LINEALIZACIÓN de modelo de movimiento transversal de un tiovivo

Este código funcionó con Matlab **R2022b**

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/tiovl1.html

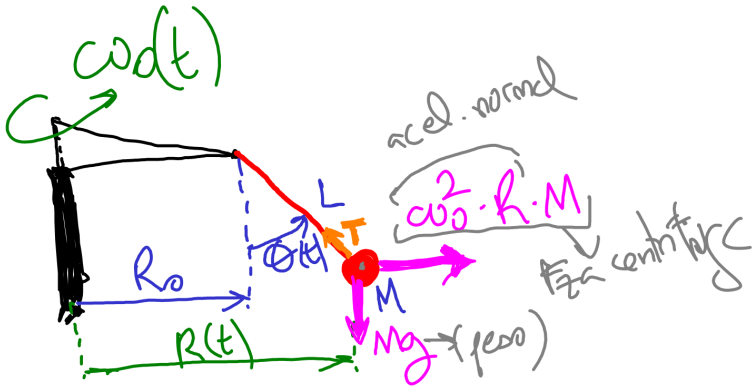
http://personales.upv.es/asala/YT/V/tiovl2.html

Objetivos: Comprender el modelado físico de un tiovivo como la fotografía (sólo la oscilación "lateral" del columpio), y su simulación. Linealizar el mismo y comparar la solución original con la linealizada.

*Image from jana_lsb0 at Pixabay

Tabla de Contenidos

Modelo dinámico



- Parámetros constantes:

```
syms M L g R_0 real %letras para las ecuaciones "bonitas"
Mnominal=20; Lnominal=4; R_0nominal=3; %valores nominales para cálculos numéricos
```

- Entrada:

```
syms omega_0 real %velocidad angular tiovivo
omega_0nominal=pi/4; %valor nominal de diseño (para equilibrio, linealización, etc.)
```

- Otras variables en las ecuaciones del sistema:

```
syms theta real %ángulo plano vertical
syms T real %tensión cuerda
syms R real %distancia al eje del tiovivo
syms omega_1 real %velocidad angular plano vertical
syms dthetadt domega1dt real %derivadas de los estados (se deben despejar en forma norm
syms Qres real %par resultante
ModeloNoNormalizado = [dthetadt == omega_1; ... %dinámica velocidad ang.
                       M*L^2*domega1dt == Qres; ... %dinámica aceleración ang.
                       Qres== -M*g*L*sin(theta) + omega_0^2*R*L*cos(theta)*M-128*omega_1;
                       R == R_0+L*sin(theta)]; %geometría
```

Sustituyamos las constantes de buenas a primeras, por no manejar tanta "letra". Resulta:

```
ModelNoNormalizSustituido=subs(ModeloNoNormalizado,{R_0,L,M,g},{R_0nominal,Lnominal,Mno
```

```
ModelNoNormalizSustituido =
```

$$\begin{pmatrix} \text{dthetadt} = \omega_1 \\ 320\,\text{domega1dt} = \text{Qres} \\ \text{Qres} = 80\,R\cos(\theta)\,{\omega_0}^2 - 128\,\omega_1 - 784\sin(\theta) \\ R = 4\sin(\theta) + 3 \end{pmatrix}$$

Tenemos 4 ecuaciones y 4 señales "incógnitas": $\theta(t)$, $R(t)$, $\omega_1(t)$, $Q_{res}(t)$. Modelo algebraico-diferencial completo.

Punto de funcionamiento (equilibrio)

Si la velocidad de giro del tiovivo (entrada) está en el valor nominal, podemos escribir

```
ModelEq=subs(ModelNoNormalizSustituido, {dthetadt,domega1dt,omega_0}, {0,0,omega_0nomir
```

```
ModelEq =
```

$$\begin{pmatrix} 0 = \omega_1 \\ 0 = \mathrm{Qres} \\ \mathrm{Qres} = 5\,R\,\pi^2\cos(\theta) - 784\sin(\theta) - 128\,\omega_1 \\ R = 4\sin(\theta) + 3 \end{pmatrix}$$

```
PtoFuncionamientoNominal=vpasolve(ModelEq)
```

```
PtoFuncionamientoNominal = struct with fields:
       Qres: 0
          R: 3.969650545730160647610419891120 3
    omega_1: 0
      theta: 0.244851891384362678119386530709 54
```

Con lo que cuando la velocidad de entrada sea "parecida" al nominal, pues theta será parecido a lo de arriba. Lo gastaremos, pues, como "punto de tangencia" para linealizaciones.

***Nota:** La solución del problema no es única (hay un equilibrio "inestable" tipo "motociclista en curva" si el columpio se sujetase con una barra a compresión), pero como la obtenida arriba sí tiene sentido físico para nuestro problema, no modificamos el código que llama vpasolve.

Forma Normalizada (ec. de estado) del modelo no lineal

Si el modelo está "completo", podremos despejar todo en función de estados y entradas (y constantes), en particular las derivadas:

```
sol=solve(ModelNoNormalizSustituido,{dthetadt,domega1dt,Qres,R});
Estado=[theta;omega_1]; %pos y vel angulares
dEstado_dt= simplify([sol.dthetadt; sol.domega1dt],15)
```

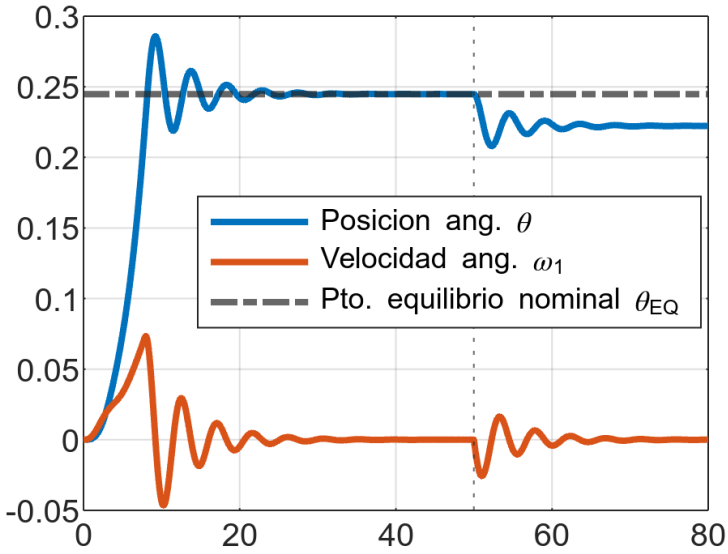
```
dEstado_dt =
```

$$\begin{pmatrix} \omega_1 \\ \frac{3\,\omega_0^2\cos(\theta)}{4} - \frac{49\sin(\theta)}{20} - \frac{2\,\omega_1}{5} + \omega_0^2\cos(\theta)\sin(\theta) \end{pmatrix}$$

Simulación ode45 del modelo dinámico no lineal

```
dEstado_num=matlabFunction(dEstado_dt,Vars={Estado,omega_0}); %traducir a función Matla
Tiempofinal=80;
x0=[0;0];
u=@(t) min(0.1*t,omega_0nominal)-0.03*(t>50); %entrada
```

```
opts=odeset("RelTol",1e-5,"AbsTol",1e-5);
dEstado_paraode45=@(t,VectorDeEstado) dEstado_num(VectorDeEstado,u(t));
[Tiempos,Estados]=ode45(dEstado_paraode45,[0 Tiempofinal],x0,opts);
plot(Tiempos,Estados,LineWidth=2), grid on,
yline(eval(PtoFuncionamientoNominal.theta),'-.',LineWidth=2), xline(50,":")
legend("Posicion ang. \theta","Velocidad ang. \omega_1","Pto. equilibrio nominal \theta
```



Linealización

Alternativa 1: forma original (no normalizada)

Se puede linealizar cualquier modelo, esté o no en forma "normalizada"... realmente se recomienda lo segundo (son menos ecuaciones), pero a lo mejor un modelo no normalizado tiene más ecuaciones pero más sencillas de comprender (y de linealizar). Vamos a hacerlo con el "no" normalizado, pues, al menos esta vez.

Recordamos modelo y punto funcionamiento:

```
theta_PF=eval(PtoFuncionamientoNominal.theta)
```

```
theta_PF = 0.2449
```

```
R_PF=eval(PtoFuncionamientoNominal.R)
```

```
R_PF = 3.9697
```

```
ModelNoNormalizSustituido
```

```
ModelNoNormalizSustituido =
```

$$\begin{pmatrix} \text{dthetadt} = \omega_1 \\ 320\,\text{domega1dt} = \text{Qres} \\ \text{Qres} = 80\,R\cos(\theta)\,{\omega_0}^2 - 128\,\omega_1 - 784\sin(\theta) \\ R = 4\sin(\theta) + 3 \end{pmatrix}$$

La linealización, pues significa cambiar $z = f(x)$, siendo z cualquier expresión o término en el modelo no lineal, por $\Delta z = \frac{\partial f}{\partial x}_{|p.f.} \Delta x$ (recta tangente en unidades "incrementales", proporcionalidad en caso x monovariable) o, si x es multivariable

$\Delta z = \sum_i \frac{\partial f}{\partial x_i}_{|p.f.} \Delta x_i$ simbolizando plano o hiperplano tangente.

Si las ecuaciones ya eran lineales (como la primera y segunda), entonces se "copian" y punto (eliminando constantes si hubiera).

Por ejemplo:

$$\frac{d\,\Delta\theta}{dt} = \frac{d(\theta - \theta_{PF})}{dt} = \frac{d\theta}{dt} = \omega_1 = \omega_1 - 0 = \omega_1 - \omega_{1,PF} = \Delta\omega_1$$

Definimos variables incrementales:

```
syms theta_inc omega_1inc Qres_inc omega_0inc R_inc d_thinc_dt d_om1inc_dt  real %nueva
```

Aunque los comandos "diff" y "jacobian" me ayudan con las derivadas, vamos primero a hacerlo "a mano" para aprender (luego gastaremos eso):

```
ModeloNoNormalizadoLinealizado = ...
    vpa( [ d_thinc_dt == omega_1inc;  %ya era lineal, sólo hay que cambiar el "cero" de
      320*d_om1inc_dt == Qres_inc;  %idem de arriba
      Qres_inc == 80*cos(theta_PF)*omega_0nominal^2 *R_inc+80*R_PF*(-sin(theta_PF))*ome
      R_inc == 4*cos(theta_PF) *theta_inc    ],8)
```

```
ModeloNoNormalizadoLinealizado =
```

$$\begin{pmatrix} d_{\text{thinc,dt}} = \omega_{\text{1inc}} \\ 320.0\,d_{\text{om1inc,dt}} = \text{Qres}_{\text{inc}} \\ \text{Qres}_{\text{inc}} = 47.87613\,R_{\text{inc}} + 483.96219\,\omega_{\text{0inc}} - 128.0\,\omega_{\text{1inc}} - 808.1031\,\theta_{\text{inc}} \\ R_{\text{inc}} = 3.880693\,\theta_{\text{inc}} \end{pmatrix}$$

Con lo que podemos pasarlo a representación normalizada:

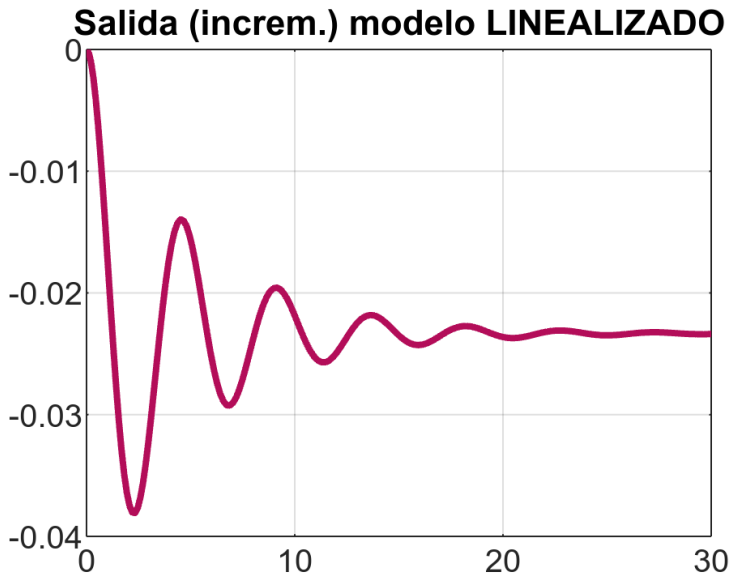
```
solLin=solve(ModeloNoNormalizadoLinealizado,{d_thinc_dt,d_om1inc_dt,Qres_inc,R_inc});
EcEstadoLinealizado=vpa([solLin.d_thinc_dt;solLin.d_om1inc_dt],6)
```

```
EcEstadoLinealizado =
```

$$\begin{pmatrix} \omega_{1\text{inc}} \\ 1.51238\,\omega_{0\text{inc}} - 0.4\,\omega_{1\text{inc}} - 1.94472\,\theta_{\text{inc}} \end{pmatrix}$$

Para simularlo el modelo linealizado, pasamos a "numérico" y usamos ode 45, como con el no lineal:

```
EcEstadoLinealizadoNum=matlabFunction(EcEstadoLinealizado,Vars={[theta_inc;omega_1inc],
uinc=@(t) -0.03; %entrada INCREMENTAL
[TiemposLin,EstadosLin]=ode45(@(t,x) EcEstadoLinealizadoNum(x,uinc(t)),[0 30],[0;0],opt
plot(TiemposLin,EstadosLin(:,1),Color=[0.7 0.05 0.35],LineWidth=2), grid on, title("Sal
```

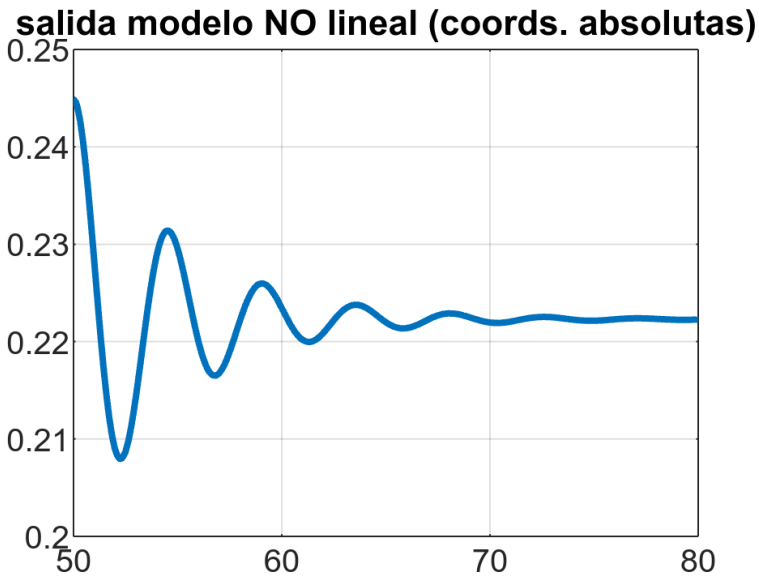


Vamos a superponerlo con el "no lineal"

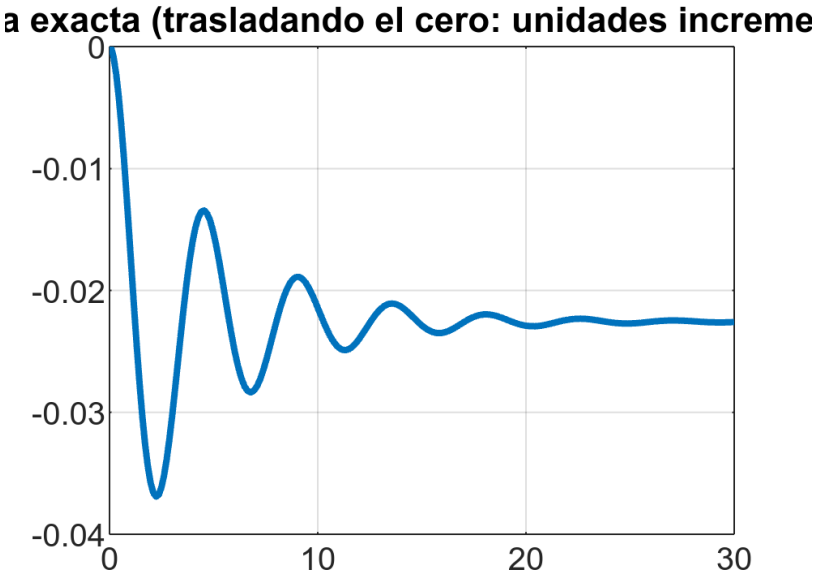
```
i1=find(Tiempos>=50,1)
```

```
i1 = 280
```

```
plot(Tiempos(i1:end),Estados(i1:end,1),LineWidth=2), grid on, title("salida modelo NO l
```



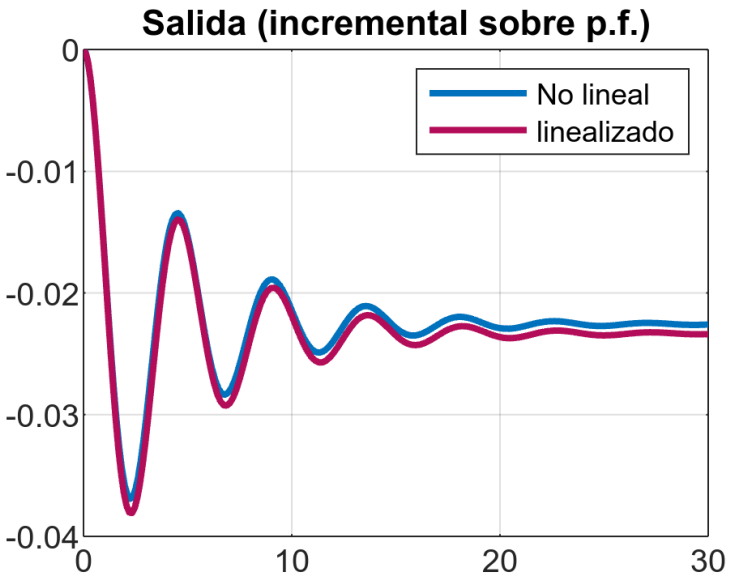

```
plot(Tiempos(i1:end)-Tiempos(i1),Estados(i1:end,1)-theta_PF,LineWidth=2), grid on %camb
title("Salida exacta (trasladando el cero: unidades incrementales)")
```



La gráfica realmente es "la misma": sólo han cambiado las etiquetas de los ejes.

Para comparar ambos modelos, superponemos y todo se ve más claro:

```
plot(Tiempos(i1:end)-Tiempos(i1),Estados(i1:end,1)-theta_PF,LineWidth=2), grid on %camb
hold on
plot(TiemposLin,EstadosLin(:,1),Color=[0.7 0.05 0.35],LineWidth=2) %superponemos lineal
hold off, legend("No lineal","linealizado"), title("Salida (incremental sobre p.f.)")
```



Alternativa 2: Linealización del modelo no-lineal en representación como ec. de estado normalizada

El resultado final sería el mismo si hubiésemos partido de la forma normalizada, que posiblemente sea recomendable: al haber eliminado variables habrá menos "letras" de las que hacer derivadas parciales.

```
theta_PF
```

```
theta_PF = 0.2449
```

```
dEstado_dt
```

```
dEstado_dt =
```

$$\begin{pmatrix} \omega_1 \\ \frac{3\,\omega_0^2\cos(\theta)}{4} - \frac{49\sin(\theta)}{20} - \frac{2\,\omega_1}{5} + \omega_0^2\cos(\theta)\sin(\theta) \end{pmatrix}$$

Haciendo "manualmente" derivadas, hay que derivar respecto a ω_0 y respecto a θ... respecto a ω_1 todo sale ya lineal, por lo que la linealización es "inmediata".

```
SegundaEcLin= 6/4*omega_0nominal*cos(theta_PF)*omega_0inc+3/4*omega_0nominal^2*(-sin(th
      -49/20*cos(theta_PF)*theta_inc -2/5*omega_1inc +2*omega_0nominal*cos(theta_PF)*si
      +omega_0nominal^2*(-sin(theta_PF)^2+cos(theta_PF)^2)*theta_inc;
vpa(SegundaEcLin,6)
```

```
ans =
```

$$1.51238\,\omega_{0\text{inc}} - 0.4\,\omega_{1\text{inc}} - 1.94472\,\theta_{\text{inc}}$$

```
dEstado_dt_linealizado = vpa([omega_1inc; SegundaEcLin],6)
```

```
dEstado_dt_linealizado =
```

$$\begin{pmatrix} \omega_{1\text{inc}} \\ 1.51238\,\omega_{0\text{inc}} - 0.4\,\omega_{1\text{inc}} - 1.94472\,\theta_{\text{inc}} \end{pmatrix}$$

Para hacer un montón de derivadas parciales tenemos "jacobian":

```
A=eval(subs(jacobian(dEstado_dt,Estado),{theta,omega_0},{theta_PF,omega_0nominal}))
```

```
A = 2×2
         0    1.0000
   -1.9447   -0.4000
```

```
B=eval(subs(jacobian(dEstado_dt,omega_0),{theta,omega_0},{theta_PF,omega_0nominal}))
```

```
B = 2×1
         0
    1.5124
```

Introducción del modelo linealizado en el "control systems toolbox"

Usualmente la respuesta ante entradas sencillas (escalón) y las propiedades de la respuesta (amplitud, frecuencia, duración de oscilaciones) en el modelo linealizado pueden calcularse "analíticamente" con fórmulas de teoría de EDO lineales, en vez de con simulación `ode45`... de hecho, esa es la justificación más importante de la utilidad de la linealización, porque para simular con ode45 ya tendríamos el no lineal, je.

La forma normalizada de sistemas lineales en tiempo continuo es:

$$\frac{dx}{dt} = Ax + Bu$$

$$y = Cx + Du$$

formando las matrices A, B, C, D con los coeficientes que multiplican a las distintas variables...

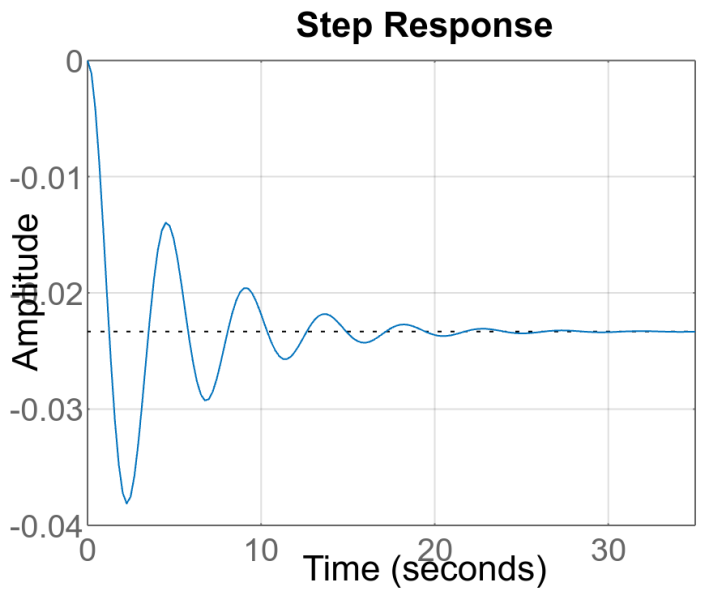
```
A=eval(jacobian(EcEstadoLinealizado,[theta_inc,omega_1inc]))
```

```
A = 2×2
         0    1.0000
   -1.9447   -0.4000
```

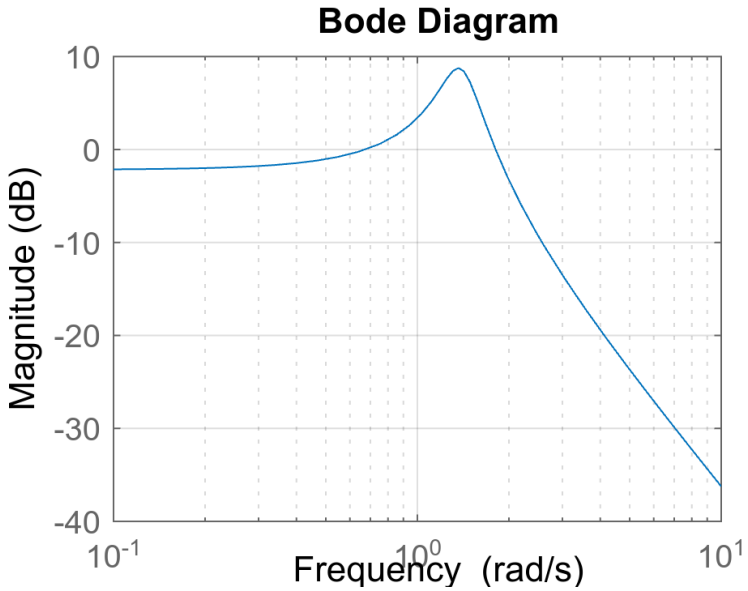
```
B=eval(jacobian(EcEstadoLinealizado,omega_0inc))
```

```
B = 2×1
         0
    1.5124
```

```
C=[1 0]; D=0; %sólo me interesa posición en mi aplicación
syslin=ss(A,B,C,D);
step(syslin*-0.03), grid on %debe coincidir (tolerancias aparte) con la salida de ode45
```




```
bodemag(syslin), grid on %respuesta ante senoides (amplitud, escala logarítmica)
```



Comentarios finales: ¿linealización por cambio de variable (inversión de no-linealidad)?

En el código de arriba se ha seguido, "sin hacerse preguntas filosóficas", la metodología estándar de linealización.

Pero, dado que la entrada siempre sale como ω_0^2, podríamos haber pensado en hacer un cambio de variable y haber llamado $u := \omega_0^2$, teniendo un modelo:

```
syms u real
Modelo2=subs(dEstado_dt,omega_0^2,u)
```

```
Modelo2 =
```

$$\begin{pmatrix} \omega_1 \\ \frac{3\,u\cos(\theta)}{4} - \frac{49\sin(\theta)}{20} - \frac{2\,\omega_1}{5} + u\cos(\theta)\sin(\theta) \end{pmatrix}$$

Obviamente, seguiría siendo no lineal por los senos y cosenos de θ y por el producto $u\cos(\theta)$, pero al menos una causa de error (cambiar parábola por recta) la habríamos eliminado.

Es una **opción RECOMENDADA**, pues, seguir este camino, en muchos casos.

*****El probema es que en aplicaciones de control o filtrado podrían ser necesarios bloques que hicieran ***cuadrados*** o ***raíces cuadradas***; por ejemplo, si para posicionar en cierto "ángulo deseado" calculamos que se necesita $u = 2$, ello implica $\omega_0 = \sqrt{2}$.

Esto no es problema si es implementado tecnológicamente como "*control por computador*" o "*procesado digital de señal*", pero podría serlo si la lógica debe ser realizada con "*electrónica analógica*" o "*electromecánica*".