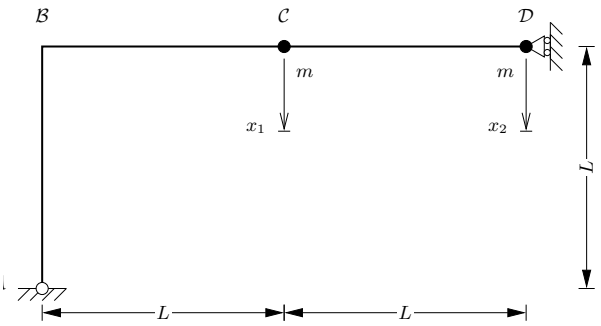
2DOF

Giacomo Boffi

0.1 2 DOF System



the dynamic system

The data of the problem

```
K = np.array(((24, -9),(-9,4)))*2/5
M = np.eye(2)
F = inv(K)
```

Solution of the eigenvalues problem, generation of relevant output

```
evals, evecs = eigh(K,M) ; evecs = -evecs
dl(dmat('\\boldsymbol M=m',M,','))
dl(dmat('\\boldsymbol K=\\frac{EJ}{L^3}\\frac25\,', K*5/2, ''))
dl(dmat('\\boldsymbol F=\\frac{L^3}{6EJ}\,', 6*F,'.'))
dl(r'''$$\Lambda = \frac52 \frac{\omega^2}{\omega_0^2}\rightarrow\omega^2=\frac25 \Lambda\omega_0^2$$
$$\Lambda^2-28\Lambda+15=0,\qquad\psi_{2i}=\frac{24-\Lambda_i}9\psi_{1i}$$
$$\Lambda_i = 14\mp\sqrt{181}\rightarrow$$
$$\psi_{11}=1,\psi_{21}=\frac{10+\sqrt{181}}9;\quad
  \psi_{12}=1,\psi_{22}=\frac{10-\sqrt{181}}9
$$''')
print('In terms of numerical values (eigenvalues are mass normalized)')
dl(dmat('\\boldsymbol\Omega^2 = \omega_0^2\,', np.diag(evals),','))
dl(dmat('\\boldsymbol\Psi =', evecs, '.'))
```

$$\boldsymbol{M} = m\begin{bmatrix}+1 & +0\\ +0 & +1\end{bmatrix}, \tag{1}$$

$$\boldsymbol{K} = \frac{EJ}{L^3}\frac{2}{5}\begin{bmatrix}+24 & -9\\ -9 & +4\end{bmatrix} \tag{2}$$

$$\boldsymbol{F} = \frac{L^3}{6EJ}\begin{bmatrix}+4 & +9\\ +9 & +24\end{bmatrix}. \tag{3}$$

$$\Lambda = \frac{5}{2}\frac{\omega^2}{\omega_0^2} \rightarrow \omega^2 = \frac{2}{5}\Lambda\omega_0^2$$

$$\Lambda^2 - 28\Lambda + 15 = 0, \qquad \psi_{2i} = \frac{24-\Lambda_i}{9}\psi_{1i}$$

$$\Lambda_i = 14 \mp \sqrt{181} \rightarrow$$

$$\psi_{11} = 1, \psi_{21} = \frac{10+\sqrt{181}}{9}; \quad \psi_{12} = 1, \psi_{22} = \frac{10-\sqrt{181}}{9}$$

In terms of numerical values (eigenvalues are mass normalized)

$$\boldsymbol{\Omega}^2 = \omega_0^2 \begin{bmatrix} +0.21855 & +0 \\ +0 & +10.9814 \end{bmatrix}, \tag{4}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} +0.358264 & +0.93362 \\ +0.93362 & -0.358264 \end{bmatrix}. \tag{5}$$

Determination of initial conditions In the beginning the structure is loaded in $\mathscr{D}$ by a(n unknown) load, say P, and we can write the static displacements in terms of a load vector and the flexibility matrix

$$\boldsymbol{x} = \boldsymbol{F}\,\boldsymbol{p} = \boldsymbol{F} \begin{Bmatrix} 0 \\ P \end{Bmatrix}$$

expanding the matrix product on the right we have two equations

$$x_1 = f_{12} P \tag{6}$$
$$x_2 = f_{22} P \tag{7}$$

but $x_2 = x_{2,0} = 1\,\text{mm}$ and we can solve the second equation for P and substitute in the first one

$$x_{2,0} = f_{22} P \rightarrow P = \frac{x_{2,0}}{f_{22}} \rightarrow x_{1,0} = x_{2,0}\,\frac{f_{12}}{f_{22}}$$

```
x20 = 1.0
x10 = x20 * F[0,1]/F[1,1]
x0 = np.array((x10,x20))
dl(dmat(r'\boldsymbol x_0 =', x0[:,None],r'\,\text{mm}.'))
q0 = inv(evecs)@x0
dl(dmat(r'\boldsymbol q_0 =', q0[:,None],r'\,\text{mm}.'))
```

$$\boldsymbol{x}_0 = \begin{bmatrix} +0.375 \\ +1 \end{bmatrix} \text{mm}. \tag{8}$$

$$\boldsymbol{q}_0 = \begin{bmatrix} +1.06797 \\ -0.00815611 \end{bmatrix} \text{mm}. \tag{9}$$

```
responses = homo(evecs, x0, evals)
dl('With $a = \omega_0 t$ and the coefficients of cosines in millimetres, the nodal responses are')
dl(responses)
dl('Note that $x_1(0)=0.375\,{}$mm and $x_2(0)=1\,{}$mm.')
```

```
The coefficients of nodal responses:
[[ 0.38261471 -0.00761471]
 [ 0.99707796  0.00292204]]
```

With $a = \omega_0 t$ and the coefficients of cosines in millimetres, the nodal responses are

$$x_1 = +0.383\cos 0.467\,a - 0.00761\cos 3.31\,a \tag{10}$$
$$x_2 = +0.997\cos 0.467\,a + 0.00292\cos 3.31\,a \tag{11}$$

Note that $x_1(0) = 0.375\,\text{mm}$ and $x_2(0) = 1\,\text{mm}$.

Initialization Let;s define a function to compute the homogeneous response due to an assigned initial displacement vector (no initial velocity).

The initial modal coordinates are computed using the inverse of the eigenvectors matrix, the matrix `coefs` contains in each row, for each *nodal* coordinate, the coefficients of the cosines at different frequencies.

```
def homo(evecs,x0,evals):
    from scipy.linalg import inv
    omegas = np.sqrt(evals)
    ivecs = inv(evecs)
    coefs = np.einsum('ij,jk,k->ij', evecs, ivecs, x0)
    print('The coefficients of nodal responses:')
    print(coefs)
    line = lambda i, c: "x_{%d}&="%(i+1)+''.join("%+.3g\\cos%.3g\\,a"%(k,w) for k,w in zip(c,omegas))
    lines = '\\\\'.join(line(i,c) for i, c in enumerate(coefs))
    lines = '\\begin{align}'+lines+'\\end{align}'
    return lines
```

Moreover, two simple utility functions to format and display matrices.

```
from IPython.display import Latex
def dmat(pre, mat, post, mattype='b'):
    s = r'\begin{align}' + pre + r'\begin{%smatrix}'%mattype
    s += r'\\'.join('&'.join('%+.6g'%val for val in row) for row in mat)
    s += r'\end{%smatrix}'%mattype + post + r'\end{align}'
    return s
def dl(ls):
    display(Latex(ls))
    return None
```