04_Water_Tower

May 23, 2014

1 The water tower

The tower is in reinforced concrete,

```
g  = 2500.0
E  = 30E9
g0 = 9.80665 # standard acceleration of gravity
```

the heigth of the tower `H` and the lumped mass `M` at its top (representing the mass of the tank and of its content) are respectively

```
H = 45.0
M = 1.2E6
```

the cross-section is annular, the outer radius varies linearly from `R0` to `RH` and the thickness varies linearly from `T0` to `TH`

```
R0 = 3.20 ; RH = 2.40
T0 = 0.25 ; TH = 0.20
```

We can define the radius, the thickness, the cross section area and flexural inertia of the section as functions of the vertical coordinate `z`,

```
def R(z):
    return R0+(RH-R0)*z/H

def T(z):
    return T0+(TH-T0)*z/H

def A(z):
    t = T(z)
    return pi*(2*t*R(z)-t*t)

def J(z):
    r_ext = R(z)
    r_int = r_ext - T(z)
    return pi/4*(r_ext**4-r_int**4)
```

1.1 Total mass of the beam

We are ready to compute the total mass of the beam and to print aside of the lumped mass, so that we can judge the relevance of the distributed mass on our results.

To compute the beam mass, `mass`, we have to integrate $\gamma A(z)\,\mathrm{d}z$ over the length of the beam. To this purpose we use the library function `quad` that has 3 mandatory arguments:

1. a function, library or user defined, to be numerically integrated,
2. the lower limit of the interval of integration,
3. the upper limit of the interval of integration.

The integrand is here *defined on the spot* using the `lambda` syntax (in `matlab` it is `@` syntax).

Note that the function `quad` returns both the definite integral value and an estimate of the error, and that we discard the error estimate by assigning it to the dummy variable `_`.

```
mass, _ = quad(lambda z:A(z)*g, 0, H)
print M, mass
print mass/M
```

```
1200000.0 429710.970149
0.358092475124
```

The mass of the beam represent a significant fraction of the lumped mass, we can expect a small, but significant correction from the naive estimate $\omega^2 \approx EJ/3ML^3$

2 Choice of the shape function

I will simply use the one-minus-quarter-cosine shape function, no discussion, sorry. . .

$$\psi(z) = 1 - \cos\left(\frac{\pi}{2}\frac{z}{H}\right), \tag{1}$$

$$\psi'(z) = \frac{\pi}{2H}\sin\left(\frac{\pi}{2}\frac{z}{H}\right), \qquad \delta(z) = \int_0^z \psi'^2(s)ds = \frac{1}{8}\frac{\pi}{H}\left(\frac{\pi}{H}z - \sin\left(\frac{\pi}{H}z\right)\right), \tag{2}$$

$$\psi''(z) = \frac{\pi^2}{4H^2}\cos\left(\frac{\pi}{2}\frac{z}{H}\right). \tag{3}$$

2.0.1 Definitions

We define here the shape function and its derivatives in terms of Z, a constant proportional to the wave length.

While we're at it, $\int_0^z \psi'^2(s)\,\mathrm{d}s = \frac{\pi}{8H}\left(\frac{\pi}{H}z - \sin\left(\frac{\pi}{H}z\right)\right)$ is the increment of vertical displacement at quote z for a unit δZ_0.

```
Z = 2*H/pi

def psi0(z): return 1-cos(z/Z)
def psi1(z): return sin(z/Z)/Z
def psi2(z): return cos(z/Z)/Z/Z
def de_u(z):  return pi*(-sin(pi*z/H) + pi*z/H)/(8*H)
```

3 Generalised properties

We compute the generalized mass and stiffness, and also the contributions to the geometric stiffness due to the lumped mass and the distributed mass

```
mstar = quad(lambda z: psi0(z)**2*A(z)*g, 0, H)[0]
kstar = quad(lambda z: psi2(z)**2*E*J(z), 0, H)[0]
kgeoM = quad(lambda z: psi1(z)**2,        0, H)[0]*M*g0
kgeom = quad(lambda z: de_u(z)*A(z)*g,    0, H)[0]*g0
```

4 Results

Using a helper function, we display the results in terms of eigenvalues, frequencies, periods under different assumptions regarding the geometrical stiffness

```
def pr_res(s,w2):
    wn = sqrt(w2)
    fn = wn/2/pi
    Tn = 1.0/fn
    print "%36s%12.6f%8.5f%8.5f%8.5f" % (s, w2, wn, fn, Tn)
    return None
```

```
print
print "                  k*:", kstar,"N/m"
print "                  m*:", mstar,"kg"
print "                   M:     ", M,"kg"
print "             k_G*M*g0:", kgeoM,"N/m"
print "    g0 \int de_u dm: ", kgeom,"N/m"

print
print " "*39+"       w^2 "+"       w "+"       f "+"       T"
pr_res("k*/M ", kstar/M)
pr_res("k*/(M+m*)", kstar/(M+mstar))
pr_res("(k*-kg*M*g0)/(M+m*)", (kstar-kgeoM)/(M+mstar))
pr_res("(k*-kg M g0-\\int(kg dm)g0)/(M+m*)", (kstar-kgeoM-kgeom)/(M+mstar))
```

```
k*: 17347995.0148 N/m
                 m*: 81844.8215573 kg
                  M:     1200000.0 kg
           k_G*M*g0: 322625.853333 N/m
    g0 \int de_u dm:  29517.750381 N/m

                                              w^2       w       f       T
                               k*/M    14.456663 3.80219 0.60514 1.65252
                          k*/(M+m*)    13.533616 3.67881 0.58550 1.70794
                (k*-kg*M*g0)/(M+m*)    13.281927 3.64444 0.58003 1.72405
   (k*-kg M g0-\int(kg dm)g0)/(M+m*)   13.258899 3.64128 0.57953 1.72554
```

4.1 Comments

We have a large dependency of ω^2 on the account of distributed mass, a small but not negligible dependency on the geometrical stiffness associated with the weight of the lumped mass and a very small dependency on the geometrical stiffness associated with the distributed weight of the beam