

Modelado de oscilaciones transversales de un tiovivo

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

Este código funcionó con Matlab R2022b

Presentaciones en vídeo:

<http://personales.upv.es/asala/YT/V/tiovs.html>

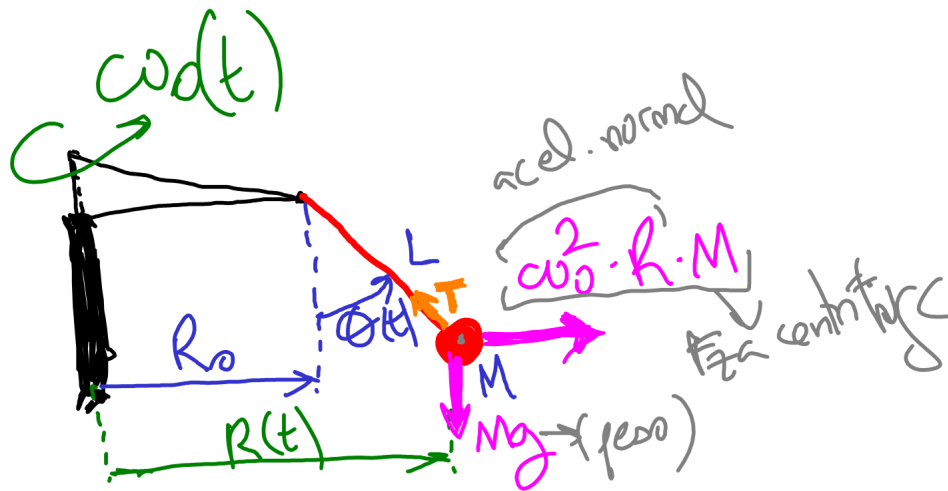
<http://personales.upv.es/asala/YT/V/tiovd.html>

Objetivos: Comprender el modelado físico de un tiovivo como la figura (sólo la oscilación lateral del columpio), y su simulación.



*Image from [jana_lsb0](#) at [Pixabay](#) , Pixabay License 2022 (free even for commercial use).

Abstracción del problema, diagrama esquemático



• 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 (las que sean "de interés" se etiquetarán como salidas):

```
syms theta real %ángulo plano vertical
syms T real %tensión cuerda
syms R real %distancia al eje del tiovivo
```

Modelado estático

NOTA: Esto no es necesario si tenemos las ecuaciones de abajo dinámicas (el estático es el caso particular con derivadas igualadas cero), pero por comprobarlo y para entender mejor el problema y las diferencias con modelado dinámico, lo hacemos...

Ante una velocidad angular de entrada dada, se alcanzará el equilibrio cuando las tres fuerzas estén equilibradas; hacemos balances horizontal y vertical:

```
ModeloEquilibrio = [ T*sin(theta) == M*omega_0^2*R ; ... %balance horizontal
                    R == R_0+L*sin(theta); %geometría
                    T*cos(theta) == M*g]; %balance vertical
```

Son 3 ecs y 3 incógnitas (T , R , θ)... debe tener solución... El comando "solve" se me vuelve un poco loco

```
solEstatica=solve(ModeloEquilibrio,{T,R,theta}) %da error...
```

Warning: Unable to find explicit solution. For options, see help.

```
solEstatica = struct with fields:
```

```
    T: [0×1 sym]
    R: [0×1 sym]
  theta: [0×1 sym]
```

Los ordenadores son un poco "tontos"... Bueno, igual nosotros tampoco estamos del todo "inspirados".

Por lo que vamos a hacer un ejemplo "numérico" en el "punto de operación nominal"

```
ValorsR0LMgOmega0={R_0nominal,Lnominal,Mnominal,9.8,omega_0nominal};
ME3=subs(ModeloEquilibrio,{R_0,L,M,g,omega_0},ValorsR0LMgOmega0)
```

```
ME3 =
```

$$\begin{cases} T \sin(\theta) = \frac{5 R \pi^2}{4} \\ R = 4 \sin(\theta) + 3 \\ T \cos(\theta) = 196 \end{cases}$$

```
solEquilibrio=vpasolve(ME3)
```

```
solEquilibrio = struct with fields:
  R: 3.9696505457301606476104198911203
  T: 202.02577448880263203052541900639
  theta: 0.24485189138436267811938653070954
```

```
eval(solEquilibrio.theta*180/pi) %ángulo en grados
```

```
ans = 14.0290
```

Por ejemplo la sobrecarga (el incremento de tensión) respecto al peso para "sujetar" la fuerza centrífuga será:

```
ExtraTension=eval(solEquilibrio.T-Mnominal*9.8)
```

```
ExtraTension = 6.0258
```

*Nota sobre la solución dada por `vpasolve`:

El problema *no tiene solución única* por dos razones:

- 1.) Que theta podría valer $0.25 + 2k\pi$ para cualquier k entero.
- 2.) Que existe una solución "inestable" con el columpio hacia arriba y tensión negativa (necesitaríamos una sujección, claro, con una barra a compresión, no valdría con cuerda o cadena) del estilo "motociclista en una curva".

Como el tipo de soluciones 2. no es el que buscamos en esta aplicación, estamos contentos con la solución de `vpasolve` y ya está... Si no lo estuviéramos, necesitaríamos darle un "estimado inicial" para que hiciera una búsqueda local a su alrededor, ver la documentación de `vpasolve` para detalles.

Modelado dinámico

Otras señales:

```
syms omega_1 real %velocidad angular plano vertical
syms dthetadt domegaldt real %derivadas de los estados (se deben despejar en forma normalizada)
syms Qres real %par resultante
```

```
ModeloNoNormalizado = [dthetadt == omega_1; ...
    M*L^2*domegaldt == Qres; ...
    Qres== -M*g*L*sin(theta) + omega_0^2*R*L*cos(theta)*M-200*omega_1;
    R == R_0+L*sin(theta)]; %geometría
```

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

Inciso: modelo estático como caso particular del dinámico

```
MNN2=subs(ModeloNoNormalizado,{R_0,L,M,g,omega_0},ValorsR0LMgOmega0); %valores nominales
ModelEq=subs(MNN2,{dthetadt,domegaldt},{0,0}) %derivadas a cero
```

```
ModelEq =
```

$$\begin{pmatrix} 0 = \omega_1 \\ 0 = Q_{res} \\ Q_{res} = 5 R \pi^2 \cos(\theta) - 784 \sin(\theta) - 200 \omega_1 \\ R = 4 \sin(\theta) + 3 \end{pmatrix}$$

```
solve(ModelEq)
```

Warning: Unable to solve symbolically. Returning a numeric solution using vpasolve.

```
ans = struct with fields:
    Qres: 0
    R: 3.9696505457301606476104198911203
    omega_1: 0
    theta: 0.24485189138436267811938653070954
```

Forma Normalizada (ec. de estado)

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

```
symvar(ModeloNoNormalizado)
```

```
ans = (L M Qres R R_0 domegaldt dthetadt g omega_0 omega_1 theta)
```

```
sol=solve(ModeloNoNormalizado,{dthetadt,domegaldt,Qres,R})
```

```
sol = struct with fields:
    dthetadt: omega_1
    domegaldt: -(- M*cos(theta)*sin(theta)*L^2*omega_0^2 - M*R_0*cos(theta)*L*omega_0^2 + M*g*sin(theta)*L
    Qres: M*cos(theta)*sin(theta)*L^2*omega_0^2 + M*R_0*cos(theta)*L*omega_0^2 - M*g*sin(theta)*L - 2
    R: R_0 + L*sin(theta)
```

```
Estado=[theta;omega_1]; %pos y vel angulares
dEstado_dt= simplify([sol.dthetadt; sol.domegaldt],15)
```

```
dEstado_dt =
```

$$\begin{pmatrix} \omega_1 \\ \omega_0^2 \cos(\theta) \sin(\theta) - \frac{g \sin(\theta)}{L} - \frac{200 \omega_1}{L^2 M} + \frac{R_0 \omega_0^2 \cos(\theta)}{L} \end{pmatrix}$$

Simulación

```
dEstado_conconstantesnumericas=subs(dEstado_dt,{R_0,L,M,g},{R_0nominal,Lnominal,Mnominal,gnominal})
```

```
dEstado_conconstantesnumericas =
```

$$\begin{pmatrix} \omega_1 \\ \frac{3 \omega_0^2 \cos(\theta)}{4} - \frac{49 \sin(\theta)}{20} - \frac{5 \omega_1}{8} + \omega_0^2 \cos(\theta) \sin(\theta) \end{pmatrix}$$

El comando matlabFunction me traduce de "expresión simbólica" a función de Matlab...

así me evita poner el código "teclado a mano":

```
dxdt=@(x,u) [x(2); ...
u^2*cos(x(1))*3/4-49*sin(x(1))/20+u^2*cos(x(1))*sin(x(1))];
```

que yo debería hacer si la ecuación de arriba la hubiese obtenido con "lápiz y papel" en vez de Symbolic Toolbox de Matlab.

```
dEstado_num=matlabFunction(dEstado_conconstantesnumericas,Vars={Estado,omega_0})
```

```
dEstado_num = function_handle with value:
```

```
@(in1,omega_0)[in1(2,:);in1(2,:).*(-5.0./8.0)-sin(in1(1,:)).*(4.9e+1./2.0e+1)+omega_0.^2.*cos(in1(1,:))];
```

```
Tfinal=40;
x0=[0;0];
u=@(t) min(0.1*t,omega_0nominal); %entrada
opts=odeset("RelTol",1e-5,"AbsTol",1e-5);
dEstado_paraode45=@(t,VectorDeEstado) dEstado_num(VectorDeEstado,u(t))
```

```
dEstado_paraode45 = function_handle with value:
```

```
@(t,VectorDeEstado)dEstado_num(VectorDeEstado,u(t))
```

```
[Tiempos,Estados]=ode45(dEstado_paraode45,[0 Tfinal],x0,opts);
size(Tiempos)
```

```
ans = 1x2
    201     1
```

```
plot(Tiempos,Estados,LineWidth=2), grid on,
yline(eval(solEquilibrio.theta),'-.',LineWidth=2)
legend("Posicion ang. \theta","Velocidad ang. \omega_1","Pto. equilibrio nominal \theta")
```

