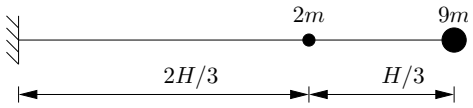
Rayleigh

Giacomo Boffi

1 Rayleigh Quotient Approximation

A tower structure is modeled by a straight, uniform beam clamped at the base, supporting two massive bodies.



The model of the tower structure

The beam is characterized by its length H, its negligible mass and its flexural stiffness E J, the supported bodies are characterized by their masses and their positions (see figure).

1. Using the shape function $\phi(z) = 1 - \cos(\pi z/2H)$ compute the Rayleigh Quotient estimate of the first eigenvalue of the structural model, R00.
2. Compute the first refinement and the second refinement of the Rayleigh Quotient estimate, R01 and R11.
3. Compare your results with the first eigenvalue of a 2 dof model.

1.1 Solution

We start definining the symbols we are going to use, the displacement function and its second spatial derivative.

```
z, H, EJ, m, Z0, w2 = symbols('z_H_EJ_m_Z_0_Omega')
v0 = Z0*(1 - cos(pi*z/2/H))
v2 = v0.diff(z, 2)
```

1.1.1 Problem data

The data of our problem is simply the position and the mass of the two supported bodies

```
z_1, M_1 = 3*H/3, 9*m
z_2, M_2 = 2*H/3, 2*m
```

1.1.2 Maximum strain and kinetic energies

Next, we compute the maximum values of the strain energy, taking into account only the flexural contributions, and of the kinetic energy (no contribution from the beam, only the two bodies are contributing)

```
V2 = EJ*(integrate(v2**2, (z, 0, H), manual=True))
T2 = w2*(M_1*v0.subs(z,z_1)**2 + M_2*v0.subs(z,z_2)**2)
dlatex('2V_\\text{max}=', latex(V2))
dlatex('2T_\\text{max}=', latex(T2))
```

$$2V_{\text{max}} = \frac{\pi^4 E J Z_0^2}{32 H^3}$$

$$2T_{\text{max}} = \frac{19\Omega}{2} Z_0^2 m$$

1.1.3 First approximation to ω^2

We solve for ω^2 the equation $T - V = 0$ to have the first approximation.

```
R00 = solve(T2-V2, w2)[0]
display(HTML('<h2>R_00</h2>'))
dlatex('\\omega^2=', latex(R00), '\\approx', latex(R00.evalf(6)))
```

```
<IPython.core.display.HTML object>
```

$$\omega^2 = \frac{\pi^4 EJ}{304 H^3 m} \approx \frac{0.320425 EJ}{H^3 m}$$

1.1.4 Inertial forces

We start from the accelerations as a function of z, then we compute the inertial forces for the two bodies

```
acc = -w2*v0
dlatex('\\ddot v(z) =', latex(acc))
p_1, p_2 = -M_1*acc.subs(z,z_1), -M_2*acc.subs(z,z_2)
dlatex('p_1 = ', latex(p_1), ',\\quad p_2 =', latex(p_2),'.')
```

$$\ddot{v}(z) = -\Omega Z_0\left(-\cos\left(\frac{\pi z}{2H}\right)+1\right)$$

$$p_1 = 9\Omega Z_0 m, \quad p_2 = \Omega Z_0 m.$$

1.1.5 Flexibility matrix

The flexibility matrix, that was computed separately, is

$$\boldsymbol{F} = \frac{1}{81}\frac{H^3}{EJ}\begin{bmatrix} 27 & 14 \\ 14 & 8 \end{bmatrix}$$

```
f0 = H**3/81/EJ
f_11, f_12 = 27*f0, 14*f0
f_21, f_22 = 14*f0,  8*f0
```

1.1.6 Displacements

Using the flexibility matrix we compute the displacements of the two bodies

```
v_1 = f_11*p_1 + f_12*p_2
v_2 = f_21*p_1 + f_22*p_2
dlatex('v_1 = ', latex(v_1), ',\\quad v_2 =', latex(v_2),'.')
```

$$v_1 = \frac{257\Omega Z_0 m}{81EJ}H^3, \quad v_2 = \frac{134\Omega Z_0 m}{81EJ}H^3.$$

1.1.7 New approximation to max strain energy

The maximum strain energy is one half of the work done by the inertial forces for the displacements of their point of application

```
V2 = v_1*p_1 + v_2*p_2
dlatex('2V_\\text{max} = ', latex(V2),'.')
```

$$2V_{\text{max}} = \frac{2447H^3\Omega^2 Z_0^2}{81EJ}m^2.$$

1.1.8 New approximation to ω^2

We have a new approximation to V_{max}, it is possible to compute a new approximation to ω^2

```
R01 = solve(T2-V2, w2)[1]
display(HTML('<h2>R_01</h2>'))
dlatex('\\omega^2 = ', latex(R01), '\\approx', latex(R01.evalf(9)))
```

```
<IPython.core.display.HTML object>
```

$$\omega^2 = \frac{1539EJ}{4894H^3m} \approx \frac{0.314466694EJ}{H^3m}$$

1.1.9 Velocities

```
vel_1, vel_2 = sqrt(w2)*v_1, sqrt(w2)*v_2
```

1.1.10 Maximum kinetic energy

```
T2 = M_1*vel_1**2 + M_2*vel_2**2
dlatex('2T_\\text{max} = ', latex(T2))
```

$$2T_{\text{max}} = \frac{630353 H^6 \Omega^3 Z_0^2}{6561 EJ^2} m^3$$

1.1.11 New approximation to ω^2

```
R11 = solve(T2-V2, w2)[1]
display(HTML('<h2>R_11</h2>'))
dlatex('\\omega^2 = ', latex(R11), '\\approx', latex(R11.evalf(9)))
```

```
<IPython.core.display.HTML object>
```

$$\omega^2 = \frac{198207 EJ}{630353 H^3 m} \approx \frac{0.314438101 EJ}{H^3 m}$$

1.2 Eigenvalues of the 2DOF model

```
from numpy import matrix
from scipy.linalg import eigh, inv
MassM = matrix(((9.0, 0.0), (0.0, 2.0)))
FlexM = matrix(((27.0, 14.0), (14.0, 8.0)))/81.0
StifM = inv(FlexM)
evals, evecs = eigh(StifM, MassM)
print("The first eigenvalue of the 2DOF model is ", evals[0], ".", sep='')
```

```
The first eigenvalue of the 2DOF model is 0.31443794459.
```

1.3 Initialization

```
from sympy import *
init_printing(use_latex=1)
from IPython.display import display, HTML, Latex
def dlatex(*strings):
    display(Latex('$$' + ' '.join(strings) + '$$'))
display(HTML(open('01.css').read()))
```

```
<IPython.core.display.HTML object>
```