Rayleigh

Giacomo Boffi

```
from sympy import *
from sympy import init_printing
init_printing(use_latex=True)
from IPython.display import Math
def dm(latex_string): display(Math(latex_string))
def de(a, b): display(Eq(a, b))
```

```
x = symbols('x:4')

m, L, k, w, w02, Zo = symbols('m, L, k, omega, omega_0^2, Z_0')
w2 = w*w
display(x)
```

$$(x_0,\quad x_1,\quad x_2,\quad x_3)$$

To compute the total energy, we start with 0 energy.

Next we have a loop in which we have, in turn, $x_i = x_0,x_1,x_2$ and $x_j = x_1,x_2,x_3$ and formally increment the total energy with the contribution (translational and rotational) for the bar that is delimited by the hinges whose displacements are x_i and x_j.

Next we specify the constraints (the first and the last coordinates are equal to zero) and substitute in the expression of the kinetic energy, that must be multiplied by ω^2.

```
T = 0
for xi, xj in zip(x[:-1], x[1:]):
    T += m * ((xi+xj)/2)**2 + m*L**2/12 * ((xj-xi)/L)**2
T = T.expand().collect(m)
constraints = {x[0]:0, x[-1]:0}
T = T.subs(constraints)*w2/2
display(Eq(symbols('T'),T))
```

$$T = \frac{m\omega^2}{2}\left(\frac{2x_1^2}{3}+\frac{x_1x_2}{3}+\frac{2x_2^2}{3}\right)$$

A similar procedure is followed for the energy in the flexural springs and we add the contribution of the extensional one.

```
V = 0
for xi, xj, xk in zip(x[0:-2], x[1:-1], x[2:]):
    V += k*L**2 * ((xk-xj)/L - (xj-xi)/L)**2
V = V.expand().collect(k)
V = V.subs(constraints)/2
V += k * x[1]**2/2
display(Eq(symbols('V'),V))
```

$$V = \frac{kx_1^2}{2}+\frac{k}{2}\left(5x_1^2-8x_1x_2+5x_2^2\right)$$

The structural matrices can be derived by inspection of the expressions of the energies, but here we derive them by... derivation, using a fictitious 1x1 matrix and using twice the operation of partial derivation (the `jacobian` method) to have the individual coefficients

```
K = Matrix((V,)).jacobian(x[1:3]).jacobian(x[1:3])
M = Matrix((T/w2,)).jacobian(x[1:3]).jacobian(x[1:3])

dm('K=k'+latex(K/k))
dm('M=\\frac{m}6'+latex(6*M/m))
```

$$K = k\begin{bmatrix}6 & -4\\ -4 & 5\end{bmatrix}$$

$$M = \frac{m}{6}\begin{bmatrix}4 & 1\\ 1 & 4\end{bmatrix}$$

Next, we have a straightforward implementation of the Rayleigh quotient and of the refinement procedure.

```
x0 = Matrix((Zo,2*Zo))
v0 = x0*sqrt(w2)
a0 = -x0*w2

dm('x_0 = '+latex(x0)+',\qquad \dot x_0 = '+latex(v0)+',\qquad \ddot x_0 = '+latex(a0)+'.')

V2_0 = x0.T*K*x0
T2_0 = v0.T*M*v0

dm('2V_0='+latex(V2_0))
dm('2T_0='+latex(T2_0))

w2_00 = solve(T2_0-V2_0, w2)[w2]
display(Eq(w2, w2_00))
display(Eq(w2, w2_00.evalf().subs(k,m*w02)))
```

$$x_0 = \begin{bmatrix} Z_0 \\ 2Z_0 \end{bmatrix}, \qquad \dot{x}_0 = \begin{bmatrix} Z_0\sqrt{\omega^2} \\ 2Z_0\sqrt{\omega^2} \end{bmatrix}, \qquad \ddot{x}_0 = \begin{bmatrix} -Z_0\omega^2 \\ -2Z_0\omega^2 \end{bmatrix}.$$

$$2V_0 = \begin{bmatrix} 10Z_0^2 k \end{bmatrix}$$

$$2T_0 = \begin{bmatrix} 4Z_0^2 m\omega^2 \end{bmatrix}$$

$$\omega^2 = \frac{5k}{2m}$$

$$\omega^2 = 2.5\omega_0^2$$

```
f = -M*a0
x1 = K.inv()*f
V2_1 = x1.T*f

dm('f_0 = '+latex(f)+',\quad x_1 = K^{-1}f_0 ='+latex(x1))
dm('2 V_1 ='+latex(V2_1))
w2_01 = solve(T2_0-V2_1, w2)[1][w2]
display(Eq(w2, w2_01))
display(Eq(w2, w2_01.evalf().subs(k,m*w02)))
```

$$f_0 = \begin{bmatrix} Z_0 m\omega^2 \\ \frac{3Z_0}{2} m\omega^2 \end{bmatrix}, \quad x_1 = K^{-1} f_0 = \begin{bmatrix} \frac{11Z_0 m\omega^2}{14k} \\ \frac{13Z_0 m\omega^2}{14k} \end{bmatrix}$$

$$2V_1 = \begin{bmatrix} \frac{61Z_0^2 m^2\omega^4}{28k} \end{bmatrix}$$

$$\omega^2 = \frac{112k}{61m}$$

$$\omega^2 = 1.83606557377049\omega_0^2$$

```
v1 = x1*sqrt(w2)
T2_1 = v1.T*M*v1
dm('\dot x_1 = ' + latex(v1))
dm('2T_1=' + latex(T2_1))
w2_11 = solve(V2_1-T2_1, w2)[1][w2]

display(Eq(w2, w2_11))
display(Eq(w2, w2_11.evalf().subs(k, m*w02)))
```

$$\dot{x}_1 = \begin{bmatrix} \frac{11Z_0 m\omega^2}{14k}\sqrt{\omega^2} \\ \frac{13Z_0 m\omega^2}{14k}\sqrt{\omega^2} \end{bmatrix}$$

$$2T_1 = \begin{bmatrix} \frac{241Z_0^2 m^3\omega^6}{196k^2} 1 \end{bmatrix}$$

$$\omega^2 = \frac{427k}{241m}$$

$$\omega^2 = 1.77178423236515\omega_0^2$$