ANEXOS

Programa para la identificación del sistema con modelo ANFIS

```
clc;
clear;
close all;
colordef white;
load data_pH_Identificacion_new;%%%%%<<<<<Data patrones para identificación
%load parametros_1_R2_;%%%%%%%<<<<<Carga valores ya entrenados

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

In_01_Orig = F_ac; In_02_Orig = F_ba; Out_01_Orig = pH_new;%<con load data_pH_Identificacion_new
In_01_Orig_ = zeros(length(In_01_Orig),24);
In_02_Orig_ = zeros(length(In_02_Orig),24);
Out_01_Orig_ = zeros(length(Out_01_Orig),24);

for j = 1:5 %Genera In_01_Orig(k-1) hasta In_01_Orig(k-5) en cada columna
   if j == 1
   In_01_Orig_(2:end,j) = In_01_Orig(1:end-1,1);
   In_02_Orig_(2:end,j) = In_02_Orig(1:end-1,1);
   Out_01_Orig_(2:end,j) = Out_01_Orig(1:end-1,1);
   else
   In_01_Orig_(2:end,j) = In_01_Orig_(1:end-1,j-1);
   In_02_Orig_(2:end,j) = In_02_Orig_(1:end-1,j-1);
   Out_01_Orig_(2:end,j) = Out_01_Orig_(1:end-1,j-1);
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X = [In_01_Orig, In_01_Orig_(:,1), In_02_Orig, In_02_Orig_(:,1), Out_01_Orig_(:,1), Out_01_Orig_(:,2)];
%Substractive Clustering
[C,S] = subclust(X,0.3);%%%%%%%<C como centros
```

```
[filC, colC] = size(C);
A = S.*ones(filC,colC).*sqrt(2.0);%<<<A como anchos

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Out_01 = Out_01_Orig;
npatrones = length(Out_01);
kpatron = 1;
niter = 4;%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
kiter = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TrgtOut = Out_01_Orig;
ninp = colC;
ninpar = 2; %parametros de la función A y C
nout = 1;
nreglas = filC;
nparam_ = ninp + 1 ; %07 parámetros a identificar por modelo lineal: p(k1), p(k2), q(k1), q(k2), r(k1), r(k2), s(k1) , con k desde 1 hasta nreglas
n_param = (nparam_ ) * nreglas; %Cantidad de parámetros a identificar: cantidad de parámetros a identificar por neurona (4) * nro. de reglas (kX * kY * kZ)
theta = zeros(n_param,1);%%%%%%%<<<<Se define los parámetros theta de las funciones lineales Capa 4
ThetaL4_old = zeros(n_param,1);
alpha = 1e8; % Para inicializar la matriz P. Por ejemplo 10000
lambda = 0.995;%0.995; % En el caso de tener factor de olvido
P = alpha*eye(n_param);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Fx = zeros(nreglas, ninp);
mat_param_dep_consec = zeros(nreglas,ninp);
mat_param_indep_consec = zeros(nreglas,1);
Dx = zeros(nreglas,(nreglas*nparam_));

eta = 0.1;
dJ_dc = 0;
```

```
dJ_da = 0;

rmse  = zeros(niter,1);
dFx_dC = zeros(filC,colC);
dFx_dA = zeros(filC,colC);
dFx_db1 = zeros(filC,colC);
dW_dFx_aux = zeros(colC,filC);
out_anfis = zeros(npatrones,1);
step_size_p = 0.01;

eta_c = 1.*ones(filC,colC);
eta_a = 1.*ones(filC,colC);
eta_b1 = 0.005.*ones(filC,colC);
alpha2 = 0.05;%0.05;%0.95;
delta_dJ_dc = zeros(filC,colC);
delta_dJ_da = zeros(filC,colC);

J = zeros(niter,1);

for kiter = 1:niter

   e1 = zeros(npatrones,1);
   dJ_dc = 0;
   dJ_da = 0;
   dJ_db1 = 0;
   JJ = 0;
   %************************************************%
   %************************************************%
   %PASO HACIA ADELANTE:
   for kpatron = 1:npatrones

      inp = (X(kpatron,:))';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%Primera Capa%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      for i=1:colC
```

```
        Fx(:,i) =  exp(-(((inp(i,1)*ones(filC,1)-C(:,i))./A(:,i)).^2));
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%Segunda Capa%%%%%%%%%%%%%%%%%%%
    Fxx = Fx';

    ww = (prod(Fxx));
    W = ww';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%Tercera Capa%%%%%%%%%%%%%%%%%%

    Wb = W./sum(W);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%Cuarta Capa%%%%%%%%%%%%%%%%%%%%%
    m=1;
    n=1;
    for j = 1:n_param
        if (mod(j,nparam_)~= 0)
            mat_param_dep_consec(m,n) = theta(j,1); %Matriz de los parámetros dependientes de x e y del consecuente, dimension: (kX*kY*kZ)x(3)
            n = n+1;
        else
            mat_param_indep_consec ((j/nparam_),1) = theta(j,1);  %Matriz de los parámetros NO dependientes de x e y del consecuente (kX*kY*kZ)x1
            m = m+1;
            n = 1;
        end
    end

  ff = mat_param_dep_consec * inp + mat_param_indep_consec; %inp son entradas al sistema en este caso 3x1


  out4 = Wb.*ff;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%Quinta Capa%%%%%%%%%%%%%%

   out5 = Wb' * ff ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   e1(kpatron) = Out_01(kpatron,1)- out5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   h = [inp', 1];
   m = 1;
   for j = 1:nreglas
      for k = 1:nparam_ %07 parámetros a identificar por neurona: ok, pk, qk y rk, con k desde 1 hasta (kX*kY*kZ)
         Dx(j,m) = h(1,k); %Matriz consituida por x, y, z. Dimensión: [(nreglas) x (nparam por neurona x nreglas)]
         m = m+1;
      end
   end

   %XX será nuestra matriz de regresión [w1b*x w1b*y w1b w2b*x w2b*y w2b...] Dimension: 1x(nparam_ x kX*kY*kZ)
   XX = Wb' * Dx;
   XX = XX';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%Actualialización de theta%%%%%%%%%%%%%%%%%%%%
   ThetaL4_old = theta;
   K = (P)/(lambda+(XX)'*P*(XX)); % Calculamos la nueva matriz K
   theta = theta+K*(XX)*e1(kpatron); % Estimamos los nuevos parámetros
   P = (P - (P*(XX)*(XX)'*P)/(lambda + (XX)'*P*(XX))); % Calculamos la nueva matriz P

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   end
```

```
%PASO HACIA ATRAS:
for kpatron = 1:npatrones

    inp = (X(kpatron,:))';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%Primera Capa%%%%%%%%%%%%%%%%%%

    for i=1:colC
       Fx(:,i) =  exp(-(((inp(i,1)*ones(filC,1)-C(:,i))./A(:,i)).^2));
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%Segunda Capa%%%%%%%%%%%%%%%%%%
    Fxx = Fx';

    ww = (prod(Fxx));
    W = ww';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%Tercera Capa%%%%%%%%%%%%%%%%%%

    Wb = W./sum(W);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%Cuarta Capa%%%%%%%%%%%%%%%%%%
    m=1;
     n=1;
    for j = 1:n_param
       if (mod(j,nparam_)~= 0)
%Matriz de los parámetros dependientes de x e y del consecuente, dimension: (kX*kY*kZ)x(3)
          mat_param_dep_consec(m,n) = theta(j,1);
          n = n+1;
       else
%Matriz de los parámetros NO dependientes de x e y del consecuente (kX*kY*kZ)x1
```

```
            mat_param_indep_consec ((j/nparam_),1) = theta(j,1);
            m = m+1;
            n = 1;
        end
    end
%inp son entradas al sistema en este caso 3x1
  ff = mat_param_dep_consec * inp + mat_param_indep_consec;
  out4 = Wb.*ff;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%Quinta
Capa%%%%%%%%%%%%%%%%%%%%%%

  out5 = Wb' * ff ;
  out_anfis(kpatron) = out5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  e1(kpatron) = Out_01(kpatron,1)- out5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%Cálculo de Derivadas Parciales de Todas las Capas%%%%%%%%%%
  dJ_de = e1(kpatron);

  de_dout5 = -1;

  dout5_dwb = ff'; %Derivada salida capa 5 respecto a capa 4. Dimension: 1 x (kX.kY)

  dwb_dw = zeros(nreglas, nreglas); %Derivada salida capa 3 respecto a capa 2. Dimension: (kX.kY)x(kX.kY)
  for j = 1:nreglas
     for k = 1:nreglas
        if(j == k)
           dwb_dw(j,k) = (sum(ww)- ww(1,j))./(sum(ww)^2);
        else
           dwb_dw(j,k) =  -ww(1,j)./(sum(ww)^2);
```

```
        end
      end
    end


    for j = 1:colC
      Fx_aux  = Fxx;
      Fx_aux(j,:) = [];
      dW_dFx_aux(j,:) = prod(Fx_aux);
    end

    dW_dFx = dW_dFx_aux';

    for i=1:colC
           dFx_dC(:,i) = 2*Fx(:,i).*((inp(i,1)*ones(filC,1) - C(:,i)) ./ (A(:,i).^2));
           dFx_dA(:,i) = 2*Fx(:,i).*((inp(i,1)*ones(filC,1) - C(:,i)).^2 ./ (A(:,i).^3));

    end

    JJ = JJ + 0.5*e1(kpatron)'*e1(kpatron);
    dJ_dc = dJ_dc + ( (dJ_de) * (de_dout5) * (dout5_dwb) * (dwb_dw) )' .* (dW_dFx) .*
(dFx_dC);
    dJ_da = dJ_da + ( (dJ_de) * (de_dout5) * (dout5_dwb) * (dwb_dw) )' .* (dW_dFx) .*
(dFx_dA);

end

    J(kiter,1) = JJ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %Actualiza el tamaño de paso para variar la razon de aprendizaje (eta C y A)
    decrease_rate = 0.8;
    increase_rate = 1.2;

    if (kiter == 1)
      last_change =1;
```

```
    end

    if (kiter - last_change < 4)

    elseif (J(kiter) < J(kiter-1)) && (J(kiter-1) > J(kiter-2)) && (J(kiter-2) < J(kiter-3))...
            && (J(kiter-3) > J(kiter-4))

            step_size_p = step_size_p * decrease_rate;
            last_change = kiter;

    elseif (J(kiter) < J(kiter-1)) && (J(kiter-1) < J(kiter-2)) &&(J(kiter-2) < J(kiter-3))...
            && (J(kiter-3) < J(kiter-4))

            step_size_p = step_size_p * increase_rate;
            last_change = kiter;

    else

    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    eta_c = step_size_p./(sqrt(((dJ_dc)).^2));
    eta_a = step_size_p./(sqrt(((dJ_da)).^2));

    delta_dJ_dc = eta_c .* (dJ_dc)/(npatrones) - alpha2 * delta_dJ_dc;%%%%%%%
    delta_dJ_da = eta_a .* (dJ_da)/(npatrones) - alpha2 * delta_dJ_da;%%%%%%%
    C = C - delta_dJ_dc;%%%%%%%Actualización de centros
    A = A - delta_dJ_da;%%%%%%%Actualización de anchos


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    rmse(kiter,1) = norm(e1)/sqrt(npatrones);
    fprintf('\n RMSE: %2.4f \n', rmse(kiter,1));

    if(abs(J(kiter))<1e-2)
       break;
```

```
    end
end

figure
subplot(3,1,1),semilogy(rmse,'*r');
title('RMSE');
grid on

TT = 1:npatrones;
subplot(3,1,2)
plot(TT,TrgtOut,'b',TT,out_anfis,'g')
title('Target Output(Blue) VS Actual Output(Green)')
grid on

subplot(3,1,3),plot(e1,'b')
title('Approximation Error(n)')
grid on

figure(2)
plot(TT,In_01_Orig,'b',TT,Out_01_Orig,'r',TT,In_02_Orig,'g');

%Guarda los parámetros hallados:
save('parametros_1_R2_','A','C',
'theta','mat_param_indep_consec','mat_param_dep_consec');%<<<<<<<<<<<<<<<<<

%Calcula los índices de similitud del modelo hallado
vaf = (1-(var(Out_01_Orig - out_anfis)./var(Out_01_Orig)))*100;
rmse = rmse(kiter,1);
fit = goodnessOfFit(out_anfis, Out_01_Orig ,'NRMSE');

%%%%%%%%%Graficos Despues de Identificación%%%%%
figure(3);
plot(TT,TrgtOut,'-b',TT,out_anfis,'--r','linewidth',1)
legend('Salida Planta', 'Salida ANFIS' ,'Location','SouthEast');
axis ([15000 20000 2 14]);
title('Comparación Salida ANFIS VS Salida Real');
xlabel('iteración (k)');
```

```
ylabel('pH');

figure(4);
plot(TT,e1,'--g','linewidth',1)
legend('Error' ,'Location','SouthEast');
axis ([15000 20000 -0.5 0.5]);
title('Aproximación del error');
xlabel('iteración (k)');
ylabel('error');

fprintf('\n          VAF    |    RMSE    |    FIT    \n');
fprintf('  ------------------------------------------------------------\n');
fprintf('                          %2.4f    |    %2.4f    |        %2.4f  \n', vaf, rmse, fit);
```

Programa que desarrolla la implementación del controlador GPC–ANFIS

```
close all
clear
clc

warning('off','all')
warning
% Definición del proceso:
load parametros_1_R2_;

[filC, colC] = size(C);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ftz1 = tf(); ftz2 = tf(); ftzq = tf();
D_g = [mat_param_dep_consec(:,3) mat_param_dep_consec(:,4)]; %Perturbación
B_g = [mat_param_dep_consec(:,1) mat_param_dep_consec(:,2)];
A_g = [ones(filC,1), -1.*mat_param_dep_consec(:,5), -1.*mat_param_dep_consec(:,6)];
C_g = mat_param_indep_consec(:,1); %Constante
for j=1:filC
ftz1(j,1)=filt(B_g(j,:),A_g(j,:),1);
ftz1(j,1).iodelay = 0;
ftz2(j,1)=filt(D_g(j,:),A_g(j,:),1); %Perturbación
ftz2(j,1).iodelay = 0;
ftzq(j,1)=filt(C_g(j,:),A_g(j,:),1); %Constante
ftzq(j,1).iodelay = 0;
end

D_gpc = zeros(filC,size(D_g,2)); A_gpc = zeros(filC,size(A_g,2)); B_gpc = zeros(filC,size(B_g,2)); C_gpc = zeros(filC,size(C_g,2));
for j=1:filC
D_gpc(j,:) = D_g(j,:);
A_gpc(j,:) = A_g(j,:);
B_gpc(j,:) = B_g(j,:);
C_gpc(j,:) = C_g(j,:);
end
```

```
Pz1 = ftz1(:,1);
Pz2 = ftz2(:,1);  %FT perturbacion
Pq = ftzq(:,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dm=0; %Retardo
dq=0; %Retardo

%% Parametros de sintonia del GPC
N = zeros(filC,1); N1 = zeros(filC,1); N2 = zeros(filC,1); Nu = zeros(filC,1);
lambda = zeros(filC,1); delta = zeros(filC,1);
for j = 1 : filC
   N(j,1)=4;        %Horizonte de Prediccion (8)
   N1(j,1)=dm+1;    %Horizonte Inicial
   N2(j,1)=dm+N(j,1);    %Horizonte final
   Nu(j,1)=4;       %Horizonte de Entrada (4)
   lambda(j,1)=200;   %Parametro de ponderacion del Control(1200)
   delta(j,1)=15;    %Parametro de ponderacion del seguimiento a referencia(70)
end

Ql=cell(filC,1); Ql(:,1) = {0};
Qd=cell(filC,1); Qd(:,1) = {0};
for j = 1:filC
   Ql{j,1} = lambda(j,1)*eye(Nu(j,1));  %(lambda -> Acción de Control)
   Qd{j,1} = delta(j,1)*eye(N(j,1));      %(delta  -> Seguimiento de Referencia)
End

%% Calculo de la ecuacion Diofantica
En=cell(filC,1); En(:,1) = {0};
F=cell(filC,1); F(:,1) = {0};
E=cell(filC,1); E(:,1) = {0};

for j = 1 : filC
   [En{j,1},F{j,1}] = diophantine(A_gpc(j,:),N2(j,1),0); %Calculo de la funcion Diofantina
   E{j,1}=En{j,1}(end,:); %Polinomio E seria la ultima fila arrojada por la funcion
   F{j,1}=F{j,1}(N1(j,1):N2(j,1),1:end);
```

```
end

%% Determina los coeficientes de control pasados
%Elimino el cero de la primera posicion de B
for j = 1 : filC
   if B_gpc(j,1)==0
      B_gpc(j,1:end-1)=B_gpc(j,2:end);
   end
end
B_gpc(:,end) =[];

uG=cell(filC,1); uG(:,1) = {0};
Fq=cell(filC,1); Fq(:,1) = {0};

for j = 1 : filC
   uG{j,1}=zeros(N(j,1),N1(j,1)); %Vector de controles pasados (Pertenece a la respuesta libre)
   Fq{j,1}=zeros(N(j,1),N1(j,1)); %Vector de perturbaciones pasadas (Pertenece a la respuesta
libre)
end

for j=1:filC
   m=2;
   for i=N1(j,1):N2(j,1)
      aux=conv(En{j,1}(i,:),B_gpc(j,:));
      aux1=conv(En{j,1}(i,:),D_gpc(j,:));
      if length(aux) < N1(j,1)+m-1  %Si la longitud del auxiliar es menor que m
         aux=[aux zeros(1,(N1(j,1)+m-1)-length(aux))]; %Completar con ceros
      end
      if length(aux1) < N1(j,1)+m-1  %Si la longitud del auxiliar es menor que m
         aux1=[aux1 zeros(1,(N1(j,1)+m-1)-length(aux1))]; %Completar con ceros
      end
      uG{j,1}(m-1,1:N1(j,1))=aux(m:N1(j,1)+m-1);  %Controles pasados (B siempre comienza
en un instante atras)
      %Fq{j,1}(m-1,1:N1(j,1)+1)=aux1(m:N1(j,1)+m);  %Perturbaciones pasada (D puede
comenzar desde el instante actual)
      Fq{j,1}(m-1,1:N1(j,1))=aux1(m:N1(j,1)+m-1);
      m=m+1;
   end
```

```
end

g=cell(filC,1); g(:,1) = {0};
Lj=cell(filC,1); Lj(:,1) = {0};

for j = 1:filC
   g{j,1}=conv(E{j,1},B_gpc(j,:)); %Calcula polinomio g
   Lj{j,1}=conv(D_gpc(j,:),E{j,1}); %Calcula polinomio Lj
end

G=cell(filC,1); G(:,1) = {0};
L=cell(filC,1); L(:,1) = {0};
for j = 1:filC
G{j,1}=zeros(N(j,1),Nu(j,1));% Inicializa la matriz G
L{j,1}=zeros(N(j,1),Nu(j,1));%Crio a Matriz L (Coeficientes de perturbación)
end

for j =1:filC
   for k=1:Nu(j,1)
      G{j,1}(k:end,k)=g{j,1}(1:N(j,1)-k+1);       % Forma a matriz G
      L{j,1}(k:end,k)=Lj{j,1}(1:N(j,1)-k+1);       % Forma a matriz L
   end
end

Mn=cell(filC,1); Mn(:,1) = {0};
for j = 1:filC
   Mn{j,1}=inv(G{j,1}'*Qd{j,1}*G{j,1}+Ql{j,1})*G{j,1}'*Qd{j,1}; %Calculo de la Funcion de Costo sin Restriccion
end

K1=cell(filC,1); K1(:,1) = {0};
%Calculo del controldor K1 (Primera fila de Mn)
for j=1:filC
   K1{j,1}=Mn{j,1}(1,:);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

inc_u=cell(filC,1); inc_u(:,1) = {0};
for j = 1:filC
 inc_u{j,1}=0;
end

lduf = zeros(filC,1); luG = zeros(filC,1); lFq = zeros(filC,1);
for j = 1:filC
    lduf(j,1)=size(F{j,1},2);     %Averiguo la longitud del polinomio F
    luG(j,1)=size(uG{j,1},2);     %Longitud de uG
    lFq(j,1)=size(Fq{j,1},2);     %Longitud de Fq

end

duf=cell(filC,1); duf(:,1) = {0};
dqp=cell(filC,1); dqp(:,1) = {0};
dqf=cell(filC,1); dqf(:,1) = {0};
for j = 1:filC
duf{j,1}=zeros(1,luG(j,1));   %Vector incrementos de control pasados libre
dqp{j,1}=zeros(1,lFq(j,1));   %Delta de Q pasados (Entrada de perturbacion pasadas)
dqf{j,1}=zeros(1,Nu(j,1));    %DeltaQ futuros
end

f1=cell(filC,1); f1(:,1) = {0}; f2=cell(filC,1); f2(:,1) = {0};
f3=cell(filC,1); f3(:,1) = {0}; f4=cell(filC,1); f4(:,1) = {0};
free=cell(filC,1); free(:,1) = {0};

nit = 5000;%<<<<<<<<<<<<
u(1:nit,1) = 0;
ym = zeros(filC,1);

% Referencia
r(1:4,1) = 0; r(5:1500,1) = 7; r(1501:3400,1) = 11; r(3401:nit,1) = 8;

%Perturbacion externa
do(1:1000)= 0.8 ;do(1001:1550) = 0.8 ; do(1551:nit) = 0.8;
```

```
na=size(A_gpc(1,:),2);
nd=size(D_gpc(1,:),2);
nb=size(B_gpc(1,:),2);

%Creo los vectores que van a almacenar las acciones pasadas
y_ant=cell(filC,1); y_ant(:,1) = {0};%<<<<<<<****


for j = 1:filC
    y_ant{j,1}=zeros(1,na);%<<<<<<<****
end
%Entrada_1 (Acido):
Fa = 0.081;
Ca = 0.32;
%Entrada_2 (Base):
Fb(1:1000,1) = 0.512;
Fb(1001:2000,1) = 0.300; %0.280
Fb(2001:3000,1) = 0.512;
Fb(3001:4000,1) = 0.700; %0.720
Fb(4001:nit,1) = 0.512;

Cb = 0.05005;
V = 1;
Ka = 1.8e-05;
Kw = 1e-14;
cont = 1;
xa = 1;
xb = 0;
syms ph;
ts=0.05; %Tiempo de Muestreo

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Creo los vectores que van a almacenar las acciones pasadas
pH_ant =zeros(na,1);
u_ant=zeros(1,nb);
q_ant(1,1:nd)=Fb(1:nd,1); %Perturbacion ocasionada por la solución BASE
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pH = zeros(nit,1);
ref = zeros(nit,1);
Fx = zeros(filC, colC);
out_anfis = zeros(nit,1);
e_cp = zeros(nit,1);
k=5;

I=cell(filC,1); I(:,1) = {0}; InUmax=cell(filC,1); InUmax(:,1) = {0};
InUmin=cell(filC,1); InUmin(:,1) = {0}; a=cell(filC,1); a(:,1) = {0};
b=cell(filC,1); b(:,1) = {0}; H=cell(filC,1); H(:,1) = {0};
Fo=cell(filC,1); Fo(:,1) = {0}; x=cell(filC,1); x(:,1) = {0};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1_e =cell(filC,1); x1_e(:,1) = {0}; y1_e=cell(filC,1); y1_e(:,1) = {0};
z1_e =cell(filC,1); z1_e(:,1) = {0}; s1_e=cell(filC,1); s1_e(:,1) = {0};
fo_e =cell(filC,1); fo_e(:,1) = {0}; k_e=cell(filC,1); k_e(:,1) = {0};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fval=cell(filC,1); fval(:,1) = {0}; exitflag=cell(filC,1); exitflag(:,1) = {0};

 Triang=cell(filC,1); Triang(:,1) = {0};
 T=cell(filC,1); T(:,1) = {0};
 ub=zeros(filC,1); %Maxima señal de control
 u_max=cell(filC,1); u_max(:,1) = {0};
 u_min=cell(filC,1); u_min(:,1) = {0};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SIMULACION DEL PROCESO%%
for k=5+dm:nit

%%%%%Salida del proceso%%%%%
 %
   Fa = u(k-1,1);
   xa = xa + (1/V*(Fa*Ca - (Fa + Fb(k-1,1))*xa))*ts;
```

```
    xb = xb + (1/V*(Fb(k-1,1)*Cb - (Fa + Fb(k-1,1))*xb))*ts;


    pH_ = (-log10(vpasolve(ph^3 + (Ka+xb)*ph^2 + (Ka*(xb-xa)-Kw )*ph - Kw*Ka == 0 , ph)));
    pH_ = double(pH_(3,1)) +  0.01*randn() + do(k);
    pH(k,1) = pH_;


    e_cp(k) = r(k,1) - pH(k,1) ;
 %
%%%%%Salida ANFIS%%%%%
%
    inp = [u(k-1,1); u(k-2,1); Fb(k-1,1); Fb(k-2,1); pH(k-1,1); pH(k-2,1)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%Primera Capa%%%%%%%%%%%%%%%%%%%%
    for i=1:colC
        Fx(:,i) =  exp(-(((inp(i,1)*ones(filC,1)-C(:,i))./A(:,i)).^2));
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%Segunda Capa%%%%%%%%%%%%%%%%%%%%
    Fxx = Fx';
    ww = (prod(Fxx));
    W = ww';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%Tercera Capa%%%%%%%%%%%%%%%%%%%%
    Wb = W./sum(W);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%Cuarta Capa%%%%%%%%%%%%%%%%%%%%
    ff = mat_param_dep_consec * inp + mat_param_indep_consec; %inp son entradas al sistema en este caso 3x1
    out4 = Wb.*ff;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%Quinta Capa%%%%%%%%%%%%%%%%%%%%
    out5 = Wb' * ff ;
    out_anfis(k,1) = out5;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%Salida de los modelos hallados en ANFIS
    for j = 1:filC
        ym(j,1)=B_gpc(j,:)*u_ant' + D_gpc(j,:)*q_ant' - A_gpc(j,2:na)*pH_ant(1:na-1,1)+C_gpc(j,:);
    end

    aux = [];
    aux=pH_ant(1:na-1,1);
    pH_ant=[pH(k,1); aux];

%Actualizo vector de salidas pasadas
    %y_ant=[y(k-1) y(k-2) y(k-3) ......]
     aux = [];
    for j=1:filC
        if na==1
            %y_ant{j,1}=ym(j,1);%ff(j,1); %<<<<<<<<<Salida del modelo ANFIS
            y_ant{j,1}=pH(k,1);%ff(j,1); %<<<<<<<<<Salida del PROCESO (cambio importante)
        else
            aux=y_ant{j,1}(1:na-1);
            %y_ant{j,1}=[ym(j,1) aux];%<<<<<Salida del modelo ANFIS
            y_ant{j,1}=[pH(k,1) aux];%<<<<<Salida del PROCESO (cambio importante)
        end
    end

% Calculo de la Respuesta Libre
    for j=1:filC
         f1{j,1}=uG{j,1}*duf{j,1}';   %Controles pasados
         f2{j,1}=(pH_ant'*F{j,1}')';   %Salidas Pasadas
         f3{j,1}=Fq{j,1}*dqp{j,1}';   %Perturbaciones pasadas
         free{j,1}=f1{j,1} + f2{j,1} + f3{j,1};% + f4{j,1};
     end

     %----------- Restriccion del incremento de Control -----------------%
```

```
    for j = 1:filC
        I{j,1}=eye(Nu(j,1)); %Matriz identidad
        InUmax{j,1}=0.0003*ones(Nu(j,1),1); %Incremento maximo 0.00033
        InUmin{j,1}=-0.0003*ones(Nu(j,1),1); %Incremento minimo -0.00033

        a{j,1}=[I{j,1}; -I{j,1}];
        b{j,1}=[InUmax{j,1}; -InUmin{j,1}];
        H{j,1}=(G{j,1}'*Qd{j,1}*G{j,1}+Ql{j,1});

        Fo{j,1}=(free{j,1}-r(k,1))'*Qd{j,1}*G{j,1};%>original
        %Cálculo del control
        options = optimoptions('quadprog','Algorithm','interior-point-convex','Display','off');
        [x{j,1},fval{j,1},exitflag{j,1}] =
quadprog(H{j,1},Fo{j,1},a{j,1},b{j,1},[],[],[],[],[],options);
        inc_u{j,1}=x{j,1}(4);
    end

     %##############################
%     %---------- Restriccion en la Señal de Control –No Utilizada----------%
%       for j = 1:filC
%          Triang{j,1}=tril(ones(Nu(j,1)));
%          T{j,1}=ones(Nu(j,1),1);
%          ub(j,1)=0.10; %Maxima señal de control
%          u_max{j,1}=ub(j,1)*ones(Nu(j,1),1);
%          u_min{j,1}=0.06*ones(Nu(j,1),1);
%          a{j,1}=[Triang{j,1}; -Triang{j,1}];
%          b{j,1}=[u_max{j,1}-T{j,1}*u(k-1); T{j,1}*u(k-1)-u_min{j,1}];
%
%          H{j,1}=(G{j,1}'*Qd{j,1}*G{j,1}+Ql{j,1});
%          Fo{j,1}=(free{j,1}-r(k))'*Qd{j,1}*G{j,1};
%          %Calculo do Controle
%          options = optimset('LargeScale','off');
%                                                  [x{j,1},fval{j,1},exitflag{j,1}] =
quadprog(H{j,1},Fo{j,1},a{j,1},b{j,1},[],[],[],[],[],options);
%          inc_u{j,1}=x{j,1}(1);
%       end
        %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
%%%%******%%%%%%%
%%%Calculo la señal de control%%%

inc_u_ = cell2mat(inc_u) ;
inc_u_ = inc_u_'*Wb;%<<<<<<<<<<<<<<<

if k==1
   u(k,1)=inc_u_;
 else
   u(k,1)=u(k-1,1)+ inc_u_;
end

 if u(k,1)>=0.12
  u(k,1) = 0.12;
 end
if u(k,1)<=0.04
   u(k,1) = 0.04;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %actualiza vector de control
   %u_ant=[u(k-d) u(k-1-d) u(k-2-d) ..... ]

   aux = [];
   aux=u_ant(1:nb-1);
   u_ant=[u(k-dm) aux];

   %actualiza duf (incrementos de control)
   % duf= [du(k-1) du(k-2) ..... du(k-N)]
   aux = [];
   for j=1:filC
      aux=duf{j,1}(1:end-1);
      duf{j,1}=[inc_u{j,1} aux];
   end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Actualizo vector de perturbaciones pasadas
    %y_ant=[y(k-1) y(k-2) y(k-3) ......]
    aux = [];
    if na==1
        q_ant = Fb(k-dq,1); %Perturbacion
    else
        aux = q_ant(1:nd-1);
        q_ant = [Fb(k-dq,1) aux];%Perturbacion
    end


    %Calcula el incremento de la Perturbacion
     inc_q = Fb(k,1)-Fb(k-1,1);

    %Calculo los incrementos de la perturbacion futura
    %%%%%%%%%Cuando las perturbaciones son conocidas
%     for j = 1:filC
%         if k<=nit-length(dqf{j,1})
%             for i=1:length(dqf{j,1})
%                 dqf{j,1}(1,i)=Fb(k+i,1)-Fb(k+i-1,1);
%             end
%         end
%     end
    %%%%%%%%%
    %Actualiza los incrementos de la perturbacion pasados
     aux = [];
    for j = 1:filC
        aux=dqp{j,1}(1:end-1);
        dqp{j,1}=[inc_q aux];
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%%%%%%%%%Graficos Control del Proceso%%%%%
```

```
t = (1:length(pH))*ts;

figure(1);
plot(t,pH,'-r',t,r,'--g',t,do,'--b',t,Fb,'--m');%>original
title('pH proceso(Rojo) SetPoint(Verde) Perturbacion Externa(Azul) Perturb.Soluc.Base(Magenta)')
grid on

figure(2);
plot(t,pH,'-r', t, out_anfis,'-b');
title('pH PROCESO(Rojo), Salida ANFIS(Azul)')

figure(3);
plot(t,pH,'-r',t,r,'--g',t,Fb,'--m','linewidth',1.2);
legend('pH' ,'Setpoint', 'Perturbación Soluc. Base', 'Location','NorthEast');
axis ([0 100 0 14]);
title('Respuesta del Sistema pH - Planta Neutralización');
xlabel('tiempo (s)');
ylabel('pH');

figure(4);
plot(t,pH,'-r',t,r,'--g',t,Fb,'--m','linewidth',1.25);
legend('pH' ,'Setpoint', 'Perturbación Soluc. Base', 'Location','SouthEast');
axis ([0 250 0 14]);
title('Respuesta del Sistema pH - Planta Neutralización');
xlabel('tiempo (s)');
ylabel('pH');
grid ON;

save('Error con perturbación', 'e_cp');

warning('on','all')
warning('query','all')
```