Modelling of carousel dynamics (2 DoF, Euler Lagrange): SIMULATION and ANIMATION

This code worked with Matlab **`R2023a`**

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

Presentations in video:

https://personales.upv.es/asala/YT/V/tiovELEN.html

https://personales.upv.es/asala/YT/V/tiovEL2EN.html

https://personales.upv.es/asala/YT/V/tiovELsimEN.html

Objectives: Understand the physical modeling of a carousel like the one in the figure, and its simulation. Specifically, we will address the case of 2 degrees of freedom with Euler-Lagrange equations of dynamics.

Table of Contents


```
syms phi theta real %degrees of freedom to describe motion
syms R_0 L M g I_0 real
assume(M>0), assume(g>0), assume(I_0>0), assume(L>0), assume(R_0>0)
%we help symbolic toolbox
```

Kinematics

```
R=R_0+L*sin(theta); %Distance from mass to central axis of rotation
x=R*cos(phi);
y=R*sin(phi);
z=L*(1-cos(theta)); %z=0 is downwards equilibrium point (when not
rotating)
r=[x;y;z]
```

r =

$$\begin{pmatrix} \cos(\phi)\ (R_0 + L\sin(\theta)) \\ \sin(\phi)\ (R_0 + L\sin(\theta)) \\ -L\ (\cos(\theta) - 1) \end{pmatrix}$$

```
q=[phi; theta];
syms omega_0 omega_1 real %angular velocities dot_phi dot_theta
syms alpha_0 alpha_1 real %angular accelerations ddot_phi ddot_theta
 v_q=[omega_0;omega_1]; a_q=[alpha_0;alpha_1];
```

$$v(q,\dot{q}) = \frac{dr}{dt} = \frac{\partial r}{\partial q}\cdot\frac{dq}{dt} = \frac{\partial r}{\partial q}\cdot\dot{q}$$

```
veloc = jacobian(r,q)*v_q %dr/dt
```

veloc =

$$\begin{pmatrix} L\,\omega_1\cos(\phi)\cos(\theta) - \omega_0\sin(\phi)\ (R_0 + L\sin(\theta)) \\ \omega_0\cos(\phi)\ (R_0 + L\sin(\theta)) + L\,\omega_1\cos(\theta)\sin(\phi) \\ L\,\omega_1\sin(\theta) \end{pmatrix}$$

If we wished to express the suspended mass acceleration as a function of position, velocity and angular accelerations, it would come out:

$$a(q,\dot{q},\ddot{q}) = \frac{d^2r}{dt^2} = \frac{dv}{dt} = \frac{\partial v}{\partial q}\cdot\dot{q} + \frac{\partial v}{\partial \dot{q}}\cdot\ddot{q} = \begin{pmatrix} \frac{\partial v}{\partial q} & \frac{\partial v}{\partial \dot{q}} \end{pmatrix}\cdot\begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix}$$

```
acel = simplify( jacobian(veloc,[q; v_q])*[v_q; a_q] ) %d2r/dt^2
```

acel =

$$\begin{pmatrix} L\,\alpha_1\cos(\phi)\cos(\theta) - \omega_0\ (\omega_0\cos(\phi)\,\sigma_1 + L\,\omega_1\cos(\theta)\sin(\phi)) - \alpha_0\sin(\phi)\,\sigma_1 - \omega_1\ (L\,\omega_0\cos(\theta)\sin(\phi) + L \\ \omega_1\ (L\,\omega_0\cos(\phi)\cos(\theta) - L\,\omega_1\sin(\phi)\sin(\theta)) - \omega_0\ (\omega_0\sin(\phi)\,\sigma_1 - L\,\omega_1\cos(\phi)\cos(\theta)) + \alpha_0\cos(\phi)\,\sigma_1 + L \\ L\ \left(\cos(\theta)\,{\omega_1}^2 + \alpha_1\sin(\theta)\right) \end{pmatrix}$$

where

$$\sigma_1 = R_0 + L\sin(\theta)$$

***Note:** this acceleration is NOT needed in the Euler-Lagrange methodology.

Energies and Euler-Lagrange equations of motion

```
T=simplify( 0.5*M*(veloc'*veloc) + 0.5*I_0*omega_0^2, 100) %kinetic energy
```

T =

$$\frac{M\,L^2\,\omega_0{}^2\,\sin(\theta)^2}{2}+\frac{M\,L^2\,\omega_1{}^2}{2}+M\,L\,R_0\,\omega_0{}^2\,\sin(\theta)+\frac{M\,R_0{}^2\,\omega_0{}^2}{2}+\frac{I_0\,\omega_0{}^2}{2}$$

```
V=M*g*z %potential energy
```

V = $-L\,M\,g\,(\cos(\theta)-1)$

```
Lagrangian=T-V;
```

Let us define $p(q,\dot{q})=\dfrac{\partial L}{\partial \dot{q}}$:

```
p = simplify( jacobian(Lagrangian,v_q) ) %momentum generalized
```

p = $\left(\omega_0\,\left(M\,L^2\,\sin(\theta)^2+2\,M\,L\,R_0\,\sin(\theta)+M\,R_0{}^2+I_0\right)\quad L^2\,M\,\omega_1\right)$

Euler-Lagrange equations of motion for each coordinate q_i are:

$$\frac{dp_i}{dt}-\frac{\partial L}{\partial q_i}=\tau_i$$

where, of course, we may need chain rule:

$$\frac{dp_i}{dt}=\frac{\partial p_i}{\partial q}\cdot\dot{q}+\frac{\partial p_i}{\partial \dot{q}}\cdot\ddot{q}=\begin{pmatrix}\frac{\partial p_i}{\partial q} & \frac{\partial p_i}{\partial \dot{q}}\end{pmatrix}\cdot\begin{pmatrix}\dot{q}\\ \ddot{q}\end{pmatrix}$$

```
dpdt = simplify(jacobian(p,[q;v_q])*[v_q;a_q]);
syms tau real %torque in carousel top
%we arrange in "column" form each equation of motion
EcsMotion = (dpdt - jacobian(Lagrangian,q)'==[tau;0]) %Euler-Lagrange
```

EcsMotion =

$$\begin{pmatrix}\alpha_0\,\left(M\,L^2\,\sin(\theta)^2+2\,M\,L\,R_0\,\sin(\theta)+M\,R_0{}^2+I_0\right)+L\,M\,\omega_0\,\omega_1\,\left(2\,R_0\cos(\theta)+2\,L\cos(\theta)\sin(\theta)\right)=\tau\\ -M\cos(\theta)\sin(\theta)\,L^2\,\omega_0{}^2+M\,\alpha_1\,L^2-M\,R_0\cos(\theta)\,L\,\omega_0{}^2+M\,g\sin(\theta)\,L=0\end{pmatrix}$$

We can rewrite as $M(q)\ddot{q}=\tau-C(q,\dot{q})\dot{q}+G(q)$

$$(I_0+M(R_0+L\sin\theta)^2)\cdot\alpha_0=\tau-2M(R_0+L\sin\theta)L\cos\theta\cdot\omega_0\omega_1 \text{ \%top platform}$$

$$M L^2 \cdot \alpha_1 = -MgL\sin\theta + M\omega_0^2(R_0 + L\sin\theta)L\cos\theta$$ %suspended mass

```
MassMatrix = simplify(jacobian(lhs(EcsMotion),a_q))
```

```
MassMatrix =
```

$$\begin{pmatrix} M\,L^2\sin(\theta)^2 + 2\,M\,L\,R_0\sin(\theta) + M\,{R_0}^2 + I_0 & 0 \\ 0 & L^2\,M \end{pmatrix}$$

Particular cases

Zero suspended length

```
simplify(subs(EcsMotion,L,0))
```

```
ans =
```

$$\begin{pmatrix} \alpha_0\,\left(M\,{R_0}^2 + I_0\right) = \tau \\ \text{symtrue} \end{pmatrix}$$

We have a 1 DoF model with equivalent moment of inertia $I_0 + MR_0^2$.

R0=0, I0=0, tau=0: spherical pendulum

```
simplify(subs(EcsMotion,{R_0,I_0,tau},{0,0,0}))
```

```
ans =
```

$$\begin{pmatrix} \frac{\theta}{\pi} \in \mathbb{Z} \vee \alpha_0\sin(\theta) + 2\,\omega_0\,\omega_1\cos(\theta) = 0 \\ L\,{\omega_0}^2\sin(2\,\theta) = 2\,L\,\alpha_1 + 2\,g\sin(\theta) \end{pmatrix}$$

Lateral oscillations with forced angular rotation speed of the platform

We would need to look just at the second equation of motion:

$$ML^2\alpha_1 = -MgL\sin\theta + M\omega_0^2(R_0 + L\sin\theta)L\cos\theta,$$

We have a torque caused by weight and by centrifugal force.

Fixing $\omega_0(t)$ to any desired trajectory, the first equation of motion, replacing $\alpha_0 = \frac{d\omega_0}{dt}$, would give the motor torque needed to keep the desired velocity profile.

Variations of the above idea are behind the so-called "computed torque" methodologies in robotics.

Normalised state-space equation

```
State=[phi; theta; omega_0; omega_1];
accels=solve(EcsMotion,a_q);
EcEstado4=simplify([omega_0;omega_1;accels.alpha_0;accels.alpha_1])
```

```
EcEstado4 =
```

$$\begin{pmatrix} \omega_0 \\ \omega_1 \\ -\dfrac{2\,M\,\omega_0\,\omega_1\cos(\theta)\sin(\theta)\,L^2 + 2\,M\,R_0\,\omega_0\,\omega_1\cos(\theta)\,L - \tau}{M\,L^2\sin(\theta)^2 + 2\,M\,L\,R_0\sin(\theta) + M\,{R_0}^2 + I_0} \\ \dfrac{R_0\,{\omega_0}^2\cos(\theta) - g\sin(\theta) + L\,{\omega_0}^2\cos(\theta)\sin(\theta)}{L} \end{pmatrix}$$