Anexos

```
clear all;
close all;
format long

%% Parametros
pinta1=0;    % flag grafica errores, coste
pinta2=0;    % flag grafica pasos algoritmo
pinta3=0;    % flag grafica phi xi
pinta4=1;    % flag grafica nmse
ipinta=20;  % numero de iteraciones entre refresco de gráficas

T=1000;     % ventana de señal con la que se hará el cálculo
P = 5;                    % 2P-1 nonlinearity degree of the Volterra model
rank = 2;                 % tensors rank
memory = 0;               % memory Q

% step size for the algorithm
mu0=1e-5; % 1e-5 ajuste muy fino. 1e-2 normal
c = 1;

%% Definicion de variables
r = rank*ones(1,P);           % rango para cada orden:   r(ip)
m = memory*ones(1,P);         % memoria para cada orden: m(ip)
a = 0.01*(1+i)*ones(P,memory+1,rank);   % coeficientes a: a (2P-1, m, r)
b = 0.01*(1+i)*ones(P,memory+1,rank);   % También se puede inicializar a eps.
y = zeros(1,T);               % señal de salida

gamma = zeros(rank,T,P,memory+1);
delta = zeros(rank,T,P,memory+1);

v = zeros(P,rank,memory+1);
w = zeros(P,rank,memory+1);

phi = zeros(P,T,memory+1,rank);
xi = zeros(P,T,memory+1,rank);

mu_t = zeros(P,T);
mu_w = zeros(P,T);

alpha = zeros(P,rank,memory+1);
beta = zeros(P,rank,memory+1);
```

```
J=zeros(1,T); J(1)=1;       % coste
mmse=zeros(1,T);
nmse=zeros(1,T);

%% Carga de señales
load xmed
load ymed

x=(xsw(1:T,7));
d=(ymed(1:T,7));

%% prueba con señal sintetica
% load sintetico.mat;
% x = x.';
% d = d.';

[h, ymod, nmse] = valor_inicial(d,x,memory,2*P-1,0);

%% Traduccion de h a a.
%   abs(a) = abs(h).^(1/(2p-1))
%   angle(a) = angle(h)

for ip=1:P
    for im=1:m(ip)+1
        for ir=1:1
            a (ip, im, ir) = abs(h(ip,im)).^(1/(2*ip-1)).*exp(i*angle(h(ip,im)));
        end
    end
end

b = conj(a);
theta = a;
omega = b;

%% variables auxiliares
paso=0;
s=1;
datonmse2=zeros(1,length(xsw(:,7))-1); %vector de valores nmse
```

```
%% Inicio del algoritmo
for shift=1:fix(length(xsw(:,7))/T)
    paso=shift;

    xaux=xsw((shift-1)*T+1:(shift)*T+memory,7).';
    d=ymed((shift-1)*T+1:(shift)*T,7).';
    % x(n-i) = x(i-1,:) -- x(n) = x(1,:) -- x(n-1) = x(2,:)
    x=toeplitz([xaux(1);zeros(memory,1)],xaux);

    x = xaux((repmat(memory-[0:memory], T, 1)+repmat((1:T)', 1, memory+1)).');
    d=ymed((shift-1)*T+1+memory:(shift)*T+memory,7).';

    for i=1:T
         paso
        % Calculo de las para cada rango
        y_aux = zeros(T,P,rank,memory+1);
        for ip=1:P            %lo calculamos para todos los grados del modelo
            for ir=1:r(ip)      %para todo los rangos
                for im=1:m(ip)+1
                        gamma(ir,:,ip,im) = a(ip, im, ir) * x(im,:);
                        delta(ir,:,ip,im) = b(ip, im, ir) * conj(x(im,:));
            y_aux(:,ip,ir,im) = ( ( gamma(ir,:,ip,im) ).^ip ).*(( delta(ir,:,ip,im) ).^(ip-1));

                    alpha(ip,ir,im) = (gamma(ir,i,ip,im))^(ip-1) .* (delta(ir,i,ip,im))^(ip-1);
                    beta(ip,ir,im)  = (gamma(ir,i,ip,im))^(ip)   .* (delta(ir,i,ip,im))^(ip-2);
                end
            end
        end

        % Calculo de la señal completa
        y = zeros(1,T);
        %parfor in=1:T
        for in=1:T
            for ip=1:P            %lo calculamos para todos los grados del modelo
                for im=1:m(ip)+1
                    for ir=1:r(ip)      %para todo los rangos
                        y(in) =y(in) + y_aux(in,ip,ir,im);
                    end
                end
            end
        end



        % Calculo del error, coste
        e=d-y;
```

```
J(i+1)=0.99*J(i)+0.01*abs(e(i)).^2;
mmse(i)=sum(abs(e).^2)/i;
nmse(i) = 20*log10(norm(y-d)/norm(d));
datonmse2(1,s)=nmse(i);
s=s+1;



% suavizamos el error para evitar muestras malas
e = smooth(e);

v = alpha;
w = beta;

for ip=1:P
    for im=1:m(ip)+1

            aux1 =  repmat([1:P]',1,r(ip)) .* kron ( v(:,1:r(ip),im) , x(im,i) );
            phi(:,i,im,:) = aux1;
            aux2 = repmat([1:P]',1,r(ip)) .* kron ( w(:,1:r(ip),im) , conj(x(im,i)) );
            xi(:,i,im,:)  = aux2;

    end
end

for ip=1:P
    for im=1:m(ip)+1

        mu_t(ip,i,im) = mu0/(c+norm(phi(ip,i,im))^2);
        mu_w(ip,i,im) = mu0/(c+norm(xi(ip,i,im))^2);

        for ir=1:r(ip)
     theta(ip, im, ir) = theta(ip,im, ir) + mu_t(ip,i,im) * conj (phi(ip,i,im)) * e(i);
         omega(ip, im, ir) = omega(ip,im, ir) + mu_w(ip,i,im) * ( xi(ip,i,im)) * e(i);
        end
    end
end

a = theta;
b = omega;




 % Presentacion de variables en tiempo de ejecucion
%disp(['mu_t:']);disp([mu_t(:,ip)'])
```

```
        %disp(['mu_w:']);disp([mu_w(:,ip)'])
        disp(['a:']);disp([a])
        disp(['b:']);disp([b])

        if pinta1==1 && fix(i/ipinta)==i/ipinta
            figure(1)
            subplot(4,3,1);
            plot(x(1:i),'.k'); title('linear: x'); grid;
            subplot(4,3,4);
            plot(d(1:i),'.b'); title('non-linear: y'); grid; axis([-50 50 -50 50]);
            subplot(4,3,7);
            plot(y(1:i),'.g'); title('estimate: y'); grid;
            subplot(4,3,10);
            plot(e(1:i),'.r'); title('error: e'); grid; axis([-50 50 -50 50]);
            subplot(3,3,2:3)
            plot(1:i,J(1:i)); title('Cost J = |e(k)|^2'); grid;
            subplot(3,3,5:6)
            semilogy(mmse); title('MSE'); grid;
            subplot(3,3,8:9)
            plot(nmse(1:i-1)); title('Normalized MSE (dB)'); grid;
        end

        if pinta2==1 && fix(i/ipinta)==i/ipinta
            figure(2)
            for ip=1:P
                subplot(2,P,ip)
                plot(mu_t(ip,:))
                axis([0 T 0 1.1*max(mu_t(ip,:))]);
                subplot(2,P,P+ip)
                plot(mu_w(ip,:))
                axis([0 T 0 1.1*max(mu_w(ip,:))]);
            end
        end

        if pinta3==1 && fix(i/ipinta)==i/ipinta
            figure(3)
            for ip=1:P
                subplot(2,P,ip)
```

```
                plot(abs(phi(ip,:)))
                subplot(2,P,P+ip)
                plot(abs((xi(ip,:))))

            end
        end

        if pinta4==1 && fix(i/ipinta)==i/ipinta
            figure(4)
            plot(nmse(1:i-1)); title('Normalized MSE (dB)'); grid;
        end
        pause(.001)

    end
end
indicador=find(datonmse2==min(datonmse2)); %indica la posición con nmse más baja
```