ANEXO A: Generación de un Perfil de Carretera Aleatorio

En las Secciones 2 y 3, se hace uso de un perfil de carretera tipo ruido blanco. A continuación se detalla el procedimiento para generar dicho perfil.

Los perfiles que se toman a lo largo de una línea lateral muestran la superelevación y la corona del diseño de la carretera, además de la rutina y otras dificultades. Los perfiles longitudinales muestran el grado de diseño, la aspereza y la textura como se aprecia en la siguiente Figura (Sayers *et al*, 1998).

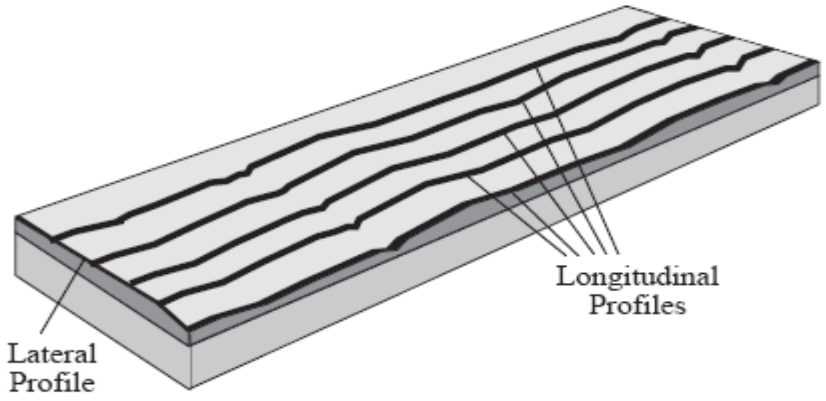

Figura A.1. Perfil de carretera (Sayers *et al*, 1998).

En este trabajo nos enfocamos en los perfiles longitudinales, de los cuales su clasificación está basada en la Organización Internacional de Normalización (ISO 8606). La ISO propuso la clasificación de la aspereza usando los valores de la densidad espectral de potencia (PSD).

Es bien sabido que una señal $z_r(t)$ que representa satisfactoriamente la aspereza de la mayoría de carreteras es la salida de un filtro lineal de primer orden cuya entrada es un ruido blanco $w(t)$.

La cantidad de excitación de carretera impuesta al neumático del vehículo depende de dos factores:

1) La aspereza del camino la cual es una función del coeficiente de aspereza de la carretera.

2) La velocidad del vehículo V.

El perfil de la carretera puede ser representado por una función PSD. Para determinar la función de densidad espectral de potencia, o PSD, es necesario medir el perfil de superficie con respecto a un plano de referencia. Los perfiles aleatorios de carretera pueden ser aproximados por un PSD en forma de

$$\Phi(\Omega) = \Phi(\Omega_0)\left(\frac{\Omega}{\Omega_0}\right)^{-\omega} \quad \text{o} \quad \Phi(n) = \Phi(n_0)\left(\frac{n}{n_0}\right)^{-\omega}$$

Donde

$\Omega = \frac{2\pi}{\lambda}$ en (rad/m) la frecuencia espacial angular, λ es la longitud de onda, $\Phi_0 \triangleq \Phi(\Omega_0)$ en m^2/(rad/s) describe los valores del PSD al número de onda de referencia Ω_0 =1rad/m, $n = \frac{\Omega}{2\pi}$ es la frecuencia espacial, n_0 =ciclos/m, ω es la ondulación, para la mayoría de las superficies de carretera, ω =2.

Las características de la aspereza de la carretera serán expresadas por una señal cuya función de distribución PSD es (Tyan *et al*, 2009).

$$\Phi(\omega) = \frac{2\alpha V\sigma^2}{\omega^2 + \alpha^2 V^2}$$

Donde σ^2 denota la varianza de la aspereza de la carretera y V la velocidad del vehículo, α depende de la superficie de la carretera, $2\alpha V\sigma^2$ es la densidad espectral del proceso de ruido blanco.

La señal $z_r(t)$ cuya PSD viene dada por la ecuación anterior puede ser obtenida al integrar la salida de un filtro lineal expresado por la ecuación diferencial

$$\dot{z}_r(t) = -\alpha V z_r(t) + w(t)$$

En (Corriga *et al*, 1991) se determina la varianza de un perfil de carretera aleatorio para diferentes clases de carretera como se aprecia en la siguiente tabla, además, llega a la conclusión que $\alpha = 0.127$.

Tabla A.1. Desviación estándar de aspereza de la carretera (Corriga *et al*, 1991).

Clase de Carretera	$\sigma\ (10^{-3}\mathrm{m})$	$\Phi(\Omega_0)(10^{-6}\mathrm{m}^3),\ \Omega_0 = 1$	α(rad/m)
A (muy buena)	2	1	0.127
B (buena)	4	4	0.127
C (promedio)	8	16	0.127
D (pobre)	16	64	0.127
E (muy pobre)	32	256	0.127

El diagrama de bloques en Simulink del perfil de carretera generado se muestra en la siguiente Figura.

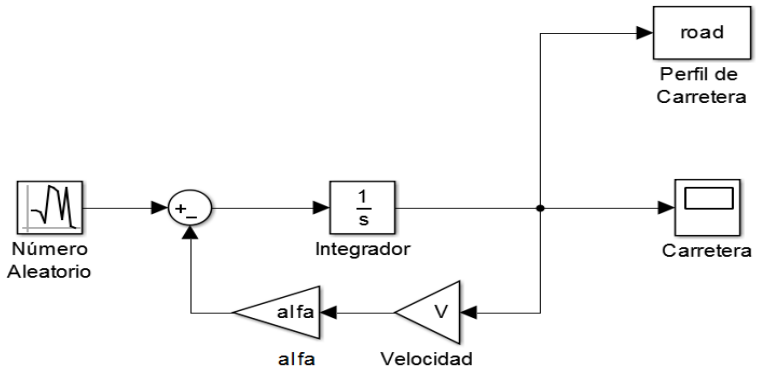

Figura A.2. Sistema lineal de primer orden.

Con lo cual se obtiene el siguiente perfil.

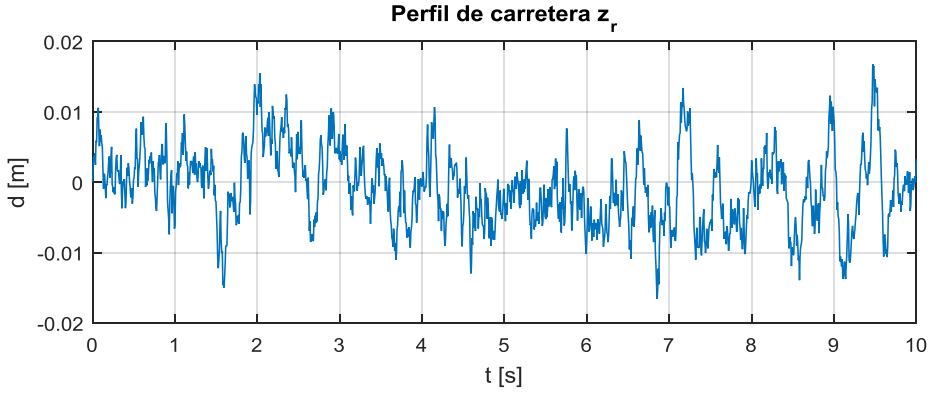

Figura A.3. Perfil de carretera generado por filtro de conformación

ANEXO B: Código del Sistema de Diagnóstico de Fallas

```
clear; clc; close all;

%% Parámetros del modelo se suspensión semi-activa

ms  = 470;          % masa suspendida
mus = 110;          % masa no suspendida
Ks  = 86378;        % coeficiente de rigidez de la suspensión
Kt  = 270000;       % coeficiente de rigidez del neumático
Kp  = -7897.21;     % coeficiente de rigidez interno del
amortiguador MR
Cp  = 2830.86;      % coeficiente de amortiguamiento viscoso
fc  = 600.95;       % coeficiente de fricción interno del
amortiguador MR
av  = 37.85;        % propiedad de histeresis del amortiguador MR
ad  = 22.15;        % propiedad de histeresis del amortiguador MR

%% Matrices de estados
% xp = Ax + Bu + Dv
% y  = Cx

A = [       0         1              0         0
     -(Kp+Ks)/ms   -Cp/ms      (Kp+Ks)/ms    Cp/ms
             0         0              0         1
     (Ks+Kp)/mus   Cp/mus -(Ks+Kp+Kt)/mus  -Cp/mus ];

Af = A;
Cpf = Cp;

B = [0; -1/ms; 0; 1/mus];

C = [ 0  1   0  0
      0  0   0  1
      1  0  -1  0];

D = [ 0; 0; 0; Kt/mus];

%% Banco de Observadores de Entradas Desconocidas (B_UIO)
% zp = Nz + Ly + Gu
% x  =  z - Ey
% Matrices C para cada observador
C1 = C(1:2,:);
C2 = C(2:3,:);
load BankUIO;       % Carga Matrices N, L, G y E del Banco de
Observadores

%% Tiempo de Simulación
ti = 0;             % Tiempo inicial
tf = input('Ingresar tiempo de simulación: ');    % Tiempo final
dt = 0.007;         % Periodo de muestreo
t = ti:dt:tf;       % Vector tiempo
t = t';             % vector columna
nt = length(t);     % tamaño del vector tiempo
```

```
%% Perfil de Carretera
load proad;          % Carga Perfil de Carretera
vv = road.signals.values(1:end);

%% Condiciones Iniciales
x   = [ 0; 0; 0; 0];        % Estados iniciales del sistema
z1  = [ 0; 0; 0; 0];        % Estados iniciales UIO 01
z2  = [ 0; 0; 0; 0];        % Estados iniciales UIO 02
xh1 = [ 0; 0; 0; 0];        % Estados estimados iniciales UIO 01
xh2 = [ 0; 0; 0; 0];        % Estados estimados iniciales UIO 02

%% Condiciones iniciales para estimación paramétrica
Pk = 10e8*eye(3);           % Matriz de covarianza inicial
lambda = 0.95;              % Factor de olvido
theta = [ 0; 0; 0];         % Vector de parámetros inicial

smp = 5;                    % tamaño del vector de datos guardados
vrcp = 0*ones(smp,1)';      % vectores iniciales
vrkp = 0*ones(smp,1)';
vrfc = 0*ones(smp,1)';

%% Ruido de medición
fn = [0.0001*randn(1,nt)
      0.0001*randn(1,nt)
      0*ones(1,nt) ];

%% Pesos Sinápticos de la Red Neuronal
load pesosfallas20 v w a c

%% Condiciones iniciales de los filtros pasa altas
% xf = as/(s+a)x
zo11 = 0;
zo13 = 0;
zo21 = 0;
zo23 = 0;
aa=1.1;
%% Condiciones de Fallas
f1 = 0*ones(3,nt);
tfm = tf;
tds1 = tf;
tds2 = tf;
tds3 = tf;

tipfal = input('Introducir Tipo de falla\n 1 desconexión\n 2
aditiva\n 3 fuga\n: ');

if (tipfal==1)
faldes = input('Desconexion Senor(es)\n [1] 1,2 y 3\n [2] 1\n [3]
2\n [4] 3\n [5] 1 y 2\n [6] 1 y 3\n [7] 2 y 3\n: ');
tiefal = input('Tiempo en que ocurre la falla: ');
        if(faldes==1)
        tds1=tiefal; tds2=tiefal; tds3=tiefal;
        elseif (faldes==2)
        tds1=tiefal; tds2=tf+1; tds3=tf+1;
```

```
        elseif (faldes==3)
        tds1=tf+1; tds2=tiefal; tds3=tf+1;
        elseif (faldes==4)
        tds1=tf+1; tds2=tf+1; tds3=tiefal;
        elseif (faldes==5)
        tds1=tiefal; tds2=tiefal; tds3=tf+1;
        elseif (faldes==6)
        tds1=tiefal; tds2=tf+1; tds3=tiefal;
        elseif (faldes==7)
        tds1=tf+1; tds2=tiefal; tds3=tiefal;
        end
elseif (tipfal==2)
 faladi = input('Falla sensor\n [10]+1\n [11]-1\n [12]+2\n [13]-2\n
[14]+3\n [15]-3: ');
 tiefal = input('Tiempo en que ocurre la falla: ');
        if (faladi==10)
        f1(1,round(tiefal/dt):end) = 0.02*ones(1,nt-
round(tiefal/dt)+1);
        elseif (faladi==11)
        f1(1,round(tiefal/dt):end) =-0.02*ones(1,nt-
round(tiefal/dt)+1);
        elseif (faladi==12)
        f1(2,round(tiefal/dt):end) = 0.05*ones(1,nt-
round(tiefal/dt)+1);
        elseif (faladi==13)
        f1(2,round(tiefal/dt):end) =-0.05*ones(1,nt-
round(tiefal/dt)+1);
        elseif (faladi==14)
        f1(3,round(tiefal/dt):end) = 0.002*ones(1,nt-
round(tiefal/dt)+1);
        elseif (faladi==15)
        f1(3,round(tiefal/dt):end) =-0.002*ones(1,nt-
round(tiefal/dt)+1);
        end
elseif (tipfal==3)
        pcf = input('Poncentaje de Fuga: [del 6 al 30]: ');
        tfm = input('Tiempo en que ocurre la falla: ');
end
k = 1;
j = 1;
for tt = ti:dt:tf
 yr(k,1:3) = (C*x)'; % Salida Real del Sistema
%% Fallas Aditivas
    y = C*x + f1(1:3,k) + fn(1:3,k); % Salidas medidas, con ruidos y
falla de sensores

%% Fallas por Desconexión
    % Falla 4 desconexión sensor velocidad masa suspendida
    if (tt>=tds1)
        y(1,1) = 0;
    else
        y(1,1) = y(1,1);
    end
    % falla 7 desconexión sensor velocidad masa no suspendida
    if (tt>=tds2)
        y(2,1)=0;
    else
```

```
        y(2,1) = y(2,1);
    end
    % falla 10 desconexión sensor deflexión
    if (tt>=tds3)
        y(3,1)= 0;
    else
        y(3,1) = y(3,1);
    end

%% Fallas Multiplicativas
   % Falla Multiplicativa; fuga MR, 6% - 30%
    if (tt>=tfm)
        Cpf = Cp*(100-pcf)*0.01;
        Af = [          0          1                0          0
               -(Kp+Ks)/ms   -Cpf/ms       (Kp+Ks)/ms     Cpf/ms
                        0          0                0          1
                (Ks+Kp)/mus   Cpf/mus -(Ks+Kp+Kt)/mus  -Cpf/mus ];
    else
        Cpf = Cp;
        Af = A;
    end

%%
% Salidas medidas
    y1 = y(1:2,1);          % Salidas alimentan UIO 01
    y2 = y(2:3,1);          % Salidas alimentan UIO 02
    yk(k,1:3) = y';

%% Salidas estimadas

  % UIO 01
    xh1 = z1 - E1*y1;       % Estados estimados UIO 01
    y11 = C*xh1;            % Salidas estimadas UIO 01
    y11k(k,1)=y11(1,1);
    y12k(k,1)=y11(2,1);
    y13k(k,1)=y11(3,1);

  % UIO 02
    xh2= z2 - E2*y2;        % Estados estimados UIO 02
    y22 = C*xh2;            % Salidas estimadas UIO 02
    y21k(k,1)=y22(1,1);
    y22k(k,1)=y22(2,1);
    y23k(k,1)=y22(3,1);

  %% guarda estados del sistema
    xx1(k,1) = x(1,1);      % desplazamiento de la masa suspendida
    xx2(k,1) = x(2,1);      % velocidad de la masa suspendida
    xx3(k,1) = x(3,1);      % desplazamiento de la masa no
suspendida
    xx4(k,1) = x(4,1);      % velocidad de la masa no suspendida

  % guarda estados estimados del UIO 01
    xxh11(k,1) = xh1(1,1);
    xxh12(k,1) = xh1(2,1);
    xxh13(k,1) = xh1(3,1);
    xxh14(k,1) = xh1(4,1);
```

```
  % guarda estados estimados del UIO 02
    xxh21(k,1) = xh2(1,1);
    xxh22(k,1) = xh2(2,1);
    xxh23(k,1) = xh2(3,1);
    xxh24(k,1) = xh2(4,1);

  %% Variable de Control, Control Sky-Hook
  % I = 10*abs(Vms)    Vms*[Vms-Vmus] >= 0
  % I = 0              Vms*[Vms-Vmus] <  0
  % Corriente
  if (y(1,1)*(y(1,1)-y(2,1))>=0); % Salidas medidas
      u=10*abs(y(1,1));
  else
      u=0;
  end
  if (u>1)
      u=1;  % max 1A
  end

  %% Parte no lineal MR real
  pp=fc*tanh(av*(x(2,1)-x(4,1)) + ad*(x(1,1)-x(3,1)));
  FD = u*pp; % Fuerza producida por el amortiguador MR

  % Limita Fuerza MR
  if (FD>3000)
       FD=3000;
  else if (FD<-3000)
       FD=-3000;
       end
  end
  % guarada la variable fuerza MR
  FDk(k,1)=FD;

  % Sensor Fuerza MR
  ymr = Kp*(x(1,1)-x(3,1))+Cpf*(x(2,1)-x(4,1))+FD;

  %% parte no lineal MR calculada a partir de las mediciones
  ppc=fc*tanh(av*(y(1,1)-y(2,1)) + ad*(y(3,1)));
  FDc = u*ppc; % calculada, se usa en estimación paramétrica

 %% Sistema Real
  xp = Af*x + B*FD + D*vv(k,1);
  x = x + dt*xp;

%% observadores
    %% UIO 01
    pp1=fc*tanh(av*(xh1(2,1)-xh1(4,1)) + ad*(xh1(1,1)-xh1(3,1)));
    FD1 = u*pp1; % Se calcula con los estados estimados
    zp1 = N1*z1 + L1*y1 + G1*FD1;
    zo11 = zo11 - aa*dt*zo11 + aa*dt*zp1(1,1);  % Se filtra estado 1
    zo13 = zo13 - aa*dt*zo13 + aa*dt*zp1(3,1);  % Se filtra estado 3
    z1 = z1 + dt*zp1;
    z1(1,1) = zo11;
    z1(3,1) = zo13;
```

```
    %% UIO 02
    pp2=fc*tanh(av*(xh2(2,1)-xh2(4,1)) + ad*(xh2(1,1)-xh2(3,1)));
    FD2 = u*pp2; % Se calcula con los estados estimados
    zp2 = N2*z2 + L2*y2 + G2*FD2;
    zo21 = zo21 - aa*dt*zo21 + aa*dt*zp2(1,1);  % Se filtra estado 1
    zo23 = zo23 - aa*dt*zo23 + aa*dt*zp2(3,1);  % Se filtra estado 3
    z2 = z2 + dt*zp2;
    z2(1,1) = zo21;
    z2(3,1) = zo23;

%% Estimación Paramétrica
   phi = [y(3,1) (y(1,1)-y(2,1)) FDc]'; %1 Vector de regresión
   e = ymr - phi'*theta;                %2 Calcula el error de
estimación
   Lk = Pk*phi/(lambda + phi'*Pk*phi);  %3 Ganancia
   theta = theta + Lk*e;                %4 Vector de parámetros
estimados
   Pk =1/lambda*(Pk-Lk*phi'*Pk);        %5 Actualiza matriz de
covarianzas

 %% Residuales Estimación Paramétrica
 r_Cp = (1-theta(2,1)/Cp)*100;
 if (abs(r_Cp)<=5)
     r_Cp = 0;
 end

 r_Kp = (1-theta(1,1)/Kp)*100;
 if (abs(r_Kp)<=5)
     r_Kp = 0;
 end

 r_fc = (1-theta(3,1))*100;
 if (abs(r_fc)<=5)
     r_fc = 0;
 end

 %% Guarda Residuales Estimación Paramétrica
 rcpk(k,1) = r_Cp;
 rkpk(k,1) = r_Kp;
 rfck(k,1) = r_fc;

 %% Residuales B_UIO
 r1 = y11(1,1)-y(1,1);
 r2 = y11(2,1)-y(2,1);
 r3 = y11(3,1)-y(3,1);
 r4 = y22(1,1)-y(1,1);
 r5 = y22(2,1)-y(2,1);
 r6 = y22(3,1)-y(3,1);

 %% Guarda datos Estimación Paramétrica
  vrcp(1,1) = r_Cp;              % paramétro estimado
  mcp=mean(vrcp);                % promedio de datos guardados
  vrcp(2:end)=vrcp(1:end-1);     % actualiza del vector de datos
guardados
```

```
  vrkp(1,1) = r_Kp;              % paramétro estimado
  mkp=mean(vrkp);                % promedio de datos guardados
  vrkp(2:end)=vrkp(1:end-1);     % actualiza del vector de datos
guardados

  vrfc(1,1) = r_fc;              % paramétro estimado
  mfc=mean(vrfc);                % promedio de datos guardados
  vrfc(2:end)=vrfc(1:end-1);     % actualiza del vector de datos
guardados

 %% Entradas Red Neuronal
  in1 = xh1(1,1);
  in2 = xh2(1,1);
  in3 = in1-in2;
  in4 = xh1(3,1);
  in5 = xh2(3,1);
  in6 = in4-in5;
  xin = 100*[in1 in2 in3 in4 in5 in6];

 %% Diagnóstico de Desconexión de Sensores
 if      (r4-y(1,1)==r4)&&(r2-y(2,1)==r2)&&(r3-y(3,1)==r3)
     K1 = 1; % desconexion 3 sensores
 elseif (r4-y(1,1)==r4)&&(r2-y(2,1)~=r2)&&(r3-y(3,1)~=r3)
     K1 = 2; % desconexion sensor 1
 elseif (r4-y(1,1)~=r4)&&(r2-y(2,1)==r2)&&(r3-y(3,1)~=r3)
     K1 = 3; % desconexion sensor 2
 elseif (r4-y(1,1)~=r4)&&(r2-y(2,1)~=r2)&&(r3-y(3,1)==r3)
     K1 = 4; % desconexion sensor 3
 elseif (r4-y(1,1)==r4)&&(r2-y(2,1)==r2)&&(r3-y(3,1)~=r3)
     K1 = 5; % desconexion sensor 1 y sensor 2
 elseif (r4-y(1,1)==r4)&&(r2-y(2,1)~=r2)&&(r3-y(3,1)==r3)
     K1 = 6; % desconexion sensor 1 y sensor 3
 elseif (r4-y(1,1)~=r4)&&(r2-y(2,1)==r2)&&(r3-y(3,1)==r3)
     K1 = 7; % desconexion sensor 2 y sensor 3
 % Diagnóstico de Fuga
 elseif (r4-y(1,1)~=r4)&&(r2-y(2,1)~=r2)&&(r3-y(3,1)~=r3)
      if (mcp~=0)&&(mkp==0)&&(mfc==0);
          K1 = 8; % fuga líquido MR
      elseif (mcp==0)&&(mkp==0)&&(mfc==0);
          K1 = 9; % Normal
      elseif (mcp~=0)&&(mkp~=0)&&(mfc~=0);
 % Diangnóstico Fallas Aditivas en los Sensores
 % Red Neuronal
        in = xin';
        m = v'*in;
        n = 1.0./(1+exp(-(m-c)./a));     % Sigmoidea 1
        out = w'*n;
        yf = out;  % Estado de funcionamiento del sistema
        [maxy, pos] = max(yf);
        K1 = pos+9; % 10 11 12 13 14 15
      end
 end
 %% Indica Condición de Operación del Sistema
 KK(k,1) = K1;
```

```
 if (K1==1)
     disp('Falla 3 sensores');
 elseif (K1==2)
     disp('Desconexión sensor Vms');
 elseif (K1==3)
     disp('Desconexión sensor Vmus');
 elseif (K1==4)
     disp('Desconexión sensor Zdef');
 elseif (K1==5)
     disp('Desconexión sensor 1 y 2');
 elseif (K1==6)
     disp('Desconexión sensor 1 y 3');
 elseif (K1==7)
     disp('Desconexión sensor 2 y 3');
 elseif (K1==8)
     disp('Fuga amortiguador MR');
 elseif (K1==9)
     disp('Normal Funcionamiento');
 elseif (K1==10)
     disp('Falla positiva sensor Vms');
 elseif (K1==11)
     disp('Falla negativa sensor Vms');
 elseif (K1==12)
     disp('Falla positiva sensor Vmus');
 elseif (K1==13)
     disp('Falla negativa sensor Vmus');
 elseif (K1==14)
     disp('Falla positiva sensor Zdef');
 elseif (K1==15)
     disp('Falla negativa sensor Zdef');
 end
 %%
 k = k + 1;
 j = j + 1;
end
%% Grafica Estados del Sistema y Estados Estimados
figure(1);
subplot(4,1,1); plot(t,xx1,'b',t,xxh11,'r',t,xxh21,'g');
legend('Zs','UIO 01','UIO 02'); xlabel('t [seg]'); ylabel('d [m]');
title('Desplazamiento Masa Suspendida'); grid;
subplot(4,1,2); plot(t,xx2,'b',t,xxh12,'r',t,xxh22,'g');
legend('dZs','UIO 01','UIO 02');  xlabel('t [seg]'); ylabel('v
[m/s]');
title('Velocidad Masa Suspendida'); grid;
subplot(4,1,3); plot(t,xx3,'b',t,xxh13,'r',t,xxh23,'g');
legend('Zus','UIO 01','UIO 02'); xlabel('t [seg]'); ylabel('d [m]');
title('Desplazamiento Masa No Suspendida'); grid;
subplot(4,1,4); plot(t,xx4,'b',t,xxh14,'r',t,xxh24,'g');
legend('dZus','UIO 01','UIO 02'); xlabel('t [seg]'); ylabel('v
[m/s]');
title('Velocidad Masa No Suspendida'); grid;

%% Grafica Residuales de la Estimación Paramétrica
figure(2)
plot(t,rcpk,'b',t,rkpk,'--r',t,rfck,'-.g'); ylim([-70 70]); grid;
legend('r_C_p','r_K_p','r_f_c'); title('Estimación Paramétrica');
```

```
%% Grafica Condición de Operación del Sistema
figure(3)
plot(t,KK); title('Código de Falla Diagnosticada');
legend('K'); grid; xlabel('t [seg]');
ylim([1 15]);

%% Grafica Salidas Reales, Salidas Medidas y Salidas Estimadas
figure(4)
subplot(3,1,1);
plot(t,yr(1:end,1),'k',t,yk(1:end,1),'--b',t,y11k,'-
.r',t,y21k,':g'); grid;
l=legend('y1 real','sensor 1','yh11','yh21');
l.Orientation='horizontal';
xlabel('t [seg]'); ylabel('v [m/s]'); title('Velocidad Masa
Suspendida');
subplot(3,1,2);
plot(t,yr(1:end,2),'k',t,yk(1:end,2),'--b',t,y12k,'-
.r',t,y22k,':g'); grid;
l=legend('y2 real','sensor 2','yh12','yh22');
l.Orientation='horizontal';
xlabel('t [seg]'); ylabel('v [m/s]'); title('Velocidad Masa No
Suspendida');
subplot(3,1,3);
plot(t,yr(1:end,3),'k',t,yk(1:end,3),'--b',t,y13k,'-
.r',t,y23k,':g'); grid;
l=legend('y3 real','sensor 3','yh13','yh23');
l.Orientation='horizontal';
xlabel('t [seg]'); ylabel('d [m]'); title('Deflexión de la
Suspensión');
```