Simulación con ODE45 de un bucle cerrado control PD de péndulo (inestable)

© 2022, Antonio Sala Piqueras. Universitat Politècnica de València. Todos los derechos reservados.

Este código funcionó sin errores en Matlab **R2021b**

Presentación en vídeo: http://personales.upv.es/asala/YT/V/bcode45.html

Objetivo: comprender cómo simular un bucle cerrado de control usual, sin bucles algebraicos, con integración numérica `ode45`.

Tabla de Contenidos

Modelado

Proceso a controlar

El modelo no lineal de un péndulo es:

$$\frac{dp}{dt}=v, \qquad \frac{dv}{dt}=\frac{g}{l^2}\cdot\sin(p)-\frac{f}{Ml^2}\cdot v+\frac{1}{Ml^2}T$$

siendo T el par en el eje de rotación.

```
M=0.5;g=9.81; fric=0.25; l=1;%parámetros constantes
```

Como el controlador será lineal, debemos especificar puntos de operación de sus entradas y salidas para hacer los cambios adecuados

```
pto_oper_y=0; %equilibrio superior
pto_oper_T=0; %se mantiene sin fuerza, teóricamente
```

En este caso, por "casualidad" el "cero" en incrementales y en absolutas coincide, pero no tiene por qué ser así en un caso general.

Representación en variables de estado no lineal:

```
Model.Proceso.EcuacionEstado=@(t,x,T) [x(2); g*sin(x(1))-fric/M*x(2)+T/M ];
Model.Proceso.EcuacionSalida=@(t,x) x(1); %medida disponible para realimentación
ordenProceso=2;
```

Controlador lineal PD

Usaremos un PD (con filtro de ruido en derivada), sintonizado "a ojo":

```
s=tf('s');
Kp=7;Kd=1.3;filterbandwith=20;
KLaplace=Kp+Kd*s/(s/filterbandwith+1);
K=ss(KLaplace)
```

```
K =

  A =
        x1
   x1  -20

  B =
       u1
   x1  32

  C =
           x1
   y1  -16.25

  D =
       u1
   y1  33

Continuous-time state-space model.
```

```
ordenControlador=size(K.A,1)
```

```
ordenControlador = 1
```

Ecuación de estado del controlador

```
Model.Controlador.EcuacionEstado=@(t,x,y) K.A*x+K.B*(-(y-pto_oper_y)); %realim. negativ
```

Nótese que la realimentación negativa, le entra $-\Delta y =$ -(y-pto_oper_y) al regulador, es lo mismo que si entrara (pto_oper_y-y), esto es, error de bucle si pto_oper_y lo consideramos como "referencia".

Ecuación de salida del controlador

En no-lineal, hay que pasar de incrementales a absolutas, hay que saturar, etc... no es sólo `T=K.C*x+K.D*(-incr_y)`

```
Limit_T_sup=3.5;Limit_T_inf=-3.5; %límites de saturación
satura=@(T) max(min(T,Limit_T_sup),Limit_T_inf); %función estática que satura T entre s

Model.Controlador.EcuacionSalida= ...
    @(t,x,y) satura(pto_oper_T + K.C*x+K.D*(-(y-pto_oper_y)));
% si hubiera acción integral en controlador, habría que añadir antiwindup.
```

```
% Aquí no es necesario
```

Simulación Numérica

Preparamos para el formato específico que `ode45` necesita (estados de proceso y controlador combinados en un vector de estado conjunto):

```
IndiceEstadosProceso=1:ordenProceso
```

```
IndiceEstadosProceso = 1×2
     1     2
```

```
IndiceEstadosControl=ordenProceso+(1:ordenControlador)
```

```
IndiceEstadosControl = 3
```

```
odefun= @(t,Estado) ...
    EcEstadoBucleCerrado(t,Estado(IndiceEstadosProceso),Estado(IndiceEstadosControl),Mo
```

Establecemos **condiciones iniciales**

```
EstadoIniProceso=[pi/30;0]; %probar pi/30 (no satura), pi/10 (diferencia por saturación
EstadoIniControlador=0;
x0=[EstadoIniProceso;EstadoIniControlador];%cond. inicial proceso+regulador
```

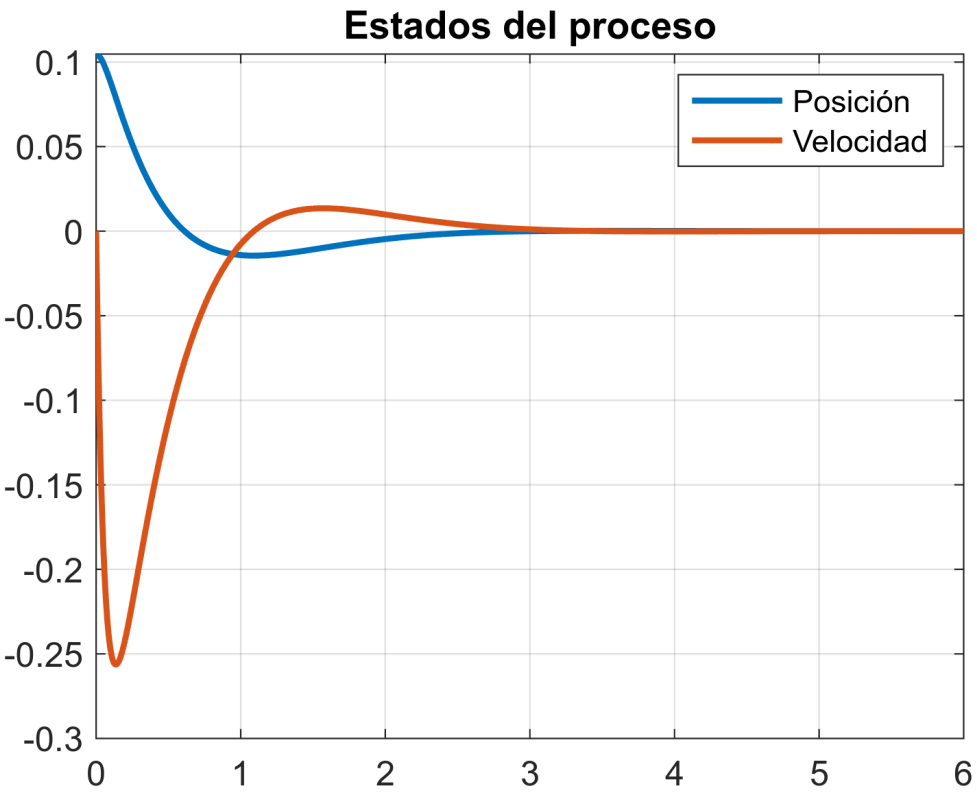
Esta es la línea que hace la simulación (integración numérica):

```
t_final=6;
[T,X]=ode45(odefun, [0 t_final], x0, odeset('RelTol',1e-6,'AbsTol',1e-6));
```

Gráficas de resultados

Este es, directamente, el resultado que ha calculado ode45:

```
plot(T,X(:,IndiceEstadosProceso),LineWidth=2)
title("Estados del proceso"), legend("Posición","Velocidad")
grid on
```

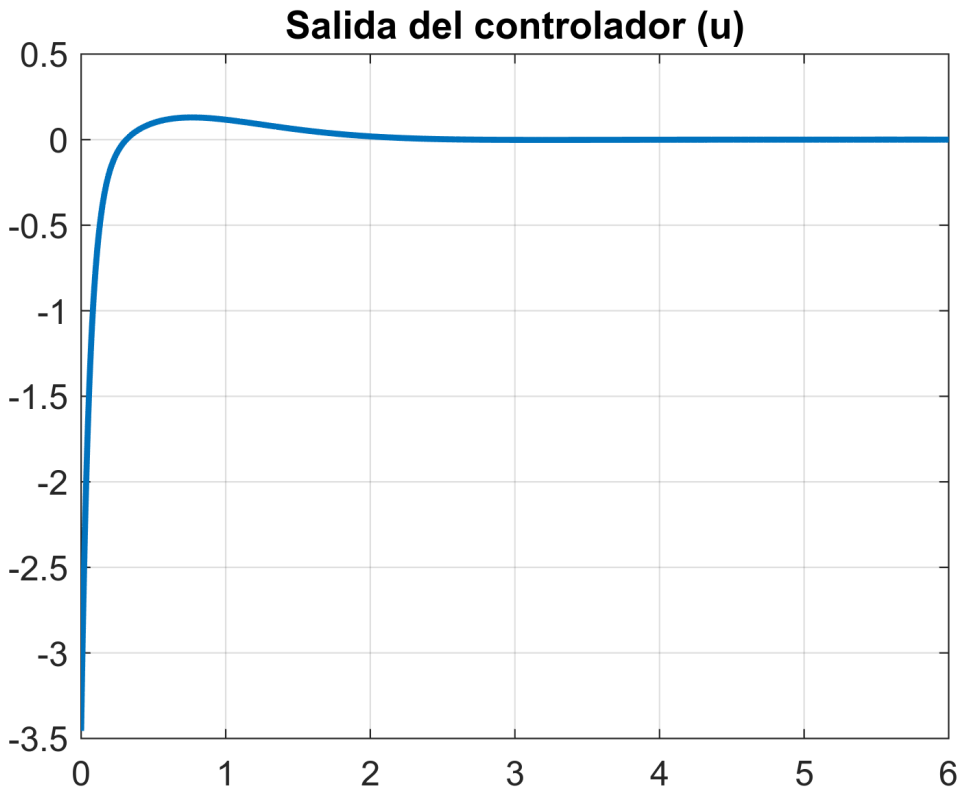


Calculemos las salidas, que es lo que suele tener significado físico, para representarlas gráficamente

```
Ns=length(T);
y=zeros(1,Ns); u=zeros(1,Ns);
for k=1:Ns
    y(k)=Model.Proceso.EcuacionSalida(T(k),X(k,IndiceEstadosProceso)'); %Salidas Proces
    u(k)=Model.Controlador.EcuacionSalida(T(k),X(k,IndiceEstadosControl)',y(k)); %Salid
end

plot(T,y,'k',LineWidth=3)
grid on
title("Salida del proceso (y)")
```


```
plot(T,u,LineWidth=2)
grid on
title("Salida del controlador (u)")
```



Comparación con modelo linealizado (control systems toolbox)

Modelo Linealizado

$$\frac{d\bar{p}}{dt} = \bar{v}, \qquad \frac{d\bar{v}}{dt} = \frac{g}{l^2} \cdot \cos(p_0)\bar{p} - \frac{f}{Ml^2} \cdot \bar{v} + \frac{1}{Ml^2}\bar{T}$$

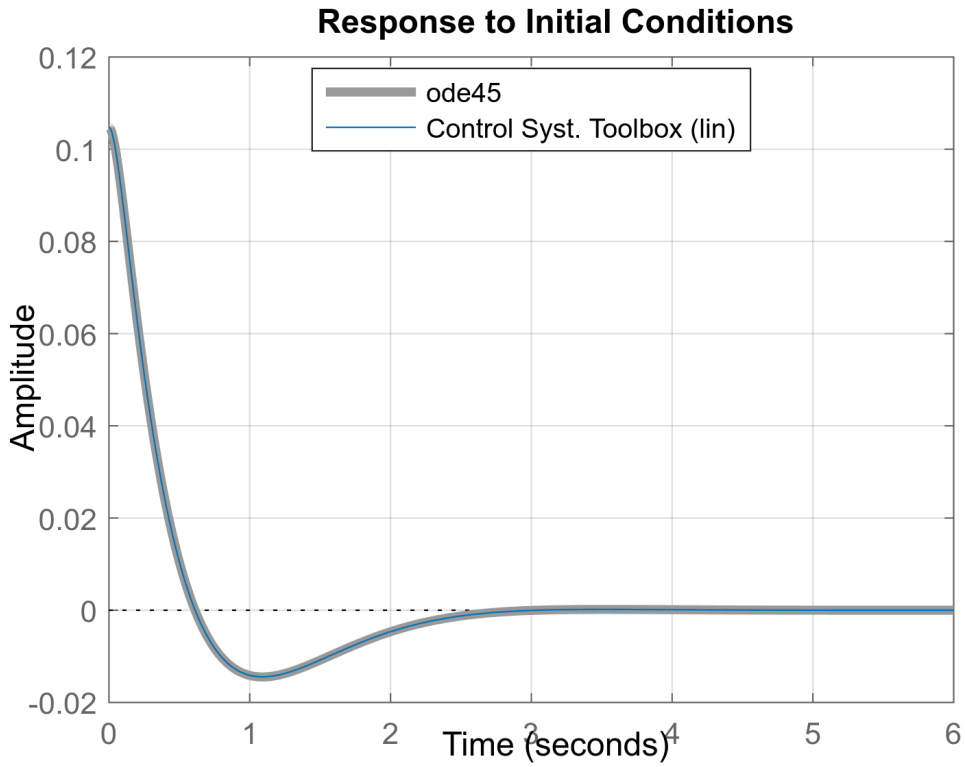
```
A=[0 1; cos(pto_oper_y)*g/l^2 -fric/M/l^2]; B=[0; 1/M/l^2]; C=[1 0]; %D=0, si no no fun
sysLin=ss(A,B,C,0); %para el control system toolbox, no lo necesitamos aquí, pero por c
```

Comparemos con la salida linealizada que daría el control systems toolbox:

```
BC=feedback(sysLin,K);
pole(BC)
```

```
ans = 3×1 complex
 -16.8711 + 0.0000i
  -1.8144 + 1.2942i
  -1.8144 - 1.2942i
```

```
plot(T,y,Color=[.6 .6 .6],LineWidth=3), hold on
initial(BC,x0-[pto_oper_y;0;0],t_final), grid on, hold off,legend("ode45","Control Syst
```



Funciones auxiliares: ecuación de estado de bucle cerrado

```
function dxdtbc=EcEstadoBucleCerrado(t,EstadoProceso,EstadoControl, M)
  Medidas=M.Proceso.EcuacionSalida(t,EstadoProceso); %D=0, no resolvemos bucles algebra
  dEstadoControladordt=M.Controlador.EcuacionEstado(t,EstadoControl,Medidas);
  u=M.Controlador.EcuacionSalida(t,EstadoControl,Medidas);
  dEstadoProcesodt=M.Proceso.EcuacionEstado(t,EstadoProceso,u);
  dxdtbc=[dEstadoProcesodt;dEstadoControladordt];
end
```

***Nota 1:** si "Medidas" dependiera directamente de "u", habría bucle algebraico; aquí suponemos que no hay, porque si lo hubiese, el código sería incorrecto.

***Nota 2:** si el controlador fuera "estático" (proporcional) entonces no haría falta el argumento "EstadoControlador", y las lineas 57 y 58 se cambiarían a

```
u=M.Controlador.EcuacionSalida(t,Medidas);
```

Se dejan los detalles al lector, como ejercicio propuesto para implementar un control proporcional... aunque NO será estable con este proceso, si se analiza el lugar de las raíces de la linealización.