



Proiect la Identificarea Sistemelor

Identificarea unei axe actionate cu motor BLDC

Profesor coordonator:

Prof.univ.dr.ing. Petru Dobra

Student:

Batin Razvan-Paul

Grupa 30131/1, an III

Capitolul 1

Vizualizarea datelor

În Matlab vom încărca fișierul cu datele experimentale primite ("batin.mat") în care se afla două campuri, X' (intervalul de timp în care s-a făcut achiziția de date) și Y' (unde se afla valorile intrării, $u \rightarrow$ adică factorul de umplere al PWM-ului, valorile vitezei unghiulare, ω [rad/s] și valorile poziției unghiulare, Θ [impulsuri]).

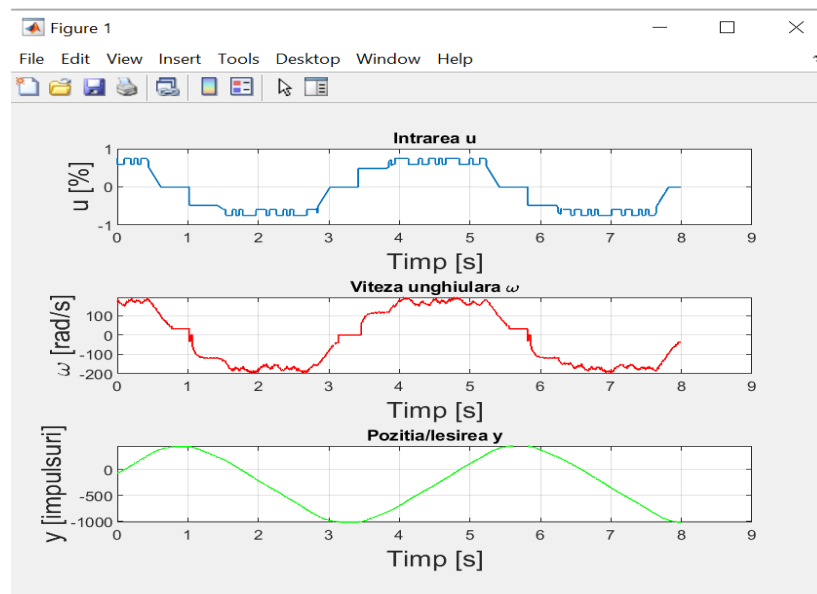


Fig.1: Datele experimentale eluate separat

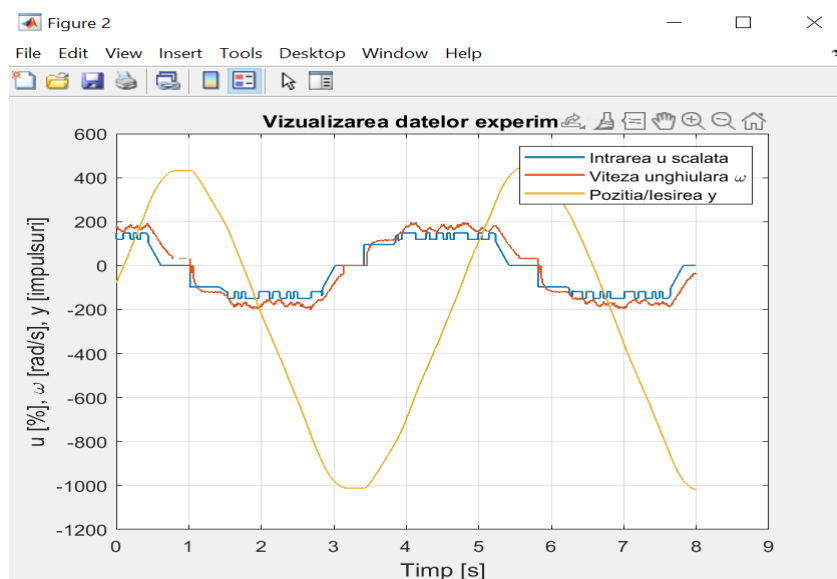


Fig.2: Datele experimentale suprapuse (u scalat cu 200 de unitati)

Capitolul 2

Partea 1 :Identificarea neparametrica a sistemului

1.1 Identificarea modelului intrare – viteza folosind metoda logaritmarilor successive

Pentru aceasta identificare a modelului matematic pentru viteza unghiulara vom selecta o parte din semnalul de intrare unde se regaseste o treapta. Aceasta secventa se observa in figura de mai jos(Fig.3).

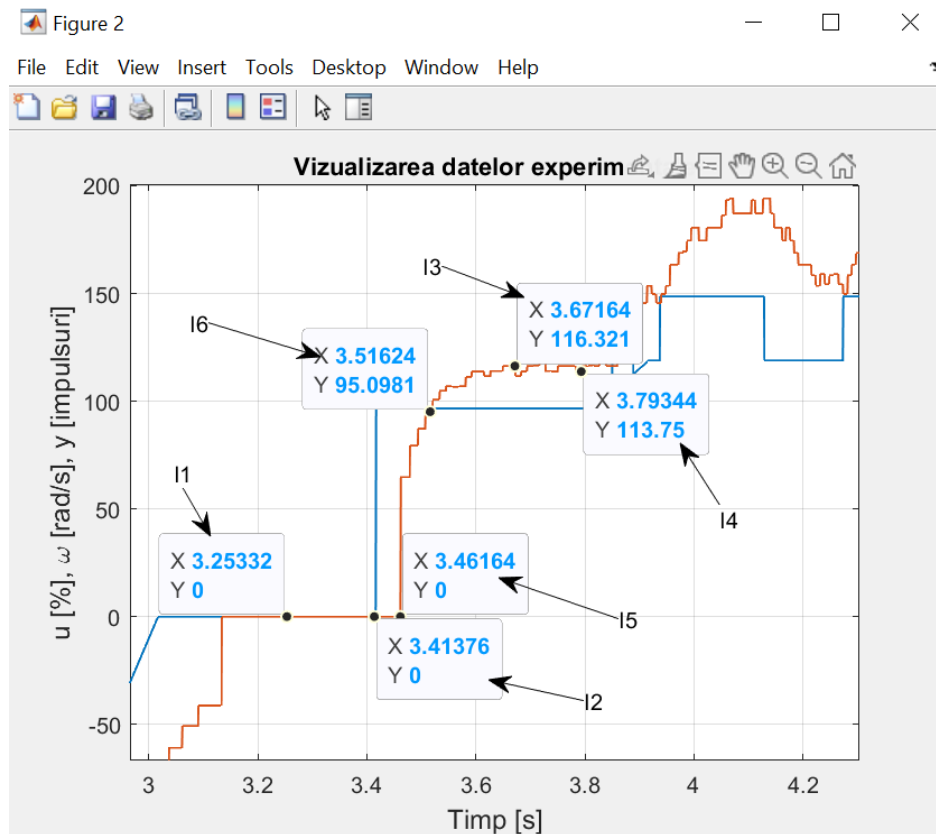


Fig.3: Portiunea de semnal unde avem intrare de tip treapta

Ne propunem sa obtinem un model matematic LTI (linear time invariant) de ordin I care este descris de o functie de transfer de tipul $H(s) = \frac{K}{Ts+1}$, K fiind factorul de proportionalitate, iar T, fiind constanta de timp mecanica, constanta care are cea mai mare influenta asupra sistemului, celelalte doua constante electrice fiind neglijabile in cazul de fata.

1.2 Calcularea factorului de proportionalitate K

Factorul de proportionalitate, K, este dat de raportul dintr viteza si intrare in regim stationar. Zgomotul de masura poate aduce dificultati in alegerea punctelor de pe semnal, deci va trebui sa le selectam cu atentie.

$$K = \frac{wst - w0}{ust - u0}$$

Pentru calculul factorului de proportionalitate ne vom folosi de formula prezentata mai sus, unde wst este o valoare medie a vitezei unghiulare in regim stationar, medie ce se face intre indecsii i3 si i4 de pe Fig.3, w0 este o valoare medie initiala a vitezei unghiulare, valoare calculate intre indecsii i1 si i2 de pe aceeasi figura. Mergem pe acelasi principiu si pentru ust si u0, care prezinta valori medii ale intrarii luate pe aceleasi intervale. Valorile gasite in Matlab pentru aceste variabile vor fi prezentate in secventa urmatoare de cod:

```
%%  
%Indecsii regasiti de pe grafic folosind cursor_info.DataIndex  
i1=3814;  
i2=3959;  
i3=4415;  
i4=4520;  
  
%Valorile medii pentru viteza si intrare  
wst=mean(w(i3:i4));  
ust=mean(u(i3:i4));  
  
w0=mean(w(i1:i2));  
u0=mean(u(i1:i2));  
  
%Factorul de proportionalitate  
K=(wst-w0)/(ust-u0) %239.72
```

In urma calculelor se observa ca factorul de proportionalitate, K are valoarea de 239.72.

1.3 Calcularea constantei de timp T, timpului mort si determinarea functiei de transfer

Vom folosi metoda logaritmilor successive pentru a calcula constanta de timp T care reprezinta dinamica dominanta a sistemului.

In practica, raspunsul indicial este dat de relatia:

$$w(t) = w_0 + (w_{st} - w_0) * (1 - e^{-\frac{t}{T}})$$

Se logaritmeaza relatia de mai sus, si se obtine dupa o eventuala scalare pentru a rezolva problema existentei logaritmului :

$$\ln(|w(tk) - y_{st}|) = c - \frac{1}{T} * tk$$

unde tk sunt esantioanele de timp, iar c este o constanta de scalare. Alegem aceste esantioane de la momentul declansarii treptei si pana inainte de regimul stationar. Ne vom folosi de indecsii i5 si i6 de pe Fig.3, care vor reprezenta intervalul pentru tk in cazul nostru.

```
%Timpul de urcare
i5=4112;
i6=4280;
tk=t(i5:i6);
```

In continuare vom presupune ca avem un α si un β , pentru care se constituie ecuatia drepte, $w = \alpha * x + \beta$. Pentru determinarea acestor parametrii vom rezolva urmatorul sistem:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^N tk^2 & \sum_{k=1}^N tk \\ \sum_{k=1}^N tk & N \end{pmatrix} \begin{pmatrix} \sum_{k=1}^N xk * tk \\ \sum_{k=1}^N xk \end{pmatrix}$$

-Unde $xk = \log(w_{st} - w(i5:i6))$, adica dreapta regresiei calculata pe intervalul definit de tk.

```
%Algoritmul Matlab pentru regresie|
xk=log(wst-w(i5:i6));
Areg=[sum(tk.^2) sum(tk);
      sum(tk) length(tk)];
breg=[sum(tk.*xk);
      sum(xk)];
sol=Areg\breg;
T=-1/sol(1);
```

-unde constanta de timp T are valoarea de 46.9 [ms].

Timpul mort se observa usor de pe grafic pe intrarea noastra de tip treapta si se calculeaza de la momentul declansarii treptei, pana cand viteza se modifica, deci sistemul nostru raspunde la aceasta intrare. Concret, pe graficul de mai sus, reprezentat de Fig.3, calculam diferenta dintre timpul la valoare lui i5 si timpul la valoare i2 pentru aflarea timpului mort.

Avand parametrii necesari calculati, putem determina fara probleme functia de transfer a modelului dat.

```
Continuous-time transfer function.

>> H = tf(K, [T 1], 'iodelay', Tm)

H =

                239.7
exp(-0.047*s) * ----
                0.0469 s + 1
```

1.4 Validarea modelului si calculul erorii medii patratic normalizata

Prin modelul spatial starilor vom simula in continuare rezultatul obtinut si ne vom pastra conditiile initiale nenule, pentru o suprapunere cat mai exacta a datelor. Din viteza simulata vom scoate eroarea medie patratica cu formula $e_{MIN_w} = \text{norm}(w - w_{sim}) / \text{norm}(w - \text{mean}(w))$. Urmeaza sa ne folosim de un artificiu de calcul prin care introducem comportarea de sistem cu timp mort in intrare. Acesti pasi sunt descrisi de urmatorul cod Matlab:

```
N = round(Tm/(t(2)-t(1)));
uN = [u(1)*ones(N,1); u(1:end-N)];
A = -1/T;
B = K/T;
C = 1;
D = 0;
figure
plot(t, u*200);
wsim = lsim(A,B,C,D,uN,t,w(1));
hold on
plot(t,[w wsim]), grid;
legend('Intrarea u scalata', 'Viteza unghiulara \omega', 'Viteza unghiulara simulata \omegasim');
xlabel('Timp [s]', 'FontSize', 12);
ylabel('u [%], \omega [rad/s], \omegasim [rad/s]', 'FontSize', 12);
%eroarea medie patratica
eMIN_w = norm(w-wsim)/norm(w-mean(w));
```

- Aflam eroare medie patratica de valoare 13.39%, iar graficul rezultat din suprapunerea vitezei simulate peste cea data va arata in felul urmator:

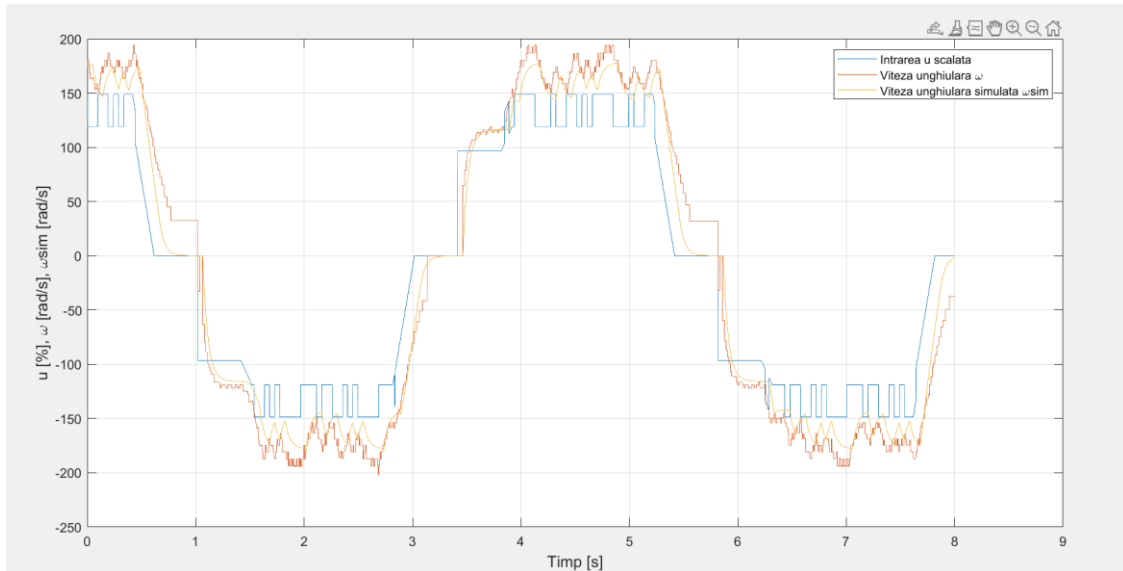


Fig.4: Viteza unghiulara simulata

1.5 Testul de albire a erorii de predictie

Incepem prin a define un interval n , care ia valori intre 1 si N (unde N reprezinta lungimea valorilor pentru viteza) si ne calculam secventa de erori $\varepsilon[n]$.

```
e = w(i1:i4)-wsim(i1:i4);
N = i4-i1;
```

-> $i1$ si $i4$ sunt indecsii ce se regasesc in Fig.3 si defines intervalul la care avem intrare de tip treapta

Ne propunem sa calculam autocorelatia acestei secvente de erori prin formula:

$$R[i] = \frac{1}{N} \sum_{k=1}^N \varepsilon[k] \varepsilon[k-i].$$

In continuare, intr o bucla de tip „for” vom calcula autocorelatiile normalizate dupa formula $RN[i] = \frac{R[i]}{R[0]}$, unde $R0$ reprezinta eroare medie patratica calculata ca si raport dintre suma patratelor erorii dintre viteza masurata si cea simulata prezentata mai sus si lungimea valorilor pentru viteza. ($R0 = \text{sum}(e.^2)/N$)

Vom verifica mai departe cate erori nu se incadreaza in banda noastra de incredere, care e delimitate de valoare de $\frac{2.17}{\sqrt{N}}$. In cazul de fata, vom observa ca 125 de erori ies din banda de incredere, deci ne rezulta ca 581 de erori se comporta cum am dori. ($N-N1=581$, unde N este definit mai sus iar $N1$ numara erorile ce ies din band cand conditia noastra impusa e respectata).

```
if abs(RN(i)) <= (2.17/sqrt(N)) % 2.17/sqrt(N) = 0.0817
    N1 = i;
```

Urmatorul pas este verificarea manuala a rezultatului obtinut mai sus:

```
R1 = sum(e(1:N-1).*e(2:N))/N;
Rn1 = R1/R0; % Rn1 = 0.9760
R2 = sum(e(1:N-2).*e(3:N))/N;
Rn2 = R2/R0; % Rn2 = 0.8517
R125 = sum(e(1:N-124).*e(125:N))/N;
Rn125 = R125/R0; % Rn125 = 0.0819

R126 = sum(e(1:N-125).*e(126:N))/N;
Rn126 = R126/R0; % Rn126 = 0.0810 < 2.17/sqrt(N) = 0.0817
```

Dupa cum se observa, intr-adevar, doar dupa 125 de valori eroarea de predictie formeaza o secventa de tip „zgomet alb”. Concluzia ar fi ca modelul nostru gasit la treapta nu e validat prin autocorelatie din cauza rezultatelor ce nu se incadreaza in banda de incredere.

```
% Autocorelatia pt modelul obtinut la treapta
e = w(i1:i4)-wsim(i1:i4);
N = i4-i1;
R0 = sum(e.^2)/N;
N1 = 0;
for i = 1:N
    R(i) = sum([e(1:N-i)]'*e((i+1):N))/N;
    RN(i) = R(i)/R0;
    if abs(RN(i)) <= (2.17/sqrt(N)) % 2.17/sqrt(N) = 0.0817
        N1 = i; % N1 = 125
        break
    end
end
Nr_e = N-N1; % Nr_e = 581
R1 = sum(e(1:N-1).*e(2:N))/N;
Rn1 = R1/R0; % Rn1 = 0.9760
R2 = sum(e(1:N-2).*e(3:N))/N;
Rn2 = R2/R0; % Rn2 = 0.8517
R125 = sum(e(1:N-124).*e(125:N))/N;
Rn125 = R125/R0; % Rn125 = 0.0819
R126 = sum(e(1:N-125).*e(126:N))/N;
Rn126 = R126/R0; % Rn126 = 0.0810 < 2.17/sqrt(N) = 0.0817
```


Partea 2 :Identificarea sistemului prin metode neparametrice de la viteza la pozitie

2.1. Calculul factorului de integrare K_i si determinarea functiei de transfer

Stim ca legatura dintre viteza si pozitie este aceea ca viteza integrata se transpune in pozitie. In cazul nostru, regimul stationar al vitezei se translateaza intr-o dreapta oblica pe graficul pozitiei, in functie de sensul semnalului.



Analizăm al hoien lloc:

$$\begin{aligned}
 \theta(t) &= \int_{t_0}^{t_1} K_i \cdot \omega(t) dt = \int_{t_0}^{t_1} K_i \cdot \underbrace{\omega_{st}}_{\frac{d\theta}{dt}} dt \\
 &= K_i \cdot \omega_{st} \cdot t \Big|_{t_0}^{t_1} + C \\
 &= K_i \cdot \omega_{st} \cdot (\underbrace{t_1 - t_0}_{\Delta t}) + \underbrace{C}_{\theta(t_0)} \\
 \Rightarrow \theta(t) - \theta(t_0) &= K_i \cdot \omega_{st} \cdot \Delta t \\
 \Rightarrow K_i &= \frac{\theta(t) - \theta(t_0)}{\Delta t \cdot \omega_{st}}
 \end{aligned}$$

Avand formula pentru factorul de integrare, il vom calcula pe exemplul nostru si vom ajunge la urmatorul rezultat:

```

>> Ki_yw = (y(i4)-y(i3))/wst/(t(i4)-t(i3))

Ki_yw =

    4.6036
  
```

În continuare, o să aflăm funcția de transfer dintre viteză și poziție:

```
>> H_yw = tf(Ki_yw, [1, 0])

H_yw =

    4.604
    -----
         s

Continuous-time transfer function.
```

-observăm integratorul care ne dă relația dintre viteză și poziție și factorul de integrare care ne arată o amplificare.

2.2 Validarea modelului obținut și calcularea erorii medii pătratice normalizate

Modelul matematic se va simula cu ajutorul spațiului stărilor și vom calcula eroarea medie pătratică dintre semnalul dat și cel simulat (eroare de 10.74%).

```
Ayw = 0;
Byw = Ki_yw;
Cyw = 1;
Dyw = 0;
ysim_yw = lsim(Ayw, Byw, Cyw, Dyw, w, t, y(1));
figure
plot(t, [y, ysim_yw]), grid;
title('Pozitia si Pozitia simulata');
legend('y', 'y simulat')
xlabel('Timp [s]', 'FontSize', 12);
ylabel('y [impulsuri], ysim [impulsuri]', 'FontSize', 12);
eMIN_yw = norm(y-ysim_yw)/norm(y-mean(y))% 10.74
```

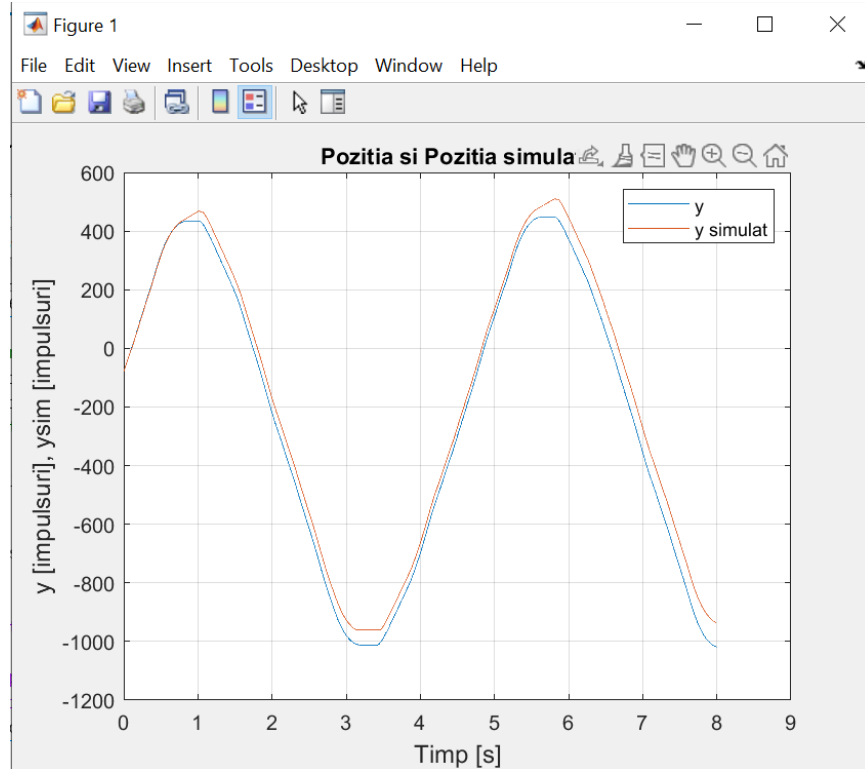


Fig.5: Suprapunerea pozitiei date si a pozitiei calculate

Capitolul 3

Partea 2 :Identificarea sistemului prin metode parametrice

2.1 Identificarea parametrica la viteza

2.1.1 Vizualizarea si alegerea datelor de identificare si validare

Alegem datele de pe SPAB-uri (semnal periodic aleatoriu binar), deoarece acestea sunt semnale bogate in armonice, iar acest lucru duce la o viteza maxima a motorului pe acele portiuni. Le alegem dupa figura de mai jos, intre i7 si i8 vom face identificarea, iar pe celalalt sens, intre i9 si i10 vom face validarea datelor.

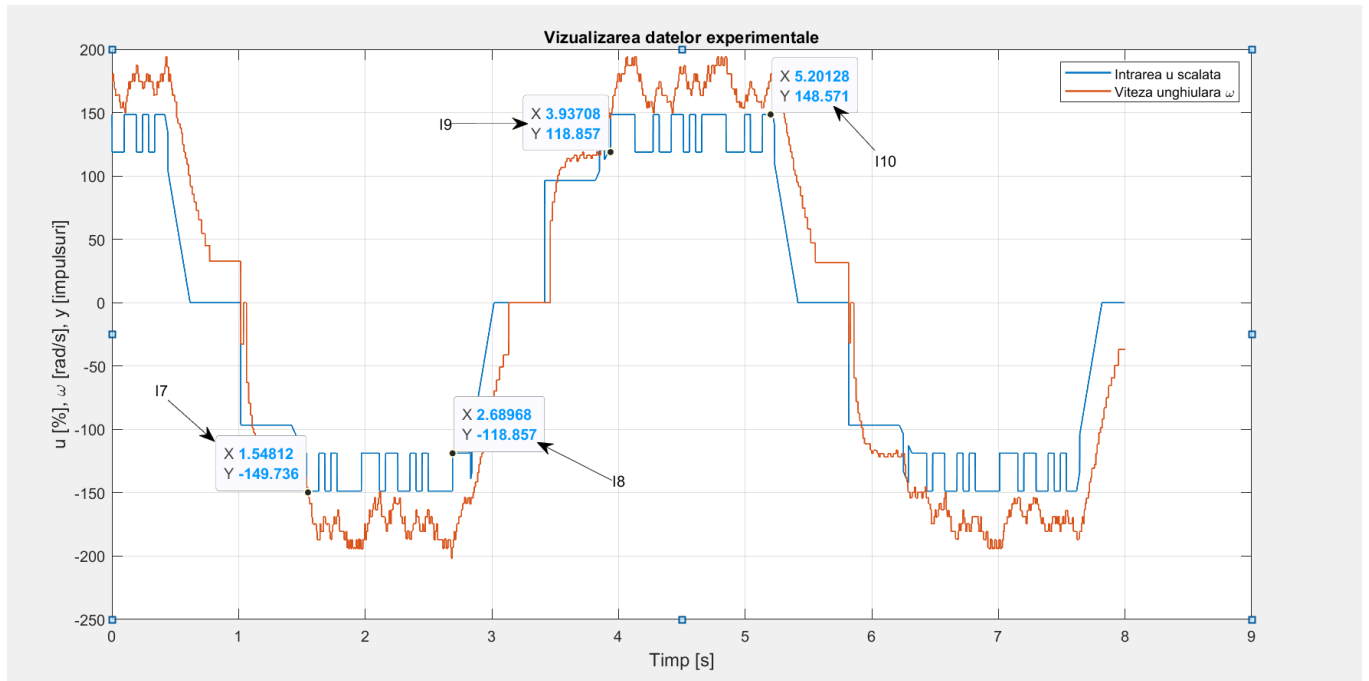


Fig.6: Datele de identificare si validare

Pentru retinerea datelor, ne vom folosi de functia “iddata” din Matlab, care preia ca si argumente in cazul de fata viteza pe intervalul de identificare, apoi intrarea pe acelasi interval si la final perioada de esantionare si respectiv viteza si intrarea pe intervalul de validare, si implicit perioada de esantionare.

```
i7=1830;
i8=3348;
i9=4688;
i10=6199;

Te=t(2)-t(1);

data_id_w=iddata(w(i7:i8),u(i7:i8),Te);
data_vd_w=iddata(w(i9:i10),u(i9:i10),Te);
```

In figurile urmatoare vom vedea reprezentarea datelor de identificare, cat si a datelor de validare:

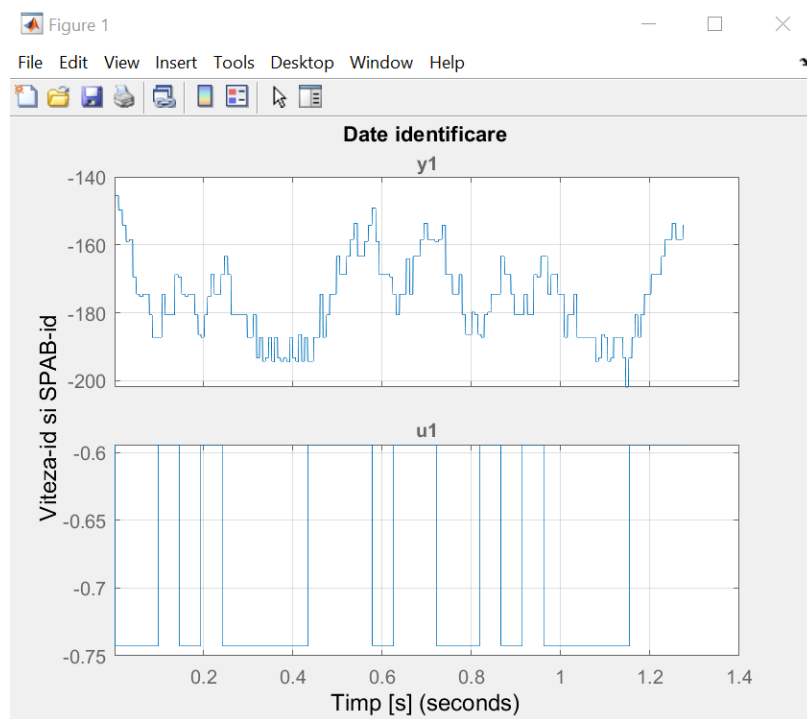


Fig.7: Date identificare

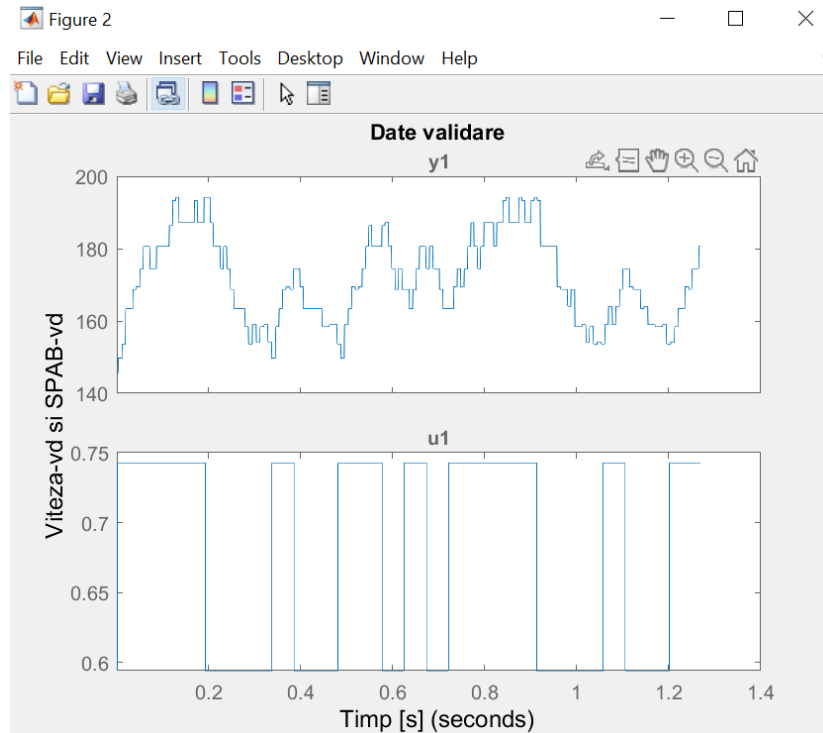
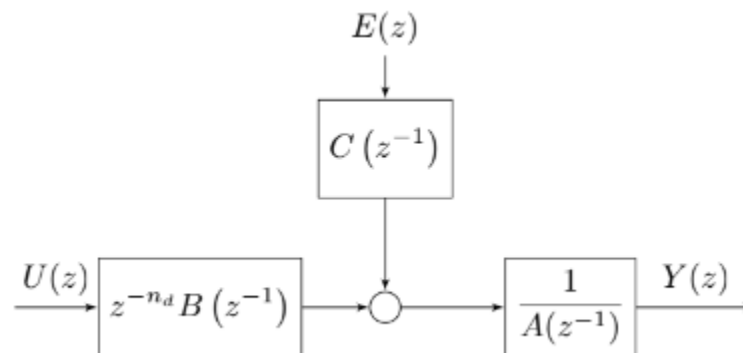


Fig.8: Date de validare

2.1.2 Validarea sistemului prin autocorelatie folosind metoda ARMAX(metoda celor mai mici patrate extinsa)

Pentru validarea sistemului prin autocorelatie alegem sa folosim metoda ARMAX in cazul de fata, deoarece prezinta cele mai bune rezultate. Functioneaza fara decimarea datelor, deci vom lucra cu datele initiale. Functia ARMAX primeste ca si parametrii un obiect de tipul "iddata" si parametrii de structura nA(numarul de poli), nB(numarul de zerouri + 1), nC(zgomotul) si nD(numarul tactilor de intarziere). Functia va returna un obiect de tipul "idpoly" care ne prezinta modelul matematic al sistemului.



Este recomandat, pentru aflarea fitului optim, sa modificam numarul tactilor de intarziere pana vom avea o eroare minima si o validare cat mai buna prin autocorelatie.

```
% armax fara decimare
mw_armax=armax(data_id_w,[1,1,1,1]);
mw_armax9=armax(data_id_w,[1,1,1,9])%mai bun fit
```

Se observa ca modificand nD-ul pana la 9, vom obtine un fit semnificativ mai bun. Modelul matematic obtinut prin ARMAX cu numarul de tacti optim este prezentat in continuare:

```
mw_armax9 =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
A(z) = 1 - 0.9846 z^{-1}

B(z) = 3.986 z^{-9}

C(z) = 1 - 0.01272 z^{-1}
```

Verificarea statistica a rezultatului se va face cu functia “resid”, iar gradul de suprapunere se va observa cu ajutorul functiei “compare”, care compara datele de validare selectate anterior cu datele primite de la functia ARMAX.

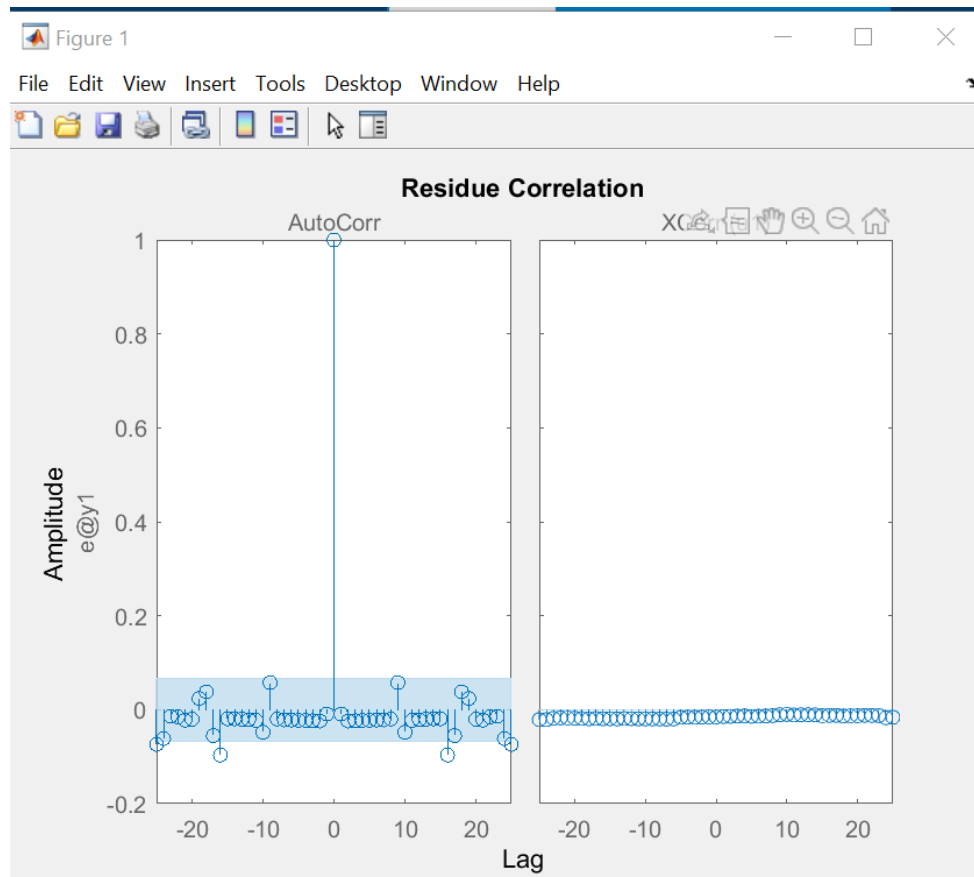


Fig.8: Validarea statistica prin autocorelatie

Se observa ca primele $nA+nB$ valori care ne intereseaza se afla in banda de incredere, deci semnalul nostrum se valideaza prin autocorelatie.

In Fig.9, prezentata mai jos, vom observa ca modelul definit prin ARMAX cu $nD=9$ ne da un fit mai bun cu 5.44% decat cel cu $nD=9$ si o eroare de $100-65.29=34.71\%$, deci vom pastra acest model pe mai departe pentru viteza.

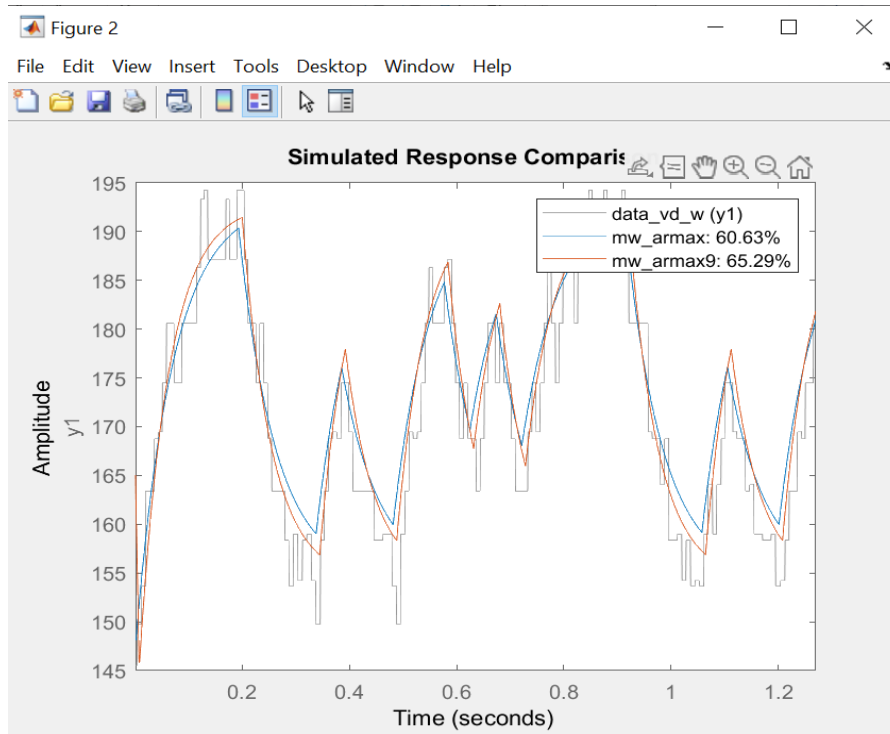


Fig.9: Gradul de suprapunere

In continuare, vom trece modelul in continuu si vom scoate functia de transfer cu metoda “zero order hold”, care se aplica in practica pentru modelarea sistemelor.

```
mw_armax9_inceput=tf([0 3.986],[1 -0.9846],Te,'variable','z^-1')
Mw_armax9_inceput=d2c(mw_armax9_inceput,'zoh')
Hw_armax_c = tf(4782,[1 18.48],'iodelay',9*Te)
```

In urma rularii acestui cod vom observa ca functia de transfer rezultata este:

```
Hw_armax_c =

          4782
exp(-0.00756*s) * -----
                s + 18.48

Continuous-time transfer function.
```

Pentru simulare, se trece acest model in spatial starilor pentru a avea control asupra conditiilor initiale si pentru o suprapunere cat mai exacta a vitezei calculate peste cea determinata experimental.


```

sys_w_armax = ss(Hw_armax_c);
figure
plot(t,200*u);
wsim2 = lsim(sys_w_armax.A, sys_w_armax.B, sys_w_armax.C, sys_w_armax.D, u, t, w(1)/sys_w_armax.C);
hold on
plot(t,[w,wsim2]), grid;
legend('Intrarea u scalata', 'Viteza unghiulara \omega', 'Viteza unghiulara simulata cu armax \omegasim-armax');
xlabel('Timp [s]','FontSize',12);
ylabel('u [%], \omega [rad/s], \omegasim [rad/s]', 'FontSize',12);

```

Dupa rularea acestui cod vom obtine urmatoarul rezultat:

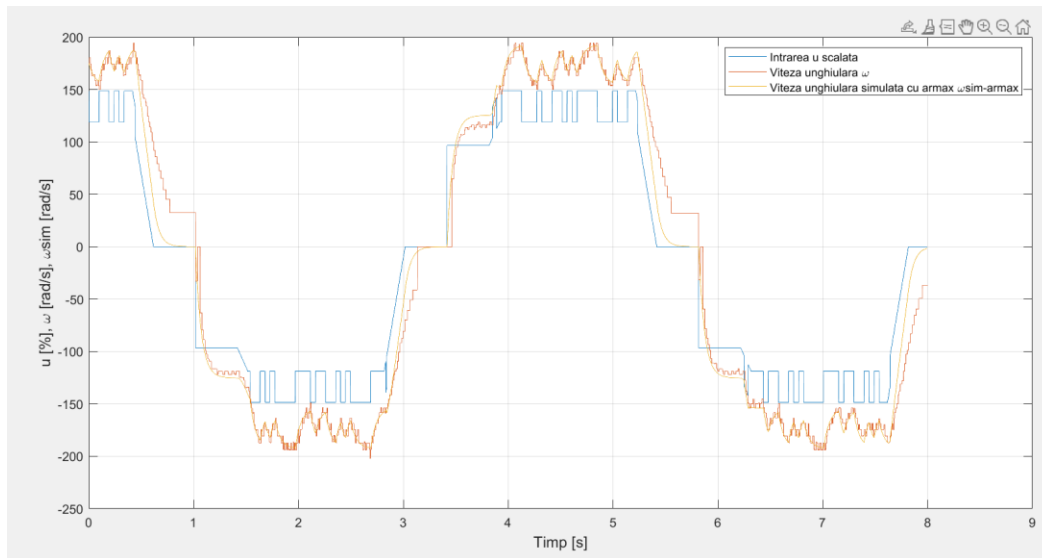


Fig.10: Viteza unghiulara simulata

Si o eroare medie patratica de 13.48%, data de relatia $e_{MIN_w2} = \text{norm}(w - w_{sim2}) / \text{norm}(w - \text{mean}(w))$.

2.1.3 Decimarea datelor experimentale

Incerand metodele de intercorelatie pe viteza si autocorelatia si intercorelatia pentru pozitie vom observa ca datele noastre trebuie rafinate. Am ales metoda de preprocesare data prin decimarea datelor, deoarece stim ca valorile culese sunt redundante. Vom verifica cate esantioane se repeta de cele mai putine ori pentru varful de valoare maxima al vitezei si vom observa ca datele trebuie scalate cu un $N=8$.

In urmatoarea secventa Matlab se prezinta codul pentru decimarea datelor si implicit valorile pe care vom aplica urmatoarele metode parametrice.

```

%decimare date viteza si pozitie
%194.2334 apare de 8 ori
idx_id=[i9:i10]';
idx_vd=[i11:i12]';
N=8;
%decimare
u_decimat_id=u(i9:N:i10);
w_decimat_id=w(i9:N:i10);
y_decimat_id=y(i9:N:i10);

u_decimat_vd=u(i11:N:i12);
w_decimat_vd=w(i11:N:i12);
y_decimat_vd=y(i11:N:i12);

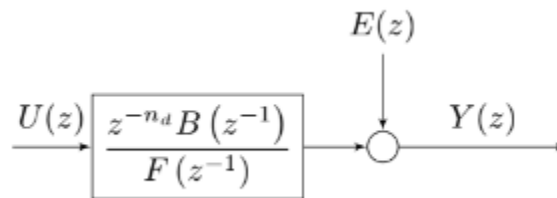
%date id si validare la viteza cu decimare
dw_decimat_id=iddata(w_decimat_id,u_decimat_id,N*Te);
dw_decimat_vd=iddata(w1_vd,u_decimat_vd,N*Te);

%date id si validare la pozitie cu decimare
dy_decimat_id=iddata(y_decimat_id,w_decimat_id,N*Te);
dy_decimat_vd=iddata(y_decimat_vd,w_decimat_vd,N*Te);

```

2.1.4 Validarea sistemului prin intercorelatie folosind metoda OE(Output Error)(cu date decimate)

Verificand metodele de intercorelatie, ajungem la concluzia ca “OE” ne ofera cele mai bune rezultate. Aceasta metoda primeste ca si parametri de intrare un obiect de tipul “iddata” si parametri de structura nF(numarul de poli), nB(numarul de zerouri + 1) si nD(numarul tactilor de intarziere) si ne da la iesire un obiect de tipul “idpoly” care contine modelul matematic al sistemului. Toata perturbatia se gaseste nemodelata la iesire.



Cu ajutorul functiei “resid” vom verifica validarea statistica prin intercorelatie, iar prin functia “compare” vom identifica gradul de suprapunere al modelului obtinut prin “OE”.

Setul de date folosit va fi acela de date decimate anterior.

```

%metoda oe viteza(cu decimare)
mwd_oe=oe(dw_decimat_id,[1, 1, 1]);
figure
resid(dw_decimat_vd,mwd_oe);
figure
compare(dw_decimat_vd,mwd_oe);

```

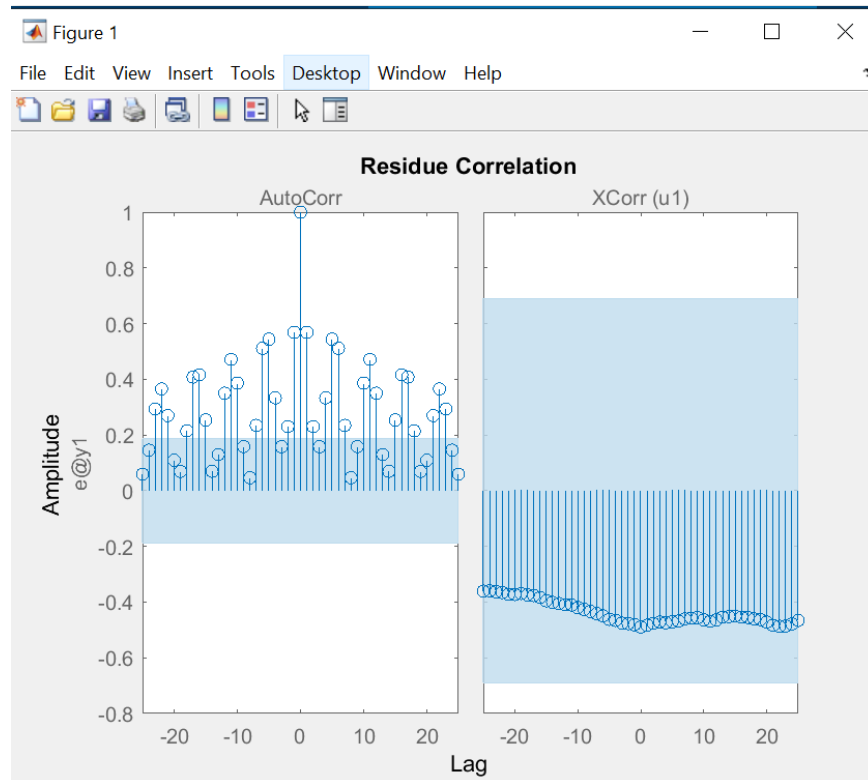


Fig.11: Validarea statistica prin intercorelatie

Observam ca termenii $nB+nF$ se afla in banda de incredere, deci testul de intercorelatie este trecut cu success.

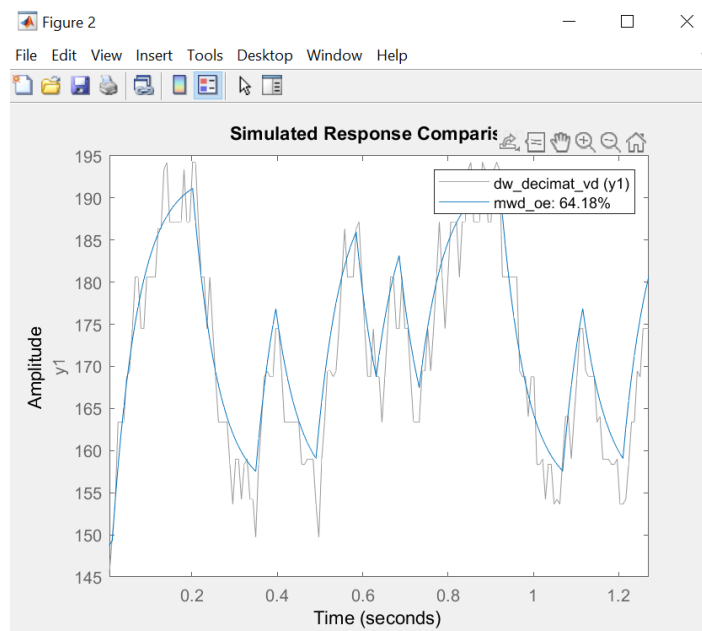


Fig.12: Gradul de suprapunere

Din functia “compare” se observa ca am obtinut un fit de 64.18%, deci o eroare de 100-64.18%=35.82%.

Vom trece in continuu modelul dat de catre “OE” si vom scoate functia de transfer cu metoda “zero order hold”.

```
>> Hw_oe_inceput=tf(mwd_oe.B,mwd_oe.F,Te,'iodelay',1,'variable','z^-1')
Hw_oe=d2c(Hw_oe_inceput,'zoh')

Hw_oe_inceput =

          27.58 z^-1
z^(-1) * -----
        1 - 0.8939 z^-1

Sample time: 0.00084 seconds
Discrete-time transfer function.

Hw_oe =

          3.471e04
exp(-0.00084*s) * -----
                s + 133.6

Continuous-time transfer function.
```

In continuare, vom trece functia de transfer in spatiul starilor pentru a putea simula viteza si pentru a controla conditiile initiale.Mai apoi vom scoate eroarea medie patratica=23.35%.

```
sys_w_oe = ss(Hw_oe);
figure
plot(t,200*u);
wsim3 = lsim(sys_w_oe.A, sys_w_oe.B, sys_w_oe.C, sys_w_oe.D, u, t, w(1)/sys_w_oe.C);
hold on
plot(t,[w,wsim3]), grid;
legend('Intrarea u scalata', 'Viteza unghiulara \omega', 'Viteza unghiulara simulata cu oe \omegasim-oe');
xlabel('Timp [s]','FontSize',12);
ylabel('u [%], \omega [rad/s], \omegasim [rad/s]', 'FontSize',12);

eMIN_w3 = norm(w-wsim3)/norm(w-mean(w))
```

2.2 Identificarea parametrica la pozitie

2.2.1 Vizualizarea si alegerea datelor de identificare respective validare

Datele folosite pentru identificarea pozitiei sunt aceleasi cu datele decimate folosite pentru viteza.

```
%decimare date viteza si pozitie
%194.2334 apare de 8 ori
idx_id=[i9:i10]';
idx_vd=[i11:i12]';
N=8;
%decimare
u_decimat_id=u(i9:N:i10);
w_decimat_id=w(i9:N:i10);
y_decimat_id=y(i9:N:i10);

u_decimat_vd=u(i11:N:i12);
w_decimat_vd=w(i11:N:i12);
y_decimat_vd=y(i11:N:i12);

%date id si validare la viteza cu decimare
dw_decimat_id=iddata(w_decimat_id,u_decimat_id,N*Te);
dw_decimat_vd=iddata(w1_vd,u_decimat_vd,N*Te);

%date id si validare la pozitie cu decimare
dy_decimat_id=iddata(y_decimat_id,w_decimat_id,N*Te);
dy_decimat_vd=iddata(y_decimat_vd,w_decimat_vd,N*Te);
```

2.2.3 Validarea sistemului prin autocorelatie folosind metoda ARMAX(metoda celor mai mici patrute extinsa)

Vom aplica metoda ARMAX la pozitie, analog vitezei, cu precizarea ca timpul mort lipseste deci nD(numarul tactilor de intarziere) va fi nul.

```
%armax pozitie cu decimare
myd_armax=armax(dy_decimat_id,[1,1,1,0]);
figure
resid(dy_decimat_vd,myd_armax);
figure
compare(dy_decimat_vd,myd_armax);%e validata
```

In Fig.13 de mai jos se observa ca sistemul nostru trece testul de autocorelatie deoarece cele nA+nB valori se afla in banda de incredere.

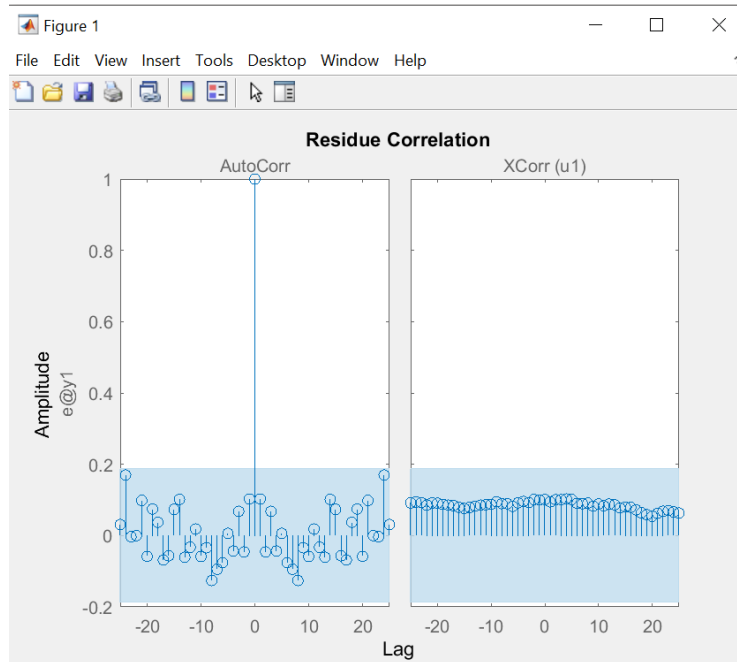


Fig.13: Autocorelatia la pozitie

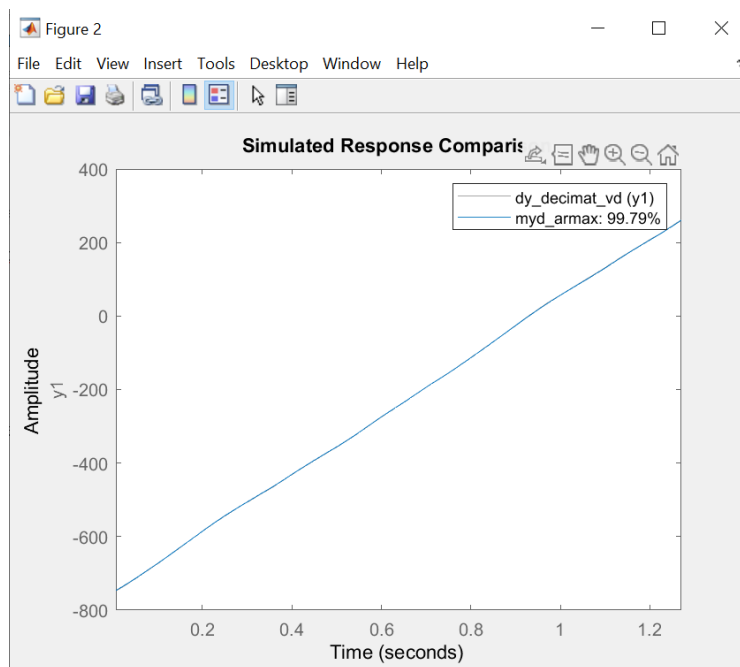


Fig.14: Gradul de suprapunere

In Fig.14 se observa gradul de suprapunere dat de functia "compare". Acesta este de 99.79% si ne impune o eroare de $100 - 99.79 = 0.21\%$.

In continuare vom scoate functia de transfer in discret din metoda ARMAX pentru pozitie.

```
>> Hyw = tf(myd_armax.B, myd_armax.A, Te, 'variable', 'z^-1')

Hyw =

    0.0312
    -----
    1 - z^-1
```

Urmeaza sa calculam functia de transfer in continuu. Pentru factorul de integrare Ki vom folosi formula $Ki = \frac{b1}{Te_p}$, unde b1 este matricea B din metoda ARMAX, iar Te_p, perioada de esantionare dupa decimare. Deoarece Matlab nu are impuls de arie unitara, suntem nevoiti sa impartim b1 la perioada de esantionare. Cu metoda d2c nu reusim sa scoatem corect functia de transfer deoarece nu ne apare integratorul ideal. Deci vom construi aceasta functie de transfer si mai apoi vom face trecerea in spatiul starilor pentru simulare si pentru controlul asupra conditiilor initiale.

```
Hyw = tf(myd_armax.B, myd_armax.A, Te, 'variable', 'z^-1');
b1 = myd_armax.B;
Te_ARMAX = 0.00672; % Perioada de esantionare dupa decimare

% Constanta de integrare:
Ki_nou = b1/Te_ARMAX;

Hyw_c_2 = tf(Ki_nou, [1, 0]);
sys_y = ss(Hyw_c_2);
ysim2 = lsim(sys_y.A, sys_y.B, sys_y.C, sys_y.D, w, t, y(1)/sys_y.C);
```

Functia de transfer in continuu va fi:

```
>> Hyw_c_2 = tf(Ki_nou, [1, 0])

Hyw_c_2 =

    4.643
    -----
    s
```

Iar simularea pozitiei va fi prezentata in Fig.15 cu o eroare de 10.50%.

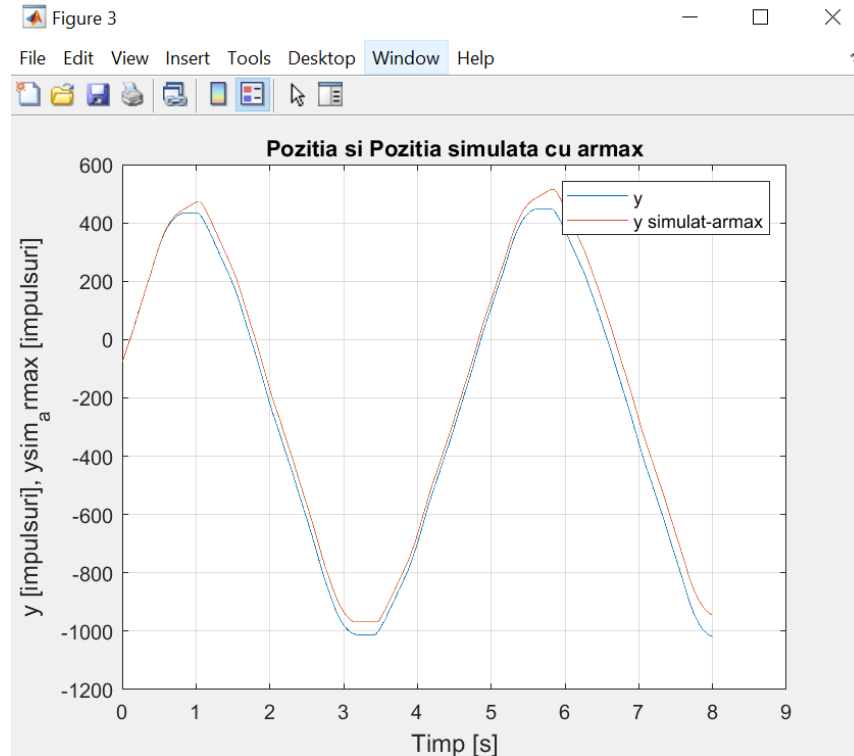


Fig.15: Simularea pozitiei

2.2.4 Validarea sistemului prin intercorelatie folosind metoda OE(Output Error)

Vom aplica metoda OE la pozitie, analog vitezei, cu precizarea ca timpul mort lipseste deci nD(numarul tactilor de intarziere) va fi nul.

```
%metoda oe pozitie cu decimare
myd_oe=oe(dy_decimat_id,[1, 1, 0]);
figure
resid(dy_decimat_vd,myd_oe);
figure
compare(dy_decimat_vd,myd_oe);%e valida
```


In Fig.13 de mai jos se observa ca sistemul nostru trece testul de intercorelatie deoarece cele $nF+nB$ valori se afla in banda de incredere.

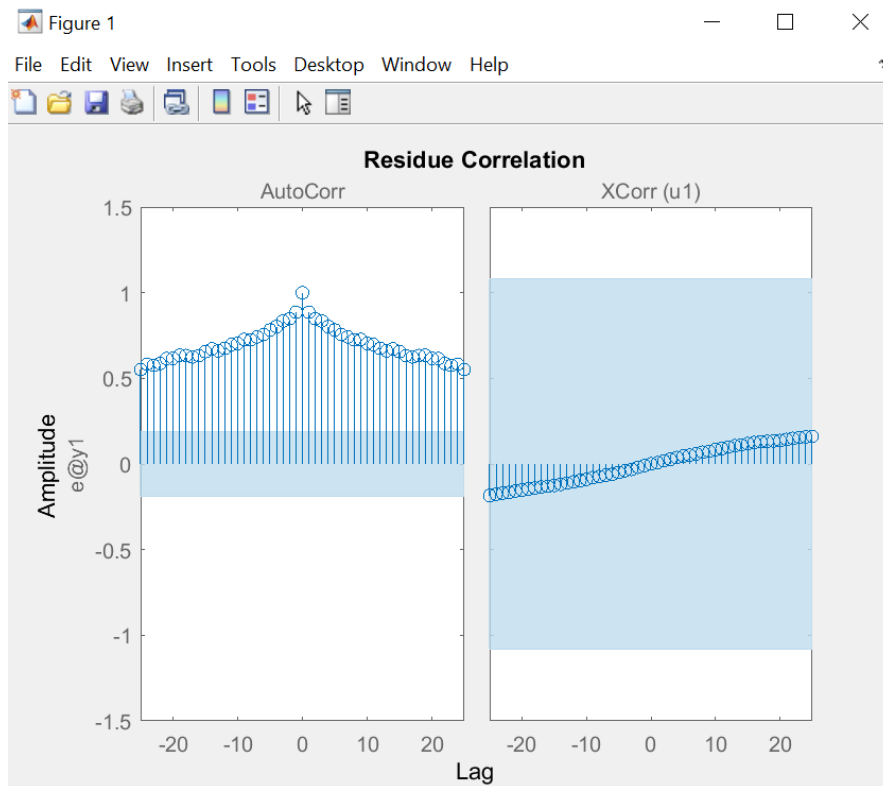


Fig.16: Intercorelatia la pozitie

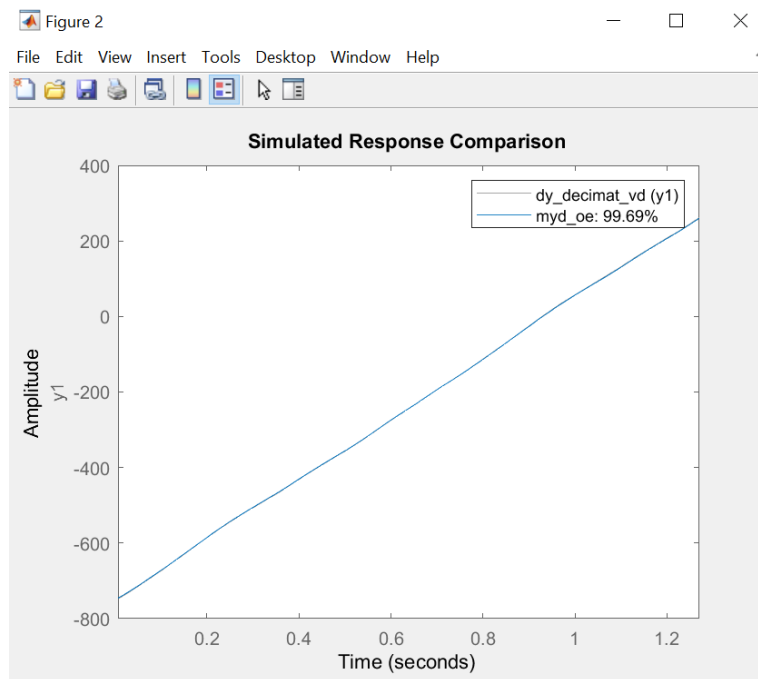


Fig.17: Gradul de suprapunere

In Fig.14 se observa gradul de suprapunere dat de functia “compare”. Acesta este de 99.69% si ne impune o eroare de $100-99.69=0.31\%$.

In continuare vom scoate functia de transfer in discret din metoda OE pentru pozitie.

```
>> Hyw_oe = tf(myd_oe.B, myd_oe.F, Te, 'variable', 'z^-1')

Hyw_oe =

    0.03117
    -----
    1 - z^-1
```

Urmeaza sa calculam functia de transfer in continuu. Pentru factorul de integrare K_i vom folosi formula $K_i = \frac{b_1}{T_{e_p}}$, unde b_1 este matricea B din metoda OE, iar T_{e_p} , perioada de esantionare dupa decimare. Deoarece Matlab nu are impuls de arie unitara, suntem nevoiti sa impartim b_1 la perioada de esantionare. Cu metoda d2c nu reusim sa scoatem correct functia de transfer deoarece nu ne apare integratorul ideal. Deci vom construi aceasta functie de transfer si mai apoi vom face trecerea in spatiul starilor pentru simulare si pentru controlul asupra conditiilor initiale.

```
Hyw_oe = tf(myd_oe.B, myd_oe.F, Te, 'variable', 'z^-1')
b1_oe = myd_oe.B;
Te_oe = 0.00672;
Ki_nou_oe = b1_oe/Te_oe;
Hyw_c_3 = tf(Ki_nou_oe, [1, 0]);
sys_y_oe = ss(Hyw_c_3);
ysim3 = lsim(sys_y_oe.A, sys_y_oe.B, sys_y_oe.C, sys_y_oe.D, w, t, y(1)/sys_y_oe.C);
```

Functia de transfer in continuu va fi:

```
>> Hyw_c_3 = tf(Ki_nou_oe, [1, 0])

Hyw_c_3 =

    4.639
    -----
    s

Continuous-time transfer function.
```

Iar simularea pozitiei va fi prezentata in Fig.15 cu o eroare de 10.52%.

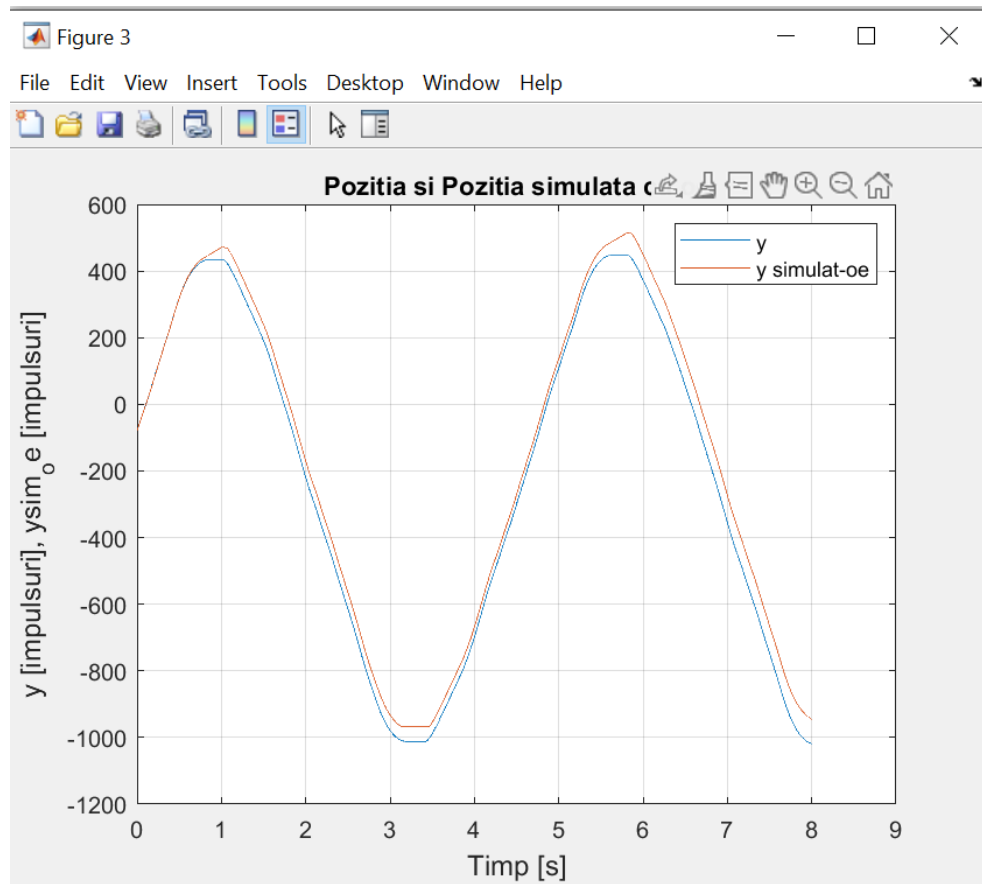


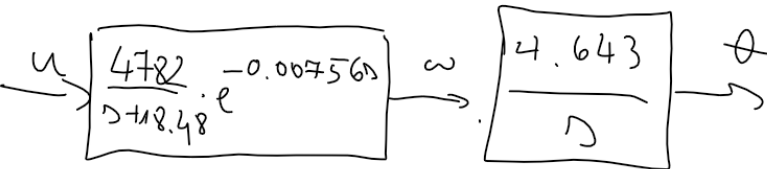
Fig.18: Simularea pozitiei

Capitolul 4

Concluzii

Dupa identificarea modelelor matematice care ne prezinta sistemul de control al axei motorului BLDC, vom observa ca metodele dorite pentru modelele matematice vor fi "ARMAX" la intrare-viteza si "ARMAX" la viteza-pozitie, deoarece aduc cea mai mica eroare, respecta validarea prin autocorelatie si impun cea mai buna suprapunere peste datele experimentale, metoda "OE" nu este atat de satisfacatoare, asa ca nu o vom folosi pe aceste date.

Ne mai ramane doar de inseriat cele doua modele obtinute pentru a vizualiza intreaga dinamica a sistemului.


$$\Rightarrow H_{\text{final}} = \frac{2.22 \cdot 10^4}{s^2 + 18.48s} \cdot e^{-0.00756s}$$

