Rayleigh Quotient

G. Boffi

1 Rayleigh Quotient

1.1 Characterization of the Problem

The physical constants involved with our problem. Because the lengths are reported in millimetres I have used exact fractions (the `frac(n,d)` used below meaning n/d) to *exactly* express the lengths in metres

```
# dimensions below are in metres
L,  B  = frac(2200, 1000), frac(1000, 1000)
H0, HL = frac( 280, 1000), frac( 200, 1000)

rho = 2500 # kg/m^3
E   = 16*10**9 # Pa
M   = 300 # kg
```

We need the height $H(\xi)$, the area $A(\xi)$ and eventually the flexural inertia $J(\xi)$ in terms of the constants previosly defined and of the position, either x or an adimensional coordinate $\xi = x/L$. Expression that contain `symbols` are, of course, symbolic equations...

```
x, xi = symbols('x_xi')

H = H0 + (HL-H0)*xi
A = H*B
J = H**3*B/12
```

1.2 The Shape Function ϕ_3

The shape function is defined in terms of the parameter a: $\phi_3 = \phi_3(\xi;a)$.

After this definition we compute its second derivative with respect to x applying the chain rule.

```
a = symbols('a')

phi3 = a*xi*xi + (1-a)*xi*xi*xi
phi3_2 = diff_x(phi3, xi, L, 2)

display(Eq(symbols('phi_3'),phi3))
```

$$\phi_3 = a\xi^2 + \xi^3(-a+1)$$

1.3 Maximum Energies, Approximate Eigenvalue

We write the maximum kinetic energy (remembering the lumped mass contribution), the maximum strain energy and we find our approximation solving for ω^2 the equation $T(\omega^2,a) = V(a)$

```
w2 = symbols('omega^2')

T_mx = w2*(integ_x(A*rho*phi3**2, xi, L)+300*(phi3.subs(xi,1))**2)/2
display(Eq(symbols('T'), T_mx))

V_mx = integ_x(E*J*phi3_2**2, xi, L)/2
display(Eq(symbols('V'), V_mx))

w2_app,  = solve((T_mx-V_mx),  w2)
display(Eq(w2, w2_app))
```

$$T = \frac{\omega^2}{2}\left(\frac{253a^2}{21} + \frac{1210a}{21} + 465\right)$$

$$V = \frac{3654400000a^2}{1331} - \frac{9728000000a}{1331} + \frac{10809600000}{1331}$$

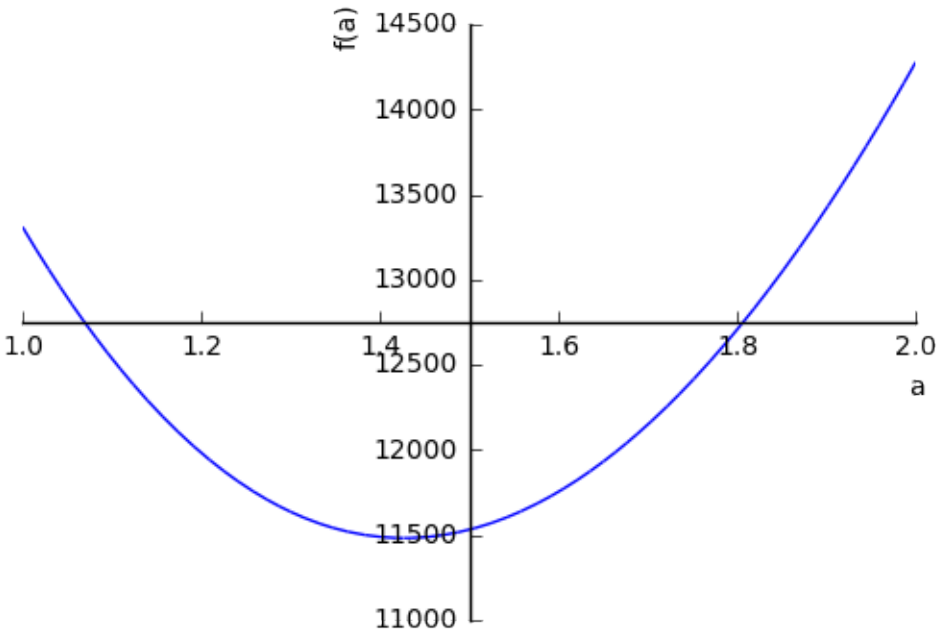
$$\omega^2 = \frac{153484800000a^2 - 408576000000a + 454003200000}{336743a^2 + 1610510a + 12997215}$$

1.3.1 Local Minimum Study

The approximation we have obtained is a function of a and we know that the *best* approximation is the lowest possible one. To have an idea of the minimum value we can plot the approximation against a — but, for which range of a?

For $a=3/2$ the bending moment ($M=EJ\phi_3''\propto EJ(x)(2a+6(1-a)x)$) is equal to zero in $x=L$, that is physically appealing, so I've chosen to plot $\omega^2_{\text{app}}(a)$ for an interval centered around 1.5, namely $1\le a\le 2$

```
sy.plot(w2_app, (a, 1, 2));
```



As you can see, $a=1.5$ was a good guess, but we can do better solving, with respect to a, the equation

$$\frac{\mathrm{d}\omega^2_{\text{app}}}{\mathrm{d}a}=0$$

(the numerator of the derivative is still quadratic in a, hence two roots).

```
a1, a2 = solve(w2_app.diff(a), a)
print("Roots:    a_1 = %+f,  a_2 = %+f;"%(a1.evalf(), a2.evalf()))
display(Eq(symbols('a_1'), a1))
display(Latex(r'$$\frac{\text{d}\,\omega^2_\text{app}}{\text{d}a}={}'+
              latex(w2_app.diff(a).simplify()) + r'$$'))
```

```
Roots:    a_1 = +1.427199,  a_2 = -11.001614;
```

$$a_1=-\frac{858083}{179245}+\frac{2\sqrt{310194062891}}{179245}$$

$$\frac{\mathrm{d}\omega^2_{\text{app}}}{\mathrm{d}a}=\frac{289086336000000a^2+2767832524800000a-4539088512000000}{85195979a^4+814918060a^3+8525307890a^2+31453260300a+126917804475}$$

To compute the requested value (the period of vibration) we have to substitute $a=a_1$ in ω^2_{app}, evaluate the resulting expression to a floating point number and eventually compute the period of vibtration.

```
w2_num = w2_app.subs(a, a1).evalf()
T_num = 2*pi/sqrt(w2_num)
print("%50s:%16.6f."%("Approximation_to_the_1st_eigenvalue", w2_num))
print("%50s:%16.6f."%("Approximation_to_the_1st_period_of_vibration", T_num))
```

```
                Approximation to the 1st eigenvalue:    11482.965526.
      Approximation to the 1st period of vibration:        0.058634.
```

1.4 Initialization Cells

First, we need help from external libraries (`sympy`, `IPython` and `math`), then we need to tell `sympy` that we want to format its output using `LaTeX`, eventually we inject in the notebook some `html` and `css` code to have a nicely presented notebook.

```
import sympy as sy
from sympy import Eq, Rational as frac, latex, solve, symbols
from math import pi, sqrt
from IPython.display import display, HTML, Latex
sy.init_printing(use_latex=1)
display(HTML(open('01.css').read()))
```

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

1.4.1 Helper Functions

We want to differentiate or integrate *with respect to x* some expressions that contain the adimensional coordinate $\xi = x/L$, so we define the following two auxiliary functions...

```
def diff_x(f, x, l, d=1): return f.subs(x, x/l).diff(x, d).subs(x, x*L)
def integ_x(f, x, l): return f.subs(x, x/l).integrate((x, 0, l)).subs(x, x*l)
```