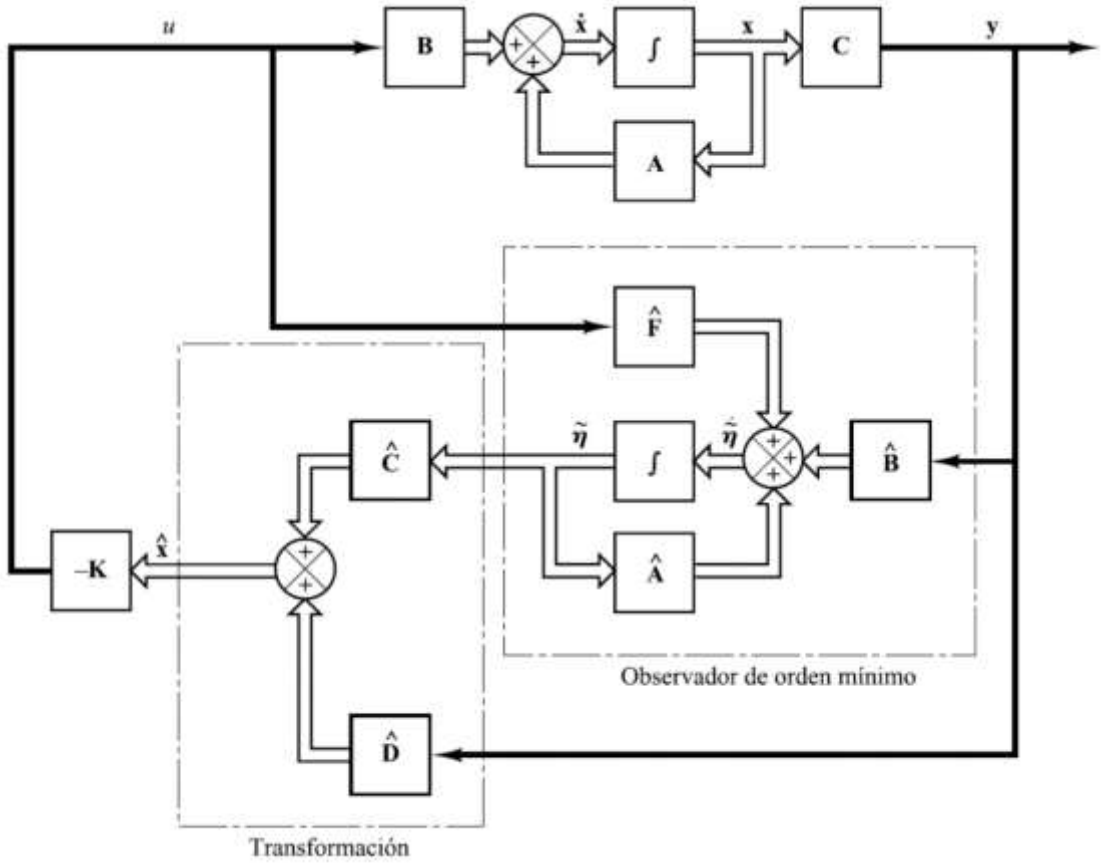
Estimación de Estados Observadores Reducidos o de Orden Mínimo

Fernando di Sciascio (2016)

Observadores reducidos o de orden mínimo

El vector de estado $x(t)$ es de dimensión n y la salida $y(t)$ es un escalar medible que se expresa como una combinación lineal de las variables de estado ($y(t)=C\boldsymbol{x}(t)$). Por lo que no necesitan estimarse las n variables de estado, sino sólo n-1. Así, el observador de orden reducido se vuelve un observador de (n-1)-ésimo orden.



Procedimiento de diseño de un observador reducido

Paso 1) El modelo de estados (A,B,C,D) se transforma a la forma Canónica Controlable 1b o beta $(\bar{A},\bar{B},\bar{C},\bar{D})$. En estas coordenadas la salida queda como el primer estado (así, no es necesario construir un observador para estimar todo el estado, sino solamente los *n* -1 restantes).

Si el par (*A*,*C*) es observable, usamos la matriz de observabilidad $\mathcal{O}$ (que es no singular) como matriz de transformación o cambio de base llevamos la planta original a la forma Canónica Controlable 1b (FCC1b).

$$\mathcal{O} = \left[C \; CA \cdots CA^{n-1} \right]^T$$

$$\bar{A} = \mathcal{O} A \mathcal{O}^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix}$$

$$\bar{B} = \underbrace{\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{bmatrix}}_{\boldsymbol{\beta}} = \underbrace{\begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}^{-1}}_{\boldsymbol{M^{-1}}} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}}_{\boldsymbol{b}} = \mathcal{O}^{-1}B$$

$$\bar{C} = C\mathcal{O}^{-1}\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad \bar{D} = 0$$

Donde los coeficientes se obtienen de la función de transferencia

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

El cambio de base a la FCC1b se hace teniendo en cuenta que la matriz de transformación es la matriz de observabilidad $\mathcal{O}$ del sistema original.

Paso 2) Se particiona el vector de estados **x** en dos partes x_a (un escalar) y $\mathbf{x}_b$ [un vector de dimensión (n-1)]. Aquí la variable de estado x_a es igual a la salida y de modo que se mide directamente, y $\mathbf{x}_b$ es la parte no medible del vector de estado. De esta forma, el estado particionado y las ecuaciones de salida son

$$\begin{bmatrix} \dot{x}_a(t) \\ \dot{x}_b(t) \end{bmatrix} = \begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix} \begin{bmatrix} x_a(t) \\ x_b(t) \end{bmatrix} + \begin{bmatrix} B_a \\ B_b \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0_{1\times n-1} \end{bmatrix} \begin{bmatrix} x_a(t) \\ x_b(t) \end{bmatrix}$$

A_{aa} = escalar

$\mathbf{A}_{ab}$ = matriz de $1 \times (n-1)$

$\mathbf{A}_{ba}$ = matriz de $(n-1) \times 1$

$\mathbf{A}_{bb}$ = matriz de $(n-1) \times (n-1)$

B_a = escalar

$\mathbf{B}_b$ = matriz de $(n-1) \times 1$

Paso 3) Se determina la matriz de ganancia del observador L, μ_1, μ_2, ..., μ_{n-1} son los valores propios deseados para el observador de orden mínimo.

$$\left| sI - A_{bb} + LA_{ab} \right| = (s-\mu_1)(s-\mu_2)\cdots(s-\mu_{n-1})$$

$$= s^{n-1} + \alpha_{n-2}s^{n-2} + \cdots + \alpha_1 s + \alpha_0 = 0$$

Determinación de la matriz de ganancia del observador de orden reducido *L* con MATLAB

Debido a la dualidad del diseño de asignación de polos y del observador, se puede aplicar el mismo algoritmo a ambos problemas. Así los comandos de Matlab `acker` y `place` se pueden utilizar para determinar la matriz de ganancia del observador *L*.

$$L = \text{acker}(A_{bb}^T, A_{ab}^T, [\mu_1\ \mu_2 \cdots \mu_{n-1}\ ])^T$$

Donde μ_1, μ_2, ..., μ_{n-1} son valores propios deseados para el observador de orden mínimo. También se puede utilizar

$$L = \text{place}(A_{bb}^T, A_{ab}^T, [\mu_1\ \mu_2 \cdots \mu_{n-1}\ ])^T$$

Paso 4) Se construye el siguiente sistema de orden $n-1$ que genera una estima asintótica de $x(t)$.

$$\dot{\eta}(t) = \hat{A}\eta(t) + \hat{F}u(t) + \hat{B}y(t) = \hat{A}\eta(t) + \begin{bmatrix}\hat{F}\\ \hat{B}\end{bmatrix}\begin{bmatrix}u(t)\\ y(t)\end{bmatrix}$$

$$\hat{x}(t) = \begin{bmatrix}y(t)\\ \eta(t) + Ly(t)\end{bmatrix} = \begin{bmatrix}0_{1\times n-1} & 1\\ I_{n-1} & L\end{bmatrix}\begin{bmatrix}\eta(t)\\ y(t)\end{bmatrix} = \begin{bmatrix}\hat{C} & \hat{D}\end{bmatrix}\begin{bmatrix}\eta(t)\\ y(t)\end{bmatrix} = \hat{C}\eta(t) + \hat{D}y(t)$$

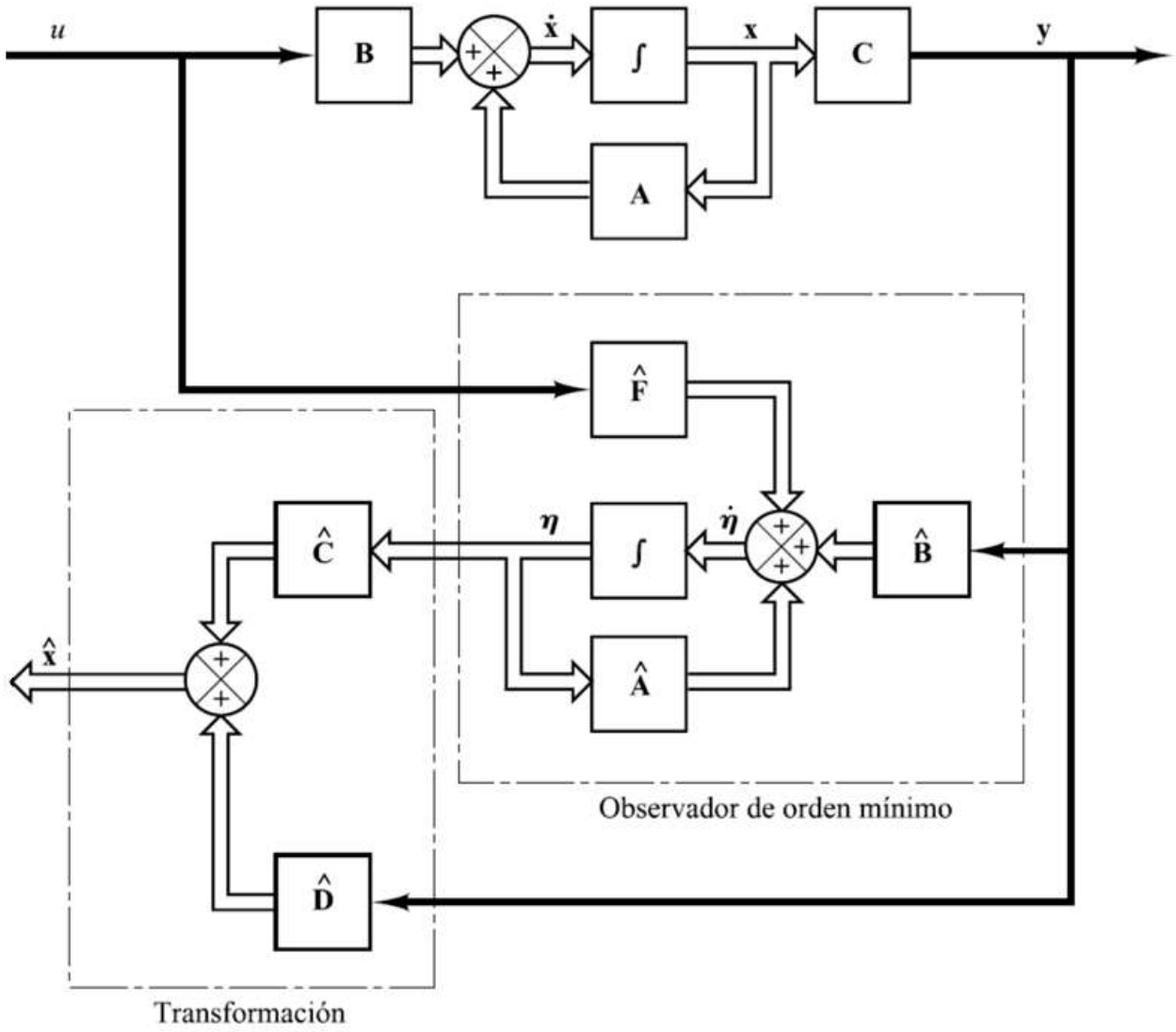
donde

$$\hat{A} = A_{bb} - LA_{ab} \qquad \hat{C} = \begin{bmatrix}0_{1\times n-1}\\ I_{n-1}\end{bmatrix}$$

$$\hat{B} = \hat{A}L + A_{ba} - LA_{aa} \qquad \hat{D} = \begin{bmatrix}1\\ L\end{bmatrix}$$

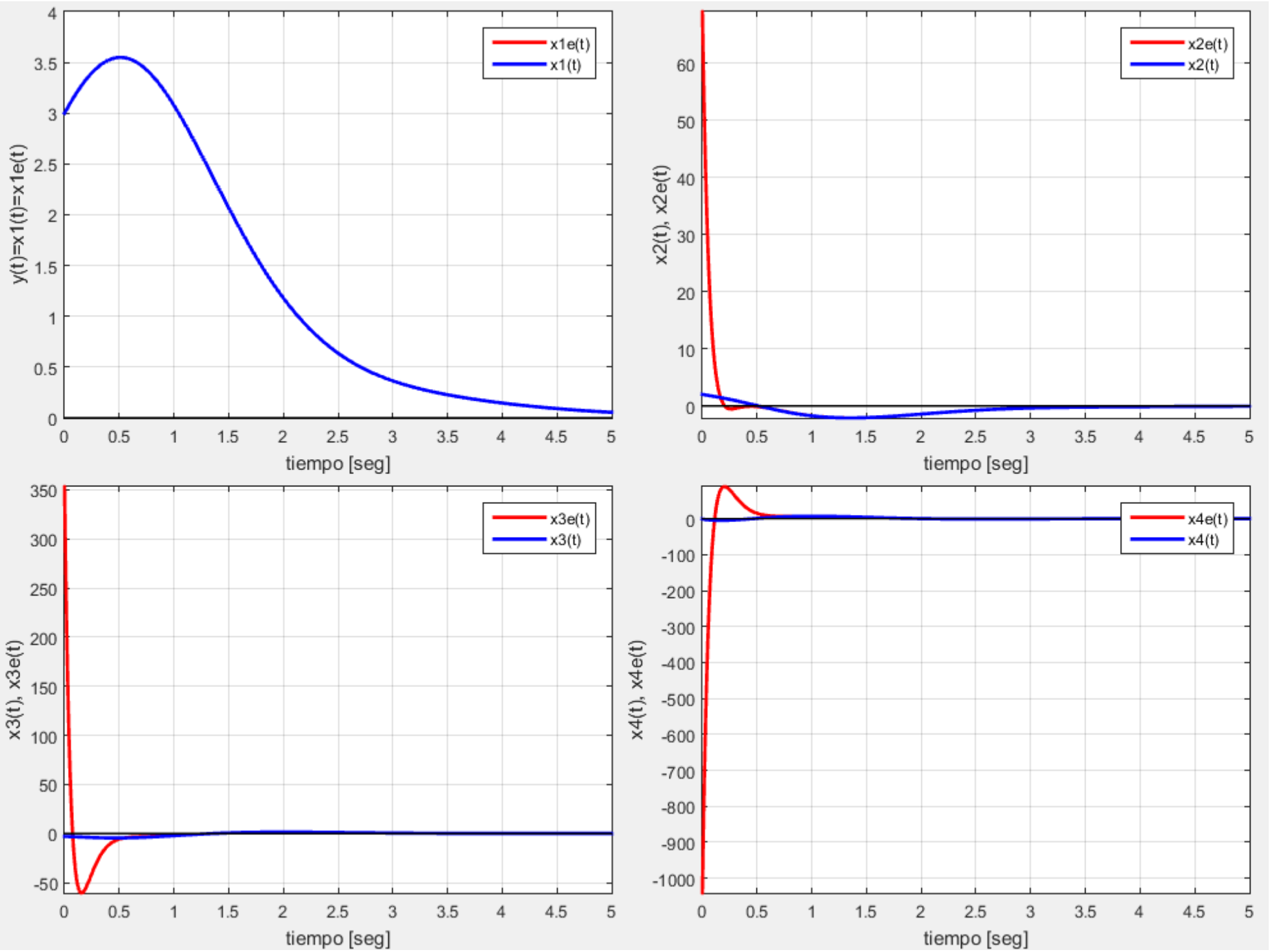
$$\hat{F} = B_b - LB_a$$

El diseño anterior lo verificamos en una simulación “sin realimentación de los estados”.



El scripts siguiente simula este caso.

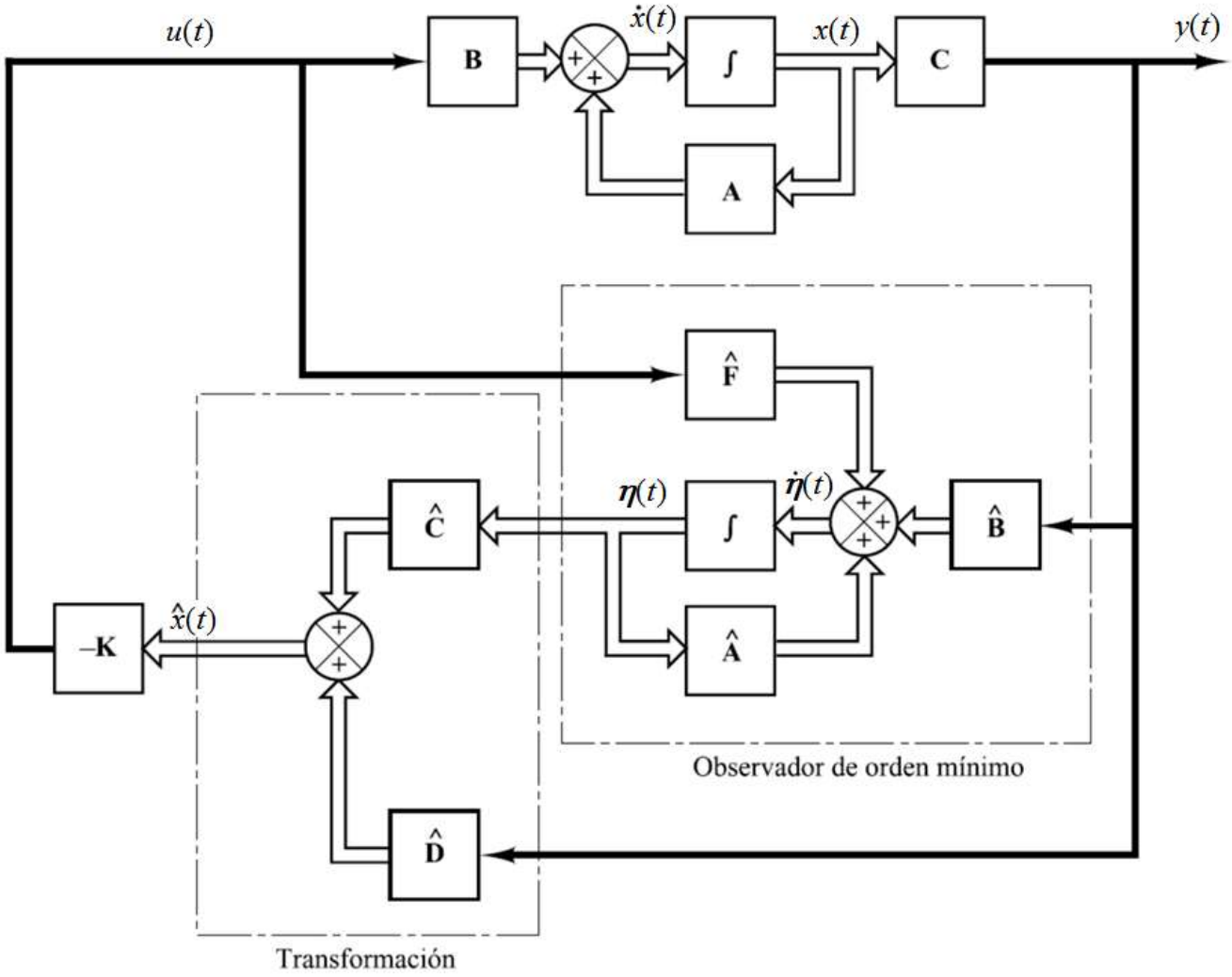
```
clear, clc, close all
planta_original=ss(zpk([-2+2i -2-2i],[-1 -1.5+2i -1.5-2i -3],2.344)); [A,B,C,D]=ssdata(planta_original);
% OBSERVADOR DE ORDEN REDUCIDO
% 1) Se pasa la planta a la FCC1b.
[Acc,Bcc,Ccc,Dcc]=ss2ss(A,B,C,D,obsv(A,C));
planta_FCC1b=ss(Acc,Bcc,Ccc,Dcc);  x0=[3 2 -3 -1]; % Condiciones iniciales
% Realizo la partición de las matrices Acc y Bcc
Aaa=Acc(1,1); Aab=Acc(1,2:end); Aba=Acc(2:end,1); Abb=Acc(2:end,2:end); Ba=Bcc(1); Bb=Bcc(2:end);
%Se especifican los autovalores deseados del observador y se determina L
lambdas_obs=[-10 -10 -10]; L=acker(Abb',Aab',lambdas_obs)';
% Calculo las matrices y vectores con sombrero (hat)Ahat, Bhat, Chat, Dhat, Fhat
Ahat=Abb-L*Aab; Bhat=Ahat*L+Aba-L*Aaa;
Chat=[zeros(1,length(A)-1);eye(length(A)-1)]; Dhat=[1;L]; Fhat=Bb-L*Ba;
observer=ss(Ahat,[Bhat Fhat],eye(length(A)-1),zeros(length(A)-1,2));
% Se simula la planta ya que se necesita y(t)
t=[0:0.01:5];u=zeros(length(t),1); [y,t,x]=lsim(planta_FCC1b,u,t, x0);
% Se simula el observador
[z,t]=lsim(observer,[y u]',t);
%xe=Chat*z'+Dhat*y;
z=z'; for i=1:length(t); xe(i,:)=Chat*z(:,i)+Dhat*y(i); end;
subplot(2,2,1);plot(t,xe(:,1),'r',t,x(:,1),'b','LineWidth',2);
line([0 max(t)],[0 0],'LineWidth',1,'Color','k','LineStyle','-')  %Eje real
legend('x1e(t)','x1(t)'); xlabel('tiempo [seg]'); ylabel('y(t)=x1(t)=x1e(t)'); grid on
subplot(2,2,2);plot(t,xe(:,2),'r',t,x(:,2),'b','LineWidth',2); axis([0 t(end)  ...
    min(min(x(:,2)),min(xe(:,2)))  max(max(x(:,2)),max(xe(:,2)))])
line([0 max(t)],[0 0],'LineWidth',1,'Color','k','LineStyle','-')  %Eje real
legend('x2e(t)','x2(t)'); xlabel('tiempo [seg]'); ylabel('x2(t), x2e(t)'); grid on
subplot(2,2,3);plot(t,xe(:,3),'r',t,x(:,3),'b','LineWidth',2); axis([0 t(end)...
    min(min(x(:,3)),min(xe(:,3)))  max(max(x(:,3)),max(xe(:,3)))])
line([0 max(t)],[0 0],'LineWidth',1,'Color','k','LineStyle','-')  %Eje real
legend('x3e(t)','x3(t)'); xlabel('tiempo [seg]'); ylabel('x3(t), x3e(t)'); grid on
subplot(2,2,4);plot(t,xe(:,4),'r',t,x(:,4),'b','LineWidth',2); axis([0 t(end) ...
    min(min(x(:,4)),min(xe(:,4)))  max(max(x(:,4)),max(xe(:,4)))])
line([0 max(t)],[0 0],'LineWidth',1,'Color','k','LineStyle','-')  %Eje real
legend('x4e(t)','x4(t)'); xlabel('tiempo [seg]'); ylabel('x4(t), x4e(t)'); grid on
```


x1e(t)
x1(t)
y(t)=x1(t)=x1e(t)
tiempo [seg]
x2e(t)
x2(t)
x2(t), x2e(t)
tiempo [seg]
x3e(t)
x3(t)
x3(t), x3e(t)
tiempo [seg]
x4e(t)
x4(t)
x4(t), x4e(t)
tiempo [seg]

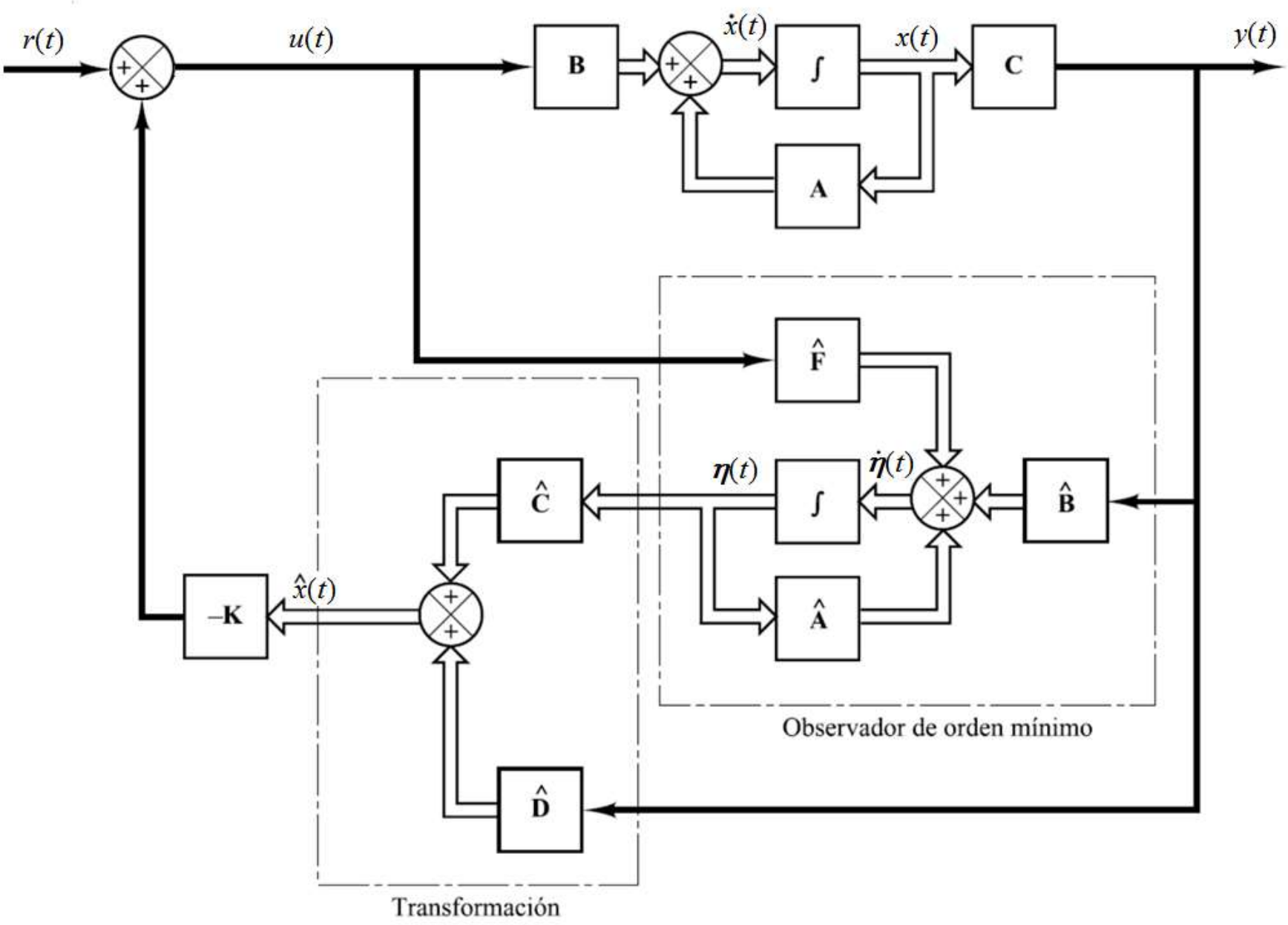
Realimentación de los Estados Estimados

Realimentación de los estados estimados

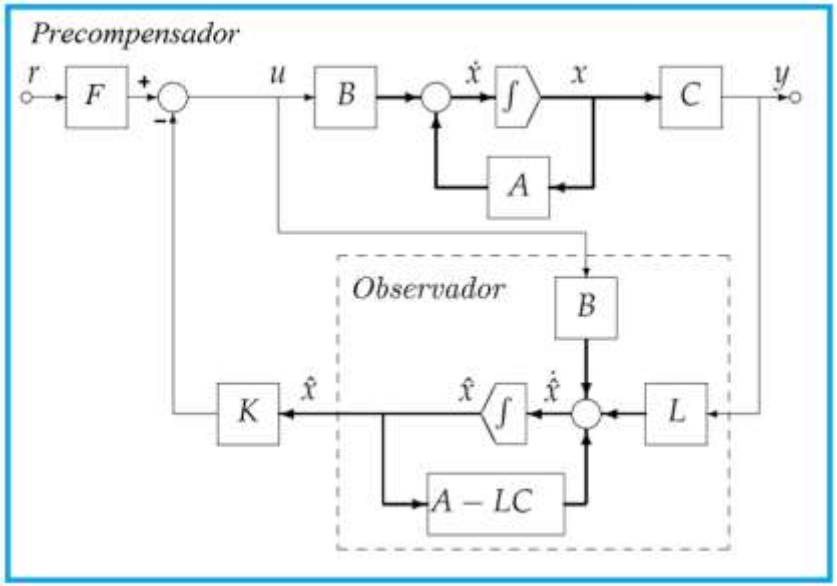
Ahora queremos utilizar el observador de orden reducido que hemos obtenido en el controlador (realimentación de los estados estimados). Para regulación como muestra la figura.



o para seguimiento.



Si analizamos la script anterior vemos que primero simulamos la planta y después el observador introduciendo la misma $u(t)$ y la salida $y(t)$ de la simulación de la planta. Ahora *con el sistema a lazo cerrado esa estrategia de simulación no la podemos emplear*. En el observador de orden completo no tuvimos este inconveniente ya que disponíamos del modelo completo a lazo cerrado.



$$\begin{bmatrix} \dot{x}(t) \\ \dot{\hat{x}}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A & -BK \\ LC & A - LC - BK \end{bmatrix}}_{A_{cl}} \begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix} + \underbrace{\begin{bmatrix} BF \\ BF \end{bmatrix}}_{B_{cl}} r(t)$$

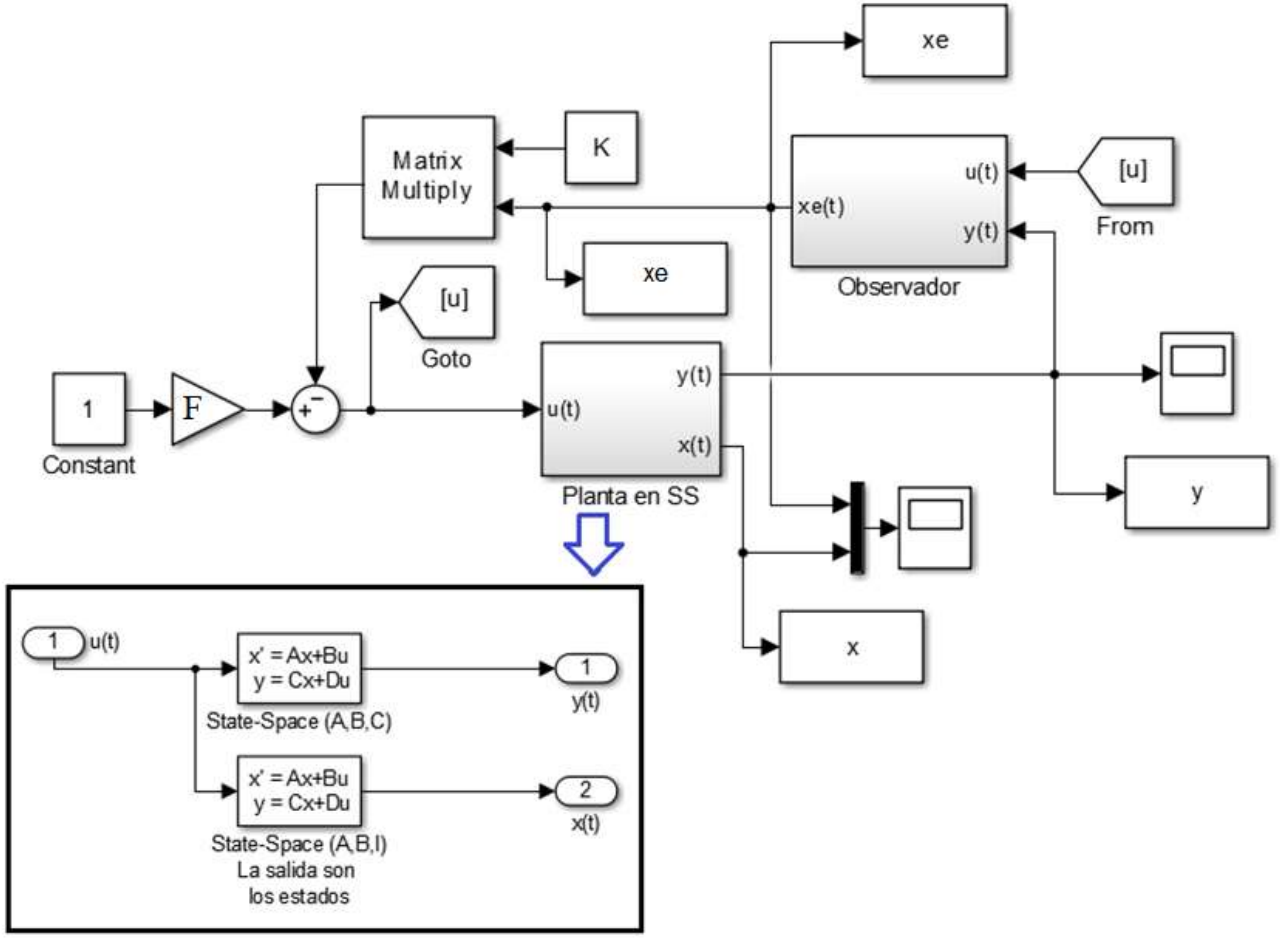
$$y(t) = \underbrace{\begin{bmatrix} C & 0 \end{bmatrix}}_{C_{cl}} \begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix}$$

No tenemos en el observador de orden reducido un modelo a lazo cerrado similar al de orden completo.

Existen varias alternativas para realizar la simulación a lazo cerrado.

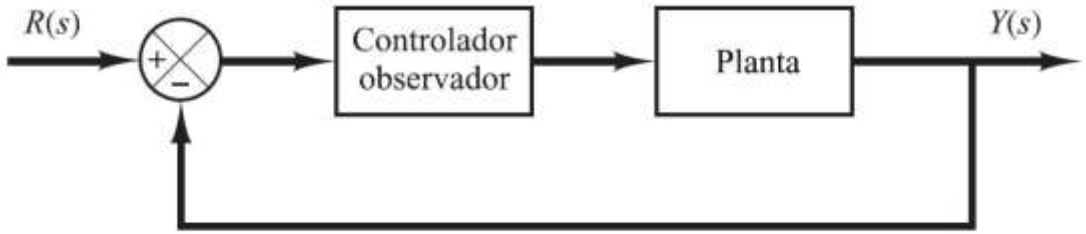
1) Utilizar el comando `connect` para construir en Matlab el modelo a lazo cerrado.

2) La solución más sencilla es construir el modelo a lazo cerrado en Simulink.

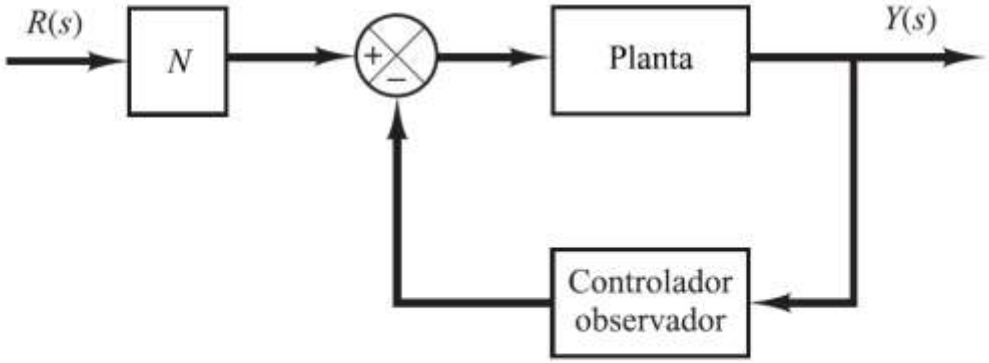


y realizar la simulación desde Matlab con el comando `sim` en vez del `lsim`.

3) Obtener la función de transferencia del observador de orden reducido-controlador $G_{oc}(s)$ como hicimos en el observador de orden completo y analizar la respuesta del sistema con la salida. En este caso perdemos la información de la evolución de los estados de la planta y de los del observador (estimaciones) pero se asume que si la salida es la adecuada todo va bien (recordemos que los estados son un medio para construir un controlador que nos de la salida deseada).



(a)

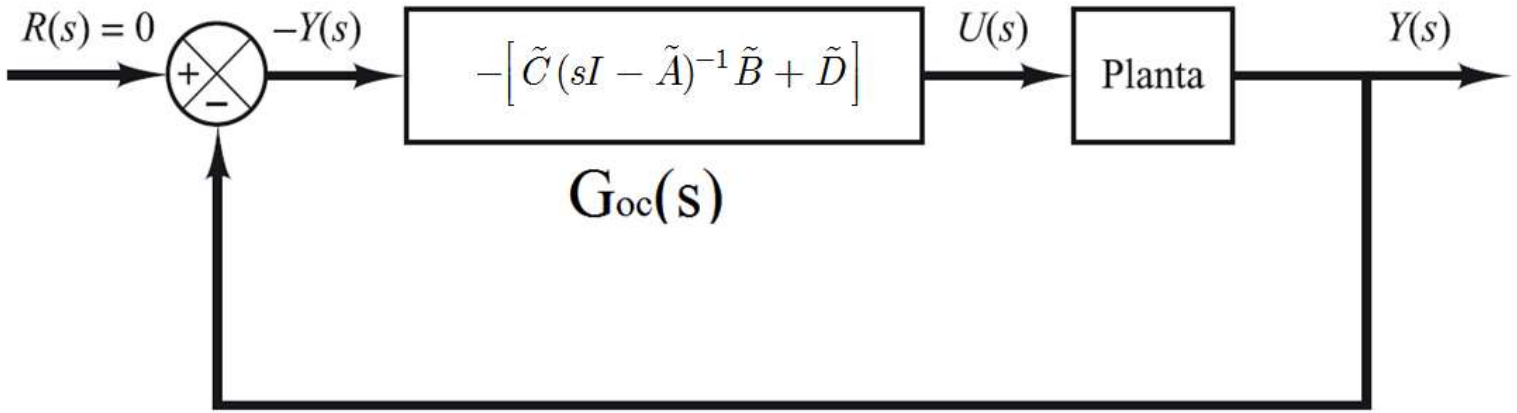


(b)

Función de Transferencia Observador de orden reducido–Controlador

La función de transferencia conjunta del observador de orden reducido-controlador $G_{oc}(s)$ es: (ver deducción en Ogata)

$$G_{oc}(s) = -\frac{U(s)}{Y(s)} = -\left[\tilde{C}\,(sI - \tilde{A})^{-1}\tilde{B} + \tilde{D}\right] = -\frac{\tilde{C}\,\mathrm{adj}(sI - \tilde{A})\,\tilde{B} + \left|sI - \tilde{A}\right|\tilde{D}}{\left|sI - \tilde{A}\right|}$$



$$\hat{A} = A_{bb} - LA_{ab}$$

Donde : $\hat{B} = \hat{A}L + A_{ba} - LA_{aa}$, $K = \begin{bmatrix} K_a & K_b \end{bmatrix}$, $K_a = k_1$, $K_b = \begin{bmatrix} k_2 & k_3 \cdots k_n \end{bmatrix}$

$$\hat{F} = B_b - LB_a$$

$$\Downarrow$$

$$\tilde{A} = \hat{A} - \hat{F}K_b$$
$$\tilde{B} = \hat{B} - \hat{F}(K_a + K_bL)$$
$$\tilde{C} = -K_b$$
$$\tilde{D} = -(K_a + K_bL)$$

```
% Cálculo de la matriz de ganancia de realimentación del estado K
A = [0 1 0;0 0 1;0 -24 -10]; B = [0;10;-80]; C = [1 0 0]; D=0;
lambdas_cont = [-1+2*1i -1-2*1i -5];
K = acker(A,B,lambdas_cont);  % K=1.2500 1.2500 0.19375
% Cálculo de la matriz de ganancia del observador L
Aaa=0; Aab=[1 0]; Aba=[0;0]; Abb=[0 1;-24 -10];Ba=0; Bb=[10;-80];
lambdas_obs= [-10 -10];
L = acker(Abb',Aab',lambdas_obs)'; % L=[10 -24]'
% Determinación de la función de transferencia del controlador-observador
```

```
Ka = 1.25; Kb = [1.25 0.19375];
Ahat = Abb-L*Aab;
Bhat = Ahat*L+Aba-L*Aaa;
Fhat = Bb-L*Ba;
Atilde = Ahat-Fhat*Kb;
Btilde = Bhat-Fhat*(Ka+Kb*L);
Ctilde = -Kb;
Dtilde = -(Ka+Kb*L);
[num,den] = ss2tf(Atilde, Btilde, -Ctilde, -Dtilde);
Gc=tf(num,den)
% La función de transferencia del observador-controlador es:
%                     9.1 s^2 + 73.5 s + 125
%            G(s) =  ----------------------
%                        s^2 + 17 s - 30
% La función de transferencia es propia (no estrictamente propia), esto significa que el
% orden del numerador es igual al del denominador (número de ceros igual al número de polos).
%Llegamos a la misma función de transferencia con la function diofantina
Acl=poly([lambdas_cont lambdas_obs]);
[b,a]=ss2tf(A,B,C,D);
[R,S] = diofantina(a,b,Acl);
Gc_diofantina=zpk(tf(S,R))
% Da la misma función de transferencia:
%                                 9.1 s^2 + 73.5 s + 125
%             G_diofantina(s) =  ----------------------
%                                    s^2 + 17 s - 30
```

Diseño alternativo de observadores de orden reducido

Paso 1) Se transforma el modelo de estados original (A,B,C,D) a la forma Canónica Controlable 1b o beta $(\bar{A},\bar{B},\bar{C},\bar{D})$. En estas coordenadas la salida queda como el primer estado.

Paso 2) a. Se elige una matriz F cualquiera que sea Hurwitz de $(n-1)\times(n-1)$ y que no tenga autovalores en común con los de A (o los de $\bar{A}$ ya que son los mismos). Una opción es a partir de los valores propios deseados del observador μ_1, μ_2, ..., μ_{n-1} construir la matriz diagonal

$$F = \begin{bmatrix} \mu_1 & 0 & \cdots & 0 \\ 0 & \mu_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_{n-1} \end{bmatrix} = \mathrm{diag}([\mu_1 \cdots \mu_{n-1}])$$

b. Se elige un vector L cualquiera de $(n-1)\times 1$ tal que el par (F,L) sea controlable. Una opción es elegir las componentes de L aleatorios `L=rand(n-1,1)`, y verificar que `ctrb(F,L)` sea de rango completa.

Paso 3) Se calcula *T*, la solución única no singular, de la ecuación de Sylvester:

$$T\bar{A} - FT = L\bar{C}$$

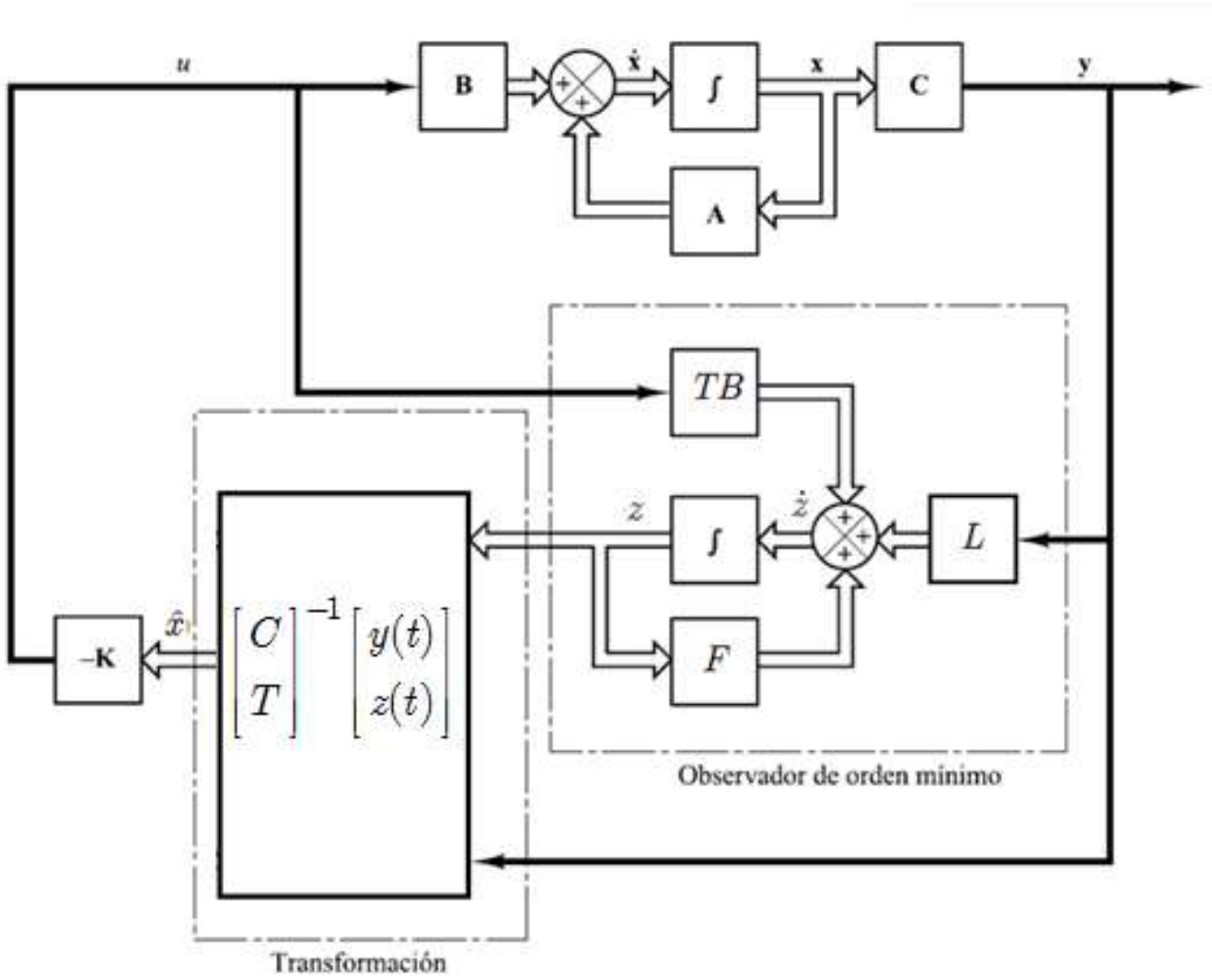
Notar que *T* es una matriz rectangular de $(n-1)\times n$.

Paso 4) Se construye el siguiente sistema de orden $n-1$ que genera *z*(*t*).

$$\dot{z}(t) = Fz(t) + T\bar{B}u(t) + Ly(t) = Fz(t) + \begin{bmatrix} T\bar{B} \\ L \end{bmatrix} \begin{bmatrix} u(t) \\ y(t) \end{bmatrix}$$

Paso 5) Se realiza la siguiente transformación lineal para obtener $\hat{x}(t)$

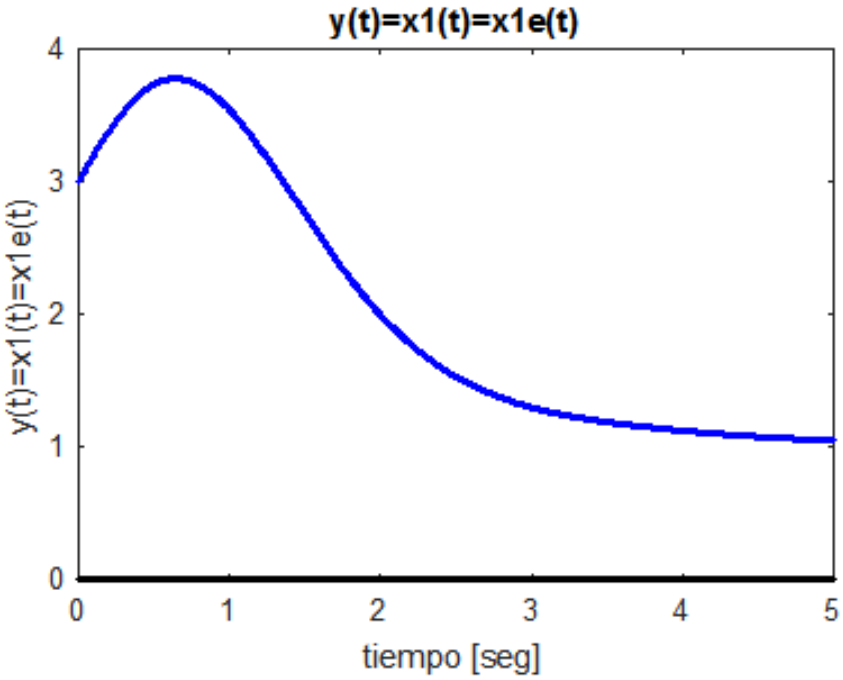
$$\hat{x}(t) = \begin{bmatrix} \bar{C} \\ T \end{bmatrix}^{-1} \begin{bmatrix} y(t) \\ z(t) \end{bmatrix}$$

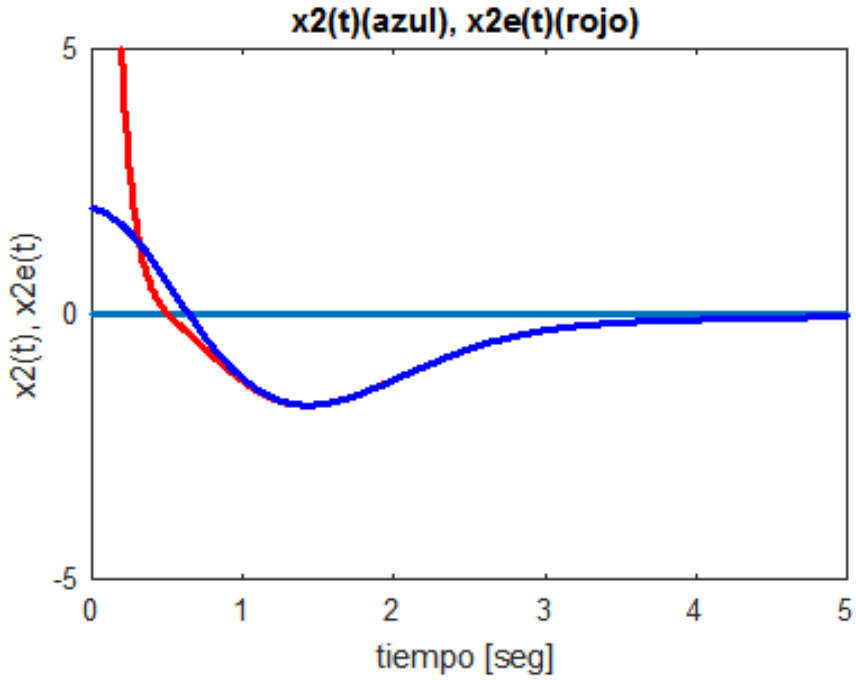


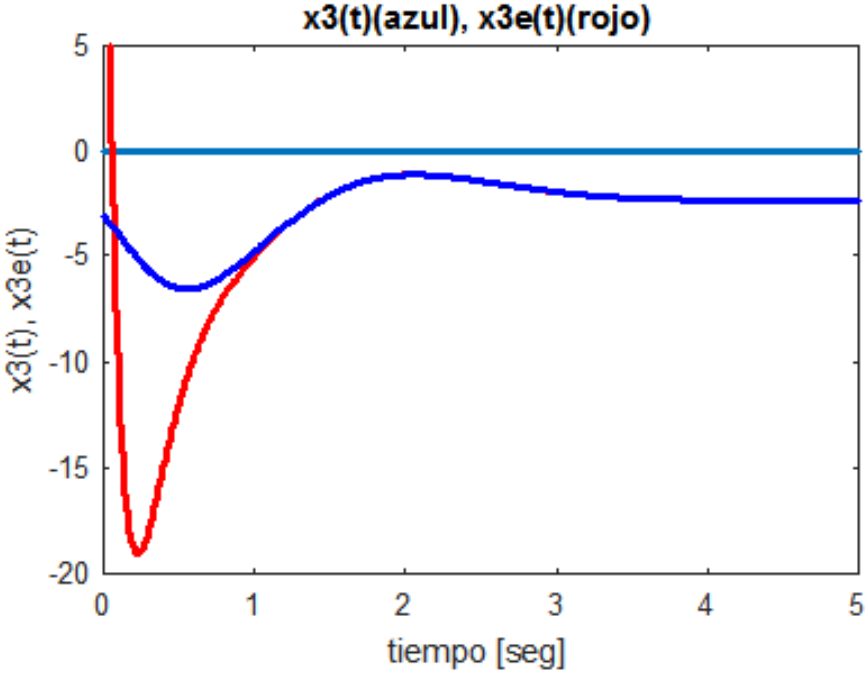
Observar que $y(t) = \hat{x}_1(t)$

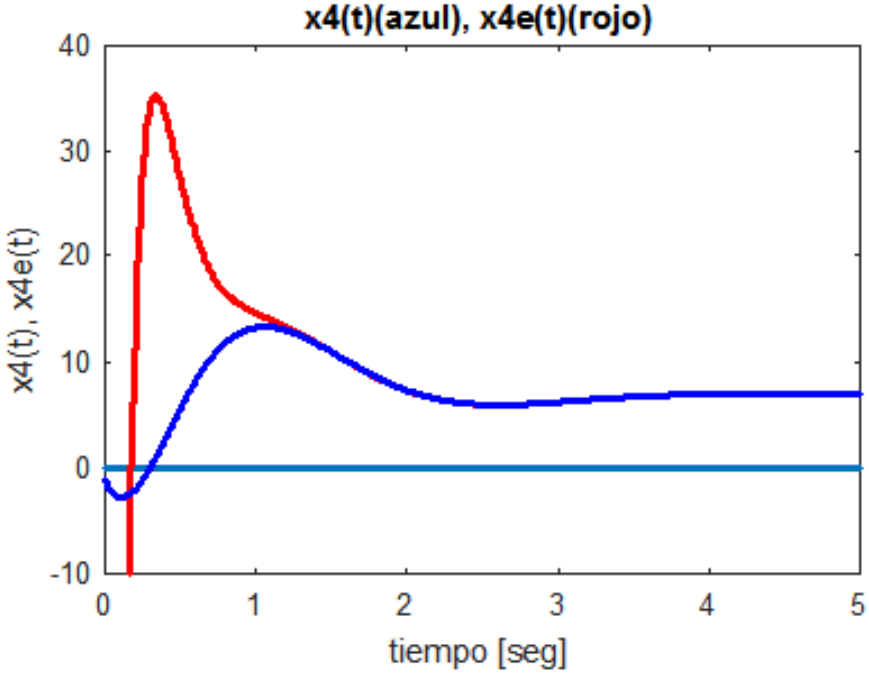
$$\hat{x}(t) = \begin{bmatrix} C \\ T \end{bmatrix}^{-1} \begin{bmatrix} y(t) \\ z(t) \end{bmatrix} \Rightarrow \begin{bmatrix} C \\ T \end{bmatrix} \hat{x}(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix} \Rightarrow \begin{cases} y(t) = C\hat{x}(t) = \hat{x}_1(t) \\ z(t) = T\hat{x}(t) \end{cases}$$

```
clear, clc, close all
planta_original=ss(zpk([-2+2i -2-2i],[-1 -1.5+2i -1.5-2i -3],2.344));
[A,B,C,D]=ssdata(planta_original);
% OBSERVADOR DE ORDEN REDUCIDO DE CHEN
% 1) Se pasa la planta a la FCC1b. La matriz de transformación es la matriz de observabilidad del
sistema original.
[Acc,Bcc,Ccc,Dcc]=ss2ss(A,B,C,D,obsv(A,C));
% Se describe la planta en las nuevas coordenadas
planta=ss(Acc,Bcc,Ccc,Dcc);
% Se simula la planta
t=[0:0.01:5];u=ones(length(t),1);
[y,t,x]=lsim(planta,u,t,[3 2 -3 -1]);
% 2) Se especifican los autovalores deseados del observador y se eligen F y L
lambdas_obs=[-5 -6 -7];
F=diag(lambdas_obs);L=rand(length(Acc)-1,1);
% 3)Se calcula T resolviendo la ecuación de Sylvester -F*T+T*Acc=L*Ccc
T = sylvester(-F,Acc,L*Ccc);
% 4) Se construye el siguiente sistema de orden n-1 que genera una estima de x(t).
Aob=F; Bob=[T*Bcc L]; Cob=eye(length(Aob));Dob=zeros(length(Aob),2);
observer=ss(Aob,Bob,Cob,Dob); [z,t]=lsim(observer,[u y]',t);
% 5) Se realiza la transformación final para obtener el estimador xe(t)
P=inv([Ccc;T]); zz=[y z]';
for i=1:length(t); xe(:,i)=P*zz(:,i); end;
subplot(2,2,1);plot(t,zeros(1,length(t)),'k',t,xe(1,:),'r',t,x(:,1),'b','LineWidth',2);
title('y(t)=x1(t)=x1e(t)'); xlabel('tiempo [seg]'); ylabel('y(t)=x1(t)=x1e(t)')
subplot(2,2,2);plot(t,zeros(1,length(t)),t,xe(2,:),'r',t,x(:,2),'b','LineWidth',2); axis([0 t(end) -5 5])
title('x2(t)(azul), x2e(t)(rojo)'); xlabel('tiempo [seg]'); ylabel('x2(t), x2e(t)')
subplot(2,2,3);plot(t,zeros(1,length(t)),t,xe(3,:),'r',t,x(:,3),'b','LineWidth',2); axis([0 t(end) -20 5])
title('x3(t)(azul), x3e(t)(rojo)'); xlabel('tiempo [seg]'); ylabel('x3(t), x3e(t)')
subplot(2,2,4);plot(t,zeros(1,length(t)),t,xe(4,:),'r',t,x(:,4),'b','LineWidth',2); axis([0 t(end) -10 40])
title('x4(t)(azul), x4e(t)(rojo)'); xlabel('tiempo [seg]'); ylabel('x4(t), x4e(t)')
```


y(t)=x1(t)=x1e(t)
y(t)=x1(t)=x1e(t)
tiempo [seg]
0
1
2
3
4
5


x2(t)(azul), x2e(t)(rojo)
x2(t), x2e(t)
tiempo [seg]
-5
0
5


x3(t)(azul), x3e(t)(rojo)
x3(t), x3e(t)
tiempo [seg]
-20
-15
-10
-5
0
5


x4(t)(azul), x4e(t)(rojo)
x4(t), x4e(t)
tiempo [seg]
-10
0
10
20
30
40

Función de Transferencia Observador de orden reducido−Controlador

Para este caso la fórmula cerrada del observador de orden reducido-controlador es muy complicada:

$$G_{oc}(s) = -\frac{U(s)}{Y(s)} = \left(1 + K\begin{bmatrix}\bar{C}\\ T\end{bmatrix}^{-1}\begin{bmatrix}0\\ (sI-F)^{-1}T\bar{B}\end{bmatrix}\right)^{-1} K\begin{bmatrix}\bar{C}\\ T\end{bmatrix}^{-1}\begin{bmatrix}1\\ (sI-F)^{-1}L\end{bmatrix}$$

```
clear, clc, close all
A = [0 1 0;0 0 1;0 -24 -10]; B = [0;10;-80]; C = [1 0 0]; D=0;
[Acc,Bcc,Ccc,Dcc]=ss2ss(A,B,C,D,obsv(A,C));
lambdas_cont = [-1+2*1i -1-2*1i -5]; lambdas_obs=[-10 -12]; K = acker(Acc,Bcc,lambdas_cont);
% OBSERVADOR DE ORDEN REDUCIDO CHEN
F = diag(lambdas_obs); L = rand(length(A)-1,1); T = sylvester(-F,Acc,L*Ccc);
syms s
G_Chen=inv(1+K*inv([Ccc;T])*[0 ; inv(eye(length(A)-1)*s-F)*T*Bcc])*K*...
    inv([Ccc;T])*[1;inv(eye(length(A)-1)*s-F)*L];
G_Chen=collect(G_Chen,s);
[num,den]=numden(G_Chen); cnum=coeffs(num); cnum=flip(cnum);
cden=coeffs(den);cden=flip(cden); cnum=double(cnum/cden(1)); cden=double(cden/cden(1));
G_Chen=zpk(tf(cnum,cden))
%                11.6 (s+5.593) (s+2.312)
%     G_Chen =  -----------------------
%                 (s+20.96) (s-1.956)
```

Es más fácil utilizar la diofantina

```
clear, clc, close all
A = [0 1 0;0 0 1;0 -24 -10]; B = [0;10;-80]; C = [1 0 0]; D=0;
lambdas_cont = [-1+2*1i -1-2*1i -5]; lambdas_obs=[-10 -12];
Acl=poly([lambdas_cont lambdas_obs]);
[b,a]=ss2tf(A,B,C,D);
[R,S] = diofantina(a,b,Acl);
Gc_Chen_diofantina=zpk(tf(S,R))
% Da la misma función de transferencia:
%                            11.6 (s+5.593) (s+2.312)
%     G_Chen_diofantina =  ------------------------
%                            (s+20.96) (s-1.956)
```