Homework1

Giacomo Boffi

Initial Conditions

Data

```
m = 42.0; k = 32000.0; z = 0.02
x0 = 200/1000; v0 = -800/1000
P = 7000; a = 1.3
```

Particular Integral

The loading and the assumed particular integral, as well its derivatives

$$p(t) = P_0 \exp(-at),$$

$$\xi(t) = C\exp(-at), \quad \dot{\xi} = -aC\exp(-at), \quad \ddot{\xi} = a^2 C\exp(-at).$$

Substituting ξ into the EoM we can simplify and solve for C:

$$C(k - ac + {}^2 m)\exp(-at) = P\exp(-at) \rightarrow C = \frac{P}{k - ac + a^2 m}.$$

```
c = 2*z*np.sqrt(k*m)
C = P/(k-a*c+a*a*m)
```
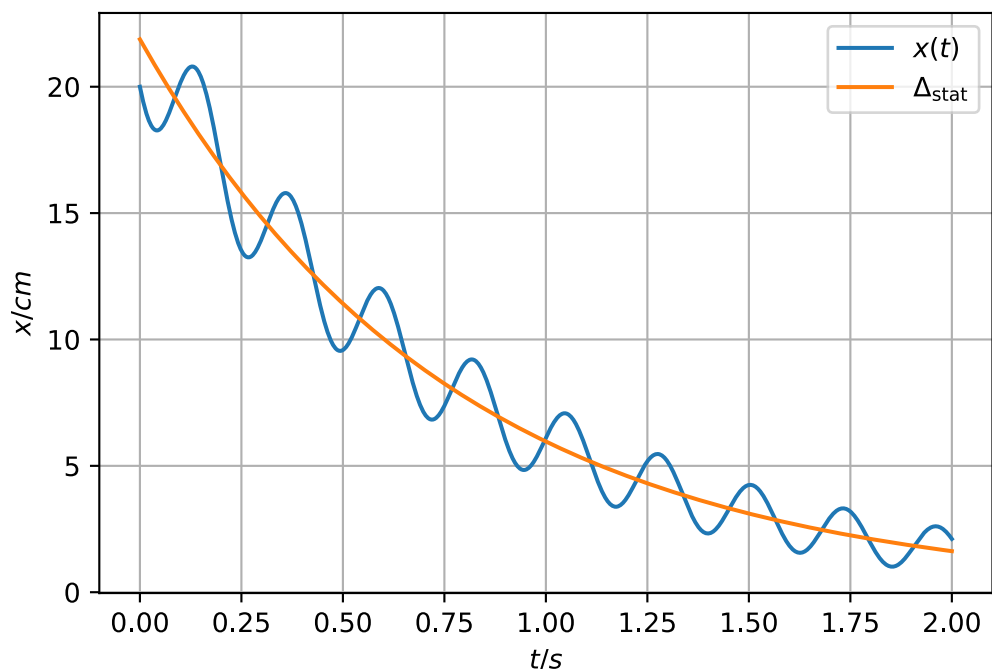
$C = 218.68$mm

General Integral

$$x = \exp(-\zeta\omega_n t)(A\cos(\omega_D t) + B\sin(\omega_D t)) + C\exp(-at),$$

$$x(0) = A + C, \quad \dot{x}(0) = \omega_D B - \zeta\omega_n A - aC.$$

```
wn2 = k/m
wn = np.sqrt(wn2)
wd = wn*np.sqrt(1-z*z)
# A+C=x0; wd B - z wn A -a C = v0
A = x0-C; B = (v0 + a*C + z*wn*A)/wd
```

$$x(t) = \exp(-0.552t)\big(-0.019\cos(27.597t) - 0.019\sin(27.597t)\big) - 0.019\exp(-1.300t)$$

$$\dot{x}(t) = \exp(-0.552t)\big(-0.516\cos(27.597t) + 0.526\sin(27.597t)\big) + 0.024\exp(-1.300t)$$

Rayleigh Quotient

We need some symbols (`sp` is an alias for the `sympy` module, aka library, that provides symbolic math to Python)

```
L, k, mu0, eta, x0, x1, omega = sp.symbols('L k mu_0 eta x_0 x_1 omega')
```

We write an expression for $\mu(\eta)$ and we integrate it to compute the total mass, $m = \int_0^L \mu(x)\,dx = \mu_0 L \int_0^1 \mu(\eta)d\eta$

```
mu = 2-eta
mass = sp.integrate(mu,(eta, 0, 1))
```

$$m = \frac{3}{2}\mu_0 L$$

We write the expression for $V_{\max}$ (that's easy) and the one for $T_{\max}$, here we start computing the integral $\int_0^L \mu(x)v^2(x)\,dx = L\int_0^1 \mu(\eta)(x_0 + (x_1 - x_0)\eta)^2\,d\eta$ to finally adjust our expression collecting terms and multiplying by ω^2

```
V = (2*x0**2+3*x1**2)*k/2
temp = sp.integrate(mu*(x0+eta*(x1-x0))**2,(eta,0,1)).expand()*L
T = omega**2*sp.collect(temp,mu0)/2
```

$$T_{\max} = V_{\max} \Rightarrow \frac{L\omega^2\left(\frac{7x_0^2}{12} + \frac{x_0 x_1}{2} + \frac{5x_1^2}{12}\right)}{2} = \frac{k(2x_0^2 + 3x_1^2)}{2}$$
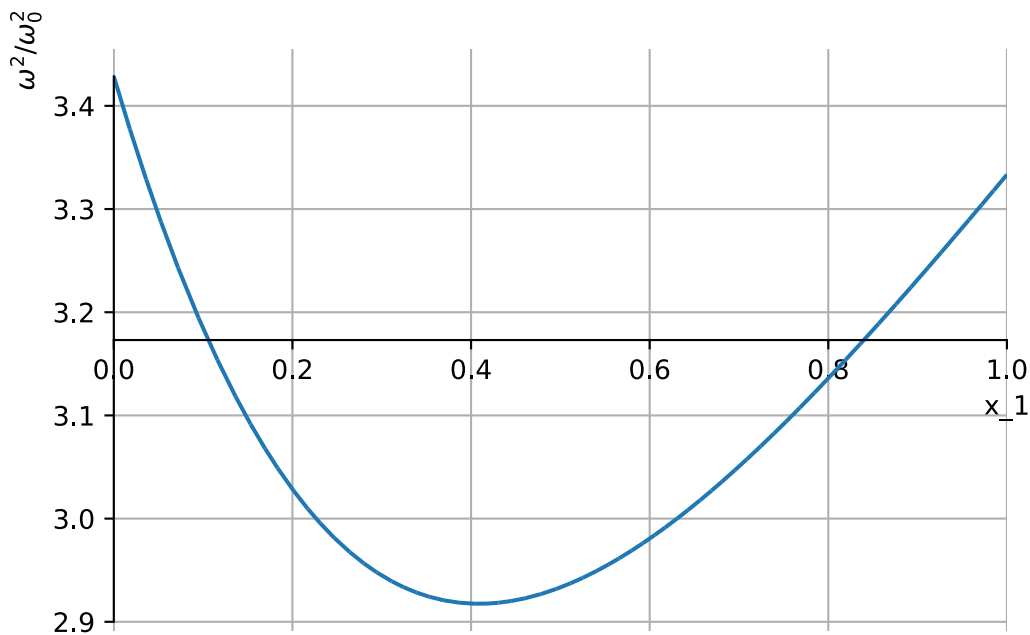
Solving the previous equation we have $\omega^2 = \omega^2(x_0,x_1)$,

2

```
w2 = sp.solve(sp.Eq(T,V),omega)[1]**2
```

$$\omega^2 = \frac{12k(2x_0^2 + 3x_1^2)}{L(7x_0^2 + 6x_0 x_1 + 5x_1^2)}$$

Our next step is to remove all dependencies except x_1 and finally plot our function of the single variable x_1

$$\left.\frac{\omega^2}{\omega_0^2}\right|_{x_0=1} = \frac{12(3x_1^2 + 2)}{5x_1^2 + 6x_1 + 7}$$

It's apparent that the minimum is near the point $x_1 = 0.4$ but also $x_1 = 0.5$ should give us a very good approximation...

$$\omega^2 = \frac{44}{15}\frac{k}{\mu_0 L} = 2.933333\frac{k}{\mu_0 L}, \quad \text{for } x_0 = 1, \, x_1 = \frac{1}{2}$$

$$\omega^2 = \frac{248}{85}\frac{k}{\mu_0 L} = 2.917647\frac{k}{\mu_0 L}, \quad \text{for } x_0 = 1, \, x_1 = 0.4$$

It's not difficult to find the location of the minimum (and hence the first eigenvector) and the corresponding eigenvalue. First we derive $\omega^2(x_1)$ with respect to x_1 and find the roots of the numerator

$$216x_1^2 + 264x_1 - 144 = 0$$

$$\hat{x}_1 = -\frac{11}{18} + \frac{\sqrt{337}}{18} = 0.408753$$

so we have an eigenvector, $\{1, \hat{x}_1\}$ and substituting $\hat{x}_1$ in $\omega(1, x_1)$ we have the corresponding eigenvalue

$$\omega^2(1, 0.408753) = 2.917486\omega_0^2$$

Structural Testing

We can estimate the stiffness and the damping ratio.

The stiffness is easy

3

```
P = 48*1000; x0 = 12/1000; x4 = 9.81/1000
k = P/x0
```

$k = 4000.0 \text{kN/m}$

and the damping ratio is easy too, mind that we don't need the period of vibration to apply our formula

$$\zeta_{n+1} = \frac{\log \frac{x_0}{x_4}}{4 \cdot 2\pi} \sqrt{1 - \zeta_n^2}.$$

Here we start with $\zeta_0 = 0$ and stop at ζ_2 because we have a meaningful value.

```
d = np.log(x0/x4); m = 4
z0 = 0
z1 = d/(2*m*np.pi)*np.sqrt(1-z0**2)
z2 = d/(2*m*np.pi)*np.sqrt(1-z1**2)
```

$\zeta_0 = 0\%$, $\zeta_1 = 0.801760\%$, $\zeta_2 = 0.801735\%$.

Just for fun, we can plot the decrement after m cycles, x_m/x_0, as a function of ζ. On the plot also a horizontal line for our $x_4/x_0 = 0.8175$, the intersection with the red curve gives the estimated value of the damping ratio.
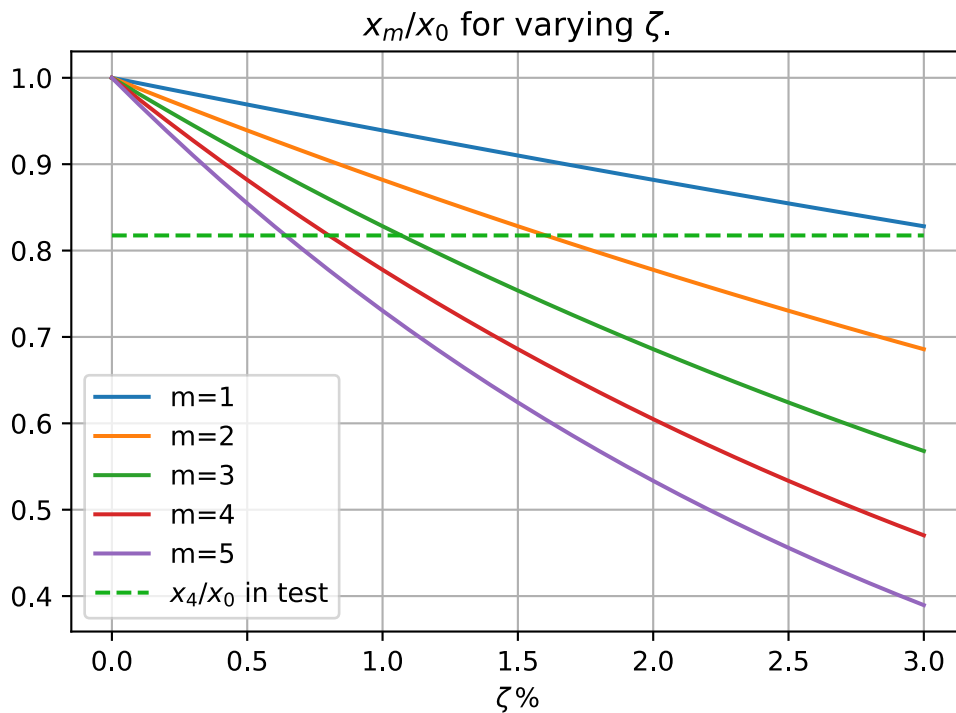
Vibration Isolation & Numerical Integration

Data, TR

```
m = 18E3; P0 = 1E3; t0 = 6; w0 = 2*np.pi*10; Pmax = 300
z00 = 0.00; z01 = 0.01; z12 = 0.12
```

4

```
TR = Pmax/P0
```

Plot the phase,

the angular velocity, the angular acceleration and the unbalanced load.

In the graph of the angular velocity I have put in evidence the natural frequency of vibration of an undamped suspension system (dashed line), a horizontal band of angular velocities around such frequency (in red) that are expected to excite the system in near-resonance and a vertical band (in blue) that highlights the time interval in which this near-resonance excitation is expected to take place.

The same vertical band is drawn on the plot of the unbalanced load and, especially, will be drawn on the plots of the transmitted force, to highlight the dynamic amplification of the response corresponding to a load that is just a fraction of the steady state load.

```
t = np.linspace(0,8,1001)
a = t/t0
phi0 = w0*t0*np.where(t<t0, a**2*(1-a/3), a-1/3)
phi1 = w0*np.where(t<t0, 2*a-a*a, 1)
phi2 = w0*np.where(t<t0, 2-2*a, 0)
P = P0*(phi1**2*np.sin(phi0)-phi2*np.cos(phi0))/w0**2
```
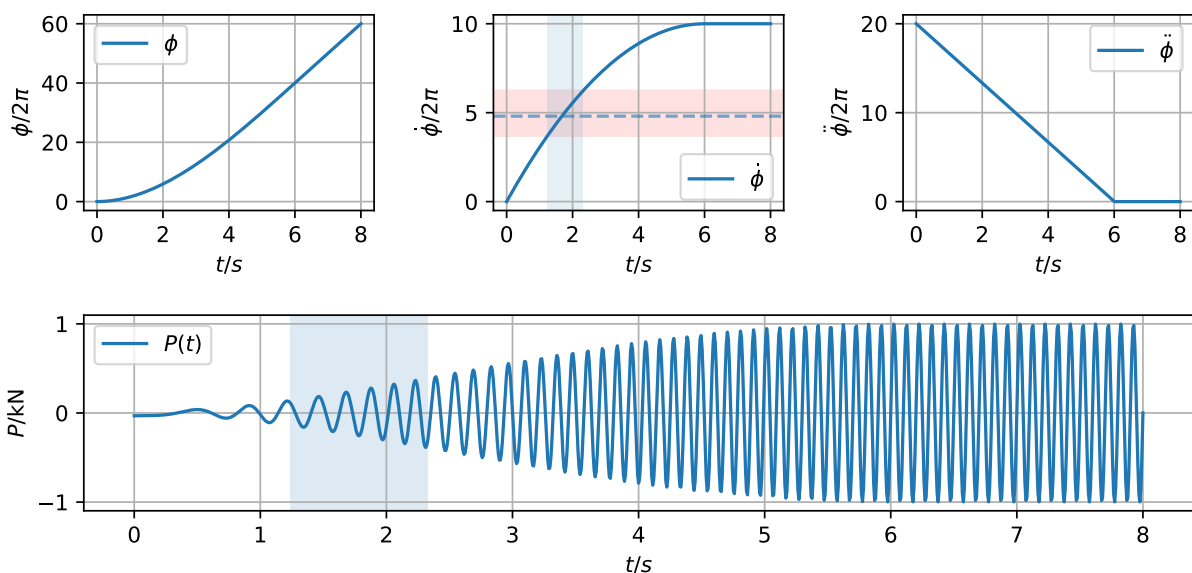

Design the Suspension System

for the two different values of the damping ratio ζ defined at the beginning of this section.

From

$$\frac{\sqrt{1+(2\zeta\beta)^2}}{\sqrt{(1-\beta^2)^2+(2\zeta\beta)^2}}=\text{TR}$$

we have

$$\text{TR}^2((1-\beta^2)^2+(2\zeta\beta)^2)-1-(2\zeta\beta)^2=0$$

that, expanded, is

$$\text{TR}^2\beta^4+((4\zeta^2-2)\text{TR}^2-4\zeta^2)\beta^2+\text{TR}^2-1=0.$$

Substituting TR = 0.3 and rearranging we have

$$9\beta^4 - (364z^2 + 18)\beta^2 - 91 = 0$$

whose positive root is

$$\beta^2 = \frac{364\zeta^2 + \sqrt{(364\zeta^2 + 18)^2 + 3276}}{18} + 1.$$

```
B201 = (364*z01**2 + np.sqrt((364*z01**2 + 18)**2 + 3276)) / 18 + 1
k01 = m * w0**2 / B201
c01 = 2*z01*np.sqrt(m*k01)

B212 = (364*z12**2 + np.sqrt((364*z12**2 + 18)**2 + 3276)) / 18 + 1
k12 = m * w0**2 / B212
c12 = 2*z12*np.sqrt(m*k12)
```

```
          mass [kg]   damper [kN·s/m]    spring [MN/m]
z=  0.0%   18000.000            0.000           16.399
z=  1.0%   18000.000           10.863           16.389
z= 12.0%   18000.000          124.896           15.045
```

Numerical Integration

A function factory

```
def new_integrator(m,c,k,h):
    c0 = 3*c + 6*m/h
    m0 = 3*m + c*h/2
    k0 = k + 3*c/h + 6*m/h/h
    def integrator(x0, v0, p0, p1):
        a0 = (p0-c*v0-k*x0)/m
        dp = p1 - p0 + a0*m0 + v0*c0
        dx = dp/k0
        dv = 3*(dx/h-v0)-a0*h/2
        return x0+dx, v0+dv
    return integrator
```

Compute the response for $\zeta = 1\%$

```
# Instantiate an integrator
next_xv = new_integrator(m, c01, k01, 8/1000)
# Set the initial conditions, x0 = 0, v0 = 0
R01 = [(0, 0)]
for p0, p1 in zip(P, P[1:]):
    R01.append(next_xv(*R01[-1], p0, p1))
```

Transmitted force $f = f(t)$ for $\zeta = 1.00\,\%$
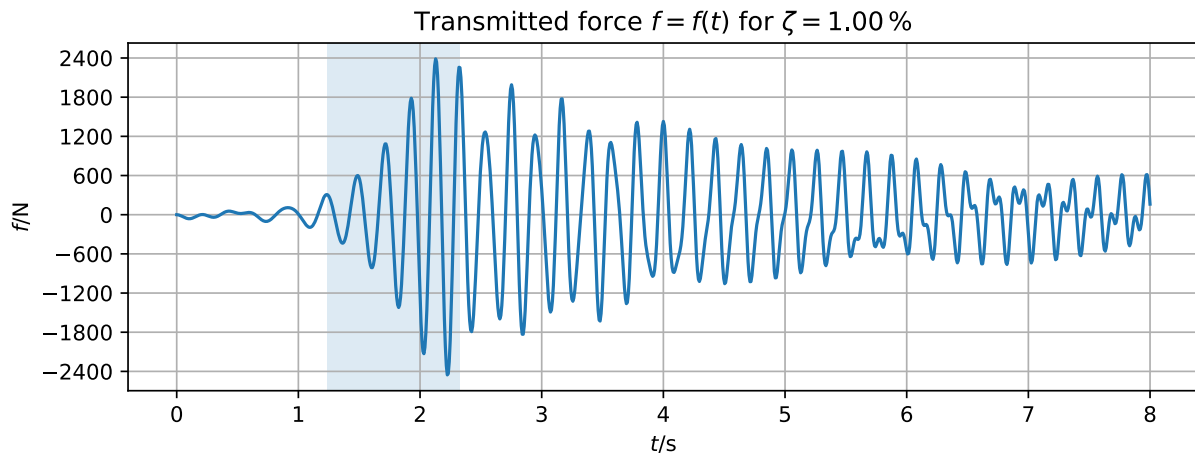
Notice how much the transmitted force has been amplified during the near-resonance interval, notice also that the response, well after this interval, has still a dominant frequency component corresponding to the natural frequency of vibration of the suspended mass, that superposed to the steady-state component of the response leads to a large value of the transmitted force.

Compute the response for $\zeta = 12\%$

```
next_xv = new_integrator(m, c12, k12, 8/1000)
R12 = [(0, 0)]
for p0, p1 in zip(P, P[1:]):
    R12.append(next_xv(*R12[-1], p0, p1))
```


Transmitted force $f = f(t)$ for $\zeta = 12.00\,\%$
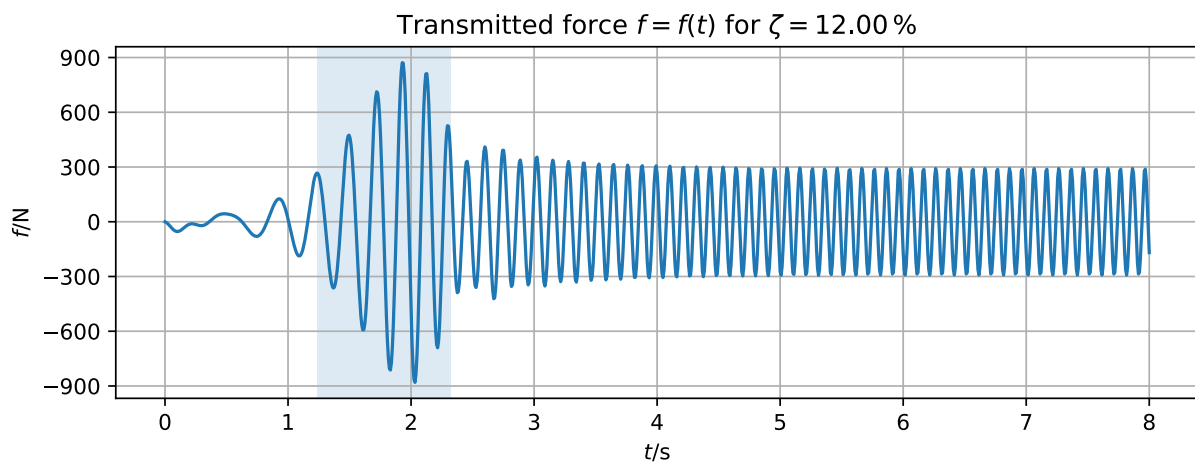
With a largish value of the damping ratio we have a controlled response during the near-resonant excitation interval and the transient response is quickly damped out, leading in a short time to a transmitted force that has exactly (numerical integration works, doesn't it?) the target amplitude of 300N.

Initialization Cells

```python
import numpy as np
import sympy as sp

import matplotlib.pyplot as plt
from matplotlib import rc_context

from IPython.display import Math, Latex

sp.init_printing(use_latex=True)
```

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