *Redovi polinoma kontrolera ν i μ, moraju biti takvi da bude zadovoljeno:*

- $\nu \geq n_a - d$ *kada je* $\nu > \mu - d + n_a - n_b$ *(3.14a)*
- $\mu \geq n_b$ *kada je* $\nu < \mu - d + n_a - n_b$. *(3.14b)*

Parametri $\hat{d}_i$ se jednoznačno mogu odrediti na osnovu parametara $\hat{\beta}_i$ ako je $r \geq n_d$, što je moguće za bilo koji kontroler dok god polinomi $D(z^{-1})$ i $\mathcal{A}(z^{-1})$ nemaju zajedničkih korijena. Ako $D(z^{-1})$ i $\mathcal{A}(z^{-1})$ imaju p zajedničkih korijena, oni se ne mogu identificirati, jer je na raspolaganju samo $l - p$ parametara $\hat{\alpha}_i$ i $r - p$ parametara $\hat{\beta}_i$. I u ovom slučaju drugi uslov identifikabilnosti je:

$$\max\left(\mu - n_b,\ \nu + d - n_a\right) \geq p \tag{3.14c}$$

3.2. Direktna identifikacija procesa u zatvorenoj upravljačkoj konturi

Kod ove metode identifikacije parametre modela procesa estimiramo na osnovu podataka o procesu, odnosno na osnovu izmjerenih vrijednosti ulaza i izlaza. Pri tome primjenjujemo metode identifikacije razmatrane za proces u otvorenom, pri tome imajući u vidu određene uslove i ograničenja. Uzmimo da model procesa koji uvažava dinamiku formiranja smetnje ima ARMAX strukturu, tj.:

$$\hat{A}(z^{-1})y(z) = \hat{B}(z^{-1})z^{-d}u(z) + \hat{C}(z^{-1})v(z) \tag{3.15}$$

Za proces u zatvorenoj konturi imamo:

$$\frac{u(z)}{e_w(z)} = \frac{Q(z^{-1})}{P(z^{-1})} \quad , \quad \text{tj.: } Q(z^{-1})y(z) = -P(z^{-1})u(z) \tag{3.16}$$

Uvrštavajući upravljački zakon (3.16) u (3.15) dobijamo sljedeće:

$$\hat{A}(z^{-1})P(z^{-1})y(z) - \hat{B}(z^{-1})z^{-d}P(z^{-1})u(z) = \hat{D}(z^{-1})P(z^{-1})v(z) \tag{3.17}$$

Nakon pojednostavljenja, dobije se praktično ista jednačina kao i kod sistema u otvorenom sa razlikom što se ulaz $u(k)$ ne može odabrati proizvoljno jer ulaz zavisi od izlaza $y(k)$ i upravljačkog zakona.
Po pitanju identifikabilnosti parametara direktnom metodom, može se pokazati da u slučaju kada vrijedi uslov (2.71) tj. $e(k) = v(k)$ vrijede uslovi identifikabilnosti iz prethodnog naslova. Uslovi identifikabilnosti se mogu vezati za zahtjev da funkcija kriterija:

$$V = h\left(R_\infty(\boldsymbol{\theta})\right) \tag{3.18}$$

ima jedinstven minimum. Gdje je:

$$R_\infty(\boldsymbol{\theta}) = \lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N} E\left\{e^2(k)\right\} \tag{3.19}$$

$e(k)$ je greška predikcije, h je skalarna funkcija. Ovaj uslov je detaljno izložen u referenci [2] za MIMO sisteme.
Za direktne metode identifikacije u zatvorenom, mogu se koristiti one metode za identifikaciju sistema u otvorenom, koje se baziraju na greški predikcije datoj sa (2.10b), tj.:

$$e(k) = y(k) - \hat{y}(k \mid k-1) = y(k) - \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k-1)$$

Pri tome uslov teoreme 1 mora biti zadovoljen, tj. signal greške $e(k)$ nije koreliran sa elementima regresora $\boldsymbol{\varphi}(k)$. Takođe se pretpostavlja da su uslovi identifikabilnosti zadovoljeni. Bitno je naglasiti da funkcija kontrolera ne mora biti poznata. Ako zbog premalog reda kontrolera nije zadovoljen drugi uslov identifikabinosti iz 3.1., moguće je ispoštovati uslove identifikabilnosti, ako se:

1. Koriste dva kontrolera sa različitim parametrima.
2. Uključi kašnjenje u povratnu spregu, tako da je: $d \geq n_a - \nu + p$.
3. Koristi nelinearni ili vremenski promjenjivi kontroler.

Stoga, sve metode koje se baziraju na minimizaciji greške predikcije $e(k)$, osiguraće konzistentne estimacije parametara ukoliko su zadovoljeni navedeni uslovi identifikabilnosti. Znači da se ove metode mogu primijeniti na signale $u(k)$ i $y(k)$ bez obzira na prisutnost povratne sprege.

4. Simulacije

U ovom dijelu rada navedeni su primjeri simulacije adaptivnih sistema upravljanja u kojima se koristi real-time identifikacija linearnog modela objekta upravljanja. Kao metoda identifikacije i zatvorenoj upravljačkoj konturi korištena je direktna metoda. Primjeri su rađeni Matlab R2020a Simulink-u, u kojima je kao blok za real-time identifikaciju korištena u ovom radu kreirana S-funkcija 'Recursive Model Estimator'. Objekat upravljanja u primjerima je izrazito nelinearan hemijski reaktor sa kontinualnim miješanjem (eng. continuous stirred tank reactor) , koji je detaljno opisan i čiji white-box model je izveden u [14]. Pojmovi on-line identifikacija i real-time identifikacija se koriste u smislu kako se koriste u referenci [2], odnosno kako su navedeni u naslovu 2.

4.1. Hemijski reaktor sa kontinualnim miješanjem

Pod pretpostavkom konstantnog volumena tečnosti, reaktor za egzotermičnu, ireverzibilnu reakciju, je opisan dinamičkim (white-box) modelom koji se bazira na molarnom i energetskom balansu za reaktant A:

$$\dot{C}_A = \frac{q}{V}\left(C_{Ai} - C_A\right) - k_0 e^{-\frac{E}{RT}C_A} \tag{4.1a}$$

$$\dot{T} = \frac{q}{V}\left(T_i - T\right) + \left(\frac{\Delta H}{\delta C_p}\right) k_0 e^{-\frac{E}{RT}C_A} + \frac{UA}{V\delta C_p}\left(T_C - T\right) \tag{4.1b}$$

gdje su:

q je volumni protok $[\mathrm{m}^3/\mathrm{s}]$,

V volumen tečnosti $[\mathrm{m}^3]$,

k_0, E, R kojeficijenti koji ulaze u konstantu prirasta hemijske reakcije, koja zavisi od temperature reaktora prema relaciji:

$$k = k_0 e^{-\frac{E}{RT}} \tag{4.2}$$

pri čemu su: k_0 faktor frekvencije, E energija aktivacije, R gasna konstanta.

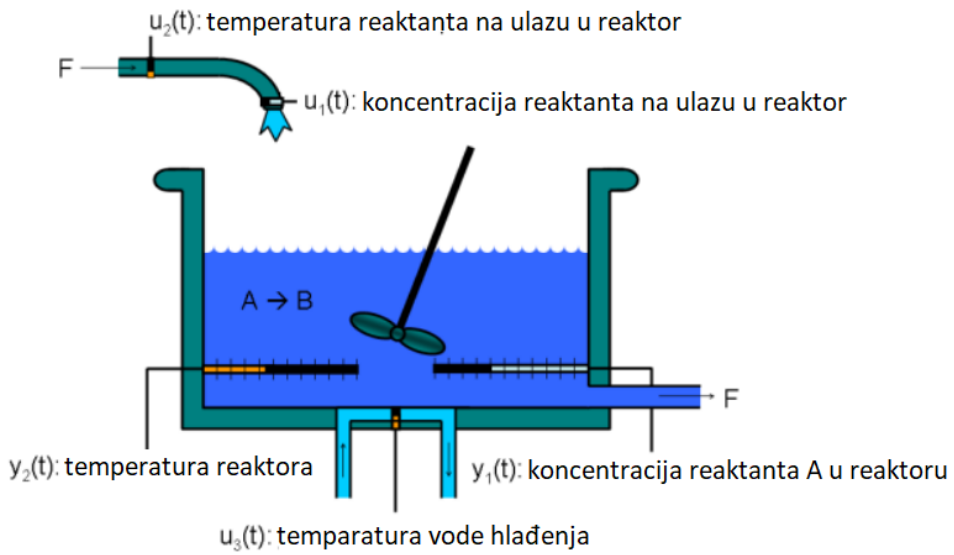
ΔH količina toplote koja se oslobađa tokom reakcije po molu reaktanta A $[\mathrm{J/mol}]$

δ gustina tečnosti $[\mathrm{kg/m}^3]$

C_p kojeficijent toplinskog kapaciteta $[\mathrm{J/gK}]$

U kojeficijent prenosa toplote $[\mathrm{W/(m^2K)}]$

A aktivna površina prenosa toplote $[\mathrm{m}^2]$



Slika 4.1. Objelat upravljanja - hemijski reaktor s kontinualnim miješanjem

Ulazi u objekat upravljanja su:

$u_1(t) = C_{Ai}(t)$ koncentracija reaktanta na ulazu u reaktor [kgmol/m^3]

$u_2(t) = T_i(t)$ temperatura reaktanta na ulazu u reaktor [K]

$u_3(t) = T_C$ temparatura vode hlađenja [K]

Izlazi iz objekta su:

$y_1(t) = x_1(t) = C_A(t)$ koncentracija reaktanta A u reaktoru [kgmol/m^3]

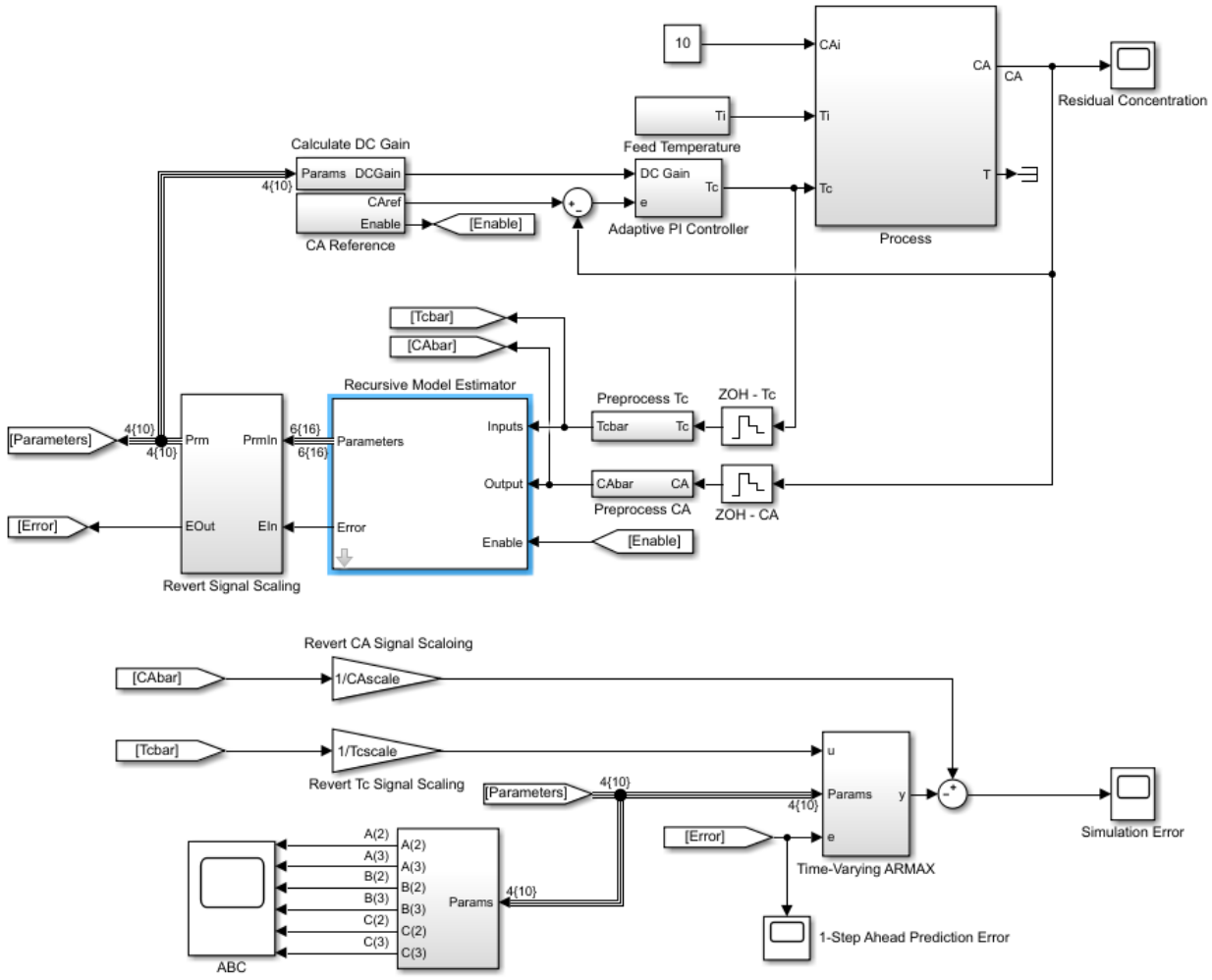
$y_2(t) = x_2(t) = T(t)$ temperatura reaktora [K]

4.2. PI adaptivno upravljanje sa real-time identifikacijom objekta upravljanja

U ovom primjeru sistema upravljanja cilj je regulirati koncentraciju $C_A(t)$ reagensa A u prethodno opisanom reaktoru, gdje je $C_{Aref}(t)$ zadana vrijednost koncentracije tog reagensa. T_C je manipulativna varijabla, čijim mijenjanjem PI regulator otklanja uticaj smetnje uzrokovane fluktuacijama temperature reaktanta na ulazu u reaktor T_i. Usvojeno je da je C_{Ai} konstantnog iznosa. Na T_i =295 [K] je superponiran šum snage 0.0075 [K^2]. Referentni signal C_{Aref} se u trenutku t=400 skokovito mijenja sa 1.5 [kgmol/m^3] na 2 [kgmol/m^3], čime se mijenja i radna tačka procesa.

Ako bi za upravljanje ovim procesom koristili PI regulaciju, pri malim promjenama radne tačke imali bi nezadovoljavajući kvalitet upravljanja a pri većim sistem bi postao nestabilan. Na primjer, ako bi za prvobitno podešen PI, zadali referentnu vrijednost na 2,3 [kgmol/m^3] imali bi kvazioscilatorni odziv sa relativno visokim prvim preskokom, dok za 2,7 [kgmol/m^3] sistem bi bio nestabilan.

U sistemu upravljanja predstavljenog simulink blok dijagramom na slici 1.2, u vremenskim intervalima [0, 200) i [400, 600) C_{Aref} sadrži i bijeli šum snage 0.015, koja je izabrana tako da je SNR=10, što obezbjeđuje dovoljnu pobudu.



Slika 4.2. *Simulink blok-dijagram sistema upravljanja sa adaptivnim PI regulatorom i real-time identifikacijom objekta upravljanja, gdje je primjenjena direktna metoda identifikacije u zatvorenoj regulacijskoj petlji*

Zbog nelinearnosti procesa, koncentracija C_A reaktanta A u reaktoru je mnogo osjetljivija na promjenu manipulativne varijable T_C pri većim koncentracijama C_A. Može se reći da se identifikacijom detektuje ta promjena osjetljivosti. Estimirani parametri su upotrijebljeni za podešavanje pojačanja PI regulatora. Cilj je izbjeći velike iznose pojačanja, zbog čega sistem može ući u nestabilnost.

Blok rekurzivnog estimatora vrši real-time estimaciju parametata prenosne funkcije $C_A(z)/T_C(z)$. Upravljački algoritam adaptivnog upravljanja koristi DC pojačanje ove prenosne funkcije tako što je greška upravljanja podjeljena sa normiranom vrijednosti ovog pojačanja. Ovo normiranje je izvršeno tako da imamo pojačanje 1 na početnoj radnoj tački na kojoj je izvršeno podešavanje regulatora. Tako na primjer, signal regulacione greške će biti podiljen brojem 2 ako pojačanje postane dvostruko u odnosu na prvobitnu vrijednost.

U simulaciji koristimo blok kreirane S-funkcije, koji u ovom primjeru estimira parametre ARMAX modela, određenog sa (2.62), tj.:

$$A(z^{-1})\bar{y}(k) = B(z^{-1})\bar{u}(k) + C(z^{-1})\bar{e}(k) \tag{4.3}$$

Ulazi 'Inputs' i 'Output'u blok odgovaraju $\bar{u}(k)$ i $\bar{y}(k)$ respektivno, koji predstavljaju odstupanja data sa:

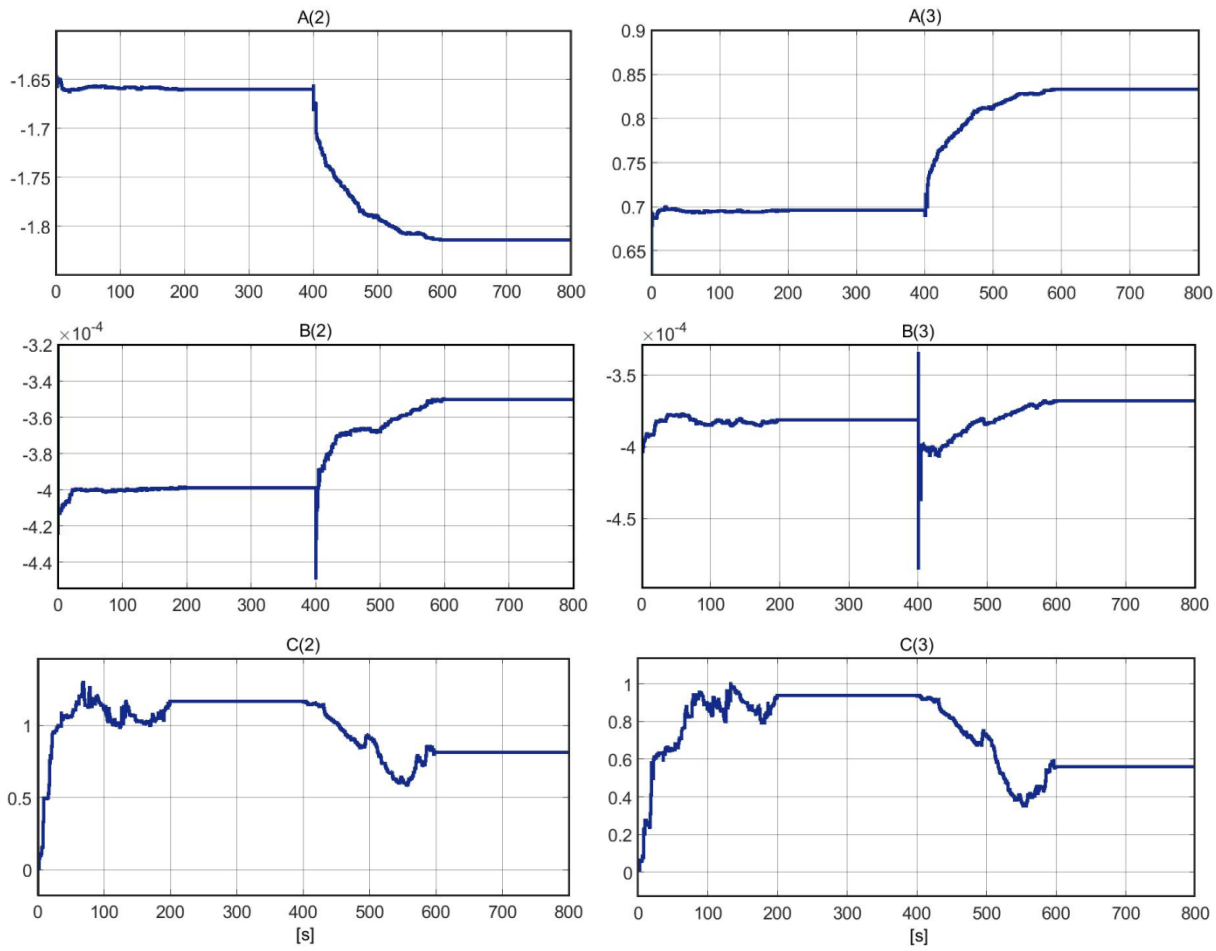
$$\bar{y}(k) = C_A(k) - \bar{C}_A(k), \quad \bar{u}(k) = T_C(k) - \bar{T}_C(k) \tag{4.4}$$

gdje su $\bar{C}_A$ i $\bar{T}_C$ filtriranjem C_A i T_C upotrebom MA (Moving Average) filtra prvog reda, što je implementirano u Preprocess T_C i Preprocess C_A subsistemima. Ovim je zapravo otklonjena DC komponenta signala $C_A(k)$, odnosno $T_C(k)$. Opcijom `Enable` u bloku rekurzivnog estimatora određuje se režim rada bloka vanjskim signalom, tako da u ovom primjeru estimacija parametara se vrši u vremenskim intervalima [0, 200) i [400, 600). Blok ne daje estimate parametara u intervalima [200, 400) i [600, 800) u kojima T_C ne sadrži dovoljnu pobudu za identifikciju sistema u zatvorenom.

U postavkama bloka estimatora 'Recursive Model Estimator' je izabrano:

- Struktura modela: ARMAX
- Broj parametara A(q) (na): 2
- Broj parametara B(q) (na): 2
- Broj parametara C(q) (na): 2
- Kašnjenje ulaza (nk): 0
- Matrica kovarijanse parametara: 100
- Period uzorkovanja: 0.1. Objekat upravljanja ima širinu propusnog opsega 1.25 [Hz] , T_S=0.1 je izabrano tako da 1/0.1 je 20 puta više od širine propusnog opsega.
- Metoda estimacije parametara: Faktor iscezavanja.
- Faktor iscezavanja: $1 - 5 \cdot 10^{-3}$, što odgovara konstanti memorije od $T_0 = T_S / (1 - \lambda) = 100$ [s] , čime je osigurano da u svakoj iteraciji je značajan dio informacija o sistemu korištenih u identifikaciji pripada vremenskom intervalu od 100 [s] .
- Greška estimacije: koristi se za validaciju identifikacijom dobijenog modela.
- Enable port: koristi se. Ova potvrda omogućenosti bloka se upotrebljava kada želimo identificirati parametre modela samo u određenim intervalima vremena kada na referenti ulaz sistema upravljanja injektiramo dodatni testni signal da bi sistem dosljedno pobudili. A blok, tj. estimaciju parametara onemogućiti kada nije injektiran testni signal. U vremenskim razmacima dok je estimacija parametara onemogučena, blok za to vrijeme nastavlja sa računanjem greške predikcije, dok vrijednost parametara bi trebala biti zadržana na vrijednosti kojoj konvergiraju, što prikazuje slika 4.3.

U svakom trenutku odabiranja ovaj blok obezbjeđuje estimirane vrijednosti parametara predpostavljenog modela i grešku $\bar{e}(k)$ koja predstavlja grešku predikcije za jedan korak naprijed. Za izabranu ARMAX strukturu modela u ovom primjeru, izlaz bloka estimatora parametara modela sadrži kojeficijente polinoma $A(z^{-1}),\ B(z^{-1}),\ C(z^{-1})$.



Slika 4.3. *Vremenski dijagrami estimiranih parametara*

Na osnovu slike 4.3., parametri u $A(z^{-1})$ i $B(z^{-1})$ brzo konvergiraju ka estimiranim vrijednostima na vremenskom razmaku [0, 200). To je zbog izabrane velike vrijednosti dijagonalnih elemenata inicijalne matrice kovarijanse parametara. Parametri u $C(z^{-1})$ nemaju osobinu konvergencije na intervalima u kojima se vrši identifikacija. To je zbog toga što smetnja sadržana u $T_C(k)$ koja se reflektuje na izlaz $C_A(k)$ nije dobro modelirana ARMAX strukturom modela. Odnos između $T_C(k)$ i $C_A(k)$ je određen prenosnom funkcijom:

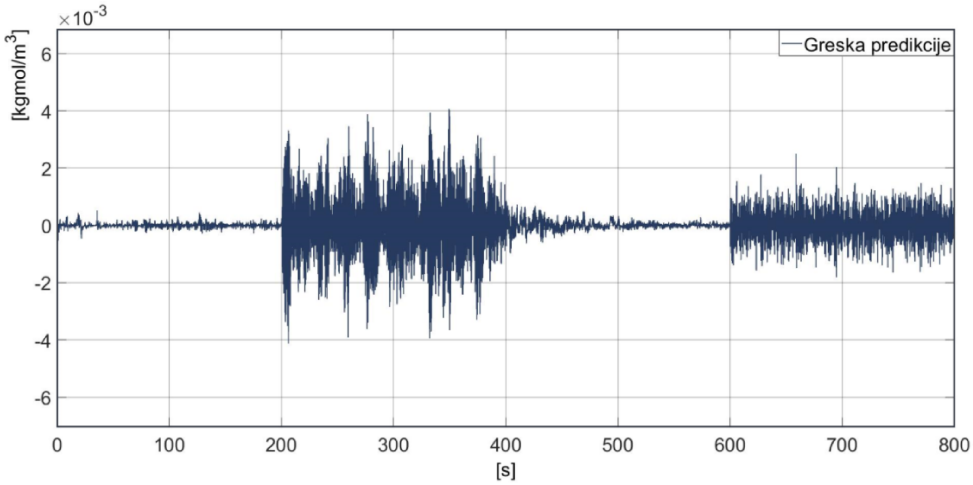
$$\frac{B(z^{-1})}{A(z^{-1})}$$

pa $C(z^{-1})$ i nije od interesa za problem identifikacije i upravljanja u ovom primjeru.

Estimirani parametri se drže konstantnim u $t \in [200,\ 400)$. Ta ovo vrijeme blok rekurzivnog estimatora nije omogučen. U trenutku $t = 400$ objekat upravljanja prelazi u novu radnu tačku i blok estimatora je ponovo pokrenut, da bi se estimirali parametri za tu radnu tačku i na osnovu pomenutog mehanizma se adaptirao PI regulator. Parametri u $A(z^{-1})$ i $B(z^{-1})$ u inervalu identifikacije $t \in [400,\ 600)$ sporije konvergiraju, što je posljedica manjih svojstvenih vrijednosti matrice kovarijanse parametara u odnosu na svojstvene vrijednosti inicijalne matrice kovarijanse. Brža konvergencija parametara u ovom vremenskom razmaku bi se mogla postići manipulacijom matrice kovarijanse.

4.2.1. Validacija modela

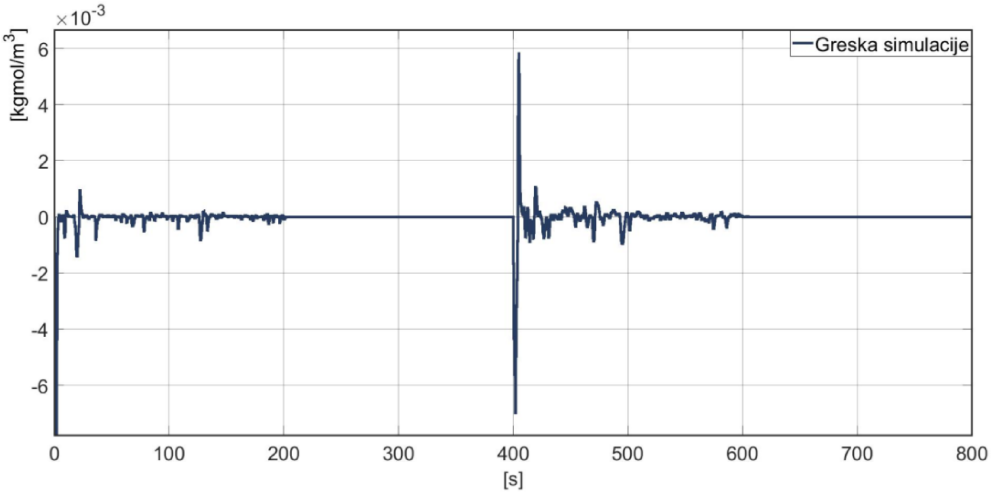
Na osnovu vremenskog dijagrama sa slike 4.4. vidimo da je greška predikcije modela većeg iznosa u intervalima vremena bez dodatnog testnog signala injektiranog u $T_C(k)$.



Slika 4.4. *Greška predikcije modela za 1 korak naprijed*

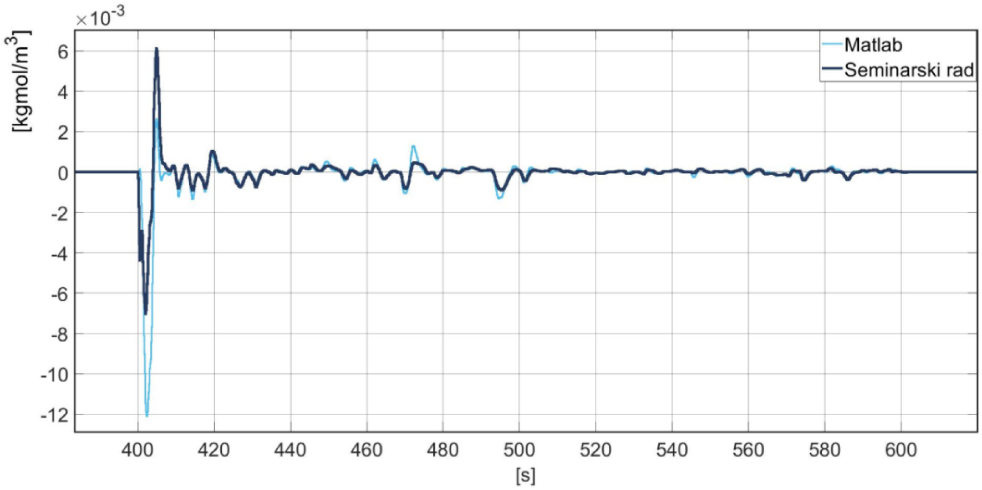
Povećanje greške može biti uzrokovano nedostatkom informacija o $T_C(k)$ koje estimator koristi. Ova greška je ograničena i malog iznosa u odnosu na fluktuacije u $C_A(k)$, što čini estimirane vrijednosti parametara pouzdanim.

Strožija validacija modela provodi se analizom greške modela, koju dobijemo simulacijom modela dobijenog identifikacijom i poređenjem izlaza ovog modela sa izlazom procesa kojeg identificiramo. Poređenje ova dva izlaza je mjerodavno za validaciju zbog povoljnog SNR-a, u protivnom trebalo bi porediti kroskorelacione funkcije $R_{uy}(\tau)$ modela i procesa. Za izabranu strukturu modela, model je implementiran (korištenjem standardnih blokova Smulink-a) u subsistemu 'time-varying ARMAX'. Na slici 4.5. je predstavljen vremenski dijagram greške modela.

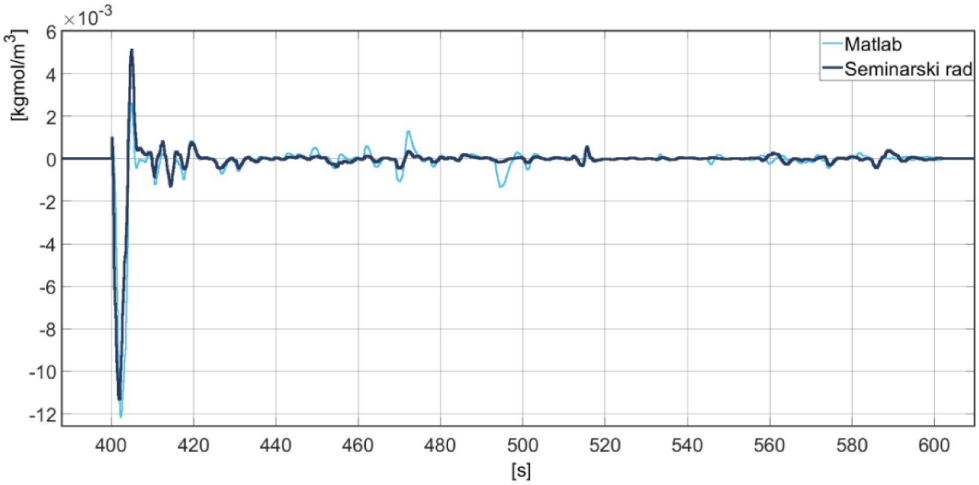


Slika 4.5. *Greška simulacije (greška modela) - razlika između izlaza procesa i izlaza modela sa estimiranim parametrima*

Vidimo da je greška modela ograničena i male vrijednosti u poređenju sa fluktuacijama u $C_A(k)$. Ovim je potvrđena validnost modela, i model je u mogučnosti da predvidi nelinearno ponašanje objekta upravljanja. Poređenja radi, na slikama 4.6. i 4.7. su prikazani vremenski dijagrami greški modela za matlabov blok 'Recursive Polynomial Model Estimator 'sa eksponencijalnim faktorom i za estimator implementiran u radu. Za algoritam sa vremenskim okvirom izabran u bloku rekurzivnog estimatora kreiranog u radu, dobiju se približno iste vrijednosti parametara A_1, A_2, B_1, B_2 kao za algoritam sa eksponencijalnim faktorom, dok vrijednosti parametara C_1, C_2 se razlikuju. Slika 4.7. prikazuje vremenski dijagram greške simulacije za matlabov rekurzivni LS estimator sa eksponencijalnim faktorom i za rekurzivni LS algoritam sa vremenskim okvirom.



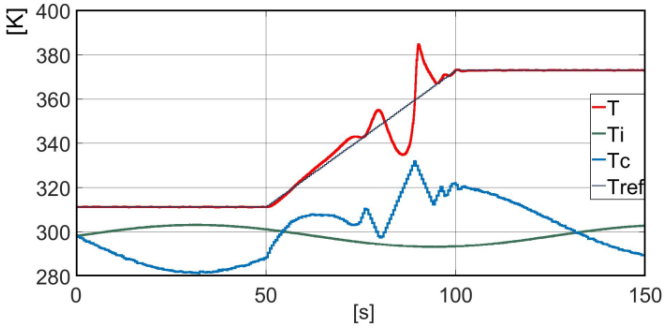
Slika 4.6. *Poređenje greški modela za blok matlabovog rekurzivnog estimatora i estimatora implementiranog u ovom radu u kojima je korišten rekurzivni LS algoritam sa eksponencijalnim faktorom iščezavanja za* $\lambda = 1 - 5 \cdot 10^{-3}$. $t \in [400,\ 600]$



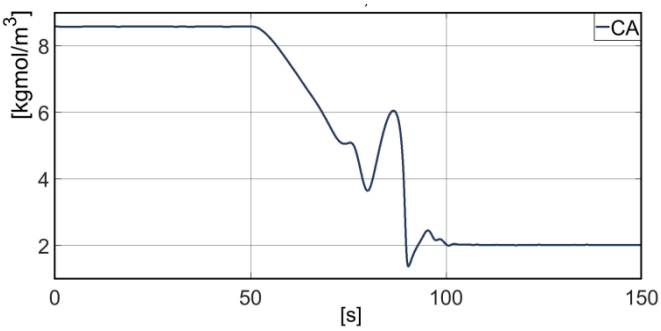
Slika 4.7. *Poređenje greški modela za blok matlabovog rekurzivnog estimatora (za* $\lambda = 1 - 5 \cdot 10^{-3}$*) i estimatora implementiranog u ovom radu. Za estimator iz rada korišten je rekurzivni LS algoritam sa vremenskim okvirom, gdje je uzeto N=1000.* $t \in [400,\ 600]$

4.3. *Adaptivno MPC upravljanje sa real-time identifikacijom objekta upravljanja*

U ovom primjeru sustema upravljanja cilj je održavati temperaturu reaktora T na željenoj referentnoj vrijednosti. Referentna vrijednost se mijenja u vremenu za vrijeme prelaza reaktora iz jednog režima rada u drugi (mali iznos konverzije reaktanta A u veći iznos konverzije). Manipulativna varijabla je temparatura vode hlađenja T_C. MPC (Model Predictive Control) kontroler prati referentnu vrijednost, tako što otklanja smetnju uzrokovanu fluktuacijama u temperaturi T_i reaktanta na ulazu u reaktor. Usvojeno je da je koncentracija C_{Ai} reaktanta na ulazu u reaktor konstantna.
U svakom trenutku uzimanja uzoraka MPC algoritam optimizira buduće ponašanje sistema, u odnosu na željeni kriterij performanse, odgovarajućim podešavanjem upravljačkih varijabli na odabranom horizontu. Ako bi za upravljanje ovim procesom implementirali osnovno MPC upravljanje (sa podešavanjima koja su zadana M-fajlom u ovom naslovu), tada bi imali rezultate koje prikazuju sljedeći vremenski dijagrami.



Slika 4.8a. *Vremenski dijagram temperature reaktora (T) za MPC upavljenje bez real-time identifikacije*



Slika 4.8b. *Vrememski dijagram koncentracije reaktanta A u reaktoru*

Sa slika 4.8a. (varijabla T) i slike 4.8b. vidimo da ako ovo upravljaje primijenimo, za širi radni dijapazon ne može se postići zadovoljavajući kvalitet upravljanja.
Za upravljanje nelinearnim procesom MPC tehnikom postoji više mogućih opcija. Obzirom da linearni model procesa može biti identifikovan on-line, u ovom primjeru koristimo adaptivni MPC sa real-time identifikacijom za postizanje zadovoljavajućeg kvaliteta upravljanja nelinearnim procesom.
Za impementaciju adaptivnog MPC upravljanja, potrebno je prvo podesiti MPC kontroler na početnoj radnoj tački gdje je C_{Ai}=10 [kgmol/m^3], T_i=298.15 [K], T_C=298.15 [K].

Pritiskom na dugme `Design MPC Controller` u simulaciji čiji je Simulink blok dijagram predstavljen na slici 4.9, pokreće se izvršavanje M-fajla:

```
%Specifikacija radne tacke:
plant_mdl = 'mpc_cstr_plant';
op = operspec(plant_mdl);


%Pocetna vrijednost koncentracije reaktanta na ulazu u reaktor  :
op.Inputs(1).u = 10;
op.Inputs(1).Known = true;
```

```
%Pocetna vrijednost temperature reaktanta na ulazu u reaktor  :
op.Inputs(2).u = 298.15;
op.Inputs(2).Known = true;


%Pocetna vrijednost temperature rashladne tecnosti:
op.Inputs(3).u = 298.15;
op.Inputs(3).Known = true;


%Racunanje pocetnih vrijednosti:
[op_point, op_report] = findop(plant_mdl,op);


%Nominalne vrijednosti za stanja x, izlaze y i ulaze u:
x0 = [op_report.States(1).x;op_report.States(2).x];
y0 = [op_report.Outputs(1).y;op_report.Outputs(2).y];
u0 = [op_report.Inputs(1).u;op_report.Inputs(2).u;op_report.Inputs(3).u];


%Linearizacija modela objekta upravljanja u tacki pocetnih vrijednosti:
sys = linearize(plant_mdl, op_point);


%CAi i CA se ne upotrebljavaju u MPC upravljanju pa ih treba iskljuciti:
sys = sys(1,2:3);


%MPC kontroler prihvata samo vremenski diskretne modele, pa model treba
diskretizovati:
Ts = 0.5;
plant = c2d(sys,Ts);


%Podesavanje kontolera
%MPC podesavamo u pocetnoj radnoj tacki. Kada je MPC pokrenut u adaptivnom
modu, model procesa se azurira u toku rada.
%Specifikacija tipova signala upotrijebljenih u MPC-u:
plant.InputGroup.MeasuredDisturbances = 1;
plant.InputGroup.ManipulatedVariables = 2;
plant.OutputGroup.Measured = 1;
plant.InputName = {'Ti','Tc'};
plant.OutputName = {'T'};


%MPC kontroler podesavamo sa difaltnim horizontom predikcije  i upravljanja
(PredictionHorizon = 10, %ControlHorizon=2):
mpcobj = mpc(plant);


%Postaviti  nominalne vrijednosti kontrolera:
mpcobj.Model.Nominal = struct('X',x0,'U',u0(2:3),'Y',y0(1),'DX',[0 0]);
```

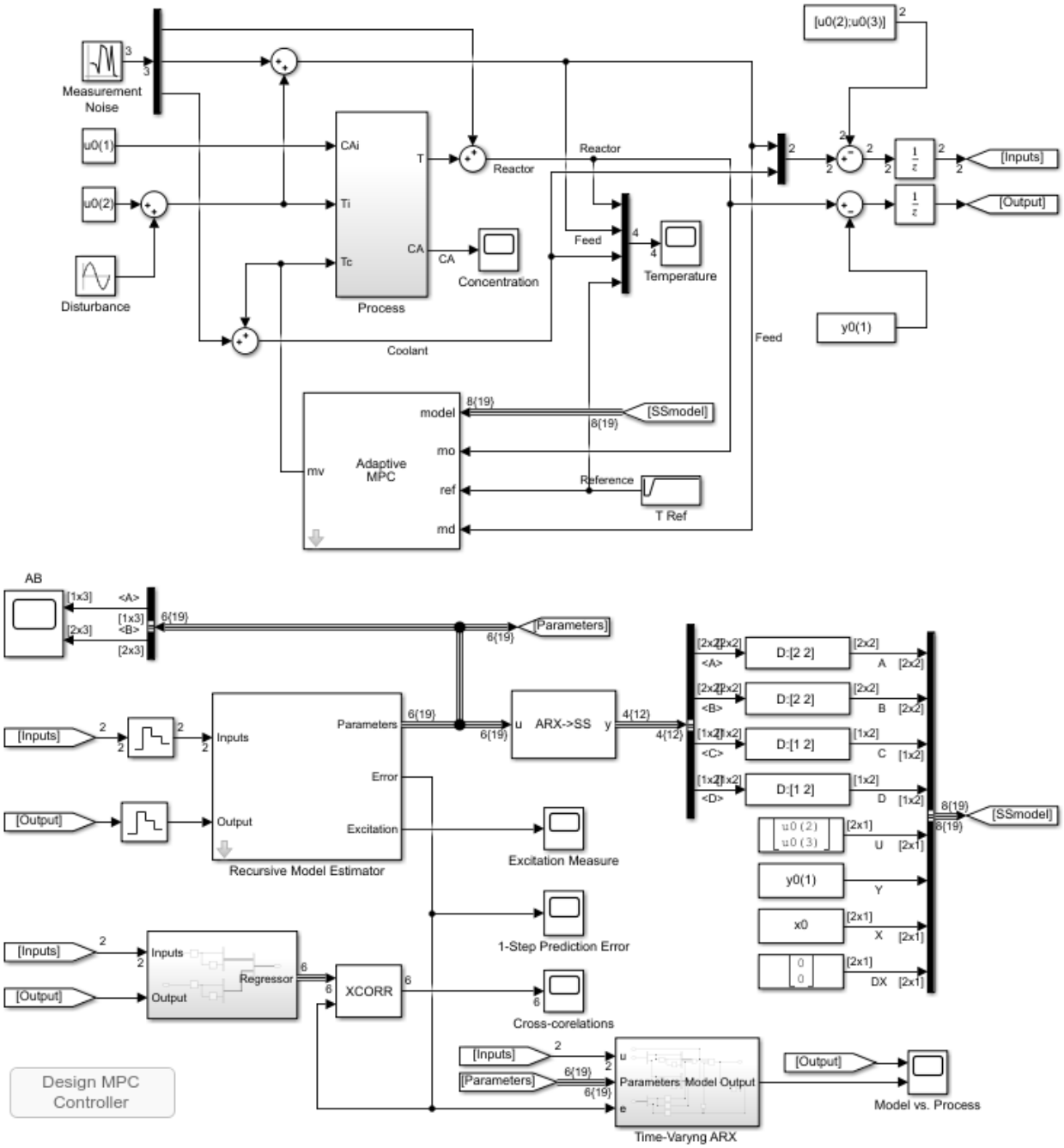
```
%Ulaz i izlaz objekta imaju razlicite redove velicina, pa treba postaviti
faktore skaliranja:
Uscale = [30 50];
Yscale = 50;
mpcobj.DV.ScaleFactor = Uscale(1);
mpcobj.MV.ScaleFactor = Uscale(2);
mpcobj.OV.ScaleFactor = Yscale;


%Zbog fizikalnih ogranicenja sistema hlađenja ,prirastaj vremenske promjene
rashladne tecnosti  TC  je ogranicena na 2 stepena u minuti:
mpcobj.MV.RateMin = -2;
mpcobj.MV.RateMax = 2;
```

U ovom primjeru se koncentracijom C_A reaktanta A u reaktoru ne upravlja direktno. Ako je ostvareno zadovoljavajuće upravljanje temperaturom T reaktora, zbog funkcionalne zavisnosti između temperature T i koncentracije C_A, upravljanje koncentracijom će imati željene performanse.
Za real-time estimaciju parametara pretpostavljenog modela ARX strukture koristimo blok rekurzivnog estimatora 'Recursive Model Estimator' .

U postavkama bloka je izabrano:

- Struktura modela: ARX
- Broj parametara A(q) (na): 2
- U identifikaciji procesa koristimo mjerene vrijednosti dva ulaza T_i i T_c , pa će je izabran broj parametara B(q) (nb): [2 2]. Za B-parametre kao rezultat estimacije imat ćemo (2×3) matricu, gdje su elementi prve kolone jednaki nuli.
- Kašnjenje ulaza (nd): [0 0]. Za mjerene ulaze procesa T_i i T_c možemo zadati različite vrijednosti kašnjejna.
- Matrica kovarijanse parametara: 10
- Metoda estimacije parametara: Kalmanov filter
- Kovarijansa procesnog šuma: 1. Ovo su vrijednosti dijagonalnih elemenata ove matrice koja je dijagonalna. On-line estimacija je osjetljiva na vrijednost ovog parametra koji se može naknadno podesiti zbog boljih rezultata estimacije. Njeni elementi se biraju kao veći od nule u slučaju estimacije vremenski promjenjivih parametara. Za brzo promjenjive parametre biraju se elementi sa velikim pozitivnim vrijednostima, što rezultira i povečanjem kovarianse estimacije parametara.
- Greška estimacije: koristi se za validaciju identifikacijom dobijenog modela.

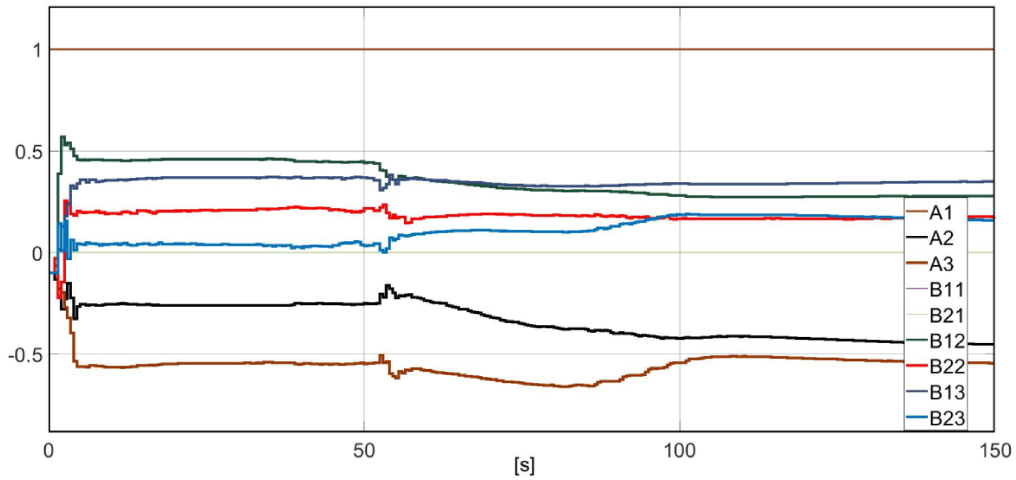
U svakom trenutku odabiranja ovaj blok obezbjeđuje estimirane vrijednosti parametara predpostavljenog modela i grešku $e(k)$ koja predstavlja grešku predikcije za jedan korak naprijed. Za izabranu ARX strukturu modela u ovom primjeru, izlaz bloka estimatora parametara modela sadrži kojeficijente polinoma $A(z^{-1}),\ \boldsymbol{B}(z^{-1})$.



Slika 4.9. *Simulink blok-dijagram sistema upravljanja sa MPC kontrolerom i real-time identifikacijom procesa, gdje je primjenjena direktna metoda identifikacije u zatvorenoj upravljačkoj konturi*

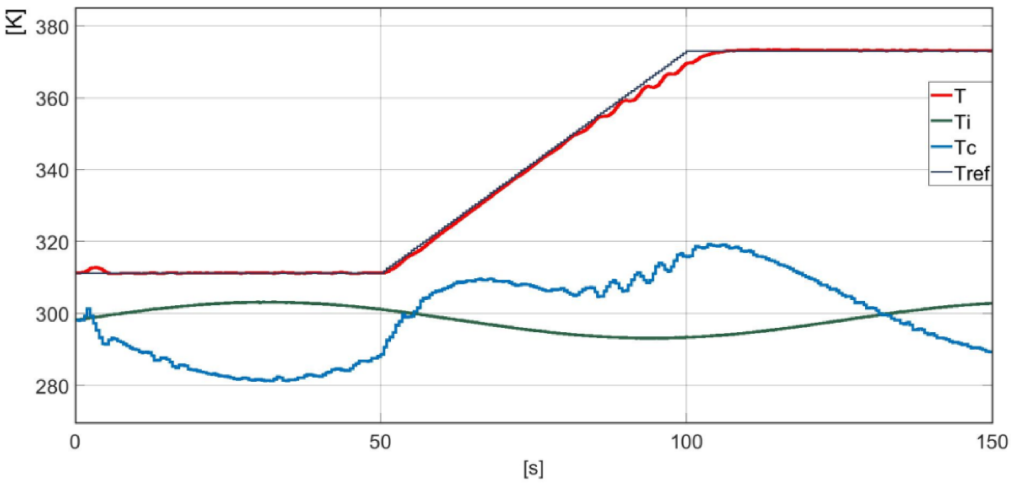
Blok adaptivnog MPC kontrolera prihvata samo vremenski diskretne modele prostora stanja, pa je korišten blok koji vrši potrebnu transformaciju ARX modela. U svakom trenutku odabiranja ovaj blok obezbjeđuje estimirane vrijednosti parametara predpostavljenog modela i grešku $e(k)$ koja predstavlja grešku predikcije za jedan korak naprijed. Za izabranu ARX strukturu modela u ovom primjeru, izlaz

bloka estimatora parametara modela sadrži kojeficijente polinoma $A(z^{-1})$, $\boldsymbol{B}(z^{-1})$, čiji su vremenski dijagrami dati na slici 4.10. Kako identificiramo sitem sa dva ulaza, to prema izabranim postavkama implementiranog bloka estimatora, B parametri čine (2×3) matricu.

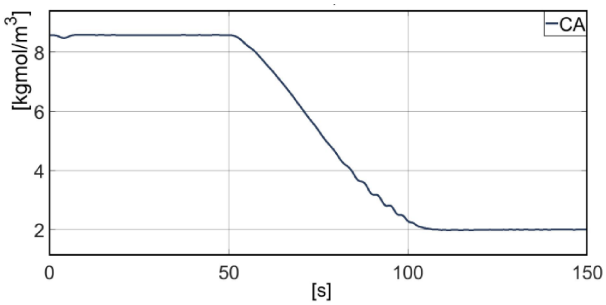


Slika 4.10. *Estimirani parametri za T_S=0.5 s*

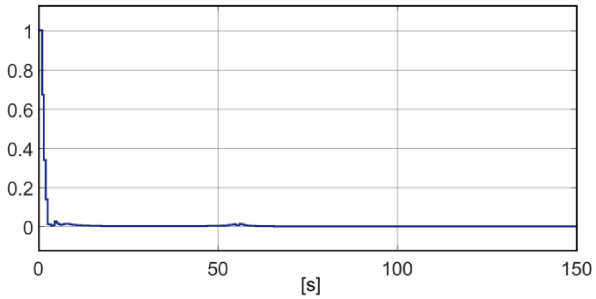
Posmatrajući sliku 4.11, na osnovu vremenskog dijagrama temperature reaktora (T) vidimo da ova upravljačka shema daje zadovoljavajuće rezultate u odnosu na osnovno MPC upravljanje istim procesom čiji su rezultati dati na slikama 4.8a. i 4.8b.



Slika 4.11. *Vremenski dijagrami: zadane vrijednosti temperature (T_{ref}), temperature reaktanta na ulazu u reaktor (T_i), temperatura rashladne tečnosti, tj. vrijednost manipulativne varijable (T_C), temperatura reaktora (T)*



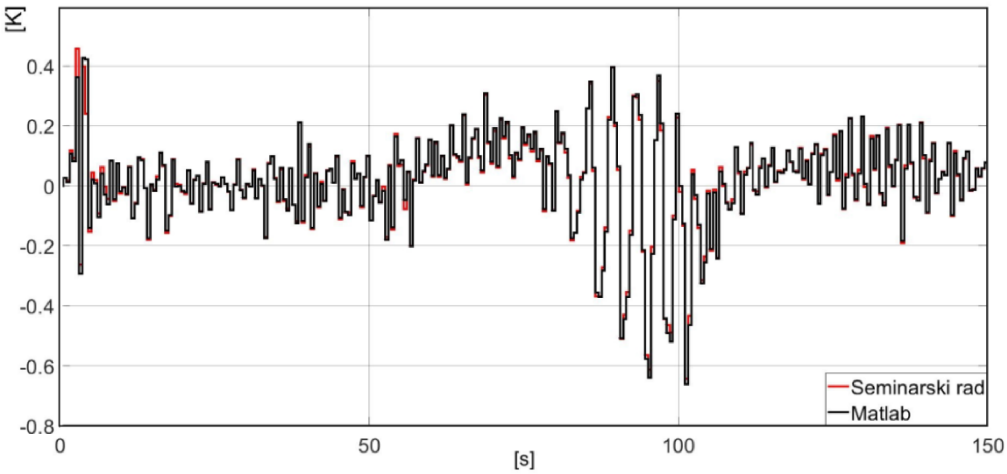
Slika 4.12. *Koncentracija reaktanta A u reaktoru*



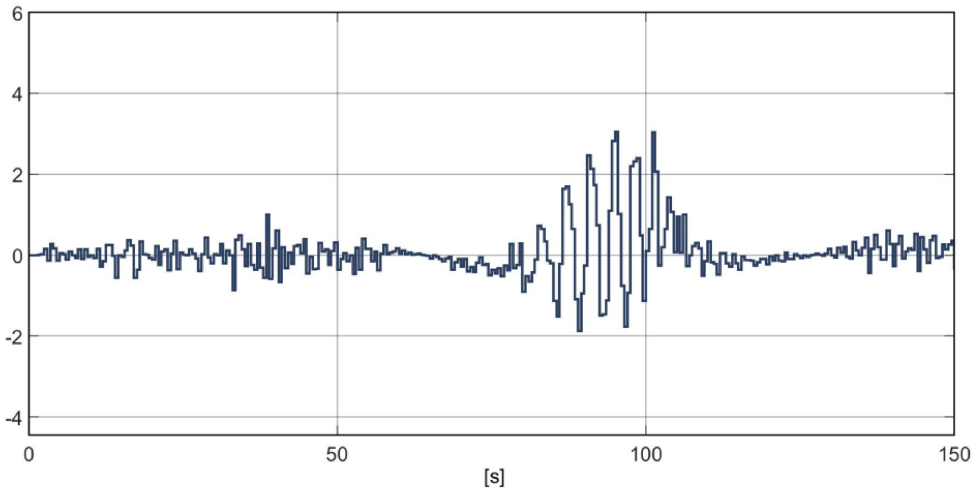
Slika 4.13. *Vremenski dijagram mjere pobude, određene relacijom (2.90b)*

4.3.1. *Validacija modela*

Prema relaciji (2.91b) i na osnovu slike 4.13 vidimo da je sistem nakon tranzijentnog režima dovoljno pobuđen. Na osnovu slike 4.14 vidimo da je greška predikcije ograničena. Na slici 4.15 je predstavljen vremenski dijagram kros-korelacione funkcije signala greške i prve komponente regresora. Kros-korelaciona funkcija signala greške i komponenti regresora, na razmaku [70, 105], ima izrazito veću vrijednost nego inaće, što se odražava na vrijednosti greške predikcije u tom intervalu, pa je u tom intervalu diskutabilna validnost modela.

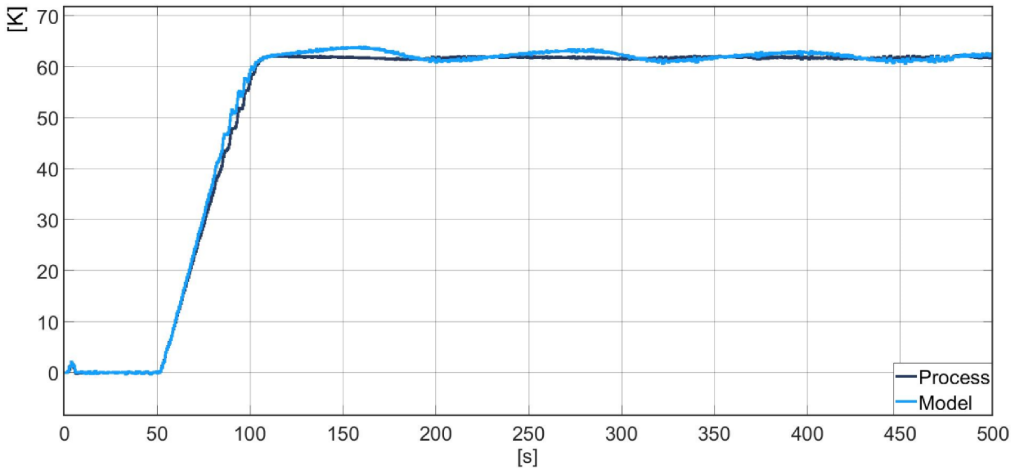


Slika 4.14. *Greška predikcije ARX modela procesa za blok Recursive Model Estimator implementiran u radu i matlabov blok Reursive Polynomial Model Estimator*



Slika 4.15. *Kros-korelaciona funkcija jednog od elemenata regresora i signala greške*

Poređenja radi, na slici 4.14. je dat vremenski dijagram i greške predikcije za matlabov blok 'Recursive Polynomial Model Estimator' , primijenjenog u sistemu upravljanja sa slike 4.9. U ovom primjeru imamo povoljan SNR, pa se u cilju validacije mogu uporediti izlaz procesa i izlaz modela, koji su predstavljeni na slici 4.16.



Slika 4.16. Vremenski dijagram izlaza procesa, tj. temperature reaktora (T) i izlaza njegovog ARX modela procesa sa vremenski promjenjivim parametrima

5. Implementacije

U ovom dijelu rada je naveden sadržaj pojedinih fajlova koji sadrže izvorni kod koji definiše funkcionalnost S-funkcije, koja u simulacijama navedenim u naslovu 3. predstavlja blok rekurzivne estimacije parametara. Ova S-funkcija na Simulink blok dijagramima na slikama 4.2 i 4.9 nazvana Recursive Model Estimator. U S-funkciji su impementirane metode navedene u 2.2.1, 2.2.2, 2.2.3 i u dodatku A. Implementirani rekurzivni LS algoritam sa ekponencijalnim faktorom i Kalmanov filter ima mogučnost on-line estimacije parametara ARX, ARMAX, BJ i OE modela. Rekurzivni LS algoritam sa vremenskim okvirom i BFGS algoritam imaju mogučnost on-line estimacije paramerara ARX, ARMAX i OE modela. Za S-funkciju koja predstavlja blok rekurzivne estimacije parametara, je kreirana maska koja daje mogučnost preglednog izbora strukture modela, metode estimacije parametara i njihovih parametara, kao i izbor dodatnih ulaza (omogučavajući ulaz) i izlaza (greška predikcije i mjera pobude). Sadržaj fajla `PSAU_SeminarskiRad.cpp` je uglavnom generiran automatski upotrebom alata S-function Builder, i zbog obima nije naveden. Funkcije koje predstavljaju operacije nad matricama su implementirane u `matrixlibrary.cpp`. BFGS algoritam je implementiran u `bfgslibrary.cpp`.

5.1. PSAU_SeminarskiRad_wrapper.cpp

```
1.    #include "simstruc.h"
2.    #include " PSAU_SeminarskiRad _bus.h"
3.    #include <math.h>
4.    #include <vector>
5.    #include <stdlib.h>
6.    #include "matrixlibrary.h"
7.    #include "matrixlibrary.cpp"
8.    #include "elslibrary.h"
9.    #include "elslibrary.cpp"
10.   #include "bfgsalgorithm.h"
11.   #include "bfgsalgorithm.cpp"

 /*  ========================================================================  *
  *  U ovoj funkciji vrsi se dinamicka alokacija i inicijalizacija           *
  *  potrebnog memorijskog prostora .                                        *
  *  ========================================================================  */

12.   void ExtendedLeastSquares1_Start_wrapper(void **pW,
13.               const int32_T *nb, const int_T nu,
14.               SimStruct *S) {

15.   int DIM;
16.   int snb=0;
17.   for (int i=0; i<nu; i++){
18.       snb += nb[i];
19.   }
20.   if (*model==1){                    /* ARX */
21.      DIM = *na + snb;
```

```
22.   }
23.   else if (*model==2){                /* ARMAX */
24.      DIM = *na + snb + *nc;
25.   }
26.   else if (*model==3){                /* BJ */
27.      DIM = snb + *nf + *nc + *nd;
28.   }
29.   else if (*model==4){               /* OE */
30.       DIM = snb + *nf;
31.   }

      /* Alokacija radnog vektora regresora. */
32.   double * psi = allocate_array(DIM);
33.   pW[0] = psi;

      /* Alokacija shift registra za pohranu (kasnjenje) izmjerenih vrijednosti
        (vise) ulaza u sistem kojeg identificiramo. */
34.   double ** inputs_delays = allocate_rough_matrix(nu, d);
35.   pW[1] = inputs_delays;

      /* Alokacija radnog vektora (shift registra) izlaznog signala iz sistema
         kojeg identificiramo. */
36.   pW[2] = allocate_array(2);

      /* Alokacija matrice kovarijanse P. */
37.   double ** p = allocate_matrix(DIM, DIM);

      /*Inicijalizacija matrice P prema (2.17).*/
38.   for (int i=0; i<DIM; i++){
39.       p[i][i] = *alpha * *alpha;
40.   }
41.   pW[3] = p;

       /* Alokacija vektora  parametara sistema. */
42.   double * theta = allocate_array(DIM);
43.   pW[4] = theta;

     /* Inicijalizacija vektora parametara vrijednostima bliskim nuli. */
44.   if (*method==3){
45.   for (int i=0; i<DIM; i++){
46.       theta[i] = -0.1;
47.   }
48.   }
      /* Alokacija prostora za pohranu vrijednosti residuala eps_hat prethodne
         iteracije. */
49.   pW[5] = allocate_array(1);

      /* Alokacija prostora za pohranu vrijednosti residuala w_hat prethodne
         iteracije. */
50.   pW[6] = allocate_array(1);

      /* Alokacija prostora za pohranu vrijednosti residuala v_hat prethodne
         iteracije. */
51.   pW[7] = allocate_array(1);
```

```
    /* U slucaju izbora rekurzivnog LS algoritma sa prozorom ili BFGS alg.,
       alocira se matrica u koju se pohranjuju regresori od k-N+1 do k-tog
       (sadasnjeg) trenutka odabiranja. Ova matrica se inicijalizuje prema
       (2.40a) i (2.40b). Takodje se alocira i vektor za pohranu prethodnih
       vrijednosti mjerenih izlaza procesa. */
52. if ((*method==1 && *history==2 && (*model==1 || *model==2 || *model==4))
    ||
53.     (*method==2 && (*model==1 || *model==2 || *model==3 || *model==4)))
    {
54.     double ** regressor_mat = allocate_matrix(*win_length, DIM);
55.     for (int i=0; i<DIM; i++){
56.         regressor_mat[i][DIM-1-i] = 1 / (*alpha);
57.     }
58.     pW[8] = regressor_mat;
59.     double * y0_vec = allocate_array(*win_length);
60.     pW[9] = y0_vec;
61. }

 /* ========================================================================  *
  * U ovoj fukciji su implementirani algoritmi iz naslova 2.2.1, 2.2.2,  *
  * 2.2.3.  , primjenjen BFGS algoritam iz dodatka A, koji je  *
  * implementiran u biblioteci 'bfgsalgorithm.cpp' i ovdje je pozvan kao  *
  * funkcija.                                                             *
  * ========================================================================  */

62.  void ExtendedLeastSquares1_Outputs_wrapper(const real_T *u0,
63.              const real_T *y0,
64.              const real_T *Enable,
65.              PSAU_BUS *Params,
66.              real_T *Error,
67.              real_T *Excitation,
68.              void **pW,
69.              const int32_T *nb, const int_T nu,
70.              SimStruct *S) {
71.  double y;
72.  double fade, pn_cov;
73.  int    DIM;
74.  double u[nu];
75.  int snb=0;
76.  for (int i=0; i<nu; i++){
77.      snb += nb[i];
78.  }
79.  if (*model==1){                  /* ARX */
80.     DIM = *na + snb;
81.  }
82.  else if (*model==2){             /* ARMAX */
83.     DIM = *na + snb + *nc;
84.  }
85.  else if (*model==3){              /* BJ */
86.     DIM = snb + *nf + *nc + *nd;
87.  }
88.  else if (*model==4){              /* OE */
89.      DIM = snb + *nf;
90.  }
91.  struct Variable psi_k1;
92.  struct Variable residual_k1 = initialize_vector_column(1);
93.  struct Variable theta_hat_k = initialize_vector_column(DIM);
```

```
94.   struct Variable theta_hat_k1;
95.   struct Variable P_k         = initialize_matrix(DIM, DIM);
96.   struct Variable P_k1;
97.   struct Variable Pk_psik1, psik1tr_Pk_psik1;
98.   struct Variable gamma_k;
99.   struct Variable psi_kN;
100.  double y0kN;

      /* Deklaracije varijabli za prethodno alocirane vektore i matrice. */
101.  double *  psi        = (double *) pW[0];
102.  double ** u_buf      = (double **)pW[1];
103.  double *  y_buf      = (double *) pW[2];
104.  double ** P_buf      = (double **)pW[3];
105.  double *  theta_buf  = (double *) pW[4];
106.  double *  eps_hat    = (double *) pW[5];
107.  double *  w_hat      = (double *) pW[6];
108.  double *  v_hat      = (double *) pW[7];

      /* Pohrana vrijednosti i-tog ulaza procesa u grbavu matricu, cija i-ta
         kolona   ima  d[i]  elemenata,  sa  ciljem  kasnjenja  i-tog  ulaza  za
         vrijednost d[i]. */
109.  for (int i=0; i<nu; i++){
110.       u[i]       = delay(u_buf[i], u0[i], d[i]+1);
111.  }
      /* Kasnjenje izlaza procesa za jedan odabirak. */
112.   y       = delay(y_buf, *y0, 1);

      /* Punjenje regresora definisanog sa (2.60) preko pokazivaca'psi'. w_hat,
         eps_hat, v_hat su odredjeni sa (2.57). */
113.   psi_k1  = regressor_grls( psi, u, y, *w_hat, *eps_hat, *v_hat, *na, nb,
      nu, *nf, *nc, *nd, *model);

114.  if (*method==1){
115.  if (*history==2 && (*model==1 || *model==2 || *model==4)){
116.      double ** regressor_matrix  = (double **)pW[8];
117.      double *  y0_vector         = (double *) pW[9];
118.      psi_kN                = regressor_flow( regressor_matrix, psi_k1,
      *win_length);
119.      y0kN                  = y0_flow( y0_vector, *y0, *win_length);
120.  }
121.  }
      /* Uzimanje pohranjenog vektora parametara. */
122.  for (int i=0; i<DIM; i++){
123.      theta_hat_k.X[i][0] = theta_buf[i];
124.  }
125.  if (*method==1 || *method==3){
126.  for (int i=0; i<DIM; i++){
127.      for (int j=0; j<DIM; j++){
128.          P_k.X[i][j] = P_buf[i][j];
129.      }
130.  }
131.  *history==2 ? fade = 1       : fade = *lambda;  /* history=2-s prozorom */
132.  *method==3  ? pn_cov = *pnc : pn_cov = 0;       /* process noise covar. */

      /* Implementacija relacija(2.34). */
133.  struct Variable psi_k1_tr = transpose(psi_k1);
134.  residual_k1.X[0][0] = *y0 - inner(psi_k1, theta_hat_k, 0, DIM);
```

```
135.  Pk_psik1             = multiply(P_k, psi_k1);
136.  psik1tr_Pk_psik1     = multiply(psi_k1_tr, Pk_psik1);
137.  gamma_k              = c( Pk_psik1, (1/(fade +
      psik1tr_Pk_psik1.X[0][0])));
138.  *Enable>0 ?
139.      theta_hat_k1     = add(theta_hat_k, multiply(gamma_k, residual_k1)) :
      theta_hat_k1 = theta_hat_k;

      /* Juryev test i algoritam projekcije dat u naslovu 2.3. */
140.  while ( jury_test( theta_hat_k1, *na, nb, nu, *nf, *nc, *nd, *model)==0){
141.      gamma_k          = c( gamma_k, 0.5);
142.      theta_hat_k1     = add( theta_hat_k, multiply(gamma_k, residual_k1));
143.  }
      /* Elementi regresora prema (2.57). */
144.  if (*model==1){
145.      *w_hat           = 0;
146.      *eps_hat         = 0;
147.      *v_hat           = 0;
148.  } else if (*model==2){
149.      *w_hat           = 0;
150.      *eps_hat         = *y0 - inner( psi_k1, theta_hat_k1, 0, DIM);
151.      *v_hat           = 0;
152.  } else if (*model==3){
153.      *w_hat           =  inner( psi_k1, theta_hat_k1, 0, snb) + inner(
      psi_k1, theta_hat_k1, snb, snb+*nf);
154.      *v_hat           = -*w_hat;
155.      *eps_hat         =  *v_hat - inner(psi_k1, theta_hat_k1, snb+*nf+*nc,
      snb+*nf+*nc+*nd) - inner(psi_k1, theta_hat_k1, snb+*nf, snb+*nf+*nc);
156.  } else if (*model==4){
157.      *w_hat           = inner(psi_k1, theta_hat_k1, 0, snb+*nf);
158.      *eps_hat         = 0;
159.      *v_hat           = 0;
160.  }
161.  P_k1                 = c( subtract( P_k, multiply( gamma_k, multiply(
      psi_k1_tr, P_k) ) ), (1/ fade));
162.  P_k                  = add(P_k1, c(eye(DIM), pn_cov));
163.  theta_hat_k          = theta_hat_k1;

      /* Racunanje greske predikcije prema (2.10b) i mjere pobude prema (2.90b)
         za ERLS i KF algoritme. */
164.  *Error = residual_k1.X[0][0];
165.  *Excitation = 1 / (fade + psik1tr_Pk_psik1.X[0][0]);

166.  if (*history==2 && (*model==1 || *model==2 || *model==4)){
167.      *method==3  ? pn_cov = *pnc : pn_cov = 0;
168.      struct Variable Pk_psikN, psikNtr_Pk_psikN;
169.      struct Variable psi_kN_tr  = transpose(psi_kN);

      /* Implementacija relacija (2.37). */
170.      residual_k1.X[0][0]        = y0kN - multiply(psi_kN_tr,
          theta_hat_k).X[0][0];
171.      Pk_psikN                   = multiply(P_k, psi_kN);
172.      psikNtr_Pk_psikN           = multiply(psi_kN_tr, Pk_psikN);
173.      gamma_k                    = c( Pk_psikN, (1/(1 -
          psikNtr_Pk_psikN.X[0][0])));
174.      *Enable>0 ?
175.          theta_hat_k1           = subtract(theta_hat_k, multiply(gamma_k,
```

```
        residual_k1)) : theta_hat_k1=theta_hat_k;
176.    P_k1                        = add( P_k, multiply( gamma_k, multiply(
        psi_kN_tr, P_k) ) );
177.    P_k                         = add(P_k1, c(eye(DIM), pn_cov));
178.    theta_hat_k                 = theta_hat_k1;
179. }
    /* Pohrana matrice kovarijanse. */
180.  for (int i=0; i<DIM; i++){
181.       for (int j=0;j<DIM; j++){
182.            P_buf[i][j] = P_k.X[i][j];
183.       }
184.  }
185. /* BFGS. */
186. } else if (*method==2){
187.    struct Variable regressor_win, y0_win;
188.    double ** regressor_matrix  = (double **)pW[8];
189.    double *  y0_vector         = (double *) pW[9];
190.
191.    regressor_win   = regressor_window( regressor_matrix, psi_k1,
        *win_length);
192.    y0_win          = y0_window( y0_vector, y0[0], *win_length);
193.    *Enable>0 ?
194.    theta_hat_k1    = BFGS( criterion, regressor_win, y0_win,
        theta_hat_k, *tolerance) : theta_hat_k1 = theta_hat_k;
195.    if (*model==1){
196.    *w_hat          = 0;
197.    *eps_hat        = 0;
198.    *v_hat          = 0;
199. } else if (*model==2){
200.    *w_hat          = 0;
201.    *eps_hat        = *y0 - inner( psi_k1, theta_hat_k1, 0, DIM);
202.    *v_hat          = 0;
203. } else if (*model==3){
204.    *w_hat          =  inner( psi_k1, theta_hat_k1, 0, snb) + inner(
        psi_k1, theta_hat_k1, snb, snb+*nf);
205.    *v_hat          = -*w_hat;
206.    *eps_hat        =  *v_hat - inner(psi_k1, theta_hat_k1, snb+*nf+*nc,
        snb+*nf+*nc+*nd) - inner(psi_k1, theta_hat_k1, snb+*nf, snb+*nf+*nc);
207. } else if (*model==4){
208.    *w_hat          = inner(psi_k1, theta_hat_k1, 0, snb+*nf);
209.    *eps_hat        = 0;
210.    *v_hat          = 0;
211. }
212.    theta_hat_k     = theta_hat_k1;

213.    /* Racunanje greske predikcije modela za BGFS algoritam. */
214.    struct Variable ss  = subtract(y0_win, multiply(regressor_win,
        theta_hat_k));
215.    double dd = 0;
216.    for (int i=0; i<regressor_win.X.size(); i++){
217.         dd += ss.X[i][0];
218.    }
219.    *Error   = dd;
220. }
221.  if (*Enable>0){
222.      for (int i=0; i<DIM; i++){
223.           theta_buf[i] = theta_hat_k.X[i][0];
```

```
224.       }
225.   }

 /*  ========================================================================  *
  *  Za proces sa vise ulaza B parametri su struktuirani u matricu. Izlaz  *
  *  bloka S-funkcije implementirane u ovom radu je sabirnica PSAU_BUS,  *
  *  koju cine A, B, F, C, D estimirani parametri. B_params funkcija vrsi  *
  *  struktuiranje B estimiranih parametara u matricu B.                  *
  *  ========================================================================  */

226.   if (*model==1){
227.       Params->A[0] = 1;
228.       for (int i=0; i<*na; i++){
229.          Params->A[i+1] = theta_buf[i];
230.       }
231.       B_params( Params, theta_buf, nb, nu, *na);
232.   } else if (*model==2){
233.       Params->A[0] = 1;
234.       for (int i=0; i<*na; i++){
235.          Params->A[i+1] = theta_buf[i];
236.       }
237.       B_params( Params, theta_buf, nb, nu, *na);
238.       Params->C[0] = 1;
239.       for (int i=*na+snb; i<*na+snb+*nc; i++){
240.          Params->C[i+1-*na-snb] = theta_buf[i];
241.       }
242.   } else if (*model==3){
243.       B_params( Params, theta_buf, nb, nu, 0);
244.       Params->F[0] = 1;
245.       for (int i=snb; i<snb+*nf; i++){
246.          Params->F[i+1-snb] = theta_buf[i];
247.       }
248.       Params->C[0] = 1;
249.       for (int i=snb+*nf; i<snb+*nf+*nc; i++){
250.          Params->C[i+1-snb-*nf] = theta_buf[i];
251.       }
252.       Params->D[0] = 1;
253.       for (int i=snb+*nf+*nc; i<snb+*nf+*nc+*nd; i++){
254.          Params->D[i+1-snb-*nf-*nc] = theta_buf[i];
255.       }
256.   } else if (*model==4){
257.       B_params( Params, theta_buf, nb, nu, 0);
258.       Params->F[0] = 1;
259.       for (int i=snb; i<snb+*nf; i++){
260.          Params->F[i+1-snb] = theta_buf[i];
261.       }
262.   }
263.   Params->Ts = *sample_time;
264.  }

 /*  ========================================================================  *
  *  U  ovoj  funkciji  vrsi  se  oslobadjanje  memorijskog  prostora.  *
  *  ========================================================================  */

265.  void ExtendedLeastSquares1_Terminate_wrapper(void **pW,
266.              SimStruct *S) {
267.
```

```
268.    for(int i=0; i<10; i++){
269.        free(pW[i]);
270.    }
271.  }
272.
```

4.2. Biblioteka ‘elslibrary.cpp’

```
1.   #include "elslibrary.h"

 /*  ========================================================================  *
  *  Ova funkcija   vrsi punjenje regresora definisanog prema (2.60),  *
  *  dinamicki alociranog, sa pokazivacem ‘psi’.   Parametri funkcije su:  *
  *  u - matrica u ciji redovi odredjuju kasnjenja pojedinih mjerenih  *
  *  ulaza MISO procesa; y - mjereni izlaz procesa; w_hat, eps_hat,  *
  *  v_hat – rezidali za opci linearni model (2.51) odredjeni relacijama  *
  *  (2.57);na, *nb, nf, nc, nd, nu – brojevi odredjeni prema (2.51);  *
  *  model – struktura modela:1=ARX, 2=ARMAX, 3=BJ, 4=OE. Izlaz funkcije  *
  *  je    vektor    kolina              deklarisan    kao    struktura.  *
  *  ========================================================================  */

2.   struct Variable regressor_grls (double * psi, double * u, double y, double
     w_hat, double eps_hat, double v_hat, int na, const int * nb, int nu, int
     nf, int nc, int nd, int model){
3.       struct Variable out;
4.       int snb = 0;
5.       for (int i=0; i<nu; i++) {snb += nb[i]; }
6.       switch (model){
7.           case 1:
8.               out = initialize_vector_column(na+snb);
9.               for (int i=na-1; i>=1; i--){
10.                   psi[i] = psi[i-1];
11.              }
12.              snb = 0;
13.              for (int j=0; j<nu; j++){
14.                  for (int i=nb[j]-1; i>=1; i--){
15.                      psi[na+snb+i] = psi[na+snb+i-1];
16.                      snb += nb[j];
17.                  }
18.              }
19.              psi[0]            = -y;
20.              snb = 0;
21.              for (int i=0; i<nu; i++){
22.                   psi[na+snb]  =  u[i];
23.                   snb += nb[i];
24.               }
25.              for (int i=0; i<na+snb; i++){
26.                   out.X[i][0] =  psi[i];
27.              }
28.          break;
29.          case 2:
30.              out = initialize_vector_column(na+snb+nc);
31.              for (int i=na-1; i>=1; i--){
32.                   psi[i] = psi[i-1];
```

```
33.                 }
34.                 snb = 0;
35.                 for (int j=0; j<nu; j++){
36.                     for (int i=nb[j]-1; i>=1; i--){
37.                         psi[na+snb+i] = psi[na+snb+i-1];
38.                         snb += nb[j];
39.                     }
40.                 }
41.                 for (int i=nc-1; i>=1; i--){
42.                     psi[na+snb+i] = psi[na+snb+i-1];
43.                 }
44.                 psi[0]     = -y;
45.                 snb = 0;
46.                 for (int i=0; i<nu; i++){
47.                     psi[na+snb]  =  u[i];
48.                     snb += nb[i];
49.                  }
50.                 psi[na+snb] =  eps_hat;
51.                 for (int i=0; i<na+snb+nc; i++){
52.                     out.X[i][0] = psi[i];
53.                 }
54.             break;
55.             case 3:
56.                 out = initialize_vector_column(snb+nf+nc+nd);
57.                 snb = 0;
58.                 for (int j=0; j<nu; j++){
59.                     for (int i=nb[j]-1; i>=1; i--){
60.                         psi[snb+i] = psi[snb+i-1];
61.                         snb += nb[j];
62.                     }
63.                 }
64.                 for (int i=nf-1; i>=1; i--){
65.                     psi[snb+i] = psi[snb+i-1];
66.                 }
67.                 for (int i=nc-1; i>=1; i--){
68.                     psi[snb+nf+i] = psi[snb+nf+i-1];
69.                 }
70.                 for (int i=nd-1; i>=1; i--){
71.                     psi[snb+nf+nc+i] = psi[snb+nf+nc+i-1];
72.                 }
73.                 snb = 0;
74.                 for (int i=0; i<nu; i++){
75.                     psi[snb]  =  u[i];
76.                     snb += nb[i];
77.                  }
78.                 psi[snb]       = -w_hat;
79.                 psi[snb+nf]    =  eps_hat;
80.                 psi[snb+nf+nc] = -v_hat;
81.                 for (int i=0; i<snb+nf+nc+nd; i++){
82.                     out.X[i][0] = psi[i];
83.                 }
84.             break;
85.             case 4:
86.                 out = initialize_vector_column(snb+nf);
87.                 snb = 0;
88.                 for (int j=0; j<nu; j++){
89.                     for (int i=nb[j]-1; i>=1; i--){
```

```
90.                     psi[snb+i] = psi[snb+i-1];
91.                     snb += nb[j];
92.                 }
93.             }
94.             for (int i=nf-1; i>=1; i--){
95.                  psi[snb+i] = psi[snb+i-1];
96.             }
97.             snb = 0;
98.             for (int i=0; i<nu; i++){
99.                  psi[snb]  =  u[i];
100.                 snb += nb[i];
101.            }
102.            psi[snb]       = -w_hat;
103.            for (int i=0; i<snb+nf; i++){
104.                 out.X[i][0] = psi[i];
105.            }
106.        break;
107.    }
108.    return out;
109. }
```

```
/*  ========================================================================  *
 *  Ova funkcija pohranjuje regresore 'psi_k1' iz prethodnih od k-N+1 do  *
 *  k trenutaka odabiranja u matricu sa pokazivacem 'regressor_matrix'.  *
 *  Sa obzirom na regresore, ova  matrica cini FIFO listu, cime je njen  *
 *  izlaz za N odabiraka zakasnjeli  regresor. Ova funkcija se koristi u  *
 *  rekurzivnom         LS         algoritmu         sa         prozorom.  *
 *  ========================================================================  */
```

```
110. struct Variable regressor_flow (double ** regressor_matrix, struct
     Variable psi_k1, int N) {
111. struct Variable out = initialize_vector_column(psi_k1.X.size());
112.     for (int i=N-1; i>=1; i--){
113.         for (int j=0; j<psi_k1.X.size(); j++){
114.             regressor_matrix[i][j] = regressor_matrix[i-1][j];
115.          }
116.     }
117.     for (int j=0; j<psi_k1.X.size(); j++){
118.         regressor_matrix[0][j] = psi_k1.X[j][0];
119.     }
120.     for (int j=0; j<psi_k1.X.size(); j++){
121.          out.X[j][0] = regressor_matrix[N-1][j];
122.     }
123.     return out;
124.  }
```

```
/*  ========================================================================  *
 *  Ova funkcija implementira shift registar sa pokazivacem y0, koji vrsi  *
 *  kasnjenje signala y za N odabiraka.                                     *
 *  ========================================================================  */
```

```
125. double y0_flow (double * y0, double y, int N){
126.     double out;
127.     for (int i=N-1; i>=1; i--){
128.         y0[i] = y0[i-1];
129.     }
130.     y0[0] = y;
```

```
131.      out   = y0[N-1];
132.      return out;
133.   }

 /*  ========================================================================  *
  *  Za proces sa vise ulaza B parametri su struktuirani u matricu. Izlaz  *
  *  bloka S-funkcije implementirane u ovom radu je sabirnica PSAU_BUS,  *
  *  koju cine A, B, F, C, D estimirani parametri. Ova funkcija vrsi  *
  *  struktuiranje B estimiranih parametara u matricu B.                  *
  *  ========================================================================  */

134.  void B_params (PSAU_BUS * Params, double * theta_buf, const int *nb, int
      nu, int na){
135.      for (int i=0; i<nu; i++) Params->B[i] = 0;
136.      int k = nu;
137.       int j=0;
138.      for ( ; ; ){
139.          int snb = 0;
140.          for (int i=0; i<nu; i++){
141.              if (j<nb[i]){
142.                  Params->B[k] = theta_buf[na + j + snb];
143.                  k++;
144.              } else {
145.                  Params->B[k] = 0;
146.                  k++;
147.              }
148.              snb += nb[i];
149.          }
150.          j++;
151.          int mx = nb[0];
152.          for (int l=0; l<nu; l++){
153.              if (nb[l]>mx) mx = nb[l];
154.          }
155.          if (j >= mx) break;
156.      }
157.  }

 /*  ========================================================================  *
  *  U ovoj funkciji je implementiran Juryjev test stabinosti polinoma po  *
  *  'z'. Ulaz ove funkcije je vektor parametara. Izmedju lb i ub uzimaju  *
  *  se elementi vektora koji su kojeficijenti polinoma cija se stabilnost  *
  *  ispituje. Ako je polinom stabilan izlaz funkcije je 1, inace izlaz  *
  *  funkcije                              je                              0.  */
     ========================================================================

158.  int jury (struct Variable theta, int lb, int ub){
159.      int i, j;
160.      vector <vector<double> > vvd;
161.      vector <double> v (ub-lb+1);
162.      v[0] = 1;
163.      for (i=lb; i<ub; i++){
164.          v[i-lb+1] = theta.X[i][0];
165.      }
166.      vvd.push_back(v);
167.      reverze(&v);
168.      vvd.push_back(v);
169.      for (i=2;;i+=2){
```

```
170.          v.clear();
171.          double mult = vvd[i-2][vvd[i-2].size()-1]/vvd[i-2][0];
172.          for (j=0; j<vvd[i-2].size()-1; j++){
173.              v.push_back(vvd[i-2][j] - vvd[i-1][j] * mult);
174.          }
175.          vvd.push_back(v);
176.          reverze(&v);
177.          vvd.push_back(v);
178.          if (v.size() == 1) break;
179.      }
180.      for (i=0; i<vvd.size(); i+=2){
181.              if (vvd[i][0]<=0) break;
182.      }
183.      if (i == vvd.size())
184.          return 1;
185.      else
186.          return 0;
187.   }

 /*  ========================================================================  *
  *  Ova funkcija  testira stabilnost specificiranog modela ciji je vektor  *
  *  parametara 'theta'.                                                     *
  *  ========================================================================  */

188.  int jury_test (struct Variable theta, int na, const int *nb, int nu, int
      nf, int nc, int nd, int model){
189.      int out;
190.      int snb = 0;
191.      for (int i=0; i<nu; i++) {snb += nb[i]; }
192.      if (model==1){
193.          out = 1;
194.      } else if (model==2){
195.          out = jury( theta, na+snb, na+snb+nc);
196.       } else if (model==3){
197.          out = (jury( theta, snb, snb+nf) &&
198.                 jury( theta, snb+nf, snb+nf+nc) &&
199.                 jury( theta, snb+nf+nc, snb+nf+nc+nd));
200.       } else if (model==4){
201.          out = jury( theta, snb, snb+nf);
202.       }
203.       return out;
204.    }
     /* Alokacija vektora sa n elemenata. */

205.  double * allocate_array (int n){
206.      double * pok = (double *) calloc(n, sizeof(double));
207.      if (pok==NULL) return NULL;
208.      return pok;
209.  }

210.  void clear_matrix (double ** matrix, int m){
211.      if(matrix==NULL) return;
212.      for (int i=0; i<m; i++)
213.          free(matrix[i]);
214.          free(matrix);
215.  }
```

```
    /* Alokacija matrice m vrsta i n kolona. */

216. double ** allocate_matrix (int m, int n){
217.     double ** matrix = (double** )calloc(m, sizeof(double *));
218.     if (matrix==NULL) return NULL;
219.     for (int i=0; i<m; i++){
220.         matrix[i] = (double * )calloc(n, sizeof(double));
221.         if (matrix[i]==NULL){
222.            clear_matrix(matrix,i);
223.            return NULL;
224.         }
225.     }
226.     return matrix;
227. }
     /* Alokacija matrice sa m vrsta i razlicitim brojem kolona. */

228. double ** allocate_rough_matrix (int m, const int *n){
229.     double ** matrix = (double** )calloc(m, sizeof(double *));
230.     if (matrix==NULL) return NULL;
231.     for (int i=0; i<m; i++){
232.         matrix[i] = (double * )calloc(n[i]+2, sizeof(double));
233.         if (matrix[i]==NULL){
234.            clear_matrix(matrix,i);
235.            return NULL;
236.         }
237.     }
238.     return matrix;
239. }
     /* Kasnjenje velicine 'u' za 'd' odabiraka, implementirano kao shift
        registar. */

240. double delay (double * array, double u, int d){
241.     double out;
242.     out = array[d];
243.     array[0] = u;
244.     for (int i=d; i>=1; i--){
245.                   array[i] = array[i-1];
246.     }
247.     return out;
248. }

 /*  ========================================================================  *
  *     Izlaz iz ove funkcije je matrica sa N vrsta i brojem kolona  *
  *     jednakom dimenziji vektora parametara (2.60), cije vste cine  *
  *     regresori od k-N+1 do k-tog (sadasnjeg) trenutka odabiranja.  *
  *  ========================================================================  */

249. struct Variable regressor_window (double ** regressor_matrix, struct
     Variable psi_k1, int N) {
250.     struct Variable out = initialize_matrix(N, psi_k1.X.size());
251.     for (int i=N-1; i>=1; i--){
252.         for (int j=0; j<psi_k1.X.size(); j++){
253.             regressor_matrix[i][j] = regressor_matrix[i-1][j];
254.          }
255.     }
256.     for (int j=0; j<psi_k1.X.size(); j++){
```

```
257.         regressor_matrix[0][j] = psi_k1.X[j][0];
258.     }
259.     for (int i=0; i<N; i++){
260.         for (int j=0; j<psi_k1.X.size(); j++){
261.              out.X[i][j] = regressor_matrix[i][j];
262.         }
263.     }
264.     return out;
265.  }

 /*  ========================================================================  *
  *  Izlaz ove funkcije jeste vektor ciji su elementi jednaki izmjerenim  *
  *  izlazima procesa od trenutka k-N+1 do k-tog (tekuceg) trenutka  *
  *  iteracije.                                                            *
  *  ========================================================================  */

266. struct Variable y0_window (double * y0, double y, int N){
267.     struct Variable out = initialize_vector_column(N);
268.     for (int i=N-1; i>=1; i--){
269.         y0[i] = y0[i-1];
270.     }
271.     y0[0] = y;
272.     for (int i=0; i<N; i++){
273.         out.X[i][0] = y0[i];
274.     }
275.     return out;
276.  }

 /*  ========================================================================  *
  *  Funkcija kriterija za BFGS algoritam. odredjena relacijom (A.9).  *
  *  ========================================================================  */

277.  double criterion (struct Variable regressor_mat, struct Variable y0,
    struct Variable theta){
278.     struct Variable s = subtract(y0, multiply(regressor_mat, theta));
279.     double d=0;
280.     for (int i=0; i<regressor_mat.X.size(); i++){
281.         d += pow(s.X[i][0], 2);
282.     }
283.     return d;
284. }
```

Dodatak A: Primjena BFGS algoritma u on-line estimaciji parametara

Metode rekurzivne estimacije parametara koje su navedene u ovom radu su primjenjive za identifikaciju sistema kod kojih je greška predikcije linearna po parametrima. Za slučaj kada ovaj uslov nije ispunjen, što je slučaj za nelinearne modele, s obzirom da problem nije moguće riješiti u zatvorenoj formi kao kod LS metoda, mogu se koristiti metode iterativne optimizacije, od kojih je BFGS (Broyden-Fletcher-Goldfarb-Shanno) metod jedan od najefikasnijih. Mada su znatno numerički zahtjevniji u odnosu na klasične LS metode, ove optimizacijske metode je moguće koristiti i kada je ispunjen uslov linearnosti greške predikcije po parametrima.

U općem slučaju, u rješavanju problema parametarske identifikacije optimizacijskim metodama, on se posmatra kao problem pronalaženja takve vrijednosti vektora parametara $\boldsymbol{\theta}*$ za kojeg vrijedi:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} f\left(\boldsymbol{\theta},\ e\right) \\ s.t. \quad & \boldsymbol{g}(\boldsymbol{\theta}) \le \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{\theta}) = \boldsymbol{0} \end{aligned} \tag{A.1}$$

gdje ograničenja određena skupom nejednakosti i jednakosti definišu problemski prostor, tj. prostor parametara koji određuju model čija je struktura unaprijed pretpostavljena. Za linearne modele to su ARX, ARMAX, BJ, OE i slične strukture. Prvo tipično ograničenje je da model mora biti stabilan. Drugo ograničenje, koje je intuitivno, je da neki parametri, čija je fizikalnost poznata, ne smiju biti negativni ili moraju pripadati nekom intervalu.

Funkciju kriterija f figuriše greška, koja može biti definisana na više načina, zavisno od postavke modela. Tipične su : greška po ulazu, greška po izlazu i generalizirana greška. Kao funkciju kriterija može se odabrati proizvoljna parna funkcija greške i obično se bira kao pozitivno definitna otežana kvadratna forma.

Princip funkcionisanja većine iterativnih algoritama optimizacije, jeste propagacija potencijalnog rješenja $\boldsymbol{\theta}(i)$ na sljedeći način:

$$\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) + \alpha\, \boldsymbol{p}(i) \tag{A.2}$$

gdje je $\boldsymbol{p}(i)$ vektor koji određuje pravac jednodimenzionalnog pretraživanja, $\boldsymbol{\theta}(0)$ početna tačka. Iteracija se zaustavlja kada je ispunjen uslov zaustavljanja. Kao uslov zaustavljanja se obično bira stagnacija funkcije kriterija, a mjera stagnacije je norma gradijenta te funkcije. Prema tome algoritam se zaustavlja kada norma gradijenta padne ispod dovoljno male vrijednosti $\varepsilon > 0$, tj. kada je:

$$\left\|\nabla f\right\| < \varepsilon \tag{A.3}$$

i ako je ispunjen uslov unimodalnosti funkcije kriterija algoritam je sa određenom tačnošću pronašao optimum.

Za Newtonov algoritam u jednačini jed je: $\alpha(i) = \left(\nabla^2 f(\boldsymbol{\theta}(i))\right)^{-1}$, $\boldsymbol{p}(i) = -\nabla f\left(\boldsymbol{\theta}(i)\right)$ (A.4)

gdje je $\nabla^2 f(\boldsymbol{x})$ Hessijan funkcije f . Ovaj algoritam ima kvadratnu konvergenciju, ali njegov nedostatak je računanje drugih izvoda i računanje inverznog Hessiana.
BFGS pripada klasi kvazi-Newtonovih algoritama, za kojeg je a pravac pretraživanja određen sa:

$$\boldsymbol{p}(i) = -\boldsymbol{H}(i)\nabla f\left(\boldsymbol{\theta}(i)\right) \tag{A.5}$$

Matrica $\boldsymbol{H}(i)$ je aproksimacija inverznog Hessiana, i inicijalizira se jediničnom matricom. Ovaj algoritam ima superlinearnu konvergenciju, ali je numerički efikasniji od Newtonovog jer je izbjegnuto računanje drugih izvoda i inverzne matrice. Ako obilježimo:

$$\boldsymbol{s}_i = \boldsymbol{\theta}(i+1) - \boldsymbol{\theta}(i) \tag{A.6a}$$
$$\boldsymbol{y}_i = \nabla f\left(\boldsymbol{\theta}(i+1)\right) - \nabla f\left(\boldsymbol{\theta}(i)\right) \tag{A.6b}$$

po BFGS metodu matrica $\boldsymbol{H}(i)$ se ažurira prema relaciji:

$$\boldsymbol{H}_{i+1} = \boldsymbol{H}_i + \frac{\left(\boldsymbol{s}_i^T\boldsymbol{y}_i + \boldsymbol{y}_i^T\boldsymbol{H}_i\boldsymbol{y}_i\right)\left(\boldsymbol{s}_i\boldsymbol{s}_i^T\right)}{\left(\boldsymbol{s}_i^T\boldsymbol{y}_i\right)^2} - \frac{\boldsymbol{H}_i\boldsymbol{y}_i\boldsymbol{s}_i^T + \boldsymbol{s}_i\boldsymbol{y}_i^T\boldsymbol{H}_i}{\boldsymbol{s}_i^T\boldsymbol{y}_i} \tag{A.7}$$

gdje zbog preglednosti koristimo indeksnu notaciju. Optimalnu dužinu koraka $\alpha > 0$ određujemo nekom od metoda jednodimenzionalnog pretraživanja po pravcu $\boldsymbol{p}(k)$,tj.:

$$\arg\min_{\alpha>0} f\left(\boldsymbol{\theta}(i) + \alpha\,\boldsymbol{p}(i)\right) \tag{A.8}$$

Dužina koraka $\alpha > 0$ određuje se nekom od metoda jednodimenzionalne optimizacije. Ova veličina ne mora biti nužno optimalna, pa se u svrhu njihovog približnog nalaženja koriste i drugi algoritmi jednodimenzionalnog pretraživanja zasnovani na Wolfe-ovim uslovima.
Niz kojeg generira BFGS algoritam konvergira superlinearno, i sljedeća teorema, čiji dokaz je izložen u [5], daje dovoljne uslove konvergencije:

Teorema A1*: Ako su zadovoljeni sljedeći uslovi:*

(i) *Kriterijalna funkcija f je dvostruko neprekidno diferencijabilna.*

(ii) *Skup $\mathcal{L}=\left\{\boldsymbol{x}\in\mathbb{R}^n \mid f\left(\boldsymbol{x}\right)\le f\left(\boldsymbol{x}^0\right)\right\}$ gdje je $\boldsymbol{x}^0$ početna tačka, je konveksan i postoje pozitivne konstante m i M tako da vrijedi*

$$m\|\boldsymbol{z}\|^2 \le \boldsymbol{z}^T\nabla^2 f\left(\boldsymbol{x}\right)\boldsymbol{z} \le M\|\boldsymbol{z}\|^2 \text{ , } \forall \boldsymbol{z}\in\mathbb{R}^n, \quad \boldsymbol{x}\in\mathcal{L} \tag{2.70}$$

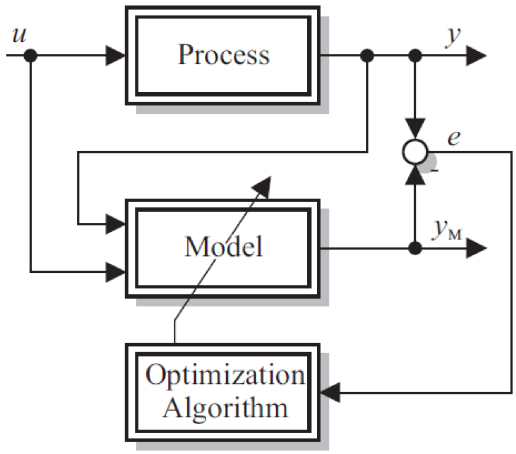
(iii) *$\boldsymbol{H}_0$ simetrična i pozitivno definitna početna matrica.*

Tada niz $\left\{\boldsymbol{x}^k\right\}$ kojeg generira BFGS algoritam konvergira tački minimuma $\boldsymbol{x}^*$ kriterijalne funkcije f .

U ovom radu BFGS algoritam je implementiran, tako da vrši on-line estimaciju parametara linearnog MISO modela sa ARX, ARMAX, i OE strukturom. Za funkciju kriterija je uzeta funkcija:

$$f(\boldsymbol{\theta}) = \sum_{j=k-N+1}^{k} e^2(j) = \sum_{j=k-N+1}^{k} \left[y(j) - y(j \mid j-1) \right]^2 = \sum_{j=k-N+1}^{k} \left[y(j) - \boldsymbol{\varphi}^T(j)\boldsymbol{\theta}(j-1) \right]^2 \qquad \text{(A.9)}$$

gdje je k tekući trenutak u on-line estimaciji parametara.



Slika A.1. *On-line estimacija parametara modela iterativnom optimizacijom*

U funkciju kriterija ulaze izmjerene vrijednosti ulaza $u(j)$ i izmjerene vrijednosti izlaza $y(j)$ za $k-N+1 \le j \le k$, pa prema tome i u kontekstu izloženog u 2.2.2, N je širina vremenskog okvira. Kao rezultat BFGS algoritma, za tekući trenutak k dobijemo vrijednost vektora parametara $\hat{\boldsymbol{\theta}}(k)$ za kojeg funkcija kriterija ima minimalnu vrijednost. Kao početna tačka algoritma za tekući trenutak k uzet je vektor parametara za prethodni trenutak, tj. $\hat{\boldsymbol{\theta}}(k-1)$, a u početnom trenutku $k=0$ uzeto je $\boldsymbol{\theta}(0) = \boldsymbol{0}$.

.

Zaključak

U prvom primjeru simulacije adaptivnog sistema upravljanja, za izabrane opcije implementiranog bloka rekurzivnog estimatora, prema vremenskim dijagramima u naslovu 4.2.1 potvrđena je validnost modela. Na osnovu slike 4.3., parametri u $A(z^{-1})$ i $B(z^{-1})$ brzo konvergiraju ka estimiranim vrijednostima na vremenskom razmaku [0, 200). To je zbog izabrane velike vrijednosti dijagonalnih elemenata inicijalne matrice kovarijanse parametara. Parametri u $C(z^{-1})$ nemaju osobinu konvergencije na intervalima u kojima se vrši identifikacija. To je zbog toga što smetnja sadržana u $T_C(k)$ koja se reflektuje na izlaz $C_A(k)$ nije dobro modelirana izabranom strukturom modela. $C(z^{-1})$ i nije od interesa za problem identifikacije i upravljanja u ovom primjeru. U trenutku $t = 400$ objekat upravljanja prelazi u novu radnu tačku i blok estimatora je ponovo pokrenut, da bi se estimirali parametri za tu radnu tačku i na osnovu navedenog mehanizma se adaptirao PI regulator. Parametri u $A(z^{-1})$ i $B(z^{-1})$ u intervalu identifikacije $t \in [400,\ 600)$ sporije konvergiraju, što je posljedica manjih svojstvenih vrijednosti matrice kovarijanse parametara u odnosu na svojstvene vrijednosti inicijalne matrice kovarijanse. Brža konvergencija parametara u ovom vremenskom razmaku bi se mogla postići manipulacijom matrice kovarijanse.
Zbog nelinearnosti procesa, koncentracija C_A reaktanta A u reaktoru je mnogo osjetljivija na promjenu manipulativne varijable T_C pri većim koncentracijama C_A. Može se reći da se identifikacijom u ovom primjeru detektuje ta promjena osjetljivosti. Estimirani parametri su upotrijebljeni za podešavanje pojačanja PI regulatora. Cilj je izbjeći velike iznose pojačanja, zbog čega sistem može ući u nestabilnost.

Za drugi primjer sistema upravljanja po pitanju validnosti modela, na osnovu vremenskog dijagrama na slici 4.14 vidimo da je za izabrane postavke bloka estimatora, greška predikcije modela u intervalu [70, 105] većeg iznosa nego inaće. Na tom inervalu je izražena korelacija između signala greške i komponenti regresora, što prikazuje slika 4.15. Poređenjem vremenskih dijagrama na slikama 4.11 i 4.8a na kojima su prikazani i odzivi procesa na istu referentnu vrijednost u različitim upravljačkim strukturma, vidimo da sistem upravljanja sa slike 4.9 daje dobre rezultate i ima prednost.
Po pitanju greške predikcije modela u ovom primjeru, na osnovu slike 4.14 vidimo da blok estimatora parametara koji je implementiran u ovom radu daje jako bliske rezultate matlabovom rekurzivnom estimatoru parametara.

Literatura

[1] R. Isermann, M. Münchof. Identification of Dynamic Systems: An Introduction with Applications, Springer, 2011.

[2] P. Stoica, T. Söderström. System Identification, Prentice Hall International, 1989.

[3] L. Ljung, T. Söderström. Theory and Practice of Recursive Identification, The MIT Press, 1989.

[4] B. Peruničić. Analiza signala i sistema, Elektrotehnički fakultet u Sarajevu, 1972.

[5] J. Nocedal, S. J. Wright. Numerical Optimization, Springer, 2006.

[6] M. Hebibović. Teorija automatskog upravljanja, Elektrotehnički fakultet u Sarajevu, 2003.

[7] Q. Zhang. Some implementatio Aspects of Sliding Window Least Squares Algorithm, IFAC, 2000.

[8] Johnstone, C. Johnson, R. Bitmead, B. Anderson. Exponencial Convergence of Recursive Least Squares vith Exponencial Forgeting Factor, IEEE, 1982.

[9] X. Hu, L. Ljung. New Convergence Results for Least Squares Identification Algorithm, IFAC, 2008.

[10] S. Brugeman, R. Bitmead. Exponential Convergence of Recursive Least Squares Method with Forgeting Factor for Multi Outputs Systems, arXiv, 2020.

[11] U. Forsell, L. Ljung. Closed-loop Identification, Linkoping University, 1998.

[12] The Matworks. Matlab Simulink: Writing S-Functions. The Mathworks, Inc. 2007.

[13] N. Gupta, R. Hauser. Kalman filtering with Inequality State Constraints, Oxford University Computing Lab., 2007.

[14] D.E. Seborg, T.F. Edgar, D.A. Melichamp, Process Dynamics and Control, John Waley & Sons Inc., 2011.