ANEXO A. PROGRAMAS EN MATLAB

control_con_falla.m

```
%%% SIMULACION SISTEMA DE CONTROL NOMINAL SIN Y CON FALLA %%%
clear; close all; clc;
run('modelo_lineal'); % llamamos al programa 'modelo_lineal'
run('referencia'); % llamamos al programa 'referencia'

%% Limites del proceso
% nivel minimo y maximo
h_min = 0;   % cm
h_max = 20; % cm
% voltaje minimo y maximo
v_min = 0;   % V
v_max = 10; % V

%% Condiciones iniciales de los niveles de los tanques
h1i = 12.4; % cm
h2i = 12.7; % cm
h3i = 1.6;   % cm
h4i = 1.4; % cm
Yi = [h1i; h2i; h3i; h4i];

%% Parametros del controlador
load('datos_controladorRE.mat');    % Controlador por realimentacion ...
    de estados
load('datos_controladorDMC.mat');    % Controlador DMC

%% Ruido de los sensores
var_y = 0.1;     % varianza
media_y = 0;     % media
krs = 0.04;       % magnitud del ruido

%% Simulacion sin fallas
% falla actuador 1 (bomba 1)
tf1i = 0;        % tiempo de inicio (s)
tf1f = 0;        % tiempo final (s)
alfa1 = 1.0;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
uf01 = 0;        % bias (V)
% falla actuador 2 (bomba 2)
tf2i = 0;        % tiempo de inicio (s)
tf2f = 0;        % tiempo final (s)
```

```
alfa2 = 1.0;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
uf02 = 0;        % bias (V)

% elegir el controlador
disp('1: controlador por realimentacion de estados')
disp('2: controlador DMC')
disp('-------------------------------------------')
condicion = false;
while ¬condicion
    tipo = input('\ncontrolador: ');
    switch tipo
        case 1
            sim('qt_controlRE'); % simulink, control por ...
                realimentacion de estados
            condicion = true;
        case 2
            sim('qt_controlDMC'); % simulink, control DMC
            condicion = true;
        otherwise
            disp('elegir 1 o 2')
    end
end

% guardamos datos
r1 = r(:,1); r2 = r(:,2); % referencias
y1 = y(:,1); y2 = y(:,2); y3 = y(:,3); y4 = y(:,4); % salidas (con ruido)
u1 = u(:,1); u2 = u(:,2); % entradas de control

%% Simulacion con falla en la bomba 1
% falla actuador 1 (bomba 1)
tf1i = 400;      % tiempo de inicio (s)
tf1f = Tsim+1;   % tiempo final (s)
alfa1 = 0.3;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
uf01 = 0;        % bias (V)

% tipo de controlador
switch tipo
    case 1
        sim('qt_controlRE'); % simulink, control por realimentacion ...
            de estados
    case 2
        sim('qt_controlDMC'); % simulink, control DMC
end
% guardamos datos
y1f = y(:,1); y2f = y(:,2); y3f = y(:,3); y4f = y(:,4);
u1f = u(:,1); u2f = u(:,2);

%% Variables de entrada
figure;
plot(t,u1,'-r','Linewidth',1.2); hold on
plot(t,u1f,'-b','Linewidth',1.2); hold off
ylabel('Bomba 1 (V)');
```

```
xlabel('Tiempo (s)');
legend('Sin Falla','Con Falla','location','best');

figure;
plot(t,u2,'-r','Linewidth',1.2); hold on
plot(t,u2f,'-b','Linewidth',1.2); hold off
ylabel('Bomba 2 (V)');
xlabel('Tiempo (s)');
legend('Sin Falla','Con Falla','location','best');

%% Variables de salida
figure;
plot(t,r1,'-.g','Linewidth',1.2); hold on
plot(t,y1,'-r','Linewidth',1.2);
plot(t,y1f,'-b','Linewidth',1.2); hold off; grid on; grid minor
ylabel('Nivel 1 (cm)');
xlabel('Tiempo (s)');
legend('referencia','sin falla','con falla','location','best');
figure;
plot(t,r2,'-.g','Linewidth',1.2); hold on
plot(t,y2,'-r','Linewidth',1.2)
plot(t,y2f,'-b','Linewidth',1.2); hold off; grid on; grid minor
ylabel('Nivel 2 (cm)');
xlabel('Tiempo (s)');
legend('referencia','sin falla','con falla','location','best');

%% ISE sin falla
nt = length(t);
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1(i));
    Je1 = Je1 + e1^2;
end
ISE_1 = Je1;

Je2 = 0;
for i=1:nt
    e2 = (r2(i)-y2(i));
    Je2 = Je2 + e2^2;
end
ISE_2 = Je2;

%% ISE con falla
nt = length(t);
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1f(i));
    Je1 = Je1 + e1^2;
end
ISE_1f = Je1;

Je2 = 0;
```

```
for i=1:nt
    e2 = (r2(i)-y2f(i));
    Je2 = Je2 + e2^2;
end
ISE_2f = Je2;

%% Resultados
fprintf('ISE       Sin falla    Con falla \n');
fprintf('---------------------------------\n');
fprintf('Nivel 1     %4.1f       %4.1f\n',ISE_1,ISE_1f);
fprintf('Nivel 2     %4.1f       %4.1f\n',ISE_2,ISE_2f);
```

control_sin_falla.m

```
%%% SIMULACION DEL SISTEMA DE CONTROL NOMINAL (SIN FALLA) %%%
clear; close all; clc;
run('modelo_lineal'); % llamamos al programa 'modelo_lineal'
run('referencia'); % llamamos al programa 'referencia'

%% Limites del proceso
% nivel minimo y maximo
h_min = 0;   % cm
h_max = 20; % cm
% voltaje minimo y maximo
v_min = 0;   % V
v_max = 10; % V

%% Condiciones iniciales de los niveles de los tanques
h1i = 12.4; % cm
h2i = 12.7; % cm
h3i = 1.6;   % cm
h4i = 1.4; % cm
Yi = [h1i; h2i; h3i; h4i];

%% Parametros del controlador
load('datos_controladorRE.mat');     % Controlador por realimentacion ...
    de estados
load('datos_controladorDMC.mat');    % Controlador DMC

%% Ruido de los sensores
var_y = 0.1;     % varianza
media_y = 0;     % media
krs = 0.04;       % magnitud del ruido

%% Datos de la falla
% falla actuador 1 (bomba 1)
tf1i = 0;        % tiempo de inicio (s)
tf1f = 0;        % tiempo final (s)
alfa1 = 1.0;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
uf01 = 0;        % bias (V)
% falla actuador 2 (bomba 2)
tf2i = 0;        % tiempo de inicio (s)
```

```
tf2f = 0;        % tiempo final (s)
alfa2 = 1.0;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
uf02 = 0;        % bias (V)

%% Simulacion
% elegir el controlador
disp('Elegir el controlador:')
disp('  1: controlador por realimentacion de estados')
disp('  2: controlador DMC')
disp('--------------------------------------------')
condicion = false;
while ¬condicion
    tipo = input('\ncontrolador: ');
    switch tipo
        case 1
            sim('qt_controlRE'); % simulink, control por ...
                realimentacion de estados
            condicion = true;
        case 2
            sim('qt_controlDMC'); % simulink, control DMC
            condicion = true;
        otherwise
            disp('elegir 1 o 2')
    end
end

%% Guardamos datos
r1 = r(:,1); r2 = r(:,2); % referencias
u1 = u(:,1); u2 = u(:,2); % entradas de control
y1 = y(:,1); y2 = y(:,2); y3 = y(:,3); y4 = y(:,4); % salidas (con ruido)

%% Variables de entrada
figure;
subplot(2,1,1);
plot(t,u1,'-b','LineWidth',1.2);
ylabel('Bomba 1 (V)'); grid on; grid minor

subplot(2,1,2);
plot(t,u2,'-b','LineWidth',1.2);
ylabel('Bomba 2 (V)'); grid on; grid minor
xlabel('Tiempo (s)')


%% Variables de salida
figure;
subplot(2,1,1);
plot(t,r1,'-.g','LineWidth',1.5); hold on
plot(t,y1,'-b','LineWidth',1.5); hold off
ylabel('Nivel 1 (cm)'); grid on; grid minor
legend('ref','nivel','location','best')

subplot(2,1,2);
```

```
plot(t,r2,'-.g','LineWidth',1.5); hold on
plot(t,y2,'-b','LineWidth',1.5); hold off
ylabel('Nivel 2 (cm)')
xlabel('Tiempo (s)'); grid on; grid minor
legend('ref','nivel','location','best')
ylim([12 16.5])

figure;
subplot(2,1,1);
plot(t,y3,'-b','LineWidth',1.2); grid on; grid minor
ylabel('Nivel 3 (cm)')
subplot(2,1,2);
plot(t,y4,'-b','LineWidth',1.2); grid on; grid minor
ylabel('Nivel 4 (cm)')
xlabel('Tiempo (s)')
```

controlador_dmc.m

```
%%% DISENO CONTROLADOR DMC %%%
close all
run('modelo_lineal');    % llamamos al programa 'modelo_lineal'
Tsim = 800;              % tiempo de simulacion
Ts = 0.1;                % tiempo de muestreo
Yi = [12.4; 12.7; 1.6; 1.4];  % punto operacion de estado estacionario

%% Respuesta paso
usp1 = 1.0;  % paso en variable manipulada 1 (variable de desviacion)
usp2 = 1.0;  % paso en variable manipulada 2 (variable de desviacion)
% G11 y G21
du1 = usp1; du2 = 0;  % paso en variable manipulada 1
sim('respuesta_paso');  % simulink, planta (modelo no lineal)
y11 = y(:,1);  % respuesta paso y1-u1
y21 = y(:,2);  % respuesta paso y2-u1
figure(1);
subplot(2,2,1);
plot(t,y11,'-b','LineWidth',1.5);
ylabel('Nivel 1 (cm)'); xlabel('Tiempo (s)'); ylim([12 19]); grid

subplot(2,2,3);
plot(t,y21,'-b','LineWidth',1.5);
ylabel('Nivel 2 (cm)'); xlabel('Tiempo (s)'); ylim([12 20]); grid

% G12 y G22
du1 = 0; du2 = usp2;  % paso en variable manipulada 2
sim('respuesta_paso');  % simulink, planta (modelo no lineal)
y12 = y(:,1);  % respuesta paso y1-u2
y22 = y(:,2);  % respuesta paso y2-u2

subplot(2,2,2);
plot(t,y12,'-b','LineWidth',1.5);
ylabel('Nivel 1 (cm)'); xlabel('Tiempo (s)'); ylim([12 19]); grid

```

```
subplot(2,2,4);
plot(t,y22,'-b','LineWidth',1.5);
ylabel('Nivel 2 (cm)'); xlabel('Tiempo (s)'); ylim([12 20]); grid

%% Calculo de los parametros del controlador DMC
% se aproxima la respuesta de cada subsistema por un modelo de primer ...
    orden
% con retardo (FOPDT) utilizando el metodo fit 3 de (Corripio, 1997).

crit_p = 5; % criterio del tiempo de establecimiento (2 % o 5 %)
crit = crit_p/100;

%% G11 - modelo FOPDT
y11 = y11 - Yi(1);   % variable de desviacion
ys = y11(end);       % valor estacionario
ys2 = 0.632*ys;
ys1 = 0.283*ys;
s = length(t);
while y11(s)<(1+crit)*ys && y11(s)>(1-crit)*ys
    s = s - 1;
end
Tes11 = (s-1)*Ts;    % tiempo de establecimiento
s = length(t);
while y11(s)> ys2
    s = s - 1;
end
t2 = (s-1)*Ts;
s = length(t);
while y11(s)> ys1
    s = s - 1;
end
t1 = (s-1)*Ts;
tau11 = 1.5*(t2-t1);    % constante de tiempo
td11 = t2-tau11;        % retardo
if td11<0
    td11 = 0;
end
K11 = ys/usp1;          % ganancia

%% G21 - modelo FOPDT
y21 = y21 - Yi(2);   % variable de desviacion
ys = y21(end);       % valor estacionario
ys2 = 0.632*ys;
ys1 = 0.283*ys;
s = length(t);
while y21(s)<(1+crit)*ys && y21(s)>(1-crit)*ys
    s = s - 1;
end
Tes21 = (s-1)*Ts;    % tiempo de establecimiento
s = length(t);
while y21(s)> ys2
    s = s - 1;
```

```
end
t2 = (s-1)*Ts;
s = length(t);
while y21(s)> ys1
    s = s - 1;
end
t1 = (s-1)*Ts;
tau21 = 1.5*(t2-t1); % constante de tiempo
td21 = t2-tau21;     % retardo
if td21<0
    td21 = 0;
end
K21 = ys/usp1;       % ganancia

%% G12 - modelo FOPDT
y12 = y12 - Yi(1); % variable de desviacion
ys = y12(end);     % valor estacionario
ys2 = 0.632*ys;
ys1 = 0.283*ys;
s = length(t);
while y12(s)<(1+crit)*ys && y12(s)>(1-crit)*ys
    s = s - 1;
end
Tes12 = (s-1)*Ts;    % tiempo de establecimiento
s = length(t);
while y12(s)> ys2
    s = s - 1;
end
t2 = (s-1)*Ts;
s = length(t);
while y12(s)> ys1
    s = s - 1;
end
t1 = (s-1)*Ts;
tau12 = 1.5*(t2-t1); % constante de tiempo
td12 = t2-tau12;     % retardo
if td12<0
    td12 = 0;
end
K12 = ys/usp2;       % ganancia

%% G22  - modelo FOPDT
y22 = y22 - Yi(2); % variable de desviacion
ys = y22(end);     % valor estacionario
ys2 = 0.632*ys;
ys1 = 0.283*ys;
s = length(t);
while y22(s)<(1+crit)*ys && y22(s)>(1-crit)*ys
    s = s - 1;
end
Tes22 = (s-1)*Ts;   % tiempo de establecimiento
s = length(t);
```

```
while y22(s)> ys2
    s = s - 1;
end
t2 = (s-1)*Ts;
s = length(t);
while y22(s)> ys1
    s = s - 1;
end
t1 = (s-1)*Ts;
tau22 = 1.5*(t2-t1); % constante de tiempo
td22 = t2-tau22;     % retardo
if td22<0
    td22 = 0;
end
K22 = ys/usp2;       % ganancia

%% Tiempo de muestreo del DMC (Tsc)
% igual a 1/10 de la menor constante de tiempo o 1/2 del tiempo de ...
    retardo
tauv = [tau11; tau12; tau21; tau22]; % cte tiempo de cada subsistema
tdv = [td11; td12; td21; td22]; % retardo de cada subsistema
Kv = [K11; K12; K21; K22]; % ganancia de cada subsistema

Tsc_cal = floor(min(max(0.1*tauv,0.5*tdv))); % calculado
Tsc = 4; % elegido

%% Horizonte de prediccion
for i=1:length(tdv)
    kij(i,1) = ceil(tdv(i)/Tsc)+1;
end
P = ceil(max(tauv/Tsc+kij));

%% Horizonte de control (probar varios valores)
M = 2;

%% Coeficientes de la respuesta paso
nt = length(t);
nsc = floor(Tsc/Ts);

% G11
N11 = P + ceil(Tes11/Tsc); % horizonte del modelo (en muestras)
NN11 = N11*nsc; % horizonte del modelo (en tiempo)
k = 1;
for i=nsc:nsc:NN11
    g11(k,1) = y11(i,1);
    tk11(k,1) = k;
    k = k + 1;
end
figure(2); subplot(2,2,1);
plot(tk11,g11,'bo');
ylabel('Coef. G_{11}'); xlabel('muestras')

```

```
% G21
N21 = P + ceil(Tes21/Tsc); % horizonte del modelo (en muestras)
NN21 = N21*nsc; % horizonte del modelo (en tiempo)
k = 1;
for i=nsc:nsc:NN21
    g21(k,1) = y21(i,1);
    tk21(k,1) = k;
    k = k + 1;
end
figure(2); subplot(2,2,3)
plot(tk21,g21,'bo');
ylabel('Coef. G_{21}'); xlabel('muestras')

% G12
N12 = P + ceil(Tes12/Tsc); % horizonte del modelo (en muestras)
NN12 = N12*nsc; % horizonte del modelo (en tiempo)
k = 1;
for i=nsc:nsc:NN12
    g12(k,1) = y12(i,1);
    tk12(k,1) = k;
    k = k + 1;
end
figure(2); subplot(2,2,2);
plot(tk12,g12,'bo');
ylabel('Coef. G_{12}'); xlabel('muestras')

% G22
N22 = P + ceil(Tes22/Tsc); % horizonte del modelo (en muestras)
NN22 = N22*nsc; % horizonte del modelo (en tiempo)
k = 1;
for i=nsc:nsc:NN22
    g22(k,1) = y22(i,1);
    tk22(k,1) = k;
    k = k + 1;
end
figure(2); subplot(2,2,4)
plot(tk22,g22,'bo');
ylabel('Coef. G_{22}'); xlabel('muestras')

%% Matriz dinamica de cada subsistema
G11 = matriz_dinamica(g11,P,M);
G12 = matriz_dinamica(g12,P,M);
G21 = matriz_dinamica(g21,P,M);
G22 = matriz_dinamica(g22,P,M);
G = [G11 G12; G21 G22];
r = 2; % numero de entradas
m = 2; % numero de salidas

%% Ganancia del controlador
mP = m*P;
rM = r*M;
R = 1*eye(mP);  % pesos en errores futuros
```

```
Q = 1*eye(rM);   % pesos en la senal de control
beta1 = 1.4;     % pesos en la senal de control 1
beta2 = 1.2;     % pesos en la senal de control 2
Q(1:M,:) = beta1*Q(1:M,:);
Q(M+1:rM,:) = beta2*Q(M+1:rM,:);
Kc = (G'*R*G+Q) \ (G'*R);
teta = zeros(1,M); teta(1) = 1;
L = zeros(r,rM);
for i=1:r
     L(i,1+(i-1)*M:i*M) = teta;
end
Kc1 = L*Kc;

%% Respuesta libre
% vector con las gk+i - gi
dg11 = respuesta_libre(g11,P,N11);
dg12 = respuesta_libre(g12,P,N12);
dg21 = respuesta_libre(g21,P,N21);
dg22 = respuesta_libre(g22,P,N22);

alf1 = 0.0; % peso en la trayectoria de referencia 1
alf2 = 0.0; % peso en la trayectoria de referencia 2

% guardamos datos
save datos_controladorDMC.mat Kc1 dg11 dg12 dg21 dg22 P M N11 N12...
    N21 N22 Tsc alf1 alf2 beta1 beta2
```

controlador_realimentacion.m

```
%%% DISENO CONTROLADOR BASADO EN REALIMENTACION DE ESTADOS %%%
run('modelo_lineal'); % llamamos al programa 'modelo_lineal'

%% Sistema aumentado (con accion integral)
% z(k+1) = z(k)+ Ts*(yr(k)-C1*x(k); % accion integral
Aid = [ Ad zeros(4,2);-Ts*C1 eye(2)];
Bid = [ Bd;zeros(2,2)];
Cid = [ C1 zeros(2,2)];
Did = Dd;
Wid = [zeros(4,2);Ts*eye(2)];

%% Controlabilidad
Co = [Bid Aid*Bid  Aid^2*Bid Aid^3*Bid Aid^4*Bid Aid^5*Bid];
rango_Co = rank(Co); % debe ser igual a 6 (orden del sistema)

%% Galculo de la ganancia del controlador (por ubicacion de polos)
% u(k) = -Kx*x(k) - Kz*z(k)
C1 = [1 0 0 0; 0 1 0 0];     % variables controladas;
D1 = zeros(2,2);
sys_c = ss (A,B,C1,D1);      % modelo continuo
pzmap(sys_c);                % grafica de polos y ceros (lazo abierto)

% Eleccion de polos en lazo cerrado, reglas):
```

```
%   1) los polos de la planta deben ser amortiguados, deben estar a la
%      izquierda de s1/Tes
polos = eig(A);  % polos de la planta (lazo abierto)
s1 = -4.62;      % 1er Polinomio de Bessel
Tes = 30;        % tiempo de establecimiento deseado (s)
s1n = s1/Tes;    % no cumple con 1)

%   2) ceros estables de la planta, para esto deben tener parte real ...
    negativa
%      y estar a la derecha de 4*s1/Tes
ceros = tzero(sys_c);    % ceros de la planta
cero1 = ceros(1);   cero2 = ceros(2);
four_s1n = 4*s1n;        % cumple con 2)

j = sqrt(-1);
s4_p1 = -4.0156 + 1*j*5.0723;    % 4to Polinomio de Bessel
s4_p2 = -5.5281 + 1*j*1.6553;

s4_p1 = -4.0156 + 0.1*j*5.0723; % se disminuye la parte imaginaria ...
    para reducir el sobreimpulso
s4_p2 = -5.5281 + 0.1*j*1.6553;

s4 = [s4_p1 conj(s4_p1) s4_p2 conj(s4_p2)]';  % polos elegidos

spoles = [cero1; cero2; s4/Tes];      % polos en tiempo continuo
zpoles = exp(spoles*Ts);              % polos en tiempo discreto

K = place(Aid,Bid,zpoles);            % ganancia del controlador
Kx = K(1:2,1:4);
Kz = K(1:2,5:6);

eigAcl = abs(eig(Aid - Bid*K));       % eigenvalores en lazo cerrado
%   para ser estable deben estar dentro del circulo unitario)

save datos_controladorRE Kx Kz;          % guardamos los datos
```

controlador_realimentacion.m

```
%%% DISENO CONTROLADOR BASADO EN REALIMENTACION DE ESTADOS %%%
run('modelo_lineal');  % llamamos al programa 'modelo_lineal'

%% Sistema aumentado (con accion integral)
% z(k+1) = z(k)+ Ts*(yr(k)-C1*x(k);  % accion integral
Aid = [ Ad zeros(4,2);-Ts*C1 eye(2)];
Bid = [ Bd;zeros(2,2)];
Cid = [ C1 zeros(2,2)];
Did = Dd;
Wid = [zeros(4,2);Ts*eye(2)];

%% Controlabilidad
Co = [Bid Aid*Bid  Aid^2*Bid Aid^3*Bid Aid^4*Bid Aid^5*Bid];
rango_Co = rank(Co);  % debe ser igual a 6 (orden del sistema)
```

```
15
16 %% Galculo de la ganancia del controlador (por ubicacion de polos)
17 % u(k) = -Kx*x(k) - Kz*z(k)
18 C1 = [1 0 0 0; 0 1 0 0];     % variables controladas;
19 D1 = zeros(2,2);
20 sys_c = ss (A,B,C1,D1);      % modelo continuo
21 pzmap(sys_c);                % grafica de polos y ceros (lazo abierto)
22
23 % Eleccion de polos en lazo cerrado, reglas):
24 %   1) los polos de la planta deben ser amortiguados, deben estar a la
25 %      izquierda de s1/Tes
26 polos = eig(A); % polos de la planta (lazo abierto)
27 s1 = -4.62;      % 1er Polinomio de Bessel
28 Tes = 30;        % tiempo de establecimiento deseado (s)
29 s1n = s1/Tes;    % no cumple con 1)
30
31 %   2) ceros estables de la planta, para esto deben tener parte real ...
       negativa
32 %      y estar a la derecha de 4*s1/Tes
33 ceros = tzero(sys_c);    % ceros de la planta
34 cero1 = ceros(1);   cero2 = ceros(2);
35 four_s1n = 4*s1n;        % cumple con 2)
36
37 j = sqrt(-1);
38 s4_p1 = -4.0156 + 1*j*5.0723;    % 4to Polinomio de Bessel
39 s4_p2 = -5.5281 + 1*j*1.6553;
40
41 s4_p1 = -4.0156 + 0.1*j*5.0723; % se disminuye la parte imaginaria ...
       para reducir el sobreimpulso
42 s4_p2 = -5.5281 + 0.1*j*1.6553;
43
44 s4 = [s4_p1 conj(s4_p1) s4_p2 conj(s4_p2)]'; % polos elegidos
45
46 spoles = [cero1; cero2; s4/Tes];      % polos en tiempo continuo
47 zpoles = exp(spoles*Ts);               % polos en tiempo discreto
48
49 K = place(Aid,Bid,zpoles);             % ganancia del controlador
50 Kx = K(1:2,1:4);
51 Kz = K(1:2,5:6);
52
53 eigAcl = abs(eig(Aid - Bid*K));       % eigenvalores en lazo cerrado
54 %   para ser estable deben estar dentro del circulo unitario)
55
56 save datos_controladorRE Kx Kz;          % guardamos los datos
```

estado_estacionario.m

```
1 %% Calculo del punto de operacion en estado estacionario
2 clear; close all; clc
3 run('modelo_lineal')
4 %% Elegir punto de operacion en estado estacionario para h1 y h2
5 h1o = 12.4; % cm
```

```
h2o = 12.7; % cm

a11 = g1*k1/A1;
a12 = (1-g2)*k2/A3;
a21 = (1-g1)*k1/A4;
a22 = g2*k2/A2;
A = [a11 a12; a21 a22];

b1 = a1/A1*sqrt(2*g*h1o);
b2 = a2/A2*sqrt(2*g*h2o);
b = [b1; b2];

%% Calculo de las entradas en estado estacionario
vo = A\b;
v1o = vo(1); % V
v2o = vo(2); % V

%% Calculo de las otras salidas en estado estacionario
h3o = ((1-g2)*k2*v2o/a3)^2/(2*g); % cm
h4o = ((1-g1)*k1*v1o/a4)^2/(2*g); % cm
```

FDI_con_falla.m

```
%%% SIMULACION DEL SISTEMA FDI CON FALLA %%%
clear; close all; clc;
run('modelo_lineal');    % llamamos al programa 'modelo_lineal'
run('referencia');       % llamamos al programa 'referencia'

%% Limites del proceso
% nivel minimo y maximo
h_min = 0;   % cm
h_max = 20;  % cm
% voltaje minimo y maximo
v_min = 0;   % V
v_max = 10;  % V

%% Condiciones iniciales de los niveles de los tanques
h1i = 12.4; % cm
h2i = 12.7; % cm
h3i = 1.6;  % cm
h4i = 1.4;  % cm
Yi = [h1i; h2i; h3i; h4i];

%% Parametros del controlador
load('datos_controladorRE.mat');     % Controlador por realimentacion ...
    de estados
load('datos_controladorDMC.mat');    % Controlador DMC

%% Ruido de los sensores
var_y = 0.1;     % varianza
media_y = 0;     % media
krs = 0.04;       % magnitud del ruido
```

```
29
30 %% Datos de la falla
31 % falla actuador 1 (bomba 1)
32 tf1i = 400;      % tiempo de inicio (s)
33 tf1f = Tsim+1;   % tiempo final
34 alfa1 = 0.3;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
35 uf01 = 0;        % bias (V)
36 % falla actuador 2 (bomba 2)
37 tf2i = 0;        % tiempo de inicio (s)
38 tf2f = 0;        % tiempo final (s)
39 alfa2 = 1.0;     % perdidad de efectividad (0-1) (1: normal, 0: bloqueo)
40 uf02 = 0;        % bias (V)
41
42 %% Datos del observador UIO
43 load('datos_UIO.mat');  % matrices del UIO
44 wo = [-1.0; 1.0; 0.3 ; -0.4]; % condiciones iniciales UIO (variables ...
       de desviacion)
45
46 %% Datos de la evaluacion de residuos
47 umb1 = 0.0315;           % umbral 1
48 umb2 = 0.0298;           % umbral 2
49 Neval = 40;              % ventana de muestras para evaluacion de residuos
50 save N_eval Neval        % guardamos datos
51 Numb = 10;               % contador auxiliar para detectar fallas
52 save N_umb Numb          % guardamos datos
53
54 %% Simulacion FDI
55 % elegir el controlador
56 disp('Elegir el controlador:')
57 disp('1: controlador por realimentacion de estados')
58 disp('2: controlador DMC')
59 disp('-------------------------------------------')
60 condicion = false;
61 while ¬condicion
62     tipo = input('\ncontrolador: ');
63     switch tipo
64         case 1
65             sim('qt_fdi_confalla'); % simulink, controlador por ...
                   realimentacion de estados
66             condicion = true;
67         case 2
68             sim('qt_fdi_confalla_DMC'); % simulink, controlador DMC
69             condicion = true;
70         otherwise
71             disp('elegir 1 o 2')
72     end
73 end
74 % guardamos datos
75 y1 = y(:,1); y2 = y(:,2); y3 = y(:,3); y4 = y(:,4); % salidas (con ruido)
76 u1 = u(:,1); u2 = u(:,2); % entradas de control
77 y11 = xh1(:,1); y21 = xh1(:,2); y31 = xh1(:,3); y41 = xh1(:,4); % UIO 1
78 y12 = xh2(:,1); y22 = xh2(:,2); y32 = xh2(:,3); y42 = xh2(:,4); % UIO 2
```

```
79  r11 = y1-y11; r21 = y2-y21; r31 = y3-y31; r41 = y4-y41; % residuos ...
        UIO 1
80  r12 = y1-y12; r22 = y2-y22; r32 = y3-y32; r42 = y4-y42; % residuos ...
        UIO 2
81
82  %% Evaluacion de residuos
83  % tiempo de deteccion de la falla
84  nt = length(t);
85  flag1 = 1;
86  flag2 = 1;
87  disp(' ')
88  disp('Evaluacion de residuos')
89  disp('----------------------')
90  for k=1:nt
91     if If(k,2) == 1 && flag1==1
92         tfe11 = (k-1)*Ts;
93                fprintf('Falla en actuador 1 a los %3.1f seg\n',tfe11);
94         flag1 = 2;
95     end
96     if If(k,2) == 0 && flag1==2
97         tfe12 = (k-1)*Ts;
98                fprintf('No hay falla en actuador 1 a los %3.1f ...
                       seg\n',tfe12);
99         flag1 = 3;
100    end
101 end
102  for k=1:nt
103    if If(k,3)== 1 && flag2==1
104        tfe21 = (k-1)*Ts;
105         fprintf('Falla en actuador 2 a los %3.1f seg\n',tfe21);
106        flag2 = 2;
107    end
108    if If(k,3) == 0 && flag2==2
109        tfe22 = (k-1)*Ts;
110         fprintf('No hay falla en actuador 2 a los %3.1f seg\n',tfe22);
111        flag2 = 3;
112    end
113  end
114
115 %% Evaluacion de residuos (Valor RMS)
116 % Convergencia del valor RMS (en 10 seg aprox.)
117 umb1v = umb1*ones(length(t),1);
118 umb2v = umb2*ones(length(t),1);
119 figure;
120 subplot(2,1,1)
121 plot(t,Jr(:,1),'-b','LineWidth',1.2); hold on
122 plot(t,umb1v,'-r','LineWidth',1.2); hold off
123 ylabel('\mid\midr_{1}\mid\mid'); grid on; grid minor
124 xlim([100 Tsim])
125
126 subplot(2,1,2)
127 plot(t,Jr(:,2),'-b','LineWidth',1.2); hold on
```

```
plot(t,umb1v,'-r','LineWidth',1.2); hold off
ylabel('\mid\midr_{2}\mid\mid'); grid on; grid minor
xlabel('Tiempo (s)'); xlim([100 Tsim])

%% Indicador de fallas
figure;
subplot(3,1,1); plot(t,If(:,1),'-b','LineWidth',1.2);
ylabel('I(sin falla)')
subplot(3,1,2); plot(t,If(:,2),'-b','LineWidth',1.2);
ylabel('I(f_{a1})')
subplot(3,1,3); plot(t,If(:,3),'-b','LineWidth',1.2);
ylabel('I(f_{a2})')
xlabel('Tiempo (s)')
```

FDI_sin_falla.m

```
%%% SIMULACION DEL SISTEMA FDI SIN FALLA PARA CALCULO DE UMBRALES %%%
clear; close all; clc;
%% Simulacion FDI
run('cond_ini')      % condiciones iniciales
run('sinfalla');     % datos de la falla
run('observador_UIO'); % datos del observador UIO
Neval = 40;          % ventana de muestras para evaluacion de residuos
save N_eval Neval    % guardamos dato
sim('qt_fdi_sinfalla'); % simulink, controlador por realimentacion de ...
    estados
% sim('qt_fdi_sinfalla_DMC'); % simulink, controlador DMC

% guardamos datos
y1 = y(:,1); y2 = y(:,2); y3 = y(:,3); y4 = y(:,4); % salidas (con ruido)
u1 = u(:,1); u2 = u(:,2); % entradas de control
y11 = xh1(:,1); y21 = xh1(:,2); y31 = xh1(:,3); y41 = xh1(:,4); % UIO 1
y12 = xh2(:,1); y22 = xh2(:,2); y32 = xh2(:,3); y42 = xh2(:,4); % UIO 2
r11 = y1-y11; r21 = y2-y21; r31 = y3-y31; r41 = y4-y41; % residuos ...
    UIO 1
r12 = y1-y12; r22 = y2-y22; r32 = y3-y32; r42 = y4-y42; % residuos ...
    UIO 2
nt = length(t);
%% Evaluacion de residuos (RMS)
figure;
subplot(2,1,1)
plot(t(100:nt),Jr(100:nt,1),'-b','LineWidth',1.2);
ylabel('\mid\midr_{1}\mid\mid');

subplot(2,1,2)
plot(t(100:nt),Jr(100:nt,2),'-b','LineWidth',1.2);
ylabel('\mid\midr_{2}\mid\mid');
xlabel('Tiempo (s)');
% Convergencia de los valores RMS (en 10 seg aprox.)
figure;
subplot(2,1,1)
plot(t(1:500),Jr(1:500,1),'-b','LineWidth',1.2);
```

```
ylabel('\mid\midr_{1}\mid\mid');

subplot(2,1,2)
plot(t(1:500),Jr(1:500,2),'-b','LineWidth',1.2);
ylabel('\mid\midr_{2}\mid\mid');
xlabel('Tiempo (s)');
%% Umbrales
umb1 = 1.10*max(Jr(100:length(t),1)); % 100 muestras para ...
    convergencia del uio
umb2 = 1.10*max(Jr(100:length(t),2)); % 100 muestras para ...
    convergencia del uio
save umbrales.mat umb1 umb2 Neval; % guardamos datos
```

FE_falla.m

```
%%% ESTIMACION DE FALLAS %%%
clear; close all; clc;
%% Simulacion FE
run('cond_ini')      % condiciones iniciales
run('confalla');     % datos de la falla
run('observador_UIO'); % datos del observador
% Estimacion de falla en actuador 1
Fd1 = Bd(:,1);
[Ui_1,Ab11_1,Ab12_1,Bb1_1,VRi_1] = estimacion_mat(Fd1,Ad,Bd);
% Estimacion de falla en actuador 2
Fd2 = Bd(:,2);
[Ui_2,Ab11_2,Ab12_2,Bb1_2,VRi_2] = estimacion_mat(Fd2,Ad,Bd);

%% Simulacion FE
% sim('qt_fe'); % simulink, controlador por realimentacion de estados
sim('qt_fe_DMC'); % simulink, controlador DMC
y1 = y(:,1); y2 = y(:,2); y3 = y(:,3); y4 = y(:,4); % salidas (con ruido)
y11 = xh1(:,1); y21 = xh1(:,2); y31 = xh1(:,3); y41 = xh1(:,4); % UIO 1
y12 = xh2(:,1); y22 = xh2(:,2); y32 = xh2(:,3); y42 = xh1(:,4); % UIO 2
r11 = y1-y11; r21 = y2-y21; r31 = y3-y31; % residual UIO 1
r12 = y1-y12; r22 = y2-y22; r32 = y3-y32; % residual UIO 2

%% Falla real
fd_real = uf-uc;
fa1 = fd_real(:,1);
fa2 = fd_real(:,2);

%% Indicador de fallas
figure;
subplot(3,1,1); plot(t,If(:,1),'-b','LineWidth',1.2);
title('Indicador de falla')
ylabel('I(sin falla)')
subplot(3,1,2); plot(t,If(:,2),'-b','LineWidth',1.2);
ylabel('I(f_{a1})')
subplot(3,1,3); plot(t,If(:,3),'-b','LineWidth',1.2);
ylabel('I(f_{a2})')
xlabel('Tiempo (s)')
```

```
38
39 %% Estimacion de la falla
40 % actuador 1
41 figure;
42 plot(t,fd(:,1),'-r','LineWidth',1.2); hold on;
43 plot(t,fa1,'--b','LineWidth',1.2); hold off;
44 ylabel('Bomba 1 (V)')
45 xlabel('Tiempo (s)'); xlim([100 Tsim]);
46 legend('estimado','real','location','best')
47
48 % actuador 2
49 figure;
50 plot(t,fd(:,2),'-r','LineWidth',1.2); hold on;
51 plot(t,fa2,'--b','LineWidth',1.2); hold off;
52 ylabel('Bomba 2 (V)')
53 xlabel('Tiempo (s)'); xlim([100 Tsim]);
54 legend('estimado','real','location','best')
55
56 % guardamos datos
57 save datos_FE Ui_1 Ab11_1 Ab12_1 Bb1_1 VRi_1 Ui_2 Ab11_2 Ab12_2 Bb1_2 ...
       VRi_2
```

filtro_pasa_bajo.m

```
1 %%% Filtro Pasa Bajo de Primer Orden %%%
2 % Para filtrar la estimacion de la falla
3
4 a = 0.75;        % con filtro
5 %a = 0.01;        % sin filtro
6 Ts = 0.1;        % tiempo de muestreo
7 % continuo
8 numa = 1;
9 dena = [a 1];
10 % discreto
11 Fs = 1/Ts;
12 [numd,dend] = bilinear(numa,dena,Fs);
13 save filtro_pb numd dend % guardamos datos
```

FTC_falla.m

```
1 %%% SIMULACION SISTEMA CONTROL TOLERANTE A FALLAS (FTC) %%%
2 % Comparacion sistema  de control sin falla , con falla y con ...
      compensacion de fallas
3 clear; close all; clc;
4 %% Simulacion FTC
5 run('cond_ini');     % condiciones iniciales
6 run('confalla');     % datos de la falla
7
8 %% Compensacion  de fallas
9 % Caso cuando la matriz B no es de rango fila completo
10 Fd1 = Bd(:,1); Fd2 = Bd(:,2);
```

```
App = Ad(1:2,1:2);
Aps = Ad(1:2,3:4);
Asp = Ad(3:4,1:2);
Ass = Ad(3:4,3:4);
Bp = Bd(1:2,1:2);
Bs = Bd(3:4,1:2);
Fap1 = Fd1(1:2,1);
Fas1 = Fd1(3:4,1);
Fap2 = Fd2(1:2,1);
Fas2 = Fd2(3:4,1);
Bpi = inv(Bp);
Kp = Kx(1:2,1:2);
Ks = Kx(1:2,3:4);
Apcl = (Aps-Bp*Ks);
eig_secund = abs(eig(Ass-Bs*Bp*Bpi*Aps)); % eigenvalores dentro del ...
    circulo unitario
%% Sistema de control sin falla
run('sinfalla')
% sim('qt_controlRE');  % simulink, controlador por realimentacion de ...
    estados
sim('qt_controlDMC'); % simulink, controlador DMC
% guardamos datos
r1s = r(:,1); r2s = r(:,2); % referencias
y1s = y(:,1); y2s = y(:,2); y3s = y(:,3); y4s = y(:,4); % salidas ...
    (con ruido)
u1s = u(:,1); u2s = u(:,2); % entradas de control

%% Sistema de control con falla
run('confalla');
% sim('qt_controlRE'); % controlador por realimentacion de estados
sim('qt_controlDMC'); % simulink, controlador DMC
% guardamos datos
r1 = r(:,1); r2 = r(:,2);
y1 = y(:,1); y2 = y(:,2); y3 = y(:,3);
u1 = u(:,1); u2 = u(:,2);

%% Sistema de control tolerante a fallas
% sim('qt_ftc'); % simulink, controlador por realimentacion de ...
    estados con compensacion
sim('qt_ftc_DMC'); % simulink, controlador DMC con compensacion
% guardamos datos
y1f = y(:,1); y2f = y(:,2); y3f = y(:,3); y4f = y(:,4);
u1f = u(:,1); u2f = u(:,2);

%% Figuras
% Variables de control
figure;
plot(t,u1s,'-g','LineWidth',1.2); hold on
plot(t,u1,'-r','LineWidth',1.2);
plot(t,u1f,'-b','LineWidth',1.2); hold off
ylabel('Bomba 1(V)');
xlabel('Tiempo (s)')
```

```
59 legend('sin falla','sin FTC','con FTC','location','best');
60
61 figure;
62 plot(t,u2s,'-g','LineWidth',1.2); hold on
63 plot(t,u2,'-r','LineWidth',1.2);
64 plot(t,u2f,'-b','LineWidth',1.2); hold off
65 ylabel('Bomba 2 (V)');
66 xlabel('Tiempo (s)')
67 legend('sin falla','sin FTC','con FTC','location','best');
68
69 % Variables de salida
70 figure;
71 plot(t,r1,'-.k','Linewidth',1.2); hold on
72 plot(t,y1s,'-g','Linewidth',1.2);
73 plot(t,y1,'-r','Linewidth',1.2);
74 plot(t,y1f,'-b','Linewidth',1.2); hold off
75 ylabel('Nivel 1 (cm)');
76 xlabel('Tiempo (s)')
77 legend('ref','sin falla','sin FTC','con FTC','location','best');
78
79 figure;
80 plot(t,r2,'-.k','Linewidth',1.2); hold on
81 plot(t,y2s,'-g','Linewidth',1.2)
82 plot(t,y2,'-r','Linewidth',1.2)
83 plot(t,y2f,'-b','Linewidth',1.2); hold off
84 ylabel('Nivel 2 (cm)');
85 xlabel('Tiempo (s)')
86 legend('ref','sin falla','sin FTC','con FTC','location','best');
87
88 %% Evaluacion de residuos
89 figure;
90 subplot(2,1,1)
91 plot(t,Jr(:,1),'-b','LineWidth',1.2);
92 ylabel('r_{u1}'); xlim([100 Tsim])
93
94 subplot(2,1,2)
95 plot(t,Jr(:,2),'-b','LineWidth',1.2);
96 ylabel('r_{u2}');
97 xlabel('Tiempo (s)'); xlim([100 Tsim])
98
99 %% Indicador de fallas
100 figure;
101 subplot(3,1,1); plot(t,If(:,1),'-b','LineWidth',1.2);
102 title('Indicador de falla')
103 ylabel('I(sin falla)')
104 subplot(3,1,2); plot(t,If(:,2),'-b','LineWidth',1.2);
105 ylabel('I(f_{a1})')
106 subplot(3,1,3); plot(t,If(:,3),'-b','LineWidth',1.2);
107 ylabel('I(f_{a2})')
108 xlabel('Tiempo (s)')
109
110 %% ISE sin falla
```

```
nt = length(t);
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1s(i));
    Je1 = Je1 + e1^2;
end
ISE_1s = Je1;

Je2 = 0;
for i=1:nt
    e2 = (r2(i)-y2s(i));
    Je2 = Je2 + e2^2;
end
ISE_2s = Je2;

%% ISE sin FTC (con falla)
nt = length(t);
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1(i));
    Je1 = Je1 + e1^2;
end
ISE_1 = Je1;

Je2 = 0;
for i=1:nt
    e2 = (r2(i)-y2(i));
    Je2 = Je2 + e2^2;
end
ISE_2 = Je2;
%% ISE con FTC (con falla)
nt = length(t);
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1f(i));
    Je1 = Je1 + e1^2;
end
ISE_1f = Je1;

Je2 = 0;
for i=1:nt
    e2 = (r2(i)-y2f(i));
    Je2 = Je2 + e2^2;
end
ISE_2f = Je2;
%% Resultados
fprintf('ISE       Sin falla    Sin FTC    Con FTC\n');
fprintf('-------------------------------------------------\n');
fprintf('Nivel 1     %4.1f       %4.1f       %4.1f\n',ISE_1s,ISE_1,ISE_1f);
fprintf('Nivel 2     %4.1f       %4.1f       %4.1f\n',ISE_2s,ISE_2,ISE_2f);
```

indices_rendimiento.m

```
%%% INDICES DE DESEMPENO PARA CONTROLADORES %%%
%% Referencia y condiciones iniciales
clear; close all; clc;
run('cond_ini')  % condiciones iniciales
run('referencia_indice_rend');
run('sinfalla')  % datos de la falla
krs = 0;

%% Simulacion
% sim('qt_controlRE');  % control por realimentacion de estados
sim('qt_controlDMC');  % control DMC

y1 = y(:,1); y2 = y(:,2);
u1 = u(:,1); u2 = u(:,2);
r1 = r(:,1); r2 = r(:,2);
nt = length(t);

%% Graficas
figure;
subplot(2,2,1); plot(t,r1,':g','LineWidth',1.5); hold on
plot(t,y1,'-r','LineWidth',1.5); hold off
ylabel('Nivel 1 (cm)'); xlabel('Tiempo (s)')
subplot(2,2,2); plot(t,r2,':g','LineWidth',1.5); hold on
plot(t,y2,'-r','LineWidth',1.5); hold off
ylabel('Nivel 2 (cm)'); xlabel('Tiempo (s)')
subplot(2,2,3); plot(t,u1,'-b','LineWidth',1.5);
ylabel('Bomba 1 (V)'); xlabel('Tiempo (s)')
subplot(2,2,4); plot(t,u2,'-b','LineWidth',1.5);
ylabel('Bomba 2 (V)'); xlabel('Tiempo (s)')

%% Indices de desempeno
%% IAE (Integral del valor Absoluto del Error)
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1(i));
    Je1 = Je1 + abs(e1);
end
IAE_1 = Je1;

Je2 = 0;
for i=1:nt
    e2 = (r2(i)-y2(i));
    Je2 = Je2 + abs(e2);
end
IAE_2 = Je2;
%% ISE (Integral del error cuadratico)
Je1 = 0;
for i=1:nt
    e1 = (r1(i)-y1(i));
    Je1 = Je1 + e1^2;
end
ISE_1 = Je1;
```

```
53
54  Je2 = 0;
55  for i=1:nt
56      e2 = (r2(i)-y2(i));
57      Je2 = Je2 + e2^2;
58  end
59  ISE_2 = Je2;
60
61  %% TVU (Variacion Total de la Senal de Control, es equivalente al IADU)
62  % IADU: Integral del valor Absoluto de la Derivada de la senal de control
63  Ju1 = 0;
64  for i=1:nt-1
65      Ju1 = Ju1 + abs(u1(i+1)-u1(i));
66  end
67  TVU_1 = Ju1;
68
69  Ju2 = 0;
70  for i=1:nt-1
71      Ju2 = Ju2 + abs(u2(i+1)-u2(i));
72  end
73  TVU_2 = Ju2;
74
75  %% ISU (ISU, Integral de la salida de control cuadratica)
76  Ju1 = 0;
77  u1es = u1(end);
78  for i=1:nt-1
79      Ju1 = Ju1 + (u1(i+1)-u1es)^2;
80  end
81  ISU_1 = Ju1;
82
83  Ju2 = 0;
84  u2es = u2(end);
85  for i=1:nt-1
86      Ju2 = Ju2 + (u2(i+1)-u2es)^2;
87  end
88  ISU_2 = Ju2;
89
90  %% Tes (Tiempo de establecimiento)
91  crit = 2;
92  s = length(t);
93  r1i = (1-(crit/100))*y1(end);
94  r1s = (1+(crit/100))*y1(end);
95  while y1(s)>r1i && y1(s)<r1s
96      s = s - 1;
97  end
98  Tes_1 = t(s-1)-t11;
99
100 s = length(t);
101 r2i = (1-(crit/100))*y2(end);
102 r2s = (1+(crit/100))*y2(end);
103 while y2(s)>r2i && y2(s)<r2s
104     s = s - 1;
```

```
105 end
106 Tes_2 = t(s-1)-t21;
107
108 %% PM (Sobreimpulso o Pico maximo)
109 y1_max = max(y1);
110 if y1_max > r1(end)
111     PM_1 = (y1_max-r1(end))/r1(end)*100;
112 else
113     PM_1 = 0;
114 end
115
116 y2_max = max(y2);
117 if y2_max > r2(end)
118     PM_2 = (y2_max-r2(end))/r2(end)*100;
119 else
120     PM_2 = 0;
121 end
122
123 %% Ees (Error estacionario)
124 Ees_1 = abs(r1(end)-y1(end))/r1(end)*100;
125 Ees_2 = abs(r2(end)-y2(end))/r2(end)*100;
126
127
128 %% Resultados
129 fprintf('Var.Controlada   ISE    ISU   Tes(s)  Ees(%%)  PM(%%)\n');
130 fprintf('-----------------------------------------------------\n');
131 fprintf('   Nivel 1        %3.1f  %3.1f    %2.1f    %1.1f     ...
        %1.1f\n',ISE_1,ISU_1,Tes_1,Ees_1,PM_1);
132 fprintf('   Nivel 2        %3.1f  %3.1f    %2.1f    %1.1f     ...
        %1.1f\n',ISE_2,ISU_2,Tes_2,Ees_2,PM_2);
```

matriz_dinamica.m

```
1  function G = matriz_dinamica(gi,P,M)
2  %G = matriz_dinamica(gi,P,M) calcula la matriz dinamica G, donde:
3  %   gi: coeficientes de la respuesta paso
4  %   P: horizonte de prediccion (P=N2-N1+1)
5  %   M: horizonte de control (M=Nu)
6  %   La matriz G tiene dimension PxM
7  G = zeros(P,M); % inicializamos
8  for j=1:M
9      G(j:P,j) = gi(1:P-j+1,1);
10 end
```

modelo_lineal.m

```
1  %%% MODELO LINEAL DEL SISTEMA DE CUATRO TANQUES ACOPLADOS %%%
2  %% Parametros del proceso
3  % area de los tanques
4  A1 = 28; A3 = 28; % cm^2
5  A2 = 32; A4 = 32; % cm^2
```

```
% area de las tuberias de salida de los tanques
a1 = 0.071; a3 = 0.071;  % cm^2
a2 = 0.057; a4 = 0.057;  % cm^2
% ganancia de las valvulas
k1 = 3.33;  % cm^3/V/s
k2 = 3.35;  % cm^3/V/s
% aperturas de las valvulas
g1 = 0.7;
g2 = 0.6;
% gravedad
g = 981;  % cm/s^2

%% Parametros de linealizacion
% Punto de operacion para las variables de estado (niveles)
h1o = 12.4;  % cm
h2o = 12.7;  % cm
h3o = 1.6;  % cm
h4o = 1.45;  % cm
Yo = [h1o; h2o; h3o; h4o];
% Punto de operacion  para las variables de entrada (voltajes)
v1o = 3;  % V
v2o = 3;  % V
Uo = [v1o; v2o];

%% Modelo lineal en espacio de estados continuo
T1 = A1/a1*sqrt(2*h1o/g);
T2 = A2/a2*sqrt(2*h2o/g);
T3 = A3/a3*sqrt(2*h3o/g);
T4 = A4/a4*sqrt(2*h4o/g);
a11 = -1/T1;
a13 = A3/A1/T3;
a22 = -1/T2;
a24 = A4/A2/T4;
a33 = -1/T3;
a44 = -1/T4;
b11 = g1*k1/A1;
b22 = g2*k2/A2;
b32 = (1-g2)*k2/A3;
b41 = (1-g1)*k1/A4;
A = [ a11  0  a13  0
       0   a22  0  a24
       0    0  a33  0
       0    0   0  a44 ];
B = [ b11 0
       0   b22
       0   b32
       b41  0 ];
C = eye(4);            % se  mide el nivel de todos los tanques
C1 = [ 1  0  0  0
        0  1  0  0  ]; % variable controladas
D = zeros(4,2);
clear T1 T2
```

```
%% Modelo lineal en espacio de estados discreto
Ts = 0.1;  % tiempo de muestreo
[Ad,Bd,Cd,Dd] = c2dm(A,B,C,D,Ts,'zoh');  % modelo discreto
```

observador_UIO.m

```
%%% DISENO OBSERVADOR UIO %%%
run('modelo_lineal');  % llamamos al programa 'modelo_lineal'
%% UIO 1
% Observador desacoplado de la falla en el actuador 1
Fd = Bd(:,1);
[E1,T1,K1,H1] = uio_lineald(Ad,Bd,Cd,Fd);
eigE1 = abs(eig(E1));  % estable si eigenvalores dentro del circulo ...
    unitario

%% UIO 2
% Observador desacoplado de la falla en el actuador 2
Fd = Bd(:,2);
[E2,T2,K2,H2] = uio_lineald(Ad,Bd,Cd,Fd);
eigE2 = abs(eig(E2));  % estable si eigenvalores dentro del circulo ...
    unitario

save datos_UIO E1 T1 K1 H1 E2 T2 K2 H2  % guardamos datos
```

rango_lineal.m

```
%% Rango lineal para el proceso de cuatro tanques
clear; close all; clc
run('modelo_lineal')
%% Grafica en 3D de los puntos estacionarios
v1 = 0:0.1:6.0;  % voltaje bomba 1
v2 = 0:0.1:6.0;  % voltaje bomba 1
[V1,V2] = meshgrid(v1,v2);
H1 = ((g1*k1*V1+(1-g2)*k2*V2)/a1).^2/(2*g);  % nivel 1 (cm)
H2 = (((1-g1)*k1*V1+g2*k2*V2)/a2).^2/(2*g);  % nivel 2 (cm)
figure(1);
mesh(V1,V2,H1); hold on
plot3(3.0486,2.9612,12.4,'r*');
xlabel('v_1 (V)'); ylabel('v_2 (V)'); zlabel('h_1 (cm)')
zlim([0 20])

figure(2);
mesh(V1,V2,H2); hold on
plot3(3.0486,2.9612,12.7,'r*');
xlabel('v_1 (cm)'); ylabel('v_2 (cm)'); zlabel('h_2 (cm)')
zlim([0 20])

%% Grafica en 2D de los puntos estacionarios
v1 = 0:0.1:6.0;  % voltaje bomba 1
k = 1;
for v2 = 1:1:5
```

```
    h1(:,k) = ((g1*k1*v1+(1-g2)*k2*v2)/a1).^2/(2*g); % nivel 1 (cm)
    h2(:,k) = (((1-g1)*k1*v1+g2*k2*v2)/a2).^2/(2*g); % nivel 2 (cm)
    k = k + 1;
end
figure(3);
plot(v1,h1); hold on
plot(3.0, 12.4,'*r'); hold off
xlabel('v_1 (V)'); ylabel('h_1 (cm)'); grid
legend('v_2 = 1 V','v_2 = 2 V','v_2 = 3 V','v_2 = 4 V','v_2 = 5 ...
    V','pto. op.','location','best')
ylim([0 20]);

figure(4);
plot(v1,h2); hold on
plot(3.0, 12.7,'*r'); hold off
xlabel('v_1 (V)'); ylabel('h_2 (cm)'); grid
legend('v_2 = 1 V','v_2 = 2 V','v_2 = 3 V','v_2 = 4 V','v_2 = 5 ...
    V','pto. op.','location','best')
ylim([0 20]);
```

referencia.m

```
%%% REFERENCIAS (SETPOINTS) %%%
clear r1 r2 t
%% Datos
Tsim = 1800;     % tiempo de simulacion (s)
Ts = 0.1;        % tiempo de muestreo, seg
ti = 0;      dt = Ts;     Tf = Tsim;
h1o = 12.4;  % punto de operacion nivel 1, cm
h2o = 12.7;  % punto de operacion nivel 2, cm

%% Vector tiempo
t = ti:dt:Tf;        t = t';  % vector tiempo
nt = length(t);               % longitud del vector tiempo
amp_p = 25;                   % amplitud del setpoint (%)
amp = 1+(amp_p/100);          % amplitud

%% Referencia para el nivel del tanque 1
t11 = 100;
nt11 = round(t11/dt);
r1(1:nt11,1) = h1o*ones(nt11,1);
t12 = 900;
nt12 = round(t12/dt);
r1(nt11+1:nt12) = amp*h1o*ones(nt12-nt11,1);
t13 = 1500;
nt13 = round(t13/dt);
r1(nt12+1:nt13) = h1o*ones(nt13-nt12,1);
r1(nt13+1:nt) = amp*h1o*ones(nt-nt13,1);

%% Referencia para el nivel del tanque 2
t21 = 300;
nt21 = round(t21/dt);
```

```
r2(1:nt21,1) = h2o*ones(nt21,1);
t22 = 1200;
nt22 = round(t22/dt);
r2(nt21+1:nt22) = amp*h2o*ones(nt22-nt21,1);
r2(nt22+1:nt) = h2o*ones(nt-nt22,1);

%% Guardamos datos
referencia_nivel = [t';r1';r2'];
save ref_nivel referencia_nivel

%% Grafica
figure;
subplot(2,1,1);
plot(t,r1,'-b','Linewidth',1.2); title('Referencias')
ylabel('Nivel 1 (cm)'); xlabel('Tiempo (s)')
subplot(2,1,2); plot(t,r2,'-b','Linewidth',1.2);
ylabel('Nivel 2 (cm)'); xlabel('Tiempo (s)')
```

referencia_desempeno.m

```
%%% REFERENCIA PARA EVALUACION DE DESEMPENO DE CONTROLADORES %%%
clear r1 r2 t;
%% Datos
Tsim = 200; % tiempo de simulacion, seg
Ts = 0.1;   % tiempo de muestreo, seg
h1o = 12.4; % punto de operacion nivel 1, cm
h2o = 12.7; % punto de operacion nivel 2, cm

%% Vector tiempo
ti = 0;      dt = Ts;    Tf = Tsim;
t = ti:dt:Tf; t = t';
nt = length(t);

%% Amplitud del escalon
amp_p = 25;                 % amplitud del setpoint (%)
amp = 1+(amp_p/100);        % amplitud

%% Referencia para el nivel del tanque 1
t11 = 12;
nt11 = round(t11/dt);
r1(1:nt11,1) = h1o*ones(nt11,1);
r1(nt11+1:nt) = amp*h1o*ones(nt-nt11,1);

%% Referencia para el nivel del tanque 2
t21 = 12;
nt21 = round(t21/dt);
r2(1:nt21,1) = h2o*ones(nt21,1);
r2(nt21+1:nt) = amp*h2o*ones(nt-nt21,1);
%% Figuras
figure;
subplot(2,1,1); plot(t,r1,'-b','LineWidth',1.2);
title('Referencias');
```

```
ylabel('nivel 1 (cm)'); xlabel('Tiempo (s)')
subplot(2,1,2); plot(t,r2,'-b','LineWidth',1.2);
ylabel('nivel 2 (cm)'); xlabel('Tiempo (s)')

%% Guardamos datos
referencia_nivel = [t';r1';r2'];
save ref_nivel referencia_nivel
```

respuesta_libre.m

```
function dg = respuesta_libre(g,P,N)
% dg = respuesta_libre(g,P,N) calcula el vector con las gk+i - gi, donde:
%   g: coeficientes de la respuesta paso
%   P: horizonte de prediccion (P=N2-N1+1)
%   N: horizonte del modelo
dg = zeros(P,N);
for i = 1:N
    for k = 1:P
        if (k+i)>N
            dg(k,i) = g(N)-g(i);
        else
            dg(k,i) = g(k+i)-g(i);
        end
    end
end
```

uio_lineald.m

```
function [F,T,K,H]=uio_lineald(A,B,C,E)
% Diseno de observador de entradas desconocidas (UIO)
% Sistema en espacio de estados discreto
%   x(k+1) = Ax(k) + Bu(k) + Ed(k)
%   y(k) = Cx(k)
% UIO
%   z(k+1) = Fz(k) + TBu(k) + Ky(k)
%   xh(k) = z(k) + Hy(k)
% donde K = K1+K2
% Procedimiento de diseno de UIO
%   1) Verificar condiciones de existencia:
%       i.-Rango(CE) = Rango(E)
%       ii.-(C,A1) es observable. Donde:
%           A1 = A-E*inv((CE)'CE)(C*E)'CA
%   2) Calcular H
%       H = E*inv((CE)'*(CE))(CE)'
%   3) Calcular T, A1
%       T = I - HC
%       A1 = A - HCA = TA
%   4) Encontrar K1 para estabilizar F = A1 - K1C
%   5) Calcular F, y K
%       F = A - HCA - K1C = TA - K1C
%       K2 = FH
```

```
%       K = K1 + K2
if nargin≠4
    error('Numero de argumentos de entrada incorrecto');
end
[n,r]=size(B);
[m,n]=size(C);
[n,q]=size(E);
if (m≤q)
    error('El numero de salidas debe ser mayor que el numero de ...
        entradas desconocidas');
end
if rank(C*E)≠rank(E)
    error('Rango(CE) diferente Rango(E)');
end
H = E*inv((C*E)'*(C*E))*(C*E)';
T = eye(n) - H*C;
A1 = T*A;
if rank(obsv(A1,C))≠n
    error('(C,A1) no es observable');
end
% Estabilizacion de (A1-K1*C) usando:
%% ubicacion de polos
j = sqrt(-1);
Ts = 0.1;
Tes = 30;
Tes_o = Tes/5;
s41 = -4.0156+j*5.0723;
s42 = -5.5281+j*1.6553;
s4 = [s41 conj(s41) s42 conj(s42)];
s_opolos = s4/Tes_o;
z_opolos = exp(s_opolos*Ts);
K1 =place(A1',C',z_opolos); K1 = K1';

%% matrices
F = A1 - K1*C;
K2 = F*H;
K = K1 + K2;
```

ANEXO B. PROGRAMAS EN SIMULINK

qt_controlRE.slx

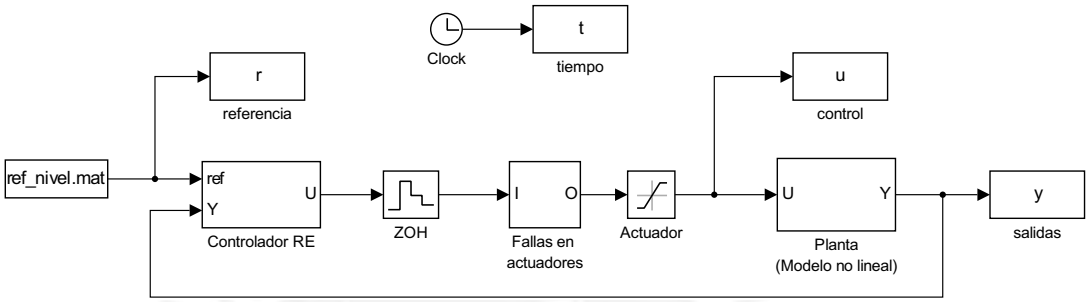

Figura B.1: Control por realimentación de estados.

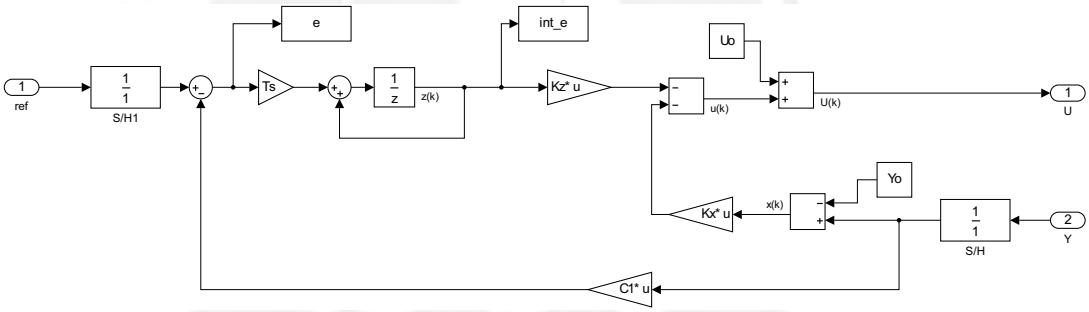

Figura B.2: Bloque "Controlador RE".

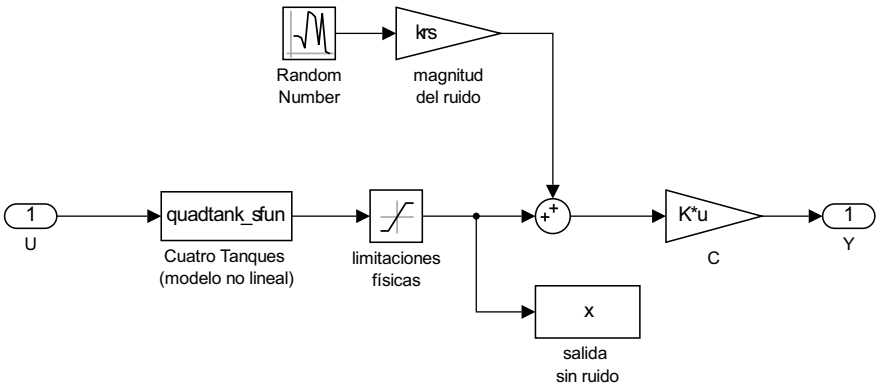

Figura B.3: Bloque "Planta"

quadtank_sfun.m (función de la Fig. B.3)

```
function [sys,x0,str,ts] = quadtank_sfun(t,x,u,flag,h0)
%
% Resuelve las ecuaciones diferenciales del modelo
% Sistema de Cuatro Tanques
%
% Paper: The Quadruple-Tank Process: A Multivariable Laboratory Process
% with an Adjustable Zero (2000)
%
% Los estados son las alturas de los tanques
%
% Usar como s-function en un diagrama de Simulink
%
% Enero, 2017
% Josmell Cordova Claros

% condiciones del proceso:
%    v1o = 3; [V]
%    v2o = 3; [V]
%    h1o = 12.4; [cm]
%    h2o = 12.7; [cm]
%    h3o = 1.6; [cm]
%    h4o = 1.4; [cm]
%    x0  = [h1o; h2o; h3o; h4o] = condiciones iniciales;
switch flag

    case 0 % inicializacion;

        sizes = simsizes;
        sizes.NumContStates = 4;
        sizes.NumDiscStates = 0;
        sizes.NumOutputs    = 4;
        sizes.NumInputs     = 2;
        sizes.DirFeedthrough = 0;
        sizes.NumSampleTimes = 1;
        sys = simsizes(sizes);

        str = [];
        ts = [0 0];
        x0 = h0 ;

    case 1 % derivadas;

        % es conveniente usar notaciones comunes para los estados

        h1 = x(1);
        h2 = x(2);
        h3 = x(3);
        h4 = x(4);

```

```
50          % entradas
51
52          v1 = u(1);  % manipulada
53          v2 = u(2);  % manipulada
54
55          % parametros del proceso
56
57          % area de los tanques
58          A1 = 28;  % [cm2]
59          A2 = 32;  % [cm2]
60          A3 = 28;  % [cm2]
61          A4 = 32;  % [cm2]
62          % area de las salidas de los tanques
63          a1 = 0.071;  % [cm2]
64          a2 = 0.057;  % [cm2]
65          a3 = 0.071;  % [cm2]
66          a4 = 0.057;  % [cm2]
67          % ganancia de las bombas
68          k1 = 3.33;  % cm3/V/s
69          k2 = 3.35;  % cm3/V/s
70          % bypass de las valvulas
71          g1 = 0.70;
72          g2 = 0.60;
73          % gravedad
74          g = 981;  % cm2/s
75
76          % derivadas de los estados
77          if h1 < 0
78              h1 = 0;  % para evitar valores negativos en la raiz cuadrada
79          end
80          if h2 < 0
81              h2 = 0;
82          end
83          if h3 < 0
84              h3 = 0;
85          end
86          if h4 < 0
87              h4 = 0;
88          end
89          dh1dt = -a1/A1*sqrt(2*g*h1)+a3/A1*sqrt(2*g*h3)+g1*k1*v1/A1;
90          dh2dt = -a2/A2*sqrt(2*g*h2)+a4/A2*sqrt(2*g*h4)+g2*k2*v2/A2;
91          dh3dt = -a3/A3*sqrt(2*g*h3)+(1-g2)*k2*v2/A3;
92          dh4dt = -a4/A4*sqrt(2*g*h4)+(1-g1)*k1*v1/A4;
93          sys = [dh1dt; dh2dt; dh3dt; dh4dt];
94
95      case 3  % salidas;
96
97          sys = [x(1); x(2); x(3); x(4)];
98
99      case {2, 4, 9}
100         sys = [];
101
```

```
102      otherwise
103          error(['unhandled flag = ',num2str(flag)]);
104
105  end
```

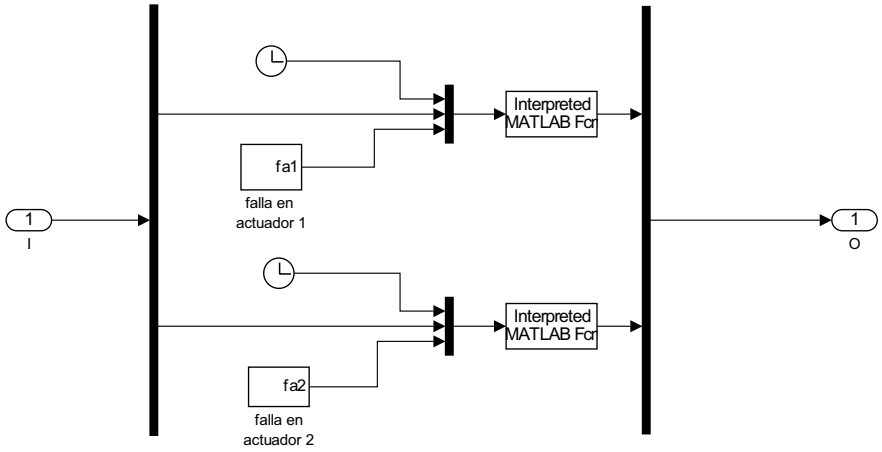

Figura B.4: Bloque “Fallas en actuadores”.

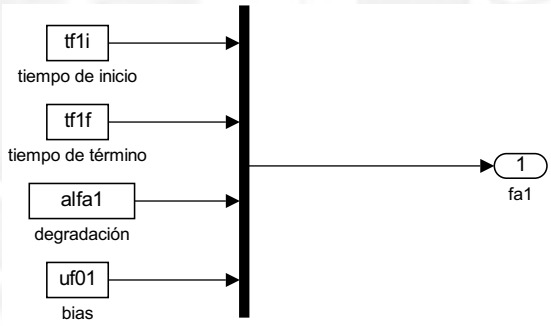

Figura B.5: Bloque “falla en actuador 1”.

actuador_falla.m (función de la Fig. B.4)

```
function u_falla = actuador_falla(input)
% falla en el actuador
tiempo = input(1);          % tiempo de simulacion [s]
u = input(2);                 % entrada de control (sin falla)
tiempo_ini = input(3);     % tiempo en el que inicia la falla [s]
tiempo_fin =  input(4);    % tiempo en el que termina la falla [s]
alfa = input(5);              % perdida de efectividad del actuador [0-1]
bias = input(6);             % sesgo (bias) [V]

if tiempo > tiempo_ini && tiempo < tiempo_fin
    u_falla = alfa*u + bias; % entrada de control con falla
else
    u_falla = u; % entrada de control sin falla
end

end
```

qt_controlDMC.slx

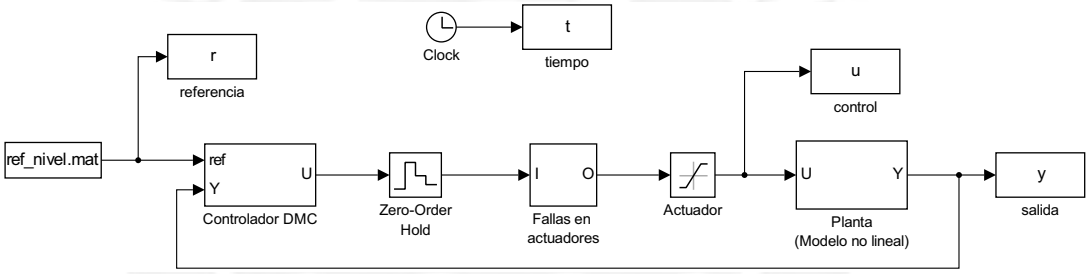

Figura B.6: Control DMC.

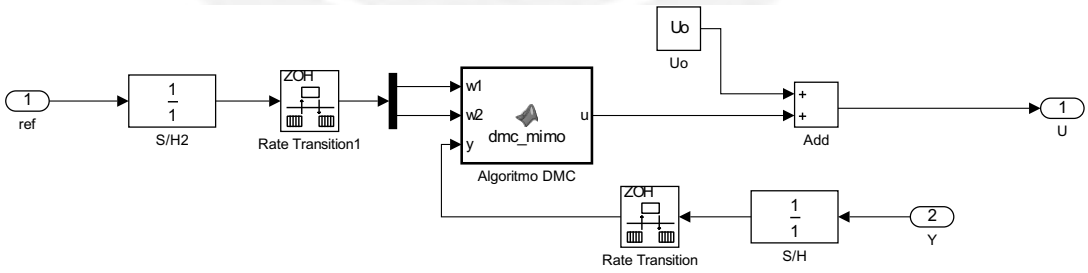

Figura B.7: Bloque “Controlador DMC”.

dmc_mimo.m (función de la Fig. B.7)

```
function u = dmc_mimo(w1,w2,y)
%#codegen
datos = coder.load('datos_controladorDMC.mat');
dg11 = datos.dg11;
dg12 = datos.dg12;
dg21 = datos.dg21;
```

```
dg22 = datos.dg22;
Kc1 = datos.Kc1;
P = datos.P;
N11 = datos.N11;
N12 = datos.N12;
N21 = datos.N21;
N22 = datos.N22;
N = max([N11,N12,N21,N22]);
alfa1 = datos.alf1;
alfa2 = datos.alf2;
persistent dup1 dup2 u1_ant u2_ant
if isempty(u1_ant)
    u1_ant = 0;  % entrada de control anterior u1(k-1)
    u2_ant = 0;  % entrada de control anterior u2(k-1)
    dup1 = zeros(N,1);  % cambio en la entrada de control pasadas
    dup2 = zeros(N,1);  % cambio en la entrada de control pasadas
end
%% referencia interna
r1 = zeros(P,1);
r2 = zeros(P,1);
for k=1:P
    if k<2
        r1(k) = alfa1*y(1) + (1-alfa1)*w1;
        r2(k) = alfa2*y(2) + (1-alfa2)*w2;
    else
        r1(k) = alfa1*r1(k-1) + (1-alfa1)*w1;
        r2(k) = alfa2*r2(k-1) + (1-alfa2)*w2;
    end
end

%% respuesta libre
f1 = y(1) + dg11*dup1(1:N11) + dg12*dup2(1:N12);
f2 = y(2) + dg21*dup1(1:N21) + dg22*dup2(1:N22);
%% entrada de control
r = [r1; r2];
f = [f1; f2];
du = Kc1*(r-f);
du1 = du(1);
du2 = du(2);
u1 = u1_ant + du1;  % entrada de control  1
u2 = u2_ant + du2;  % entrada de control  2
u = [u1; u2];
%% Actualizacion del cambio en la entrada de control pasadas dup
dup1 = [du1; dup1(1:N-1)];
dup2 = [du2; dup2(1:N-1)];
%% Actualizacion de la entrada de control anterior
u1_ant = u1;
u2_ant = u2;
```

qt_fdi_sinfalla.slx

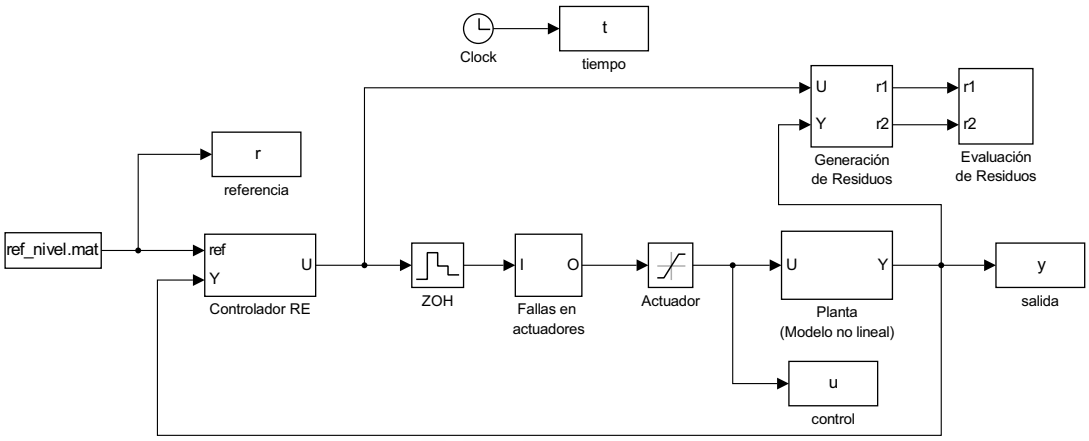

Figura B.8: Detección y aislamiento de fallas (con controlador por realimentación de estados).

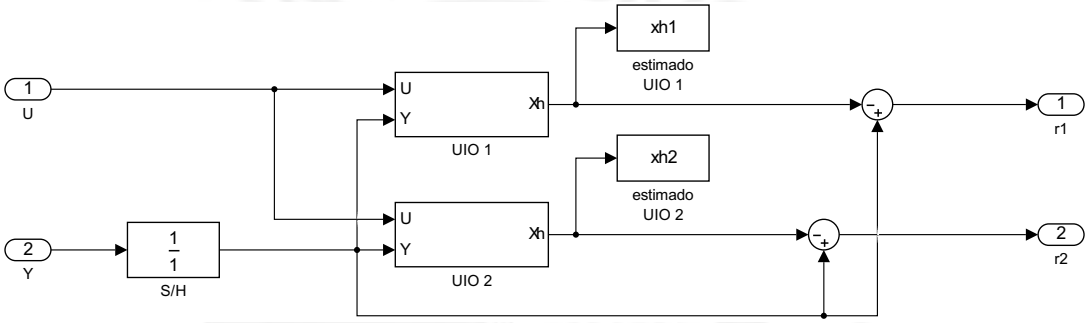

Figura B.9: Bloque “Generación de Residuos”.

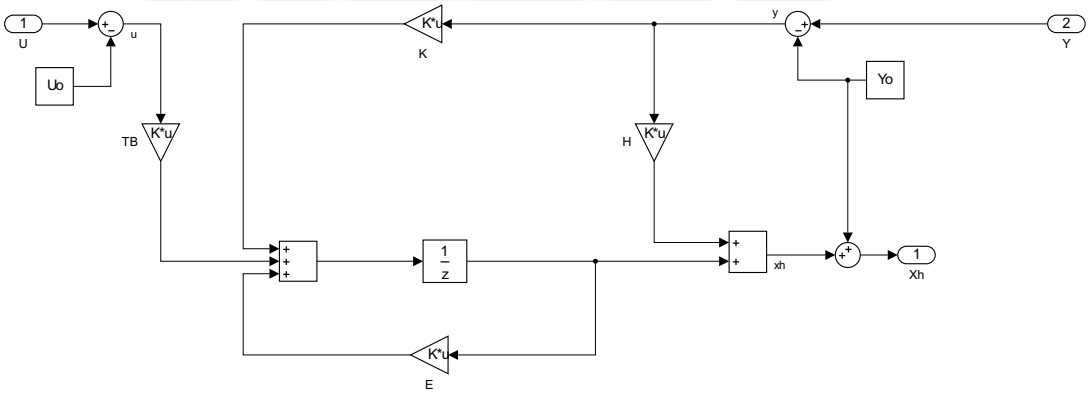

Figura B.10: Bloque “UIO”.

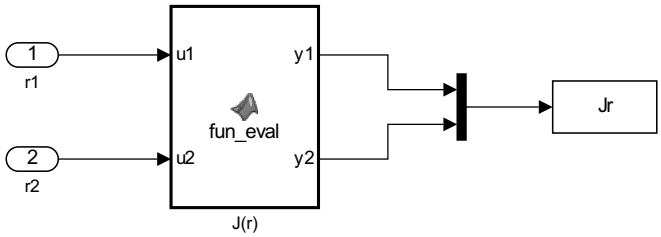

Figura B.11: Bloque “Evaluación de Residuos”.

fun_eval.m (función de la Fig. B.11)

```
function [y1,y2] = fun_eval(u1,u2)
% Funcion de evaluacion de residuos: Raiz de la media cuadratica (RMS)
% entradas:
%           u1 = residuo UIO 1
%           u2 = residuo UIO 2
% salidas:
%           y1 = Evaluacion de r1
%           y2 = Evaluacion de r2
r1 = u1;
r2 = u2;
datos = coder.load('N_eval.mat'); % cargamos datos
Neval = datos.Neval; % ventana de datos para evaluacion de residuos
N = Neval;
persistent  norma_r1 norma_r2 cont
if isempty(norma_r1)
    norma_r1 = zeros(N,1);
end
if isempty(norma_r2)
    norma_r2 = zeros(N,1);
end
if isempty(cont)
    cont = 1;
end
norma_r1(1) = sum(r1.*r1);
norma_r2(1) = sum(r2.*r2);
if cont<N
    rms1 = (sum(norma_r1)/cont)^0.5;
    rms2 = (sum(norma_r2)/cont)^0.5;
    cont = cont + 1;
else
    rms1 = (sum(norma_r1)/N)^0.5;
    rms2 = (sum(norma_r2)/N)^0.5;
end
for i = 0: N-2
    norma_r1(N-i) = norma_r1(N-i-1);
    norma_r2(N-i) = norma_r2(N-i-1);
end

y1 = rms1;
```

```
40  y2 = rms2;
```

qt_fdi_sinfalla_DMC.slx

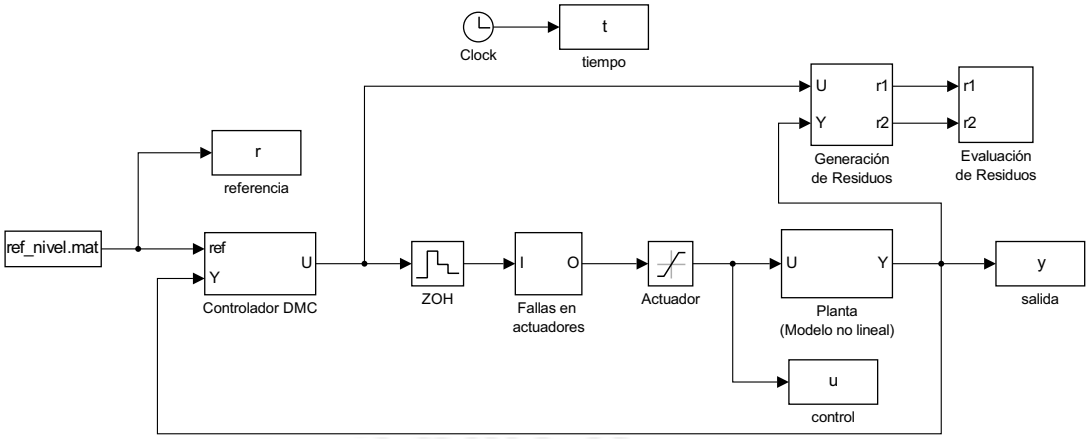

Figura B.12: Detección y aislamiento de fallas (controlador DMC).

qt_fdi_confalla.slx

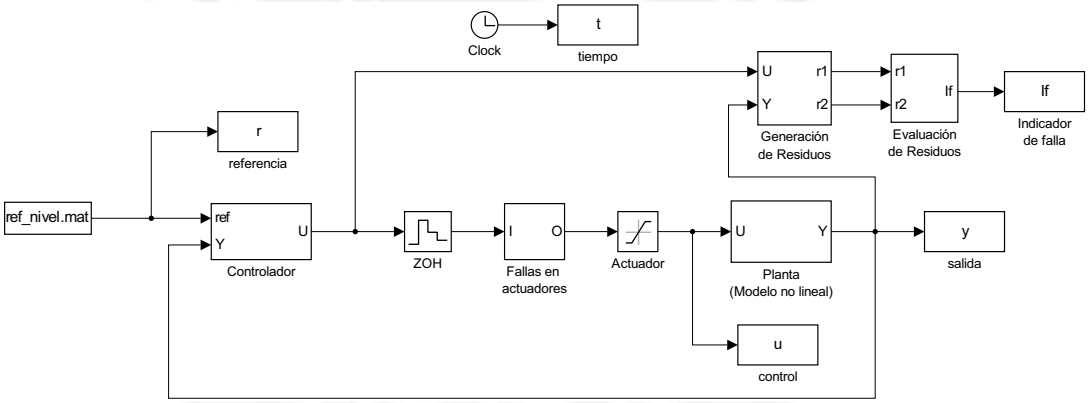

Figura B.13: Detección y aislamiento de fallas (controlador por realimentación de estados).

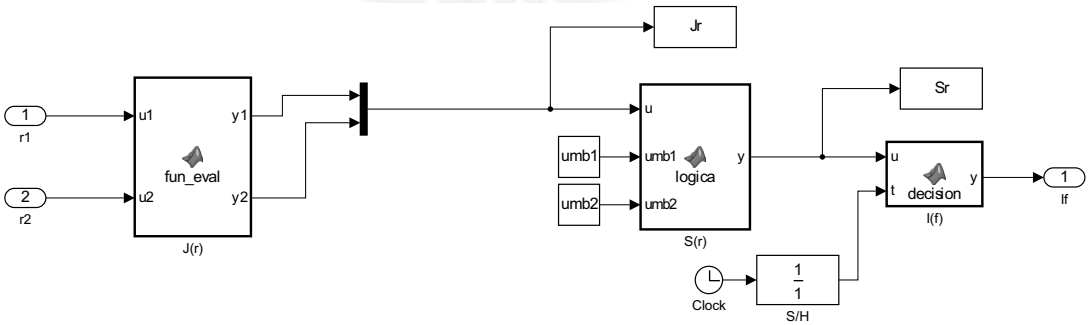

Figura B.14: Bloque “Evaluación de Residuos”.

logica.m (función de la Fig. B.14)

```
function y = logica(u,umb1,umb2)
%#codegen
datos = coder.load('N_umb.mat');
Numb = datos.Numb; % contador para detectar fallas
cont = Numb;
persistent cont1 cont2
if isempty(cont1)
    cont1 = 0;
    cont2 = 0;
end
Jr1 = u(1);
Jr2 = u(2);
if Jr1>umb1
    cont1 = cont1 + 1;
    if cont1 > cont
    Sr1 = 1; % falla en actuador 1
    else
        Sr1 = 0;
    end
else
    Sr1 = 0;
    cont1 = 0;
end
if Jr2>umb2
    cont2 = cont2 + 1;
    if cont2 > cont
    Sr2 = 1; % falla en actuador 2
    else
        Sr2 = 0;
    end
else
    Sr2 = 0;
    cont2 = 0;
end
y = [Sr1,Sr2]';
```

decision.m (función de la Fig. B.14)

```
function y = decision(u,t)
%#codegen
Sr1 = u(1);
Sr2 = u(2);
MFF = [ 0 0 1
        0 1 0 ]; % Matriz de firma de fallas
If = [0 0 0 0]'; % Indicador de falla
if t<10 % 10 segundos para convergencia del UIO
    If(1) = 1; % No hay falla
else
    if Sr1 == MFF(1,1) && Sr2 == MFF(2,1)
        If(1) = 1; % No hay falla
    else
        If(1) = 0;
    end
    if Sr1 == MFF(1,2) && Sr2 == MFF(2,2)
        If(2) = 1; % falla actuador 1
    else
        If(2) = 0;
    end
    if Sr1 == MFF(1,3) && Sr2 == MFF(2,3)
        If(3) = 1; % falla actuador 2
    else
        If(3) = 0;
    end
    if Sr1 == 1 && Sr2 == 1
        If(4) = 1; % si es que se activan ambas fallas (raro)
    else
        If(4) = 0;
    end
end
y = If;
```

qt_fdi_confalla_DMC.slx

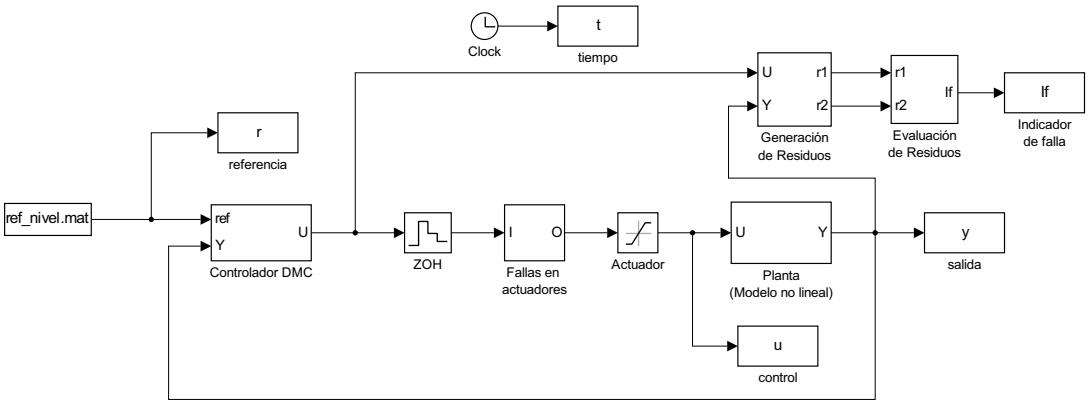

Figura B.15: Detección y aislamiento de fallas (controlador DMC).

qt_fe.slx

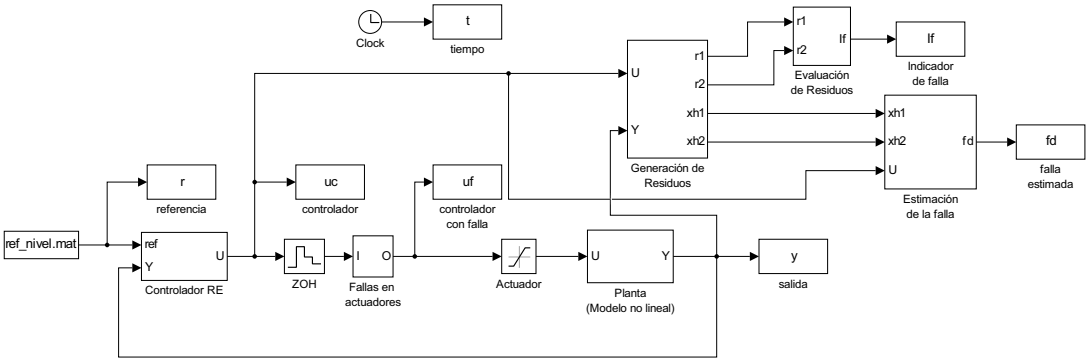

Figura B.16: Estimación de fallas (controlador por realimentación de estados).

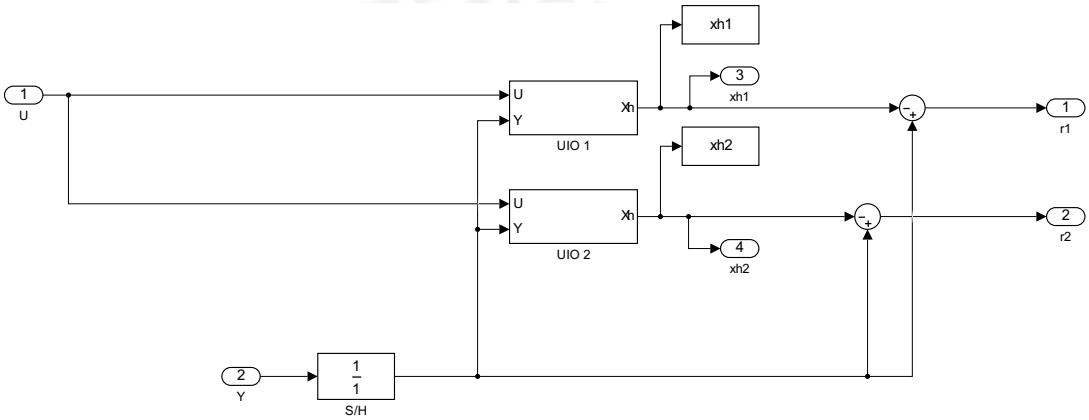

Figura B.17: Bloque “Generación de Residuos”.

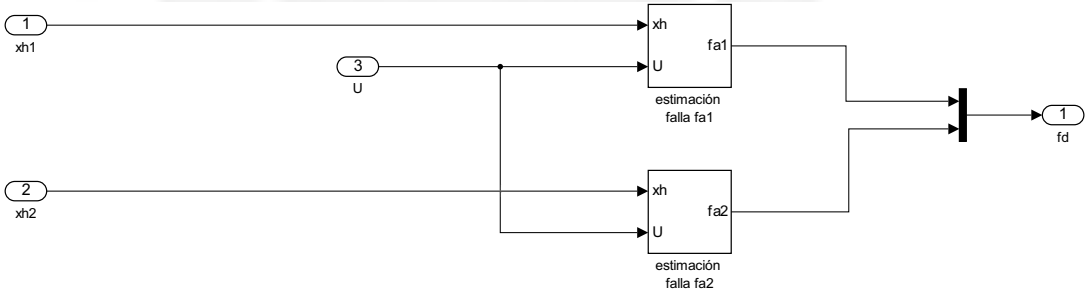

Figura B.18: Bloque “Estimación de la falla”.

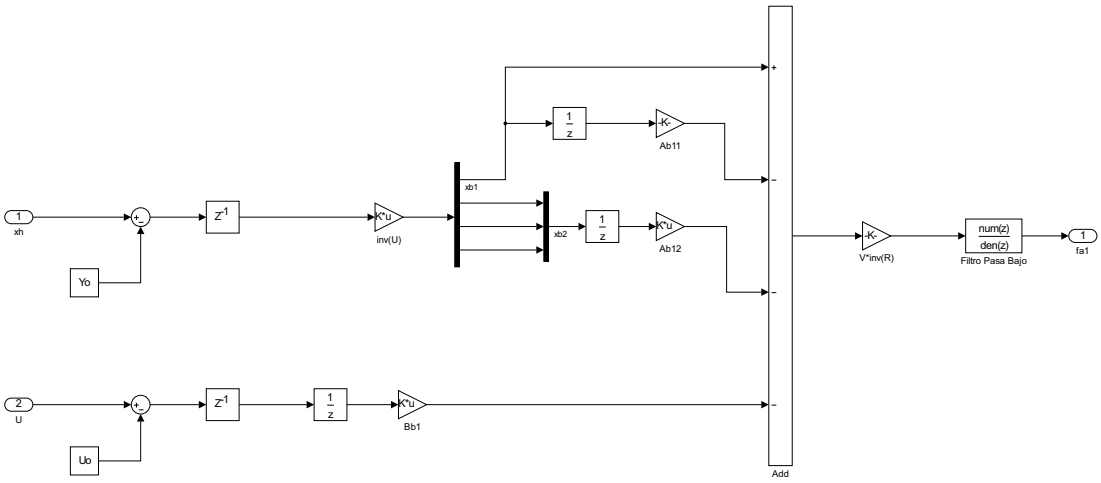

Figura B.19: Bloque “estimación falla fa1”.

qt_fe_DMC.slx

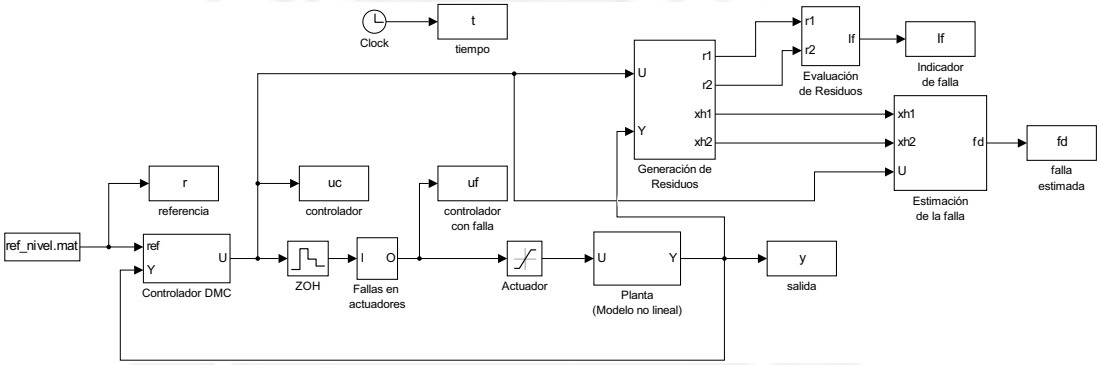

Figura B.20: Estimación de fallas (controlador DMC).

qt_ftc.slx

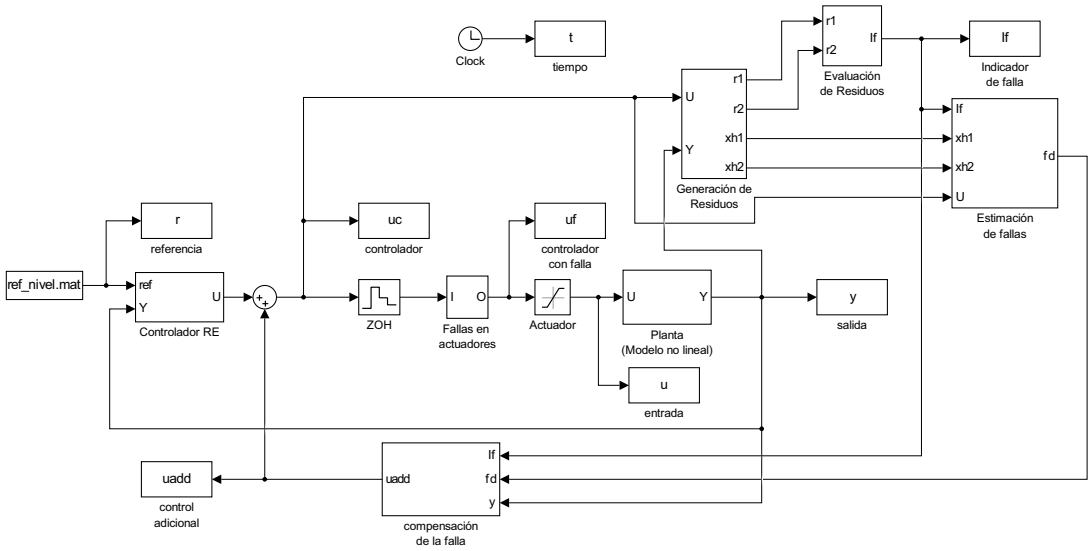

Figura B.21: Controlador por realimentación de estados con compensación de fallas.

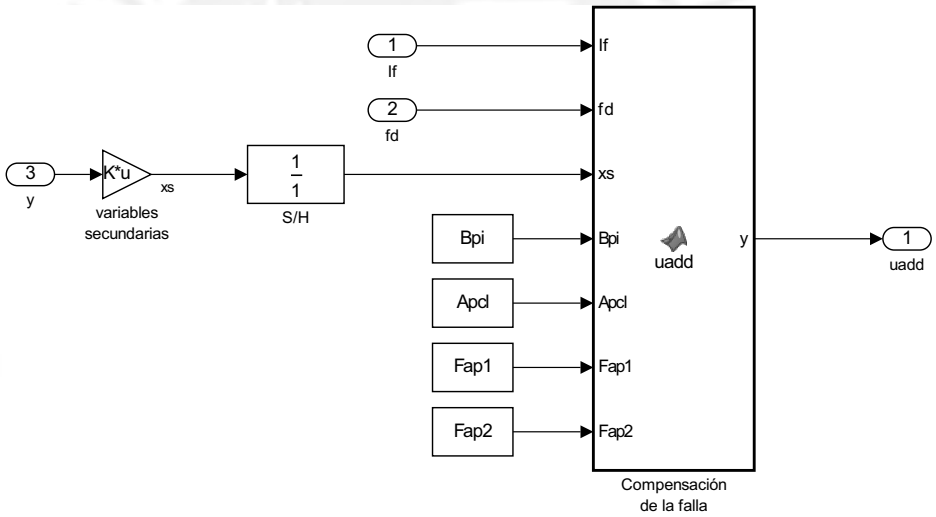

Figura B.22: Bloque “compensación de fallas”.

uadd.m (función de la Fig. B.22)

```
function y = uadd(If,fd,xs,Bpi,Apcl,Fap1,Fap2)
%#codegen
Badd1 = [1; 0];
Badd2 = [0; 1];
if If(1) == 1 && If(2) == 0 && If(3)== 0 &&  If(4)== 0 % no hay falla
    y = [0; 0];
elseif If(2) == 1 && If(1) == 0 && If(3)== 0 &&  If(4)== 0 % falla en ...
    actuador 1
     % y = -Bpi*(Apcl*xs + Fap1*fd);
    y = -Badd1*fd;
    y = [y(1); 0];
elseif If(3) == 1 && If(1) == 0 && If(2)== 0 &&  If(4)== 0 % falla en ...
    actuador 2
     % y = -Bpi*(Apcl*xs + Fap2*fd);
    y = -Badd2*fd;
    y = [0; y(2)];
else
    y = [0; 0]; % no hay falla
end
```

qt_ftc_DMC.slx

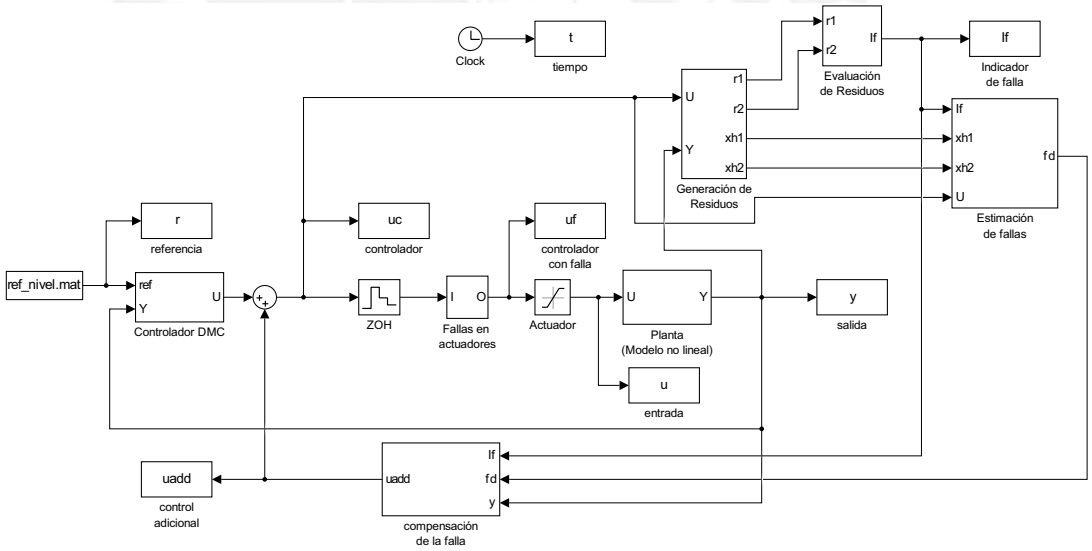

Figura B.23: Controlador DMC con compensación de fallas.

ANEXO C. PROGRAMAS EN RSLOGIX 5000

Rutinas del programa COMPENSACIÓN

Main()

```
JSR(Cond_ini);
JSR(MecanismoComp);
```

Cond_ini()

```
IF S:FS THEN // flag
// CONDICIONES INICIALES
uadd_1 := 0;
uadd_2 := 0;
// PARAMETROS
JSR(Param);
END_IF;
```

MecanismoComp()

```
// MECANISMO DE COMPENSACION DE FALLAS
x3 := h3 - 1.6;
x4 := h4 - 1.4;

xs[0] := x3; // VARIABLES SECUNDARIAS
xs[1] := x4;

// COMPENSACION FALLA EN BOMBA 1
IF I_f2 = 1 THEN // FALLA EN BOMBA 1
uadd_1 := -(Bpi[0,0]*Apcl[0,0]+Bpi[0,1]*Apcl[0,1])*xs[0] - ...
     (Bpi[0,0]*Apcl[0,1]+Bpi[0,1]*Apcl[1,1])*xs[1] - ...
     (Bpi[0,0]*Fap1[0]+Bpi[0,1]*Fap1[1])*fa1;
uadd_2 := 0;
// COMPENSACION FALLA EN BOMBA 2
ELSIF I_f3 = 1 THEN // FALLA EN BOMBA 2
uadd_1 := 0;
uadd_2 := -(Bpi[1,0]*Apcl[0,0]+Bpi[1,1]*Apcl[1,0])*xs[0] - ...
     (Bpi[1,0]*Apcl[0,1]+Bpi[1,1]*Apcl[1,1])*xs[1] - ...
     (Bpi[1,0]*Fap2[0]+Bpi[1,1]*Fap2[1])*fa2;
ELSE // NO HAY FALLA
```

```
17 uadd_1 := 0;
18 uadd_2 := 0;
19 END_IF;
```

Param()

```
1 // PARAMETROS
2 Fap1[0] := 0.0083;
3 Fap1[1] := 0;
4 Fap2[0] := 0;
5 Fap2[1] := 0.0063;
6 Apcl[0,0] := -0.0005;   Apcl[0,1] := 0.0004;
7 Apcl[1,0] := -0.0063;   Apcl[1,1] := 0.0097;
8 Bpi[0,0] := 120.2161;   Bpi[0,1] := -0.2030;
9 Bpi[1,0] := -0.0978;    Bpi[1,1] := 159.2923;
```

Rutinas del programa CONTROLADOR (por realimentación de estados)

Main()

```
1 JSR(Cond_ini);
2 JSR(RealimEstados);
```

Cond_ini()

```
1  IF S:FS THEN //flag
2      // CONDICIONES INICIALES
3      z1 := 0; // integrador 1
4      z2 := 0; // integrador 2
5      // SETPOINTS
6      h1sp := 12.4;
7      h2sp := 12.7;
8      // FALLAS EN LAS BOMBAS
9      alfa1 := 1.0;
10     alfa2 := 1.0;
11     // CONTROLADOR EN MODO MANUAL
12     v1m := 3.0;
13     v2m := 3.0;
14     // PARAMETROS DEL CONTROLADOR
15     JSR(Param);
16 END_IF;
```

RealimenEstados()

```
1 // ALGORITMO DE CONTROL
2 IF MODO THEN // AUTOMATICO
3     x1 := h1 - 12.4;
```

```
    x2 := h2 - 12.7;
    x3 := h3 - 1.6;
    x4 := h4 - 1.4;

    z1 := z1 + Ts*(h1sp - h1);
    z2 := z2 + Ts*(h2sp - h2);

    IF MODO2 THEN // CONTROL TOLERANTE
        u1 := -K[0,0]*x1 - K[0,1]*x2 - K[0,2]*x3 - K[0,3]*x4 - ...
            K[0,4]*z1 - K[0,5]*z2 + uadd_1;
        u2 := -K[1,0]*x1 - K[1,1]*x2 - K[1,2]*x3 - K[1,3]*x4 - ...
            K[1,4]*z1 - K[1,5]*z2 + uadd_2;
    ELSE // CONTROL NOMINAL
        u1 := -K[0,0]*x1 - K[0,1]*x2 - K[0,2]*x3 - K[0,3]*x4 - ...
            K[0,4]*z1 - K[0,5]*z2;
        u2 := -K[1,0]*x1 - K[1,1]*x2 - K[1,2]*x3 - K[1,3]*x4 - ...
            K[1,4]*z1 - K[1,5]*z2;
    END_IF;

    v1 := u1 + 3;
    v2 := u2 + 3;

ELSE // MANUAL
    v1 := v1m;
    v2 := v2m;
END_IF;
```

Param()

```
// GANANCIAS DEL CONTROLADOR
K[0,0] := 3.5168;
K[0,1] := -0.0465;
K[0,2] := 0.5328;
K[0,3] := -0.0013;
K[0,4] := -0.2770;
K[0,5] := -0.0127;
K[1,0] := -0.4611;
K[1,1] := 4.9645;
K[1,2] := -0.0007;
K[1,3] := 0.5206;
K[1,4] := 0.0878;
K[1,5] := -0.4102;

Ts := 0.1; // TIEMPO DE MUESTREO
```

Rutinas del programa CONTROLADOR (DMC)

Main()

```
JSR(Cond_ini);
JSR(DMC);
```

Cond_ini()

```
IF S:FS THEN // FLAG
    // CONDICIONES INICIALES
    u1_ant := 0; // ENTRADA DE CONTROL ANTERIOR
    u2_ant := 0; // ENTRADA DE CONTROL ANTERIOR
    // SETPOINTS
    h1sp := 12.4;
    h2sp := 12.7;
    // CONTROLADOR EN MODO MANUAL
    v1m := 3.0;
    v2m := 3.0;

    // PARAMETROS CONTROLADOR DMC
    Pr := 37; // HORIZONTE DE PREDICCION
    Nu := 2; // HORIZONTE DE CONTROL

    alf1 := 0.0; // PESO DE LA REFERENCIA INTERNA
    alf2 := 0.0; // PESO DE LA REFERENCIA INTERNA

    N11:=93;
    N12 := 98;
    N21 := 121;
    N22 := 117;
    Tsc := 4; // TIEMPO DE MUESTREO CONTROLADOR DMC

    ////////////////////////////////////////
    // COEFICIENTES DE LA RESPUESTA PASO
    ////////////////////////////////////////
    // HORIZONTES DEL MODELO
    Δ := 0.1; // PASO DE LA SIMULACION DE LA RESPUESTA PASO

    nsc := Tsc*10-1; // PASOS PARA EL MUESTREO (1/0.1 = 10)

    // PARAMETROS DEL MODELO DE CUATRO TANQUES
    aa1 := 28; aa2 := 32; aa3 := 28; aa4 := 32;
    a1 := 0.071; a2 := 0.057; a3 := 0.071; a4 := 0.057;
    k1 := 3.35; k2 := 3.33;
    g1 := 0.7; g2 := 0.6; g := 981;

    //g11
    v1k := 3 + 1; v2k := 3;
    h1k := 12.4; h2k := 12.7; h3k:=1.6; h4k:=1.4;
    jj:=0; tk:=0;
    tf_11 := N11*Tsc*10;
    FOR tt:=0 TO tf_11 DO
        h1k := h1k + Δ...
            *(-a1/aa1*SQRT(2*g*h1k)+a3/aa1*SQRT(2*g*h3k)+g1*k1/aa1*v1k);
        h2k := h2k + Δ...
```

```
            *(-a2/aa2*SQRT(2*g*h2k)+a4/aa2*SQRT(2*g*h4k)+g2*k2/aa2*v2k);
47          h3k := h3k + Δ*(-a3/aa3*SQRT(2*g*h3k)+(1-g2)*k2/aa3*v2k);
48          h4k := h4k + Δ*(-a4/aa4*SQRT(2*g*h4k)+(1-g1)*k1/aa4*v1k);
49          tk := tk + 1;
50          IF tk>nsc THEN
51              g11[jj] := h1k - 12.4;
52              jj := jj + 1;
53              tk := 0;
54          END_IF;
55      END_FOR;
56
57      //g12
58      v1k := 3; v2k := 3+1;
59      h1k := 12.4; h2k := 12.7; h3k:=1.6; h4k:=1.4;
60      jj:=0; tk:=0;
61      tf_12 := N12*Tsc*10;
62      FOR tt:=0 TO tf_12 DO
63          h1k := h1k + Δ...
            *(-a1/aa1*SQRT(2*g*h1k)+a3/aa1*SQRT(2*g*h3k)+g1*k1/aa1*v1k);
64          h2k := h2k + Δ...
            *(-a2/aa2*SQRT(2*g*h2k)+a4/aa2*SQRT(2*g*h4k)+g2*k2/aa2*v2k);
65          h3k := h3k + Δ*(-a3/aa3*SQRT(2*g*h3k)+(1-g2)*k2/aa3*v2k);
66          h4k := h4k + Δ*(-a4/aa4*SQRT(2*g*h4k)+(1-g1)*k1/aa4*v1k);
67          tk := tk + 1;
68          IF tk>nsc THEN
69              g12[jj] := h1k - 12.4;
70              jj := jj + 1;
71              tk := 0;
72          END_IF;
73      END_FOR;
74
75      //g21
76      v1k := 3+1; v2k := 3;
77      h1k := 12.4; h2k := 12.7; h3k:=1.6; h4k:=1.4;
78      jj:=0; tk:=0;
79      tf_21 := N21*Tsc*10;
80      FOR tt:=0 TO tf_21 DO
81          h1k := h1k + Δ...
            *(-a1/aa1*SQRT(2*g*h1k)+a3/aa1*SQRT(2*g*h3k)+g1*k1/aa1*v1k);
82          h2k := h2k + Δ...
            *(-a2/aa2*SQRT(2*g*h2k)+a4/aa2*SQRT(2*g*h4k)+g2*k2/aa2*v2k);
83          h3k := h3k + Δ*(-a3/aa3*SQRT(2*g*h3k)+(1-g2)*k2/aa3*v2k);
84          h4k := h4k + Δ*(-a4/aa4*SQRT(2*g*h4k)+(1-g1)*k1/aa4*v1k);
85          tk := tk + 1;
86          IF tk>nsc THEN
87              g21[jj] := h2k - 12.7;
88              jj := jj + 1;
89              tk := 0;
90          END_IF;
91      END_FOR;
92      //g22
93      v1k := 3; v2k := 3+1;
```

```
94      h1k := 12.4; h2k := 12.7; h3k:=1.6; h4k:=1.4;
95      jj:=0; tk:=0;
96      tf_22 := N22*Tsc*10;
97      FOR tt:=0 TO tf_22 DO
98          h1k := h1k + Δ...
                *(-a1/aa1*SQRT(2*g*h1k)+a3/aa1*SQRT(2*g*h3k)+g1*k1/aa1*v1k);
99          h2k := h2k + Δ...
                *(-a2/aa2*SQRT(2*g*h2k)+a4/aa2*SQRT(2*g*h4k)+g2*k2/aa2*v2k);
100         h3k := h3k + Δ*(-a3/aa3*SQRT(2*g*h3k)+(1-g2)*k2/aa3*v2k);
101         h4k := h4k + Δ*(-a4/aa4*SQRT(2*g*h4k)+(1-g1)*k1/aa4*v1k);
102         tk := tk + 1;
103         IF tk>nsc THEN
104             g22[jj] := h2k - 12.7;
105             jj := jj + 1;
106             tk := 0;
107         END_IF;
108     END_FOR;
109     ////////////////////////////////////////////
110     // RESPUESTA LIBRE, VECTOR CON LAS gk+i-gi
111     ////////////////////////////////////////////
112     // vector dg11
113     FOR ik := 0 TO (N11-1) DO
114         FOR jk := 0 TO (Pr-1) DO
115             IF (ik+jk)>(N11-2) THEN
116                 dg11[jk,ik] := g11[N11-1] - g11[ik];
117             ELSE
118                 dg11[jk,ik] := g11[jk+ik+1] - g11[ik];
119             END_IF;
120         END_FOR;
121     END_FOR;
122
123     // vector dg12
124     FOR ik := 0 TO (N12-1) DO
125         FOR jk := 0 TO (Pr-1) DO
126             IF (ik+jk)>(N12-2) THEN
127                 dg12[jk,ik] := g12[N12-1] - g12[ik];
128             ELSE
129                 dg12[jk,ik] := g12[jk+ik+1] - g12[ik];
130             END_IF;
131         END_FOR;
132     END_FOR;
133
134     // vector dg21
135     FOR ik := 0 TO (N21-1) DO
136         FOR jk := 0 TO (Pr-1) DO
137             IF (ik+jk)>(N21-2) THEN
138                 dg21[jk,ik] := g21[N21-1] - g21[ik];
139             ELSE
140                 dg21[jk,ik] := g21[jk+ik+1] - g21[ik];
141             END_IF;
142         END_FOR;
143     END_FOR;
```

```
144
145     // vector dg22
146     FOR ik := 0 TO (N22-1) DO
147         FOR jk := 0 TO (Pr-1) DO
148             IF (ik+jk)>(N22-2) THEN
149                 dg22[jk,ik] := g22[N22-1] - g22[ik];
150             ELSE
151                 dg22[jk,ik] := g22[jk+ik+1] - g22[ik];
152             END_IF;
153         END_FOR;
154     END_FOR;
155
156     // GANANCIAS DEL CONTROLADOR DMC
157     JSR(Param);
158 END_IF;
```

DMC()

```
1  // REFERENCIA INTERNA
2  FOR j := 0 TO (Pr-1) DO
3      IF j < 1 THEN
4      r1[j] := alf1*h1 + (1-alf1)*h1sp;
5      r2[j] := alf2*h2 + (1-alf2)*h2sp;
6      ELSE
7      r1[j] := alf1*r1[j-1] + (1-alf1)*h1sp;
8      r2[j] := alf2*r2[j-1] + (1-alf2)*h2sp;
9      END_IF;
10 END_FOR;
11
12 FOR j := 0 TO Pr-1 DO
13     r[j] := r1[j];
14 END_FOR;
15 FOR j := 0 TO Pr-1 DO
16     r[j+Pr] := r2[j];
17 END_FOR;
18
19 // RESPUESTA LIBRE
20
21 // dg11*dup1
22 FOR i:= 0 TO (Pr-1) DO
23     suma := 0;
24     FOR j := 0 TO (N11-1) DO
25         suma := suma + dg11[i,j]*dup1[j];
26     END_FOR;
27     dg11dup1[i] := suma;
28 END_FOR;
29
30 // dg12*dup2
31 FOR i:= 0 TO (Pr-1) DO
32     suma := 0;
33     FOR j := 0 TO (N12-1) DO
34         suma := suma + dg12[i,j]*dup2[j];
```

```
35      END_FOR;
36      dg12dup2[i] := suma;
37  END_FOR;
38
39  // dg21*dup1
40  FOR i:= 0 TO (Pr-1) DO
41      suma := 0;
42      FOR j := 0 TO (N21-1) DO
43          suma := suma + dg21[i,j]*dup1[j];
44      END_FOR;
45      dg21dup1[i] := suma;
46  END_FOR;
47
48  // dg22*dup2
49  FOR i:= 0 TO (Pr-1) DO
50      suma := 0;
51      FOR j := 0 TO (N22-1) DO
52          suma := suma + dg22[i,j]*dup2[j];
53      END_FOR;
54      dg22dup2[i] := suma;
55  END_FOR;
56
57  // RESPUESTA LIBRE
58  FOR i:=0 TO (Pr-1) DO
59      f1[i] := h1 + dg11dup1[i] + dg12dup2[i];
60      f2[i] := h2 + dg21dup1[i] + dg22dup2[i];
61  END_FOR;
62
63  FOR j := 0 TO Pr-1 DO
64      f[j] := f1[j];
65  END_FOR;
66  FOR j := 0 TO Pr-1 DO
67      f[j+Pr] := f2[j];
68  END_FOR;
69
70  // ALGORITMO DE CONTROL
71
72  IF MODO THEN // AUTOMATICO
73      // u1
74      suma1 := 0;
75      FOR j := 0 TO (2*Pr-1) DO
76      suma1 := suma1 + Kc1[0,j]*(r[j]-f[j]);
77      END_FOR;
78      u1 := suma1 + u1_ant; // ENTRADA DE CONTROL
79      u1_ant := u1; // ACTUALIZACION ENTRADA DE CONTROL ANTERIOR
80
81      FOR j:=0 TO N21-2 DO
82          dup1[N21-1-j] := dup1[N21-2-j]; // ACTUALIZACION ENTRADAS DE ...
                CONTROL PASADAS
83      END_FOR;
84      dup1[0] := suma1;
85
```

```
86      // u2
87      suma2 := 0;
88      FOR j := 0 TO (2*Pr-1) DO
89      suma2 := suma2 + Kc1[1,j]*(r[j]-f[j]);
90      END_FOR;
91      u2 := suma2 + u2_ant; // ENTRADA DE CONTROL
92      u2_ant := u2; // ACTUALIZACION ENTRADA DE CONTROL ANTERIOR
93
94      FOR j:=0 TO N21-2 DO
95          dup2[N21-1-j] := dup2[N21-2-j]; // ACTUALIZACION ENTRADAS DE ...
                CONTROL PASADAS
96      END_FOR;
97      dup2[0] := suma2;
98
99      IF MODO2 THEN // COMPENSACION
100         u1 := u1 + uadd_1;
101         u2 := u2 + uadd_2;
102     END_IF;
103     v1 := u1 + 3;
104     v2 := u2 + 3;
105
106 ELSE // MANUAL
107     v1 := v1m;
108     v2 := v2m;
109 END_IF;
```

Param()

```
1  // GANANCIAS DEL CONTROLADOR DMC
2
3  Kc1[0,0]:=0.0876;       Kc1[0,1]:=0.0823;   Kc1[0,2]:=0.0753;   ...
       Kc1[0,3]:= 0.0687;  Kc1[0,4]:=0.0626;   Kc1[0,5]:=0.0569;
4  Kc1[0,6]:=0.0516;       Kc1[0,7]:=0.0466;   Kc1[0,8]:=0.0420;   ...
       Kc1[0,9]:= 0.0376;  Kc1[0,10]:= 0.0335; Kc1[0,11]:=0.0297;
5  Kc1[0,12]:=0.0261;      Kc1[0,13]:=0.0227;  Kc1[0,14]:=0.0195;  ...
       Kc1[0,15]:=0.0165;  Kc1[0,16]:=0.0137;  Kc1[0,17]:=0.0110;
6  Kc1[0,18]:=0.0085;      Kc1[0,19]:=0.0062;  Kc1[0,20]:=0.0039;  ...
       Kc1[0,21]:=0.0018; Kc1[0,22]:=-0.0002; Kc1[0,23]:=-0.0020;
7  Kc1[0,24]:=-0.0038;     Kc1[0,25]:=-0.0055; Kc1[0,26]:=-0.0070; ...
       Kc1[0,27]:=-0.0085; Kc1[0,28]:=-0.0099; Kc1[0,29]:=-0.0113;
8  Kc1[0,30]:= -0.0125;    Kc1[0,31]:=-0.0137; Kc1[0,32]:=-0.0148; ...
       Kc1[0,33]:=-0.0159; Kc1[0,34]:=-0.0169; Kc1[0,35]:=-0.0178;
9  Kc1[0,36]:=-0.0187;     Kc1[0,37]:=-0.0038; Kc1[0,38]:=-0.0003; ...
       Kc1[0,39]:=0.0026;  Kc1[0,40]:=0.0049;  Kc1[0,41]:=0.0065;
10 Kc1[0,42]:=0.0078;      Kc1[0,43]:=0.0086;  Kc1[0,44]:=0.0092;  ...
       Kc1[0,45]:=0.0095;  Kc1[0,46]:=0.0095;  Kc1[0,47]:=0.0094;
11 Kc1[0,48]:=0.0092;      Kc1[0,49]:=0.0088;  Kc1[0,50]:=0.0083;  ...
       Kc1[0,51]:=0.0078;  Kc1[0,52]:=0.0071;  Kc1[0,53]:=0.0064;
12 Kc1[0,54]:=0.0057;      Kc1[0,55]:=0.0049;  Kc1[0,56]:=0.0042;  ...
       Kc1[0,57]:=0.0033;  Kc1[0,58]:=0.0025;  Kc1[0,59]:=0.0017;
13 Kc1[0,60]:=0.0009;      Kc1[0,61]:=0.0000;  Kc1[0,62]:=-0.0008; ...
       Kc1[0,63]:=-0.0016; Kc1[0,64]:=-0.0024; Kc1[0,65]:=-0.0032;
```

```
Kc1[0,66]:=-0.0040;     Kc1[0,67]:=-0.0048; Kc1[0,68]:=-0.0055; ...
    Kc1[0,69]:=-0.0063; Kc1[0,70]:=-0.0070; Kc1[0,71]:=-0.0077;
Kc1[0,72]:=-0.0084;     Kc1[0,73]:=-0.0091;

Kc1[1,0]:=-0.0079;      Kc1[1,1]:=0.0009;   Kc1[1,2]:=0.0076;   ...
    Kc1[1,3]:=0.0124;   Kc1[1,4]:=0.0157;   Kc1[1,5]:=0.0178;
Kc1[1,6]:=0.0191;       Kc1[1,7]:=0.0197;   Kc1[1,8]:=0.0197;   ...
    Kc1[1,9]:=0.0193;   Kc1[1,10]:=0.0185;  Kc1[1,11]:=0.0174;
Kc1[1,12]:=0.0162;      Kc1[1,13]:=0.0148;  Kc1[1,14]:=0.0133;  ...
    Kc1[1,15]:=0.0117;  Kc1[1,16]:=0.0101;  Kc1[1,17]:=0.0084;
Kc1[1,18]:=0.0068;      Kc1[1,19]:=0.0051;  Kc1[1,20]:=0.0034;  ...
    Kc1[1,21]:=0.0018;  Kc1[1,22]:=0.0002;  Kc1[1,23]:=-0.0013;
Kc1[1,24]:=-0.0028;     Kc1[1,25]:=-0.0043; Kc1[1,26]:=-0.0057; ...
    Kc1[1,27]:=-0.0070; Kc1[1,28]:=-0.0084; Kc1[1,29]:=-0.0096;
Kc1[1,30]:=-0.0108;     Kc1[1,31]:=-0.0120; Kc1[1,32]:=-0.0131; ...
    Kc1[1,33]:=-0.0141; Kc1[1,34]:=-0.0151; Kc1[1,35]:=-0.0161;
Kc1[1,36]:=-0.0170;     Kc1[1,37]:=0.0824;  Kc1[1,38]:=0.0788;  ...
    Kc1[1,39]:=0.0733;  Kc1[1,40]:=0.0680;  Kc1[1,41]:=0.0631;
Kc1[1,42]:=0.0583;      Kc1[1,43]:=0.0538;  Kc1[1,44]:=0.0495;  ...
    Kc1[1,45]:=0.0454;  Kc1[1,46]:=0.0415;  Kc1[1,47]:=0.0377;
Kc1[1,48]:=0.0341;      Kc1[1,49]:=0.0307;  Kc1[1,50]:=0.0274;  ...
    Kc1[1,51]:=0.0243;  Kc1[1,52]:=0.0213;  Kc1[1,53]:=0.0184;
Kc1[1,54]:=0.0156;      Kc1[1,55]:=0.0130;  Kc1[1,56]:=0.0105;  ...
    Kc1[1,57]:=0.0080;  Kc1[1,58]:=0.0057;  Kc1[1,59]:=0.0035;
Kc1[1,60]:=0.0013;      Kc1[1,61]:=-0.0007; Kc1[1,62]:=-0.0027; ...
    Kc1[1,63]:=-0.0046; Kc1[1,64]:=-0.0064; Kc1[1,65]:=-0.0082;
Kc1[1,66]:=-0.0099;     Kc1[1,67]:=-0.0115; Kc1[1,68]:=-0.0131; ...
    Kc1[1,69]:=-0.0145; Kc1[1,70]:=-0.0160; Kc1[1,71]:=-0.0174;
Kc1[1,72]:=-0.0187;     Kc1[1,73]:=-0.02;
```

Rutinas del programa DIAGNÓSTICO

Main()

```
JSR(Cond_ini);
JSR(UIO);
```

Cond_ini()

```
IF S:FS THEN // FLAG
    // CONDICIONES INICIALES
    w11 := 0;
    w21 := 0;
    w31 := 0;
    w41 := 0;

    w12 := 0;
    w22 := 0;
    w32 := 0;
```

```
    w42 := 0;

    // EVALUACION DE RESIDUOS
    N := 40; // VENTANDA DE MUESTRAS
    cont := 0;
    rms_1 := 0;
    rms_2 := 0;
    FOR i := 0 TO (N-1) DO
        norm_r1_v[i] := 0;
        norm_r2_v[i] := 0;
    END_FOR;

    // LOGICA
    cont_aux := 10; // CONTADOR AUXILIAR
    cont1 := 0;
    cont2 := 0;

    // UMBRALES
    umb1 := 0.05;
    umb2 := 0.05;

    // MATRIZ DE FIRMA DE FALLAS
    MFF[0,0] := 0; MFF[1,0] := 0; // NO HAY FALLA
    MFF[0,1] := 0; MFF[1,1] := 1; // FALLA EN BOMBA 1
    MFF[0,2] := 1; MFF[1,2] := 0; // FALLA EN BOMBA 2
    MFF[0,3] := 1; MFF[1,3] := 1; // FALLA EN AMBAS BOMBAS

    // ESTIMACION DE FALLAS
    xb11d[0] := 0; xb11d[1] := 0;
    xb21d[0] := 0; xb21d[1] := 0;
    xb31d[0] := 0; xb31d[1] := 0;
    xb41d[0] := 0; xb41d[1] := 0;

    xb12d[0] := 0; xb12d[1] := 0;
    xb22d[0] := 0; xb22d[1] := 0;
    xb32d[0] := 0; xb32d[1] := 0;
    xb42d[0] := 0; xb42d[1] := 0;

    u1d[0] := 0; u1d[1] := 0;
    u2d[0] := 0; u2d[1] := 0;

    // PARAMETROS UIO
    JSR(Param);
END_IF;
```

UIO()

```
// GENERACION DE RESIDUOS
x1 := h1 - 12.4;
x2 := h2 - 12.7;
x3 := h3 - 1.6;
x4 := h4 - 1.4;
```

```
6
7  u1 := v1 - 3;
8  u2 := v2 - 3;
9
10 y1 := x1;
11 y2 := x2;
12 y3 := x3;
13 y4 := x4;
14
15 // UIO 1
16
17 xh11 := w11 + HH1[0,0]*y1 + HH1[0,1]*y2 + HH1[0,2]*y3 + HH1[0,3]*y4;
18 xh21 := w21 + HH1[1,0]*y1 + HH1[1,1]*y2 + HH1[1,2]*y3 + HH1[1,3]*y4;
19 xh31 := w31 + HH1[2,0]*y1 + HH1[2,1]*y2 + HH1[2,2]*y3 + HH1[2,3]*y4;
20 xh41 := w41 + HH1[3,0]*y1 + HH1[3,1]*y2 + HH1[3,2]*y3 + HH1[3,3]*y4;
21
22 w11 := E1[0,0]*w11 + E1[0,1]*w21 + E1[0,2]*w31 + E1[0,3]*w41 + ...
       TB1[0,0]*u1 + TB1[0,1]*u2 + K1[0,0]*y1 + K1[0,1]*y2 + K1[0,2]*y3 ...
       + K1[0,3]*y4;
23 w21 := E1[1,0]*w11 + E1[1,1]*w21 + E1[1,2]*w31 + E1[1,3]*w41 + ...
       TB1[1,0]*u1 + TB1[1,1]*u2 + K1[1,0]*y1 + K1[1,1]*y2 + K1[1,2]*y3 ...
       + K1[1,3]*y4;
24 w31 := E1[2,0]*w11 + E1[2,1]*w21 + E1[2,2]*w31 + E1[2,3]*w41 + ...
       TB1[2,0]*u1 + TB1[2,1]*u2 + K1[2,0]*y1 + K1[2,1]*y2 + K1[2,2]*y3 ...
       + K1[2,3]*y4;
25 w41 := E1[3,0]*w11 + E1[3,1]*w21 + E1[3,2]*w31 + E1[3,3]*w41 + ...
       TB1[3,0]*u1 + TB1[3,1]*u2 + K1[3,0]*y1 + K1[3,1]*y2 + K1[3,2]*y3 ...
       + K1[3,3]*y4;
26
27
28 r11 := y1 - xh11;
29 r21 := y2 - xh21;
30 r31 := y3 - xh31;
31 r41 := y4 - xh41;
32
33 norm_r1 := r11*r11 + r21*r21 + r31*r31 + r41*r41;
34
35 // UIO 2
36
37 xh12 := w12 + HH2[0,0]*y1 + HH2[0,1]*y2 + HH2[0,2]*y3 + HH2[0,3]*y4;
38 xh22 := w22 + HH2[1,0]*y1 + HH2[1,1]*y2 + HH2[1,2]*y3 + HH2[1,3]*y4;
39 xh32 := w32 + HH2[2,0]*y1 + HH2[2,1]*y2 + HH2[2,2]*y3 + HH2[2,3]*y4;
40 xh42 := w42 + HH2[3,0]*y1 + HH2[3,1]*y2 + HH2[3,2]*y3 + HH2[3,3]*y4;
41
42 w12 := E2[0,0]*w12 + E2[0,1]*w22 + E2[0,2]*w32 + E2[0,3]*w42 + ...
       TB2[0,0]*u1 + TB2[0,1]*u2 + K2[0,0]*y1 + K2[0,1]*y2 + K2[0,2]*y3 ...
       + K2[0,3]*y4;
43 w22 := E2[1,0]*w12 + E2[1,1]*w22 + E2[1,2]*w32 + E2[1,3]*w42 + ...
       TB2[1,0]*u1 + TB2[1,1]*u2 + K2[1,0]*y1 + K2[1,1]*y2 + K2[1,2]*y3 ...
       + K2[1,3]*y4;
44 w32 := E2[2,0]*w12 + E2[2,1]*w22 + E2[2,2]*w32 + E2[2,3]*w42 + ...
       TB2[2,0]*u1 + TB2[2,1]*u2 + K2[2,0]*y1 + K2[2,1]*y2 + K2[2,2]*y3 ...
```

```
    + K2[2,3]*y4;
w42 := E2[3,0]*w12 + E2[3,1]*w22 + E2[3,2]*w32 + E2[3,3]*w42 + ...
    TB2[3,0]*u1 + TB2[3,1]*u2 + K2[3,0]*y1 + K2[3,1]*y2 + K2[3,2]*y3 ...
    + K2[3,3]*y4;

r12 := y1 - xh12;
r22 := y2 - xh22;
r32 := y3 - xh32;
r42 := y4 - xh42;

norm_r2 := r12*r12 + r22*r22 + r32*r32 + r42*r42;

// EVALUACION DE RESIDUOS (VALOR RMS)

norm_r1_v[0] := norm_r1;
norm_r2_v[0] := norm_r2;

IF cont < N THEN
    cont := cont + 1;
    FOR i := 0 TO (N-1) DO
        rms_1 := rms_1 + norm_r1_v[i];
        rms_2 := rms_2 + norm_r2_v[i];
    END_FOR;
    rms_1 := SQRT(rms_1 / cont);
    rms_2 := SQRT(rms_2 / cont);
ELSE
    FOR j := 0 TO (N-1) DO
            rms_1 := rms_1 + norm_r1_v[j];
            rms_2 := rms_2 + norm_r2_v[j];
    END_FOR;
    rms_1 := SQRT(rms_1 / N);
    rms_2 := SQRT(rms_2 / N);
END_IF;

FOR jj := 0 TO (N-2) DO
    norm_r1_v[N-1-jj] := norm_r1_v[N-2-jj];
    norm_r2_v[N-1-jj] := norm_r2_v[N-2-jj];
END_FOR;

// LOGICA

IF rms_1 > umb1 THEN
    cont1 := cont1 + 1;
    IF cont1 > cont_aux THEN
        Sr_1 := 1;
    ELSE
        Sr_1 := 0;
    END_IF;
ELSE
    Sr_1 := 0;
    cont1 := 0;
END_IF;
```

```
94
95  IF rms_2 > umb2 THEN
96      cont2 := cont2 + 1;
97      IF cont2 > cont_aux THEN
98          Sr_2 := 1;
99      ELSE
100         Sr_2 := 0;
101     END_IF;
102 ELSE
103     Sr_2 := 0;
104     cont2 := 0;
105 END_IF;
106
107 // DECISION
108
109 IF Sr_1 = MFF[0,0] & Sr_2 = MFF[1,0] THEN
110     I_f1 := 1; // NO HAY FALLAS
111 ELSE
112     I_f1 := 0;
113 END_IF;
114
115 IF Sr_1 = MFF[0,1] & Sr_2 = MFF[1,1] THEN
116     I_f2 := 1; // FALLA EN BOMBA 1
117 ELSE
118     I_f2 := 0;
119 END_IF;
120
121 IF Sr_1 = MFF[0,2] & Sr_2 = MFF[1,2] THEN
122     I_f3 := 1; // FALLA EN BOMBA 2
123 ELSE
124     I_f3 := 0;
125 END_IF;
126
127 IF Sr_1 = MFF[0,3] & Sr_2 = MFF[1,3] THEN
128     I_f4 := 1; // FALLA EN AMBAS BOMBAS
129 ELSE
130     I_f4 := 0;
131 END_IF;
132
133 // ESTIMACION DE FALLAS
134 // ESTIMACION FALLA EN BOMBA 1
135 xb11 := Ui_1[0,0]*xh11 + Ui_1[0,1]*xh21 + Ui_1[0,2]*xh31 + ...
        Ui_1[0,3]*xh41;
136 xb21 := Ui_1[1,0]*xh11 + Ui_1[1,1]*xh21 + Ui_1[1,2]*xh31 + ...
        Ui_1[1,3]*xh41;
137 xb31 := Ui_1[2,0]*xh11 + Ui_1[2,1]*xh21 + Ui_1[2,2]*xh31 + ...
        Ui_1[2,3]*xh41;
138 xb41 := Ui_1[3,0]*xh11 + Ui_1[3,1]*xh21 + Ui_1[3,2]*xh31 + ...
        Ui_1[3,3]*xh41;
139
140 xb11d[1] := xb11d[0]; // xb1(k)
141 xb11d[0] := xb11; // xb1(k-1)
```

```
xb21d[1] := xb21d[0];
xb21d[0] := xb21;
xb31d[1] := xb31d[0];
xb31d[0] := xb31;
xb41d[1] := xb41d[0];
xb41d[0] := xb41;

u1d[1] := u1d[0]; // u1(k)
u1d[0] := u1; // u1(k-1)
u2d[1] := u2d[0];
u2d[0] := u2;

IF I_f2 = 1 THEN // FALLA EN BOMBA 1
    fa1 := VRi_1*(xb11d[0] - Ab11_1*xb11d[1] - (Ab12_1[0]*xb21d[1] + ...
        Ab12_1[1]*xb31d[1] + Ab12_1[2]*xb41d[1]) - (Bb1_1[0]*u1d[1] + ...
        Bb1_1[1]*u2d[1]));
ELSE
    fa1 := 0;
END_IF;


// ESTIMACION FALLA EN BOMBA 2
xb12 := Ui_2[0,0]*xh12 + Ui_2[0,1]*xh22 + Ui_2[0,2]*xh32 + ...
    Ui_2[0,3]*xh42;
xb22 := Ui_2[1,0]*xh12 + Ui_2[1,1]*xh22 + Ui_2[1,2]*xh32 + ...
    Ui_2[1,3]*xh42;
xb32 := Ui_2[2,0]*xh12 + Ui_2[2,1]*xh22 + Ui_2[2,2]*xh32 + ...
    Ui_2[2,3]*xh42;
xb42 := Ui_2[3,0]*xh12 + Ui_2[3,1]*xh22 + Ui_2[3,2]*xh32 + ...
    Ui_2[3,3]*xh42;

xb12d[1] := xb12d[0]; // xb1(k)
xb12d[0] := xb12; // xb1(k-1)
xb22d[1] := xb22d[0];
xb22d[0] := xb22;
xb32d[1] := xb32d[0];
xb32d[0] := xb32;
xb42d[1] := xb42d[0];
xb42d[0] := xb42;

IF I_f3 = 1 THEN // FALLA EN BOMBA 2
    fa2 := VRi_2*(xb12d[0] - Ab11_2*xb12d[1] - (Ab12_2[0]*xb22d[1] + ...
        Ab12_2[1]*xb32d[1] + Ab12_2[2]*xb42d[1]) - (Bb1_2[0]*u1d[1] + ...
        Bb1_2[1]*u2d[1]));
ELSE
    fa2 := 0;
END_IF;
```

Param()

```
//////////////////////
// UIO 1
```

```
3  ////////////////////
4  E1[0,0] := 0.9319;  E1[0,1] := 0.0790;  E1[0,2] := 0;       E1[0,3] ...
       := 0;
5  E1[1,0] := -0.0790; E1[1,1] := 0.9319;  E1[1,2] := 0;       E1[1,3] ...
       := 0;
6  E1[2,0] := 0;       E1[2,1] := 0;       E1[2,2] := 0.9116;  E1[2,3] ...
       := 0.0252;
7  E1[3,0] := 0;       E1[3,1] := 0;       E1[3,2] := -0.0252; E1[3,3] ...
       := 0.9116;
8
9  TB1[0,0] := 0;      TB1[0,1] := 0;
10 TB1[1,0] := 0;      TB1[1,1] := 0.0063;
11 TB1[2,0] := 0;      TB1[2,1] := 0.0048;
12 TB1[3,0] := 0;      TB1[3,1] := 0;
13
14 K1[0,0] := 0.0082;  K1[0,1] := -0.0790; K1[0,2] := 0.0005;  K1[0,3] ...
       := -0.0213;
15 K1[1,0] := 0.0097;  K1[1,1] := 0.0669;  K1[1,2] := 0;       K1[1,3] ...
       := -0.0227;
16 K1[2,0] := 0.0083;  K1[2,1] := 0;       K1[2,2] := 0.0839;  K1[2,3] ...
       := -0.0221;
17 K1[3,0] := -0.0285; K1[3,1] := 0;       K1[3,2] := 0.0237;  K1[3,3] ...
       := 0.0746;
18
19 HH1[0,0] := 0.8769; HH1[0,1] := 0.0005; HH1[0,2] := 0;      HH1[0,3] ...
       := 0.3286;
20 HH1[1,0] := 0.0005; HH1[1,1] := 0;      HH1[1,2] := 0;      HH1[1,3] ...
       := 0.0002;
21 HH1[2,0] := 0;      HH1[2,1] := 0;      HH1[2,2] := 0;      HH1[2,3] ...
       := 0;
22 HH1[3,0] := 0.3286; HH1[3,1] := 0.0002; HH1[3,2] := 0;      HH1[3,3] ...
       := 0.1231;
23
24 ////////////////////
25 // UIO 2
26 ////////////////////
27
28 E2[0,0] := 0.9319;  E2[0,1] := 0.0790;  E2[0,2] := 0;       E2[0,3] ...
       := 0;
29 E2[1,0] := -0.0790; E2[1,1] := 0.9319;  E2[1,2] := 0;       E2[1,3] ...
       := 0;
30 E2[2,0] := 0;       E2[2,1] := 0;       E2[2,2] := 0.9116;  E2[2,3] ...
       := 0.0252;
31 E2[3,0] := 0;       E2[3,1] := 0;       E2[3,2] := -0.0252; E2[3,3] ...
       := 0.9116;
32
33 TB2[0,0] := 0.0083; TB2[0,1] := 0;
34 TB2[1,0] := 0;      TB2[1,1] := 0;
35 TB2[2,0] := 0;      TB2[2,1] := 0;
36 TB2[3,0] := 0.0031; TB2[3,1] := 0;
37
38 K2[0,0] := 0.0666;  K2[0,1] := -0.0290;     K2[0,2] := 0.0424;  ...
```

```
        K2[0,3] := 0;
39  K2[1,0] := 0.0789;  K2[1,1] := 0.0245;      K2[1,2] := -0.0307; ...
        K2[1,3] := 0.0012;
40  K2[2,0] := -0.0001; K2[2,1] := -0.042;      K2[2,2] := 0.0532;  ...
        K2[2,3] := -0.0267;
41  K2[3,0] := 0;       K2[3,1] := -0.0121;     K2[3,2] := 0.0159;  ...
        K2[3,3] := 0.0851;
42
43  HH2[0,0] := 0;      HH2[0,1] := 0.0011; HH2[0,2] := 0.0008;     ...
        HH2[0,3] := 0;
44  HH2[1,0] := 0.0011; HH2[1,1] := 0.6335; HH2[1,2] := 0.4819;     ...
        HH2[1,3] := 0;
45  HH2[2,0] := 0.0008; HH2[2,1] := 0.4819; HH2[2,2] := 0.3665;     ...
        HH2[2,3] := 0;
46  HH2[3,0] := 0;      HH2[3,1] := 0;      HH2[3,2] := 0;          ...
        HH2[3,3] := 0;
47
48  ////////////////////////////
49  // ESTIMACION FALLA BOMBA 1
50  ////////////////////////////
51
52  Ui_1[0,0] := 0.9364;    Ui_1[0,1] := 0.0006;    Ui_1[0,2] := 0;     ...
        Ui_1[0,3] := 0.3509;
53  Ui_1[1,0] := -0.0006;   Ui_1[1,1] := 1.0;       Ui_1[1,2] := 0;     ...
        Ui_1[1,3] := -0.0001;
54  Ui_1[2,0] := 0;         Ui_1[2,1] := 0;         Ui_1[2,2] := 1.0;   ...
        Ui_1[2,3] := 0;
55  Ui_1[3,0] := -0.3509;   Ui_1[3,1] := -0.0001;   Ui_1[3,2] := 0;     ...
        Ui_1[3,3] := 0.9364;
56
57  Ab11_1 := 0.9982;
58  Ab12_1[0] := 0;     Ab12_1[1] := 0.0041;    Ab12_1[2] := -0.0005;
59
60  Bb1_1[0] := 0.0089;     Bb1_1[1] := 0;
61
62  VRi_1 := 112.5733;
63
64  ////////////////////////////
65  // ESTIMACION FALLA BOMBA 2
66  ////////////////////////////
67
68  Ui_2[0,0] := 0.0013;    Ui_2[0,1] := 0.7959;    Ui_2[0,2] := 0.6054; ...
          Ui_2[0,3] := 0;
69  Ui_2[1,0] := -0.7959;   Ui_2[1,1] := 0.3674;    Ui_2[1,2] := -0.4812; ...
         Ui_2[1,3] := 0;
70  Ui_2[2,0] := -0.6054;   Ui_2[2,1] := -0.4812;   Ui_2[2,2] := 0.6340; ...
          Ui_2[2,3] := 0;
71  Ui_2[3,0] := 0;         Ui_2[3,1] := 0;         Ui_2[3,2] := 0;     ...
           Ui_2[3,3] := 1;
72
73  Ab11_2 := 0.9977;
74  Ab12_2[0] := 0.0010;    Ab12_2[1] := -0.0013;   Ab12_2[2] := 0.0026;
```

```
75
76 Bb1_2[0] := 0;  Bb1_2[1] := 0.0079;
77
78 VRi_2 := 126.7833;
```

Rutinas del programa PLANTA

Main()

```
1 JSR(Cond_ini);
2 JSR(CuatroTanques);
```

Cond_ini()

```
1  IF S:FS THEN // FLAG
2      // CONDICIONES INICIALES
3      h1 := 12.4;
4      h2 := 12.7;
5      h3 := 1.6;
6      h4 := 1.4;
7      v1 := 3;
8      v2 := 3;
9      // FALLAS EN LAS BOMBAS
10     alfa1 := 1.0;
11     alfa2 := 1.0;
12     // PARAMETROS DE LA PLANTA
13     JSR(Param);
14 END_IF;
```

CuatroTanques()

```
1  v1f := alfa1*v1; //Falla en la bomba 1
2  v2f := alfa2*v2; //Falla en la bomba 2
3
4  IF v1f > 10 THEN
5      v1f := 10;
6  ELSIF v1f < 0 THEN
7      v1f := 0;
8  END_IF;
9
10 IF v2f > 10 THEN
11     v2f := 10;
12 ELSIF v2f < 0 THEN
13     v2f := 0;
14 END_IF;
15
16 // MODELO DEL PROCESO DE CUATRO TANQUES
17 h1 := h1 + dT*(-a1/aa1*SQRT(2*g*h1)+a3/aa1*SQRT(2*g*h3)+g1*k1/aa1*v1f);
```

```
h2 := h2 + dT*(-a2/aa2*SQRT(2*g*h2)+a4/aa2*SQRT(2*g*h4)+g2*k2/aa2*v2f);
h3 := h3 + dT*(-a3/aa3*SQRT(2*g*h3)+(1-g2)*k2/aa3*v2f);
h4 := h4 + dT*(-a4/aa4*SQRT(2*g*h4)+(1-g1)*k1/aa4*v1f);

IF h1 < 0 THEN
    h1 := 0;
ELSIF h1 > 20 THEN
    h1 := 20;
END_IF;

IF h2 < 0 THEN
    h2 := 0;
ELSIF h2 > 20 THEN
    h2 := 20;
END_IF;

IF h3 < 0 THEN
    h3 := 0;
ELSIF h3 > 20 THEN
    h3 := 20;
END_IF;

IF h4 < 0 THEN
    h4 := 0;
ELSIF h4 > 20 THEN
    h4 := 20;
END_IF;
```

Param()

```
//PARAMETROS DEL PROCESO
aa1 := 28;
aa2 := 32;
aa3 := 28;
aa4 := 32;
a1 := 0.071;
a2 := 0.057;
a3 := 0.071;
a4 := 0.057;
k1 := 3.35;
k2 := 3.33;
g1 := 0.7;
g2 := 0.6;
g := 981;
dT := 0.02; // PASO PARA APROX. EULER
```