

# Modelado de un móvil (Montaña Rusa) sobre curva paramétrica

© 2021, Antonio Sala Piqueras, *Universitat Politècnica de València*. Todos los derechos reservados.

Presentación en vídeo: <http://personales.upv.es/asala/YT/V/rollerco.html>

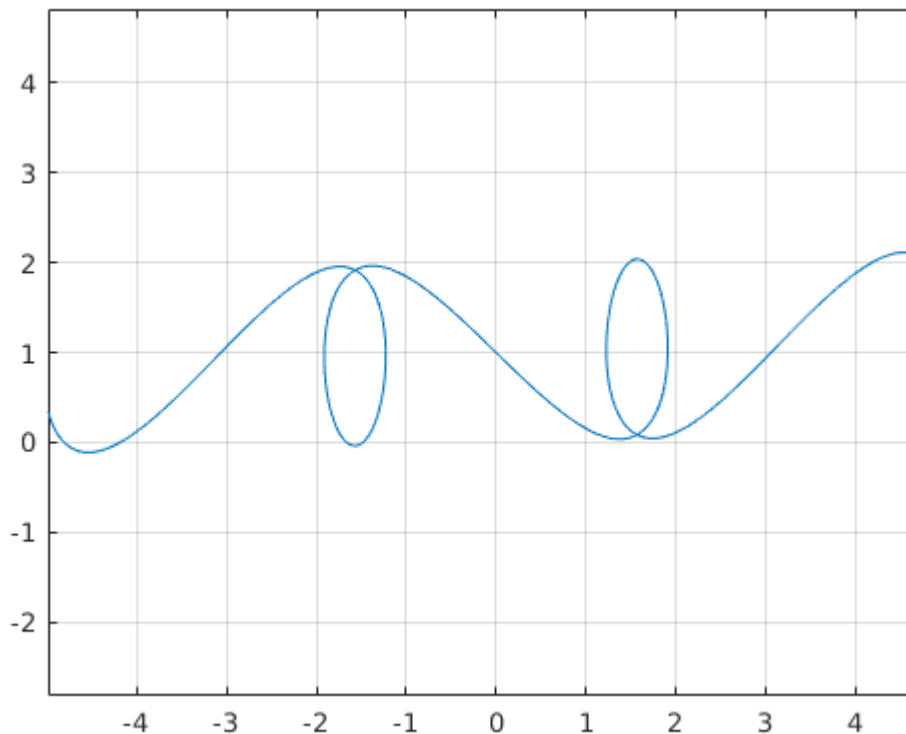
Este código funcionó sin errores en Matlab R2021a

**Objetivo:** modelar una "montaña rusa", por varios métodos (Newton, Euler-Lagrange).

## Tabla de Contenidos

Cinemática.....	2
Dinámica (Newton, balance de fuerzas).....	3
Euler-Lagrange 2nd kind.....	5
Simulación temporal.....	7

```
b=0.0258; %coeficiente de fricción
M=1;g=9.8;
xq= @(q) sin(q)+0.5*q;
yq= @(q) 1-sin(1.5*q)+0.025*xq(q);
qr=-8:0.005:7.5;
plot(xq(qr),yq(qr)), axis equal, grid on
```



# Cinemática

```
syms q v_q a_q real
x=xq(q); y=simplify(yq(q));
r=[x;y]
```

$$r = \begin{pmatrix} \frac{q}{2} + \sin(q) \\ \frac{q}{80} - \sin\left(\frac{3q}{2}\right) + \frac{\sin(q)}{40} + 1 \end{pmatrix}$$

```
drdq=diff(r,q)
```

$$\text{drdq} = \begin{pmatrix} \cos(q) + \frac{1}{2} \\ \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \end{pmatrix}$$

```
%prueba:
veloc=drdq*v_q
```

$$\text{veloc} = \begin{pmatrix} v_q \left( \cos(q) + \frac{1}{2} \right) \\ v_q \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right) \end{pmatrix}$$

```
acel=simplify(jacobian(veloc,[q v_q])*[ v_q; a_q])
```

$$\text{acel} = \begin{pmatrix} a_q \left( \cos(q) + \frac{1}{2} \right) - v_q^2 \sin(q) \\ \left( \frac{9 \sin\left(\frac{3q}{2}\right)}{4} - \frac{\sin(q)}{40} \right) v_q^2 + a_q \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right) \end{pmatrix}$$

Calculemos el ángulo de la trayectoria con la horizontal

```
coseno=simplify(drdq(1)/sqrt(drdq(1)^2+drdq(2)^2))
```

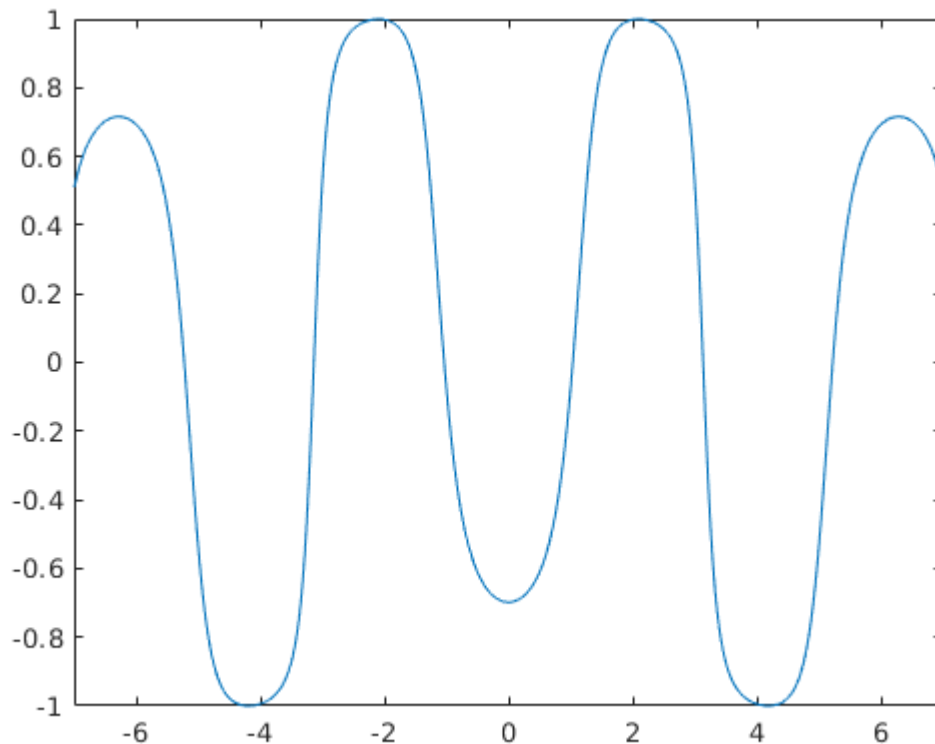
```
coseno =
```

$$\frac{\cos(q) + \frac{1}{2}}{\sqrt{\left(\frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80}\right)^2 + \left(\cos(q) + \frac{1}{2}\right)^2}}$$

```
seno=simplify(drdq(2)/sqrt(drdq(1)^2+drdq(2)^2))
```

```
ccc = function_handle with value:
```

```
@(q)1.0./sqrt((cos(q.*(3.0./2.0)).*(-3.0./2.0)+cos(q)./4.0e+1+1.0./8.0e+1).^2+(cos(q)+1.0./2.0).^2).*
```



```
seno =
```

$$\frac{\frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80}}{\sqrt{\left(\frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80}\right)^2 + \left(\cos(q) + \frac{1}{2}\right)^2}}$$

Las fórmulas elegidas darán un ángulo de la curva "en la dirección en la que crece  $q$ ". Por lo tanto, el vector "tangente" no coincidirá con el vector  $T$  en Frenet-Serret cuando el móvil se desplace "hacia atrás" con  $v_q < 0$ .

## Dinámica (Newton, balance de fuerzas)

```
F_friccion_vect=-b*veloc %ya es tangencial, porque la velocidad es tangente a la trayectoria
```

```
F_friccion_vect =
```

$$\begin{pmatrix} -\frac{129 v_q \left(\cos(q) + \frac{1}{2}\right)}{5000} \\ -\frac{129 v_q \left(\frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80}\right)}{5000} \end{pmatrix}$$

Balance de fuerzas horizontal y vertical

```
syms R real %reacción
```

```
ecs=[ M*acel(1)==-R*seno+F_friccion_vect(1); %horizontal x  
      M*acel(2)==-M*g+R*coseno+F_friccion_vect(2)]; %vertical y
```

Aunque el seno y coseno siempre son de un ángulo apuntando hacia adelante, si es el otro con la misma pendiente (hacia atrás), el signo de tanto el seno como el coseno cambian... esto se absorberá en un cambio de signo en  $R$ , con lo que el desarrollo está correcto aunque el móvil se mueva "hacia atrás".

```
sol=solve(ecs,a_q,R);  
a_qNewton=simplify(sol.a_q,150)
```

```
a_qNewton =
```

$$-\frac{\frac{206529 v_q}{5000} - 94080 \sigma_1 + 1568 \cos(q) - \frac{774 v_q \sigma_1}{125} + \frac{206529 v_q \cos(q)^2}{1250} - 3202 v_q^2 \sin(q) + \frac{9288 v_q \sigma_1^2}{25} - 32}{14400 \sigma_1^2 - 480 \sigma_1 \cos(q)}$$

where

$$\sigma_1 = \cos\left(\frac{3q}{2}\right)$$

$$\sigma_2 = \sin\left(\frac{3q}{2}\right)$$

```
sR=simplify(sol.R,150)
```

```
sR =
```

$$\frac{2 \sqrt{(2 \cos(q) - 120 \sigma_1 + 1)^2 + 1600 (2 \cos(q) + 1)^2} \left( 240 v_q^2 \sin\left(\frac{q}{2}\right)^5 - 480 v_q^2 \sin\left(\frac{q}{2}\right)^3 + 285 v_q^2 \sin\left(\frac{q}{2}\right) \right)}{14400 \sigma_1^2 - 480 \sigma_1 \cos(q) - 240 \sigma_1 + 6404 \cos(q)^2 + 6404 \cos(q) + 1601}$$

where

$$\sigma_1 = \cos\left(\frac{3q}{2}\right)$$

```
R_vect=simplify(sR*[-seno;coseno],50)
```

R\_vect =

$$\begin{pmatrix} 160 \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right) \sigma_2 \\ - \frac{\sigma_1}{\frac{160 \left( \cos(q) + \frac{1}{2} \right) \sigma_2}{\sigma_1}} \end{pmatrix}$$

where

$$\sigma_1 = 14400 \cos\left(\frac{3q}{2}\right)^2 - 480 \cos\left(\frac{3q}{2}\right) \cos(q) - 240 \cos\left(\frac{3q}{2}\right) + 6404 \cos(q)^2 + 6404 \cos(q) + 1601$$

$$\sigma_2 = 240 v_q^2 \sin\left(\frac{q}{2}\right)^5 - 480 v_q^2 \sin\left(\frac{q}{2}\right)^3 + 285 v_q^2 \sin\left(\frac{q}{2}\right) - 784 \sin\left(\frac{q}{2}\right)^2 + 588$$

## Euler-Lagrange 2nd kind

El sistema se mueve en (x(q),y(q)) , tiene 1GL.

**Euler-Lagrange 1st kind:** No se puede aplicar, porque no tenemos una expresión  $\psi(x, y) = 0$  para la ligadura cinemática.

```
T=simplify(0.5*M*(veloc(1)^2+veloc(2)^2))
```

T =

$$\frac{v_q^2 \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right)^2}{2} + \frac{v_q^2 \left( \cos(q) + \frac{1}{2} \right)^2}{2}$$

$$V=M \cdot g \cdot r \cdot (2)$$

$$V =$$

$$\frac{49q}{400} - \frac{49 \sin\left(\frac{3q}{2}\right)}{5} + \frac{49 \sin(q)}{200} + \frac{49}{5}$$

$$L=T-V; \text{ \%Lagrangiano}$$

Momentum (cantidad de movimiento) generalizado:

$$p=\text{diff}(L, v_q);$$

La ecuación Euler-Lagrange es:

$$\frac{d}{dt} \left( \frac{\partial L}{\partial v_q} \right) - \frac{\partial L}{\partial q} = \frac{dp}{dt} - \frac{\partial L}{\partial q} = Q$$

siendo  $Q$  la fuerza generalizada dada por el trabajo de las fuerzas externas cuando se desplace

virtualmente  $\delta q$ , esto es con un desplazamiento  $\delta x = \frac{\partial x}{\partial q} \delta q$ ,  $\delta y = \frac{\partial y}{\partial q} \delta q$  :

$$Q1=\text{simplify}(F_{\text{friccion\_vect}}(1)*\text{diff}(x)+F_{\text{friccion\_vect}}(2)*\text{diff}(y),50)$$

$$Q1 =$$

$$-\frac{129 v_q \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right)^2}{5000} - \frac{129 v_q \left( \cos(q) + \frac{1}{2} \right)^2}{5000}$$

El  $Q$  con la fórmula de Rayleigh también es correcto:

$$Q=-\text{diff}(0.5*b*(\text{veloc}'*\text{veloc}),v_q) \text{ \%Rayleigh dissipation function}$$

$$Q =$$

$$-\frac{129 v_q \left( \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80} \right)^2}{5000} - \frac{129 v_q \left( \cos(q) + \frac{1}{2} \right)^2}{5000}$$

Pasemos a las ecuaciones del movimiento Euler-Lagrange propiamente dichas:

$$\text{derivada\_de\_p}=\text{simplify}(\text{jacobian}(p,[q \ v_q])*[ \ v_q; \ a_q])$$

$$\text{derivada\_de\_p} =$$

$$a_q \left( \sigma_1^2 + \left( \cos(q) + \frac{1}{2} \right)^2 \right) + v_q \left( 2 v_q \left( \frac{9 \sin\left(\frac{3q}{2}\right)}{4} - \frac{\sin(q)}{40} \right) \sigma_1 - 2 v_q \sin(q) \left( \cos(q) + \frac{1}{2} \right) \right)$$

where

$$\sigma_1 = \frac{\cos(q)}{40} - \frac{3 \cos\left(\frac{3q}{2}\right)}{2} + \frac{1}{80}$$

```
acelq_EulerLagrange=simplify(solve(derivada_de_p-diff(L,q)==Q, a_q),550)
```

acelq\_EulerLagrange =

$$-\frac{129 v_q}{5000} - \frac{1568 \cos(q) - \sigma_3 (9600 v_q^2 \sigma_1 + 94080) + 6400 v_q (2 v_q \sigma_1 \sigma_2 - 2 v_q \sin(q) \sigma_4) - 6400 v_q^2 \sigma_1 \sigma_2 + (2 \cos(q) - 120 \sigma_3 + 1)^2 + 1600 (2 \cos(q) + 1)^2}{(2 \cos(q) - 120 \sigma_3 + 1)^2 + 1600 (2 \cos(q) + 1)^2}$$

where

$$\sigma_1 = \frac{9 \sin\left(\frac{3q}{2}\right)}{4} - \frac{\sin(q)}{40}$$

$$\sigma_2 = \frac{\cos(q)}{40} + \frac{1}{80}$$

$$\sigma_3 = \cos\left(\frac{3q}{2}\right)$$

$$\sigma_4 = \cos(q) + \frac{1}{2}$$

## Simulación temporal

El Euler-Lagrange (en posición y velocidad horizontal):

```
opts=odeset('AbsTol',1e-6,'RelTol',1e-5);
d2qdtEL=matlabFunction(acelq_EulerLagrange,'Vars',{q,v_q});
Tfin=20;
Xinicial=[-5.5;2.15];
[TiemposEL,EstadosEL]=ode45(@(t,estado) [estado(2);d2qdtEL(estado(1),estado(2))], [0 Tf], Xinicial);
plot(TiemposEL,EstadosEL(:,:)), grid on, legend("q(t)", "v_q(t)")
xlabel("Tiempo (s)")
%Si queremos simular las ecuaciones de Newton... da exactamente lo mismo,
%claro.
d2qdtNewton=matlabFunction(a_qNewton)
```

d2qdtNewton = *function\_handle with value:*

@(q,v\_q)-(v\_q.\*4.13058e+1-cos(q.\*(3.0./2.0)).\*9.408e+4+cos(q).\*1.568e+3-v\_q.\*cos(q.\*(3.0./2.0)).\*(7.74

```
[TiemposN,EstadosN]=ode45(@(t,estado) [estado(2);d2qdtNewton(estado(1),estado(2))], [0
hold on, plot(TiemposN,EstadosN(:,1)), grid on, hold off
```

