Acción integral y antiwindup en un controlador basado en observador (dLQR+dLQE, o place) discreto: implementación y simulación discreta

Presentación en vídeo: http://personales.upv.es/asala/YT/V/ofexi2.html

Este código funcionó correctamente con Matlab **R2021b**

Objetivo: Diseñar reguladores con/sin acción integral, cerrar el bucle y comprobar su comportamiento. Comprobar el efecto antiwindup en un regulador basado en observador tanto en implementación explícita de "observador+realim. estado" como en implementación genérica regulador ss(Areg,Breg,Creg,Dreg).

Tabla de Contenidos

Modelo del proceso a controlar

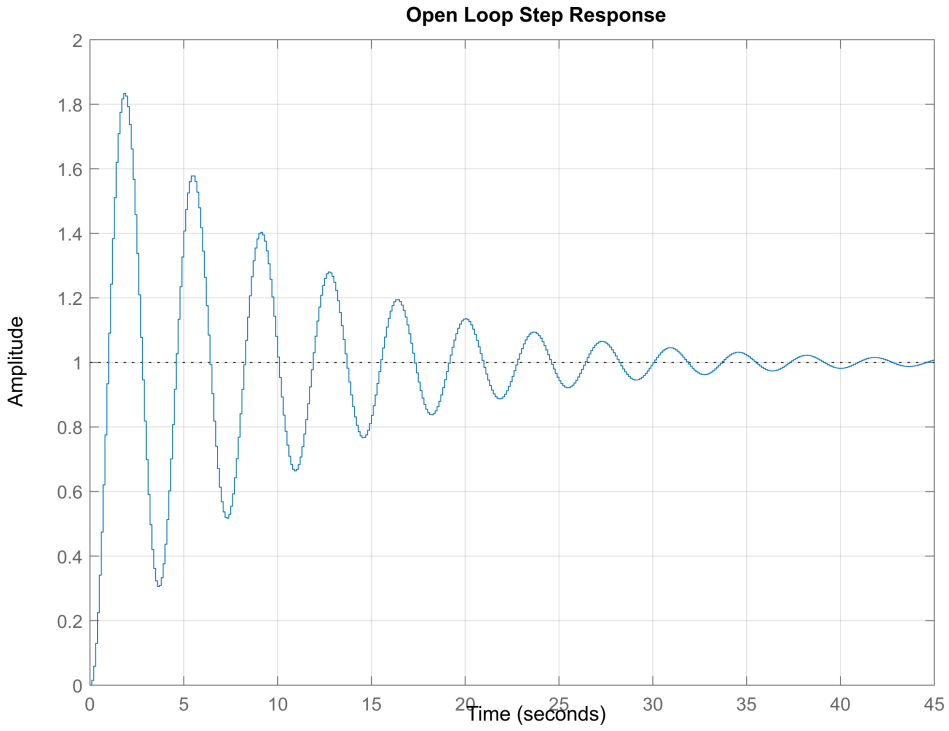
```
Ac=[0 1 ;-3 -0.2]; Bc=[0;3]; C=[1 0];
sysc=ss(Ac,Bc,C,0);
tf(sysc)
```

```
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; B=sysd.B;
step(sysd,45), grid on, title("Open Loop Step Response")
```



Diseño de reguladores

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 $z = 1$:

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

La realimentación del estado se calcula con modelo NO ampliado:

```
%Opción 1: control óptimo LQR
%Q=C'*C+0.01*eye(2); R=1;
%K=dlqr(A,B,Q,R)

%Opción 2 control por asign. polos:
K=place(A,B,[0.9+0.17*1j 0.9-0.17*1j]);
Kaug=[K 1]; %La misma que sin pert. entrada, añadiendo "restar el estimado de la pert.'
```

El observador lo haremos de orden 3:

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

```
%Opción 2: asign. polos
L=place(Aaug',Aaug'*Caug', [0.87+0.1*1j 0.87-0.1*1j 0.1])';
```

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

```
Reguladorr= ...
    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
```

```
Regulador =

  A =
             x1        x2        x3
   x1  -0.09947   0.09117         0
   x2    -1.614    0.8195         0
   x3   -0.8171         0         1

  B =
           u1
   x1    1.08
   x2   1.226
   x3  0.8171

  C =
           x1      x2      x3
   y1  -1.747  0.4937       1

  D =
         u1
   y1  2.06

Sample time: 0.1 seconds
Discrete-time state-space model.
```

```
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:

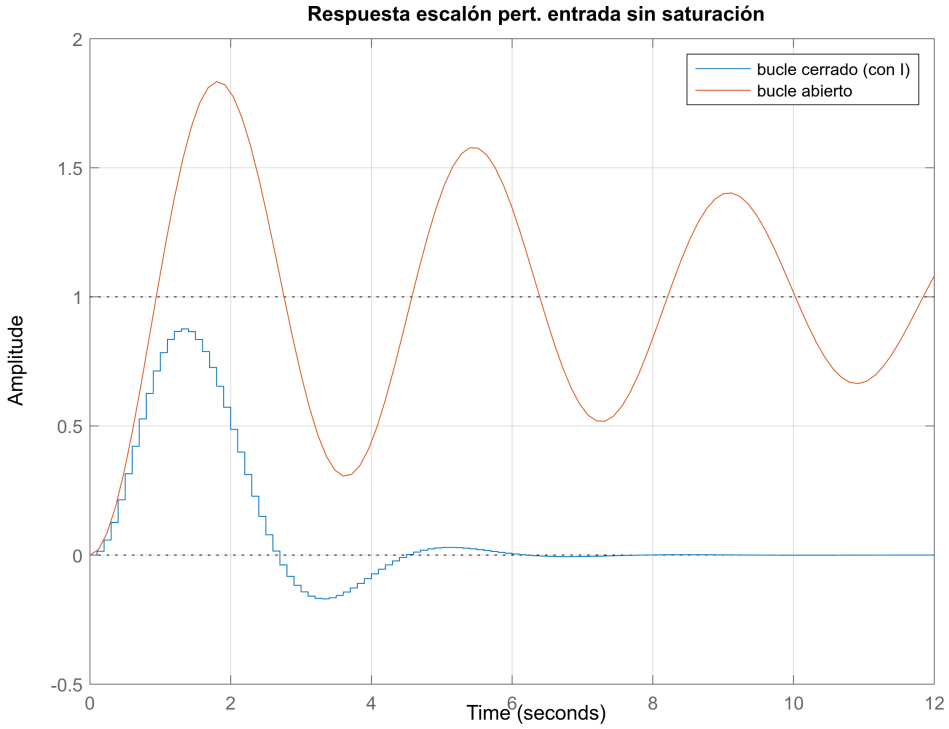
```
bc_du_to_y2 = feedback(sysd,Regulador);
eig(bc_du_to_y2) %estable, principio separación:
```

```
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_du_to_y2, sysc, 12), grid on, title("Respuesta escalón pert. entrada sin satura
legend("bucle cerrado (con I)","bucle abierto")
```



*****Un análisis completo de prestaciones requeriría simular ante referencia (≈ 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).

Simulación del regulador integral con saturación en actuador con/sin anti-windup

Ya no vale el comando "`feedback`", hay que usar Simulink o codificar en un bucle las ecuaciones en tiempo discreto, porque la saturación es no lineal. Probaremos varias opciones con/sin AW:

- Opción 1: programación separada explícita de observador y realim. estado estimado
- Opción 2: programación de regulador genérico ABCD

Incluiremos un poco de ruido y un punto de func. no cero.

```
ypf=0.25; %referencia constante
upf=dcgain(sysd)\ypf %esto realmente sería con no-lineal, deriv. a cero...
```

```
upf = 0.2500
```

```
Tsim=(0:1000)*Ts;
Nsamples=length(Tsim);
```

Condiciones iniciales (arbitrarias)

```
x=[1.5;0];%cond. iniciales estado (el proceso "real" es de orden 2)
xhat=[0;0;0]; %cond. iniciales observador de orden 3
```

Guardamos datos para representarlos gráficamente

```
grfy=zeros(1,Nsamples); %gráfica de la salida
grfu=zeros(1,Nsamples); %gráfica de la acción de control
grfusat=zeros(1,Nsamples); %gráfica de la acción de control saturada real
grfp=zeros(1,Nsamples); %gráfica de la perturbación que metemos en salida (no medible..
```

Opciones de implementación a comparar:

```
AW_conectado=true; %pruebas con/sin antiwindup
Implementacion_observer=false; %dos casos de implementación del control (y AW asociado)
```

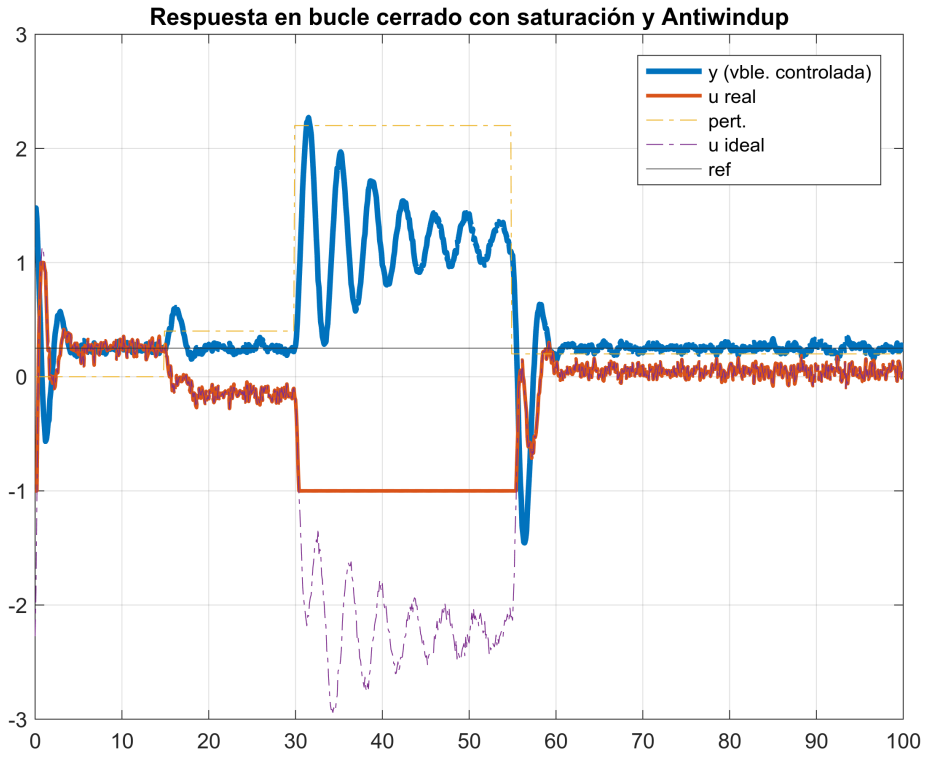
Bucle de simulación:

```
for k=1:Nsamples
    %simularemos ante perturbación no medible = tren de escalones
    pert=-0.4*(k<150)-1.8*(k<300)+2*(k<550)+0.2;
    grfp(k)=pert;

     %simulamos la medida del proceso real
    %y=C*x+pert; %Esto sería simular perturb. a la salida; al ser en "salida" y no "ent
    y_abs=C*x+0.02*randn();
    grfy(k)=y_abs;

    %implementamos el controlador
    yincr=y_abs-ypf;% incrementales
    if Implementacion_observer %observador explícitamente implementado

        xhat= xhat + L*(yincr-Caug*xhat);% current observer
        u_inc= -[K 1]*xhat;%realim. estado estimado
        u_abs=u_inc+upf; %unidades absolutas
        usat= max(-limiteSatU,min(limiteSatU,u_abs)); %acción real al proceso
        %si fuera una medida física, mejor... p.ej. si hay
        %control en cascada y el límite de saturación no fuera conocido,
        %"tracking mode antiwindup"

        if ~AW_conectado
            xhat= Aaug*xhat + Baug*u_inc; %hay "windup" si no lo tenemos en cuenta usat
        else %CON antiwindup
            uincsat=usat - upf;
            xhat= Aaug*xhat + Baug*uincsat; %basado en observador es fácil: alimentar c
```

```
        end

    else %implementamos un ABCD de regulador genérico, sin saber qué hay dentro:

        u_inc= Regulador.C*xhat + Regulador.D*(-yincr); %regulador lineal (unidades inc
        u_abs= u_inc+ upf;          %unidades absolutas
        usat= max(-limiteSatU,min(limiteSatU,u_abs)); %acción real al proceso
        %si fuera una medida física, mejor... p.ej. si hay
        %control en cascada y el límite de saturación no fuera conocido.
        %"tracking mode antiwindup"

        if ~AW_conectado
            xhat= Regulador.A*xhat + Regulador.B*(-yincr); %sin corregir nada por satur
        else %CON antiwindup
            L_antiwindup=Baug;
            u_incsat=usat-upf; %u incremental "real"...
            xhat= Regulador.A*xhat + Regulador.B*(-yincr) - L_antiwindup*(u_incsat-u_in
        end

    end

    grfusat(k)=usat; %gráfica acc. control real aplicada
    grfu(k)=u_abs; %gráfica acc. control calculada(absolutas)

    %Simulamos estado de la planta 1 período de muestreo más tarde
    x=A*x+B*(usat+pert+0.055*randn()); %la planta real actualiza su estado así (si no h
end
```

Una vez concluida la simulación, representamos resultados gráficamente:

```
plot(Tsim,grfy,LineWidth=3), hold on
plot(Tsim,grfusat,LineWidth=2)
plot(Tsim,[grfp;grfu]','-.')
hold off, yline(ypf), legend('y (vble. controlada)','u real','pert.','u ideal','ref',Lo
grid on,
if ~AW_conectado
    title("Respuesta en bucle cerrado con saturación (sin Antiwindup)")
else
     title("Respuesta en bucle cerrado con saturación y Antiwindup")
end
```




```
return
```

Conclusiones (código de regulador basado en observador adelantado "limpio")

Como no hay razón para no implementar antiwindup si se conocen límites de saturación, tendríamos:

- Implementación explícita "observador adelantado" con saturación interna:

```
yincr=LEER_SENSORES()-ypf; % incrementales, ypf podría ser referencia
xhat= xhat+L*(yincr-Caug*xhat);% current observer
u_inc= -[K 1]*xhat;%realim. estado estimado
u_abs=u_inc + upf; %unidades absolutas
usat= max(-limiteSatU,min(limiteSatU,u_abs)); %acción real al proceso
uincsat=usat-upf;
xhat= Aaug*xhat + Baug*uincsat;
ESCRIBIR_ACTUADOR(usat)
```

- Implementación "genérica" (pero sabiendo que es un regulador basado en observador, sea adelantado o no) con saturación interna:

```
yincr=LEER_SENSORES()-ypf; % incrementales, ypf podría ser referencia
u_inc= Regulador.C*xhat+Regulador.D*(-yincr); %regulador lineal (unidades incrementales
u_abs= u_inc + upf;          %unidades absolutas
usat= max(-limiteSatU,min(limiteSatU,u_abs)); %acción real al proceso
```

```
u_incsat=usat - upf; %u incremental "real"...
xhat= Regulador.A*xhat + Regulador.B*(-yincr) - B_aug*(u_incsat-u_inc);
ESCRIBIR_ACTUADOR(usat)
```

*****Si se mide la acción de control actualmente aplicada a la planta (*tracking mode* antiwindup), habría que hacer cambios, o si hiciéramos implementación 2GL pensando que ypf podría variar ("referencia") y calculásemos upf en el código de control. Por brevedad, no se discuten aquí.

Ventajas implementación observador explícito:

-- Se sabe el significado de xhat, quien examina el código sabe que se trata de un observador del estado.

-- Aunque K, L se hayan calculado con modelo linealizado, se puede simular un modelo no lineal xhat=EcEstado(xhat,usat) para disminuir error de linealización (ojo con incrementales/absolutas, claro), es una especie de control por "modelo interno" donde se simula el proceso dentro del controlador.

Ventajas implementación genérica:

-- El código sirve para cualquier regulador lineal realizable, dado que existe una representación interna `ss(Ar,Br,Cr,Dr,Ts)...` valdría para PIDs, observador adelantado, no adelantado, control $\mathscr{H}_2$, $\mathscr{H}_\infty$, ...

-- Con el diseño adecuado, `xhat= Regulador.A*xhat+Regulador.B*(-yincr)-L_antiwindup*(u_incsat-u_inc)` permite hacer antiwindup en cualquier regulador lineal.

-- A cambio (desventajas), no se "entiende" cómo o por qué funciona, y no se puede incorporar un "modelo interno" no-lineal del proceso de forma sencilla.