

glockenspiel

October 12, 2024

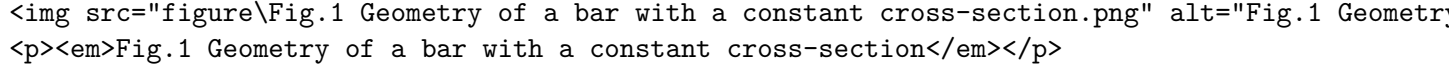
1 Modelisation of the glockenspiel

1.1 Physical model: transverse vibration of bars

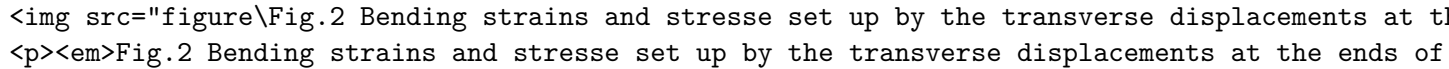
1.1.1 Establish the differential equation and its solution

Bar models are well suited to the description of tuned mallet percussion instruments with keyboard such as the xylophone, glockenspiel, etc. For these instruments, the bending transverse vibrations are dominant. However, other modes, including torsional modes, can also be excited. This is particularly true in the upper frequency range of these instruments, when the length of the beam becomes comparable to the other dimensions, and **therefore treating the beam as a “slender solid” is no longer valid.**

Consider a straight bar of length L , having a uniform cross section S with bilateral symmetry, as shown on *Fig. 1*.

 **Fig.1 Geometry of a bar with a constant cross-section**

When the bar is bent as indicated in *Fig. 2*, the lower part is compressed and the upper part is stretched. Somewhere between the top and the bottom of the bar there will be a neutral axis whose length remains unchanged.

 **Fig.2 Bending strains and stresses set up by the transverse displacements at the ends of**

Now consider a segment of the bar of length dx , and assume that the bending of the bar is measured by the radius of curvature R of the neutral axis. Let $\delta x = (\partial y / \partial x) dx$ be the increment of length, due to bending, of a filament of the bar located at a distance r from the neutral axis. Then the longitudinal force df is given by

$$df = -EdS(\delta x / \partial x) = -EdS(\partial y / \partial x)$$

where E is a constant of Young's Modulus, dS is the cross-sectional area of the filament. The value of δx for the particular filament considered in *Fig. 2* is positive, so that df is a tension, and consequently negative. For filaments below the neutral axis δx is negative, giving a positive force of compression.

Now from the geometry $(dx + \delta x) / (R + r) = dx / R$ and hence $\delta x / dx = r / R$. Substitution into equation of df yields

$$df = -(E/R)r dS$$

The total longitudinal force $f = \int df$ is zero, negative forces above the neutral axis being canceled by positive forces below it. However, a bending moment M is present in the bar,

$$M = \int r df = -\frac{E}{R} \int r^2 dS$$

we define a constant $\kappa = \frac{1}{S} \int r^2 dS$, then

$$M = -ES\kappa^2/R.$$

The constant κ can be thought of as the radius of gyration of the cross-sectional area S , by analogy with the definition of the radius of gyration of a solid. The value of κ for a bar of rectangular cross section and thickness t (measured in the direction) is $\kappa = t/\sqrt{12}$. For a circular rod of radius a , $\kappa = a/2$.

The radius of curvature R is not in general a constant but is rather a function of position along the neutral axis. If the displacements y of the bar are limited to small values, $\partial y/\partial x \ll 1$, then we may use the approximate relation

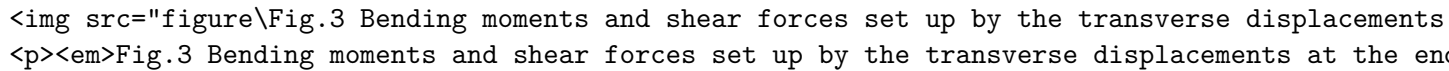
$$R = \frac{[1 + (\partial y/\partial x)^2]^{3/2}}{\partial^2 y/\partial x^2} = \frac{1}{\partial^2 y/\partial x^2}$$

and therefore

$$M = -ES\kappa^2 \frac{\partial^2 y}{\partial x^2}$$

In the situation illustrated in *Fig. 2*, the curvature makes $\frac{\partial^2 y}{\partial x^2}$ negative, and the bending moment M is consequently positive. It is apparent that to obtain the curvature illustrated, the torque applied to the left end of the segment dx must act in a counterclockwise or positive angular direction, so that $M = -ES\kappa^2 \frac{\partial^2 y}{\partial x^2}$ gives the torque acting on the left end of the segment both as to magnitude and as to direction. Similarly, the torque acting on the right end of the segment must be clockwise, with the result that it is negative and is therefore represented both in direction and in magnitude by $-M$.

The effect of distorting the bar is to produce not only bending moments but also shear forces. Consider an upward shear force F_y acting on the left end of the segment dx as positive as *Fig. 3*. Then the associated shear force acting on the right end of the segment must be downward, and is consequently negative. When a bent bar is in a condition of static equilibrium, the torques and shear forces acting on any segment must produce no net turning moment.

 **Fig.3 Bending moments and shear forces set up by the transverse displacements at the ends of a segment.**

Taking moments about the left end of the segment of *Fig. 3*, we have

$$M(x) - M(x + dx) = F_y(x + dx)dx$$

For segments of small length dx , $M(x + dx)$ and $F_y(x + dx)$ can be expanded in Taylor's expansions about x , and this yields

$$F_y = -(\partial M/\partial x) = ES\kappa^2(\partial^3 y/\partial x^3)$$

where second-order terms in dx have been dropped.

This relation between the shear force F_y and the bending moment M has been derived for a condition of static equilibrium. For transverse vibrations of a bar the equilibrium is dynamic, rather than static, and the right side of equation ($M(x) - M(x + dx) = F_y(x + dx)dx$) must equal the rate of increase of angular momentum of the segment. However, if the displacement and slope of the bar are limited to small values, the variations in angular momentum may be neglected, and F_y serves as an adequate approximation for the relation between F_y and y .

The net upward force dF_y acting on the segment dx is then given by

$$dF_y = F_y(x) - F_y(x + dx) = -\frac{\partial F_y}{\partial x}dx = -ES\kappa^2 \frac{\partial^4 y}{\partial x^4}dx$$

By Newton's second law, this force will give the mass (ρSdx) of the segment an upward acceleration $\partial^2 y / \partial t^2$ so that the equation of motion is

$$\frac{\partial^2 y}{\partial t^2} = -(\kappa c)^2 \frac{\partial^4 y}{\partial x^4}$$

where $c^2 = E/\rho$. One significant difference between this differential equation and the simpler equation for the transverse waves on a string is the presence of a fourth partial derivative with respect to x , rather than a second partial. As a result, direct substitution shows that functions of the form $f(ct - x)$ are not solutions of previous differential equation. Transverse waves do not travel along the bar with a constant speed c and unchanging shape.

Assume that the differential equation may be solved by separation of variables, and write the complex transverse displacement as

$$y(x, t) = Y(x)T(t)$$

Upon the substitution, the differential equation could be written as

$$-\frac{(c\kappa)^2}{Y(x)} \frac{\partial^4 Y(x)}{\partial x^4} = \frac{1}{T(t)} \frac{\partial^2 T(t)}{\partial t^2} = -\omega^2$$

and we could easily get the form of $T(t)$, $Y(x)$, and divide this equation into two separate equations

$$T(t) = E \cos(\omega t) Y(x) = e^{\gamma x}$$

Solve these two equations, we get $\gamma^4 = \frac{\omega^2}{(c\kappa)^2}$. Thus $\gamma = \pm \omega/v$ or $\gamma \pm j \pm \omega/v$, where $v^2 = \omega \kappa c$ and v has the dimensions of a speed. If we define a quantity g by $g = \omega/v$, then a complete monofrequency solution can be written as

$$y(x, t) = [A \cosh(gx) + B \sinh(gx) + C \cos(gx) + D \sin(gx)] \cos(\omega t + \phi)$$

1.1.2 Boundary conditions

The glockenspiel can be seen as a free-free ends model.

At a free end there can be neither an externally applied torque nor a shearing force, and hence both M and F_y are zero at the end. However, the displacement and slope are not constrained, except that their values must be small. Then the boundary conditions are

$$\frac{\partial^2 y}{\partial x^2} = 0 \quad \frac{\partial^3 y}{\partial x^3} = 0$$

Using these boundary conditions on glockenspiel at $x = 0$ and $x = L$, we could get $A = C, B = D$, and $\cosh(gL) \cos(gL) = 1$.

We obtain the natural frequencies for the transversely vibrating free-free bar

$$f = [3.011^2, 5^2, 7^2, 9^2, \dots] \frac{\pi \kappa C}{8L^2}$$

1.1.3 Model summary

From the differential equation

$$\frac{\partial^2 y}{\partial x^2} = -(\kappa c)^2 \frac{\partial^4 y}{\partial x^4}$$

where $\kappa = \frac{1}{S} \int r^2 dS$ and $\kappa^2 = SI$, $c^2 = E/\rho$, we obtain the solution

$$y(x, t) = [A \cosh(gx) + B \sinh(gx) + C \cos(gx) + D \sin(gx)] \cos(\omega t + \phi)$$

where $g = \omega/v$, $v^2 = \omega \kappa c$. Using the boundary conditions of free-free bar, the solution could be simplified as

$$A = C, B = D \cosh(gL) \cos(gL) = 1$$

and the natural frequencies are

$$f = [3.011^2, 5^2, 7^2, 9^2, \dots] \frac{\pi \kappa C}{8L^2}$$

.

Therefore, if we know the parameters and material of the bar (L, b, h, ρ, E) , we could get the sound of one piece of glockenspiel.

1.2 Simulation

1.2.1 imports

```
[1]: import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Audio
```

1.2.2 parameters

```
[2]: # all these parameters are in SI units, assumed the material is steel
L = 0.5 # [m]
b = 0.05 # [m]
h = 0.008 # [m]
rho = 7800 # [kg/m^3]
E = 2.1e11 # [Pa]

Fs = 44100 # [Hz] sampling frequency
C = 460 # [J/kgK] specific heat capacity
k = 11 # [W/mK] thermal conductivity

kappa = np.sqrt(h**2 / 12)
```

```

c = np.sqrt(E / rho)

time_duration = 5 # [s]

x0 = 0.5 * L # [m] initial position of the hammer
v0 = 1 # [m/s] initial velocity of the hammer

xc = 0.5 * L # [m] position of the sensor

N = 5 # number of modes to consider

time_scale = np.linspace(0, time_duration, Fs*time_duration)

```

1.2.3 natural frequencies and modes

```

[3]: def natural_freq(n, kappa, c, L):
    # @param n: mode number, strat from 1
    # @param kappa: bending stiffness
    # @param c: wave speed
    # @param L: length of the beam
    # @return: natural frequency of the beam

    if n == 1:
        param = 3.011**2
    else:
        param = (2*n + 1)**2
    return param * np.pi * kappa * c / (8 * L**2)

def mode_shape(n, x, kappa, c, L):
    # @param n: mode number, strat from 1
    # @param x: position along the beam, 0 <= x <= L
    # @param kappa: bending stiffness
    # @param c: wave speed
    # @param L: length of the beam
    # @return: mode shape of the beam

    g = np.sqrt((2 * np.pi * natural_freq(n, kappa, c, L)) / (c * kappa))
    portion = (np.cos(g*L) - np.cosh(g*L)) / (np.sinh(g*L) - np.sin(g*L))
    return np.cosh(g*x) + np.cos(g*x) + portion * (np.sinh(g*x) + np.sin(g*x))

```

```

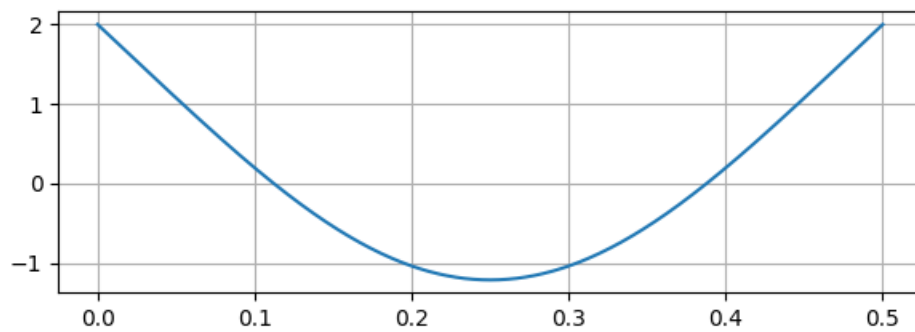
[4]: x = np.linspace(0, L, 1000)

fig, ax = plt.subplots(N, 1, figsize=(6, 12))
for i in range(1, N+1):
    # plot the mode shapes in a subfigure positioned at (1,i)
    ax[i-1].plot(x, mode_shape(i, x, kappa, c, L))
    ax[i-1].set_title(f"Mode {i}")

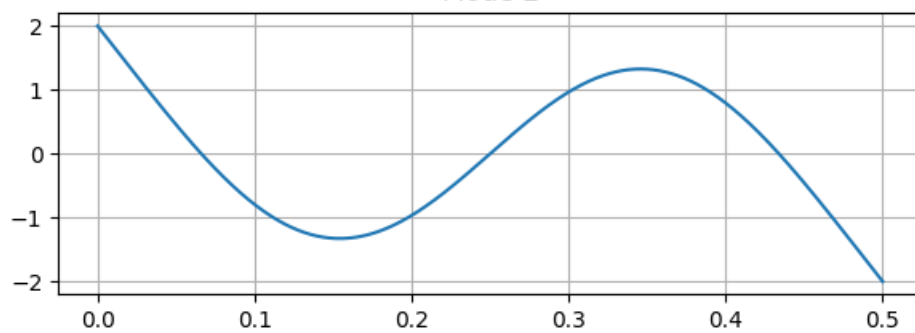
```

```
    ax[i-1].grid()
    ax[i-1].xlim = (-2, 2)
plt.tight_layout()
plt.show()
```

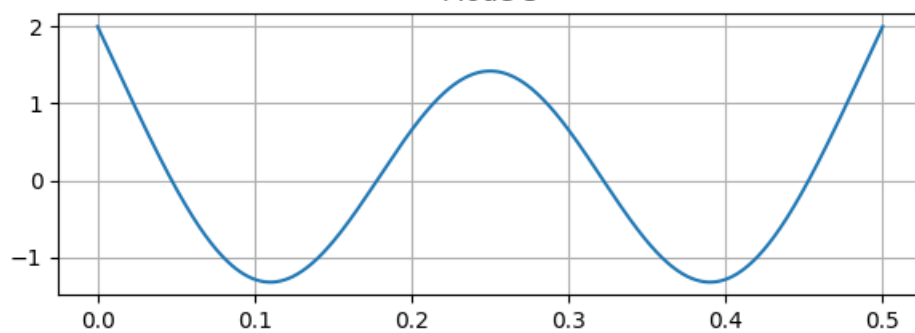
Mode 1



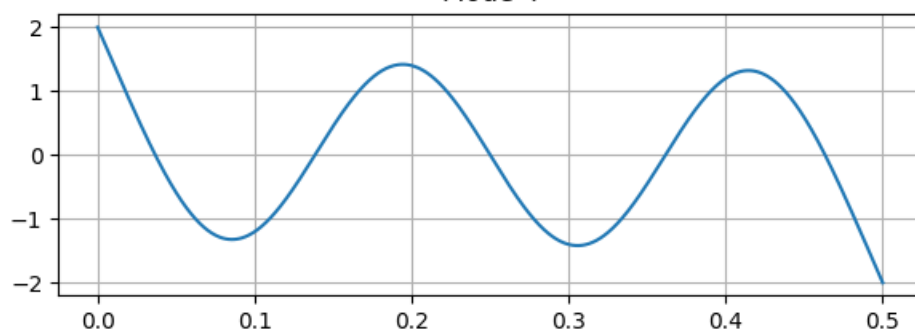
Mode 2



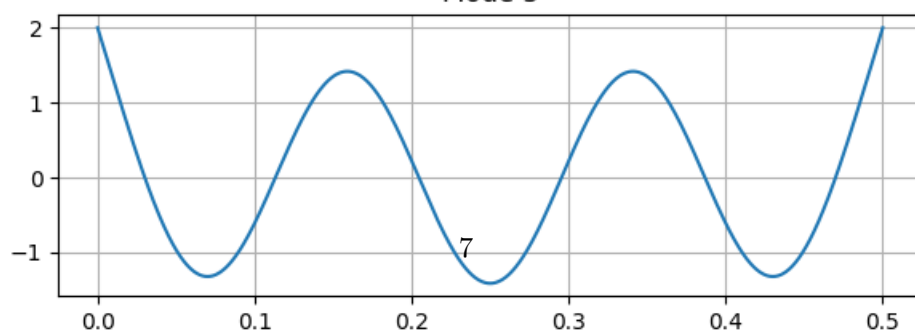
Mode 3



Mode 4



Mode 5



1.2.4 use initial conditions to decide ϕ

For each mode n we have the $Y_n(x)$ and $T_n(t)$, therefore use the superposition theorem, the total wave solution is

$$y(x, t) = \sum_{n=1}^N \alpha_n Y_n(x) T_n(t)$$

where α_n is the parameter.

Assume we knock the bar at $x = x_0$ with speed of v_0 , then the initial condition is

$$y(x, 0) = 0 \quad \frac{\partial y}{\partial t}(x, 0) = v_0 \delta(x - x_0)$$

where $\delta(x)$ is the Dirac delta function.

Therefore, we could get the ϕ by

$$\cos(\phi) = 0 \quad \phi = \frac{\pi}{2}$$

and the α_n by

$$\sum_{n=1}^N \alpha_n Y_n(x) \omega_n = v_0 \delta(x - x_0)$$

```
[5]: def getAlpha(N, kappa, c, L, x0, v0, Fs):
    # @param N: number of modes
    # @param kappa: bending stiffness
    # @param c: wave speed
    # @param L: length of the beam
    # @param x0: initial position of the impact
    # @param v0: initial velocity of the impact
    # @param Fs: sampling frequency
    # @return: alpha, a matrix of shape (N, 1)

    x = np.linspace(0, L, int(Fs*L))
    direc = np.zeros((int(Fs*L), 1))
    direc[int(Fs*L*x0)] = v0

    omegas = np.array([2 * np.pi * natural_freq(n, kappa, c, L) for n in
    ↪range(1, N+1)])
    Ys = np.array([mode_shape(n, x, kappa, c, L) for n in range(1, N+1)])

    beta = np.linalg.lstsq(Ys.T, direc, rcond=None)[0]
    # beta = alpha * omega
    alpha = (beta / omegas[:, np.newaxis]).flatten()
    return alpha

def getDisplacement(alpha, kappa, c, L, x, t):
```



```

# @param alpha: a matrix of shape (N, 1) indicating the parameter of each mode
# @param kappa: bending stiffness
# @param c: wave speed
# @param L: length of the beam
# @param x: position along the beam, 0 <= x <= L
# @param t: time
# @return: displacement of the beam at position x and time t

N = len(alpha)
omegas = np.array([2 * np.pi * natural_freq(n, kappa, c, L) for n in
↪range(1, N+1)])
Ys = np.array([mode_shape(n, x, kappa, c, L) for n in range(1, N+1)])
Ts = np.sin(omegas * t)
return np.sum(alpha.T @ (Ts.T * Ys), axis=0)

```

```

[6]: alpha_n = getAlpha(N, kappa, c, L, x0, v0, Fs)
psi_n = np.array([getDisplacement(alpha_n, kappa, c, L, xc, t) for t in
↪time_scale])

```

1.2.5 add term of damping

The thermoelastic damping is a consequence of the coupling between elastic strain and heat diffusion. It affects the vibrations of solids with a noticeable thermal conductivity, such as metals. We simply assume that the term of damping is

$$e^{-t/\tau} \tau = \frac{\rho C h^2}{2\pi^2 k}$$

where τ is the damping time and C is the specific heat capacity [J/kg.°C], k is the thermal conductivity [W/m.K], and add this term directly to the solution. Thus the solution is

$$y(x, t) = \sum_{n=1}^N \alpha_n Y_n(x) T_n(t) e^{-t/\tau}$$

and we draw the curve of the term of damping:

```

[7]: def getTermAmorti(rho, C, k, t):
# @param rho: density of the material
# @param C: specific heat capacity
# @param k: thermal conductivity
# @param t: time
# @return: term of amorti with respect to time t
tau = rho * C * h**2 / (2 * np.pi**2 * k)
return np.exp(-t / tau)

```

```

[8]: amorti = np.array([getTermAmorti(rho, C, k, t) for t in time_scale])

# plot the amortissement term
fig, ax = plt.subplots()

```

```

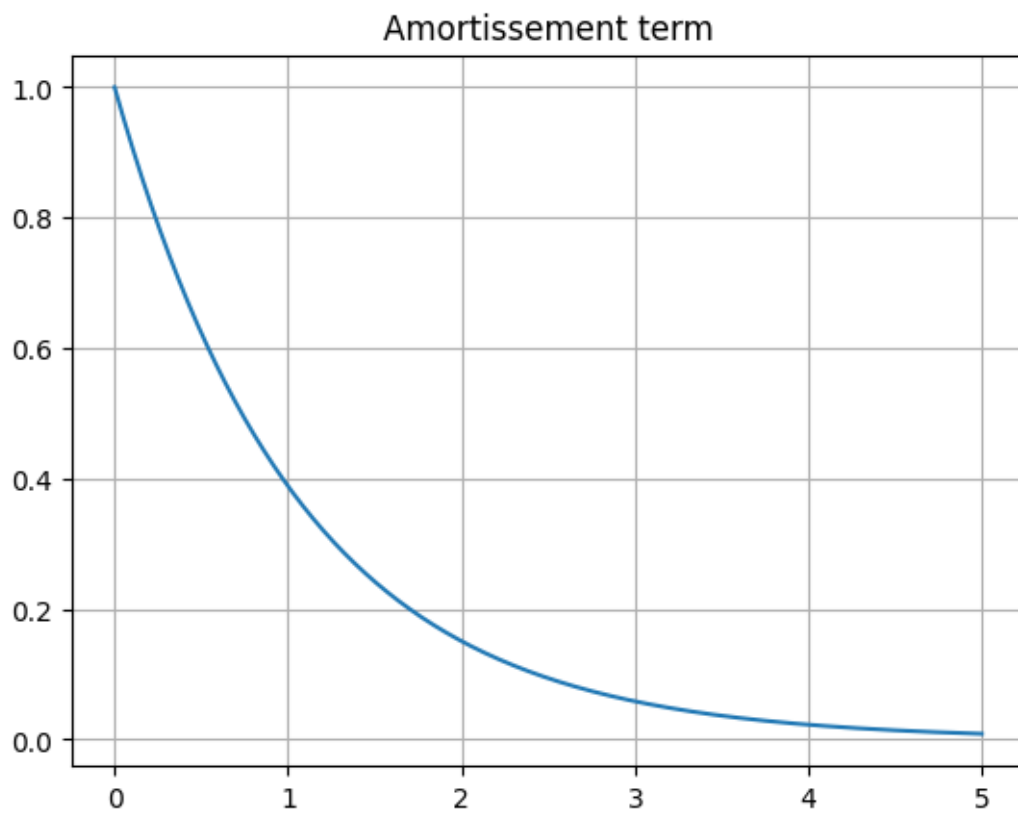
ax.plot(time_scale, amorti)
ax.set_title("Amortissement term")
ax.grid()
plt.show()

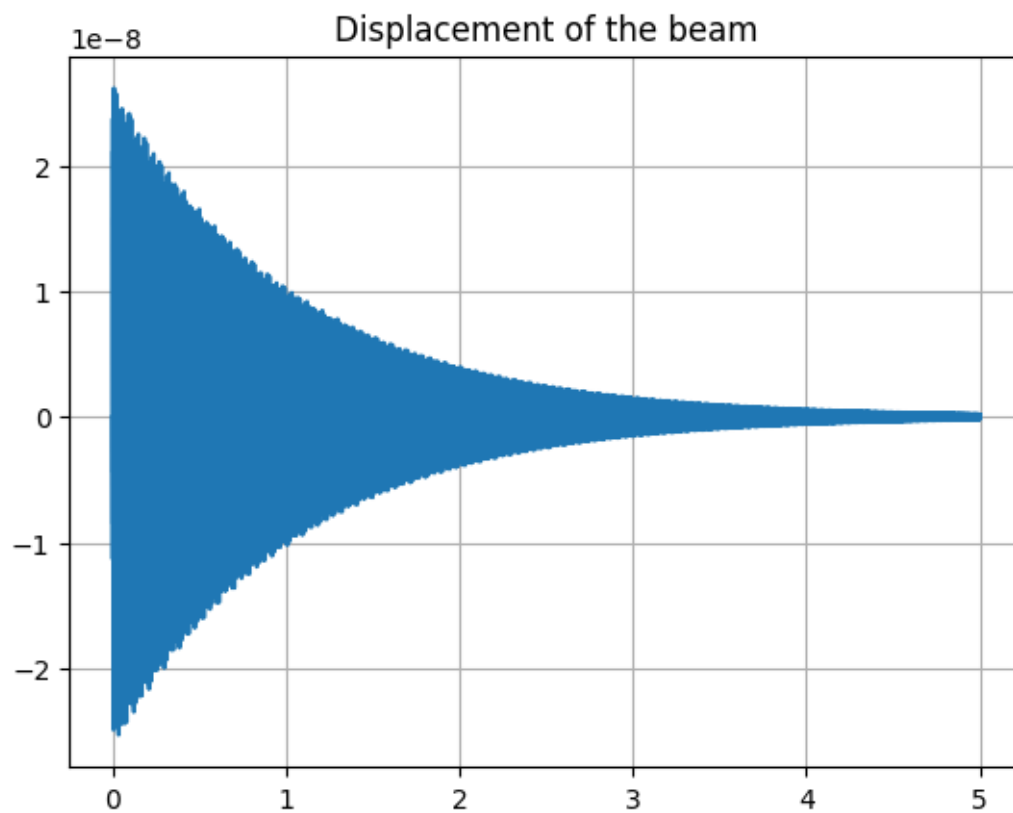
y = psi_n * amorti

# plot the displacement of the beam
fig, ax = plt.subplots()
ax.plot(time_scale, y)
ax.set_title("Displacement of the beam")
ax.grid()
plt.show()

Audio(y, rate=Fs)

```





[8]: <IPython.lib.display.Audio object>