Proiect IS

Olaru Miruna-Paraschiva; Trif Rebeca; Iosif Alexandru-Mihai

November 12, 2022

(Numar set de date: 23)

1 Introducere

In acest proiect vom studia componentele vandute de un magazin de instalatii. Informatiile furnizate pentru realizarea proiectului sunt reprezentate de cantitatile vandute lunar pe parcursul mai multor ani. Si anume: time si y din Figura 1. In cadrul vectorului "time" regasim indexarea lunilor corespondente anilor pe baza carora au fost furnizate datele. Vectorul are 108 elemente (1-12 primul an, 13-24 al doilea an, etc; observam o periodicitate de 12 luni). Al doilea vector, "y", contine valoarea asociata fiecarui element din vectorul time (de exemplu, in luna corespunzatoare indexului 3 s-a vandut o cantitate de 8 produse).

Scopul proiectului este realizarea unui model care poate fi utilizat pentru prezicerea cantitatilor de produse ce vor fi vandute. Noi stim cate produse au fost vandute, din datele de identificare (date primite pe o anumita perioada), dar folosindu-ne de modelul creat putem anticipa vanzarile viitoare ce vor fi efectuate de magazin.

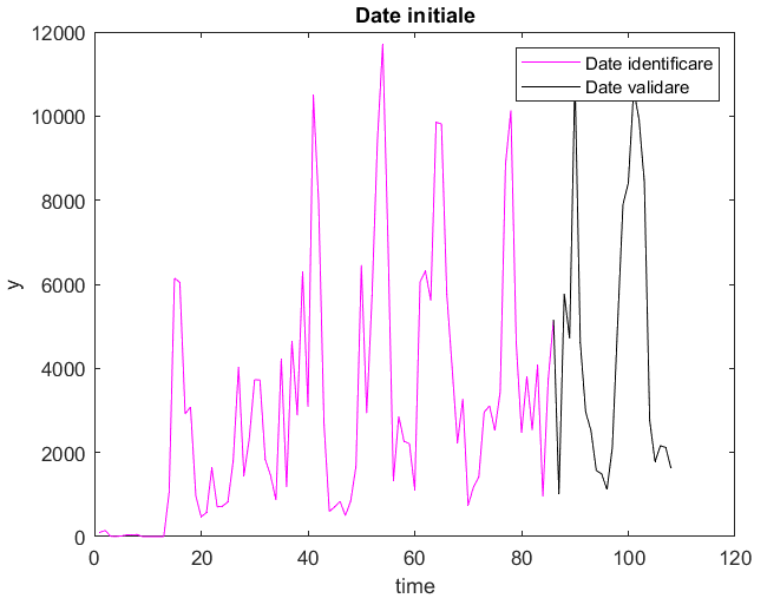

Figure 1:

2 Descrierea structurii aproximatorului si gasirea parametrilor

In vederea realizarii modelului am impartit setul de date primit in date de identificare (80%) si date de validare (20%) (Figura 1). Modelul este reprezentat printr-o functie de aproximare care utilizeaza setul de date de identificare si in componenta acesteia se regasesc: partea liniara si o serie de armonice

(baza Fourier). Prima parte are o tendinta liniara, de rampa, iar seria contine un numar de termeni "m" pe care il putem modifica pentru a obtine rezultate cat mai optime (suprapuneri cat mai bune intre datele de validare si reprezentarea grafica a functiei de aproximare pentru datele de validare). Modelul ar fi liniar, dar prin adaugarea seriei Fourier, va lua forma unor armonice.

In cadrul functiei de aproximare fiecare termen al partii liniare cat si al partii armonice este inmultit cu un coeficient, iar toti coeficientii sunt parametrii ce formeaza o matrice pe care o numim "Tetha". Pentru aflarea matricii "Tetha" avem nevoie de o matrice pe care o numim "Xf" (PHI) si de 80% din vectorul initial " y' ". Matricea "Xf" contine pe fiecare linie termenii functiei de aproximare lipsiti de coeficienti si elementele fiecarei linii iau valori in functie de indexul corespunzator y-ului de identificare. Dupa aflarea acestor matrici, rezolvam ecuatia

$$Y = Xf * Tetha \tag{1}$$

folosind functia Matlab linsolve: "Tetha" = linsolve(Xf,y1').

3 Explicarea implementarii

In cadrul unei bucle care merge de la 1 pana la o valoare "m" aleasa, folosind setul de identificare am calculat "Tetha" pentru fiecare valoare a lui "m". Tot in cadrul acestei bucle am calculat erorile medii patratice "MSE" (suma patratului diferentei dintre valorile functiei aproximate si setul inital de date impartita la numarul de elemente) pentru identificare si validare folosind tot fiecare valoare distincta a lui "m" si le-am retinut in cate un vector. In plus, am determinat cea mai mica eroare patratica, iar prin intermediul acesteia am dedus "Tetha" pentru "m" minim, care au fost memorate la randul lor. Pe parcursul implementarii am realizat de asemenea o serie de grafice ce vor fi explicate ulterior.

4 Interpretare MSE

Am reprezentat MSE in functie de "m" sub forma de grafice.

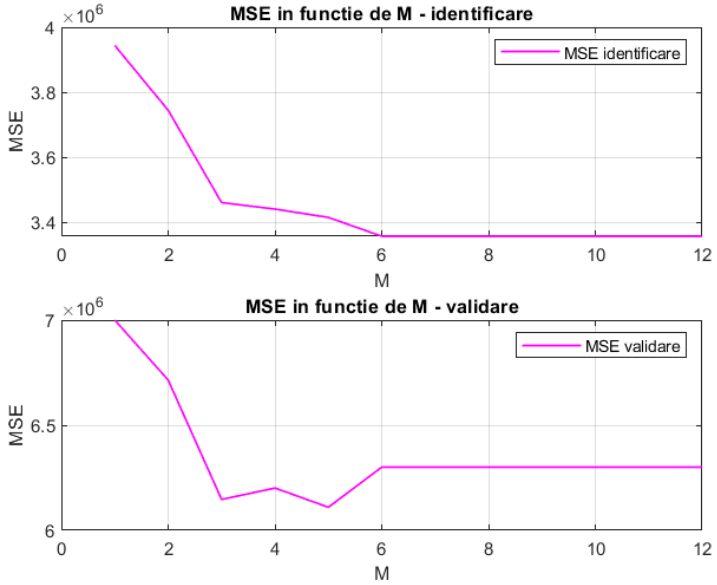

Figure 2:

Din primul grafic din figura 2, putem observa faptul ca MSE are o tendinta descendenta pana intr-o anumita valoare (6 in acest caz), dupa care valoarea erorii ramane constanta, deoarece matricea dupa acea valoare a lui "m" nu mai este inversabila, eroarea ramane constanta. Astfel, nu mai pot fi generate alte valori pentru "Tetha".

Din al doilea grafic din figura 2, putem observa faptul ca punctul in care valoarea este minima ("m" = 5) nu coincide cu punctul in care incepe sa devina constanta, dar pentru modelul nostru vom alege fix punctul in care eroarea este cea mai mica pentru a obtine cea mai buna modelare posibila pe care ne-o poate furniza setul nostru de date (reprezentarea grafica pentru a functiei de aproximare pentru un set de date se apropie cel mai mult de forma reala a setului de date).

Implementare MSE identificare in matlab (idem validare):

```
for i = 1:86
    e = e + (y(i) - f(i,M,Tetha,nap))^2;
end

%calculam erorile pentru datele de identificare
MSE_id = e/(length(1:86));
MSE_identificare(M) = MSE_id;
```

5 Grafice reprezentative

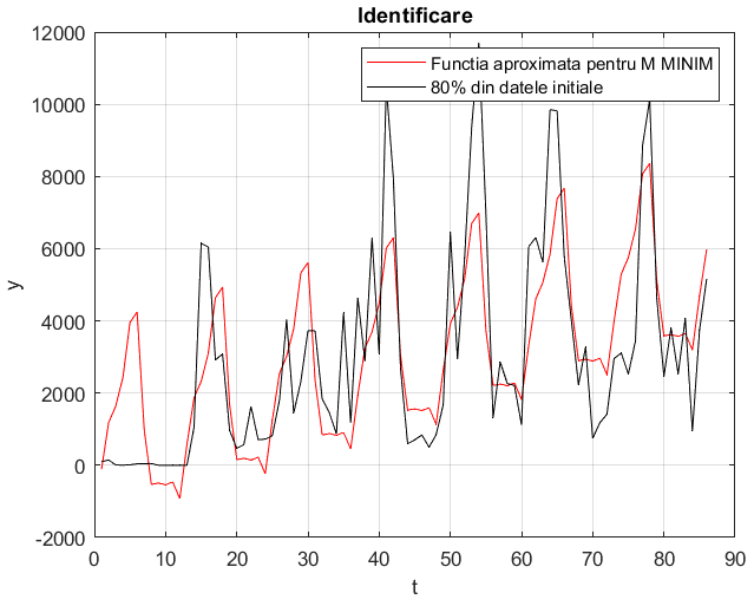

Figure 3:

Dupa ce am gasit "m" optim ca fiind 5, am realizat graficul din figura 3 in care putem observa capacitatea modelului de a se mula pe setul nostru de date de identificare.

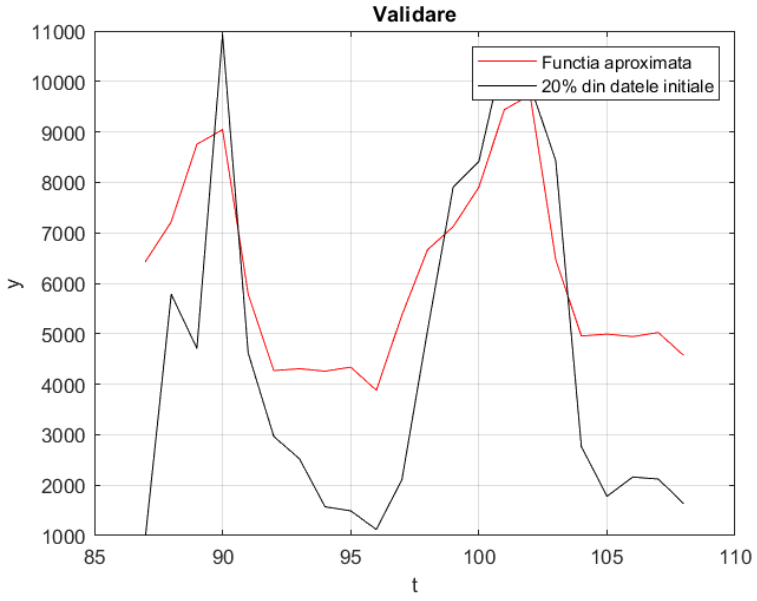

Figure 4:

Dupa ce am aflat modelul pentru datele de identificare, am verificat pliabilitatea pe datele de validare pentru "m" minim si am observat cat de bine se muleaza analizand Figura 4.

6 Imbunatatiri

Observam ca erorile sunt foarte mari chiar si pentru "m" optim (ajungand la ordinul milioanelor), asa ca am incercat sa aducem eventuale imbunatatiri pentru a incerca sa scadem aceste valori. O posibila modalitate ar fi extinderea componentei liniare a functiei de aproximare cu un polinom de aproximare al carui grad este reglabil (cu ajutorul unei variabile notate "nap", dar polinomul este de gradul nap-1). Pentru realizarea acestei imbunatatiri sunt necesare cateva modificari la nivelul codului. In primul rand, atunci cand matricea "Xf" este realizata, este necesara introducerea noului polinom in locul componentei liniare precedente, deci si "Tetha" va fi modificat, iar acest lucru se va realiza cu ajutorul functiei "o". Aceasta functie calculeaza un polinom de grad variabil.

```
  function o=MP(k,n)
o=[];
for i=1:n
    o=[o k^(i-1)];
end
end
```

In al doilea rand, functia de aproximare sufera modificari, apare un parametru in plus, reprezentat de gradul nap-1. Partea liniara este inlocuita cu polinomul aproximat.

```
 function yhead = f(time, m, tetha,n)
yhead=0;
for i=1:n
    yhead=yhead+tetha(i)*time.^(i-1);
end
for i=1:m

    yhead=yhead+cos(2*pi*i*time/12)*tetha(2*i+n1)
    +sin(2*pi*i*time/12)*tetha(2*i+n);
```

```
end
end
```

OBSERVATIE: Pentru "nap" = 2, se pastreaza forma initiala a functiei de aproximare, ceruta in proiect!!!! Imbunatatirile se aduc pentru "nap" mai mare decat 2.

Pentru "nap" luand valorile 3(Figura 5), 4(Figura 6) si 5(Figura 7) observam o mulare mai buna pe datele de validare. Eroarea medie patratica minima scade odata cu cresterea nap. Pentru "nap" = 5 avem cea mai mica eroare, mai mica decat cea gasita anterior(nap=3: MSE_minim = 5960320, nap=4:MSE_minim = 5193890, nap=5:MSE_minim = 4324580).

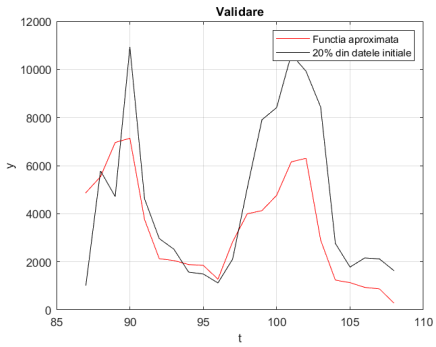

Figure 5:

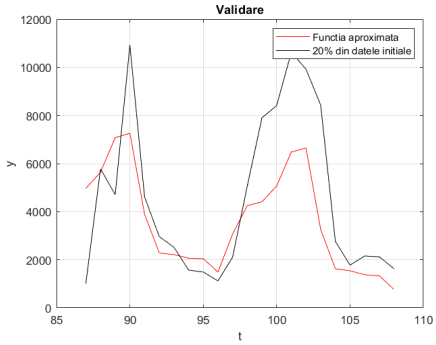

Figure 6:

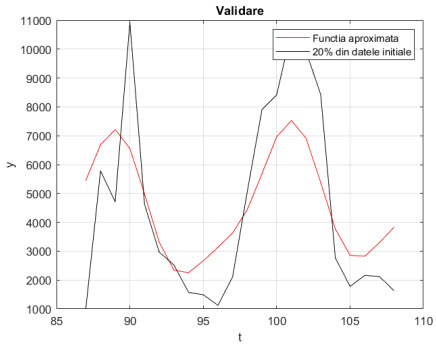

Figure 7:

Pentru "nap" mai mare sau egal cu 6 (Figura 8), functia de aproximare intra in supraantrenare pe datele de validare.

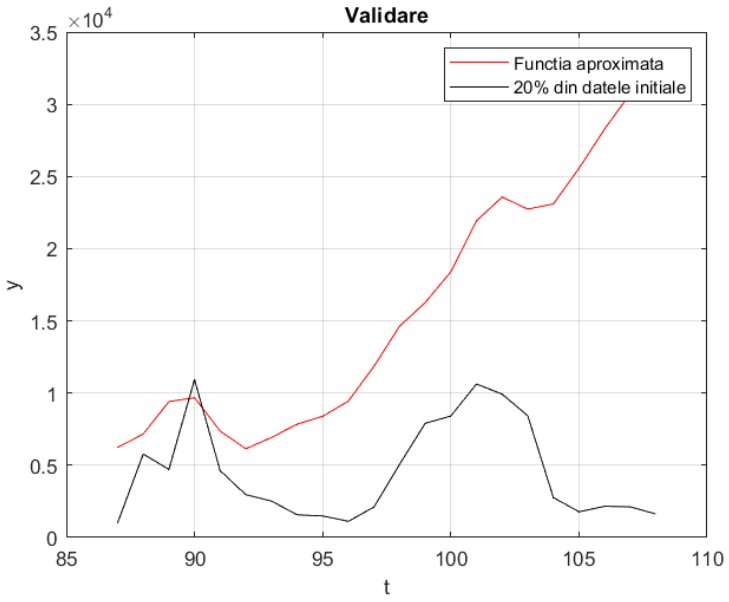

Figure 8:

7 Concluzii

In cadrul proiectului am analizat cat de optima este functia de aproximare propusa, pentru o aplicatie reala. Modelul poate fi utilizat pentru anticiparea unor posibile vanzari pe perioade viitoare. General vorbind, functia de aproximare nu se pliaza eficient pe datele magazinului, adica nu estimeaza corect vanzarile reale ce vor fi efectuate, ulterior am venit cu imbunatatiri care au redus considerabil erorile(de la 6109730, pana la 4324580), rezultand un model mai bun.

A Appendix

A.1 Cod initial fara imbunatatiri

```
clear all
clc
load('product_23.mat')

minim=10^10;
TethaMinim=[];
MMinim=0;

%afisam datele initiale din product_23
plot(time, y,'k'), title('Date de intrare - product23'), xlabel('t'),
ylabel('y'), grid
figure
% utilizam o bucla "for" pentru a calcula tetha si eroarea pentru fiecare
valoare a lui M
for M=1:12

    %selectam 80% din datele initiale
    NumberT=round(length(time)*80/100);

    %formam matricea PHI = Xf pentru primele 86 de date (80%)
    Xf=[];
    %k cuprinde indecsii pentru datele de identificare
    for k=1:86
        X1=[1 k]; %formam componenta liniara
        for m=1:M
            X1 = [X1 cos(2*pi*m*k/12) sin(2*pi*m*k/12)];% alipim componenta armonica
        end
        Xf = [Xf;X1];%alipim linia formata si trecem la linia urmatoarea pentru
        urmatorul k

    end
    %phi
    Xf;

    %date identificare
    for i=1:86
        y1(i)=y(i);
    end
    %aflam tetha
    Tetha = linsolve(Xf,y1');
    %date validare
    for i=1:23
        y2(i)=y(i+85);
    end

    %afisam functia aproximata pentru ultima valoare a lui Tetha folosind 80%
    din datele initiale si comparam cu datele de identificare
    plot(1:86,f(1:86,M,Tetha),'r',1:86,y(1:86),'k'),title('Identificare'),
    legend('Functia aproximata', '80% din datele initiale'), xlabel('t'),
    ylabel('y'), grid calculam eroarea pentru datele de identificare si o
    memoram intr-un vector, MSE_identificare
    e = 0;
```

```
    for i = 1:86
        e = e + ( y(i)-f(i,M,Tetha) )^2;
    end


    MSE_id = e/(length(1:86));
    MSE_identificare(M)=MSE_id;
    %calculam eroarea pentru datele de validare si o memoram intr-un
    %vector, MSE_validare
    e = 0;
    for i = 86:108
        e = e + ( y(i)-f(i,M,Tetha) )^2;
    end

    k=86:108;
    MSE_val = e/(length(k));
    MSE_validare(M)=MSE_val;

    %aflam M minim pentru a afisa graficul pentru acel M, memoram si Tetha
    %pentru M minim
    if minim>MSE_validare(M)
        minim=MSE_validare(M);
        MMinim=M;
        TethaMinim= Tetha;
    end

end
%afisam functia aproximata pentru M minim folosind 80% din datele initiale
si comparam cu datele de identificare
figure
plot(1:86,f(1:86,MMinim,TethaMinim),'r',1:86,y(1:86),'k'),title('Identificare'),
legend('Functia aproximata pentru M MINIM', '80% din datele initiale'),
xlabel('t'), ylabel('y'), grid
%afisam functia aproximata pentru  M minim folosind 20% din datele initiale si
comparam cu datele de validare
figure
plot(87:108,f(87:108,MMinim,TethaMinim),'r',87:108,y(87:108),'k'),title('Validare'),
legend('Functia aproximata', '20% din datele initiale'), xlabel('t'),
ylabel('y'), grid
%afisam vectorul de erori atat pentru identificare, cat si pentru validare
figure
subplot(2,1,1),plot(1:M,MSE_identificare,'m','LineWidth',1),
title('MSE in functie de M - identificare')
,legend("MSE identificare")
, xlabel('M'), ylabel('MSE'), grid
subplot(2,1,2),plot(1:M,MSE_validare, 'm','LineWidth',1),
title('MSE in functie de M - validare')
,legend("MSE validare")
, xlabel('M'), ylabel('MSE'), grid

%afisare MSE minim pt identificare, validare si M minim
minim_identificare = min(MSE_identificare)
minim_validare = min(MSE_validare)
MMinim
```

```
%~~~~~Functie pentru functia aproximata yhead~~~~~


function yhead = f(time, m, tetha)
yhead=tetha(1)+tetha(2)*time;%construim partea liniara
for i=1:m
    yhead=yhead+cos(2*pi*i*time/12)*tetha(2*i+1)+sin(2*pi*i*time/12)*tetha(2*i+2);
    %adaugam partea armonica

end
end
```

A.2 Cod cu imbunatatiri

```
clear all
clc
%load('product_23.mat')
load('product_23.mat')
figure
plot(1:86,y(1:86),'m',86:108,y(86:108),'k'),title("Date initiale"),legend("Date identific
minim=10^10;
TethaMinim=[];
MMinim=0;
%daca nap>5, intra in supraantrenare
%daca nap>8, crapa programul
%--------PENTRU A RESPECTA CERINTELE INITIALE, AVEM NEVOIE DE NAP=2--------
nap = 5;%nap=p => polinom de grad p-1
%afisam datele initiale din product_23
plot(time, y,'k'), title('Date de intrare - product23'), xlabel('t'),
ylabel('y'), grid
figure
% utilizam o bucla for pentru a calcula tetha si eroarea pentru fiecare
valoare a lui M
for M=1:8

    %selectam 80% din datele initiale
    NumberT=round(length(time)*80/100);

    %formam matricea PHI = Xf pentru primele 86 de date (80%)
    Xf=[];
    %k cuprinde indecsii pentru datele de identificare
    for k=1:86
        X1=[ MP(k,nap)];%formam componenta liniara, un polinom de grad nap-1
        for m=1:M
            X1 = [X1 cos(2*pi*m*k/12) sin(2*pi*m*k/12)];% alipim componenta armonica
        end
        Xf = [Xf;X1];%alipim linia formata si trecem la urmatoarea linie pentru
        %urmatorul k

    end
    %phi
    Xf;

    %date identificare
    for i=1:86
```

```
        y1(i)=y(i);
    end
    %aflam tetha
    Tetha = linsolve(Xf,y1');
    %date validare
    for i=1:23
        y2(i)=y(i+85);
    end

    %afisam functia aproximata pentru ultima valoare a lui Tetha folosind 80%
    %din datele initiale si comparam cu
    %datele de identificare
    plot(1:86,f(1:86,M,Tetha,nap),'r',1:86,y(1:86),'k'),title('Identificare'),
    legend('Functia aproximata', '80% din datele initiale'), xlabel('t'),
    ylabel('y'), grid
    %calculam eroarea pentru datele de identificare si o memoram intr-un
    %vector, MSE_identificare
    e = 0;
    for i = 1:86
        e = e + ( y(i)-f(i,M,Tetha,nap) )^2;
    end


    MSE_id = e/(length(1:86));
    MSE_identificare(M)=MSE_id;
    %calculam eroarea pentru datele de validare si o memoram intr-un
    %vector, MSE_validare
    e = 0;
    for i = 86:108
        e = e + ( y(i)-f(i,M,Tetha,nap) )^2;
    end

    k=86:108;

    MSE_val = e/(length(k));
    MSE_validare(M)=MSE_val;

    %aflam M minim pentru a afisa graficul pentru acel M, memoram si Tetha
    %pentru M minim
    if minim>MSE_validare(M)
        minim=MSE_validare(M);
        MMinim=M;
        TethaMinim= Tetha;
    end

end
%afisam functia aproximata pentru M minim folosind 80% din datele initiale si
%comparam cu datele de
%identificar
figure
plot(1:86,f(1:86,MMinim,TethaMinim,nap),'r',1:86,y(1:86),'k'),title('Identificare'),
legend('Functia aproximata pentru M MINIM', '80% din datele initiale'),
xlabel('t'), ylabel('y'), grid
%afisam functia aproximata pentru ultimul M, folosind 20% din datele
%initiale si comparam cu datele de validare
figure
```

```
plot(87:108,f(87:108,M,Tetha,nap),'r',87:108,y(87:108),'k'),
title('Validare pt ultimul M'),
legend('Functia aproximata', '20% din datele initiale'), xlabel('t'),
ylabel('y'),
grid
%afisam vectorul de erori atat pentru identificare, cat si pentru validare
%afisam functia aproximata pentru  M minim folosind 20% din datele initiale si
%comparam cu datele de validare
figure
plot(87:108,f(87:108,MMinim,TethaMinim,nap),'r',87:108,y(87:108),'k'),
title('Validare; M minim'),
legend('Functia aproximata', '20% din datele initiale'), xlabel('t'),
ylabel('y'), grid
figure
subplot(2,1,1),plot(1:M,MSE_identificare,'m','LineWidth',1),
title('MSE in functie de M - identificare'), xlabel('M'), ylabel('MSE'), grid
subplot(2,1,2),plot(1:M,MSE_validare, 'm','LineWidth',1),
title('MSE in functie de M - validare'), xlabel('M'), ylabel('MSE'), grid

%afisare MSE minim pt identificare, validare si M minim

minim_identificare = min(MSE_identificare)
minim_validare = min(MSE_validare)
MMinim



%~~~~~~~~~~~~~~~~Functie pentru crearea polinomului~~~~~~~~~~~~~~

function o = MP(k,n)
o=[];
for i=1:n
    o=[o k^(i-1)];%construim polinomul de grad n-1
end
end

%~~~~~~~~~~~~~~~~Functie pentru functia aproximata yhead~~~~~~~~~~~~~~
function yhead = f(time, m, tetha,n)
yhead=0;

for i=1:n

    yhead=yhead+tetha(i)*time.^(i-1);%construim componenta liniara

end
for i=1:m

    yhead=yhead+cos(2*pi*i*time/12)*tetha(2*i+n-1)+sin(2*pi*i*time/12)*tetha(2*i+n);
    %adaugam componenta armonica

end
end
```