Acción integral en un controlador basado en observador adelantado (dLQR+dLQE, o place) discreto para proceso de 2o orden

Presentación en vídeo: http://personales.upv.es/asala/YT/V/ofexi.html

Este código funcionó correctamente con Matlab **R2021b**

Objetivo: Diseñar reguladores con/sin acción integral, cerrar el bucle y comprobar su comportamiento.

Tabla de Contenidos

Modelo del proceso a controlar

```
Ac=[0 1 ;-3 -0.2]; Bc=[0;3]; C=[1 0];
sysc=ss(Ac,Bc,C,0);
gan=dcgain(sysc)
```

```
gan = 1
```

```
tf(sysc)
```

```
ans =

         3
  ---------------
  s^2 + 0.2 s + 3

Continuous-time transfer function.
```

```
limiteSatU=1; %para simulacion con saturacion, antiwindup etc.
```

Vamos a discretizarlo:

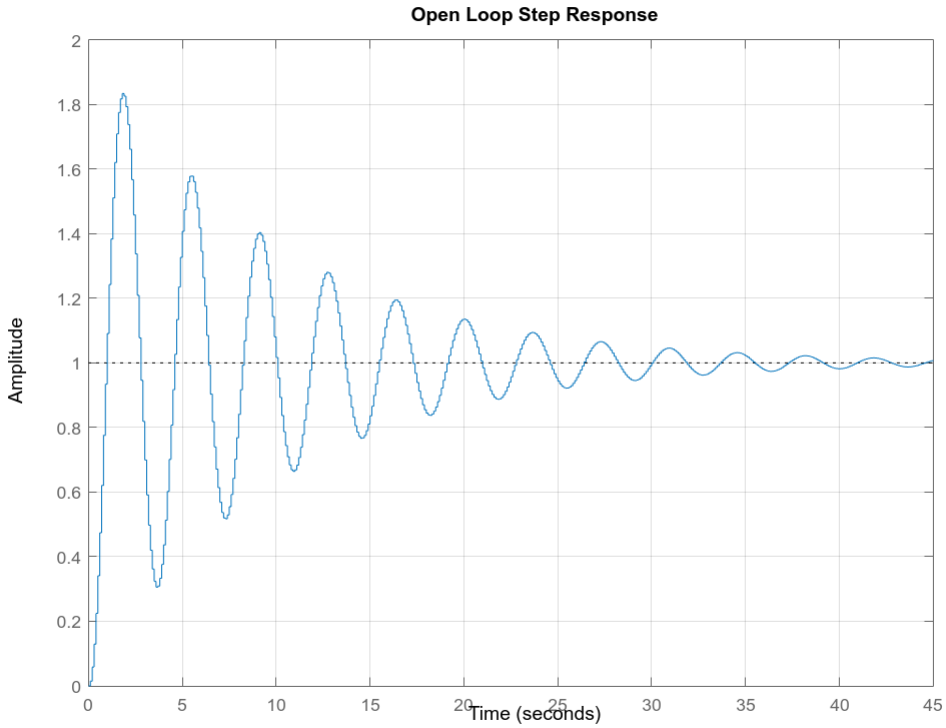
```
Ts=0.1; %periodo de muestreo
sysd=c2d(sysc,Ts,'zoh');
A=sysd.A
```

```
A = 2×2
    0.9851    0.0985
   -0.2955    0.9654
```

```
B=sysd.B
```

```
B = 2×1
    0.0149
    0.2955
```

```
sysd.StateName={'xp1','xp2'};
step(sysd,45), grid on, title("Open Loop Step Response")
```



Diseño de reguladores

Control sin acción integral place/LQG

```
Q=C'*C+0.01*eye(2); R=1;
K=dlqr(A,B,Q,R) %realimentación del estado LQR
```

```
K = 1×2
    0.3205    0.4554
```

```
K=place(A,B,[0.9+0.17*1j 0.9-0.17*1j]) %alternativa diseño asign. polos
```

```
K = 1×2
    0.3130    0.4937
```

```
eig(A-B*K)
```

```
ans = 2×1 complex
   0.9000 + 0.1700i
   0.9000 - 0.1700i
```

```
abs(ans) %estable, módulo < 1
```

```
ans = 2×1
    0.9159
    0.9159
```

```
L=dlqe(A,B,C,1,.1) %observador adelantado Kalman, ruido en entrada y medida
```

```
L = 2×1
    0.2970
    0.5207
```

```
L=place(A',A'*C',[0.8+.21*1j 0.8-0.21*1j])' %asign. polos
```

```
L = 2×1
    0.3021
    0.5378
```

```
eig(A-L*C*A)
```

```
ans = 2×1 complex
   0.8000 + 0.2100i
   0.8000 - 0.2100i
```

```
abs(ans) %estable, módulo < 1
```

```
ans = 2×1
    0.8271
    0.8271
```

Dados K y L podemos construir la ecuación de estado del regulador por realimentación de la salida:

```
Regulador_noI= ...
    ss((A-B*K)*(eye(2)-L*C), (A-B*K)*L, K*(eye(2)-L*C), K*L, Ts);
Regulador_noI.StateName={'xc1','xc2'};
zpk(Regulador_noI)
```

```
ans =

    0.36008 z (z-1.056)
  ------------------------
  (z^2 - 1.455z + 0.5855)

Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
```

El bucle cerrado tiene los polos deseados (separación):

```
BC_sinI=feedback(sysd,Regulador_noI); %BC de pert. entrada a salida
pole(BC_sinI)
```

```
ans = 4×1 complex
   0.8000 + 0.2100i
   0.8000 - 0.2100i
   0.9000 + 0.1700i
   0.9000 - 0.1700i
```

Control con con acción integral (place/LQG modelo ampliado)

Añadiremos al modelo una tercera ecuación de estado, perturbación cte. a la entrada.

Es un polo discreto en $z = 1$:

$$\xi_{k+1} = \begin{pmatrix} A & B \\ 0 & 1 \end{pmatrix} \xi_k + \begin{pmatrix} B \\ 0 \end{pmatrix} u_k, \qquad y_k = (C \quad 0)\xi_k + Du_k$$

```
Aaug=[A B;0 0 1]; Caug=[C 0]; Baug=[B;0];
L=dlqe(Aaug,eye(3),Caug,eye(3),.1) %observador de Kalman
```

```
L = 3×1
    0.9333
    1.9796
    0.8170
```

```
L=place(Aaug',Aaug'*Caug', [0.87+0.1*1j 0.87-0.1*1j 0.1])' %asig. polos
```

```
L = 3×1
    0.9218
    1.9324
    0.8171
```

```
eig(Aaug-L*Caug*Aaug)
```

```
ans = 3×1 complex
   0.1000 + 0.0000i
   0.8700 + 0.1000i
   0.8700 - 0.1000i
```

```
abs(ans)%estable, módulo < 1
```

```
ans = 3×1
    0.1000
    0.8757
    0.8757
```

```
Kaug=[K 1]; %La misma que sin pert. entrada, añadiendo "restar el estimado de la pert.'
```

Dados K y L (aumentados) podemos construir la ecuación de estado del regulador por realimentación de la salida:

```
Reguladors= ...
    ss((Aaug-Baug*Kaug)*(eye(3)-L*Caug), (Aaug-Baug*Kaug)*L, ...
         Kaug*(eye(3)-L*Caug),   Kaug*L,   Ts); %realim negativa u=-Kx_est
zpk(Regulador) %efectivamente, incorpora un integrador
```

```
ans =

  2.0597 z (z^2 - 1.945z + 0.9607)
  --------------------------------
     (z-1) (z-0.613) (z-0.1071)

Sample time: 0.1 seconds
Discrete-time zero/pole/gain model.
```

Cerremos un bucle de pert. entrada a salida:

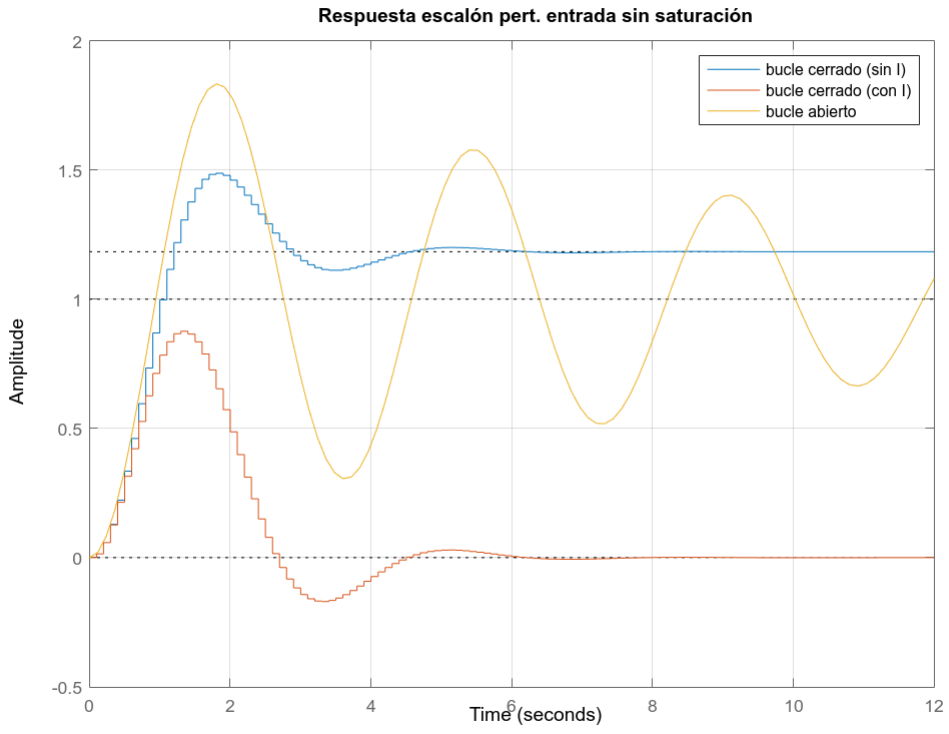
```
bc_du_to_y2 = feedback(sysd,Regulador);
eig(bc_du_to_y2) %estable, principio separación:
```

```
ans = 5×1 complex
   0.1000 + 0.0000i
   0.9000 + 0.1700i
   0.9000 - 0.1700i
   0.8700 + 0.1000i
   0.8700 - 0.1000i
```

Simulación bucles resultantes sin saturación

Simulación sin saturación (con comando feedback) pert. entrada escalón

```
step(BC_sinI, bc_du_to_y2, sysc, 12), grid on, title("Respuesta escalón pert. entrada s
legend("bucle cerrado (sin I)","bucle cerrado (con I)","bucle abierto")
```

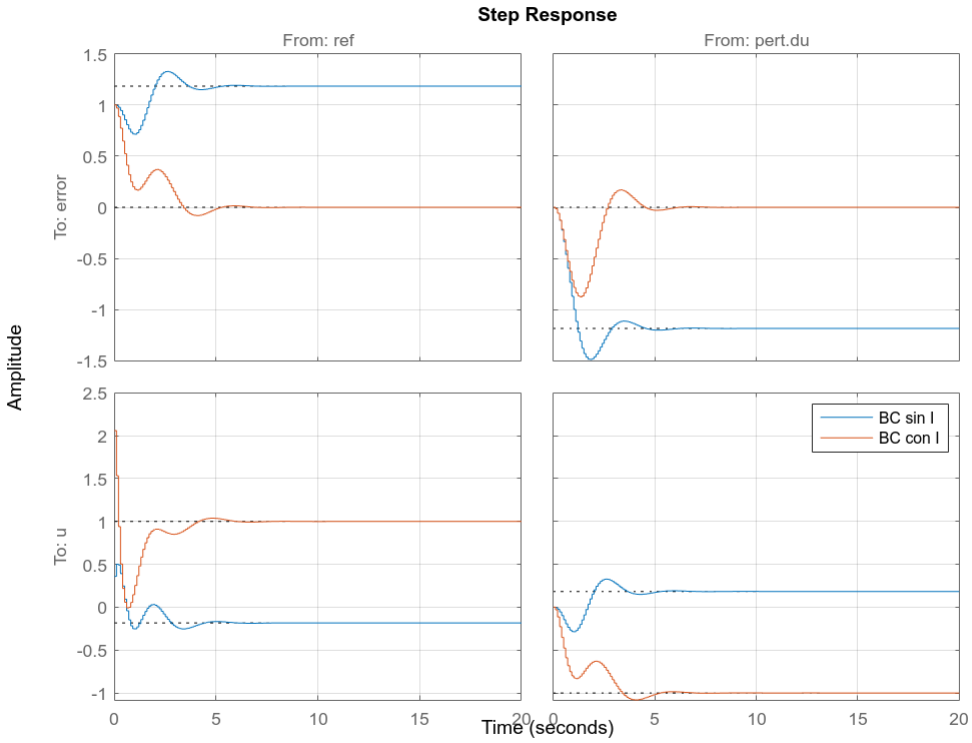


*****Un análisis completo de prestaciones requeriría simular ante referencia ($\approx$ ruido medida), etc. en tiempo y frecuencia, así como ver la acción de control que se aplica para cancelar perturbación de entrada o seguir referencias).

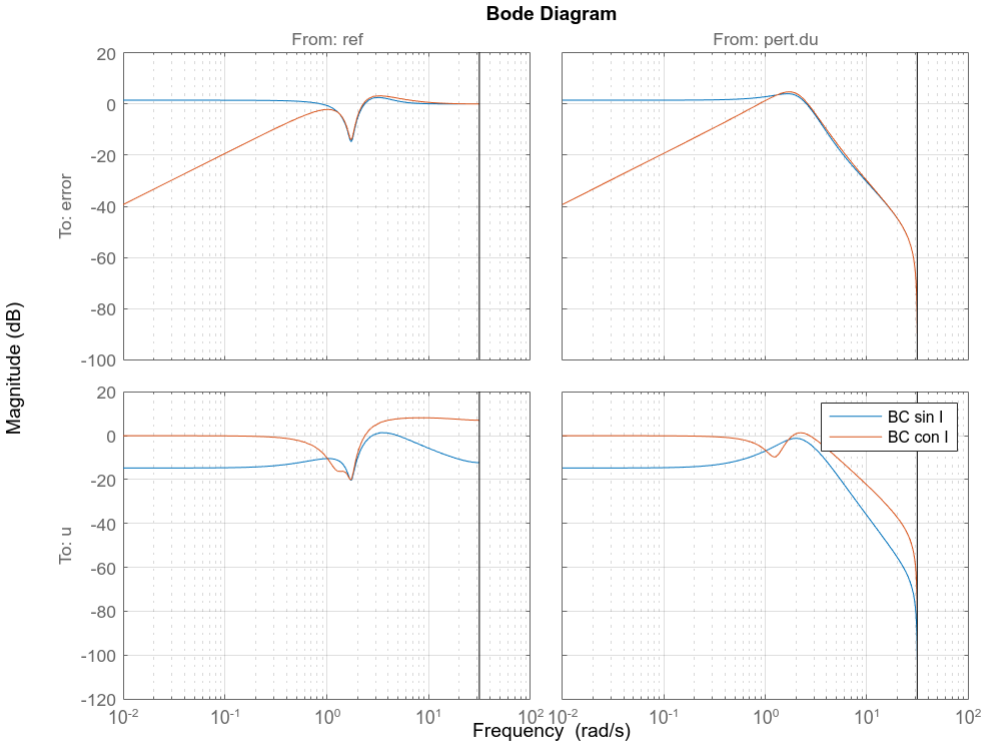
Este código a continuación presenta dicha simulación. Usa el comando `lft` para ahorrar en vez de usar cuatro "feedback". Detalles omitidos por brevedad.

```
PG=[1 0 0;0 0 1;1 0 0]+[-1;0;-1]*sysd*[0 1 1];
PG.InputName={'ref','pert.du','u'};PG.OutputName={'error','u','err'};
BC4noI=lft(PG,Regulador_noI);
BC4I=lft(PG,Regulador);

step(BC4noI,BC4I,20), grid on, legend("BC sin I","BC con I")
```




```
bodemag(BC4noI,BC4I), grid on, legend("BC sin I","BC con I")
```



*Si la entrada "ref" es una referencia conocida, se puede hacer 2GL, y corregir el error (omitido por brevedad). No será posible si es una perturbación "no medible".