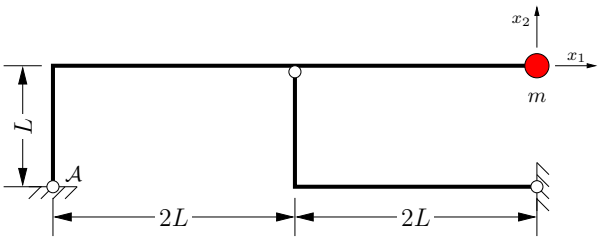
Homework no.3

the solutions

Giacomo B. Boffi

Imposed Displacements



The dynamic system in figure is composed of two uniform beams of negligible mass, their flexural stiffness being $EJ=\text{const}$, supporting a lumped mass of negligible rotatory inertia (the flexibility matrix is $\boldsymbol{F}=\frac{L^3}{6EJ}\begin{bmatrix}3 & 2\\ 2 & 96\end{bmatrix}$).

The left support is subjected to an imposed horizontal displacement that varies between 0 and Δ

$$u_A=\Delta\begin{cases}\frac{20\tau^3-15\tau^4+3\tau^5}{16} & \text{for } 0\le\tau\le 2,\\ 1 & \text{for } \tau>2,\end{cases}$$

where $\tau=\omega_0 t$ and $\omega_0^2=\frac{EJ}{mL^3}$.

Plot the *total* vertical displacement of the mass in the time interval $0\le\tau\le 10$.

Solution

We are going to

- determine the structural matrices for the 2 DoF dynamic system,
- compute its eigenvalues and its eigenvectors,
- determine the pseudo-static mass motion when the support is displaced
- write the EoM for the inertial forces due to the support motion,
- determine the particular integrals and the homogeneous solution first, and next the frre vibrations,
- present our results.

Structural Matrices

Because the text of the problem gives us $\boldsymbol{F}$, we can easily compute $\boldsymbol{K}$ by inversion of the flexibility; the mass matrix is super easy... $\boldsymbol{M}=m\boldsymbol{I}$

```
F = array(((3., 2.), (2., 96.)))/6
K = array(((96., -2.), (-2., 3)))*6/(96*3-2*2)
M = array(((1., 0.), (0., 1.)))
```

Eigenvalues, eigenvectors, frequencies

Next we compute the eigenvalues and the eigenvectors

```
l2, Psi = eigh(K, M)
l1 = sqrt(l2)
L2 = diag(l2) ; L1 = sqrt(L2)
```

The structural matrices, the eigen* and the frequencies

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

$$\boldsymbol{F} = \frac{L^3}{6EJ}\begin{bmatrix} +3 & +2 \\ +2 & +96 \end{bmatrix}, \qquad \boldsymbol{K} = \frac{3EJ}{142L^3}\begin{bmatrix} +96 & -2 \\ -2 & +3 \end{bmatrix},$$

$$\boldsymbol{\Lambda}^2 = \begin{bmatrix} +0.062472 & +0 \\ +0 & +2.02908 \end{bmatrix}, \qquad \boldsymbol{\Lambda} = \begin{bmatrix} +0.249944 & +0 \\ +0 & +1.42446 \end{bmatrix},$$

$$\boldsymbol{\Psi} = \begin{bmatrix} +0.0214905 & -0.999769 \\ +0.999769 & +0.0214905 \end{bmatrix}.$$

Kinematics of the Problem

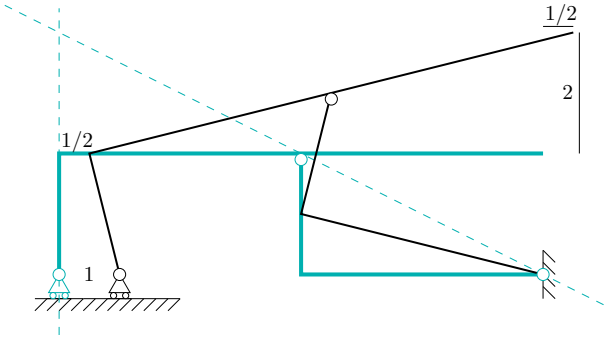
Starting from the imposed displacement of the left node we compute also its velocity and its acceleration, next we analyze the effects of a static displacement on the position of the supported mass.

The node displacement and its derivatives We use a *polynomial class* to represent the displacement of $\mathcal{A}$ (the coefficients must be given in descending order). The class supports evaluation and derivation (and other things that we are not going to use).

```
d = p(3, -15, 20, 0, 0, 0)/16
v = d.deriv()
a = v.deriv()
```

$$\begin{aligned} x_{\mathcal{A}} &= 0.1875\tau^5 - 0.9375\tau^4 + 1.25\tau^3, \\ \frac{dx_{\mathcal{A}}}{d\tau} = x'_{\mathcal{A}} &= 0.9375\tau^4 - 3.75\tau^3 + 3.75\tau^2, \\ \frac{d^2x_{\mathcal{A}}}{d\tau^2} = x''_{\mathcal{A}} &= 3.75\tau^3 - 11.25\tau^2 + 7.5\tau = \frac{15}{4}\left(\tau^3 - 3\tau^2 + 2\tau\right). \end{aligned}$$

The influence matrix The displacement of the supported mass is analyzed degrading the $\mathcal{A}$ hinge to a roller: the structure is hence a rigid system with 1 DoF and the CIR of the left-top bar is on the vertical of the roller and on the line joining the two hinges.

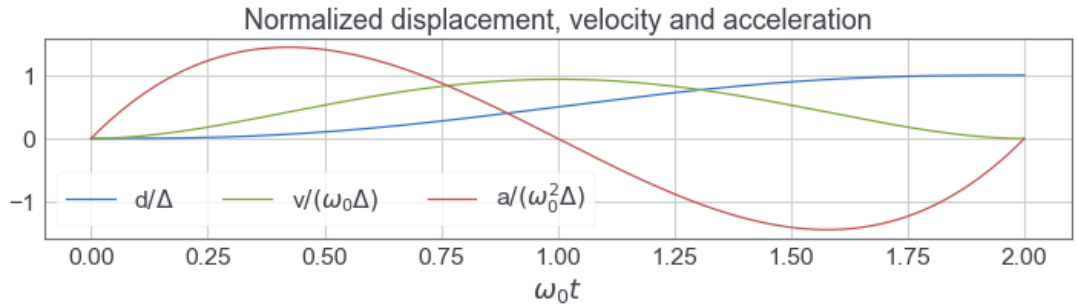


For $x_{\mathcal{A}} = 1$ we have $x_1 = 1/2$ and $x_2 = 2$, hence $E = \{1/2 \quad 2\}^T$. The modal load is $\boldsymbol{p}^\star = -\boldsymbol{\Psi}^T \boldsymbol{M} \boldsymbol{E} \ddot{x}_{\mathcal{A}} = -m \boldsymbol{\Psi}^T \boldsymbol{E} \ddot{x}_{\mathcal{A}} = m \boldsymbol{\Gamma} \ddot{x}_{\mathcal{A}}$ where $\boldsymbol{\Gamma} = -\boldsymbol{\Psi}^T \boldsymbol{E}$.

```
E = array((0.5, 2.0))
G = -Psi.T@E
```

$$\boldsymbol{E} = \begin{Bmatrix} +0.5 \\ +2 \end{Bmatrix}, \qquad \boldsymbol{\Gamma} = \begin{Bmatrix} -2.01028 \\ +0.456904 \end{Bmatrix}.$$

Plots of displacement, velocity and acceleration of the node ... against a dimensionless time that runs from `0` to `2` (note the normalization factors, especially the ones for velocity and acceleration).



Dynamic Response

1. We write the EoM with reference to the dimensionless time $\tau = \omega_0 t$,
2. we find the particular integral that satisfies the EoM when the mass is subjected to the *static acceleration*,
3. the forced modal responses are determined imposing the initial conditions for $\tau = 0$ and
4. eventually we determine the free response, this time imposing that the displacement and the velocity of the forced and the free responses are equal for $\tau = 2$.

The modal EoM with respect to dimensionless time The equation of motion is $\boldsymbol{M}\ddot{\boldsymbol{x}}+\boldsymbol{K}\boldsymbol{x}=-\boldsymbol{M}\boldsymbol{E}\ddot{x}_{\mathcal{A}}$.

If we denote with primes the derivative with respect to $\tau=\omega_0 t$, it is $\dot{f}=\omega_0 f'$ and the equation of motion is

$$\omega_0^2\boldsymbol{M}\boldsymbol{x}''+\boldsymbol{K}\boldsymbol{x}=-\omega_0^2\boldsymbol{M}\boldsymbol{E}x''_{\mathcal{A}}.$$

Applying the modal transformation, with $\boldsymbol{M}^\star=\boldsymbol{M}=m\boldsymbol{I}$ (that implies that $\boldsymbol{M}\boldsymbol{E}=m\boldsymbol{E}$)

$$m\omega_0^2\boldsymbol{q}''+m\omega_0^2\boldsymbol{\Lambda}^2\boldsymbol{q}=-m\omega_0^2\boldsymbol{\Psi}^T\boldsymbol{E}x''_{\mathcal{A}}$$

and, simplifying $m\omega_0^2$ and developing the right member

$$\boldsymbol{q}''+\boldsymbol{\Lambda}^2\boldsymbol{q}=-\boldsymbol{\Psi}^T\boldsymbol{E}x''_{\mathcal{A}}=\frac{15}{4}\boldsymbol{\Gamma}(\tau^3-3\tau^2+2\tau),\qquad \text{where } \boldsymbol{\Gamma}=-\boldsymbol{\Psi}^T\boldsymbol{E}.$$

Particular Integral A particular integral (omitting the indices) is $\xi(\tau)=P\tau^3+Q\tau^2+R\tau+S$. With the provision that we have to multiply our results by $15\boldsymbol{\Gamma}/4$ we can write

$$6P\tau+2Q+\lambda^2\left(P\tau^3+Q\tau^2+R\tau+S\right)=\tau^3-3\tau^2+2\tau.$$

Equating for each power of τ the coefficients on both sides, we have

$$\lambda^2P=1,\qquad \lambda^2Q=-3,\qquad 6P+\lambda^2R=2,\qquad 2Q+\lambda^2S=0$$

a set of equations that can be readily translated to code (taking also into account the multiplication by $15\gamma_i/4$).

```
P = 1/l2 ; Q = -3/l2 ; R = (2-6*P)/l2 ; S = (0-2*Q)/l2
P, Q, R, S = map(lambda coeff: 3.75*coeff*G, (P, Q, R, S))
xi = [p(P[i], Q[i], R[i], S[i]) for i in (0, 1)]
```

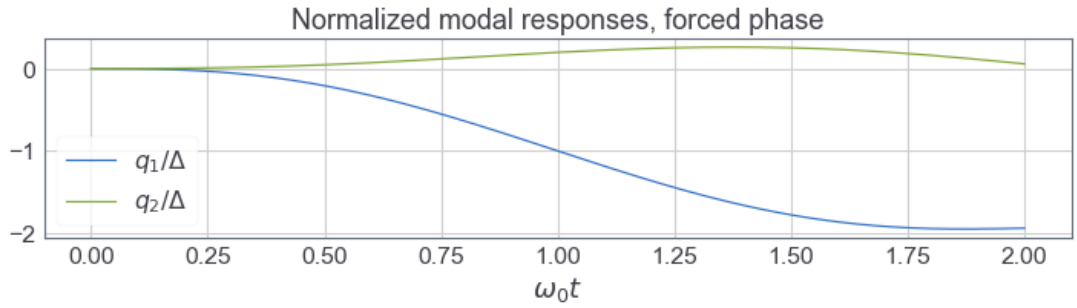
$$\xi_1=-120.671\tau^3+362.013\tau^2+11348.3\tau-11589.6$$
$$\xi_2=0.844418\tau^3-2.53325\tau^2-0.808115\tau+2.49695$$

Modal Responses, Forced Phase It is $q_i(\tau)=A_i\cos\lambda_i\tau+B_i\sin\lambda_i\tau+\xi_i(0)$; the ξ have been determined, we have to determine the constants of integration A and B imposing that $\boldsymbol{q}(0)=0$ and $\boldsymbol{q}'(0)=0$: it is $A_i=-\xi_i(0)$ and $\lambda_i B_i=-\xi_i'(0)$.

```
xi0 = array([Xi(0) for Xi in xi])          ; A = -xi0
dxi0 = array([Xi.deriv()(0) for Xi in xi]) ; B = -dxi0/l1
```

Plot of the forced modal responses We compute first the 2D array `l1t1`, that has $\lambda_i\omega_0 t$ in its two columns, next the modal response `q02` because it's the first phase of the response for $0\le\tau\le2$...

```
l1t02 = outer(t02, l1)
q02 = A*cos(l1t02) + B*sin(l1t02) + array([Xi(t02) for Xi in xi]).T
dq02 = -A*l1*sin(l1t02) + B*l1*cos(l1t02) + array([Xi.deriv()(t02) for Xi in xi]).T
```



Modal Responses, free phase The free response starts at `t22=2`, we can impose that the modal displacements and derivatives of the free response, computed at `t22`, are equal to the modal displacements and derivatives for the forced phase. Indicating with a ^{1} the quantities relative to the forced phase and with ^{2} the ones relative to free response,

$$\begin{bmatrix} \cos\lambda_i\tau_{22} & \sin\lambda_i\tau_{22} \\ -\sin\lambda_i\tau_{22} & \cos\lambda_i\tau_{22} \end{bmatrix} \begin{Bmatrix} {}^2A_i \\ {}^2B_i \end{Bmatrix} = \begin{Bmatrix} {}^1q_i(\tau_{22}) \\ \frac{{}^1\dot{q}_i(\tau_{22})}{\lambda_i} \end{Bmatrix}.$$

The coefficient matrix is an orthogonal matrix (i.e., $\boldsymbol{A}^{-1} \equiv \boldsymbol{A}^T$), so we can write our solution as follows

```
t22 = 2
C22 = cos(l1*t22)
S22 = sin(l1*t22)
q22 = q1[-1]
dq22_over_l1 = dq1[-1]/l1
Afree = C22*q22-S22*dq22_over_l1
Bfree = S22*q22+C22*dq22_over_l1
```

It is now possible to compute the modal response in $2 \leq \tau \leq 10$, so that we can compute the total displacements of the mass in said interval, as requested by the problem.

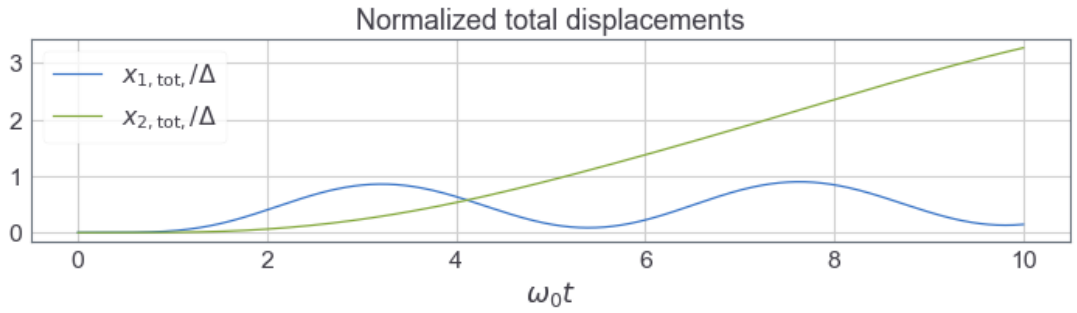
```
t210 = linspace(2.01, 10, 800)
l1t210 = outer(t210, l1)
q210 = Afree*cos(l1t210) + Bfree*sin(l1t210)
dq210 = -l1*Afree*sin(l1t210) + l1*Bfree*cos(l1t210)
```

Summing Up and Showing the Results

Finally we stick together the forced and the free responses, and compute the nodal deformations by post-multiplying the modal responses by the transpose of `Psi` and the static displacement by evaluating the polynomial `d` and taking the outer product with the influence matrix, finally we compute the total displacement of the mass

```
t = hstack((t02, t210))
q = vstack((q02, q210))
dq = vstack((dq02, dq210))
x = q@Psi.T
xst = outer(where(t<t22,d(t), 1), E)
xtot = xst + x
```

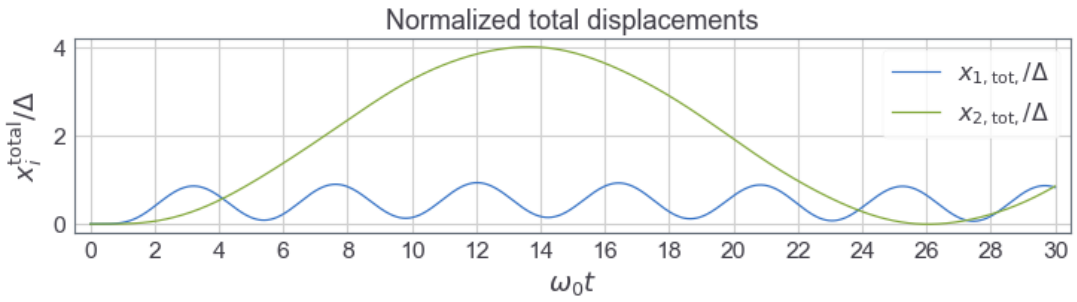
The Plot of the Total Displacements ... normalized with respect to the final support of the displacement.



The Total Displacements over a Longer Time Span The requirements of the problem are satisfied, but I'm left with the curiosity of what happens to the vertical displacements in the following... let's plot the total displacements for $0 \leq \tau \leq 30$.

```
t230 = linspace(2.01, 30, 2800)
l1t230 = outer(t230, l1)
q230 = Afree*cos(l1t230) + Bfree*sin(l1t230)
dq230 = -l1*Afree*sin(l1t230) + l1*Bfree*cos(l1t230)

t = hstack((t02, t230))
q = vstack((q02, q230))
dq = vstack((dq02, dq230))
x = q@Psi.T
xst = outer(where(t<t22,d(t), 1), E)
xtot = xst + x
```



Quite a sling effect...

Post Scriptum

That's all, except that here I show you the code that is executed at the beginning of the notebook.

There is the initialization of the plotting machinery and the imports, a further initialization of the plotting *style* and eventually the definition of a number of utility functions.

If you were interested (who knows...) in the gory details of the production of the mathematical displays and the plots, you can examine the notebook (code+text) that is the source of this paper.

Initialization, Imports

```
%matplotlib inline

import numpy as np
from numpy import array, cos, diag, hstack, linspace
from numpy import outer, sin, sqrt, vstack, where
import matplotlib.pyplot as plt
from scipy.linalg import eigh
```

```
from jupyterthemes import jtplot
jtplot.style(context='paper', fscale=1.5, figsize=(15, 3))
```

Function Definitions

An helper funtion for the decoration of plots The Matplotlib API is a little verbose, this is an attempt to squeeze the typical decoration that is applied to a plot in a single statement

```
def plt_labels(xl='', yl='', xt='', yt='', t='', l=1, c=1):
    if xl: plt.xlabel(xl)
    if yl: plt.ylabel(yl)
    if xt: plt.xticks(xt)
    if yt: plt.yticks(yt)
    if t:  plt.title(t)
    if l:  plt.legend(ncol=c)
```

Polynomials' stuff A function to instantiate a library provided polynomial class and a formatter function to prepare the mathematical display of a polynomial.

```
def p(*coefs):
    from numpy import poly1d
    return poly1d(coefs)

def ltx_p(p, leading_sign=False):
    order = p.order
    coeffs = p.coefficients
    def _fmt(i, c):
        coef = ('%+g' if (i or leading_sign) else '%g') % c
        if c == 1 : coef = '+' if (i or leading_sign) else ''
        if c == -1: coef = '-'
        var  = '{\\tau}' if i < order else ''
        expn = '^{%d}'%(order-i) if i < order-1 else ''
        return coef + ('\\,' if var else '') + var + expn
    return ''.join(_fmt(i, c) for i, c in enumerate(coeffs) if c)
```

Other math stuff A formatter to prepare the math display of a matrix and the function that is actually used to render the math displays.

```
def prmat(mat, fmt='%+.6g', type='b'):
    return (r'\begin{%smatrix}' +
            r'\\'.join('&'.join(fmt%num for num in row) for row in mat) +
            r'\end{%smatrix}')%(type, type)

def dL(*pieces):
    from IPython.display import Latex
    display(Latex(' '.join(pieces)))
```

Free Vibrations of an Uniform Beam

A uniform beam, its length L, its stiffness $EJ = \text{const}$ and its unit mass $\bar{m} = \text{const}$ is clamped at $x = 0$ and is simply supported at $x = L$.

The beam is in a condition of static equilibrium (see figure) under a couple $W = w\frac{EJ}{L}$ applied at $x = L$ when, at time $t = 0$, the external load is suddenly released.

Determine the modal responses for the first 3 modes and plot the bending moment $M_b(0,t)$ in the interval $0 \leq \omega_0 t \leq 5$, where $\omega_0^2 = \frac{EJ}{\bar{m}L^4}$

Solution

We are going to

- recall the general form of the eigenfunctions of an uniform beam, solutions of the indefinite eq. of equilibrium,
- determine the eigenfunctions of the actual problem applying the boundary conditions,
  - determine the wavenumbers,
  - determine the frequencies of free vibration,
- determine the static equilibrium position of the beam (i.e., the initial conditions) using the moment-curvature relationship and
  - determine the coefficients of the eigenfunctions expansion of said initial condition,
- determine the modal responses (free vibrations following given initial conditions),
- write the bending moment as a sum of second derivatives of the modal responses,
  - specialize for $t = 0$ and discuss the problems arising from our particular boundary conditions,
  - specialize for $x = 0$ and plot for $0 \leq \omega_0 t \leq 10$ and discuss the role of the high frequency components with respect to the effects of damping.

The eigenfunctions

The eigenfunctions of a uniform beam have the form

$$\phi_n = A_n \sin\beta_n x + B_n \cos\beta_n x + C_n \sinh\beta_n x + D_n \cosh\beta_n x,$$

where the *wave number* β is associated to the relevant frequency of vibration by the relationship

$$\beta_n^4 = \frac{\bar{m}\omega_n^2}{EJ} \qquad \text{or, multiplying by L}^4, \qquad (\beta_n L)^4 = \omega_n^2 \frac{\bar{m}L^4}{EJ} = \frac{\omega_n^2}{\omega_0^2}.$$

The boundary conditions, determination of wavenumbers and frequencies The boundary conditions for our problem are

$$\begin{aligned} \phi_n(0) &= 0 & \phi_n'(0) &= 0 \\ \phi_n(L) &= 0 & -EJ\phi_n''(L) &= 0 \end{aligned}$$

From the first two equations we find that $B_n + D_n = 0$ and $A_n + C_n = 0$, solving for C_n and D_n and substituting in the general integral we have

$$\begin{aligned} \phi_n &= A_n(\sin\beta_n x - \sinh\beta_n x) + B_n(\cos\beta_n x - \cosh\beta_n x) \\ \phi_n'' &= -A_n\lambda_n^2(\sin\beta_n x + \sinh\beta_n x) - B_n\lambda_n^2(\cos\beta_n x + \cosh\beta_n x) \end{aligned}$$

The boundary conditions in L can be written, after some simplifications, in terms of a homogeneous linear system

$$\begin{bmatrix} sin\beta_n L - \sinh\beta_n L & \cos\beta_n L - \cosh\beta_n L \\ sin\beta_n L + \sinh\beta_n L & \cos\beta_n L + \cosh\beta_n L \end{bmatrix} \begin{Bmatrix} A_n \\ B_N \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

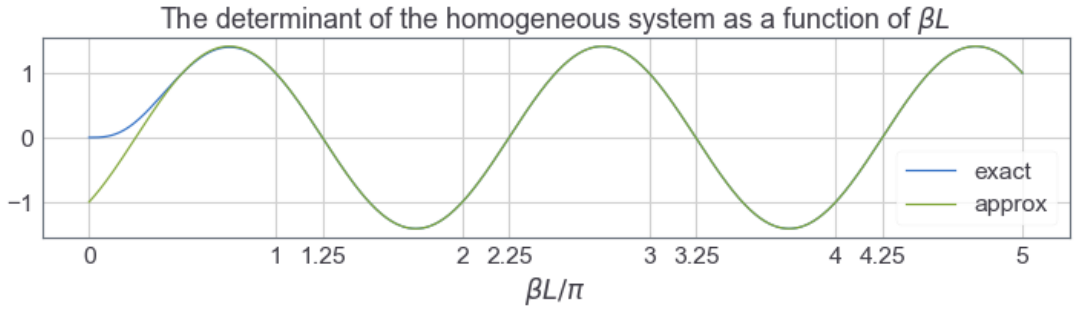
Equating the determinant of the coefficent matrix to zero gives

$$2\cosh\beta_n L \sin\beta_n L - 2\cos\beta_n L \sinh\beta_n L = 0 \quad \leftrightarrow \quad \sin\beta_n L = \cos\beta_n L \tanh\beta_n L.$$

Because for moderately large arguments it is $\tanh\beta L \simeq 1$ we can write that the determinant is (almost) equal to zero when $\sin\beta_n L = \cos\beta_n L$, i.e., when $\beta L = \pi/4 + n\pi$.

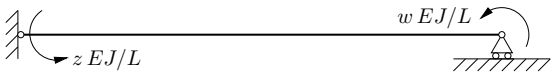
It is useful to plot $\sin\beta_n L - \cos\beta_n L \tanh\beta_n L$ and $\sin\beta_n L \cos\beta_n L$ against $\beta L/\pi$,

```
BetaL = linspace(0,5,501)
det_exact  = sin(pi*BetaL)-cos(pi*BetaL)*tanh(pi*BetaL)
det_approx = sin(pi*BetaL)-cos(pi*BetaL)
```



Not interested in the trivial solution $\beta_n L = 0$, it's apparent that the roots are $\beta_n L \simeq (n + 1/4)\pi$ but we can compute numerically more precise approximations to the roots of the determinant and eventually the frequencies of vibration, $\omega_n^2 = (\beta_n L)^4 \omega_0^2$.

```
N = 3
roots = array([newton(lambda x: sin(x)-cos(x)*tanh(x), (r+1.25)*pi)
               for r in range(N)])
BetaL = roots
w1 = BetaL**2
w2 = w1**2
```



The adimensional wavenumbers

$$\beta_1 L = 3.9266, \qquad \beta_2 L = 7.06858, \qquad \beta_3 L = 10.2102.$$

The adimensional frequencies

$$\frac{\omega_1}{\omega_0} = 15.4182, \qquad \frac{\omega_2}{\omega_0} = 49.9649, \qquad \frac{\omega_3}{\omega_0} = 104.248.$$

The adimensional squared frequencies

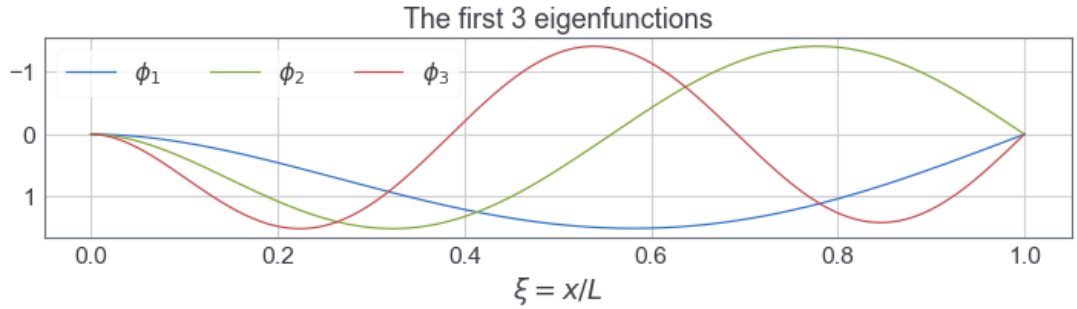
$$\frac{\omega_1^2}{\omega_0^2} = 237.721, \qquad \frac{\omega_2^2}{\omega_0^2} = 2496.49, \qquad \frac{\omega_3^2}{\omega_0^2} = 10867.6.$$

The eigenfunctions It is $\phi_n = (\sin\beta_n x - \sinh\beta_n x)A_n + (\cos\beta_n x - \cosh\beta_n x)B_n$, with (from the displacement boundary condition in L)

$$(\sin\beta_n L - \sinh\beta_n L)A_n + (\cos\beta_n L - \cosh\beta_n L)B_n = \mathcal{C}_{A_n} A_n + \mathcal{C}_{B_n} B_n = 0$$

```
CA, CB = sin(roots)-sinh(roots), cos(roots)-cosh(roots)
A = 1.0 ; B = -CA*A/CB

xi = linspace(0, 1, 1001)
b_xi = outer(xi, roots)
phi = A*(sin(b_xi)-sinh(b_xi)) + B*(cos(b_xi)-cosh(b_xi))
```



Static equilibrium position

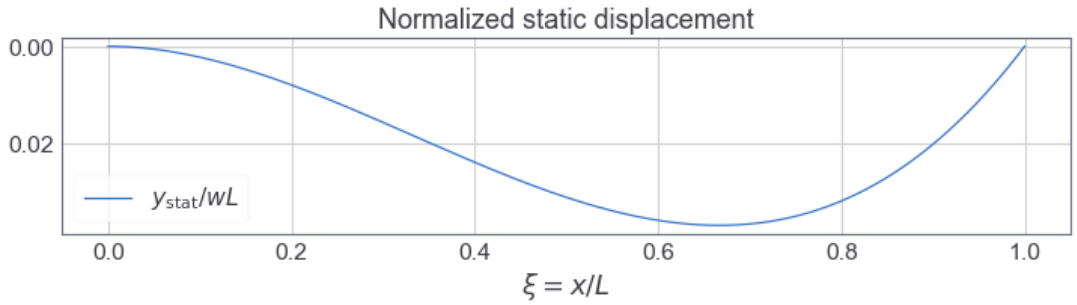
We put in evidence an hyperstatic reaction $Z = z\,EJ/L$, the bending moment is $M = (x-L)Z/L + xW/L$; the curvature is $y'' = -EJM = -(x-L)z/L^2 - xw/L^2$; integrating $y' = -(x-L)^2/2z/L^2 - x^2/2w/L^2 + A$ and $y = -(x-L)^3/6z/L^2 - x^3/6w/L^2 + Ax + B$.

Imposing the boundary conditions $y(0) = zL/6 + B = 0$ and $y'(0) = -z/2 + A = 0$ we have $A = z/2$ and $B = -zL/6$

Substituting in y it is $y=-(x-L)^3/6z/L^2-x^3/6w/L^2+x/2z-L/6z$. evaluating in $x=L$ and imposing a zero displacement we have $y(L)=0=(1/2-1/6)zL-1/6wL\rightarrow z=w/2$.

Substituting again in y and simplifying we have

$$y(x)=\frac{wL}{4}\frac{x^2L-x^3}{L^3}=L\frac{\xi^2-\xi^3}{4}w.$$



Modal expansion of static displacement We can represent the (known) static displacements in terms of a weighted sum of eigenfunctions

$$y(x)=\sum_1^\infty \eta_n\phi_n(x),$$

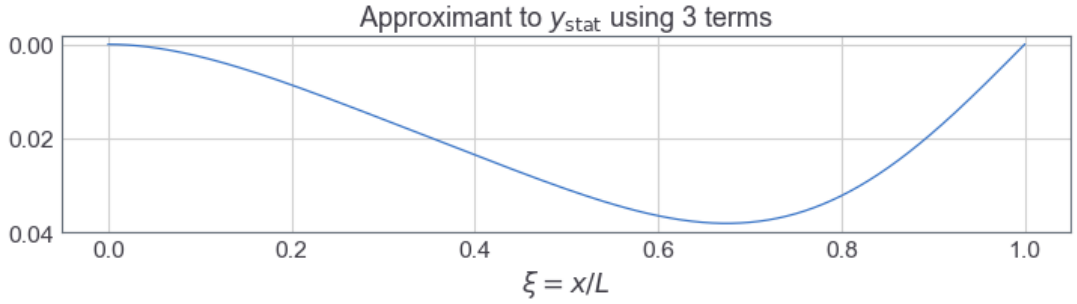
whose coefficients can be easily determined by exploiting the orthogonality relation of the eigenfunctions:

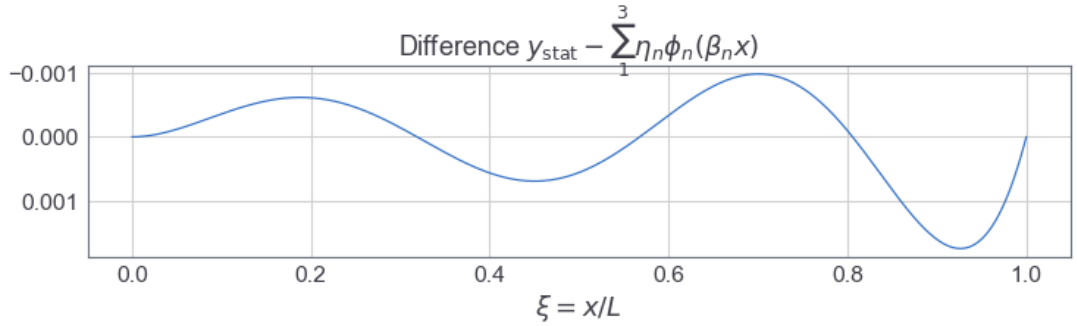
$$\int_0^L \bar{m}(x)\phi_m y\,dx=\delta_{m,n}m_n\eta_n\rightarrow\eta_n=\frac{\int_0^L\bar{m}(x)\phi_n y\,dx}{\int_0^L\bar{m}(x)\phi_n^2 dx}.$$

```
modal_mass =  array([trapz(phi_i**2, dx=0.001) for phi_i in phi.T])
modmass_eta = array([trapz(phi_i*y,  dx=0.001) for phi_i in phi.T])
eta = modmass_eta/modal_mass
```

$$\eta_1=+0.0240387, \qquad \eta_2=-0.00399941, \qquad \eta_3=+0.00132873.$$

Plotting the approximation to the static displacements obtained using a few eigenfunctions is not really useful, but it's so easy... We plot the approximant and the difference between the static displacements and the approximant.





The Modal Responses

We know that the initial velocity is zero, the modal displacements are equal to η_i and so the modal responses are simply

$$\frac{q_n(t)}{wL} = \eta_n \cos\omega_n t = \eta_n \cos(\beta_n^2 L^2 \omega_0 t)$$

$$\frac{q_1}{wL} = +0.0240387\cos(15.4182\omega_0 t), \qquad \frac{q_2}{wL} = -0.00399941\cos(49.9649\omega_0 t),$$

$$\frac{q_3}{wL} = +0.00132873\cos(104.248\omega_0 t).$$

The Bending Moment $M(0,t)$

It is $M = -EJy''$, or

$$M = -EJwL\sum \eta_n \frac{1}{L^2}\frac{d^2\phi_n(\beta_n L\xi)}{d\xi^2}\cos\omega_i t = -\left(\sum \eta_n \frac{d^2\phi_n(\beta_n L\xi)}{d\xi^2}\cos\omega_i t\right)W.$$

Because

$$-\frac{d^2\phi_n(\beta_n L\xi)}{d\xi^2}\bigg|_{\xi=0} = \beta_n^2 L^2\big((\sin\beta_n\xi + \sinh\beta_n\xi)A_n + (\cos\beta_n\xi + \cosh\beta_n\xi)B_n\big)\big|_{\xi=0} = 2\frac{\omega_n}{\omega_0}B_n$$

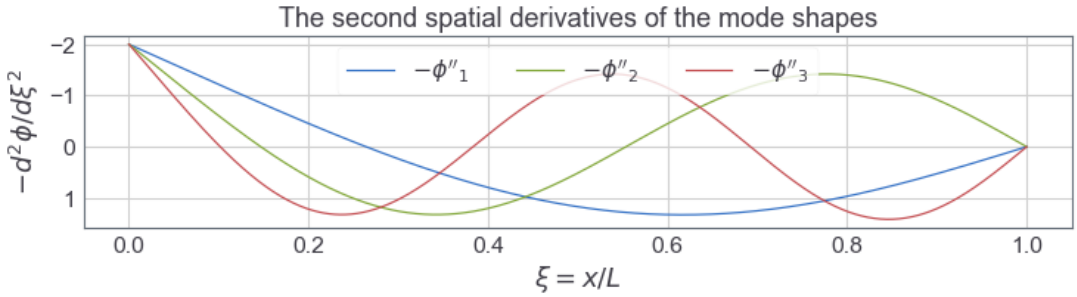
we can finally write $M(0,t) = 2W\sum \frac{\omega_n}{\omega_0}\eta_n B_n \cos(\frac{\omega_n}{\omega_0}\omega_0 t)$

$$\frac{M(0)}{W} = -0.740691\cos(15.42\omega_0 t) + 0.399659\cos(49.96\omega_0 t) - 0.277035\cos(104.25\omega_0 t) + \ldots$$

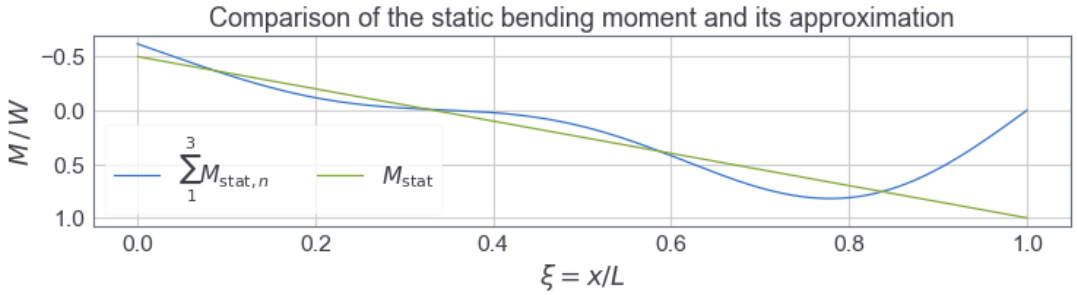
Static bending moment Before proceding with $M(0,t)$ it is interesting to compute the approximation to $M(x,0)$, that is *how well* we are approximating the bending moment at the beginning of our analysis.

$$M(x,0) = -EJwL\sum \eta_n \frac{d^2\phi_n}{dx^2}\cos(\omega_n 0) = \\ = W\sum \frac{\omega_n}{\omega_0}\eta_n\left((\sin\beta_n\xi + \sinh\beta_n\xi)A_n + (\cos\beta_n\xi + \cosh\beta_n\xi)B_n\right)$$

```
# d2y is the NEGATIVE of the second spatial derivative of \phi
d2y = A*(sin(b_xi)+sinh(b_xi)) + B*(cos(b_xi)+cosh(b_xi))
M_stat = d2y@(eta*BetaL**2)
```



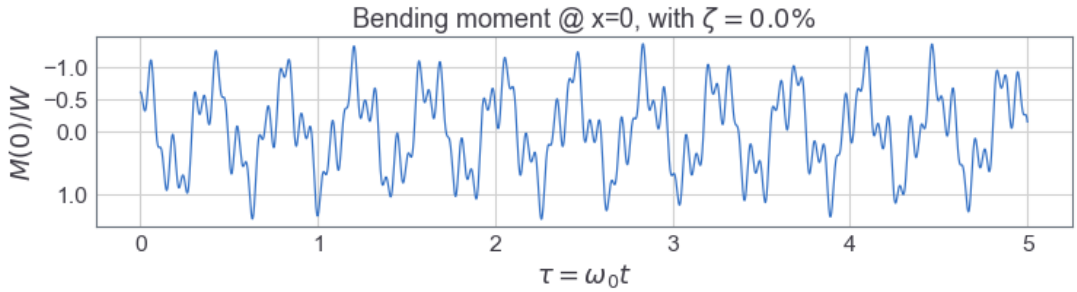

```
-0.6180668193431146
```



As you can see, we are in trouble inasmuch the bending moment in $x = L$ *can not* be approximated correctly, as we are summing contributions of the type $\phi''(L) \equiv 0$. Ask your favourite search engine about "Gibbs phenomenon".

The response in terms of bending at the fixed support We need an 1D array with the instants of (dimensionless) time, a 2D array with the arguments of the cosines (i.e., time multiplied by the frequencies), a 2D array with the cosines of said arguments and finally we multiply the cosines by the coefficients

```
t = linspace(0, 5, 5001)
wt = outer(t, BetaL**2)
cwt = cos(wt)
M = cwt@(2*B*eta*BetaL**2)
```



Let's see what happens if we take into account a very small value of viscous damping.

Similar to what we have done for MDoF systems, we assume that the modal shapes are not affected by the presence of damping and that the modal response is

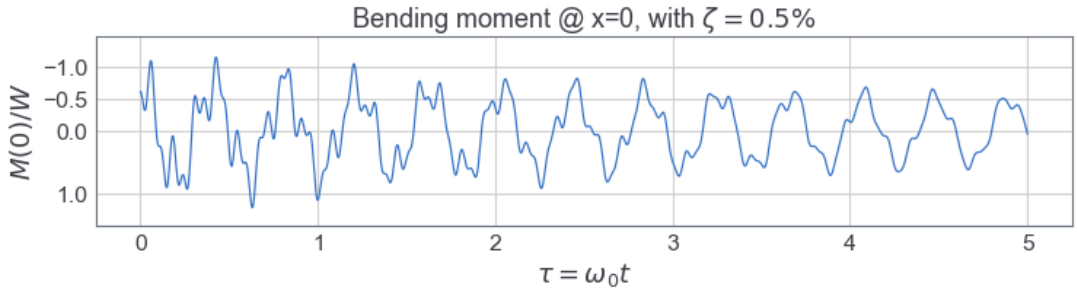
$$q_i(t) = \exp(-\zeta\omega_i t)(A_i \sin\omega_{D,i} t + B_i \cos\omega_{D,i} t).$$

In our case, $q_i(0) = q_{o,i}$ and $\dot{q}_i(0) = 0$, it is $B_i = q_{o,i}$ and $A_i = \zeta B_i / \sqrt{1-\zeta^2}$; for small ζ, we can approximately write $\omega_{D,i} \simeq \omega_i$ and $A_i \simeq \zeta B_i$.

Let's say $\zeta = 0.5\%$ and compute and plot the (approximated) response

```
zeta = 0.005
swt = sin(wt)
Md = (exp(-zeta*wt)*(zeta*swt+cwt))@(2*B*eta*BetaL**2)
```

```
if 1:
    plt.plot(t, Md)
    plt.title('Bending moment @ x=0, with $\\zeta=0.5\\%$')
    plt.xlabel(r'$\tau=\omega_0t$')
    plt.ylabel(r'$M(0)/W$')
    plt.ylim((-1.5, 1.5))
    plt.yticks((-1, -0.5, 0, 1))
    rev_y()
```



As you can see, the effects of viscous damping are really important for the high frequency modal components, that is because the decrement

$$\exp(-\zeta\omega_n t) = \exp(-\zeta\frac{2\pi}{T_n}t) = e^{-2\pi\zeta}e^{\frac{t}{T_n}} = e^{-2\pi\zeta}e^{n_{\text{cycles}}}.$$

is proportional to the *exponential* of the number of cycles involved...

Here we stop with this printed solution, but if you like you can download the notebook that is the source of this paper and as an added bonus you'll receive an interactive plotting widget that lets you investigate the variation of the response with respect to the value of the damping ratio. A further bonus (well, possibly so...) is the code that produced the plots and the mathematical displays, code that's been *hidden* in this paper to streamline the narrative but could be interesting from a different point of view.

```
def p(z=0.5):
    zeta = z/100.0
    M = (exp(-zeta*wt)*(zeta*swt+cwt))@(2*B*eta*BetaL**2)
    plt.plot(t, M)
    plt.title('Bending moment @ x=0, with $\\zeta=%.1f\\%%$'%z)
    plt.xlabel(r'$\tau=\omega_0t$')
    plt.ylabel(r'$M(0)/W$')
    plt.ylim((-1.5, 1.5))
    plt.yticks((-1, -0.5, 0, 1))
    rev_y()
interact(p, z=(0, 2.0)) ;
```

```
interactive(children=(FloatSlider(value=0.5, description='z', max=2.0), Outp
inter...
```

The Initialization Cells

I have placed the inizialization cells here, rather than at the beginning of the notebook, to not disturb the flow of the narrative. We have all the necessary imports, plus the definition of some helper functions, mostly connected to the formatting of the results...

The import statements

```
%matplotlib inline
import numpy as np
from numpy import array, cos, cosh, exp, linspace, outer
from numpy import pi, sin, sinh, sqrt, tanh, trapz
import matplotlib.pyplot as plt
from scipy.optimize import newton
from jupyterthemes import jtplot
from itertools import count, zip_longest
from ipywidgets import interact
```

Configuration statements and function definitions

```
jtplot.style(context='paper', fscale=1.5, figsize=(15, 3))
def rev_y(): plt.ylim(plt.ylim()[::-1])

def dL(*l):
    from IPython.display import Latex
    display(Latex(' '.join(l)))

def grouper( iterable, N):
    return zip_longest(*([iter(iterable)]*N))

def groupalign(seq, N, variable='x_{%d}', fmt='%g', closing='.'):
    beg = r'\begin{align*}'
    end = closing + r'\end{align*}'
    xfmt = variable + '&=' + fmt
    g = grouper(seq, N)
    body = r',\\'.join(
        ',&'.join(
            xfmt%(1+i+N*j, x) for i, x in enumerate(items) if x != None)
        for j, items in enumerate(g))
    return ''.join((beg, body, end))
```