Modelling structures with piezoelectric materials : full, reduced order, and state-space models

Theory and SDT Tutorial

Etienne Balmes
Arnaud Deraemaeker

SDTools
44 rue Vergniaud
75013 Paris (France)

Tel.	+33 1 44 24 63 71
General info	http://www.sdtools.com
Support	http://support.sdtools.com

Contents

1
Release notes

This manual gives a more detailed set of examples for the use of SDT for the modeling of piezoelectric structures.
Major modifications for last release are

- General reorganisation of the document
- New format of graphics and figures with improved style sheets
- New script to compute dynamic impedance of a piezoelectric disk in `d_piezo('ScriptTutoPz_disk_impedance')`

Major modifications for SDT 7.1 are

- Inclusion of new materials (Ferroperm, Sonox, MFCs) in the `m_piezo` database.
- Introduction of tutorials in `d_piezo('Tuto')`.
- New script for Macro Fiber Composites in `d_piezo('TutoPlate_mfc')`.
- Theory and new script for point load actuator using a shaped triangular piezoelectric transducer in `d_piezo('TutoPlate_triang')`.
- Theory and new script for vibration damping using RL shunt and piezoelectric patches in `d_piezo('TutoPz_shunt')`.
- Theory and new script for piezoelectric homogenization on RVEs of piezocomposites (application to MFCs) in `d_piezo('TutoPz_P1_homo')` and `d_piezo('TutoPz_P2_homo')` .
- Color visualization of stress and strain added to IDE patch script `d_piezo('TutoPatch_num_IDE')`.

Major modifications for SDT 6.6 were

- Writing of the present manual
- Significant generalization of `p_piezo('Electrode')` commands.
- Inclusion of elastic properties in the `m_piezo` database.
- Introduction of electrical and charge viewing illustrated in this manual.
- Specialized meshing capabilities and examples are grouped in `d_piezo('Mesh')`.

2

Basics of piezoelectricity

Contents

Polarization consists in the separation of positive and negative electric charges at different ends of the dielectric material on the application of an external electric field (Figure 2.1).

Spontaneous polarization is the phenomenon by which polarization appears without the application of an external electric field. Spontaneous polarization has been observed in certain crystals in which the centers of positive and negative charges do not coincide. Spontaneous polarization can occur more easily in perovskite crystal structures.

The level and direction of the polarization is described by the electric displacement vector D:

$$D = \varepsilon E + P \tag{2.1}$$

where P is the *permanent polarization* which is retained even in the absence of an external electric field, and εE represents the polarization induced by an applied electric field. ε is the dielectric permittivity. If no spontaneous polarization exists in the material, the process through which permanent polarization is induced in a material is known as *poling*.

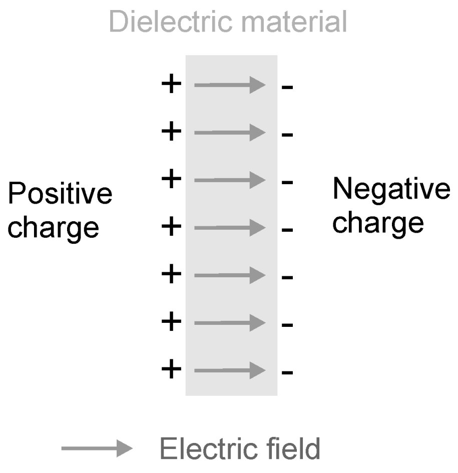


Figure 2.1: Polarization: separation of positive and negative electric charges on the two sides of a dielectric material

Ferroelectric materials have permanent polarization that can be altered by the application of an external electric field, which corresponds to poling of the material. As an example, perovskite structures are ferroelectric below the *Curie temperature*. In the ferroelectric phase, polarization can therefore be induced by the application of a (large) electric field.

Piezoelectricity was discovered by Pierre and Jacques Curie in 1880. The *direct piezoelectric effect* is the property of a material to display electric charge on its surface under the application of an external mechanical stress (i.e. to change its polarization). (Figure 2.2a). *The converse piezoelectric effect* is the production of a mechanical strain due to a change in polarization (Figure 2.2b).

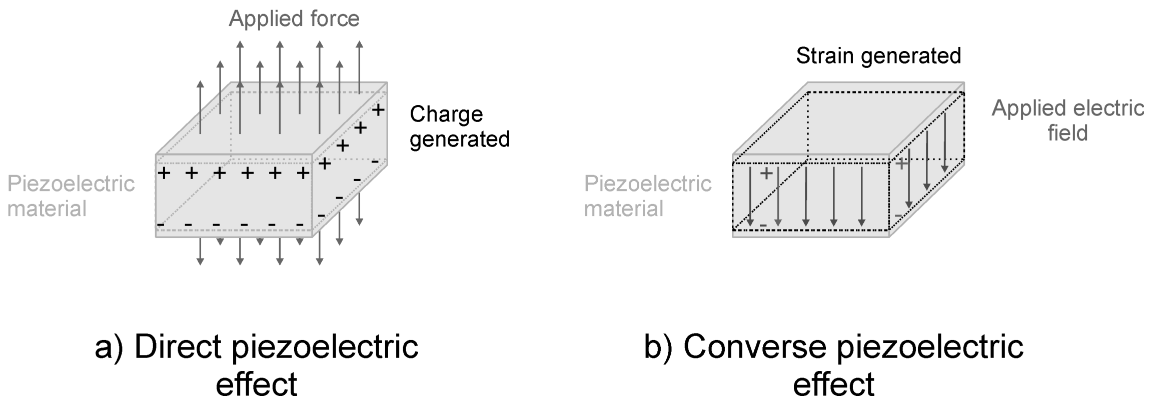


Figure 2.2: Direct and converse piezoelectric effect

Piezoelectricity occurs naturally in non ferroelectric single crystals such as quartz, but the effect is not very strong, although it is very stable. The direct effect is due to a distortion of the crystal lattice caused by the applied mechanical stress resulting in the appearance of electrical dipoles. Conversely, an electric field applied to the crystal causes a distortion of the lattice resulting in an induced mechanical strain. In other materials, piezoelectricity can be induced through *poling.* This can be achieved in *ferroelectric crystals, ceramics or polymers.*

A piezoelectric ceramic is produced by pressing ferroelectric material grains (typically a few micrometers in diameter) together. During fabrication, the ceramic powder is heated (sintering process) above Curie temperature. As it cools down, the perovskite ceramic undergoes phase transformation from the paraelectric state to the ferroelectric state, resulting in the formation of randomly oriented ferroelectric domains. These domains are arranged in grains, containing either 90° or 180° domains (Figure 2.3a). This random orientation leads to zero (or negligible) net polarization and piezoelectric coefficients (Figure 2.3b)).

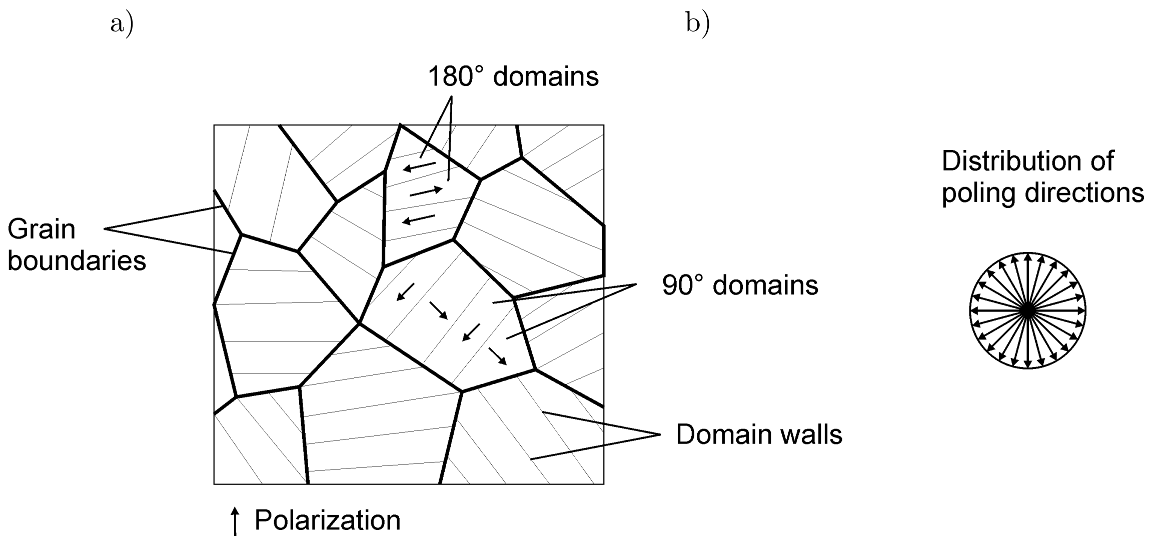


Figure 2.3: Piezoelectric ceramic : a) ferroelectric grains and domains, b) distribution of poling directions

The application of a sufficiently high electric field to the ceramic causes the domains to reorient in the direction of the applied electric field. Note however that the mobility of the domains is not such that all domains are perfectly aligned in the poling direction, but the total net polarization increases with the magnitude of the electric field (Figure 2.4). After removal of the applied electric field, the ferroelectric domains do not return in their initial orientation and a permanent polarization remains in the direction of the applied electric field (the poling direction). In this state, the application of a moderate electric field results in domain motions which are responsible for a deformation of the ceramic and are the source of the piezoelectric effect. The poling direction is therefore a very important material property of piezoelectric materials and needs to be known for a proper modeling.

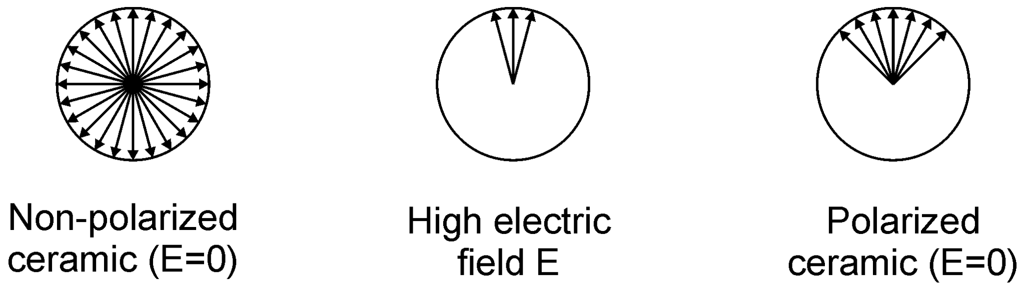


Figure 2.4: Orientation of the ferroelectric domains in non-polarized and polarized ceramics

Typical examples of simple perovskites are Barium titanate ($BaTiO3$) and lead titanate ($PbTiO3$). The most common perovskite alloy is lead zirconate titanate (PZT- PbZr TiO3). Nowadays, the most common ceramic used in piezoelectric structures for structural dynamics applications (active control, shape control, structural health monitoring) is PZT, which will be used extensively in the documented examples.

In certain polymers, piezoelectricity can be obtained by orienting the molecular dipoles within the polymer chain. Similarly to the ferroelectric domains in ceramics, in the natural state, the molecular dipole moments usually cancel each other resulting in an almost zero macroscopic dipole. Poling of the polymer is usually performed by stretching the polymer and applying a very high electric field, which causes the molecular dipoles to orient with the electric field, and remain orientated in this preferential direction after removal of the electric field (permanent polarization). This gives rise to piezoelectricity in the polymer. The technology of piezoelectric polymers has been largely dominated by ferroelectric polymers from the polyvinylidene fluoride (PVDF) family, discovered in 1969. The main advantage is the good flexibility, but their piezoelectric coefficients are much lower compared to ferroelectric ceramics.

2.1 Piezoelectric constitutive laws in 3D

Up to a certain level of electric field and strain, piezoelectric materials behave linearly. This tutorial is restricted to linear piezoelectricity, but the interested reader can refer to [1] for more details on non-linear piezoelectricity.

Assuming a linear piezoelectric material and adopting the notations of the IEEE Standards on piezoelectricity [2], the 3D constitutive equations are given by:

$$\left\{\begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ D_1 \\ D_2 \\ D_3 \end{array}\right\} = \left[\begin{array}{ccccccccc} c_{11}^E & c_{12}^E & c_{13}^E & 0 & 0 & 0 & 0 & 0 & -e_{31} \\ c_{12}^E & c_{22}^E & c_{23}^E & 0 & 0 & 0 & 0 & 0 & -e_{32} \\ c_{13}^E & c_{23}^E & c_{33}^E & 0 & 0 & 0 & 0 & 0 & -e_{33} \\ 0 & 0 & 0 & c_{44}^E & 0 & 0 & 0 & -e_{24} & 0 \\ 0 & 0 & 0 & 0 & c_{55}^E & 0 & -e_{15} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66}^E & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e_{15} & 0 & \varepsilon_{11}^S & 0 & 0 \\ 0 & 0 & 0 & e_{24} & 0 & 0 & 0 & \varepsilon_{22}^S & 0 \\ e_{31} & e_{32} & e_{33} & & 0 & 0 & 0 & 0 & \varepsilon_{33}^S \end{array}\right] \left\{\begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ E_1 \\ E_2 \\ E_3 \end{array}\right\} \tag{2.2}$$

where E_i and D_i are the components of the electric field vector and the electric displacement vector, and T_i and S_i are the components of stress and strain vectors, defined according to:

$$\left\{\begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{array}\right\} = \left\{\begin{array}{c} T_{11} \\ T_{22} \\ T_{33} \\ T_{23} \\ T_{13} \\ T_{12} \end{array}\right\} \qquad \left\{\begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{array}\right\} = \left\{\begin{array}{c} S_{11} \\ S_{22} \\ S_{33} \\ 2\,S_{23} \\ 2\,S_{13} \\ 2\,S_{12} \end{array}\right\} \tag{2.3}$$

Matrix notations are usually adopted leading to:

$$\begin{array}{l} \{T\} = \left[C^E\right]\{S\} - [e]^T\{E\} \\ \{D\} = [e]\{S\} + \left[\varepsilon^S\right]\{E\} \end{array} \tag{2.4}$$

A widely used alternative and equivalent representation consists in writing the constitutive equations in the following form:

$$\begin{array}{l} \{S\} = \left[s^E\right]\{T\} + [d]^T\{E\} \\ \{D\} = [d]\{T\} + \left[\varepsilon^T\right]\{E\} \end{array} \tag{2.5}$$

where the following relationships hold:

$$\left[s^E\right] = \left[c^E\right]^{-1} \tag{2.6}$$

$$[e] = [d]\left[c^E\right] \tag{2.7}$$

$$\left[\varepsilon^S\right] = \left[\varepsilon^T\right] - [d]\,[e]^T \tag{2.8}$$

There are also two additional possibilities to write these constitutive equations, which are less commonly used but are given here for completeness:

$$\begin{array}{l} \{S\} = \left[s^D\right]\{T\} + [g]^T\{D\} \\ \{E\} = -[g]\{T\} + \left[\beta^T\right]\{D\} \end{array} \tag{2.9}$$

$$\begin{array}{l} \{T\} = \left[c^D\right]\{S\} - [h]^T\{D\} \\ \{E\} = -[h]\{S\} + \left[\beta^S\right]\{D\} \end{array} \tag{2.10}$$

The following relationships hold:

$$\left[c^D\right]\left[s^D\right] = I_6 \tag{2.11}$$

$$\begin{array}{l} \left[\beta^S\right]\left[\varepsilon^S\right] = \left[\beta^T\right]\left[\varepsilon^T\right] = I_3 \\ \left[c^D\right] = \left[c^E\right] + [e]^T[h] \\ \left[s^D\right] = \left[c^D\right] - [d]^T[g] \\ \left[\beta^S\right] = \left[\beta^T\right] - [g]^T[h] \\ [d] = \left[\varepsilon^T\right][g] \\ [g] = [h]\left[s^D\right] \end{array} \tag{2.12}$$

$$[h] = \left[\varepsilon^S\right][e] \tag{2.13}$$

The piezoelectric coefficients are contained in the matrix $[d]$ whose structure is specific to each type of piezoelectric material. The typical structure for a z-polarized PZT material is

$$[d] = \begin{bmatrix} 0 & 0 & 0 & 0 & d_{15} & 0 \\ 0 & 0 & 0 & d_{24} & 0 & 0 \\ d_{31} & d_{32} & d_{33} & 0 & 0 & 0 \end{bmatrix} \tag{2.14}$$

Regular PZT ceramics are isotropic in the plane perpendicular to the poling direction ($d_{31} = d_{32}$, $d_{15} = d_{24}$), but piezoelectric composites can have orthotropic properties [3]. PVDF material does not exhibit piezoelectricity in the shear mode, so that the typical structure is:

$$[d] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ d_{31} & d_{32} & d_{33} & 0 & 0 & 0 \end{bmatrix} \tag{2.15}$$

PVDF can be either isotropic or orthotropic in the plane perpendicular to the poling direction, depending on the fabrication process (uni-axial or bi-axial). Table 2.1 gives typical piezoelectric coefficients for PZT ceramics and PVDF films. Note that these properties can vary significantly from the figures in the table, as there are many different material types. The permittivity is usually given with its relative value which is the ratio of the permittivity by the permittivity of vacuum ($\varepsilon_0 = 8.854\,10^{-12} F/m$).

Material properties	PZT	PVDF (bi-axial)
Piezoelectric properties		
d_{33} (pC/N)	440	-25
d_{31} (pC/N)	-185	3
d_{32} (pC/N)	-185	3
Relative permittivity		
ε_r	1800	12
Young's Modulus		
$Y_1 (GPa)$	54	3
$Y_2 (GPA)$	54	3
$Y_3 (GPA)$	48	10
ρ (kg/m^3)	7600	1800

Table 2.1: Typical piezoelectric properties of PZT ceramics and PVDF films

2.2 Piezoelectric constitutive laws in plates

When thin piezoelectric transducers are used with plate structures, the common plane stress hypothesis ($T_3 = 0$) must be used together with an hypothesis for the electric field. When the ceramic is poled through the thickness, the hypothesis commonly adopted is that the electric field is zero in the plane of the transducer ($E_1 = E_2 = 0$). The constitutive equations then reduce to:

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_4 \\ T_5 \\ T_6 \\ D_3 \end{Bmatrix} = \begin{bmatrix} c_{11}^{E*} & c_{12}^{E*} & 0 & 0 & 0 & -e_{31}^{*} \\ c_{12}^{E*} & c_{22}^{E*} & 0 & 0 & 0 & -e_{32}^{*} \\ 0 & 0 & c_{44}^{E*} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{55}^{E*} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{66}^{E*} & 0 \\ e_{31}^{*} & e_{32}^{*} & 0 & 0 & 0 & \varepsilon_{33}^{S*} \end{bmatrix} \begin{Bmatrix} S_1 \\ S_2 \\ S_4 \\ S_5 \\ S_6 \\ E_3 \end{Bmatrix} \tag{2.16}$$

where the superscript * denotes the properties under the "piezoelectric plates" hypothesis ($T_3 = E_1 = E_2 = 0$). These properties are related to the 3D properties with the following relationships:

$$c_{11}^{E*} = \left[c_{11}^{E} - \frac{(c_{13}^{E})^2}{c_{33}^{E}} \right] \tag{2.17}$$

$$c_{12}^{E*} = \left[c_{12}^{E} - \frac{c_{13}^{E}\, c_{23}^{E}}{c_{33}^{E}} \right] \tag{2.18}$$

$$c_{22}^{E*} = \left[c_{22}^{E} - \frac{(c_{23}^{E})^2}{c_{33}^{E}} \right] \tag{2.19}$$

$$e_{31}^{*} = \left[e_{31} - \frac{c_{13}^{E}\, e_{33}}{c_{33}^{E}} \right] \tag{2.20}$$

$$e_{32}^{*} = \left[e_{32} - \frac{c_{23}^{E}\, e_{33}}{c_{33}^{E}} \right] \tag{2.21}$$

$$\varepsilon_{33}^{S*} = \left[\varepsilon_{33}^{S} + \frac{(e_{33})^2}{c_{33}^{E}} \right] \tag{2.22}$$

The distinction is very important, as it is often not well understood and many errors can arise from the confusion between plate and 3D properties of piezoelectric materials. Note however that the d_{ij}, s_{ij}^{E} and ε^{T} coefficients are equal for plate and 3D constitutive equations. It is therefore preferable to handle the material properties of piezoelectric materials in the form of (2.5).

Similarly to the 3D equations, the constitutive equations can be written in a matrix form, separating the mechanical and the electrical parts:

$$\begin{aligned} \{T\} &= \left[c^{E*}\right]\{S\} - \left[e^{*}\right]^{T}\{E\} \\ \{D\} &= \left[e^{*}\right]\{S\} + \left[\varepsilon^{S*}\right]\{E\} \end{aligned} \tag{2.23}$$

Using (2.7) in equations ((2.20),(2.21),(2.22)), one can further show that

$$[e^*] = [d^*] \left[c^{E*}\right] \tag{2.24}$$

and for the permittivity:

$$\varepsilon_{33}^{S*} = \varepsilon_{33}^{T} - [d^*] [e^*]^T \tag{2.25}$$

with

$$[d^*] = \left[\begin{array}{ccccc} d_{31} & d_{32} & 0 & 0 & 0 \end{array} \right] \tag{2.26}$$

and

$$[e^*] = \left[\begin{array}{ccccc} e_{31}^* & e_{32}^* & 0 & 0 & 0 \end{array} \right] \tag{2.27}$$

The values of e_{31}^*, e_{32}^* and ε_{33}^{S*} can therefore be computed knowing the elastic matrix $\left[c^{E*}\right]$ and the values of d_{31} and d_{32} and ε_{33}^{T}

2.3 Database of piezoelectric materials

`m_piezo Dbval` includes a number of material characteristics for piezoelectric materials. The properties are obtained from the datasheet of the material, but as we will illustrate, the data is not always sufficient to calculate all the material properties needed for the computations. Most of the information in the datasheet is generally related to the constitutive equations written in the form of (2.5). For PZT, PVDF, or piezoelectric composites based on PZT and PVDF, the general form of these matrices is:

$$\left\{ \begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ D_1 \\ D_2 \\ D_3 \end{array} \right\} = \left[\begin{array}{ccccccccc} s_{11}^E & s_{12}^E & s_{13}^E & 0 & 0 & 0 & 0 & 0 & d_{31} \\ s_{12}^E & s_{22}^E & s_{23}^E & 0 & 0 & 0 & 0 & 0 & d_{32} \\ s_{13}^E & s_{23}^E & s_{33}^E & 0 & 0 & 0 & 0 & 0 & d_{33} \\ 0 & 0 & 0 & s_{44}^E & 0 & 0 & 0 & d_{24} & 0 \\ 0 & 0 & 0 & 0 & s_{55}^E & 0 & d_{15} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & s_{66}^E & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_{15} & 0 & \varepsilon_{11}^T & 0 & 0 \\ 0 & 0 & 0 & d_{24} & 0 & 0 & 0 & \varepsilon_{22}^T & 0 \\ d_{31} & d_{32} & d_{33} & & 0 & 0 & 0 & 0 & \varepsilon_{33}^T \end{array} \right] \left\{ \begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ E_1 \\ E_2 \\ E_3 \end{array} \right\} \tag{2.28}$$

For an orthotropic material, the compliance matrix $\left[s^E\right]$ can be written as a function of the engineering constant E_i,ν_{ij} and G_{ij} as follows:

$$\left[s^E\right] = \left[\begin{array}{cccccc} \frac{1}{E_x} & \frac{-\nu_{yx}}{E_y} & \frac{-\nu_{zx}}{E_z} & 0 & 0 & 0 \\ \frac{-\nu_{xy}}{E_x} & \frac{1}{E_y} & \frac{-\nu_{zy}}{E_z} & 0 & 0 & 0 \\ \frac{-\nu_{xz}}{E_x} & \frac{-\nu_{yz}}{E_y} & \frac{1}{E_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{yz}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{xz}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{xy}} \end{array} \right] \tag{2.29}$$

where z is aligned with the poling direction 3, and x, y with directions $1, 2$ respectively. Note that the matrix is symmetric so that:

$$\frac{\nu_{yx}}{E_y} = \frac{\nu_{xy}}{E_x}, \quad \frac{\nu_{zx}}{E_z} = \frac{\nu_{xz}}{E_x}, \quad \frac{\nu_{zy}}{E_z} = \frac{\nu_{yz}}{E_y} \tag{2.30}$$

A bulk piezoelectric ceramic exhibits transverse isotropic properties: the properties of the material are the same in the plane perpendicular to the poling direction. In this case, the compliance matrix reduces to:

$$\left[s^E\right] = \begin{bmatrix} \frac{1}{E_p} & \frac{-\nu_p}{E_p} & \frac{-\nu_{zp}}{E_z} & 0 & 0 & 0 \\ \frac{-\nu_p}{E_p} & \frac{1}{E_p} & \frac{-\nu_{zp}}{E_z} & 0 & 0 & 0 \\ \frac{-\nu_{pz}}{E_p} & \frac{-\nu_{pz}}{E_p} & \frac{1}{E_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{zp}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{zp}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{2(1+\nu_p)}{E_p} \end{bmatrix} \tag{2.31}$$

and due to the symmetry we have:

$$\frac{\nu_{zp}}{E_z} = \frac{\nu_{pz}}{E_p} \tag{2.32}$$

where the subscript p refers to the in-plane properties. The matrix of piezoelectric coefficients is:

$$[d] = \begin{bmatrix} 0 & 0 & 0 & 0 & d_{15} & 0 \\ 0 & 0 & 0 & d_{15} & 0 & 0 \\ d_{31} & d_{31} & d_{33} & 0 & 0 & 0 \end{bmatrix} \tag{2.33}$$

and the matrix of dielectric permittivities:

$$\left[\varepsilon^T\right] = \begin{bmatrix} \varepsilon_{11}^T & 0 & 0 \\ 0 & \varepsilon_{11}^T & 0 \\ 0 & 0 & \varepsilon_{33}^T \end{bmatrix} \tag{2.34}$$

In order to use such a piezoelectric material in a 3D model, it is therefore necessary to have access to the 5 elastic constants E_p,E_z,ν_p,ν_{zp} and G_{zp}, 3 piezoelectric constants d_{31},d_{33}, and d_{15} and two dielectric constants ε_{11}^T,ε_{33}^T. Unfortunately, such constants are generally not given in that form, but can be calculated from the material properties found in the datasheet.

It is important to introduce the electromechanical coupling factors which are generally given in the datasheet and are a function of the elastic, piezoelectric and dielectric properties of the material. They measure the effectiveness of the conversion of mechanical energy into electrical energy (and vice-versa). There is one coupling factor for each piezoelectric mode:

$$\begin{aligned} k_{31}^2 &= \frac{d_{31}^2}{\varepsilon_{33}^T s_{11}^E} \\ k_{33}^2 &= \frac{d_{33}^2}{\varepsilon_{33}^T s_{33}^E} \\ k_{15}^2 &= \frac{d_{15}^2}{\varepsilon_{11}^T s_{55}^E} \end{aligned} \tag{2.35}$$

In addition, coupling factors k_p for radial modes of thin discs, and k_t for thickness modes of arbitrary shaped thin plates are also commonly given in datasheet. k_p is related to k_{31} through:

$$k_p^2 = \frac{2k_{31}^2}{1 + \frac{s_{12}^E}{s_{11}^2}} \tag{2.36}$$

k_t is always lower than k_{33} but there does not seem to be a simple explicit expression of k_t as a function of the material properties. The fact that k_t is lower than k_{33} means that electrical energy conversion in the d_{33}-mode is less effective for a thin plate than for a rod. The definition of the coupling factors k_{33} and k_{15} also allows to write alternative expressions:

$$\begin{aligned} k_{33}^2 &= 1 - \frac{s_{33}^D}{s_{33}^E} \\ k_{15}^2 &= 1 - \frac{s_{55}^D}{s_{55}^E} = 1 - \frac{\varepsilon_{11}^S}{\varepsilon_{11}^T} \end{aligned} \tag{2.37}$$

We illustrate the use of these different relationships to form the full set of mechanical, piezoelectric and dielectric properties for the material *SONOX P502* from Ceramtec (http://www.ceramtec.com/) which is a soft piezoceramic. The properties found in the datasheet on matweb.com are given in Table 2.2 .

Material property	value	unit
Piezoelectric properties		
d_{33}	440	$10^{-12} m/V$
d_{31}	-185	$10^{-12} m/V$
d_{15}	560	$10^{-12} m/V$
e_{33}	16.7	$C/m^2 = As/m^2$
g_{33}	$26.9\ 10^{-3}$	Vm/N
Permittivity		
ε_{33}^T	$1850\ \varepsilon_0$	F/m
ε_{33}^S	$875\ \varepsilon_0$	F/m
ε_{11}^T	$1950\ \varepsilon_0$	F/m
ε_{11}^S	$1260\ \varepsilon_0$	F/m
Elastic properties		
s_{11}^E	$18.5\ 10^{-12}$	m^2/N
s_{33}^E	$20.7\ 10^{-12}$	m^2/N
c_{33}^D	$15.7\ 10^{10}$	N/m^2
c_{55}^D	$6.5\ 10^{10}$	N/m^2
Coupling coefficients		
k_{33}	0.72	
k_{15}	0.74	
k_{31}	0.33	
k_p	0.62	
k_t	0.48	
Density		
ρ	7740	kg/m^3

Table 2.2: Properties of *SONOX P502* from the datasheet found on https://www.matweb.com (2013)

E_p and E_z are computed directly from the definitions of s_{11}^E and s_{33}^E:

$$E_p = \frac{1}{s_{11}^E} = 54.05GPa \tag{2.38}$$

$$E_z = \frac{1}{s_{33}^E} = 48.31GPa \tag{2.39}$$

Knowing the value of s_{11}^E, d_{31}, ε_{33}^T and k_p, s_{12}^E can be computed:

$$s_{12}^E = -s_{11}^E + 2\frac{d_{31}^2}{k_p^2 \varepsilon_{33}^T} = -7.6288\,10^{-12} m^2/N$$

allowing to compute the value of ν_p:

$$\nu_p = -E_p s_{12}^E = 0.4124$$

and the value of G_p

$$G_p = \frac{E_p}{2(1+\nu_p)} = 19.17GPa$$

From the value c_{55}^D and k_{15}, we compute

$$s_{55}^E = \frac{1}{c_{55}^D(1-k_{15}^2)} = 34\,10^{-12} m^2/N$$

from which the the value of G_{zp} is computed:

$$G_{zp} = \frac{1}{s_{55}^E} = 29.41GPa$$

The value of ν_{zp} cannot be calculated from the datasheet information. We therefore assume that, as for most PZT ceramics:

$$\nu_{zp} = 0.39$$

The value of ν_{pz} is calculated as:

$$\nu_{pz} = \frac{E_p}{E_z}\nu_{zp} = 0.44$$

The complete set of values is summarized in Table 2.3. These are the values used in `m_piezo`.

Material property	value	unit
Piezoelectric properties		
d_{33}	440	$10^{-12}m/V$
d_{31}	-185	$10^{-12}m/V$
d_{15}	560	$10^{-12}m/V$
Permittivity		
ε_{33}^T	1850 ε_0	F/m
ε_{11}^T	1950 ε_0	F/m
Mechanical properties		
E_p	54.05	GPa
E_z	48.31	GPa
G_{zp}	29.41	GPa
G_p	19.17	GPa
ν_p	0.4124	
ν_{zp}	0.39	
ν_{pz}	0.44	
ρ	7740	kg/m^3

Table 2.3: Properties of *SONOX P502* to be used in 3D finite element models

Note that there is some redundancy in the data from the datasheet, which allows to check for consistency. The two following coupling factors are computed from the data available and checked against the tabulated values.

$$k_{31} = \sqrt{\frac{d_{31}^2}{\varepsilon_{33}^T s_{11}^E}} = 0.3361$$

$$k_{33} = \sqrt{\frac{d_{33}^2}{\varepsilon_{33}^T s_{33}^E}} = 0.7556$$

The values are close to the values in Table 2.2. In addition, the value of g_{33} is given by:

$$g_{33} = \frac{d_{33}}{\varepsilon_{33}^T} = 0.0269Vm/N$$

and corresponds exactly to the value tabulated. The value of e_{33} can be computed using Equation (2.7), leading to:

$$e_{33} = 19.06C/m^2$$

where there is a difference of about 15% with the tabulated value of $e_{33} = 16.7C/m^2$. Note however that this last value was found on matweb.com and is not given in the more recent datasheet on

Ceramtec website (in 2023).

Using (2.37) to compute k_{15} with the values from the datasheet, one gets:

$$k_{15} = \sqrt{1 - \frac{\varepsilon_{11}^S}{\varepsilon_{11}^T}} = 0.5948$$

which shows the non-consistency of the value of ε_{11}^S in the datasheet. In fact, when computed using (2.8), one gets:

$$\varepsilon_{11}^S = 908\varepsilon_0$$

This illustrates the fact that it is difficult to obtain the full set of parameters needed for computation for piezoelectric materials, as there are often some inconsistencies amongst the data available from the manufacturers. What we believe is a ”best compromise” was used in the material properties available in SDT.

From the input values in `m_piezo` (Table 2.3), it is possible to compute the mechanical, piezo-electric and permittivity matrices used in the four different forms of the constitutive equations (2.4),(2.5),(2.9),(2.10) using the relationships (2.6)-(2.8)) and (2.11)-(2.13).

The command `p_piezo('TabDD',model)` gives access to all the matrices based in the input values in `m_piezo`. This will be illustrated in section section 3.5.1 .

As the mechanical properties of PZT are not strongly orthotropic, a simplification can be done by considering that the material is isotropic (for the mechanical and dielectric properties, not the piezoelectric properties). An isotropic version of *SONOX P502* is included in `m_piezo` under the name of *SONOX_P502_iso* whose properties are given in Table 2.4.

Material property	value	unit
Piezoelectric properties		
d_{33}	440	$10^{-12} m/V$
d_{31}	-185	$10^{-12} m/V$
d_{15}	560	$10^{-12} m/V$
Permittivity		
ε^T	1850 ε_0	F/m
Mechanical properties		
E	54	GPa
ν	0.41	
ρ	7740	kg/m^3

Table 2.4: Simplified material properties for *SONOX P502* considering mechanical isotropy

The second example is the *PIC 255* PZT, also a soft piezoceramic, from PI ceramics. The properties found in the datasheet in the year 2013 are given in Table 2.5 (https://www.piceramic.com). Note that C_{33}^D was not given, therefore we estimated it from the value of *PIC 155* given in the same datasheet, which is just slightly stiffer.

Material property	value	unit
Piezoelectric properties		
d_{33}	400	$10^{-12} m/V$
d_{31}	-180	$10^{-12} m/V$
d_{15}	550	$10^{-12} m/V$
g_{31}	-11.3 10^{-3}	Vm/N
g_{33}	25 10^{-3}	Vm/N
Permittivity		
ε_{33}^T	1750 ε_0	F/m
ε_{11}^T	1650 ε_0	F/m
Elastic properties		
s_{11}^E	16.1 10^{-12}	m^2/N
s_{33}^E	20.7 10^{-12}	m^2/N
c_{33}^D	11 10^{10}	N/m^2
Coupling coefficients		
k_{33}	0.69	
k_{15}	0.66	
k_{31}	0.35	
k_p	0.62	
k_t	0.47	
Density		
ρ	7800	kg/m^3

Table 2.5: Properties of *PIC 255* from the datasheet (2013)

E_p and E_z are computed directly from the definitions of s_{11}^E and s_{33}^E:

$$E_p = \frac{1}{s_{11}^E} = 62.11\, GPa$$

$$E_z = \frac{1}{s_{33}^E} = 48.31\, GPa$$

Knowing the value of s_{11}^E, d_{31}, ε_{33}^T and k_p, s_{12}^E can be computed:

$$s_{12}^E = -s_{11}^E + 2\frac{d_{31}^2}{k_p^2 \varepsilon_{33}^T} = -5.22\, 10^{-12}\, m^2/N$$

allowing to compute the value of ν_p:

$$\nu_p = -E_p s_{12}^E = 0.3242$$

and the value of G_p

$$G_p = \frac{E_p}{2(1+\nu_p)} = 23.53\,GPa$$

The value of s_{55}^E can be computed as:

$$s_{55}^E = \frac{d_{15}^2}{\varepsilon_{11}^T k_{15}^2} = 4.75\,10^{-11}\,m^2/N$$

which leads to:

$$G_{zp} = \frac{1}{s_{55}^E} = 21.03\,GPa$$

Again, the value of ν_{zp} cannot be calculated from the datasheet information. We cannot assume a value of 0.39 as previously, as it would lead to a non-physical value of ν_{pz}. As ν_p is in the range of 0.32 and ν_{zp} is typically slightly lower, we assume that :

$$\nu_{zp} = 0.30$$

The value of ν_{pz} is calculated as:

$$\nu_{pz} = \frac{E_p}{E_z}\nu_{zp} = 0.39$$

The complete set of values is summarized in Table 2.6. These are the values used in `m_piezo`. Note that there is some redundancy in the data from the datasheet, which allows to check for consistency. The two following coupling factors are computed from the data available and checked against the tabulated values.

$$k_{31} = \sqrt{\frac{d_{31}^2}{\varepsilon_{33}^T s_{11}^E}} = 0.36$$
$$k_{33} = \sqrt{\frac{d_{33}^2}{\varepsilon_{33}^T s_{33}^E}} = 0.70$$

The values are very close to the values in Table 2.5. In addition, the value of g_{33} and g_{31} are given by:

$$g_{31} = \tfrac{d_{31}}{\varepsilon_{33}^T} = -11.6\,10^{-3}\,Vm/N$$
$$g_{33} = \tfrac{d_{33}}{\varepsilon_{33}^T} = 25.8\,10^{-3}\,Vm/N$$

and are also very close to the values tabulated.

Material property	value	unit
Piezoelectric properties		
d_{33}	400	$10^{-12} m/V$
d_{31}	-180	$10^{-12} m/V$
d_{15}	550	$10^{-12} m/V$
Permittivity		
ε_{33}^T	1750 ε_0	F/m
ε_{11}^T	1650 ε_0	F/m
Mechanical properties		
E_p	62.11	GPa
E_z	48.31	GPa
G_{zp}	21.03	GPa
G_p	23.53	GPa
ν_p	0.3242	
ν_{zp}	0.30	
ν_{pz}	0.39	
ρ	7800	kg/m^3

Table 2.6: Properties of *PIC 255* to be used in 3D finite element models from datasheet in 2013

As shown in the derivations above, the datasheet for PZT material typically do not contain the full information to derive all the coefficients needed for computations, and some hypothesis need to be made. In addition, it is usual to have a variation of 10 % or more on these properties from batch to batch, and the datasheet are not updated for each batch. Note also that the properties are given at 20 °C and are temperature dependant. The variations with temperature are rarely given in the datasheet. This may also account for inaccuracies in the computations.

The more recent datasheet found on PI Ceramics website (2023) leads to slightly different properties, and includes the value of C_{33}^D, which gives a more precise value for E_z. The new datasheet information is given in Table 2.7 and the resulting `m_piezo` input parameters in Table 2.8. The updated properties are included in the *PIC255b* material in `m_piezo`. It is advised to use this updated material property, as the main difference is for the Young's modulus in the direction of poling. This parameter has an important impact on the longitudinal natural frequency of disks and rods.

Material property	value	unit
Piezoelectric properties		
d_{33}	400	$10^{-12}m/V$
d_{31}	-180	$10^{-12}m/V$
d_{15}	550	$10^{-12}m/V$
g_{31}	-11.8 10^{-3}	Vm/N
g_{33}	25 10^{-3}	Vm/N
Permittivity		
ε_{33}^T	1800 ε_0	F/m
ε_{11}^T	1750 ε_0	F/m
Elastic properties		
s_{11}^E	16 10^{-12}	m^2/N
s_{33}^E	19 10^{-12}	m^2/N
c_{33}^D	15.4 10^{10}	N/m^2
Coupling coefficients		
k_{33}	0.69	
k_{15}	0.65	
k_{31}	0.35	
k_p	0.62	
k_t	0.47	
Density		
ρ	7800	kg/m^3

Table 2.7: Properties of *PIC 255b* from the datasheet (2023)

Material property	value	unit
Piezoelectric properties		
d_{33}	400	$10^{-12} m/V$
d_{31}	-180	$10^{-12} m/V$
d_{15}	550	$10^{-12} m/V$
Permittivity		
ε_{33}^T	1800 ε_0	F/m
ε_{11}^T	1750 ε_0	F/m
Mechanical properties		
E_p	62.5	GPa
E_z	52.63	GPa
G_{zp}	21.64	GPa
G_p	23.39	GPa
ν_p	0.3389	
ν_{zp}	0.30	
ν_{pz}	0.3562	
ρ	7800	kg/m^3

Table 2.8: Properties of *PIC 255b* to be used in 3D finite element models from datasheet in 2023

In much the same way, the material properties of *PIC 181* which is a hard piezoceramic from the same manufacturer have been updated from the *PIC181* to the *PIC181b* properties in `m_piezo`.

Table 2.9 summarizes the different material properties available in SDT, and the year of the datasheet where the original data was found.

Manufacturer	type	year	SDT name
Ceramtec	Sonox P502	2023	SONOX_P502
Ceramtec	Sonox P502 - simplified	2023	SONOX_P502_iso
PI Ceramics	PIC181	2013	PIC181
PI Ceramics	PIC181	2023	PIC181b
PI Ceramics	PIC255	2013	PIC255
PI Ceramics	PIC255	2023	PIC255b
Ferroperm	Pz21	2018	FerropermPz21
Ferroperm	Pz23	2018	FerropermPz23
Ferroperm	Pz24	2018	FerropermPz24
Ferroperm	Pz26	2018	FerropermPz26
Ferroperm	Pz27	2018	FerropermPz27
Ferroperm	Pz28	2018	FerropermPz28
Ferroperm	Pz29	2018	FerropermPz29
Ferroperm	Pz34	2018	FerropermPz34
Ferroperm	Pz46	2018	FerropermPz46
Noliac	NCE51	2012	Noliac.NCE51

Table 2.9: Piezoelectric materials available in SDT: manufacturer references and year, and SDT name

2.4 Illustration of piezoelectricity in statics: patch example

2.4.1 Patch in extensional mode

Consider a thin piezoelectric patch of dimensions b x h x w. The poling direction, noted 3 in the IEEE Standards on piezoelectricity is perpendicular to the plane of the piezoelectric patch. Continuous electrodes are present on the top and bottom surfaces ($z = 0$, $z = h$) so that the electric potential is constant on these surfaces and denoted by V_1 and V_2 respectively. We assume that a difference of potential is applied between the electrodes, resulting in an electric field parallel to the poling direction and equal to (Figure 2.5)

$$E_3 = -\frac{dV}{dz} = \frac{-(V_2 - V_1)}{h} = \frac{V_1 - V_2}{h}$$

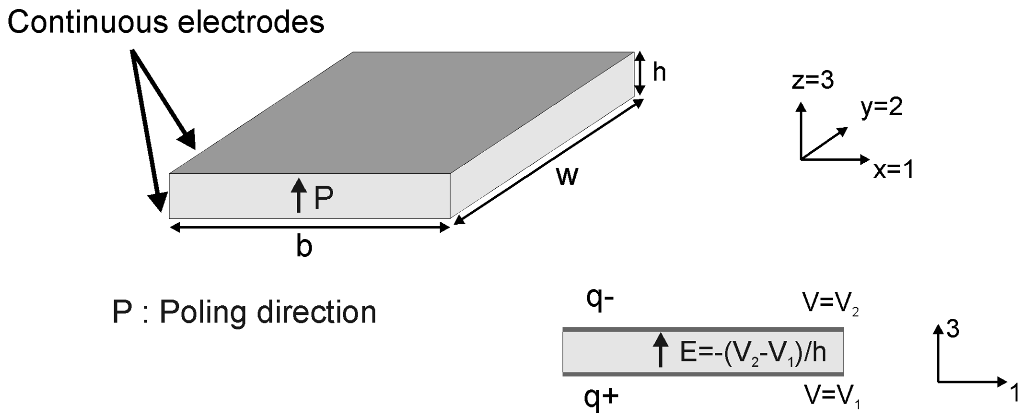


Figure 2.5: A piezoelectric patch poled through the thickness with continuous electrodes on the top and bottom surfaces

We adopt the following expression for the constitutive equations:

$$\begin{aligned} \{S\} &= \left[s^E\right]\{T\} + [d]^T\{E\} \\ \{D\} &= [d]\{T\} + \left[\varepsilon^T\right]\{E\} \end{aligned} \tag{2.40}$$

The patch is assumed to be unconstrained so that it can expand freely, leading to $\{T\} = 0$, so that we have :

$$\{S\} = \left\{\begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{array}\right\} = [d]^T\{E\} = \left\{\begin{array}{c} d_{31}\frac{V_1-V_2}{h} \\ d_{32}\frac{V_1-V_2}{h} \\ d_{33}\frac{V_1-V_2}{h} \\ 0 \\ 0 \\ 0 \end{array}\right\} \tag{2.41}$$

We have taken into account the fact that the electric field is in the z-direction only. This shows that when applying a difference of potential across the thickness (in the poling direction), strains will be induced in the directions 1,2, and 3. The magnitude of these different strains is proportional to the d_{3i} coefficients of the piezoelectric material. For a ceramic PZT material, $d_{31} = d_{32} < 0$, and $d_{33} > 0$ and is generally between 2 and 3 times larger in magnitude than d_{31} and d_{32}.

The second equation can be used in order to assess the amount of charge that is accumulated on both electrodes. We have :

$$\{D\} = \left\{\begin{array}{c} D_1 \\ D_2 \\ D_3 \end{array}\right\} = \left[\varepsilon^T\right]\{E\} \tag{2.42}$$

The only non-zero component of the D vector is D_3 given by :

$$D_3 = \varepsilon_{33}^T \frac{V_1 - V_2}{h} \tag{2.43}$$

The charge accumulated on the electrode is given by :

$$q = -\int_S \{D\}\{n\}\, dS$$

where $\{n\}$ is the normal to the electrode. For the top electrode, this leads to :

$$q = -\frac{\varepsilon_{33}^T A}{h}(V_1 - V_2)$$

where A is the surface of the electrode. For the bottom electrode

$$q = \frac{\varepsilon_{33}^T A}{h}(V_1 - V_2)$$

When $(V_1 - V_2)$ is positive, the electric field is in the direction of poling and the charge on the top electrode is negative, while the charge accumulated on the bottom electrode is positive (Figure 2.5). Note that this equation corresponds to the equation linking the charge to the difference of potential for a capacitor ($q = C\Delta V$). The value of the capacitance is therefore :

$$C^T = \frac{\varepsilon_{33}^T A}{h}$$

which corresponds to the capacitance of the free piezoelectric patch ($\{T\} = 0$).

If we now consider the case where the piezoelectric patch is fully mechanically constrained ($\{S\} = 0$), we have:

$$\begin{array}{l} \{T\} = -[e]^T\{E\} = -[e]^T\{E\} \\ \{D\} = \left[\varepsilon^S\right]\{E\} \end{array} \tag{2.44}$$

leading to :

$$\{T\} = \left\{\begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{array}\right\} = \left\{\begin{array}{c} -e_{31}\frac{V_1-V_2}{h} \\ -e_{32}\frac{V_1-V_2}{h} \\ -e_{33}\frac{V_1-V_2}{h} \\ 0 \\ 0 \\ 0 \end{array}\right\} \tag{2.45}$$

$$D_3 = \varepsilon_{33}^S \frac{V_1-V_2}{h}$$

The fact that the patch is not allowed to expand is responsible for the generation of internal stresses which are proportionnal to the e_{3i} coefficients. In this case, the capacitance is given by:

$$C^S = \frac{\varepsilon_{33}^S A}{h}$$

which corresponds to the capacitance of the constrained piezoelectric patch ($\{S\} = 0$). This illustrates the fact that the capacitance of a piezoelectric patch depends on the mechanical boundary conditions. This is not the case for other types of dielectric materials in which the piezoelectric effect is not present, and for which therefore the capacitance is independent on the mechanical strain or stress.

2.4.2 Patch in shear mode

We now consider the same patch but where the polarization is in the plane of the actuator, as represented in Figure 2.6. As in the previous example, continuous electrodes are present on the top and bottom surfaces ($z = 0$, $z = h$) so that the electric potential is constant on these surfaces and denoted by V_1 and V_2 respectively. We assume that a difference of potential is applied between the electrodes, resulting in an electric field perpendicular to the poling direction and equal to

$$E_2 = -\frac{dV}{dz} = \frac{-(V_2 - V_1)}{h} = \frac{V_1 - V_2}{h}$$

The electric field is now applied in direction 2, so that it will activate the shear $d_{24} = d_{15}$ mode of the piezoelectric material.

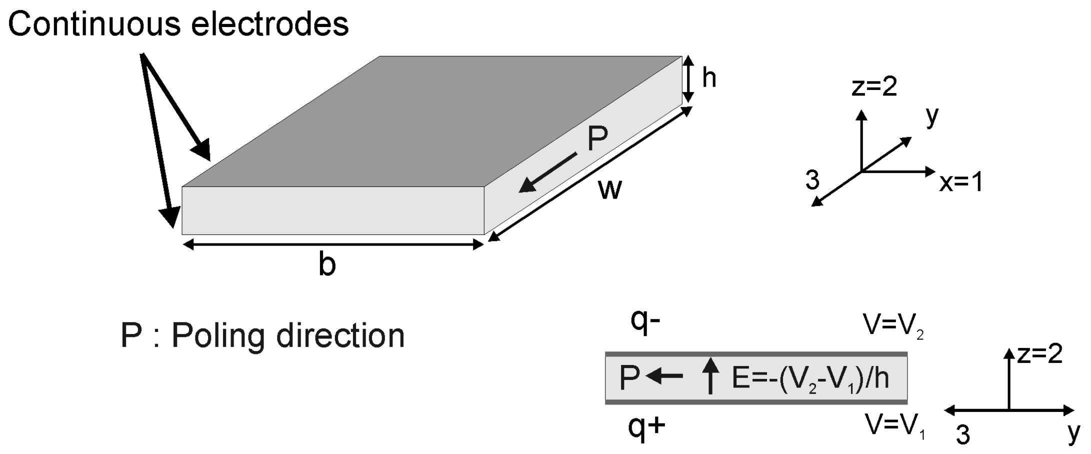


Figure 2.6: A piezoelectric patch poled in the plane with continuous electrodes on the top and bottom surfaces

The patch is assumed to be unconstrained so that it can expand freely, leading to $\{T\} = 0$, so that we have :

$$\{S\} = \left\{ \begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{array} \right\} = [d]^T \{E\} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ d_{24}\frac{V_1 - V_2}{h} \\ 0 \\ 0 \end{array} \right\} \tag{2.46}$$

We have taken into account the fact that the electric field is in the z-direction only, corresponding to direction 2 in the local axis of the piezoelectric material (direction 3 is the poling direction by convention). This shows that when the patch is poled in the plane, when applying a difference of potential across the thickness, a shear strain in the local 23 plane will be induced. The magnitude of this strain is proportional to the d_{24} coefficient of the piezoelectric material.

The second equation can be used in order to assess the amount of charge that is accumulated on both electrodes. We have :

$$\{D\} = \left\{ \begin{array}{c} D_1 \\ D_2 \\ D_3 \end{array} \right\} = \left[\varepsilon^T\right] \{E\} \tag{2.47}$$

The only non-zero component of the D vector is D_2 given by :

$$D_2 = \varepsilon_{22}^T \frac{V_1 - V_2}{h} \tag{2.48}$$

The charge accumulated on the electrode is given by :

$$q = -\int_S \{D\} \{n\} \, dS$$

where $\{n\}$ is the normal to the electrode. For the top electrode, this leads to :

$$q = -\frac{\varepsilon_{22}^T A}{h}(V_1 - V_2)$$

where A is the surface of the electrode. For the bottom electrode

$$q = \frac{\varepsilon_{22}^T A}{h}(V_1 - V_2)$$

When $(V_1 - V_2)$ is positive, the charge on the top electrode is negative, while the charge accumulated on the bottom electrode is positive (Figure 2.6). The value of the capacitance is therefore:

$$C^T = \frac{\varepsilon_{22}^T A}{h}$$

which corresponds to the capacitance of the free piezoelectric patch ($\{T\} = 0$) and is close to the capacitance when the poling is out of the plane of the transducer because $\varepsilon_{22}^T \simeq \varepsilon_{33}^T$ (in reality, there is typically a difference of 5% between these two values so that the capacitance will be slightly different).

If we now consider the case where the piezoelectric patch is fully mechanically constrained ($\{S\} = 0$), we have:

$$\begin{gathered} \{T\} = -\left[e\right]^T\{E\} = -\left[e\right]^T\{E\} \\ \{D\} = \left[\varepsilon^S\right]\{E\} \end{gathered} \tag{2.49}$$

leading to :

$$\{T\} = \left\{ \begin{array}{c} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{array} \right\} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ -e_{24}\frac{V_1-V_2}{h} \\ 0 \\ 0 \end{array} \right\} \tag{2.50}$$

$$D_2 = \varepsilon_{22}^S \frac{V_1-V_2}{h}$$

In this case, the capacitance is given by:

$$C^S = \frac{\varepsilon_{22}^S A}{h}$$

which corresponds to the capacitance of the constrained piezoelectric patch activated in shear ($\{S\} = 0$). Note that this capacitance is clearly different from C^S when the poling is out of the plane, because the value of ε_{22}^S is very different from the value of ε_{33}^S, due to the different values of stiffness and piezoelectric coefficients in shear and extensional mode.

3

Finite element formulations for piezoelectric structures

Contents

Hamilton's principle is used to derive the dynamic variational principle [1]:

$$\begin{aligned}\int_{t_1}^{t_2} \Bigg(\int_V \Big[-\rho \{\ddot{u}\}^T \{\delta u\} - \{S\}^T \left[c^E\right] \{\delta S\} + \{E\}^T [e] \{\delta S\} \\ + \{S\}^T [e]^T \{\delta E\} + \{E\}^T \left[\varepsilon^S\right] \{\delta E\} + \{f\}^T \{\delta u\} - \{\rho_e\}^T \{\delta \phi\} \Big] \, dV \\ + \int_{\Omega_1} \{t\}^T \{\delta u\} \, d\Omega - \int_{\Omega_2} \{\sigma\}^T \{\delta \phi\} \, d\Omega \Bigg) \, dt = 0\end{aligned} \tag{3.1}$$

where V is the volume of the piezoelectric structure, ρ is the mass density, $\{u\}$ is the displacement field and $\{\delta u\}$ its variation, $\{\phi\}$ is the elecctric pontential and $\{\delta\phi\}$ its variation. $\{f\}$ is the volumic force, $\{\rho_e\}$ the volumic charge density, $\{t\}$ the vector of applied surface forces on Ω_1 and $\{\sigma\}$ the charge density applied on Ω_2. The variational principle is the starting point for all discrete finite element formulations. 3D and shell approximations are detailed below.

3.1 Piezoelectric solid finite elements

For 3D solids, the discretized strain and electric fields are linked to the discretized displacement vector (u, v, w) and electric potential ϕ by:

$$\left\{ \begin{array}{c} S \\ E \end{array} \right\} = \left\{ \begin{array}{c} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{yz} \\ \gamma_{zx} \\ \gamma_{xy} \\ E_x \\ E_y \\ E_z \end{array} \right\} = \left[\begin{array}{cccc} N,x & 0 & 0 & 0 \\ 0 & N,y & 0 & 0 \\ 0 & 0 & N,z & 0 \\ 0 & N,z & N,y & 0 \\ N,z & 0 & N,x & 0 \\ N,y & N,x & 0 & 0 \\ 0 & 0 & 0 & -N,x \\ 0 & 0 & 0 & -N,y \\ 0 & 0 & 0 & -N,z \end{array} \right] \left\{ \begin{array}{c} u \\ v \\ w \\ \phi \end{array} \right\} \tag{3.2}$$

where $N,x\ u$ is a short notation for

$$\sum_i \frac{\partial N_i}{\partial x} u_i$$

and $N_i(x, y, z)$ are the finite element shape functions. Plugging (3.2) in (3.1) leads to the discrete set of equations which are written in the matrix form:

$$\left[\begin{array}{cc} M_{qq} & 0 \\ 0 & 0 \end{array} \right] \left\{ \begin{array}{c} \ddot{q_{mech}} \\ \ddot{V} \end{array} \right\} + \left[\begin{array}{cc} K_{qq} & K_{qV} \\ K_{Vq} & K_{VV} \end{array} \right] \left\{ \begin{array}{c} q_{mech} \\ V \end{array} \right\} = \left\{ \begin{array}{c} F_{mech} \\ Q \end{array} \right\} \tag{3.3}$$

where $\{q_{mech}\}$ contains the mechanical degrees of freedom (3 per node related to u, v, w), and $\{V\}$ contains the electrical degrees of freedom (1 per node, the electric potential ϕ). $\{F_{mech}\}$ is the vector of applied external mechanical forces, and $\{Q\}$ is the vector of applied external charges.

3.2 Piezoelectric shell finite elements

Shell strain is defined by the membrane, curvature and transverse shear as well as the electric field components. In the piezoelectric multi-layer shell elements implemented in SDT, it is assumed that in each piezoelectric layer $i = 1...n$, the electric field takes the form $\vec{E} = (0 \quad 0 \quad E_{zi})$. E_{zi} is assumed to be constant over the thickness h_i of the layer and is therefore given by $E_{zi} = -\frac{\Delta\phi_i}{h_i}$ where $\Delta\phi_i$ is the difference of potential between the electrodes at the top and bottom of the piezoelectric layer i. It is also assumed that the piezoelectric principal axes are parallel to the structural orthotropy axes.

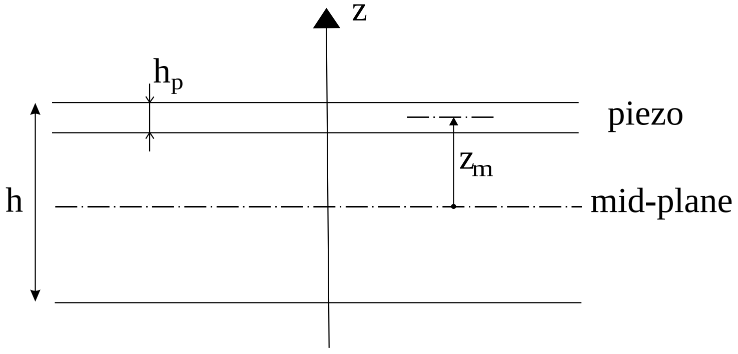


Figure 3.1: Multi-layer shell piezoelectric element

The discretized strain and electric fields of a piezoelectric shell take the form

$$\left\{\begin{array}{c} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \\ \kappa_{xx} \\ \kappa_{yy} \\ 2\kappa_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \\ -E_{z1} \\ ... \\ -E_{zn} \end{array}\right\} = \begin{bmatrix} N,x & 0 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & N,y & 0 & 0 & 0 & 0 & ... & 0 \\ N,y & N,x & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & -N,x & 0 & ... & 0 \\ 0 & 0 & 0 & N,y & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & N,x & -N,y & 0 & ... & 0 \\ 0 & 0 & N,x & 0 & N & 0 & ... & 0 \\ 0 & 0 & N,y & -N & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_1} & ... & 0 \\ ... & ... & ... & ... & ... & 0 & ... & -\frac{1}{h_n} \end{bmatrix} \left\{\begin{array}{c} u \\ v \\ w \\ ru \\ rv \\ \Delta\phi_1 \\ ... \\ \Delta\phi_n \end{array}\right\} \tag{3.4}$$

There are thus n additional degrees of freedom $\Delta\phi_i$, n being the number of piezoelectric layers in the laminate shell. The constitutive laws are obtained by using the "piezoelectric plates" hypothesis (2.16) and the definitions of the generalized forces N, M, Q and strains $\varepsilon, \kappa, \gamma$ for shells:

$$\left\{\begin{array}{c} N \\ M \\ Q \\ D_{z1} \\ ... \\ D_{zn} \end{array}\right\} = \left[\begin{array}{cccccc} A & B & 0 & G_1^T & ... & G_n^T \\ B & D & 0 & z_{m1}G_1^T & ... & z_{mn}G_n^T \\ 0 & 0 & F & 0 & ... & 0 \\ G_1 & z_{m1}G_1 & 0 & -{\varepsilon_1}^S & ... & 0 \\ ... & ... & ... & 0 & ... & 0 \\ G_n & z_{mn}G_n & 0 & 0 & ... & -{\varepsilon_n}^S \end{array}\right] \left\{\begin{array}{c} \epsilon \\ \kappa \\ \gamma \\ -E_{z1} \\ ... \\ -E_{zn} \end{array}\right\} \tag{3.5}$$

D_{zi} is the electric displacement in piezoelectric layer , z_{mi} is the distance between the midplane of the shell and the midplane of piezoelectric layer i (Figure 3.1), G_i is given by

$$G_i = \left\{\begin{array}{ccc} e_{31}^* & e_{32}^* & 0 \end{array}\right\}_i [R_s]_i \tag{3.6}$$

where * refers to the piezoelectric properties under the piezoelectric plate assumption as detailed in section 2.2 and $[R_s]_i$ are rotation matrices associated to the angle θ of the principal axes $1, 2$ of the piezoelectric layer given by:

$$[R_s] = \left[\begin{array}{ccc} \cos^2\theta & \sin^2\theta & \sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -\sin\theta\cos\theta \\ -2\sin\theta\cos\theta & 2\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{array}\right] \tag{3.7}$$

Plugging (3.4) into (3.1) leads again to:

$$\left[\begin{array}{cc} M_{qq} & 0 \\ 0 & 0 \end{array}\right]\left\{\begin{array}{c} \ddot{q}_{mech} \\ \ddot{V} \end{array}\right\} + \left[\begin{array}{cc} K_{qq} & K_{qV} \\ K_{Vq} & K_{VV} \end{array}\right]\left\{\begin{array}{c} q_{mech} \\ V \end{array}\right\} = \left\{\begin{array}{c} F_{mech} \\ Q \end{array}\right\} \tag{3.8}$$

where $\{q_{mech}\}$ contains the mechanical degrees of freedom (5 per node corresponding to the displacements u, v, w and rotations rx, ry), and $\{V\}$ contains the electrical degrees of freedom. The electrical dofs are defined at the element level, and there are as many as there are active layers in the laminate. Note that the electrical degree of freedom is the difference of the electric potential between the top and bottom electrodes $\Delta\phi$.

3.3 Full order model

Piezoelectric models are described using both mechanical q_{mech} and electric potential DOF V. As detailed in sections section 3.1 and section 3.2 , one obtains models of the form

$$\left[\begin{array}{cc} Z_{qq}(s) & Z_{qV} \\ Z_{Vq} & Z_{VV} \end{array}\right]\left\{\begin{array}{c} q_{mech} \\ V \end{array}\right\} = \left\{\begin{array}{c} F_{mech} \\ Q \end{array}\right\} \tag{3.9}$$

for both piezoelectric solids and shells, where $Z_{qq}(s)$ is the dynamic (mechanical) stiffness expressed as a function of the Laplace variable s.

For piezoelectric shell elements, electric DOFs correspond to the difference of potential on the electrodes of one layer, while the corresponding load is the charge Q. In SDT, the electric DOFs for shells are unique for a single shell property and are thus giving an implicit definition of electrodes (see `p_piezo Shell`). Note that a common error is to fix all DOF when seeking to fix mechanical DOFs, calls of the form `'x==0 -DOF 1:6'` avoid this error.

For volume elements, each volume node is associated with an electric potential DOF and one defines multiple point constraints to enforce equal potential on nodes linked by a single electrode and sets one of the electrodes to zero potential (see `p_piezo ElectrodeMPC` and section **??** for a tutorial on how to set these contraints). During assembly the constraints are eliminated and the resulting model has electrical DOFs that correspond to potential, or differences of potential (if the other electrode's potential is set to 0) and loads to charge.

Short circuit (SC), charge sensors configurations correspond to cases where the potential is forced to zero (the electrical circuit is shorted). In (3.9), this corresponds to a case where the potential (electrical DOF) is fixed and the charge corresponds to the resulting force associated with this boundary condition.

A **voltage actuator** corresponds to the same problem with $V = V_{In}$ (built in SDT using `fe_load ('DofSet')` entries). The closed circuit charge is associated with the constraint on the enforced voltage and can be computed by extracting the second row of (3.9)

$$\{Q\} = [Z_{qC}]\,\{q_{mech}\} + [Z_{VV}]\,\{V_{In}\} \tag{3.10}$$

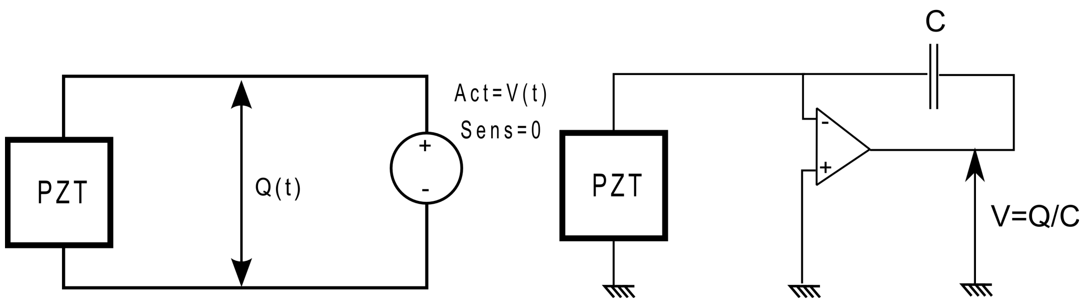


Figure 3.2: Short circuit: voltage actuator, charge sensor

`p_piezo ElectrodeSensQ` provides utilities to build the charge sensors, including sensor combinations.

SC is the only possibly boundary condition to impose in a FEM model where voltage is the unknown. The alternative is to leave the potential free which corresponds to not specifying any boundary condition.

Open circuit (OC), voltage sensor, configurations correspond to cases where the charge remains zero and a potential is created on the electrodes due to mechanical deformations.

A **piezoelectric actuator driven using a charge source** also would correspond to this configuration (but the usual is voltage driving).

The voltage DOFs $\{V\}$ associated to open-circuits are left free in (3.9). Since electrostatics are normally considered, Z_{VV} is actually frequency independent and the voltage DOFs could be condensed exactly

$$\{V\} = [Z_{VV}]^{-1} (Q_{in} - [Z_{Vq}] \{q_{mech}\}) \tag{3.11}$$

This configuration is to be used for a voltage sensor, for example when the piezoelectric transducer is attached directly to the data acquisition card or a voltage amplifier (with very large impedance for sensing). In both cases the impedance is very large leading to a configuration close to an open circuit (OC, infinite impedance). Another example of OC boundary conditions is the use of current (charge) amplifiers for actuation, which is rarely used in practice but possible.

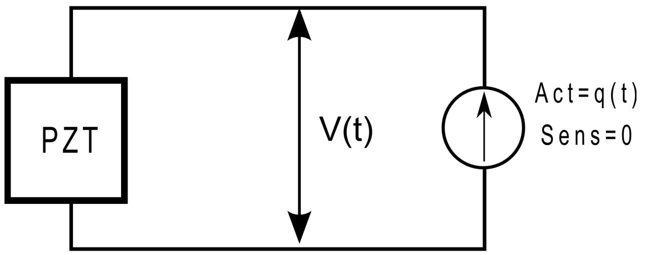


Figure 3.3: Open circuit (voltage sensor, charge actuator)

Since voltage is an explicit DOF, it can be observed using `fe_case('SensDof')` sensor entries. Similarly charge is dual to the voltage, so a charge input would be a simple point load on the active DOF associated to an electrode. Note that specifying a charge distribution does not make sense since you cannot both enforce the equipotential condition and specify a charge distribution that results from this constraint.

It is possible to observe charge in an OC condition, but this is of little interest since this charge will remain at 0.

In summary, when computing modes under voltage actuation, the proper boundary condition is a SC, while for current (charge) actuation, it would be OC. For sensing, a voltage sensor corresponds to OC, while a charge sensor requires SC.

3.4 Using the Electrode stack entry

SDT 6.6 underwent significant revisions to get rid of solver strategies that were specific to piezo applications. The `info,Electrodes` of earlier releases is thus no longer necessary. To avoid disruption of user procedures, you can still use the old format with a `.ver=0` field.

`p_piezo` `ElectrodeInit` is used to build/verify a data structure describing master electric DOFs associated with electrodes defined in your model. The `info,Electrode` stack entry is a structure with fields

`.data` rows `NodeId IsOpen` gives the electrode nodes and for each one `1` if the circuit is open (voltage free), and `0` if it is closed (voltage enforced or fixed, actuator).

`.ver=1` is used to specify that the more general piezoelectric strategies of SDT >= 6.6 are used. This is the combined with the `p_piezo` `Electrode2Case` command which builds piezo loads and sensors. For SDT 6.5 strategies, use `.ver=0`.

`.def` `.DOF`, `.lab_in` **only needed** when combining multiple electrodes into a single input. The `.lab_in` is a cell array of strings, you should end the string with `V` so that it shows Q for associated charge sensors.

Each column gives the weighting coefficients associated with each electrode. Thus `def=[1;0;1]` corresponds to a single equal input on electrodes 1 and 3. Note that it does not make sense to combine electrical DOFs that are of mixed nature (actuator/sensor).

The `.DOF` field should contain `NodeId+.21` since the potential corresponds to DOF `.21`.

The `.lab_in` field can be used to provide labels associated with each actuator/sensor defined as a column of def. You should end the label with `V` so that the collocated sensor ends with a `Q` label.

`.cta` `.lab` (optional) can be used to combine electrodes into sensors / actuators. Each row of `.cta` defines a sensor (with matching `.lab`). Each column corresponds to an electrode declared in the `.data` field. You cannot combine open and closed circuit electrodes. It is possible to use both a `.cta` and a `.def` field.

`[model,data]=p_piezo('ElectrodeInit',model);` generates a default value for the electrode stack entry. Combination of actuators and sensors (both charge and voltage) is illustrated in section section **??** .

3.5 Example 1 : Static response of a piezoelectric patch

3.5.1 Static response of a patch in extension mode

In this very simple example, the electric field and the strains are all constant, so that the electric potential and the displacement field are linear. The example is treated analytically in section section 2.4 . It is therefore possible to obtain an exact solution using a single volumic 8-node finite element (with linear shape functions, the nodal unknowns being the displacements in x,y and z and the electric potential ϕ). Consider a piezoelectric patch whose dimensions and material properties are given in Table 3.1. The material properties correspond to the material *SONOX_P502_iso* in `m_piezo`.

Property	Value
b	10 mm
w	10 mm
h	2 mm
E	54 GPa
ν	0.41
$d_{31} = d_{32}$	-185 $10^{-12} pC/N$ (or m/V)
d_{33}	440 $10^{-12} pC/N$ (or m/V)
$d_{15} = d_{24}$	560 $10^{-12} pC/N$ (or m/V)
$\varepsilon^T_{33} = \varepsilon^T_{22} = \varepsilon^T_{11}$	1850 ε_0
ε_0	8.854 $10^{-12} Fm^{-1}$

Table 3.1: Geometrical and material properties of the piezoelectric patch

We first produce the mesh, associate the material properties and define the electrodes with `d_piezo('TutoPatch-s1')` . The default material is *SONOX_P502_iso*. The number of elements in the x, y and z directions are given by n_x,n_y and n_z.

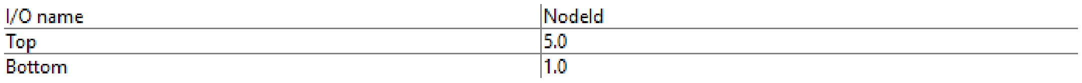
```
% See full example as MATLAB code in d_piezo('ScriptTutoPatch')
d_piezo('DefineStyles');

%% Step 1 Build mesh - Define electrodes
% Meshing script can be viewed with sdtweb d_piezo('MeshPatch')
```

```
model=d_piezo('MeshPatch lx=1e-2 ly=1e-2 h=2e-3 nx=1 ny=1 nz=1');
% Define electrodes
model=p_piezo('ElectrodeMPC Top -ground',model,'z==2e-3');
model=p_piezo('ElectrodeMPC Bottom -Input "Free patch"',model,'z==0');
```

The information about the nodes associated to each electrode can be obtained through the following call (Figure 3.4):

```
p_piezo('TabInfo',model)
```

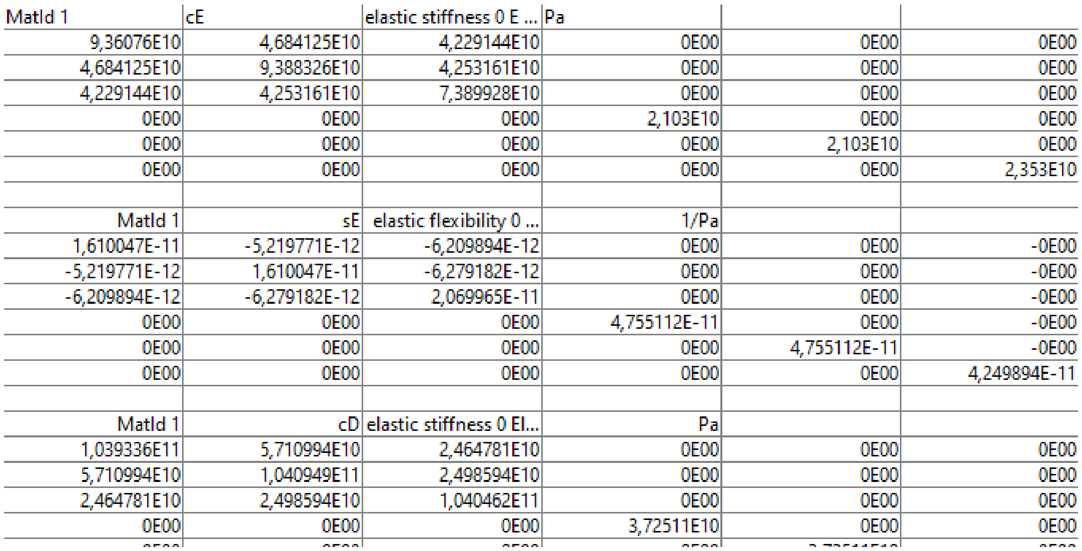


I/O name	NodeId
Top	5.0
Bottom	1.0

Figure 3.4: Tabinfo gives information about nodes associated to electrodes

The material can be changed for example to PIC_255 with the following call, and the full set of mechanical, piezoelectric and permittivity matrices can be obtained in order to check consistency with the datasheet (Figure 3.5). (`d_piezo('TutoPatch-s2')`):

```
%% Step 2 Define material properties
model.pl=m_piezo('dbval 1 -elas 2 PIC_255');
p_piezo('TabDD',model) % creates the table with full set of matrices
```



MatId 1	cE	elastic stiffness 0 E ...	Pa		
9,36076E10	4,684125E10	4,229144E10	0E00	0E00	0E00
4,684125E10	9,388326E10	4,253161E10	0E00	0E00	0E00
4,229144E10	4,253161E10	7,389928E10	0E00	0E00	0E00
0E00	0E00	0E00	2,103E10	0E00	0E00
0E00	0E00	0E00	0E00	2,103E10	0E00
0E00	0E00	0E00	0E00	0E00	2,353E10
MatId 1	sE	elastic flexibility 0 ...	1/Pa		
1,610047E-11	-5,219771E-12	-6,209894E-12	0E00	0E00	-0E00
-5,219771E-12	1,610047E-11	-6,279182E-12	0E00	0E00	-0E00
-6,209894E-12	-6,279182E-12	2,069965E-11	0E00	0E00	-0E00
0E00	0E00	0E00	4,755112E-11	0E00	-0E00
0E00	0E00	0E00	0E00	4,755112E-11	-0E00
0E00	0E00	0E00	0E00	0E00	4,249894E-11
MatId 1	cD	elastic stiffness 0 El...	Pa		
1,039336E11	5,710994E10	2,464781E10	0E00	0E00	0E00
5,710994E10	1,040949E11	2,498594E10	0E00	0E00	0E00
2,464781E10	2,498594E10	1,040462E11	0E00	0E00	0E00
0E00	0E00	0E00	3,72511E10	0E00	0E00

Figure 3.5: Example subset of table with the full set of mechanical, dielectric and piezoelectric coefficients in the 4 different forms of the constitutive equations

The next step consists in defining the boundary conditions and load case using d_piezo('TutoPatch-s3') . We consider here two cases, the first one where the patch is free to expand, and the second one where it is mechanically constrained (all mechanical degrees of freedom are equal to 0).

```
%% Step 3 Compute static response
% to avoid rigid body mode
model=stack_set(model,'info','Freq',10);
def=fe_simul('dfrf',model); def.lab={'Free patch, axial'};
def.fun=[0 1]; def=feutil('rmfield',def,'data','LabFcn');

% Append mechanically constrained structure
%    can't call fe_simul because no free DOF
%    see code with sdtweb d_piezo('scriptFullConstrain')
def=d_piezo('scriptFullConstrain',model,def);
def.lab{2}='Constrained patch, axial';
```

We can look at the deformed shape, and plot the electric field for both cases. (d_piezo('TutoPatch-s4')

```
%% Step 4 Visualize deformed shape
cf=feplot(model,def);
% Electric field representation
p_piezo('viewElec EltSel "matid1" DefLen 20e-4 reset',cf);
fecom('colormap',[1 0 0]);fecom('undef line');iimouse('resetview');
cf.mdl.name='E-field'; % Sets figure name
d_piezo('SetStyle',cf); feplot(cf);
```

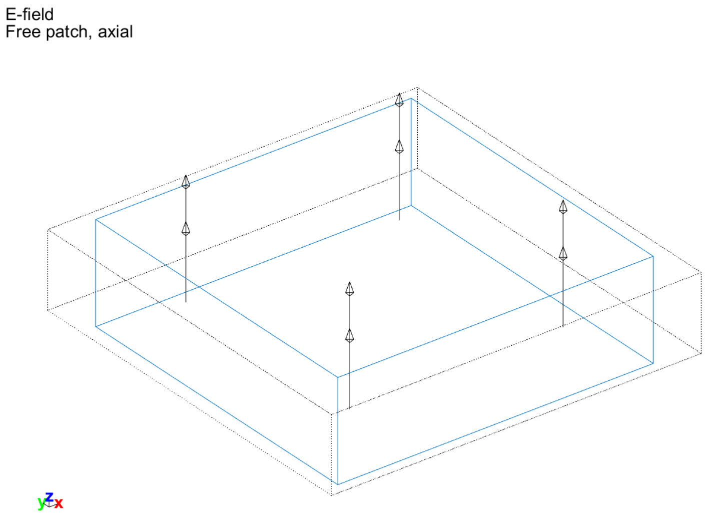


Figure 3.6: Vizualisation of the electric field and deformed shape for the free patch under unit voltage excitation

For the free patch deformed shape, we compute the mean strains from which d_{31}, d_{32} and d_{33} are deduced. The values are found to be equal to the analytical values used in the model. Note that the parameters of the constitutive equations can be recovered using (`d_piezo('TutoPatch-s5')`):

```
%% Step 5 : check constitutive law
% Decompose constitutive law
CC=p_piezo('viewdd -struct',cf); %
```

where the fields of CC are self-explanatory. The parameters which are not directly defined are computed from the equations presented in Section section 2.1 .

```
% Display and compute mean strains
a=p_piezo('viewstrain -curve -mean',cf); % Strain S
```

```
fprintf('Relation between mean strain on free structure and d_3i\n');
E3=a.Y(9,1); disp({'E3 mean' a.Y(9,1)  1/2e-3 'E3 analytic'})

disp([a.X{1}(1:3) num2cell([a.Y(1:3,1)/E3 CC.d(3,1:3)']) ...
     {'d_31';'d_32';'d_33'}])
```

For the constrained patch, we compute the mean stress from which we can compute the e_{31}, e_{32} and e_{33} values which are found to be equal to the analytical values used in the model:

```
% Display and compute mean stresses
b=p_piezo('viewstress -curve -mean',cf); % Stress T
fprintf('Relation between mean stress on pure electric and e_3i\n');
disp([b.X{1}(1:3) num2cell([b.Y(1:3,2)/-E3 CC.e(3,1:3)']) ...
     {'e_31';'e_32';'e_33'}])

% Mean stress/strain
disp([b.X{1} num2cell(b.Y(:,2)) num2cell(a.Y(:,1)) a.X{1}])
```

We can also compute the charge and the charge density (in pC/m^2) accumulated on the electrodes, and compare with the analytical values (`d_piezo('TutoPatch-s6')`):

```
%% Step 6 Check capacitance values
% Theoretical values of Capacitance and charge density - free patch
CT=CC.epst_r(3,3)*8.854e-12*1e-2*1e-2/2e-3; %% Capacitance - free patch
CdensT=CC.epst_r(3,3)*8.854e-12/2e-3*1e12; %% charge density - free patch

% Theoretical values of Capacitance and charge density - constrained patch
CS=CC.epss_r(3,3)*8.854e-12*1e-2*1e-2/2e-3; %% Capacitance - free patch
CdensS=CC.epss_r(3,3)*8.854e-12/2e-3*1e12; %% charge density - free patch

% Represent charge density (C/S) value on the electrodes
%   - compare with analytical values
cut=p_piezo('electrodeviewcharge',cf,struct('EltSel','matid 1'));
b=fe_caseg('stressobserve',cut,cf.def);b=reshape(b.Y,[],2);
disp([{'','CdensT','CdensS'};{'Numeric';'Theoretical'} ...
  num2cell([mean(abs(b));CdensT CdensS])])
  iimouse('zoom reset');

% Compute the value of the total charge (from reaction at electrical dof)
% Compare with analytical values
p_piezo('electrodeTotal',cf) %
```

```
disp('Theoretical values of capacitance')
disp([{'CT';'CS'} num2cell([CT;CS])])
cf.mdl.name='Charge'; % Sets figure name

d_piezo('SetStyle',cf); feplot(cf);
fecom('ch 1')
```

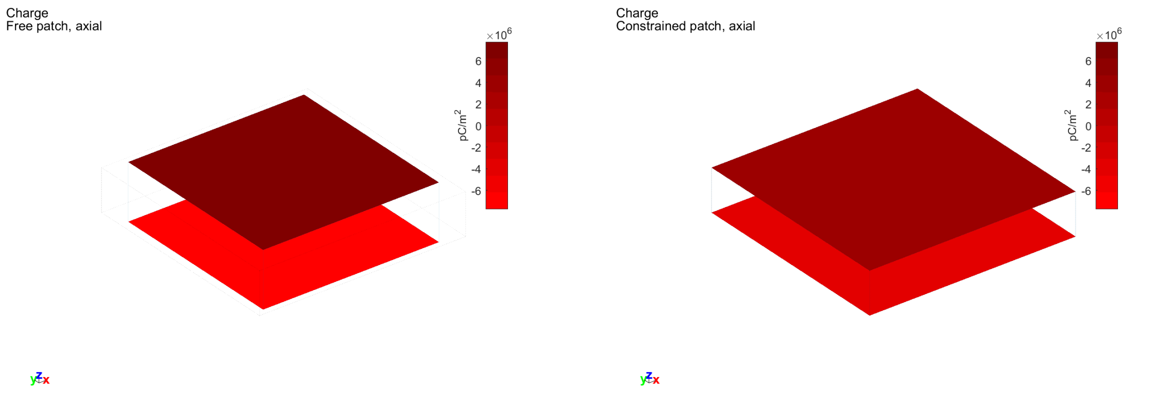


Figure 3.7: Vizualisation of the total charge on the electrodes for the unconstrained and constrained patch under unit voltage excitation

The results clearly show the very large difference of charge density between the two cases (free patch or constrained patch).

For this simple static example, a finer mesh can be used, but it does not lead to more accurate results (this can be done by changing the values in the call to `d_piezo('mesh')` for example:

```
% Build mesh with refinement
model=d_piezo('MeshPatch lx=1e-2 ly=1e-2 h=2e-3 nx=5 ny=5 nz=2');
% Now a model with quadratic elements
model=d_piezo('MeshPatch lx=1e-2 ly=1e-2 h=2e-3 Quad');
```

3.5.2 Static response of a patch in shear mode

As for the patch in extension, as the fields are also uniform (see section 2.4.2), the problem can be modelled with a single 8-node element. The patch is meshed and then the poling is aligned with the $-y$ axis by performing a rotation of 90^o around the x-axis (`d_piezo('TutoPatchShear-s1')`).

```
d_piezo('DefineStyles');

% See full example as MATLAB code in d_piezo('ScriptTutoShearPatch')
%% Step 1 Build mesh and define electrodes
%Meshing script can be viewed with sdtweb d_piezo('MeshPatch')
model=d_piezo('MeshPatch lx=1e-2 ly=1e-2 h=2e-3 nx=1 ny=1 nz=1');

% Define electrodes
model=p_piezo('ElectrodeMPC Top -ground',model,'z==2e-3');
model=p_piezo('ElectrodeMPC Bottom -Input "Free patch"',model,'z==0');

% Rotate basis to align poling direction with y (-90 deg around x)
model.bas=basis('rotate',[],'rx=-90',1); %create local basis with id=1
model=feutil('setpro 1 COORDM=1',model); % assign basis with id=1 to pro=1
```

Then the response is computed both for the free case and the fully constrained case. The deformed shape for the free case is shown in Figure 3.8 together with the applied electric field: (d_piezo('TutoShearPatch-s2'))

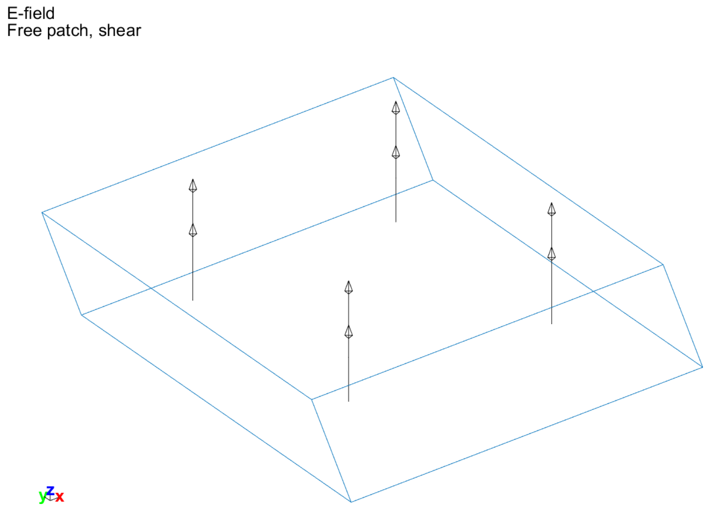


Figure 3.8: Deformed shape of a piezoelectric patch poled in the plane with an electric field applied in the out-of-plane direction

```
%% Step 2 Compute static response
% to avoid rigid body mode
```

```
model=stack_set(model,'info','Freq',10);
def=fe_simul('dfrf',model); def.lab={'Free patch, shear'};
def.fun=[0 1]; def=feutil('rmfield',def,'data','LabFcn');

% Append mechanically constrained structure
%    can't call fe_simul because no free DOF
%    see code with sdtweb d_piezo('scriptFullConstrain')
def=d_piezo('scriptFullConstrain',model,def);
def.lab{2}='Constrained patch, shear';
```

(d_piezo('TutoShearPatch-s3'))

```
%% Step 3 Vizualise deformed shape
cf=feplot(model,def); fecom('undef line');

% Electric field representation
p_piezo('viewElec EltSel "matid1" DefLen 20e-4 reset',cf);
cf.mdl.name='E-field'; % Sets figure name
d_piezo('SetStyle',cf); feplot(cf);
```

The mean of shear strain and stress is evaluated and compared to the d_{24} piezo coefficient. Note that the mean values are computed in the global yz axis for which a negative strain corresponds to a positive strain in the local 23 axis. Finally the capacitance is evaluated and compared to the theoretical values, showing a perfect agreement, and demonstrating the difference with the extension case for C^S.
(d_piezo('TutoShearPatch-s4'))

```
%% Step 4 : Check constitutive law
% Decompose constitutive law
CC=p_piezo('viewdd -struct',cf); %

% Display and compute mean strains
a=p_piezo('viewstrain -curve -mean',cf); % Strain S
fprintf('Relation between mean strain on free structure and d_24\n');
E3=a.Y(9,1); disp({'E3 mean' a.Y(9,1)  1/2e-3 'E3 analytic'})

disp([a.X{1}(4) num2cell([a.Y(4,1)/E3 CC.d(2,4)']) ...
     {'d_24'}])

% Display and compute mean stresses
b=p_piezo('viewstress -curve -mean',cf); % Stress T
```

```
fprintf('Relation between mean stress on pure electric and e_24 \n');
disp([b.X{1}(4) num2cell([b.Y(4,2)/-E3 CC.e(2,4)']) ...
     {'e_24'}])


% Mean stress/strain
disp([b.X{1} num2cell(b.Y(:,2)) num2cell(a.Y(:,1)) a.X{1}])

% Theoretical values of Capacitance and charge density - free patch
CT=CC.epst_r(2,2)*8.854e-12*1e-2*1e-2/2e-3; %% Capacitance - free patch
CdensT=CC.epst_r(2,2)*8.854e-12/2e-3*1e12; %% charge density - free patch

% Theoretical values of Capacitance and charge density - constrained patch
CS=CC.epss_r(2,2)*8.854e-12*1e-2*1e-2/2e-3; %% Capacitance - constrained patch
CdensS=CC.epss_r(2,2)*8.854e-12/2e-3*1e12; %% charge density - constrained patch

% Represent charge density (C/S) value on the electrodes
%   - compare with analytical values
cut=p_piezo('electrodeviewcharge',cf,struct('EltSel','matid 1'));
b=fe_caseg('stressobserve',cut,cf.def);b=reshape(b.Y,[],2);
disp([{'','CdensT','CdensS'};{'Numeric';'Theoretical'} ...
  num2cell([mean(abs(b));CdensT CdensS])])
    iimouse('zoom reset');
```

(d_piezo('TutoShearPatch-s5'))

```
%% Step 5 Check capacitance
% Compute the value of the total charge (from reaction at electrical dof)
% Ccompare with analytical values
p_piezo('electrodeTotal',cf) %
disp('Theoretical values of capacitance')
disp([{'CT';'CS'} num2cell([CT;CS])])
```

3.6 Example 2: Dynamic response of a piezoelectric disk

In this next example, we consider a piezoelectric disk of thickness=$2mm$ and radius=$8mm$ which has electrodes on the top and bottom surfaces. The material used is *PIC181* from *PI Ceramics*. The mesh and corresponding electrodes are generated with the following script and represented in Figure 3.9. (d_piezo('TutoDiskImpedance-s1'))

```
% See full example as Matlab code in d_piezo('ScriptTutoDiskImpedance')
d_piezo('Definestyles');

%% Step 1 Build and represent mesh and electrodes
model=d_piezo('MeshPIC181disk th=2e-3 r=8e-3 ner=10 nez=4 nrev=16');
feplot(model); cf=fecom; cf.mdl.name='PIC 181 piezo disk mesh'; iimouse('resetview')
d_piezo('setstyle',cf)


% Visualize electrodes
fecom('curtabCase',{'Top Actuator';'Bottom Actuator'}) %
fecom(';showline;proviewon;triax') %
cf.mdl.name='PIC 181 piezo disk electrodes'
d_piezo('setstyle',cf)
```

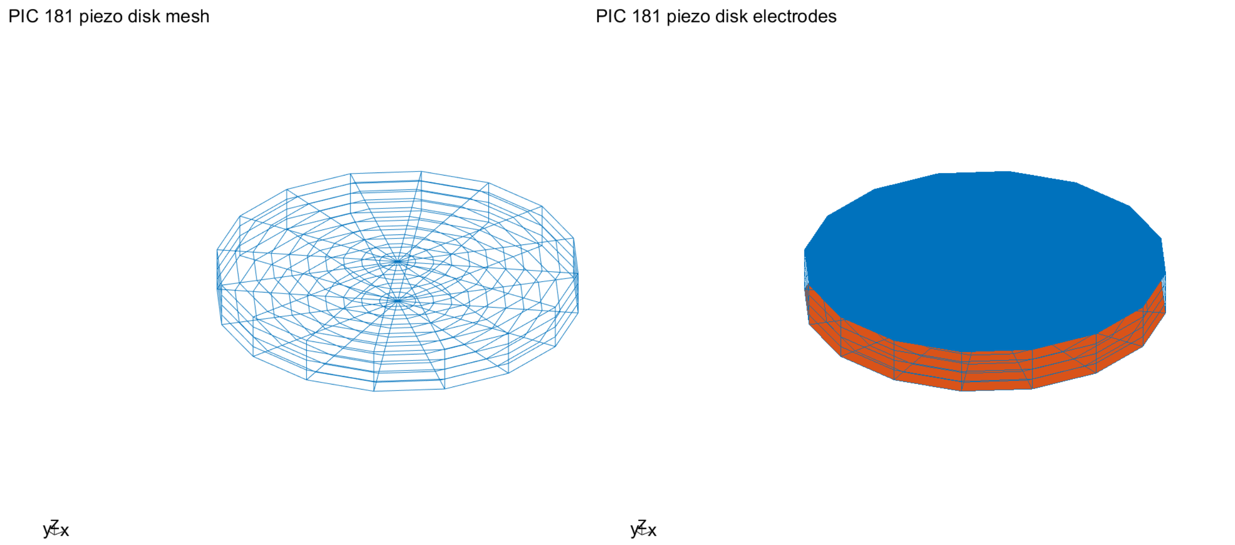


Figure 3.9: Piezo electric disk made of bulk PIC181 material (radius=8mm, thickness=2mm): mesh(left) and electrodes(right)

We compute the dynamic response of the disk subjected to an imposed voltage on the top electrode (the bottom electrode is grounded in the model), the frequency range is from 20 to 200 kHz. We represent the voltage distribution on the disk, as well as the electric field at 20 kHz in Figure 3.10. (d_piezo('TutoDiskImpedance-s2'))

```
%% Step 2 : Define range of frequencies and compute dynamic response
frq=linspace(20e3,200e3,256);
def=fe_simul('dfrf',stack_set(model,'info','Freq',frq));


% visualize potential
feplot(model,def); cf=fecom;
fecom(';showpatch;colordata21;'); cf.mdl.name='PIC 181 piezo disk voltage'
d_piezo('setstyle',cf) ;
cf.osd_('cbtr{string,Voltage(V)}')
fecom('colorscaleone') %To have the correct scale



% View electric field
fecom(';showline;scd 1e-4')
p_piezo('viewElec EltSel "matid1" DefLen 1e-4',cf);
cf.mdl.name='PIC 181 piezo disk E-field'
% To have a single color change clim (must be done with axProp to bypass normal)
st=cf.ua.axProp; st(3:4)={'@axes',{'clim',[480 510]}};cf.ua.axProp=st;
d_piezo('setstyle',cf)
cf.osd_('cbtr{string,E(V/m)}')
```

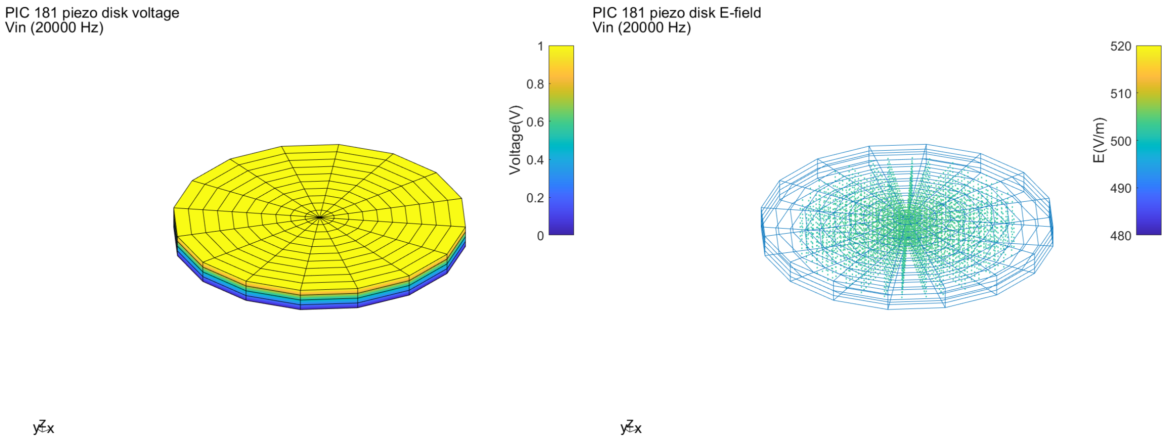


Figure 3.10: Response of a piezoelectric disk made of bulk PIC181 material (radius=$8mm$, thickness=$2mm$) at 20kHz, voltage distribution(left) and electric field (right)

A charge sensor is defined on the piezoelectric disk, allowing to compute the charge accumulated at each frequency for an accumulated input voltage. It is represented in Figure 3.11(left). (d_piezo('TutoDiskImpedance-s3'))

```
%% Step 3: Compute q/V as a function of the frequency
sens=fe_case(model,'sens');
C1=fe_case('SensObserve â^'DimPos 2 3 1',sens,def);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);
```

The electric impedance is defined as $Z = V/I = 1/(j\omega)\,V/Q$ and represented in Figure 3.11(right). The anti-resonance in the impedance curve around 140kHz corresponds to the short-circuited radial resonance frequency of the disk, while the resonance around 160kHz corresponds to the open-circuit resonance frequency of the disk. The spacing between these two frequencies can be used to compute the electromechanical coefficient of the disk for the first radial resonance. (d_piezo('TutoDiskImpedance-
)

```
%% Step 4: Compute and plot electric impedance
% extract impedance
C2=C1; C2.Y=1./(2*pi*1i*C2.X{1}.*C2.Y); C2.X{2}={'Imp(Ohm)'};
iicom(ci,'curveInit',C2.name,C2); iicom('submagpha');
d_piezo('setstyle',ci);
```

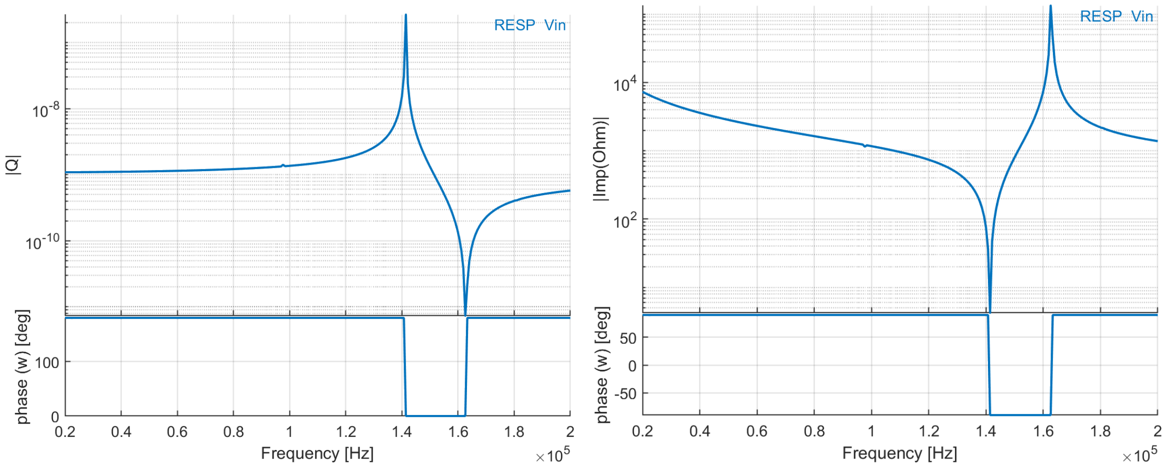


Figure 3.11: Response of a piezoelectric disk made of bulk PIC181 material (radius=8mm, thickness=2mm) from 20kHz to 200kHz, Q/V(left) and electric impedance (Ohm) (right)

4

Sensors and Actuators definition

Contents

4.1 Input/Output shape matrices

Dynamic loads applied to a discretized mechanical model can be decomposed into a product $\{F\}_q = [b]\{u(t)\}$ where

- the **input shape matrix** $[b]$ is time invariant and characterizes spatial properties of the applied forces
- the vector of inputs $\{u\}$ allows the description of the time/frequency properties.

Similarly it is assumed that the outputs $\{y\}$ (displacements but also strains, stresses, etc.) are linearly related to the model coordinates $\{q\}$ through the sensor **output shape matrix** ($\{y\} = [c]\{q\}$).
Input and output shape matrices are typically generated with fe_c, fe_case or fe_load. Understanding what they represent and how they are transformed when model DOFs/states are changed is essential.

Linear mechanical models take the general forms

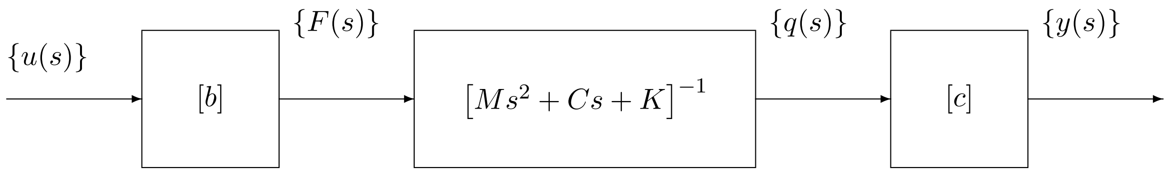
$$\begin{gathered}\left[Ms^2+Cs+K\right]_{N\times N}\{q(s)\}=[b]_{N\times NA}\{u(s)\}_{NA\times 1}\\ \{y(s)\}_{NS\times 1}=[c]_{NS\times N}\{q(s)\}_{N\times 1}\end{gathered} \tag{4.1}$$

in the frequency domain, and

$$\begin{gathered}[M]\{q''\}+[C]\{q'\}+[K]\{q\}=[b]\{u(t)\}\\ \{y(t)\}=[c]\{q(t)\}\end{gathered} \tag{4.2}$$

in the time domain.

N is the number of degrees of freedom in the model, NA is the number of independent actuators, and NS is the number of sensors.
In the model form (4.1), the first set of equations describes the evolution of $\{q\}$. The components of q are called Degrees Of Freedom (DOFs) by mechanical engineers and states in control theory. The second *observation* equation is rarely considered by mechanical engineers (hopefully the *SDT* may change this). The purpose of this distinction is to lead to the block diagram representation of the structural dynamics



which is very useful for applications in both control and mechanics.

In the simplest case of a point force input at a DOF q_l, the input shape matrix is equal to zero except for DOF l where it takes the value 1

$$[b_l] = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \leftarrow l \tag{4.3}$$

Since $\{q_l\} = [b_l]^T \{q\}$, the transpose of this Boolean input shape matrix is often called a *localization matrix*. Boolean input/output shape matrices are easily generated by `fe_c` Input/output shape matrices become really useful when not Boolean. For applications considered in the *SDT* they are key to

- distributed FEM loads, see `fe_load`.
- the description of piezoelectric loads and sensors, see `p_piezo`.
- the description of sensors that do not directly correspond to DOFs (accelerations in non global directions at positions that do not correspond to finite element nodes).
- model reduction. To allow the changes to the DOFs q while retaining the physical meaning of the I/O relation between $\{u\}$ and $\{y\}$.

4.2 Collocated force-displacement pairs

Collocated force-displacement pairs are commonly used in active vibration control, as they result in an alternance of poles and zeros in the open-loop transfer functions, leading to unconditionnaly stable control schemes (when actuators and sensors dynamics are neglected). The definition of such pairs is performed in SDT by first defining the sensors (related to DOFs in the model) and then creating the respective collocated forces. This is illustrated below on a 3D model of a U-shaped cantilever beam.

```
% See full example as MATLAB code in d_piezo('ScriptTutoBeamCollocated')
d_piezo('DefineStyles');

%% Step 1 : meshing and BC
model = femesh('test ubeam');
% BC : fix top
```

```
model=fe_case(model,'FixDOF','Clamp','z==0');

%% Step 2: Compute modes and frequencies
def=fe_eig(model,[5 10 0]);

%% Step 3 : Introduce a point displacement sensor and visualize
% sdtweb sensor#slab % URN based definition of sensors
model = fe_case(model,'SensDOF','Point Sensors',{'104:x'});
cf=feplot(model); iimouse('resetview');

% Make mesh transparent :
fecom('showfialpha') %

% Visualize sensor
 fecom proviewon
 fecom curtabcases 'Point Sensors' % Shows the case 'Point Sensors'

% Improve figure

% Arrow length and thickness
sdth.urn('Tab(Cases,Point Sensors){Proview,on,deflen,.25}',cf)
sdth.urn('Tab(Cases,Point Sensors){arProp,"linewidth,2"}',cf)

cf.mdl.name='Ubeam PS1'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);

% Insert the number for the sensor :
fecom('textnode',104,'fontsize',14)
```

Figure 4.1 shows the finite element mesh of the U-beam with a sensor added on node 104 in the x-direction. The load is then defined as being collocated to the sensor, i.e, in the x-direction and on node 104. The static response of the U-beam to this load is computed with `fe_simul` and shown in Figure 4.2.

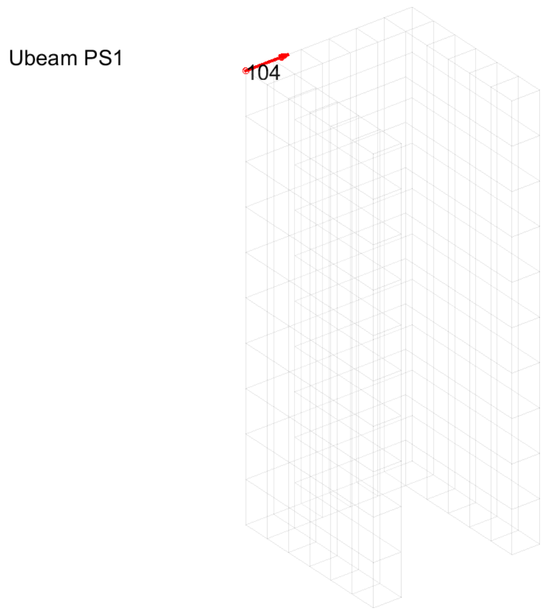


Figure 4.1: U-beam with displacement sensor added on node 104 in direction x

```
%% Step 4 : Introduce collocated force actuator
model=fe_case(model,'DofLoad SensDof','Collocated Force','Point Sensors:1') % 1 for fir

%% Step 5 : compute static response and visualize
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % Static response
%% Step 5 : compute static response and visualize
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % Static response
feplot(model,d0); fecom(';scd .3;undef line');
fecom curtabcases 'Point Sensors'
sdth.urn('Tab(Cases,Point Sensors){Proview,on,deflen,.25}',cf)
sdth.urn('Tab(Cases,Point Sensors){arProp,"linewidth,2"}',cf)
fecom('textnode',104,'fontsize',14);

% Title
cf.mdl.name='Ubeam PS1 Static';
d_piezo('SetStyle',cf); feplot
```

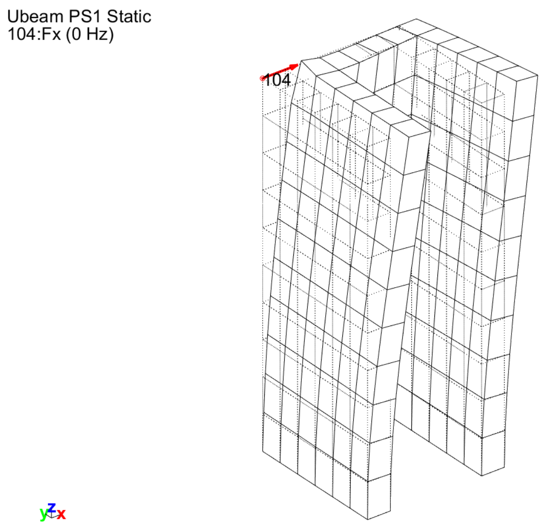


Figure 4.2: Static response of U-beam to load on node 104 in direction x

The collocated transfer function is then computed using fe_simul and plotted in the iiplot environment (Figure 4.3)

```
%% Step 6 :  Compute dynamic response in freq band of first 5 modes and plot
frq=linspace(0,def.data(6)-(def.data(6)-def.data(5))/2,300);
d1=fe_simul('dfrf',stack_set(model,'info','Freq',frq)); % Dynamic response

% Construct projection matrix in sens.cta and plot collocated FRF
sens=fe_case(model,'sens');
% Plot the 4 FRFs, FRFs 1 and 4 are collocated
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci)
```

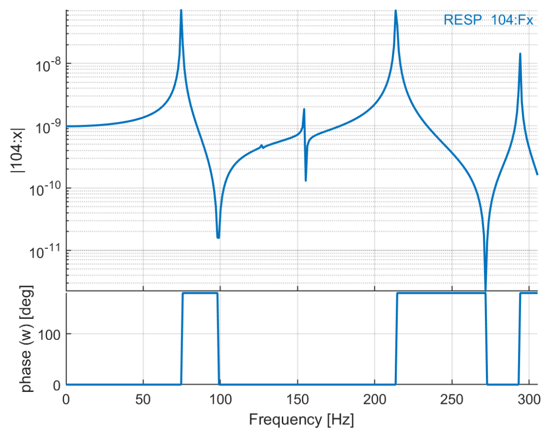


Figure 4.3: Collocated transfer function for a force applied on node 104 in direction x, and a displacement at the same node and in the same direction

Multiple sensors and actuators can also be generated. The single sensor is changed to two sensors (adding a sensor on node 344 in y-direction). Note that if the same name is used in the definition of the sensors, the previous definition is replaced (here 'Point sensors' is the name of the sensing case). The two sensors are shown on Figure 4.4.

```
%%% Step 7 : multiple collocated sensors and actuators

% Introduce two sensors and visualize
model = fe_case(model,'SensDOF','Point sensors',{'104:x';'344:y'});
cf=feplot(model);
% Visualize sensors :
fecom('showfialpha') %
fecom proviewon
fecom curtabcases 'Point Sensors' % Shows the case 'Point Sensors'

% Improve figure
sdth.urn('Tab(Cases,Point Sensors){Proview,on,deflen,.25}',cf)
sdth.urn('Tab(Cases,Point Sensors){arProp,"linewidth,2"}',cf)

% Title
cf.mdl.name='Ubeam MS1';

% Insert the number for the sensor :
fecom('textnode',[104 344],'FontSize',14)
```

```
d_piezo('SetStyle',cf)
```

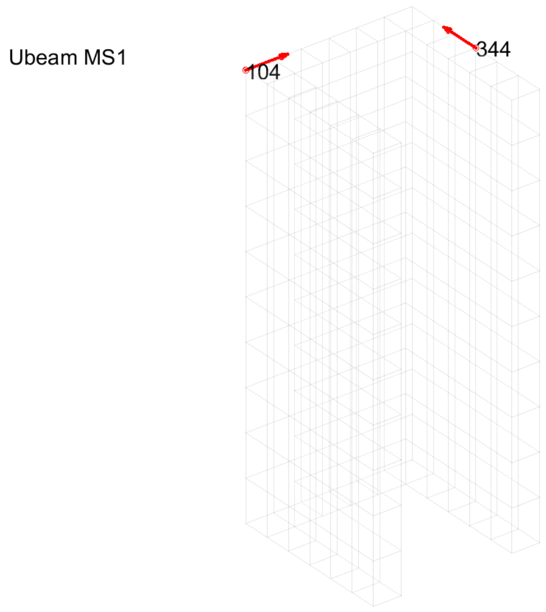


Figure 4.4: U-beam with displacement sensor added on node 104 in direction x and 344 in direction y

Two forces collocated to these sensors are then defined, and the static response is computed and shown in Figure 4.5. The four transfer functions resulting from the definition of two sensors and actuators are then computed and plotted. Two of them are collocated resulting in alternance of poles and zeros (Figure 4.6), and the two others are not collocated and do not show this alternance (Figure 4.7). Note that the two transfer functions are identical due to the reciprocity in linear systems.

```
%% Step 8 : Introduce collocated force actuators
model=fe_case(model,'DofLoad SensDof','Collocated Force','Point sensors:1:2')

%% Step 9 : compute static response and visualize (two static responses)
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % Static response
feplot(model,d0); fecom(';scd .3; undef line')
fecom('textnode',[104 344],'FontSize',14)

% Style
d_piezo('SetStyle',cf);  cf.os_('LgMl-FontSize14');% Keep both mdl.name and title
```

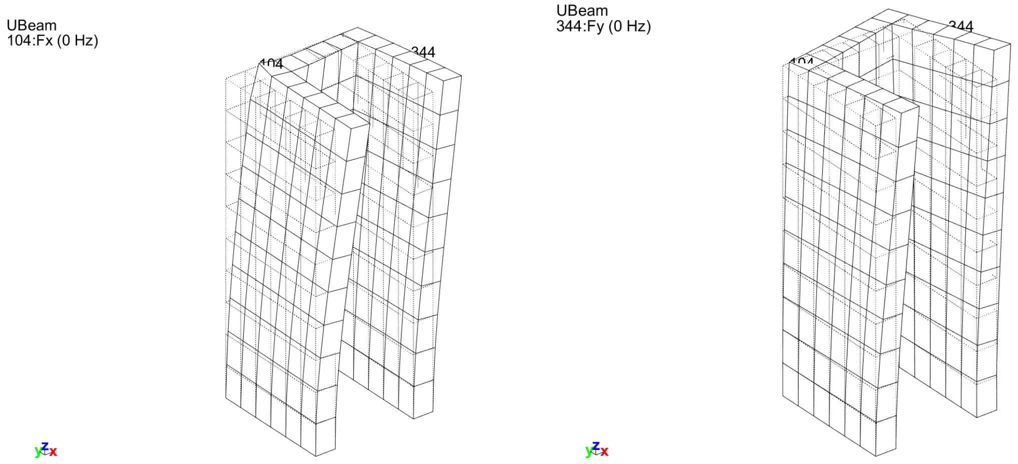


Figure 4.5: Static response of the U-beam to two loads (104-x, 344-y)

```
%% Step 10 :  Compute dynamic response in freq band of first 5 modes and plot
frq=linspace(0,def.data(6)-(def.data(6)-def.data(5))/2,300);
d1=fe_simul('dfrf',stack_set(model,'info','Freq',frq)); % Dynamic response

% Construct projection matrix in sens.cta and project resp on sensor
sens=fe_case(model,'sens');

% Plot the 4 FRFs, FRFs 1 and 4 are collocated
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);
```

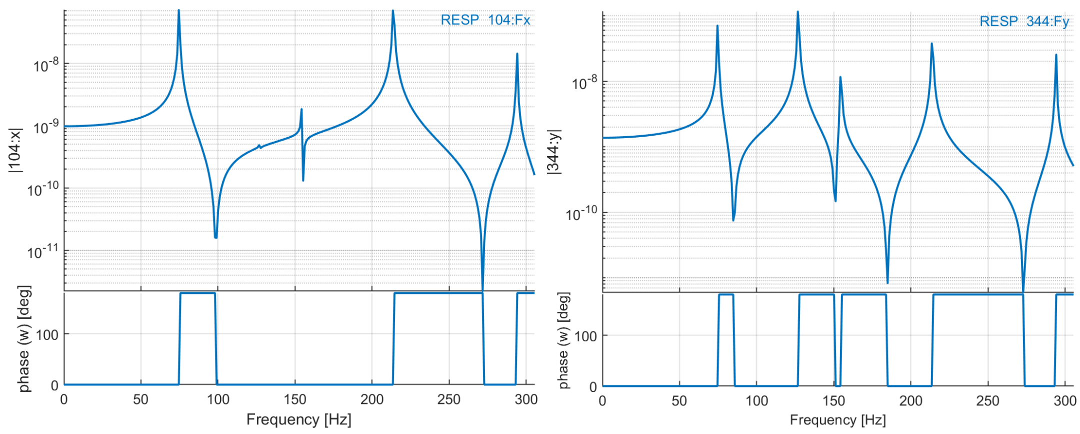


Figure 4.6: Collocated transfer functions for two loads (104-x, 344-y)

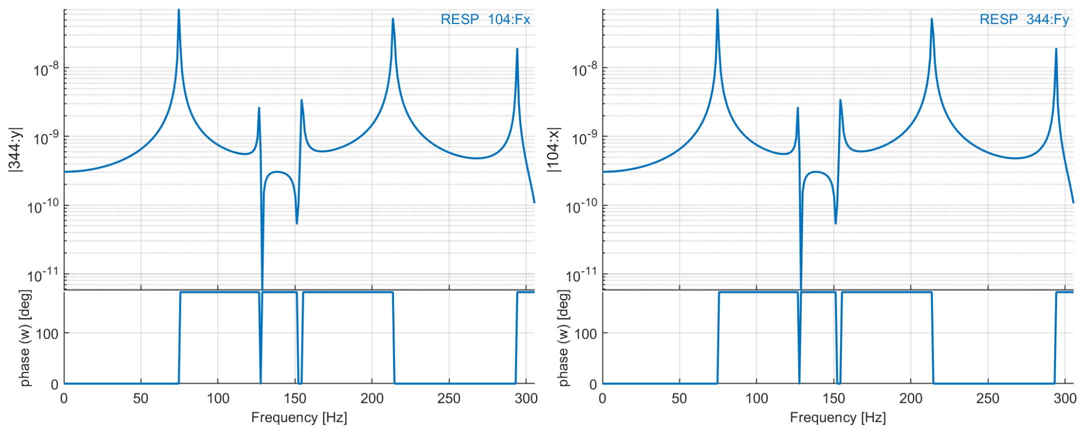


Figure 4.7: The 4 transfer functions for two loads (104-x, 344-y), and two collocated point sensors

```
% End of script
```

4.3 Non-collocated force-displacement pairs and combinations

The example below illustrates the general definition of actuators and sensors which are not collocated.

```
% See full example in d_piezo('ScriptTutoBeamNonCollocated')
d_piezo('DefineStyles'); % Init styles for figures
```

```
% Example 2 : Non collocated point sensors and actuators

%% Step 1 : meshing and BC
model = femesh('test ubeam');
% BC : fix top
model=fe_case(model,'FixDOF','Clamp','z==0');

%% Step 2: Compute modes and frequencies
def=fe_eig(model,[5 10 0]);

%% Step 3 : Define sensors
model = fe_case(model,'SensDOF','Point Sensors',{'104:x';'207:y'});
cf=feplot(model); iimouse('resetview');

% Make mesh transparent :
fecom('showfialpha') %

% Visualize sensor
 fecom proviewon
 fecom curtabcases 'Point Sensors' % Shows the case 'Point Sensors'

% Improve figure
% Arrow length and thickness
sdth.urn('Tab(Cases,Point Sensors){Proview,on,deflen,.25}',cf)
sdth.urn('Tab(Cases,Point Sensors){arProp,"linewidth,2"}',cf)

cf.mdl.name='Ubeam MSNC'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);

% Insert the number for the sensor :
fecom('textnode',[104 207],'fontsize',14)
```

Two point sensors are defined as 104-x and 207-y (Figure 4.8).

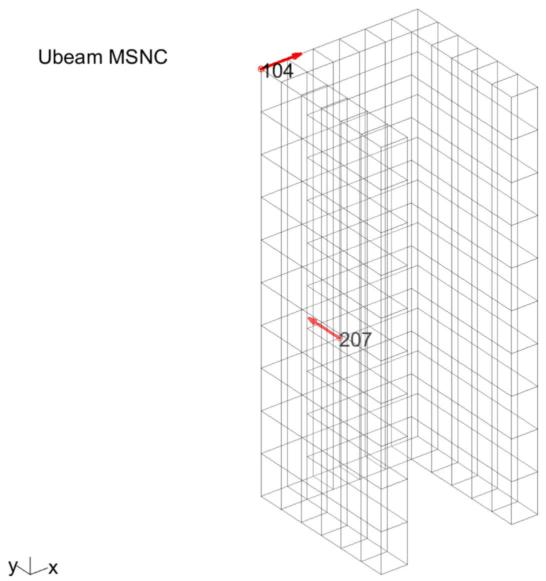


Figure 4.8: Point sensors at node 104 in direction x and 207 in direction y

Two actuators are then defined by combination of three forces (207-x, 241-x and 207-y). The first actuator consists in two forces on nodes 207 and 241 in opposite direction along x acting together, and the third force is on 207-y. The static and dynamic response to these loads is represented in Figure 4.10 and Figure 4.11.

```
%% Step 4 : Define point actuators
% relative force between DOFs 207x and 241x and one point loads at DOFs 207y
data  = struct('DOF',[207.01;241.01;207.02],'def',[1 0;-1 0;0 1]);
model=fe_case(model,'DofLoad','Actuators',data); %
cf=feplot(model); fecom('showline')
fecom curtabcases 'Actuators' % Shows the case 'Actuators'

% Improve figure
% Arrow length and thickness
sdth.urn('Tab(Cases,Actuators){Proview,on,deflen,.25}',cf)
sdth.urn('Tab(Cases,Actuators){arProp,"linewidth,2"}',cf)

cf.mdl.name='Ubeam MANC 1'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);

% Insert the number for the sensor :
```

```
fecom('textnode',[241 207],'fontsize',14)


% Visualize second combination
cf.CStack{'Actuators'}.Sel.ch=2;sdth.urn('Tab(Cases,Actuators)',cf) % second
cf.mdl.name='Ubeam MANC 2'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);
```

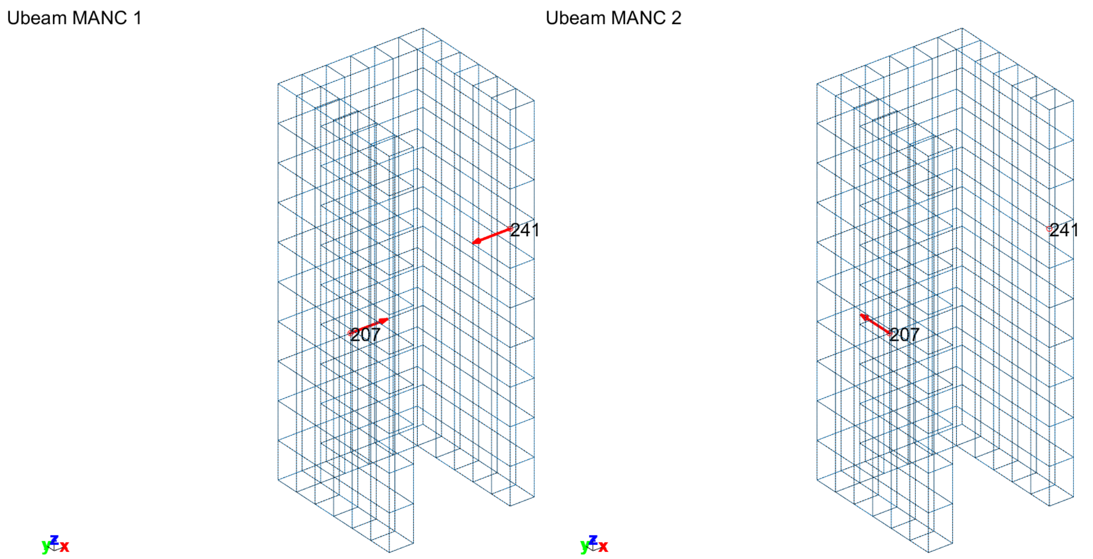


Figure 4.9: Load combinations for Ubeam

```
%% Step 5 : compute static response and visualize
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % Static response
feplot(model,d0); fecom(';scd .1; undef line;')


% Title
d_piezo('SetStyle',cf); cf.os_('LgMl-FontSize14');% Keep both mdl.name and title
fecom('textnode',[241 207],'FontSize',14)
```

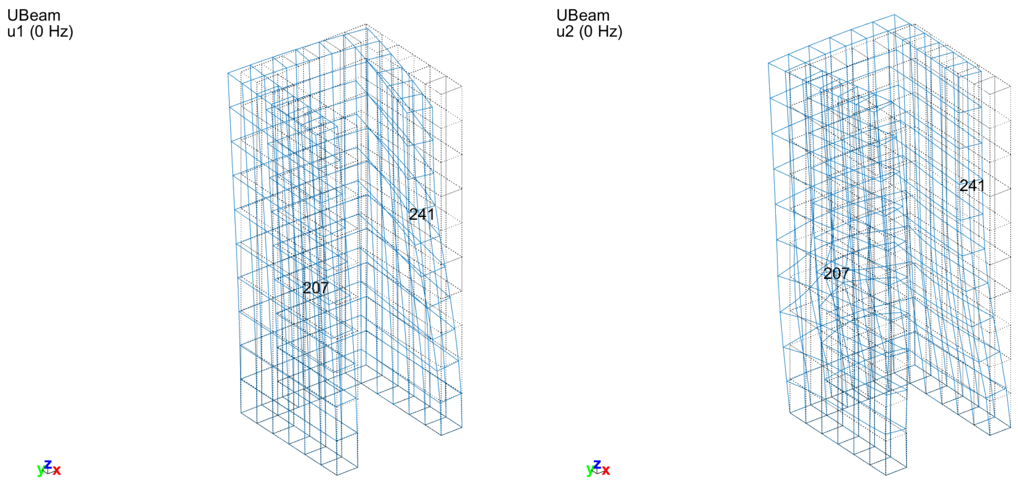


Figure 4.10: Static responses of Ubeam to load combinations

```
%% Step 6 :  Compute dynamic response in freq band of first 5 modes and plot

% compute response
frq=linspace(0,def.data(6)-(def.data(6)-def.data(5))/2,300);
d1=fe_simul('dfrf',stack_set(model,'info','Freq',frq)); % Dynamic response

% Construct projection matric in sens.cta and project resp on sensor
sens=fe_case(model,'sens');

% Plot FRFs
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);
```

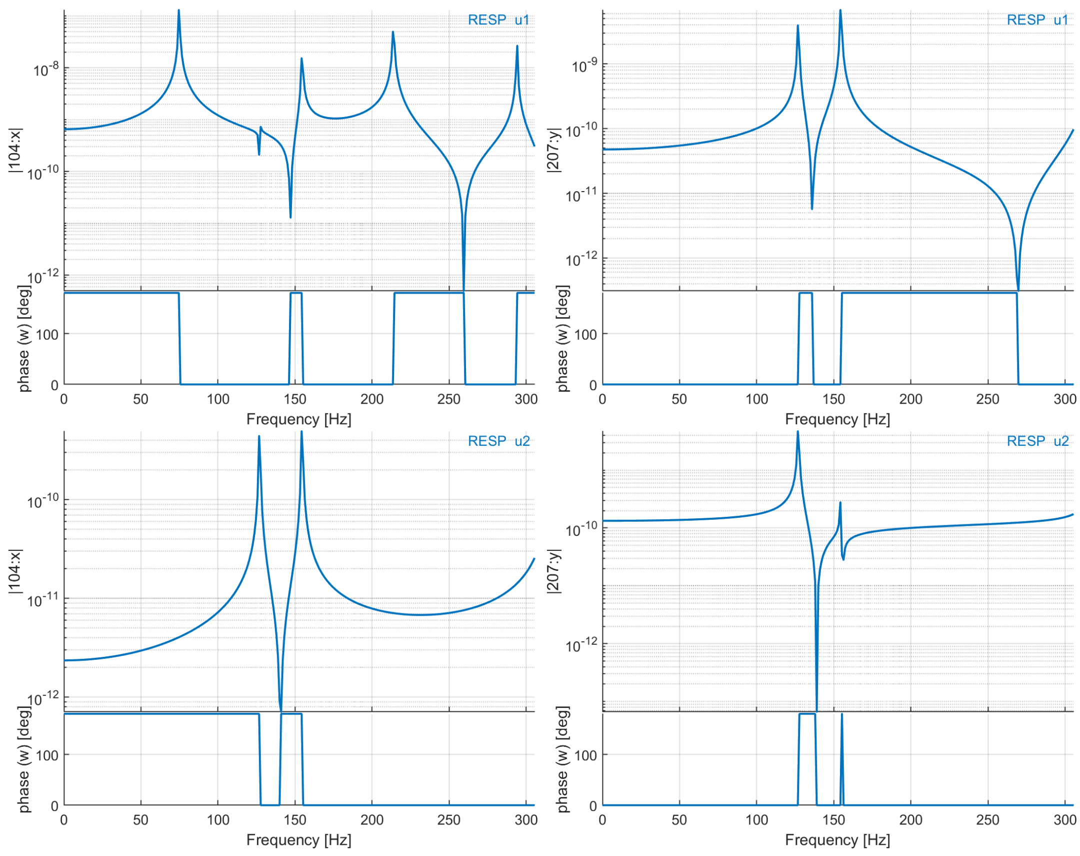


Figure 4.11: Dynamic response of Ubeam to load combinations u1 and u2, sensors on node 104 in direction x, and 207 in direction y

```
% End of script
```

4.4 Other types of actuators

Other types of loads, such as surface or volumic loads are handled by the `fe_load` command (`sdtweb('fe_load')` for more details) in SDT. The case of imposed displacement is handled with `fe_case('DofSet')` calls. The following script illustrates the use of volume and surface loads on the U-beam.

```
% See full example in d_piezo('ScriptTutoBeamSurfVol')
 d_piezo('DefineStyles'); % Define styles for figures

%% Step 1 Apply  a volumic load and represent
 model = femesh('testubeam');
 data=struct('sel','groupall','dir',[0 32 0]);
 data2=struct('sel','groupall','dir',{{0,0,'(z-1).^3.*x'}});
 model=fe_case(model,'FVol','Constant',data, ...
                     'FVol','Variable',data2);

% Visualize loads
cf=feplot(model); iimouse('resetview');

% Make mesh transparent :
fecom('showfialpha') %

% Visualize Load
 fecom proviewon

% Improve figure
 sdth.urn('Tab(Cases,Constant){deflen,.5,arProp,"linewidth,2"}',cf)
  fecom curtabcases 'Constant' % Shows the case 'Constant'


 % Set style and print
cf.mdl.name='Ubeam VLoad-Cst'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);
```

The first load is defined by 'data' and is a constant volumic load in the y-direction. The second load, defined by 'data2' is in the z-direction and its magnitude depends both on the x and z position in the U-beam ($f(x,z) = x*(z-1)^3$). The two loads are represented in Figures 4.12 and 4.13 using arrows and color codes (for the variable force only).

```
% Visualize variable Load and print
  sdth.urn('Tab(Cases,Variable){deflen,.5,arProp,"linewidth,2"}',cf)
 fecom curtabcases 'Variable' % Shows the case 'Variable'
 cf.mdl.name='Ubeam VLoad-Var'; % Model name for title
 d_piezo('SetStyle',cf); feplot(cf);
```

```
% Visualize Constant and Variable loads with colors
Load = fe_load(model); cf=feplot(model,Load); cf.mdl.name='Ubeam VLoad Color'; % Model 
fecom(';showpatch;colordataz;scd .0001;'); % display as color-code to see change of vol
d_piezo('SetStyle',cf); feplot(cf); fecom('colorbar on')
```

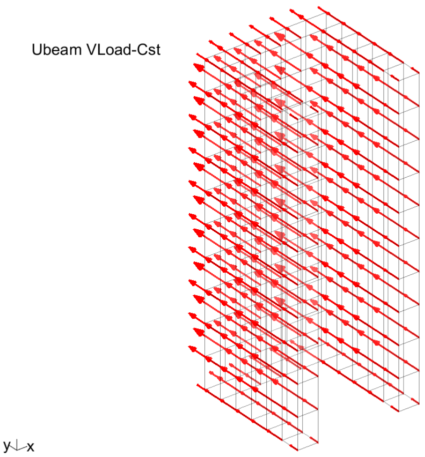


Figure 4.12: Constant volumic force applied in the y-direction

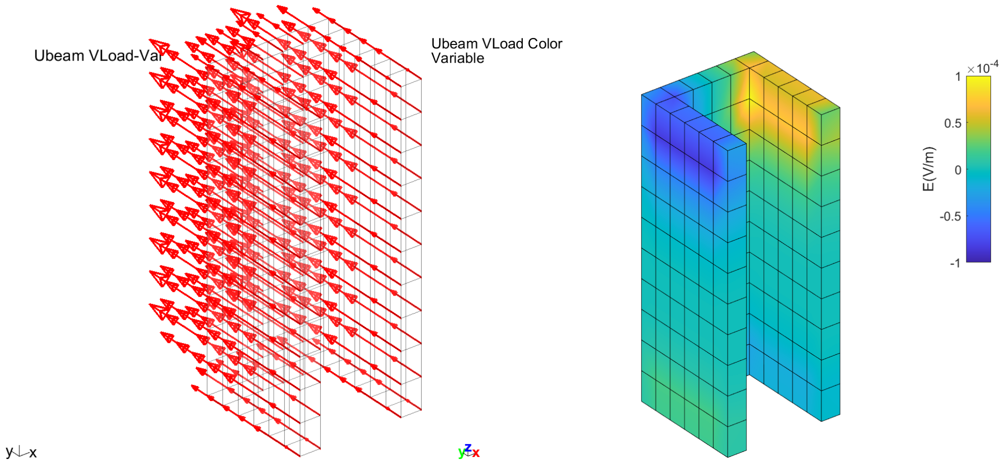


Figure 4.13: Variable volumic force applied in the z-direction represented by arrows (left) and color code proportionnal to amplitude in z-direction (right)

The following example shows how to define a surface load, using selectors to define the area of the surface where the load is applied. In this case, the load is on the surface corresponding to $x = -0.5$ and $z > 1.25$. The resulting surface load is represented in Figure 4.14.

```
%% Step 2 :  Apply a surface load case in a model using selectors
 data=struct('sel','x==-.5', ...
             'eltsel','withnode {z>1.25}','def',1,'DOF',.19);
 model=fe_case(model,'Fsurf','Surface load',data); cf=feplot(model);

% Visualize Load
 fecom proviewon
 fecom curtabcases 'Surface Load' % Shows the case 'Constant'
 fecom showline

 sdth.urn('Tab(Cases,Surface load){deflen,.5,arProp,"linewidth,2"}',cf)
 cf.mdl.name='Ubeam SLoad'; % Model name for title
 d_piezo('SetStyle',cf); feplot(cf);
```

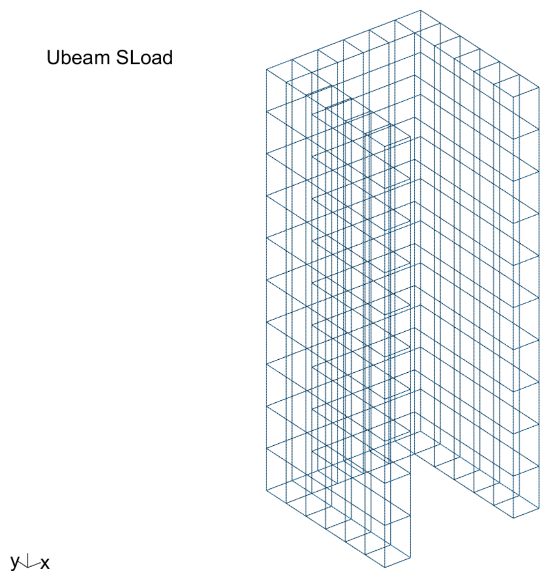


Figure 4.14: Surface load applied on a specific surface of the U-beam

The same can be done using node lists as in the example below

```
%% Step 3 : Applying a surfacing load case in a model using node lists
```

```
 data=struct('eltsel','withnode {z>1.25}','def',1,'DOF',.19);
 NodeList=feutil('findnode x==-.5',model);
 data.sel={'','NodeId','==',NodeList};
 model=fe_case(model,'Fsurf','Surface load 2',data); cf=feplot(model);

 fecom proviewon
 fecom curtabcases 'Surface load 2' %
 fecom showline

sdth.urn('Tab(Cases,Surface load){deflen,.5,arProp,"linewidth,2"}',cf)
cf.mdl.name='Ubeam SLoad2'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);
```

The last example illustrates the use of sets to define surface loads (Figure 4.15)

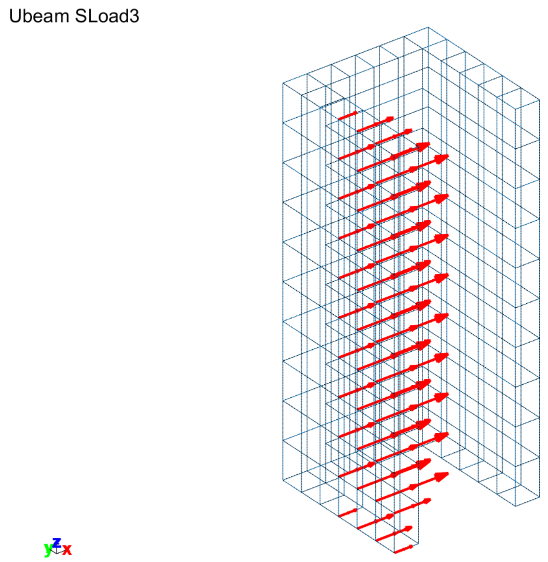


Figure 4.15: Surface load applied on a specific surface of the U-beam

```
%% Step 4 : Applying a surfacing load case in a model using sets

 % Define a face set
  [eltid,model.Elt]=feutil('eltidfix;',model);
 i1=feutil('findelt withnode {x==-.5 & y<0}',model);i1=eltid(i1);
 i1(:,2)=2; % fourth face is loaded
 data=struct('ID',1,'data',i1,'type','FaceId');
 model=stack_set(model,'set','Face 1',data);

 % define a load on face 1
 data=struct('set','Face 1','def',1,'DOF',.19);
 model=fe_case(model,'Fsurf','Surface load 3',data); cf=feplot(model);

 sdth.urn('Tab(Cases,Surface load 3){deflen,.5,arProp,"linewidth,2"}',cf)
 fecom proviewon
 fecom curtabcases 'Surface load 3' %

cf.mdl.name='Ubeam SLoad3'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);


% End of script
```

Another type of load is the case when degrees of freedom are imposed (imposed displacement, velocity, acceleration, or electric potential in the case of piezoelectric actuators). The script below illustrates the case where the U-beam has an imposed displacement on the cantilever side, in the y-direction (Figure 4.16).

```
% See full example in d_piezo('ScriptTutoBeamUimp')
d_piezo('DefineStyles');

%% Step 1 Meshing and BC
model = femesh('test ubeam');
% BC : Impose displacement
% Fix all other dofs for Base
model=fe_case(model,'FixDof','Clamping','z==0 -DOF 1 3');

%% Step 2 Apply base displacement in y-direction
% find node z==0
nd=feutil('find node z==0',model);
data.DOF=nd+.02; data.def=ones(length(nd),1);
model=fe_case(model,'DofSet','Uimp',data); cf=feplot(model); iimouse('resetview')

% Visualize
fecom proview on
fecom curtabcases 'Uimp' % Shows the case 'Constant'
```

```
 % Set style and print
cf.mdl.name='Ubeam Uimp'; % Model name for title
d_piezo('SetStyle',cf); feplot(cf);
```

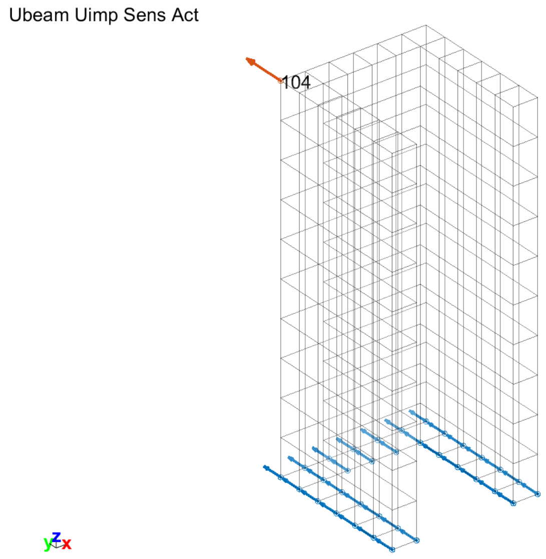


Figure 4.16: Imposed displacement of the base of the U-beam in the y-direction, and sensor at node 104 in the y-direction

```
%% Step 2 : Introduce a point displacement sensor and visualize

model = fe_case(model,'SensDOF','Point Sensors',{'104:y'});
cf=feplot(model);

% Make mesh transparent :
fecom('showfialpha') %

% Visualize sensor and actuator
 fecom proviewon
 sdth.urn('Tab(Cases,Point Sensors){arProp,"linewidth,2"}',cf)
 sdth.urn('Tab(Cases,Uimp){arProp,"linewidth,2"}',cf)
 fecom('curtabCase','#(Uimp|Point)')

cf.mdl.name='Ubeam Uimp Sens Act'; % Model name for title
```

```
d_piezo('SetStyle',cf); feplot(cf);

% Insert the number for the sensor :
fecom('textnode',[104],'fontsize',14)
```

The transfer function is then computed using `fe_simul` and plotted in the `iiplot` environment (Figure 4.17)

```
%% Step 3: Compute modes and frequencies (dofs in Dofset are fixed)
def=fe_eig(model,[5 10 0]);
feplot(model,def)

%% Step 4:  Compute dynamic response in freq band of first 5 modes and plot
frq=linspace(0,def.data(6)-(def.data(6)-def.data(5))/2,300);
d1=fe_simul('dfrf',stack_set(model,'info','Freq',frq)); % Dynamic response

% Construct projection matrix in sens.cta and plot collocated FRF
sens=fe_case(model,'sens');
% Plot the 4 FRFs, FRFs 1 and 4 are collocated
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);
```

```
% End of script
```

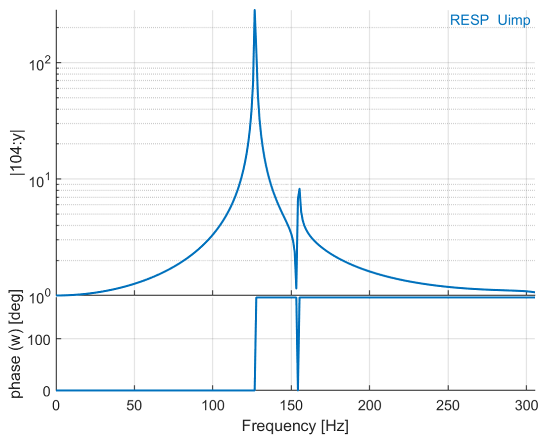


Figure 4.17: Transfer function for an imposed displacement of the base of the U-beam in the y-direction, and a displacement at node 104 in the y-direction

As expected, the static response is unitary, as the U-beam does not deform when a uniform unitary displacement of the base is imposed, so that node 104 moves in the same direction and with the same magnitude as the base.

Note that it is also possible to impose an acceleration when building state-space models, which will be detailed in a dedicated chapter.

4.5 Other types of sensors

For point sensors, velocity and acceleration sensors can also be defined. The short script below introduces a collocated sensor actuator pair on node 104 in direction x for the U-beam. The sensor is either a displacement, velocity or acceleration sensor, and the resulting collocated FRFs are plotted in Figure 4.18.

```
% See full example in d_piezo('ScriptTutoBeamDispVelAcc')
d_piezo('DefineStyles');
%% Step 1 : meshing, BC and compute modeshapes
model = femesh('test ubeam');
model=fe_case(model,'FixDOF','Clamp','z==0');

%% Step 2: Compute modes and frequencies
def=fe_eig(model,[5 10 0]);

%% Step 3 : Introduce a point displ/vel/acc sensor and collocated force
```

```
model = fe_case(model,'SensDOF','Sensors',{'104:x';'104:vx';'104:ax'});
model=fe_case(model,'DofLoad SensDof','Collocated Force','Sensors:1') % 1 for first sen

%% Step 4 : Compute response and plot FRFs
frq=linspace(10,def.data(6)-(def.data(6)-def.data(5))/2,300);

d1=fe_simul('dfrf',stack_set(model,'info','Freq',frq)); % Dynamic response

% Construct projection matrix in sens.cta and plot collocated FRF
sens=fe_case(model,'sens');

% Plot the 3 FRFs
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot; iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);
```

```
% End of script
```

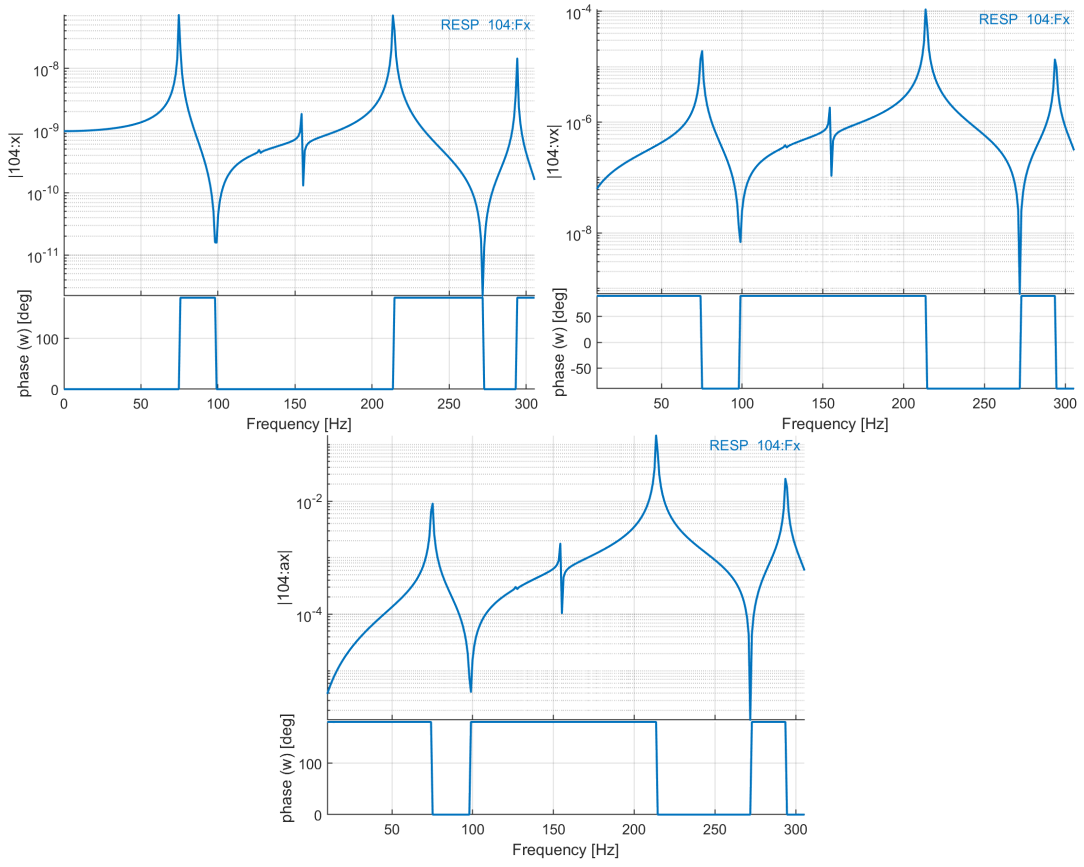


Figure 4.18: Collocated transfer functions with displacement, velocity and acceleration at the same point

Other types of sensors are also defined in SDT using `fe_case('Sens...')` calls. They are detailed in Section 4.6 of the general SDT manual and are of the following type:

- `general` general sensor (low level).
- `rel` relative displacement sensor.
- `trans` translation sensor.
- `triax` triaxial sensors.
- `laser` for laser vibrometers, defining line of sight of laser.
- `resultant` resultant force sensor.

- `strain` strain or stress sensor.

Refer to SDT general documentation for examples of definition of such transducers. SDT also provides advanced tools to define sensors which are not at the location of nodes in the finite element models. For more details, see the documentation of the `fe_sens` command.

In addition, piezoelectric sensors are discussed in the next section (section 4.6)

4.6 Piezoelectric sensors and actuators

4.6.1 General theory

Essentially, when using piezoelectric materials in finite element models of structures, additional electrical degrees of freedom are added to the model, generally in the form of voltage degrees of freedom. The general form of the equations of motion was already given (see (3.8)) and is recalled below:

$$\begin{bmatrix} M_{qq} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} q'' \\ V'' \end{Bmatrix} + \begin{bmatrix} C_{qq} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} q' \\ V' \end{Bmatrix} + \begin{bmatrix} K_{qq} & K_{qV} \\ K_{Vq} & K_{VV} \end{bmatrix} \begin{Bmatrix} q \\ V \end{Bmatrix} = \begin{Bmatrix} F_{mech} \\ Q \end{Bmatrix} \tag{4.4}$$

The equations of motion couple the electrical dofs to the mechanical ones through the matrices K_{Vq} and K_{qV}.

When piezoelectric materials are used for sensing and actuating, they have associated electrodes which enforce equipotentiality on the surfaces. In the finite element model, this is taken into account by adding equality constraints between all the electrical dofs associated to an electrode. One single electrical dof is then linked to each electrode, and these DOFs are used for actuation and sensing.

Note that in the shell formulation of SDT, the electrical dofs are not associated to nodes, but rather to elements, as detailed in section 3.2 . Each element can have several layers among which some can be defined with piezoelectric material. Each piezoelectric layer of the shell element has then one associated dof which corresponds to the difference of potential between the top and the bottom electrode of that layer. This is equivalent to considering that the bottom electrode of each layer is set to 0 volts and that the electrical dof represents the voltage on the top electrode.

For 3D elements, each node has an additional associated electrical dof, so that it is necessary to define boundary conditions in terms of voltage on the two electrodes. In SDT, electrical dofs are identified with .21

As explained in section 3.3 , actuation can be of two types : an electrical charge Q can be imposed on one of the electrodes, which is equivalent to defining a mechanical force, as this term is on the right-hand-side of the equations. This case should be treated as the addition of point loads using `fe_case('DofLoad')` calls. It is however more common to use piezoelectric actuators in a voltage-driven mode, which corresponds to imposing the voltage, and can be treated using a `fe_case('DofSet')`.

Assume that the voltage DOFs can be divided in two parts :

$$\{V\} = \left\{ \begin{array}{c} V_0 \\ V_{in} \end{array} \right\} \tag{4.5}$$

where V_{in} corresponds to the voltages imposed on the actuators in the model, and the other voltage DOFs are left free. We assume that the constraints related to the electrodes have already been taken into account to reduce the different matrices. The first line of the equations of motion becomes (assuming K_{VV} is a diagonal matrix, which is usually the case as it corresponds to the matrix of the individual capacitances) :

$$\left[\begin{array}{cc} M_{qq} & 0 \\ 0 & 0 \end{array} \right] \left\{ \begin{array}{c} q'' \\ V_0'' \end{array} \right\} + \left[\begin{array}{cc} C_{qq} & 0 \\ 0 & 0 \end{array} \right] \left\{ \begin{array}{c} q' \\ V_0' \end{array} \right\} + \left[\begin{array}{cc} K_{qq} & K_{qV_0} \\ K_{V_0q} & K_{V_0V_0} \end{array} \right] \left\{ \begin{array}{c} q \\ V_0 \end{array} \right\} = \left\{ \begin{array}{c} F_{mech} - K_{qV_{in}}V_{in} \\ Q_0 \end{array} \right\} \tag{4.6}$$

The fact that there is no coupling term in the mass and damping matrices leads to an equivalent problem where an imposed voltage corresponds to a mechanical load $-K_{qV_{in}}V_{in}$.

Piezoelectric sensors are also of two types. Voltage sensing corresponds to measuring the voltage DOF directly, and can thus be defined by a simple `fe_case('SensDOF')` call. If a charge amplifier is used to measure the signal generated by the piezoelectrc sensor, this is equivalent to measuring a reaction force at the associated electrical DOF. In order to do that, one first needs to fix the associated electrical DOF using a `fe_case('FixDOF')` (if set to 0) or a `fe_case('DofSet')` (if used both as voltage actuator and charge sensor as in the case of self-sensing and piezo-shunt applications) call, and then define the charge sensor with a `p_piezo ElectrodeSensQ` call. Assume again that the electrical dofs are divided in two parts

$$\{V\} = \left\{ \begin{array}{c} V_0 \\ V_q \end{array} \right\} \tag{4.7}$$

where V_q are fixed electrical DOFs where the charge needs to be measured. The charge on these electrodes is then given by:

$$Q = K_{V_qq}\{q\} + K_{V_qV_q}\{V_q\} \tag{4.8}$$

The definition of actuators and sensors of these different types are illustrated below for a plate modelled with shell elements, and an example of an accelerometer modelled with 3D elements.

4.6.2 Aluminum plate with 4pzt patches (Shell model)

The first example deals with an aluminum plate with 4 piezoceramic patches. The geometry and the material properties are given in Figure 4.19 and Table 4.1

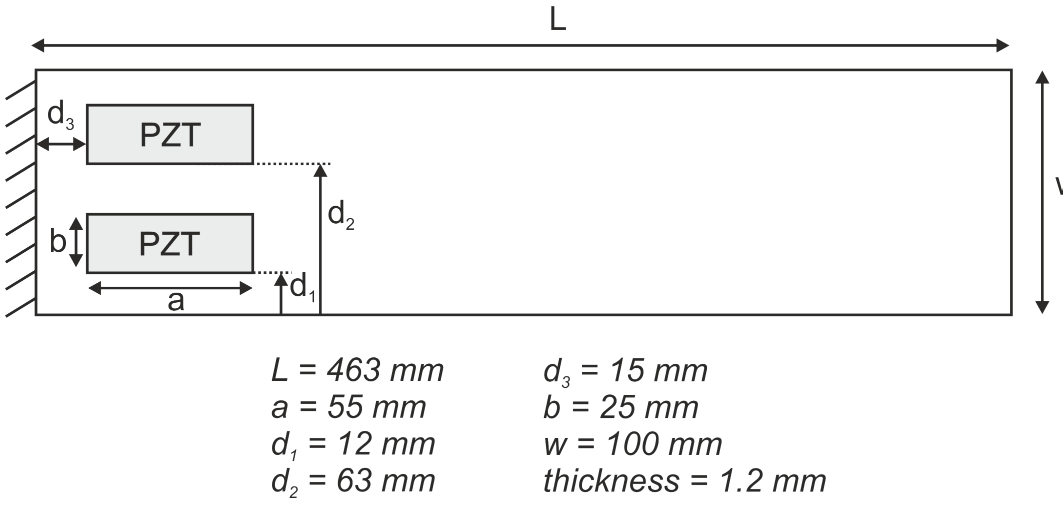


Figure 4.19: Geometric details of the aluminum plate with 4 piezoceramic patches

Property	Value
Aluminum plate	
E	$72GPa$
ν	0.3
ρ	$2700kg/m^3$
Piezoceramic patches	
E	$54.05GPa$
ν	0.41
ρ	$7740kg/m^3$
thickness	$0.25mm$
d_{31}	-185 $10^{-12}pC/N$ (or m/V)
d_{32}	-185 $10^{-12}pC/N$ (or m/V)
ε_{33}^T	1850 ε_0
ε_0	8.854 $10^{-12}Fm^{-1}$

Table 4.1: Material properties of the plate and the piezoceramic patches

It corresponds to a cantilevered plate with 4 piezoelectric patches modeled using the `p_piezo Shell` formulation. The first step consists in the creation of the model, the definition of the boundary conditions, and the definition of the default damping coefficient. The different meshing procedures are detailed further in section 5 . The resulting mesh is shown in Figure 4.20

(`d_piezo('TutoPlate_4pzt_single-s1')`)

```
% See full example in d_piezo('ScriptTutoPlate4Pzt')
d_piezo('DefineStyles');

%% Step 1 - Build model and visualize
model=d_piezo('MeshULBplate');  % creates the model
model=fe_case(model,'FixDof','Cantilever','x==0'); % Clamp plate
% Set modal default zeta = 0.01
model=stack_set(model,'info','DefaultZeta',0.01);
cf=feplot(model); fecom('colordatagroup'); set(gca,'cameraupvector',[0 1 0])
cf.mdl.name='Plate_4pzt';
d_piezo('SetStyle',cf); feplot(cf);
```

One can have access to the piezoelectric material properties (Figure 4.21) and the list of nodes associated to each pair of electrodes (Figure 4.22) using `p_piezo Tab` calls. Here nodes 1055 to 1058 are associated to the four pairs of electrodes defined in the model. The corresponding degree of freedom is the difference of potential between the electrodes in each pair corresponding to a specific

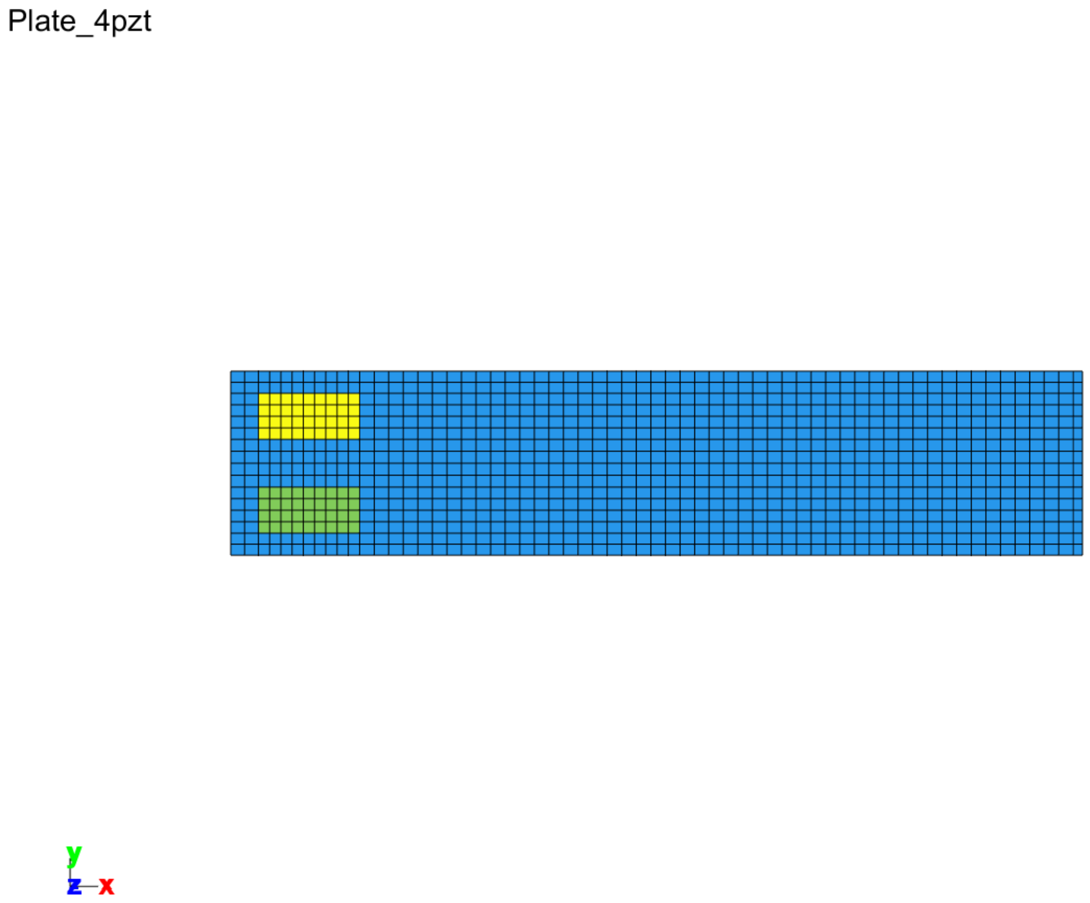
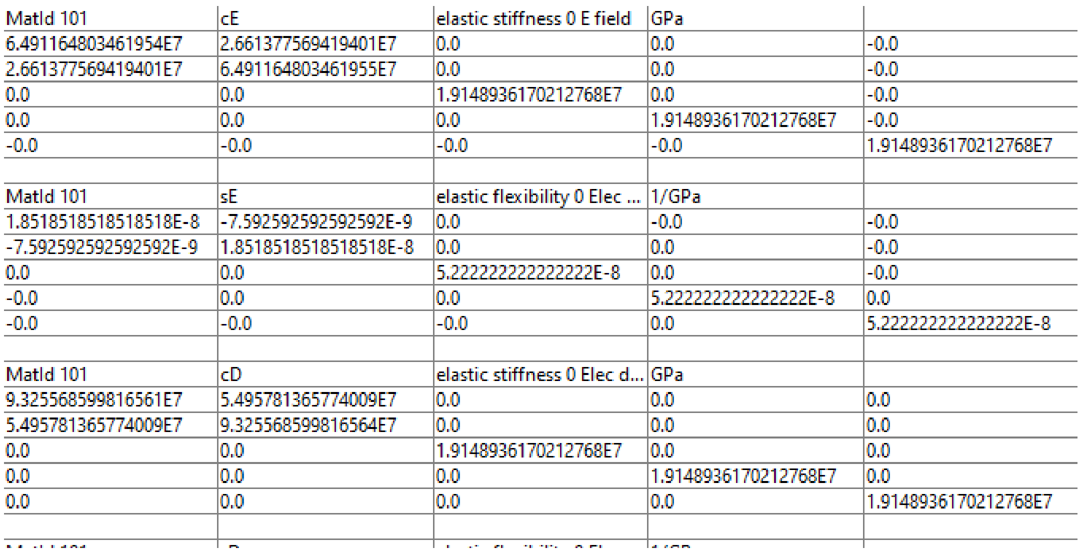


Figure 4.20: Mesh of the composite plate. The different colours represent the different groups

piezoelectric layer. In this models, layers 1 and 3 are piezoelectric in groups 2 and 3 (the internal layer correspond to supporting aluminum plate). Therefore only .21 (electrical) DOF is associated to nodes 1055-1058.

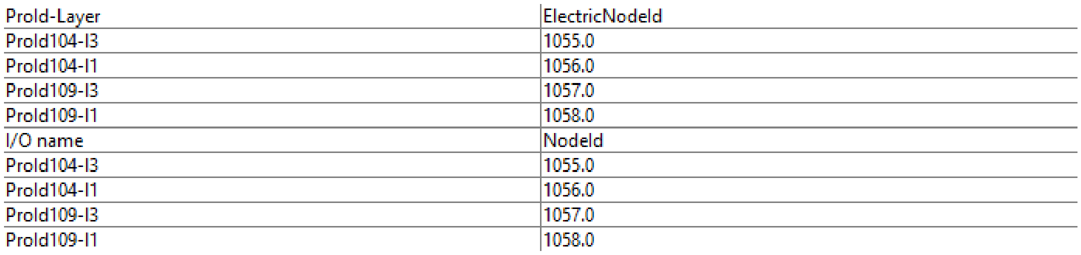
```
p_piezo('TabDD',model);      % List piezo constitutive laws
```



MatId 101	cE	elastic stiffness 0 E field	GPa	
6.491164803461954E7	2.661377569419401E7	0.0	0.0	-0.0
2.661377569419401E7	6.491164803461955E7	0.0	0.0	-0.0
0.0	0.0	1.9148936170212768E7	0.0	-0.0
0.0	0.0	0.0	1.9148936170212768E7	-0.0
-0.0	-0.0	-0.0	-0.0	1.9148936170212768E7
MatId 101	sE	elastic flexibility 0 Elec ...	1/GPa	
1.8518518518518518E-8	-7.592592592592592E-9	0.0	-0.0	-0.0
-7.592592592592592E-9	1.8518518518518518E-8	0.0	0.0	-0.0
0.0	0.0	5.222222222222222E-8	0.0	-0.0
-0.0	0.0	0.0	5.222222222222222E-8	0.0
-0.0	-0.0	-0.0	0.0	5.222222222222222E-8
MatId 101	cD	elastic stiffness 0 Elec d...	GPa	
9.325568599816561E7	5.495781365774009E7	0.0	0.0	0.0
5.495781365774009E7	9.325568599816564E7	0.0	0.0	0.0
0.0	0.0	1.9148936170212768E7	0.0	0.0
0.0	0.0	0.0	1.9148936170212768E7	0.0
0.0	0.0	0.0	0.0	1.9148936170212768E7

Figure 4.21: Example subset of table with the full set of mechanical, dielectric and piezoelectric coefficients in the 4 different forms of the constitutive equations

```
r1=p_piezo('TabInfo',model); % List piezo related properties
```



ProId-Layer	ElectricNodeId
ProId104-l3	1055.0
ProId104-l1	1056.0
ProId109-l3	1057.0
ProId109-l1	1058.0
I/O name	NodeId
ProId104-l3	1055.0
ProId104-l1	1056.0
ProId109-l3	1057.0
ProId109-l1	1058.0

Figure 4.22: Information about electrical master nodes related to each piezoelectric layer. Here layers 1 and 3 for two zones with ProID 104 and 109 define the 4 piezoelectric patches

The next step consists in the definition of the actuators and sensors in the model. Here, we consider one actuator on Node 1055 (layer 3 of group 1), the four piezoelectric patches are used as charge sensors, and the tip displacement of the cantilever beam is measured at the right-upward corner of the beam (corresponding to node 1054 here). Note that in order for Q-S1, Q-S2 and Q-S3 to measure resultant charge, the corresponding electrical difference of potential needs to be set to zero. If this is not done, then the charge sensors will measure a charge close to zero (round-off errors) as there is no charge when the difference of potential across the electrodes is free. For Q-Act, the electrical DOF is already fixed due to the fact that the patch is used as a voltage actuator.

```
%% Step 2 - Define actuators and sensors and visualize
nd=feutil('find node x==463 & y==100',model);
model=fe_case(model,'SensDof','Tip',{[num2str(nd) ':z']}); % Displ sensor
i1=p_piezo('TabInfo',model);i1=i1.Electrodes(:,1);
model=fe_case(model,'DofSet','V-Act',struct('def',1,'DOF',i1(1)+.21, ...%Act
    'Elt',feutil('selelt proid 104',model))); % Elt defined for display
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-Act',i1(1)),model); % Charge sensors
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S1',i1(2)),model);
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S2',i1(3)),model);
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S3',i1(4)),model);
% Fix ElectrodeSensQ dofs to measure resultant (charge)
model=fe_case(model,'FixDof','SC*S1-S3',i1(2:end)+.21);
cf=feplot(model); fecom('view3')
cf.mdl.name='Plate_4pzt'; d_piezo('SetStyle',cf); feplot(cf);
```

We can now visualize the voltage actuator and the tip displacement sensor. Note that to visualize the piezoelectric actuator, it is necessary to associated a group of elements when defining it (see `fe_case('DofSet')` command above).

```
fecom('showfialpha')
fecom('proviewon')

% Arrow length and thickness
sdth.urn('Tab(Cases,Tip){Proview,on,deflen,20}',cf)
sdth.urn('Tab(Cases,Tip){arProp,"linewidth,2"}',cf)
fecom('curtabCase',{'Tip';'V-Act'})

d_piezo('SetStyle',cf); feplot(cf);

% Insert the number for the sensor :
```

```
fecom('textnode',[nd],'fontsize',14)
cf.mdl.name='Plate_4pzt_Vact-Tip'; d_piezo('SetStyle',cf); feplot(cf);
```

Figure 4.23 shows the visualization of the tip sensor and voltage actuator.

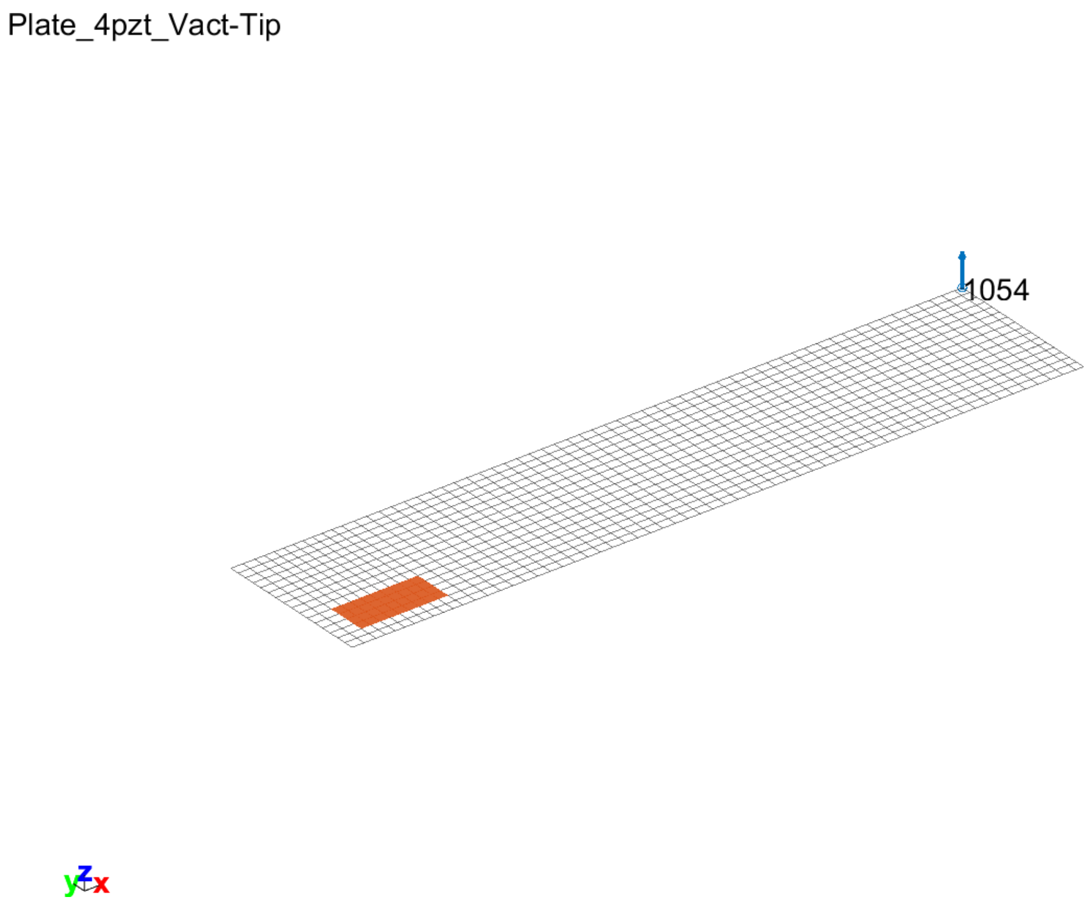


Figure 4.23: Visualization of the tip sensor and voltage actuator

We can also visualize the charge sensors (Figure 4.24). Note however that in the case of plates, it is not possible to distinguish on the visualization if the patch is on the top or the bottom of the plate, so that Q-S2 and Q-S3 would give the same visualisation.

```
fecom('curtabCase',{'Q-S1';'Q-S2'})
cf.mdl.name='Plate_4pzt_QS1-2';d_piezo('SetStyle',cf); feplot(cf);
```

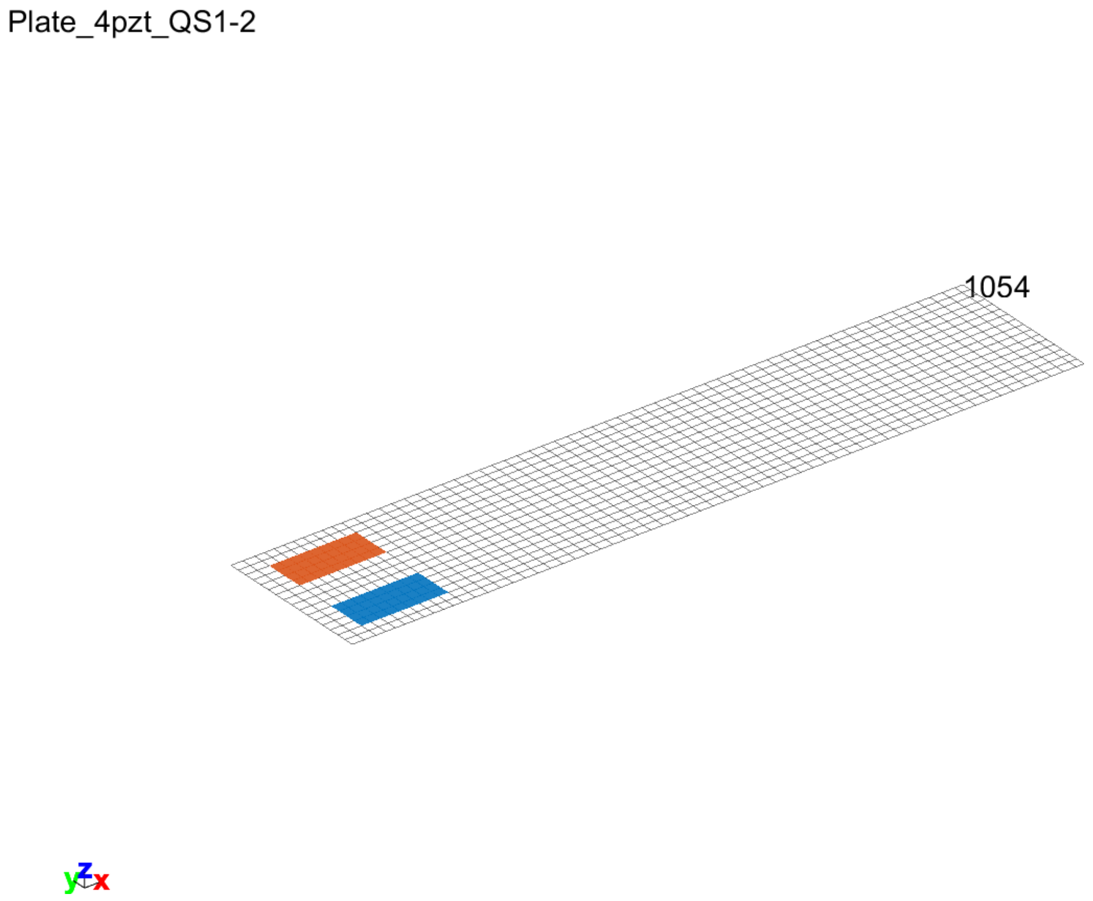


Figure 4.24: Visualization of the 2 charge sensors Q-S1 and Q-S2

In order to check the effect of the actuator, we compute the static response using the full model and represent the deformed shape (Figure 4.25).

(d_piezo('TutoPlate_4pzt_single-s3'))

```
%% Step 3 Compute static and dynamic response
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % direct refer frf at 0Hz
cf=feplot(model,d0); fecom(';view3;scd 20;colordatagroup;undefline')
cf.mdl.name='Plate_4pzt'; d_piezo('setstyles',cf);
```

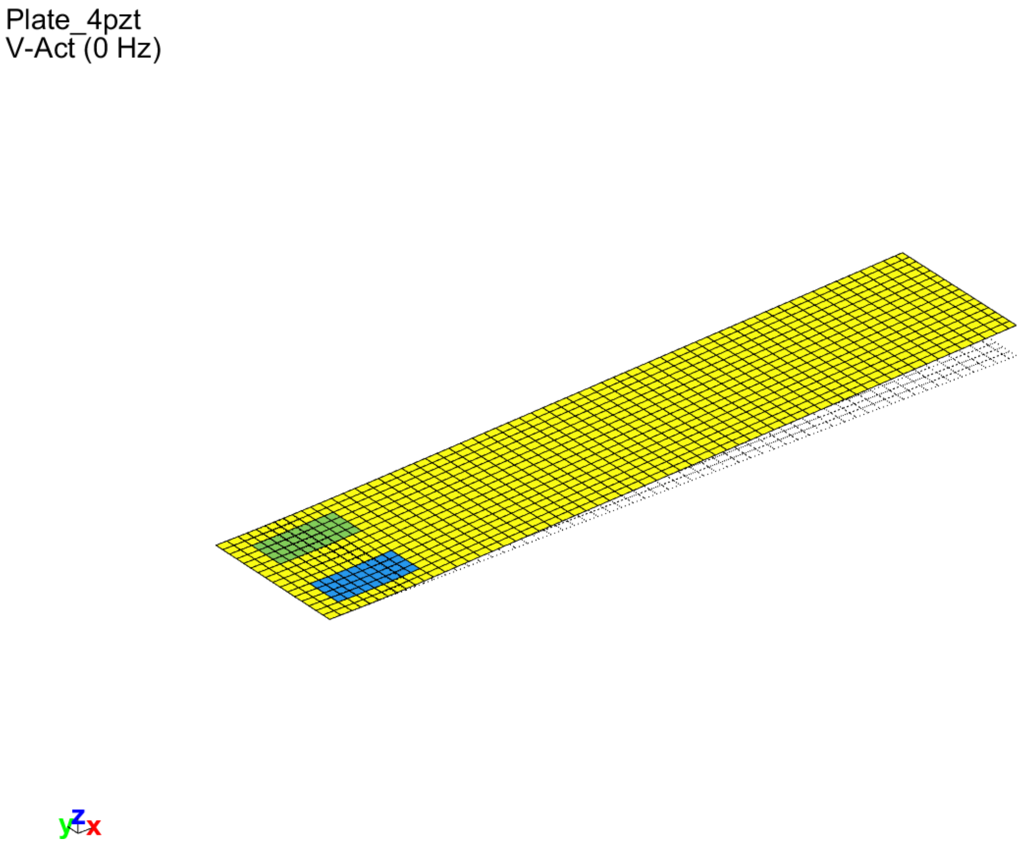


Figure 4.25: Deformed shape under voltage actuation on one of the bottom piezoelectric patches

We can now compute the transfer function between the actuator and the four charge sensors, as well as the tip sensor using the full model. The result is stored in the variable $C1$.

```
f=linspace(1,100,400); % in Hz
d1=fe_simul('dfrf',stack_set(model,'info','Freq',f(:))); % direct refer frf
sens=fe_case(model,'sens');

% Plot FRFs
C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
ci=iiplot;
iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
d_piezo('setstyle',ci);


% End of script
```

Figure 4.26 shows the different transfer functions from the voltage actuator to the tip displacement sensor (left) and to all charge sensors (right). The figure shows that Q-SAct shows an alternance of poles and zeros, but the smallest distance between the poles and zeros. This configuration corresponds to transfer functions used for shunting applications. For QS1 to QS3, the pole-zero alternance is lost due to the fact that the sensor and the actuator are not strictly collocated. The pole-zero distance is also very different, although if we are looking at the structure with the beam theory, the three FRFs should be identical. This demonstrates clearly the need for shell models, and the impact of the location of the sensor on the pole-zero pattern, and pole-zero distances.

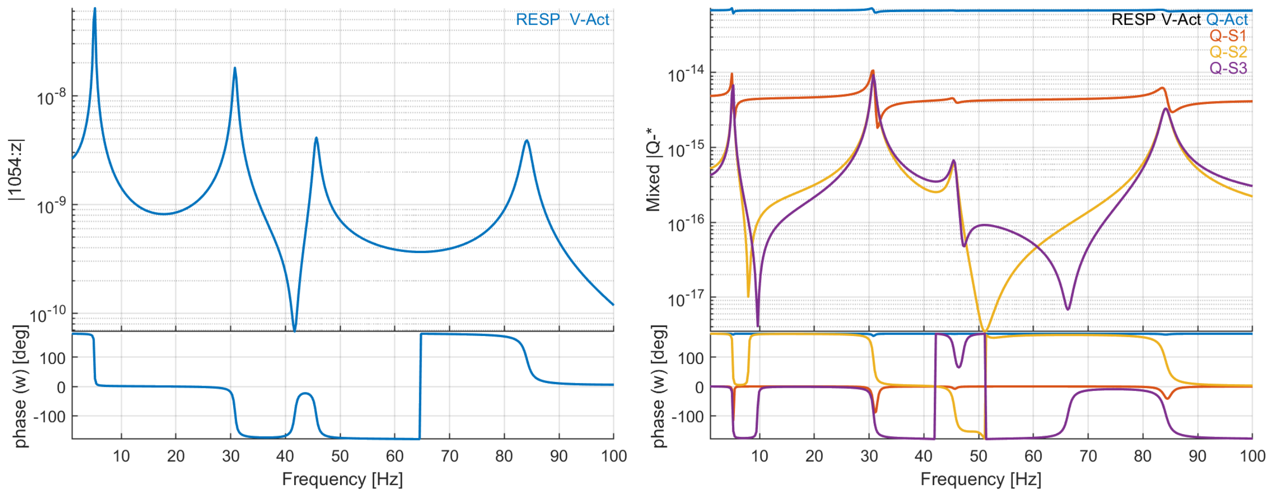


Figure 4.26: Open-loop transfer function between V-Act and tip displacement (left), the 4 charge sensors (right)

4.6.3 Piezoelectric shaker with an accelerometer mounted on top (3D model)

The second example deals with an accelerometer (sensor) mounted on a piezoelectric shaker (actuator). The piezoelectric shaker consists of two steel cylindrical parts with a piezoelectric disc inserted in between. The base of the shaker is fixed and the piezoelectric element is used as an actuator: imposing a voltage difference between the electrodes results in the motion of the top surface of the shaker to which the accelerometer is attached (Figure 4.27).

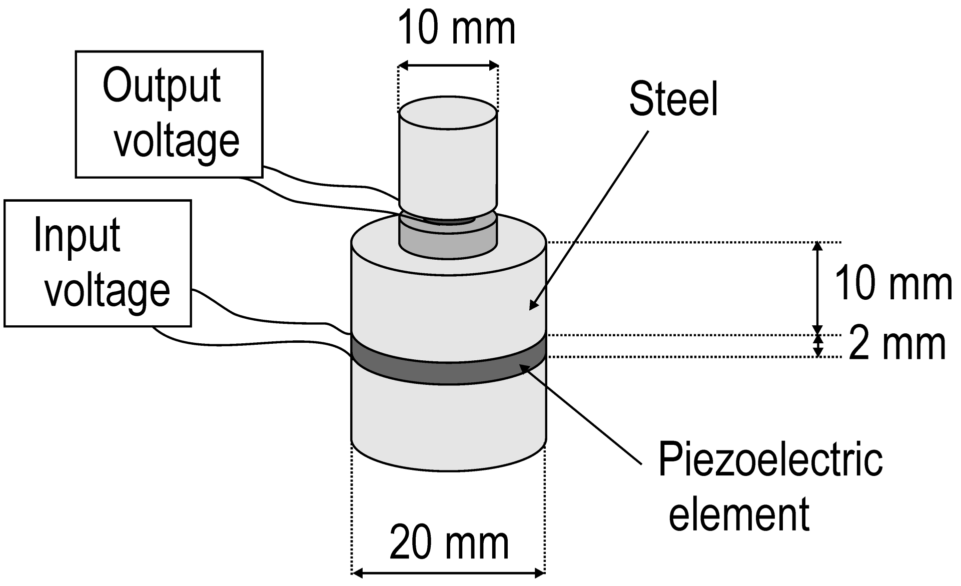


Figure 4.27: Piezoelectric accelerometer attached to a piezoelectric shaker for sensor calibration

The piezoelectric properties for the sensing element in the piezoelectric shaker are given in Table 4.3. The actuating element has the same properties as the sensing element and is poled through the thickness, as the actuator.

Part	Material	E (GPa)	ρ (kg/m^3)	ν
Wear plate	Al_2O_3	400	3965	0.22
Sensing element	Piezo	54	7740	0.44
Proof mass	Steel	210	7800	0.3

Table 4.2: Mechanical properties of the wear plate, sensing element and proof mass

Property	Value
$d_{31} = d_{32}$	-185 $10^{-12} pC/N$ (or m/V)
d_{33}	440 $10^{-12} pC/N$ (or m/V)
$d_{15} = d_{24}$	560 $10^{-12} pC/N$ (or m/V)
$\varepsilon^T_{33} = \varepsilon^T_{22} = \varepsilon^T_{11}$	1850 ε_0
ε_0	8.854 $10^{-12} Fm^{-1}$

Table 4.3: Piezoelectric properties of the sensing element

```
% Init working directory for figure generation
d_piezo('SetPlotwd');


% See full example as MATLAB code in d_piezo('ScriptTutoPzAccShaker')
d_piezo('DefineStyles');
%% Step 1 - Build mesh and visualize
% Meshing script,open with sdtweb d_piezo('MeshPiezoShaker')
model=d_piezo('MeshPiezoShaker');
cf=feplot(model); fecom('colordatapro');
```

The mesh is represented in Figure 4.28. In the meshing script, both a voltage and a charge sensor are defined for the piezoelectric disk in the accelerometer for the top electrode, and the the bottom electrode potential is set to zero. A voltage actuator is then added to the shaker top electrode, while the bottom electrode potential is set to 0. The shaker is mechanically fixed at the bottom:

```
%% Step 2 - Define actuators and sensors
  % -input "In" says it will be used as a voltage actuator
model=p_piezo('ElectrodeMPC Top Actuator -input "Vin-Shaker"',model,'z==-0.01');
  % -ground generates a v=0 FixDof case entry
model=p_piezo('ElectrodeMPC Bottom Actuator -ground',model,'z==-0.012');
% Voltage sensor will be used - remove charge sensor
model=fe_case(model,'remove','Q-Top sensor');
```

The different electrodes in the model can be visualized (Figure 4.29):

```
% Visualize electrodes
fecom(';showline;proviewon')
fecom('curtabCase',{'Top sensor';'Bottom sensor';'Top Actuator';'Bottom Actuator'}) %
```

And each actuator/sensor can be visualized separately:

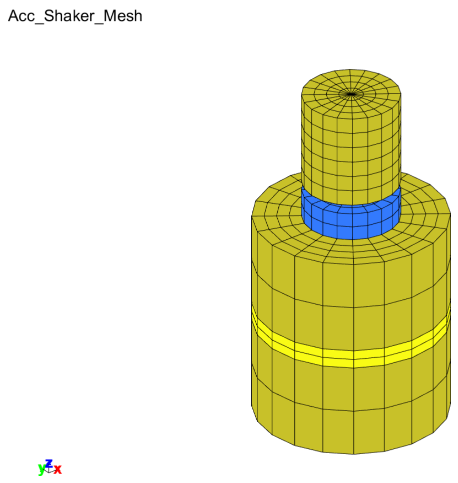


Figure 4.28: Mesh of the piezoelectric accelerometer attached to a piezoelectric shaker

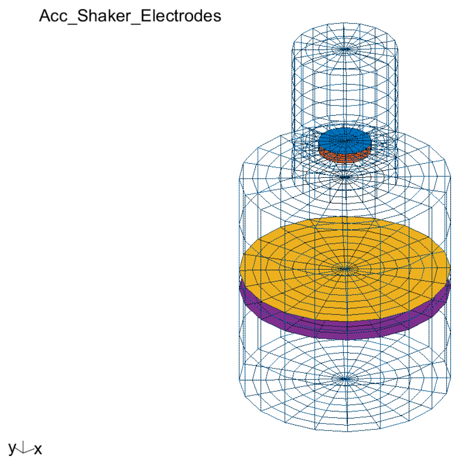


Figure 4.29: Top and bottom electrodes for shaker voltage actuator and accelerometer voltage sensor

```
% Visualize Vin electrode
fecom curtabcases Vin-Shaker %


% Visualize VSens electrode
fecom curtabcases 'V-Top sensor'
```

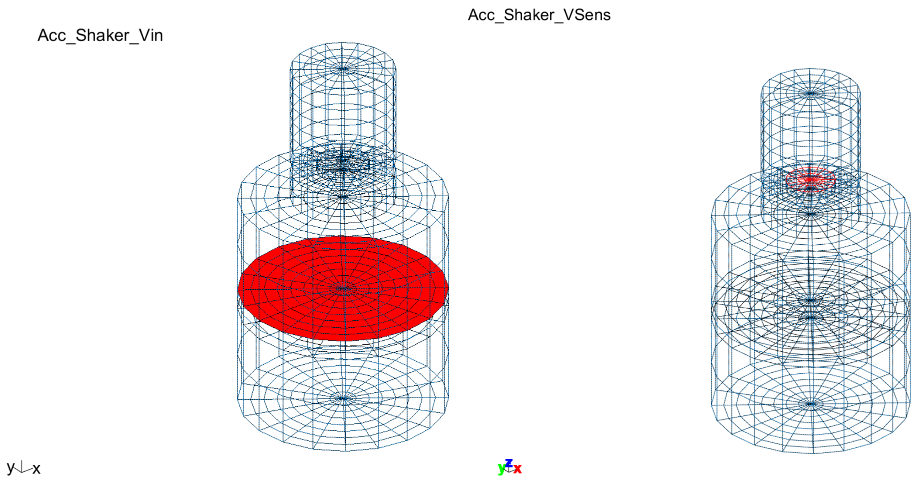
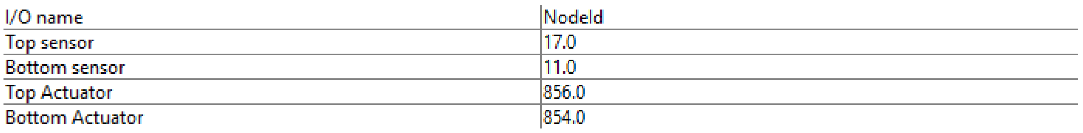


Figure 4.30: Shaker input voltage(left) and accelerometer output voltage(right)

We can have an overview of the electrodes and associated electrical dofs (Figure 4.31) :

```
r1=p_piezo('TabInfo',model); % List piezo related properties
```



I/O name	NodeId
Top sensor	17.0
Bottom sensor	11.0
Top Actuator	856.0
Bottom Actuator	854.0

Figure 4.31: Electrodes and associated nodes/dofs

After meshing, the script to obtain the sensitivity (voltage output of the sensor - electrical dof on Node 17, divided by imposed voltage on piezoelectric shaker - electrical dof on Node 856) is given below and leads to the result presented on Figure 4.32

```
ofact('silent'); f=logspace(3,5.3,400)';
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
d1=fe_simul('dfrf',stack_set(model,'info','Freq',f(:))); % direct refer frf

% Project on sensor and create output
 sens=fe_case(model,'sens');
 C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
 C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');
 ci=iiplot;
 iicom(ci,'curveInit',C1.name,C1); iicom('submagpha');
 d_piezo('setstyle',ci);


% End of script
```

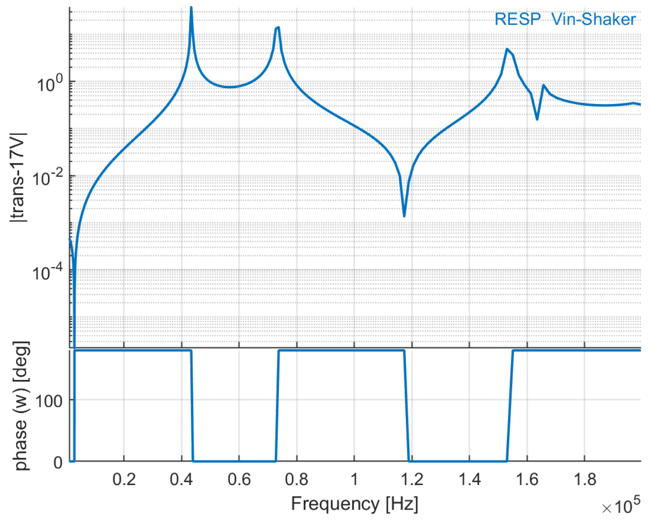


Figure 4.32: Sensor sensitivity (VSens/Base displ)

Methods for meshing plates with piezoelectric patches

Contents

This section explains how to mesh plates with piezoelectric patches of different shapes. In all cases, multi-layer plate elements are used to represent the host plate and the additional piezoelectric layers. The first (basic) strategy consists in assigning additional layers with given piezoelectric properties to regions of the mesh. The second approach uses a script to remesh locally the plate and add piezoelectric layers according to a predefined set of patch shapes and piezoelectric properties.

5.1 Manual meshing

We consider again the example treated in section 4.6.2 , represented in Figure 4.19. The material properties of the aluminum plate and the piezoceramic patches are given in Table 4.1. The thickness of the plate is $1.2mm$, and the piezoelectric material corresponds to the *SONOX_P502_iso* material in `m_piezo Database`.

The first step for manual meshing consists in meshing the plate as follows:

```
% See full example in d_piezo('ScriptTutoPlateMeshingBasics')
d_piezo('DefineStyles');

%% Step 1 - Mesh the plate
 model=struct('Node',[1 0 0 0 0 0 0],'Elt',[]);
 model=feutil('addelt',model,'mass1',1);
 % Note that the extrusion values are chosen to include the patch edges
 dx=[linspace(0,15,3) linspace(15,15+55,10) linspace(15+55,463-5,50) 463];
 model=feutil('extrude 0 1 0 0',model,.001*unique(dx));
 dy=[linspace(0,12,3) linspace(12,12+25,5) linspace(12+25,63,5) ...
    linspace(63,63+25,5) linspace(63+25,100,3)];
 model=feutil('extrude 0 0 1 0',model,.001*unique(dy));
```

Note that the meshing is such that the patch edges corresponds to limits of elements in the mesh. It is then possible to divided the mesh in different groups, related to the two areas where the patches are added, and the host structure with no piezoelectric properties. Then a different ProID is set for the regions with piezoelectric patches.

```
 %% Step 2 - Set patch areas and set different properties
 model.Elt=feutil('divide group 1 withnode{x>.015 & x<.07 & y>.013 & y<.037 }',model);
 model.Elt=feutil('divide group 2 withnode{x>.015 & x<.07 & y>.064 & y<.088 }',model);
 model.Elt=feutil('set group 1 proId3',model);
 model.Elt=feutil('set group 2 proId4',model);
```

Visualizing the mesh and using `fe_com('colordatapro')` (see Figure 5.1) allows to check that different properties have been assigned to the two regions where piezoelectric patches need to be added.

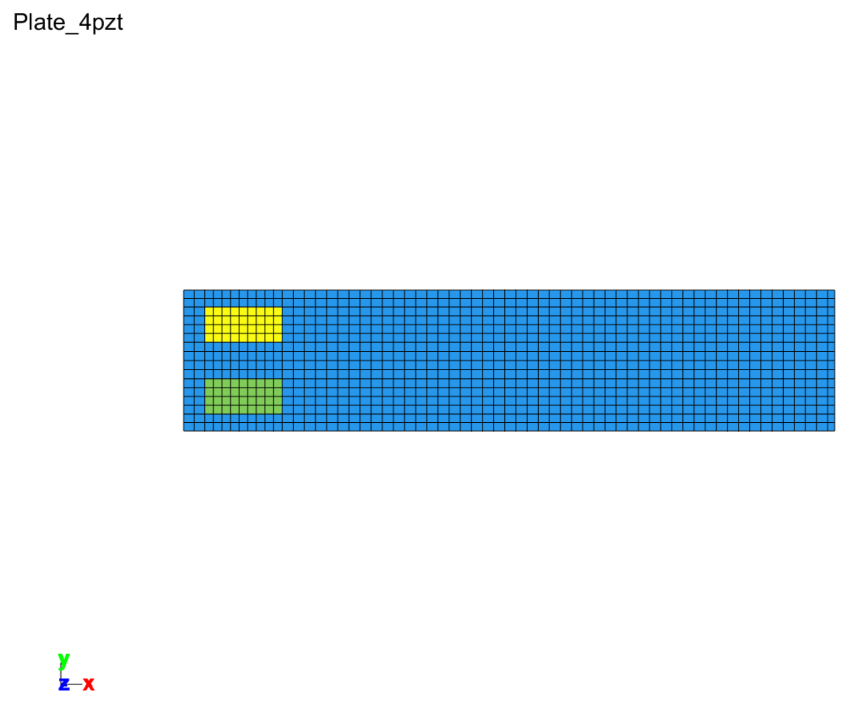


Figure 5.1: Mesh of the composite plate. The different colours represent the different groups

The next step is to define the material properties for the host structure and the patches. Here, `Sonox_P502_iso` is used:

```
%% Step 3 - Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
model.pl=m_piezo(model.pl,'dbval 3 -elas2 SONOX_P502_iso');
% To avoid warning due to the use of simplified piezo properties.
model=p_piezo('DToSimple',model)
```

Piezoelectric material properties are divided in two parts. The first one (ProId 3 here) contains the piezoelectric data, and is linked to elastic properties with ProID 2 in this example through the `-elas2` command. If the piezoelectric properties do not exist in the database, it is always possible to introduce them by hand with the following commands:

```
  model.pl=m_elastic('dbval 1 Aluminum');
  d=zeros(1,18); d([11 13])=560e-12; d([3 6])=-185e-12; d(9)=440e-12;
  eps=zeros(1,9); eps([1 5 9])= 8.854e-12*1850;
%                                      elasid |dij coeff  | dielectric coeffs
  model.pl=[ model.pl zeros(1,24);
           3 fe_mat('m_piezo','SI',2) 2        d           eps;
           2 fe_mat('m_elastic','SI',1) 54e9 0.41 7740 zeros(1,25)];
```

The piezoelectric and dielectric coefficients are stored in a vector of 18 (3x6 matrix) and 9 (3x3 matrix) values respectively. For more information see `m_piezo 2:General 3D piezo`.

Now that the material properties have been defined, it is necessary to create laminates with three layers, the top and bottom layers being made of the piezoelectric material (ProID 3), of thickness $0.25mm$ and the middle layer being the host structure made of aluminum with thickness $1.2mm$.

```
%% Step 4 - Laminate properties and piezo electrodes
model.il=p_shell('dbval 1 laminate 1 1.2e-3 0', ...
  'dbval 2 laminate 3 2.5e-4 0 1 1.2e-3 0 3 2.5e-4 0');
```

The last step consists in assigning electrodes to each patch, and an electrical dof id. This single dof represents the difference of voltage between the top and bottom electrode and is the same for all elements in the piezoelectric patch, as the electrodes impose equipotentiality. Here for example, dof 1682.21 is assigned to the first layer of group 1, and dof 1683.21 to the third layer of the same group.

```
%%% Piezo electrodes
%%%                                         NdNb LayerID   NdNb  LayerID
model.il=p_piezo(model.il,'dbval 3 shell 2 1682    1   0    1683  3 0');
model.il=p_piezo(model.il,'dbval 4 shell 2 1684    1   0    1685  3 0');
```

We can now check that the patches are implemented properly by computing the static response to an applied voltage to the patch in group 1 (bottom). The layers are numbered from bottom to top, so it corresponds to dof 1682.21. Note that in order to find the top and bottom of an element for more complex (curved) meshes, it is possible to use the following command to show the orientation of the normal of the elements (Figure 5.2):

```
%% Step 5 - show orientation of the normal
cf=feplot(model); fecom('showmap'); fecom('view3');
% scale properly
fecom('scalecoeff 1e-10'); fecom('showmap')
cf.mdl.name='Plate_4pzt_Normal_Orient'; d_piezo('SetStyle',cf); feplot(cf);
```

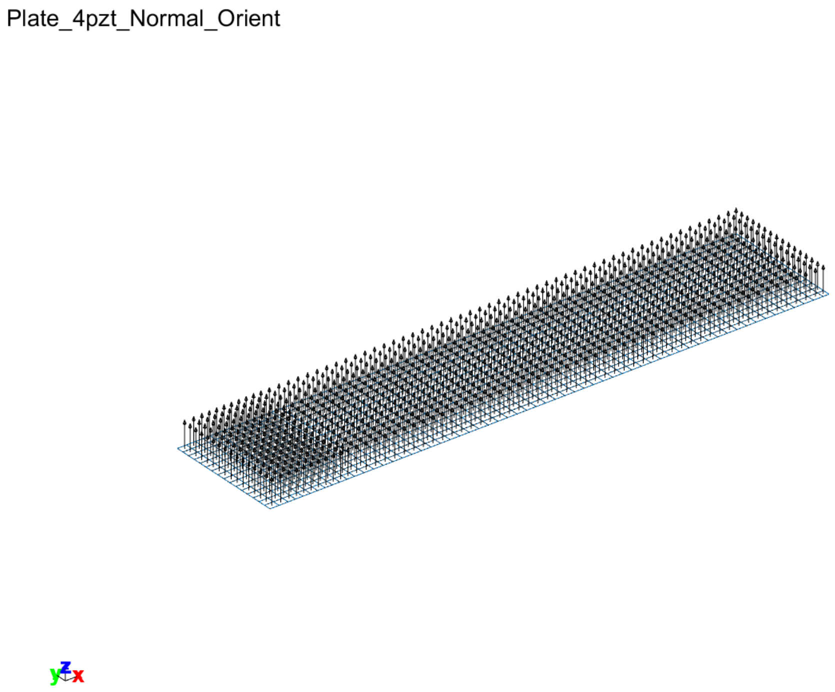


Figure 5.2: Mesh of the plate with arrows showing the orientation of the normals to the elements

The electrical dof corresponds to the difference of electrical potential between the top and bottom electrodes, so if one applies a difference of potential of $1V$, it results in a negative electric field which is in the opposite direction of the poling direction (always in the positive z (normal) direction for a plate element). Because the d_{31} and d_{32} are negative coefficient, the resulting strain is positive (negative electric field multiplied by negative constant). A positive strain at the bottom of the plate should result in an upward motion of the tip of the beam, which is verified (Figure 5.3).

```
%% Step 6 - Compute and display response to static imposed voltage
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');
model=fe_case(model,'DofSet','V-Act',struct('def',1,'DOF',1682.21)); %Act
model=p_piezo('ElectrodeSensQ  1683 Q-S1',model);
model=p_piezo('ElectrodeSensQ  1684 Q-S2',model);
model=p_piezo('ElectrodeSensQ  1685 Q-S3',model);
model=fe_case(model,'SensDof','Tip',1054.03); % Displ sensor top right corner
sens=fe_case(model,'sens');
model=fe_case(model,'FixDof','SC*1683-1685',[1682:1685]+.21);
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % direct refer frf at 0Hz
cf=feplot(model,d0); fecom(';view3;colordatagroup;undefline');
cf.mdl.name='Plate_4pzt'; d_piezo('SetStyle',cf);
d=sens.cta(4,:)*d0.def % Tip displacement is positive.
```

The script above also defines three charge sensors on dofs 1683.21, 1684.21 and 1685.21. If the electrodes are not short-circuited, it would result in a zero-charge measured, so the electrical dofs are set to zero for these three patches so that a resultant charge can be measured.

To implement a piezoelectric patch on a single side of the plate, it is important to treat properly the offset, as the layer sequence is not symmetric with respect to the neutral plane of the plate. This is done by using the *z_0* parameter when defining the laminates. z_0 corresponds to the distance from the mid-plane to the bottom of the plate (see `sdtweb('p_shell')`). The following script is the same as the previous one but with piezoelectric patches only on the bottom of the plate, in which case the offset should be $z_0 = -0.6mm - 0.25mm = -0.85mm$ where $0.6mm$ is half the thickness of the support plate and $0.25mm$ is the thickness of the piezo.

```
%% Step 7 - Piezoelectric patches on the bottom only
 model=struct('Node',[1 0 0 0 0 0 0],'Elt',[]);
 model=feutil('addelt',model,'mass1',1);
 % Note that the extrusion values are chosen to include the patch edges
 dx=[linspace(0,15,3) linspace(15,15+55,10) linspace(15+55,463-5,50) 463];
 model=feutil('extrude 0 1 0 0',model,.001*unique(dx));
 dy=[linspace(0,12,3) linspace(12,12+25,5) linspace(12+25,63,5) ...
```

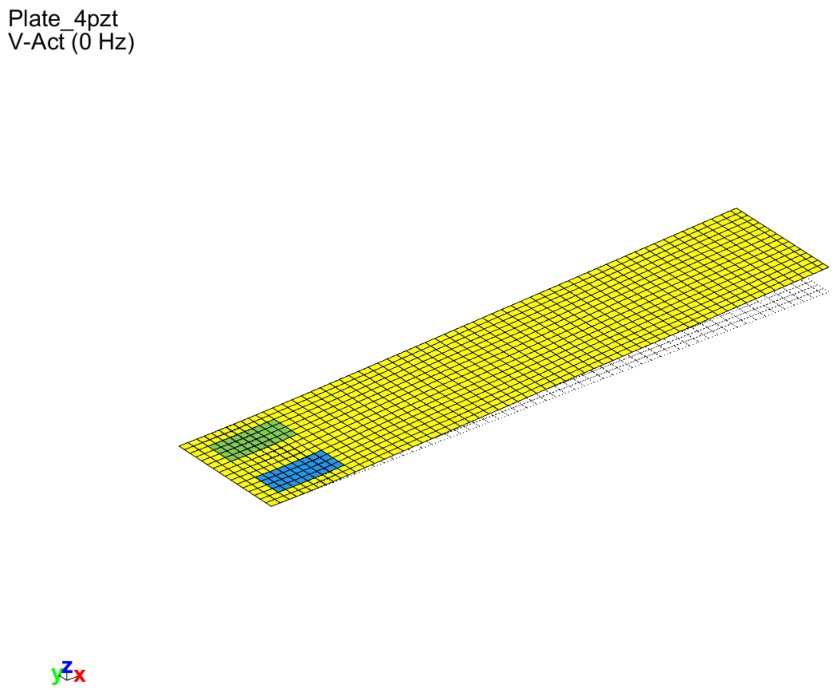


Figure 5.3: Static response to a unit voltage application on one of the bottom piezoelectric patches

```
    linspace(63,63+25,5) linspace(63+25,100,3)];
 model=feutil('extrude 0 0 1 0',model,.001*unique(dy));

% Set patch areas as set different properties
 model.Elt=feutil('divide group 1 withnode{x>.015 & x<.07 & y>.013 & y<.037 }',model);
 model.Elt=feutil('divide group 2 withnode{x>.015 & x<.07 & y>.064 & y<.088 }',model);
 model.Elt=feutil('set group 1 proId3',model);
 model.Elt=feutil('set group 2 proId4',model);

%%  Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
model.pl=m_piezo(model.pl,'dbval 3 -elas2 SONOX_P502_iso');
% To avoid warning due to the use of simplified piezo properties.
model=p_piezo('DToSimple',model)

%%  Laminate properties and piezo electrodes
% This is where an offset must be specified
%(else z0=-7.25e-4 (half of total thickness which is not correct).
model.il=p_shell('dbval 1 laminate 1 1.2e-3 0', ...
   'dbval 2 laminate z0=-8.5e-4 3 2.5e-4 0 1 1.2e-3 0 ');

%%% Piezo electrodes
%%%                                         NdNb LayerID
model.il=p_piezo(model.il,'dbval 3 shell 2 1682    1   0    ');
model.il=p_piezo(model.il,'dbval 4 shell 2 1684    1   0    ');

%%  Static response computation
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');
model=fe_case(model,'DofSet','V-Act',struct('def',1,'DOF',1682.21)); %Act
model=p_piezo('ElectrodeSensQ  1684 Q-S2',model);
model=fe_case(model,'SensDof','Tip',1054.03); % Displ sensor top right corner
sens=fe_case(model,'sens');
model=fe_case(model,'FixDof','SC*1682-1684',[1682 1684]+.21);
d0=fe_simul('dfrf',stack_set(model,'info','Freq',0)); % direct refer frf at 0Hz
feplot(model,d0); fecom(';view3;colordatagroup;undefline');
d2=sens.cta(2,:)*d0.def % Tip displacement is positive.
disp(['difference of static response' num2str((d2-d)/d*100) '%'])
```

The results show that the plate with only one piezoelectric element on the bottom is less stiff then the case with piezos on both the top and bottom: the tip displacement is 25% higher (3.17 $\mu m/V$

vs 2.52 $\mu m/V$) .

When the piezoelectric element is at the top of the plate, then the offset should be equal to the thickness of the main plate divided by two (negative value).

5.2 Automated inclusion of piezo patches

The automated procedure requires the plate model to be defined in mm. In order to add the patches, one needs to make a structure which contains the initial plate model and a description of the different patches to be added. In this example, the patches are assigned with `+Rect.Sonox_P502_iso.5525TH0_25` where the sign +/- specifies if the patch is on the top (+) or the bottom (-) of the plate, the first argument gives the geometry (Rect for Rectangular), the second argument is the piezoelectric material type from the database, the next argument is the size (55mm x 25mm here) and the last argument is the thickness (0.25mm). So the first step is to make the host plate. Here, we define a mesh so that the edges of the piezoelectric patches will correspond to the mesh (as in the previous example of meshing manually).

```
% See full example in d_piezo('ScriptTutoPlateMeshingAuto')
d_piezo('DefineStyles');

%% Step 1 : model of host plate -
 model=struct('Node',[1 0 0 0 0 0 0],'Elt',[]);
 model=feutil('addelt',model,'mass1',1);
 % Note that the extrusion values are chosen to include the patch edges
 dx=[linspace(0,15,3) linspace(15,15+55,10) linspace(15+55,463-5,50) 463];
 model=feutil('extrude 0 1 0 0',model,unique(dx));
 dy=[linspace(0,12,3) linspace(12,12+25,5) linspace(12+25,63,5) ...
    linspace(63,63+25,5) linspace(63+25,100,3)];
 model=feutil('extrude 0 0 1 0',model,unique(dy));
model.unit='mm';

% Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
% Laminate properties
model.il=p_shell('dbval 1 -punit mm laminate 1 1.2 0') % this is to specify in mm
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');
%
```

The model should is defined in *mm* to use the automated procedure. Then we will add four rectangular piezoelectric patches whose size and shape comply with the mesh.

```
%% Step 2: Add patches
RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',12+25/2,'alpha',0,'tolE',.1)
    'Act2', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',63+25/2,'alpha',0,'tolE',.1)
       };
mo1=d_piezo('MeshPlate',RG);
mo1=stack_rm(mo1,'info','Electrodes'); % Obsolete stack field to be removed
% To avoid warning due to the use of simplified piezo properties.
mo1=p_piezo('DToSimple',mo1)
```

The command `+Rect.Sonox_P502_iso.5525TH0_25` refers to a patch which is added on the top of the plate (+), is rectangular (Rect), is made of Sonox_P502_iso, is of size $55mmx25mm$ (5525) and thickness $0.25mm$ (TH0_25). Then we need to specify the position of the patch with 'xc' and 'yc' which correspond to the coordinates of its center. The model obtained is here the same as the one previously obtained with manual meshing. The static response when actuating the first piezo patch is then computed:

```
%% Step 3 : compute response
nd=feutil('find node x==463 & y==100',model);
elnd=floor(p_piezo('electrodedof.*',mo1)); % Nodes associated to electrodes
mo1=fe_case(mo1,'SensDof','Tip',nd+.03); % Displ sensor
mo1=fe_case(mo1,'DofSet','V-Act',struct('def',1,'DOF',elnd(1)+.21)); %Act
mo1=p_piezo(['ElectrodeSensQ  ' num2str(elnd(1)) ' Q-Act'],mo1); % Charge sensors
mo1=p_piezo(['ElectrodeSensQ ' num2str(elnd(2)) ' Q-S1'],mo1);
mo1=p_piezo(['ElectrodeSensQ ' num2str(elnd(3)) ' Q-S2'],mo1);
mo1=p_piezo(['ElectrodeSensQ ' num2str(elnd(4)) ' Q-S3'],mo1);

% Fix last 3 elec dofs to measure resultant (charge)
mo1=fe_case(mo1,'FixDof', ...
['SC*' num2str(elnd(2)) '/' num2str(elnd(3)) '/' num2str(elnd(4)) ], ...
elnd([2:4])+.21);
sens=fe_case(mo1,'sens');
```

```
d1=fe_simul('dfrf',stack_set(mo1,'info','Freq',0)); % direct refer frf at 0Hz
d1t=sens.cta(1,:)*d1.def; % Extract tip displ
feplot(mo1,d1);
fecom('colordatapro'); fecom('view3');
```

Note that the automated meshing introduces the patches in the order of the description, so that here the first piezoelectric patch is on the bottom (-), and the static response is downwards, as in the case of manual meshing. This allows to compared the displacement at the tip which has the same value (2.52 $\mu m/V$).

Automated meshing can also be done when the patch edges do not coincide with the mesh of the host plate. In this case, automated local remeshing is applied around the patches inserted. We take the example of an initial mesh of the plate with a uniform mesh of prescribed element size (here around 7 mm). The script to add the four patches is strictly identical, but the result is different due to the local remeshing.

```
%% Step 4 : use local remeshing element size is 7 mm
 model=feutil('objectquad 1 1',[0 0 0;1 0 0;0 1 0], ...
    feutil('refineline 7',[0 463]), ...
    feutil('refineline 7',[0 100]));
     model.unit='mm';
% Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
% Laminate properties
model.il=p_shell('dbval 1 -punit mm laminate 1 1.2 0') % this is to specify in mm
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');

%% Add patches
RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',12+25/2,'alpha',0,'tolE',.1)
    'Act2', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',63+25/2,'alpha',0,'tolE',.1)
       };
mo2=d_piezo('MeshPlate',RG);
mo2=stack_rm(mo2,'info','Electrodes'); % Obsolete stack field to be removed
```

```
% To avoid warning due to the use of simplified piezo properties.
mo2=p_piezo('DToSimple',mo2)

nd=feutil('find node x==463 & y==100',model);
elnd=floor(p_piezo('electrodedof.*',mo2)); % Nodes associated to electrodes
mo2=fe_case(mo2,'SensDof','Tip',nd+.03); % Displ sensor
mo2=fe_case(mo2,'DofSet','V-Act',struct('def',1,'DOF',elnd(1)+.21)); %Act
mo2=p_piezo(['ElectrodeSensQ  ' num2str(elnd(1)) ' Q-Act'],mo2); % Charge sensors
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(2)) ' Q-S1'],mo2);
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(3)) ' Q-S2'],mo2);
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(4)) ' Q-S3'],mo2);

% Fix last 3 elec dofs to measure resultant (charge)
mo2=fe_case(mo2,'FixDof', ...
['SC*' num2str(elnd(2)) '/' num2str(elnd(3)) '/' num2str(elnd(4)) ], ...
elnd([2:4])+.21);
sens=fe_case(mo2,'sens');

d2=fe_simul('dfrf',stack_set(mo2,'info','Freq',0)); % direct refer frf at 0Hz
d2t=sens.cta(1,:)*d2.def;

cf=feplot(mo2,d2);
fecom(';colordatapro;view3;undef line');
d_piezo('SetStyle',cf); feplot(cf);
[d1t d2t]
```

The values of $d1t$ and $d2t$ correspond to the tip displacement when actuating the first piezo patch, for the first model where the edges of the piezos correspond to the element sides in the mesh, and when there is a local remeshing. Due to the effect of the local remeshing (Figure 5.4), there is a slight difference ($\approx 1\%$). In order to check this effect, we can use a finer mesh of the host structure and check for the convergence.

```
%% Step 5 : use a finer mesh to check convergence
ref=[5 3 2];
dt=[d2t];

for ij=1:length(ref)

 model=feutil('objectquad 1 1',[0 0 0;1 0 0;0 1 0], ...
    feutil(['refineline ' num2str(ref(ij))],[0 463]), ...
    feutil(['refineline ' num2str(ref(ij))],[0 100]));
```

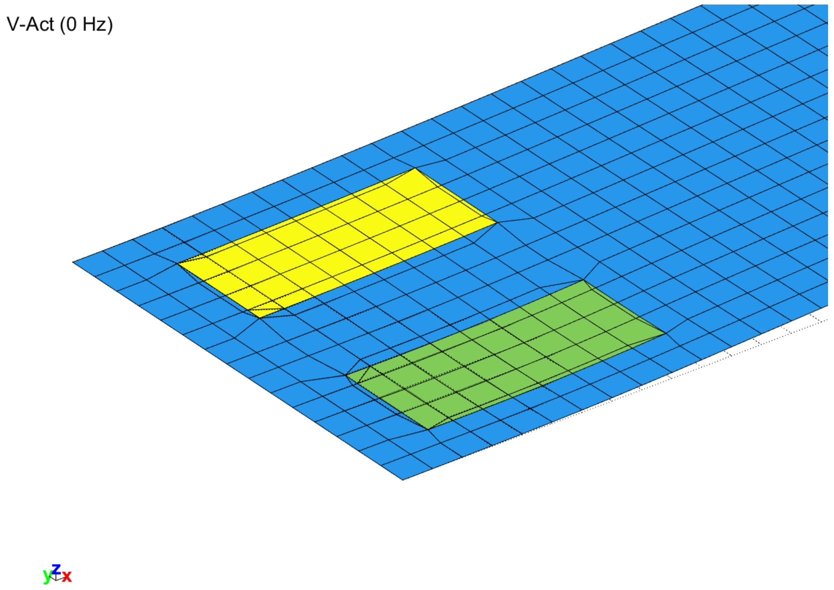


Figure 5.4: Static response to a unit voltage application on one of the bottom piezoelectric patches

```
     model.unit='mm';
% Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
% Laminate properties
model.il=p_shell('dbval 1 -punit mm laminate 1 1.2 0') % this is to specify in mm
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');

%% Add patches
RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',12+25/2,'alpha',0,'tolE',.1)
    'Act2', ... % name of patch
   'BaseId1 -Rect.Sonox_P502_iso.5525TH0_25 +Rect.Sonox_P502_iso.5525TH0_25', ...
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',15+55/2,'yc',63+25/2,'alpha',0,'tolE',.1)
       };
```

```
mo2=d_piezo('MeshPlate',RG);
mo2=stack_rm(mo2,'info','Electrodes'); % Obsolete stack field to be removed
% To avoid warning due to the use of simplified piezo properties.
mo2=p_piezo('DToSimple',mo2)

nd=feutil('find node x==463 & y==100',model);
elnd=floor(p_piezo('electrodedof.*',mo2)); % Nodes associated to electrodes
mo2=fe_case(mo2,'SensDof','Tip',nd+.03); % Displ sensor
mo2=fe_case(mo2,'DofSet','V-Act',struct('def',1,'DOF',elnd(1)+.21)); %Act
mo2=p_piezo(['ElectrodeSensQ  ' num2str(elnd(1)) ' Q-Act'],mo2); % Charge sensors
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(2)) ' Q-S1'],mo2);
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(3)) ' Q-S2'],mo2);
mo2=p_piezo(['ElectrodeSensQ ' num2str(elnd(4)) ' Q-S3'],mo2);

% Fix last 3 elec dofs to measure resultant (charge)
mo2=fe_case(mo2,'FixDof', ...
['SC*' num2str(elnd(2)) '/' num2str(elnd(3)) '/' num2str(elnd(4)) ], ...
elnd([2:4])+.21);
sens=fe_case(mo2,'sens');

d2=fe_simul('dfrf',stack_set(mo2,'info','Freq',0)); % direct refer frf at 0Hz
d2t=sens.cta(1,:)*d2.def;

dt=[dt; d2t];
end

gf=figure; plot([7 ref],1e6*dt,'linewidth',2); set(gca, 'XDir','reverse');
set(gca,'Fontsize',15); v=get(gca,'XLim'); hold on;
plot(v,1e6*[d1t d1t],'r','linewidth',2);
legend('Local remeshing','Conforming mesh')
xlabel('mesh size (mm)'); ylabel('tip displacement (mm/V)')
```

We see that when the plate mesh is finer (Figure 5.5), the local effect of remeshing is less and we converge to the tip displacement when the piezoelectric patch edges coincide with the mesh.

A last example is the addition of circular patches, where we also check for convergence. The introduction of circular patches is done with the arguments `-Disk.Sonox_P502_iso.RC10TH0_25` and `+Disk.Sonox_P502_iso.RC10TH0_25` which specifies a disk made of *Sonox_P502_iso* material with a radius RC of 10mm and a thickness TH of 0.25 mm (same thickness as the rectangular patches).

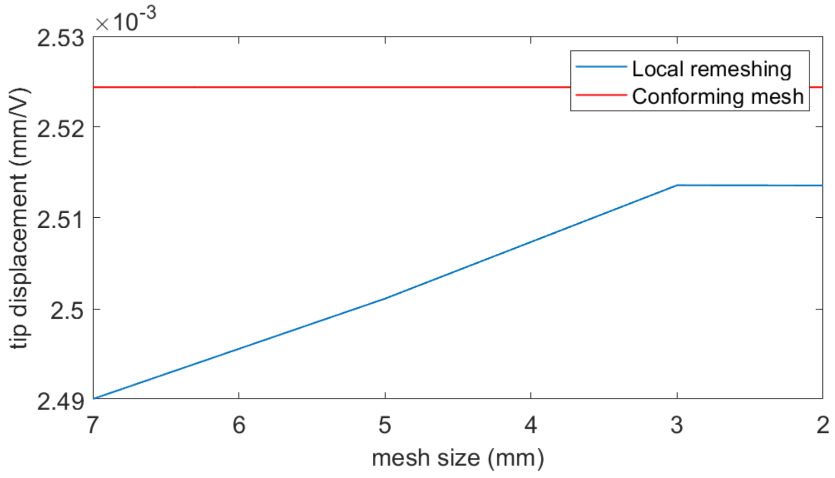


Figure 5.5: Convergence of the tip displacement when the size of the elements of the main plate is varied, using local remeshing. The red line corresponds to the tip displacement with the mesh conforming with the patch edges

```
%% Step 6 : with circular patches
  model=feutil('objectquad 1 1',[0 0 0;1 0 0;0 1 0], ...
    feutil('refineline 7',[0 463]), ...
    feutil('refineline 7',[0 100]));
     model.unit='mm';

%%%%%% Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
%%%%% Laminate properties
model.il=p_shell('dbval 1 -punit mm laminate 1 1.2 0') % this is to specify in mm

model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');

RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 -Disk.Sonox_P502_iso.RC10TH0_25 +Disk.Sonox_P502_iso.RC10TH0_25', ...
   struct('shape','lscirc','xc',15+55/2,'yc',12+25/2),
   'Act2', ... % name of patch
   'BaseId1 -Disk.Sonox_P502_iso.RC10TH0_25 +Disk.Sonox_P502_iso.RC10TH0_25', ...
   struct('shape','lscirc','xc',15+55/2,'yc',63+25/2)};

%
mo3=d_piezo('MeshPlate',RG);
mo3=stack_rm(mo3,'info','Electrodes'); % Old Stack not necessary or should be set to 0
mo3.pl([3 5 7 9],7)=0; % Set damping to zero in Noliac otherwise complex static respons
% To avoid warning due to the use of simplified piezo properties.
mo3=p_piezo('DToSimple',mo3)

nd=feutil('find node x==463 & y==100',model);
elnd=floor(p_piezo('electrodedof.*',mo3)); % Nodes associated to electrodes
mo3=fe_case(mo3,'SensDof','Tip',nd+.03); % Displ sensor
mo3=fe_case(mo3,'DofSet','V-Act',struct('def',1,'DOF',elnd(1)+.21)); %Act
mo3=p_piezo(['ElectrodeSensQ  ' num2str(elnd(1)) ' Q-Act'],mo3); % Charge sensors
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(2)) ' Q-S1'],mo3);
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(3)) ' Q-S2'],mo3);
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(4)) ' Q-S3'],mo3);

% Fix last 3 elec dofs to measure resultant (charge)
mo3=fe_case(mo3,'FixDof', ...
['SC*' num2str(elnd(2)) '/' num2str(elnd(3)) '/' num2str(elnd(4)) ], ...
elnd([2:4])+.21);
sens=fe_case(mo3,'sens');

d3=fe_simul('dfrf',stack_set(mo3,'info','Freq',0)); % direct refer frf at 0Hz
d3t=sens.cta(1,:)*d3.def; % First electrode is on top now ?

feplot(mo3,d3)
fecom(';colordatapro;view3;undef line')
d_piezo('setstyle',cf);
iimouse('view',gca,[ -1499 -1955 1462 -248.6 -325.7 275.9 0.30 0.40 0.87 2.20]); % obta
```

The local remeshing can be see in Figure 5.6.

```
%% Step 7 : Check convergence

ref=[5 3 2];
dt=[d3t];
```

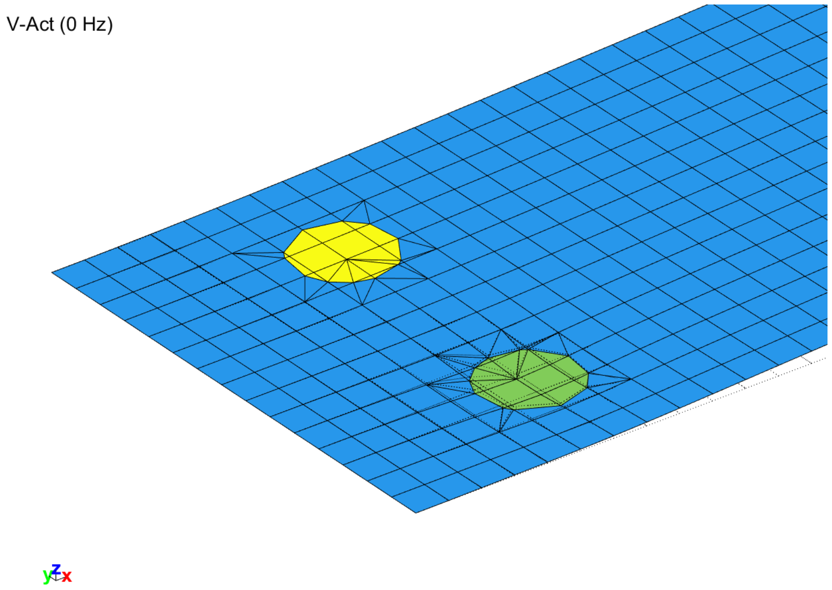


Figure 5.6: Static response to a unit voltage application on one of the bottom circular piezoelectric patches

```
for ij=1:length(ref)

 model=feutil('objectquad 1 1',[0 0 0;1 0 0;0 1 0], ...
    feutil(['refineline ' num2str(ref(ij))],[0 463]), ...
    feutil(['refineline ' num2str(ref(ij))],[0 100]));
     model.unit='mm';
% Material Properties
model.pl=m_elastic('dbval 1 Aluminum');
% Laminate properties
model.il=p_shell('dbval 1 -punit mm laminate 1 1.2 0') % this is to specify in mm
model=fe_case(model,'FixDof','Cantilever','x==0 -DOF 1:6');

RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 -Disk.Sonox_P502_iso.RC10TH0_25 +Disk.Sonox_P502_iso.RC10TH0_25', ...
   struct('shape','lscirc','xc',15+55/2,'yc',12+25/2),
   'Act2', ... % name of patch
   'BaseId1 -Disk.Sonox_P502_iso.RC10TH0_25 +Disk.Sonox_P502_iso.RC10TH0_25', ...
   struct('shape','lscirc','xc',15+55/2,'yc',63+25/2)};

%
mo3=d_piezo('MeshPlate',RG);
mo3=stack_rm(mo3,'info','Electrodes'); % Old Stack not necessary or should be set to 0
mo3.pl([3 5 7 9],7)=0; % Set damping to zero in Noliac otherwise complex respons
% To avoid warning due to the use of simplified piezo properties.
mo3=p_piezo('DToSimple',mo3)

nd=feutil('find node x==463 & y==100',model);
elnd=floor(p_piezo('electrodedof.*',mo3)); % Nodes associated to electrodes
mo3=fe_case(mo3,'SensDof','Tip',nd+.03); % Displ sensor
mo3=fe_case(mo3,'DofSet','V-Act',struct('def',1,'DOF',elnd(1)+.21)); %Act
mo3=p_piezo(['ElectrodeSensQ  ' num2str(elnd(1)) ' Q-Act'],mo3); % Charge sensors
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(2)) ' Q-S1'],mo3);
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(3)) ' Q-S2'],mo3);
mo3=p_piezo(['ElectrodeSensQ ' num2str(elnd(4)) ' Q-S3'],mo3);

% Fix last 3 elec dofs to measure resultant (charge)
mo3=fe_case(mo3,'FixDof', ...
['SC*' num2str(elnd(2)) '/' num2str(elnd(3)) '/' num2str(elnd(4)) ], ...
elnd([2:4])+.21);
sens=fe_case(mo3,'sens');

d3=fe_simul('dfrf',stack_set(mo3,'info','Freq',0)); % direct refer frf at 0Hz
d3t=sens.cta(1,:)*d3.def; % First electrode is on top now ?

dt=[dt; d3t];
end

gf=figure; plot([7 ref],1e6*(dt),'linewidth',2); set(gca, 'XDir','reverse');
set(gca,'Fontsize',15)
xlabel('mesh size (mm)'); ylabel('tip displacement (mm/V)')
```

The convergence of the tip displacement can be see in Figure 5.7. Note that the tip displacement is about 5 times smaller than with the rectangular patches, which is mainly due to the circular shape (less efficient than the rectangular shape for plate bending).

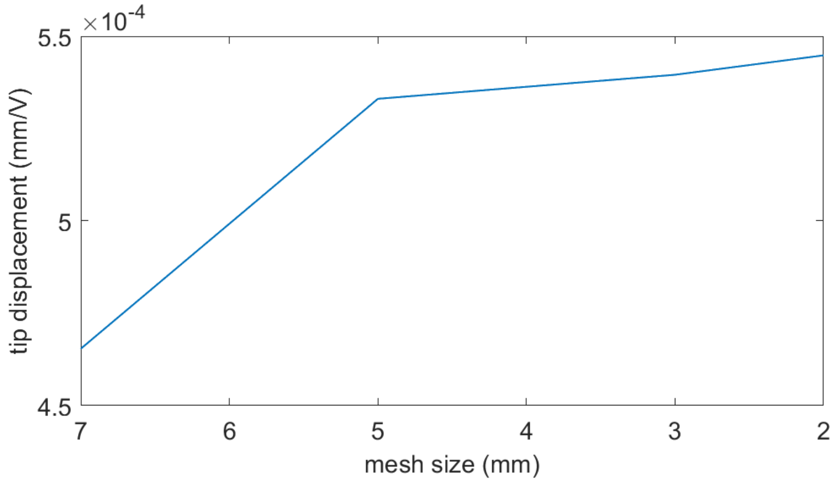


Figure 5.7: Convergence of the tip displacement when the size of the elements of the main plate is varied, for a circular piezoelectric patch

5.3 Using predefined patches

The different types of patches that can be integrated in plate meshes can be seen by typing the command `m_piezo 'patch'`. At the moment of writing this documentation, the output is

```
{'Noliac.NCE51.RC12TH1'  }    {'Noliac, material, disk ODiameter and THickness'}
{'SmartM.MFC-P1.2814'    }    {'Smart materials MFC d33, .WiLe (dims mm)'     }
{'SmartM.MFC-P2.2814'    }    {'Smart materials MFC d31, .WiLe (dims mm)'     }
{'Disk.Material.RC12TH1' }    {'Circular patch, material, geometry'           }
{'Rect.Material.2814TH_5'}    {'Rectangular patch, material, geometry'        }
```

The last two types of patches are the ones used in the previous scripts, which correspond to rectangular and circular patches, for which the material can be chosen, as well as the geometrical properties. The Noliac patch is a disk, and could be described with the generic Disk.Material.RCXXTHX command.

There also exist on the market packaged piezoelectric patches which are made of several layers, such as the Macro Fiber Composite (https://smart-material.com). The properties of the different layers are described in more details in section **??** . Two types of MFCs exist (P1, and P2).
The following example is the integration of two $P1$-type MFC patches on a beam, modeled with multi-layer shell elements. The example is further discussed in section **??** , the resulting mesh is represented in Figure 5.8.

```
% See full example in d_piezo('ScriptTutoPlateMeshingMFC')
d_piezo('DefineStyles');

%% Step 1 : Create mesh. Geometric properties in the manual
RO=struct('L',463,'w',50,'a',85,'b',28,'c',15,'d',11);

% create a rectangle with targetl = 3 mm
 model=feutil('objectquad 1 1',[0 0 0;1 0 0;0 1 0], ...
    feutil('refineline 5',[0 RO.c+[0 RO.a] RO.L]), ...
    feutil('refineline 5',[0 RO.d+[0 RO.b] RO.w]));
%%%%% Material Properties for supporting plate
 model.pl=m_elastic('dbval 1 -unit MM Aluminum'); % Aluminum
 model.il=p_shell('dbval 1 -punit MM laminate 1 1 0');
 model.unit='MM';

RG.list={'Name','Lam','shape'
   'Main_plate', model,''  % Base structure
   'Act1', ... % name of patch
   'BaseId1 +SmartM.MFC-P1.8528 -SmartM.MFC-P1.8528', ... % Layout definition
     struct('shape','LsRect', ... % Remeshing strategy (lsutil rect here)
       'xc',RO.c+RO.a/2,'yc',RO.d+RO.b/2,'alpha',0,'tolE',.1)
       };

mo1=d_piezo('MeshPlate',RG);
cf=feplot(mo1); fecom(';colordatapro;view3');
cf.mdl.name='MFC plate mesh'; % Model name for title
d_piezo('setstyle',cf)
```

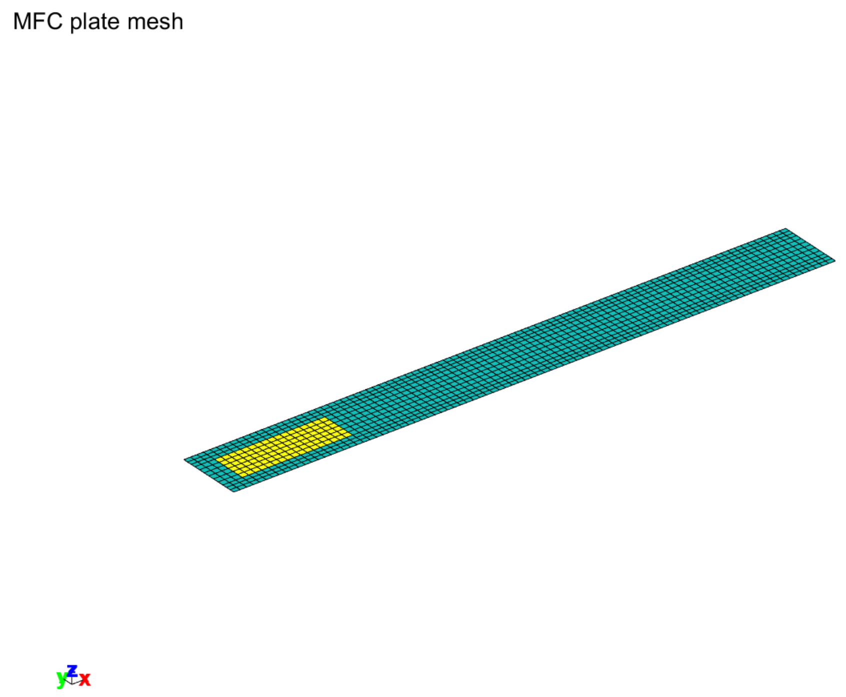


Figure 5.8: Mesh of the plate with MFC transducers on top and bottom. The different colours represent the different groups

6

Model reduction and I/O state-space models

Contents

6.1 Model reduction theory

Finite element models of structures need to have many degrees of freedom to represent the geometrical details of complex structures. For models of structural dynamics, one is however interested in

- a restricted frequency range ($s = i\omega \in [\omega_1 \;\; \omega_2]$)
- a small number of inputs and outputs (b, c)
- a limited parameter space α (updated physical parameters, design changes, non-linearities, etc.)

These restrictions on the expected predictions allow the creation of low order models that accurately represent the dynamics of the full order model in all the considered loading/parameter conditions.

6.1.1 General framework

Model reduction procedures are discrete versions of Ritz/Galerkin analyzes: they seek solutions in the subspace generated by a reduction matrix T. Assuming $\{q\} = [T]\{q_R\}$, the second order finite element model (4.1) is projected as follows

$$\begin{gathered}\left[T^TMTs^2 + T^TCTs + T^TKT\right]_{NR\times NR}\{q_R(s)\} = \left[T^Tb\right]_{NR\times NA}\{u(s)\}_{NA\times 1}\\ \{y(s)\}_{NS\times 1} = [cT]_{NS\times NR}\{q_R(s)\}_{NR\times 1}\end{gathered} \tag{6.1}$$

Modal analysis, model reduction, component mode synthesis, and related methods all deal with an appropriate selection of singular projection bases ($[T]_{N\times NR}$ with $NR \ll N$). This section summarizes the theory behind these methods with references to other works that give more details.

The solutions provided by *SDT* make two further assumptions which are not hard limitations but allow more consistent treatments while covering all but the most exotic problems. The projection is chosen to preserve reciprocity (left multiplication by T^T and not another matrix). The projection bases are assumed to be real.

An accurate model is defined by the fact that the input/output relation is preserved for a given frequency and parameter range

$$[c]\,[Z(s,\alpha)]^{-1}\,[b] \approx [cT]\left[T^TZ(s,\alpha)T\right]^{-1}\left[T^Tb\right] \tag{6.2}$$

where $Z(s,\alpha)$ is the dynamic flexibility $Z(s) = \left[K + Cs + Ms^2\right]$ for a given set of parameters α

Traditional modal analysis, combines normal modes and static responses. *Component mode synthesis* (CMS) methods extend the selection of boundary conditions used to compute the normal modes. The *SDT* further extends the use of reduction bases to parameterized problems.

A key property for model reduction methods is that the input/output behavior of a model only depends on the vector space generated by the projection matrix T. Thus range(T) = range($\tilde{T}$) implies that

$$[cT]\left[T^T Z T\right]^{-1}\left[T^T b\right] = \left[c\tilde{T}\right]\left[\tilde{T}^T Z \tilde{T}\right]^{-1}\left[\tilde{T}^T b\right] \tag{6.3}$$

This **equivalence property** is central to the flexibility provided by the *SDT* in CMS applications (it allows the decoupling of the reduction and coupled prediction phases) and modeshape expansion methods (it allows the definition of a static/dynamic expansion on sensors that do not correspond to DOFs).

6.1.2 Normal mode models

Normal modes are defined by the eigenvalue problem

$$\left(-\omega_j^2\,[M] + [K]\right)_{N\times N}\{\phi_j\}_{N\times 1} = \{0\}_{N\times 1} \tag{6.4}$$

based on inertia properties (represented by the positive definite mass matrix M) and underlying elastic properties (represented by a positive semi-definite stiffness K). The matrices being positive, there are N independent eigenvectors $\{\phi_j\}$ (forming a matrix noted $[\phi]$) and eigenvalues ω_j^2 (forming a diagonal matrix noted $\left[\backslash \omega_{j\backslash}^2\right]$).

As solutions of the eigenvalue problem (6.4), the full set of N normal modes verify two **orthogonality conditions** with respect to the mass and the stiffness

$$[\phi]^T [M] [\phi] = \left[\backslash \mu_{j\backslash}\right]_{N\times N} \quad \text{and} \quad [\phi]^T [K] [\phi] = \left[\backslash \mu_j \omega_{j\backslash}^2\right] \tag{6.5}$$

where μ is a diagonal matrix of modal masses (which are quantities depending uniquely on the way the eigenvectors ϕ are scaled).

In the *SDT*, the normal modeshapes are assumed to be mass normalized so that $[\mu] = [I]$ (implying $[\phi]^T [M] [\phi] = [I]$ and $[\phi]^T [K] [\phi] = \left[\backslash \omega_{j\backslash}^2\right]$). The **mass normalization** of modeshapes is independent from a particular choice of sensors or actuators.

Another traditional normalization is to set a particular component of $\tilde{\phi}_j$ to 1. Using an output shape matrix this is equivalent to $c_l\tilde{\phi}_j = 1$ (the observed motion at sensor c_l is unity). $\tilde{\phi}_j$, the modeshape with a component scaled to 1, is related to the mass normalized modeshape by $\tilde{\phi}_j = \phi_j/(c_l\phi_j)$.

$$m_j(c_l) = (c_l\phi_j)^{-2} \tag{6.6}$$

is called the **modal or generalized mass** at sensor c_l.

6.2 State space models

6.2.1 General theory

While normal mode models are appropriate for structures, **state-space models** allow the representation of more general linear dynamic systems and are commonly used in the *Control Toolbox* or SIMULINK. The standard form for state space-models is

$$\begin{aligned}\{\dot{x}\} &= [A]\{x(t)\} + [B]\{u(t)\} \\ \{y\} &= [C]\{x(t)\} + [D]\{u(t)\}\end{aligned} \tag{6.7}$$

with inputs $\{u\}$, states $\{x\}$ and outputs $\{y\}$. State-space models are represented in the *SDT*, as generally done in other Toolboxes for use with MATLAB, using four independent matrix variables `a`, `b`, `c`, and `d` (you should also take a look at the LTI state-space object of the *Control Toolbox*).

The transfer functions from inputs to outputs are described in the frequency domain by

$$\{y(s)\} = \Big([C]\,[s\ I - A]^{-1}\,[B] + [D]\Big)\,\{u(s)\} \tag{6.8}$$

A state-space representation of the nominal structural model (4.1) is given by

$$\begin{aligned}\left\{\begin{array}{c} q' \\ q'' \end{array}\right\} &= \left[\begin{array}{cc} 0 & I \\ -M^{-1}K & -M^{-1}C \end{array}\right]\left\{\begin{array}{c} q \\ q' \end{array}\right\} + \left[\begin{array}{c} 0 \\ M^{-1}b \end{array}\right]\{u(t)\} \\ \{y(t)\} &= [c \ \ 0]\left\{\begin{array}{c} q \\ q' \end{array}\right\}\end{aligned} \tag{6.9}$$

The interest of this representation is mostly academic because it does not preserve symmetry (a useful feature of models of structures associated to the assumption of reciprocity) and because $M^{-1}K$ is usually a full matrix (so that the associated memory requirements for a realistic finite element model would be prohibitive). The *SDT* thus always starts by transforming a model to the normal mode form and the associated state-space model .

The natural state-space representation of normal mode models (6.4) is given by

$$\begin{aligned}\left\{\begin{array}{c} p' \\ p'' \end{array}\right\} &= \left[\begin{array}{cc} 0 & I \\ -\Omega^2 & -\Gamma \end{array}\right]\left\{\begin{array}{c} p \\ p' \end{array}\right\} + \left[\begin{array}{c} 0 \\ \phi^T b \end{array}\right]\{u(t)\} \\ \{y(t)\} &= [c\phi \ \ 0]\left\{\begin{array}{c} p \\ p' \end{array}\right\}\end{aligned} \tag{6.10}$$

Transformations to this form are provided by `nor2ss` and `fe2ss`. Note however that, as demonstrated in section **??** , using strictly the modeshapes in the reduced state-space model is generally not sufficient to obtain a proper representation of the zeros, which are very important in control applications. `fe2ss` uses two different representations for reduced state-space models based on normal modes and static corrections, which are detailed in the next section.

6.2.2 State-space formulations with static correction

As shown in section **??** , it is possible to augment the basis of modeshapes with static corrections to applied loads to form an accurate reduction basis. Once the basis is orthonormalized, the additional vectors due to static corrections appear as additional modes in the basis. The transformation to a state-space model is straightforward using (6.10), but has the disadvantage to increase considerably the size of the model if a large number of input forces are considered. An alternative is presented below.

We recall that the modal decomposition of the flexibility matrix given by:

$$\left[Ms^2+K\right]^{-1}[b] \approx \sum_{j=1}^{NR} \frac{\{\phi_j\}\{\phi_j\}^T[b]}{s^2+\omega_j^2} + \sum_{j=NR+1}^{N} \frac{\{\phi_j\}\{\phi_j\}^T[b]}{\omega_j^2} \tag{6.11}$$

can be rewritten using (**??**) as:

$$\left[Ms^2+K\right]^{-1}[b] \approx \sum_{j=1}^{NR} \frac{\{\phi_j\}\{\phi_j\}^T[b]}{s^2+\omega_j^2} + [K]^{-1}[b] - \sum_{j=1}^{NR} \frac{\{\phi_j\}\{\phi_j\}^T[b]}{\omega_j^2} \tag{6.12}$$

which shows that the last two terms are proportional to the input $[b]$ and constant (do not depend on ω). They can therefore be introduced in matrix `d` of the state-space model, with matrices `a`,`b` and `c` corresponding to the case where only the normal modes are retained (6.10). The state-space model then becomes :

$$\begin{aligned} \left\{\begin{array}{c} p' \\ p'' \end{array}\right\} &= \left[\begin{array}{cc} 0 & I \\ -\Omega^2 & -\Gamma \end{array}\right] \left\{\begin{array}{c} p \\ p' \end{array}\right\} + \left[\begin{array}{c} 0 \\ \phi^T b \end{array}\right] \{u(t)\} \\ \{y(t)\} &= [c\phi \;\; 0] \left\{\begin{array}{c} p \\ p' \end{array}\right\} + [d]\,[u] \end{aligned} \tag{6.13}$$

where

$$[d] = [c]\,[K]^{-1}[b] - \sum_{j=1}^{NR} \frac{[c]\{\phi_j\}\{\phi_j\}^T[b]}{\omega_j^2} \tag{6.14}$$

The advantage of building the state-space model with such an approach is that the number of degrees of freedom is not increased, and that the $[d]$ matrix introduces a static contribution which cannot be destabilized by a controller. It introduces however a non-physical feed-through term which is directly proportional to the input force. This violates the principles of wave propagation: if the response to a rectangular impulse is computed for example, the wave should arrive to the sensors after a time related to the distance and wave velocity in the medium, and here, due to the non-zero $[d]$ matrix, a part of it arrives instantaneously.

Building state-space models with static corrections but no additional high-frequency modes can be done with the option `-dterm` in `fe2ss`.

6.2.3 State-space models with static correction: illustration on the tower example

The cantilever beam (tower) example presented in section **??** is considered again, and the two approaches (additional mode and $[d]$ matrix approach) to build a state-space model are illustrated below

```
% See full example as MATLAB code in d_piezo('ScriptTutoTowerSS')
d_piezo('DefineStyles');

%% Step 1 : Build the model and define actuator and sensor
model=d_piezo('MeshTower');



% Step 2 : State-space models
[sys,TR] = fe2ss('free 5 3 0 -dterm',model);
[sys2,TR2] = fe2ss('free 5 3 0 ',model);

w=linspace(0,30*2*pi,2048);% Extended frequency range
C1=qbode(sys,w,'struct');C1.name='SS-dterm';
C2=qbode(sys2,w,'struct');C2.name='SS-mode';
ci=iiplot;
iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C2.name,C2}); iicom('submagpha')
d_piezo('setstyle',ci);
%



% End of script
```

Figure 6.1 shows that with the two approaches, the state-space model gives a correct representation of the transfer function in the frequency band of interest [0 10] Hz, but that the behavior is different at higher frequencies. The `-dterm` option does not introduce a high frequency mode, but the transfer function does not have a high-frequency roll-off due to the added constant term.

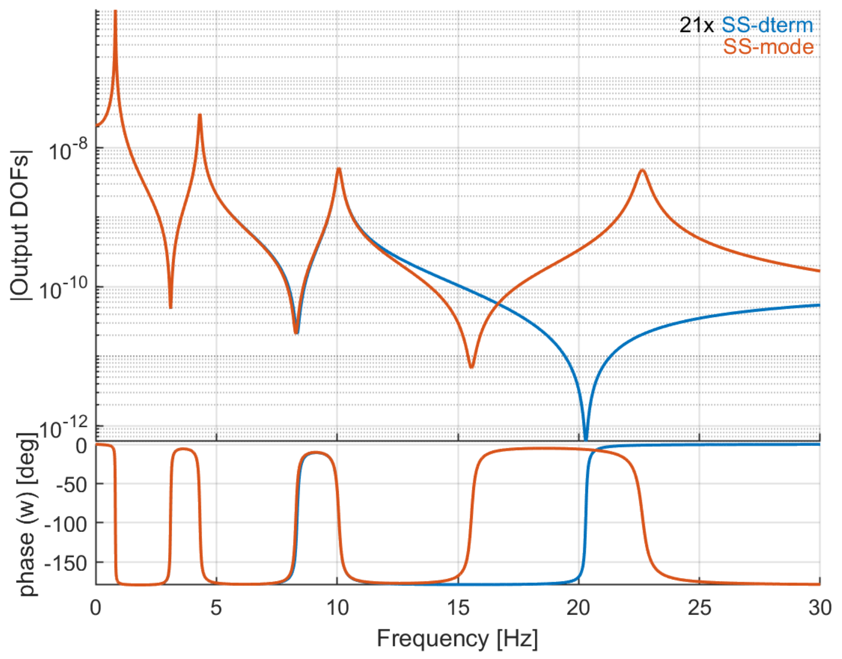


Figure 6.1: Comparison between state-space models using the d-term approach or the additional high-frequency mode for static correction

6.3 State-space models with imposed displacement and acceleration

In order to be able to impose a displacement or velocity instead of a mechanical load, the displacement vector $\{q\}$ is divided in two contributions:

$$\{q\} = [T_C]\{q_C\} + [T_I]\{q_I\} \tag{6.15}$$

where $\{q_I\}$ is the part of $\{q\}$ on which either a displacement or an acceleration need to be imposed (I is used as a reference to "interface"), and $\{q_C\}$ represents the remaining dofs where no displacement or acceleration is imposed. As will be discussed later, different choices can be used for $[T_I]$ and $[T_C]$.

We assume a general form of these matrices where the degrees of freedom related to imposed displacement or acceleration in $\{q\}$, the associated lines in $[T_C]$ must be all zeros, and for the $[T_I]$ matrix, all zeros except one on the column related to this degree of freedom. In doing so, the degrees of freedom related to $\{q_I\}$ are the physical displacements of the structure where the displacement or acceleration needs to be imposed.

If the full finite element model of the structure is used, matrix $[T]_C$ is a $N \times N$ identity matrix from which the columns corresponding to the imposed dofs have been removed.

The strategy to choose matrix $[T_I]$ will differ for the case of imposed displacement and acceleration, as will be detailed below. Using (6.15), the equations of motion (4.2) become ($u(t) = 0$)

$$[M]\left([T_C]\left\{q_C''\right\} + [T_I]\left\{q_I''\right\}\right) + [C]\left([T_C]\left\{q_C'\right\} + [T_I]\left\{q_I'\right\}\right) + [K]\left([T_C]\{q_C\} + [T_I]\{q_I\}\right) = 0 \quad (6.16)$$

and after premultiplying by $[T_C]^T$ and putting leaving the unknown displacements $\{q_C\}$ to the left-hand side of the equation, we get

$$[M_{CC}]\left\{q_C''\right\} + [C_{CC}]\left\{q_C'\right\} + [K_{CC}]\{q_C\} = -[T_C]^T\left([M][T_I]\left\{q_I''\right\} + [C][T_I]\left\{q_I'\right\} + [K][T_I]\{q_I\}\right) \quad (6.17)$$

where $[M_{CC}], [C_{CC}]$ and $[K_{CC}]$ are the mass, damping and stiffness matrices with fixed boundary conditions at the degrees of freedom corresponding to $\{q_I\}$. The subscript C stands for constrained, as the equations of motion in (6.17) correspond to a dynamic problem for which the degrees of freedom corresponding to the imposed motion are fixed, and the imposed displacement, velocity or accelerations correspond to an equivalent mechanical force vector.

As such, it is difficult to build a state-space model from these equations, whether for imposed displacement or acceleration, because only one of the two is imposed and the other is unknown.

6.3.1 State-space models with imposed displacement

For imposed displacement, the typical choice for $[T_I]$ consists in taking a matrix which is limited to the degrees of freedom where the displacement is imposed. All terms on the rows corresponding to the non-imposed degrees of freedom are therefore equal to 0. In this case, the forces acting on the structure are limited to the degrees of freedom which are coupled to the interface through the stiffness, mass, and damping matrices. If the mass matrix is diagonal, there is no such coupling, and if it is not, the coupling is very weak, so that the forces related to inertia can be neglected. This is also the case of the damping forces, as long as the damping is small. In such a case, the equations become

$$[M_{CC}]\left\{{q_C}''\right\} + [C_{CC}]\left\{{q_C}'\right\} + [K_{CC}]\{q_C\} = -[T_C]^T[K][T_I]\{q_I\} \quad (6.18)$$

where now the forcing vector is only a function of the imposed displacement. With this formulation, a state-space model can be built using the approach detailed in section **??** , where normal modes of the system with the imposed dofs set to 0 and a static correction to $-[T_C]^T[K][T_I]$ is added. The displacement at the imposed dofs can be directly recovered using the $[d]$ matrix if a sensor is defined at these location, using matric $[T]_I$.

Note that in (6.18), there are as many inputs as there are degrees of freedom in $\{q_I\}$. This can be simplified to a single input using an expansion vector such that:

$$\{q_I\} = \{L\}\, q_0 \tag{6.19}$$

where q_0 is now a single input scalar displacement to the system, and the right-hand-side of equation (6.18) reduces to a single vector multiplied by the input displacement q_0. The number of necessary static responses to build the state-space model is therefore also reduced to one instead of the number of degrees of freedom in $\{q_I\}$.

The construction of a state-space model with imposed displacement is illustrated below on the same example of the concrete tower. The function fe2ss is used to build directly the I/O state-space model from the finite element after defining properly the input (horizontal imposed displacement) and the output (displacement at the top of the tower)

```
% See full example as MATLAB code in d_piezo('ScriptTutoTowerSSUimp')
d_piezo('DefineStyles');

%% Step 1 : Build the model and define actuator and sensor
model=d_piezo('MeshTower');
model=d_avc('meshtower');
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
model=fe_case(model,'DOFSet','UImp',[1.01]); % Imposed horizontal displ


%% Step 2 : Reference method - exact solution + Inertial term neglected - full model
% Build matrices
[model,Case] = fe_case('assemble NoT -matdes 2 1 Case -SE',model) ;

% Full model
K0 = feutilb('tkt',Case.T,model.K); % Assemble matrices taking into account BCs
F1 = -Case.T'*model.K{2}*Case.TIn; % Loading due to imposed displacement
F2 = -Case.T'*model.K{1}*Case.TIn; % Inertial term
%
% compute response in freq domain
w=logspace(-2,2,2048);
for i=1:length(w)
```

```
    U0(:,i)=Case.T*((K0{2}*(1+0.02*1i)-w(i)^2*K0{1})\(F1-w(i)^2*F2)); % Take into accou
    U1(:,i)=Case.T*((K0{2}*(1+0.02*1i)-w(i)^2*K0{1})\F1); % Stiffness term only
end
CTA = fe_c(model.DOF,21.01); u0 = CTA*U0; u1 = CTA*U1;

% Change output format to be compatible with iicom
C1=d_piezo('BuildC1',w'/(2*pi),u0.','Tip displ','Uimp'); C1.name='M and K';
C2=d_piezo('BuildC1',w'/(2*pi),u1.','Tip displ','Uimp'); C2.name='K only';
ci=iiplot;iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C2.name,C2});
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.2 shows that the two curves match perfectly, meaning that the inertial term can indeed be neglected for the excitation, allowing to build an accurate modal state-space model with a static correction with the stifness term only, which is done in the following:

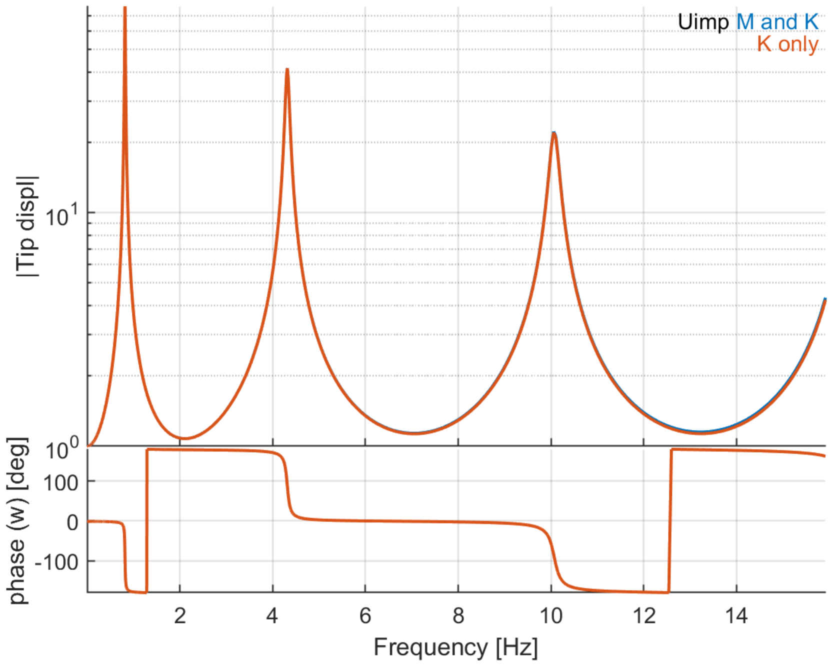


Figure 6.2: Transfer function between horizontal displacement at the bottom/top of the beam using the full model, the loading takes into account both the mass and stiffness terms, or only the stiffness term

```
%% Step 3 :  State-space model using SDT (reduced)
sys=fe2ss('free 5 5 0 -dterm',model);
u2=freqresp(sys(1,1),w); u2 = u2(:);

% Change output format to be compatible with iicom
C3=d_piezo('BuildC1',w'/(2*pi),u2,'Tip displ','Uimp'); C3.name='fe2ss-5md+st';
ci=iiplot;iicom(ci,'curveinit',{'curve',C2.name,C2;'curve',C3.name,C3});
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.3 compares the frequency response functions used with the full model and direct computation, and using a state-space model with 5 modes and a static correction for the stiffness loading term. The match between the two is excellent.

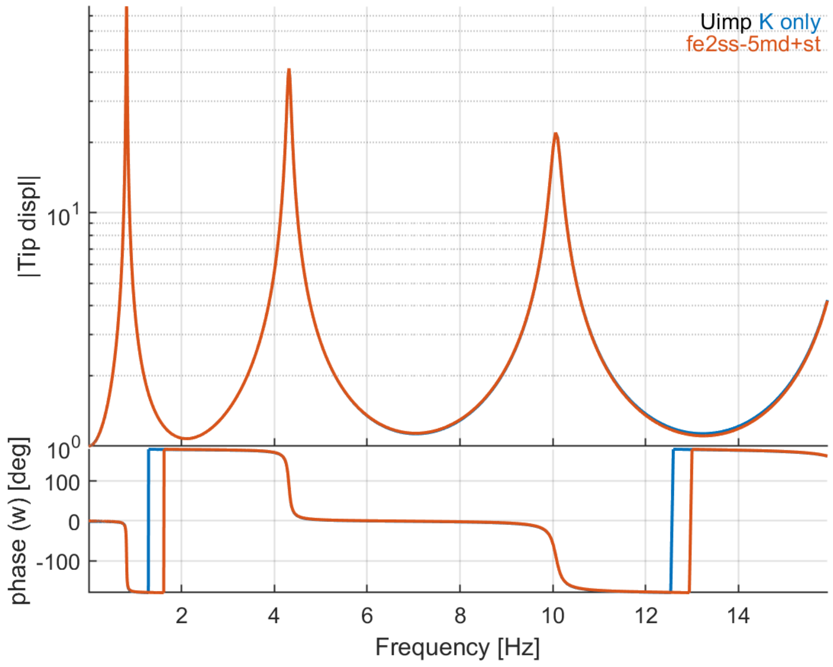


Figure 6.3: Transfer function between horizontal displacement at the bottom/top of the beam using the full model and the reduced state-space model with 5 modes, the loading takes into account only the stiffness term

6.3.2 State-space models with imposed acceleration

In the case of imposed acceleration, it is common to write the problem in terms of relative displacements with respect to the base motion, in which case matrix $[T_I]$ is not anymore limited to the imposed displacements. The approach consists in choosing $[T_I]$ as the static response to imposed displacements, so that it relates to all dofs in the structure. Let us assume that the dofs of the structure are rearranged so that the matrices can be decomposed in four blocks as previously :

$$\left[\tilde{K}\right] = \begin{bmatrix} [K_{II}] & [K_{IC}] \\ [K_{CI}] & [K_{CC}] \end{bmatrix} \tag{6.20}$$

where the index I corresponds to the imposed dofs. $[T_I]$ is then given by the static expansion of the imposed displacements:

$$[T_I] = \begin{bmatrix} I \\ -[K_{CC}]^{-1}[K_{CI}] \end{bmatrix} \{q_0\} \tag{6.21}$$

where $\{q_0\}$ represents the spatial shape of the imposed acceleration at the specific locations. Such a representation is general enough to take into account base excitation problems where the excitation of the base is uniform and in a single direction, such as a uniform translation of the base. In this case, $[T_I]$ represents the rigid body translation of the structure in the direction of the base motion, and $\{q_C\}$ is the relative motion of the structure with respect to the translation of the base. The general approach allows also to impose a combination of the six rigid-body modes of the structure. In this case, the quantity that is withdrawn from the global motion in order to represent the relative motion is different at each location in the structure in the case where rotational rigid body motions are present in the base excitation.

Lastly, the approach also allows to take into account base excitations which induce a deformation of the basis, in which case $[T_I]$ is not a rigid body motion but the static expansion of the imposed displacements on all other dofs of the structure (which reduces to rigid body motions if the imposed displacement is a translation and/or rotation of the base).

With this type of decomposition, in the case where $[T_I]$ is a linear combination of rigid-body motions, the equations of motion (6.16) reduce to:

$$[M_{CC}]\{q_C''\} + [C_{CC}]\{q_C'\} + [K_{CC}]\{q_C\} = -[T_C]^T[M][T_I]\{q_I''\} \tag{6.22}$$

because $[T_I]\{q_I\}$ represents a combination of rigid body modes so that $[K][T_I]\{q_I\} = 0$. For damping models where the damping matrix is proportional to stiffness, the damping term is also equal to zero, otherwise it can be neglected when damping is small.

When $[T_I]$ is not a combination of rigid body modes (i.e. when the imposed acceleration induces deformation of the base), the stiffness term is given by

$$[T_C]^T[K]\begin{bmatrix} I \\ -[K_{CC}]^{-1}[K_{CI}] \end{bmatrix}\{q_I\} = \left(-[K_{CC}][K_{CC}]^{-1}[K_{CI}] + [K_{CI}]\right)\{q_I\} = 0 \tag{6.23}$$

so that the stiffness term is also zero (see also in [4]).

With this simplification, it is possible to build a reduced state-space model where the force vector is related to the inertial term and the imposed acceleration only. Note however that in this case, the state-space model solves for relative displacements, and that it is not straightforward to obtain the absolute displacements (if needed), as it would require to know the base displacement, and here it is the acceleration that is imposed (so one cannot use the $[d]$ matrix here to add the base displacement).

A possible approach is to solve for the absolute displacements by modifying the mass, stiffness and damping matrices as:

$$\left[\hat{M}\right] = \begin{bmatrix} I & 0 \\ 0 & [M_{CC}] \end{bmatrix} \tag{6.24}$$

$$\left[\hat{K}\right] = \begin{bmatrix} 0 & 0 \\ 0 & [K_{CC}] \end{bmatrix} \tag{6.25}$$

$$\left[\hat{C}\right] = \begin{bmatrix} 0 & 0 \\ 0 & [C_{CC}] \end{bmatrix} \tag{6.26}$$

and the displacement and forcing vectors as

$$\{q\} = \left\{ \begin{array}{c} q_I \\ q_C \end{array} \right\} \tag{6.27}$$

$$[F] = \begin{bmatrix} \{q_0''\} \\ -[T_C]^T [M] [T_I] \{q_0''\} \end{bmatrix} \tag{6.28}$$

where $\{q_0''\}$ is the vector of imposed accelerations.

A state-space representation can be built by projecting these equations in the subspace of the constrained modeshapes which are the solution of

$$\left(-\omega_j^2 [M_{CC}] + [K_{CC}]\right) \{\phi_j\} = \{0\} \tag{6.29}$$

The reduced basis (keeping NR modes) noted $\left[\hat{T_C}\right]$ is given by :

$$\left[\hat{T_C}\right] = [\phi_{1 \ldots NR}] \tag{6.30}$$

so that

$$\{q_C\} = \left[\hat{T_C}\right] \{\hat{q_C}\} \tag{6.31}$$

which leads to the reduced matrices (with mass normalized modeshapes)

$$\left[\hat{M_{CC}}\right]=\left[\hat{T_C}\right]^T[M_{CC}]\left[\hat{T_C}\right]=I \quad , \quad \left[\hat{K_{CC}}\right]=\left[\hat{T_C}\right]^T[K_{CC}]\left[\hat{T_C}\right]=\left[\backslash\omega_j^2\backslash\right] \tag{6.32}$$

The state-space representation is then given by:

$$\begin{aligned}\left\{\begin{array}{c}\dot{q_I}\\ \dot{\hat{q_C}}\\ \ddot{q_I}\\ \ddot{\hat{q_C}}\end{array}\right\}=&\left[\begin{array}{cc}\begin{array}{c}[0]\\ \left[\begin{array}{c}0\\ -\left[\hat{T_C}\right]^T[K]\left[T_I\ \ \hat{T_C}\right]\end{array}\right]\end{array} & \begin{array}{c}[I]\\ \left[\begin{array}{c}0\\ -\left[\hat{T_C}\right]^T[C]\left[T_I\ \ \hat{T_C}\right]\end{array}\right]\end{array}\end{array}\right]\left\{\begin{array}{c}q_I\\ \hat{q_C}\\ \dot{q_I}\\ \dot{\hat{q_C}}\end{array}\right\}+\\ &\left[\begin{array}{cc}0 & 0\\ 0 & 0\\ 0 & I\\ \left[\hat{T_C}\right]^T b & \left[\hat{T_C}\right]^T[T_I]\end{array}\right]\left\{\begin{array}{c}u_F\\ \ddot{q_0}\end{array}\right\}\\ \{y\}=&\left[c\,[T_I]\quad c\left[\hat{T_C}\right]\quad 0\quad 0\right]\left\{\begin{array}{c}q_I\\ \hat{q_C}\\ \dot{q_I}\\ \dot{\hat{q_C}}\end{array}\right\}+[0]\left\{\begin{array}{c}u_F\\ q_0''\end{array}\right\}\end{aligned}\tag{6.33}$$

With this choice of $[T]_I$, the submatrix $-\left[\hat{T_C}\right]^T[K]\,[T_I]=0$ and the states are fully decoupled. The method can be used with the alternative representation where T_I is taken as the matrix limited to the imposed displacement (as for the imposed displacement formulation, in which case the submatrix $-\left[\hat{T_C}\right]^T[K]\,[T_I]$ is non-zero and remains in the left-hand-side because $\{q_I\}$ is unknown. The disadvantage of this formulation is the fact that the state-variables remain coupled in the modal domain, and that it is necessary to augment the modal basis with static responses to $[M]\,[T_I]\,\{q_0''\}$. This is not the case with the previous approach, as will be illustrated in the examples.

The general method implemented in SDT to build state-space models with imposed displacements corresponds to the second case where the states remain coupled, it requires the combination of the use of fe_reduc and nor2ss function and cannot be performed directly with fe2ss. An illustration is given in the script below. The first calculation is explicit and performed in the frequency domain. The problem is solved using relative displacements.

```
% See full example as MATLAB code in d_piezo('ScriptTutoTowerSSAimp')
d_piezo('DefineStyles');

%% Step 1 : Build the model and define actuator and sensor
model=d_piezo('MeshTower');
```

```
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
model=fe_case(model,'DOFSet','UImp',[1.01]); % Set an imposed displacement

%% Step 2 : Regular method with RHS M*Tin (relative displacement)

% --------- full model
[model,Case] = fe_case('assemble NoT -matdes 2 1 Case -SE',model) ;

K0 = feutilb('tkt',Case.T,model.K); % Assemble matrices taking into account BCs
TIn=fe_simul('static',model); TIn=TIn.def; % Compute TIn as static response to imposed
F = -Case.T'*model.K{1}*TIn; % Loading due to imposed displacement

% compute response in freq domain (relative displacement)
w=logspace(-2,2,2048);
for i=1:length(w)
    U0r(:,i)=Case.T*((K0{2}*(1+0.02*1i)-w(i)^2*K0{1})\F);
end
Uimp=1./(-w.^2); % Imposed displacement for a unit imposed acceleration

% Absolute displacement
for i=1:length(w)
U0(:,i)=U0r(:,i)+Uimp(i)*TIn;
end

CTA = fe_c(model.DOF,21.01); u0 = CTA*U0; u0r = CTA*U0r;

% Change output format to be compatible with iicom
C0=d_piezo('BuildC1',w'/(2*pi),u0.','u-top','Aimp'); C1.name='Full';
C1=d_piezo('BuildC1',w'/(2*pi),u0r.','ur-top','Aimp'); C1.name='Full';
```

The next step is to make a state-space model with fe2ss by building the adequate load based on imposed accelerations. The output of the state-space model is then a relative displacement.

```
%% Step 3 - State-space with SDT(relative) using a DofLoad
model2=d_piezo('Meshtower');
model2=fe_case(model2,'FixDof','Clamped',[1.01 1.06]); % Block all interface dofs
model2=fe_case(model2,'Remove','F-top'); % Remove point force
```

```
[model2,Case2] = fe_case('assemble NoT -matdes 2 1 Case -SE',model2) ;

SET.DOF=model2.DOF; SET.def=Case2.T*F;
model2=fe_case(model2,'DofLoad','AccImp',SET); %

sysr=fe2ss('free 5 5 0 -dterm',model2);
u1r=freqresp(sysr(1,1),w); u1r = u1r(:);
C2=d_piezo('BuildC1',w'/(2*pi),u1r,'ur-top','Aimp'); C1.name='fe2ss 5md+st';
ci=iiplot;iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C2.name,C2});
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.4 shows the comparison between the response (relative displacement at the top of the tower divided by imposed acceleration) of the full model computed in the frequency domain, and of the reduced state-space model using 5 modes and a static correction to the applied load (built from the acceleration and mass matrix).

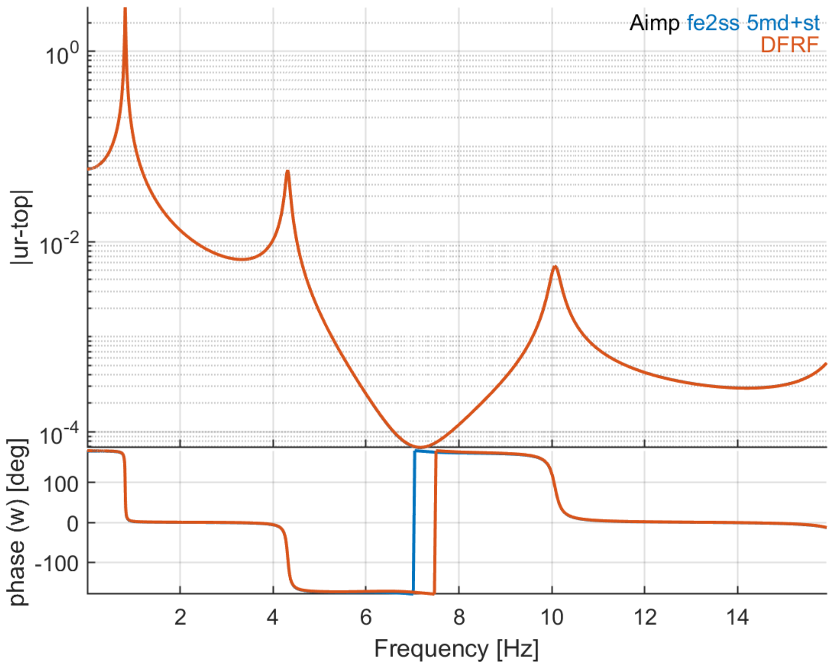


Figure 6.4: comparison between the response (relative displacement at the top of the tower divided by imposed acceleration) of the full model computed in the frequency domain, and of the reduced state-space model using 5 modes and a static correction

The approach used in SDT to build a state-space model allowing to obtain directly the absolute displacement is illustrated in the following:

```
%% Step 4 : state-space model for absolute displacements
TR2 = fe2ss('craigbampton 5 5 -basis',model); % This is a CB basis which is renormalize
% TR2.data is needed for nor2ss hence the normalization.
KCB    = feutilb('tkt',TR2.def,model.K);
sysu= nor2ss(TR2,model) ;
u1=freqresp(sysu(1,1),w); u1=u1(:);
%
C3=d_piezo('BuildC1',w'/(2*pi),u1,'u-top','Aimp'); C3.name='fereduc+nor2ss';
ci=iiplot;iicom(ci,'curveinit',{'curve',C0.name,C0;'curve',C3.name,C3});
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.5 shows that the state-space model obtained with this approach in SDT gives a response very close to the full model computed in the frequency domain.

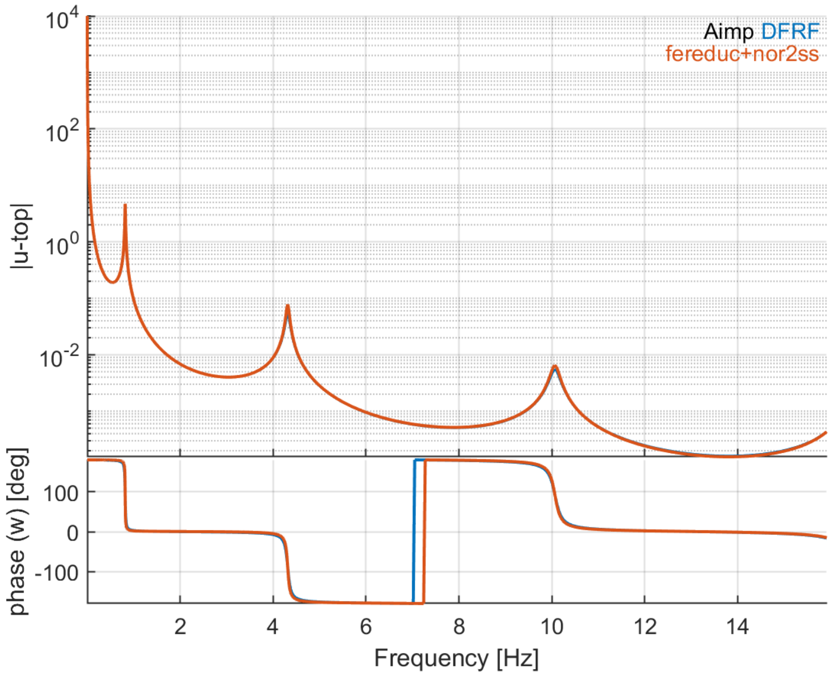


Figure 6.5: Transfer function between horizontal acceleration at the bottom and the absolute displacement at the top of the beam and the imposed horizontal acceleration at the bottom - comparison of the full model (freq domain computation) and the reduced state-space model with 5 modes and static correction

6.3.3 Model reduction and state-space models for piezoelectric structures

When building reduced or state-space models to allow faster simulation, the validity of the reduction is based on assumptions on bandwidth, which drive modal truncation, and considered loads which lead to static correction vectors.

Modes of interest are associated with boundary conditions in the absence of excitation. For the electric part, these are given by potential set to zero (grounded or shorted electrodes) and enforced by actuators (defined as `DofSet` in SDT in the case of voltage actuators) which in the absence of excitation is the same as shorting.

Excitation can be mechanical F_{mech}, charge on free electric potential DOF Q_{In} and imposed voltage V_{In}. One thus seeks to solve a problem of the form

$$\begin{bmatrix} Z_{qq}(s) & Z_{qV} \\ Z_{Vq} & Z_{VV} \end{bmatrix} \begin{Bmatrix} q \\ V \end{Bmatrix} = \begin{Bmatrix} F_{mech} \\ Q_{In} \end{Bmatrix} - \begin{bmatrix} Z_{qV_{In}} \\ Z_{VV_{In}} \end{bmatrix} \{V_{In}\} \tag{6.34}$$

The imposed voltage V_{In} is enforced using a `DofSet` in SDT and is therefore analogous to an imposed displacement. The general form of the loads given above can be simplified due to the fact that there is no coupling terms in the mass matrix between the mechanical and electrical DOFs for full models of piezoelectric structures (see (4.6)), and the fact that the capacitance matrix $[K_{VV}]$ is diagonal. The problem is then reduced to

$$\begin{bmatrix} Z_{qq}(s) & Z_{qV} \\ Z_{Vq} & Z_{VV} \end{bmatrix} \begin{Bmatrix} q \\ V \end{Bmatrix} = \begin{Bmatrix} F_{mech} \\ Q_{In} \end{Bmatrix} - \begin{bmatrix} K_{qV_{In}} \\ 0 \end{bmatrix} \{V_{In}\} \tag{6.35}$$

Using the classical modal synthesis approach (implemented as `fe2ss('free')`), one builds a Ritz basis combining modes with grounded electrodes ($V_{In} = 0$), static responses to mechanical and charge loads and static response to enforced potential:

$$\begin{Bmatrix} q \\ V \\ V_{In} \end{Bmatrix} = \left[\begin{bmatrix} \phi_q \\ \phi_V \\ 0 \end{bmatrix} \begin{bmatrix} Z(0)^{-1} \begin{Bmatrix} F_{mech} \\ Q_{In} \end{Bmatrix} \\ 0 \end{bmatrix} \begin{bmatrix} Z(0)^{-1} \begin{bmatrix} K_{qV_{In}} \\ 0 \end{bmatrix} \\ I \end{bmatrix} \right] \begin{Bmatrix} q_{mode} \\ q_{stat} \\ V_{In} \end{Bmatrix} \tag{6.36}$$

In this basis, one notes that the static response associated with enforced potential V_{In} does not verify the boundary condition of interest for the state-space model where $V_{In} = 0$. Since it is desirable to retain the modes with this boundary condition as the first vectors of basis (6.36) and to include static correction as additional vectors, the strategy used here is to rewrite reduction as

$$\{q\} = \left[\begin{bmatrix} \phi_q \\ \phi_V \\ 0 \end{bmatrix} \begin{bmatrix} Z(0)^{-1} \begin{bmatrix} F_m & K_{qV_{In}} \\ Q_{In} & 0 \end{bmatrix} \\ 0 \end{bmatrix} \right] \{q_R\} + \begin{Bmatrix} 0 \\ 0 \\ V_{In} \end{Bmatrix} \tag{6.37}$$

where the response associated with reduced DOFs q_R verifies $V_{In} = 0$ and the total response is found by adding the enforced potential on the voltage DOF only. The presence of this contribution corresponds to a D term in state-space models. The usual SDT default is to include it as a residual vector as shown in (6.36), but to retain the shorted boundary conditions, form (6.37) is prefered.

The example of the cantilever beam with 4 piezoelectric transducers detailed in section 4.6.2 is considered again. A state-space model is built using the first 10 modes and static corrections discussed above, and the dynamic response is compared to the full-model response.

```
% See full example in d_piezo('ScriptTutoPlate4Pzt')
d_piezo('DefineStyles');

%% Step 1 - Build model and visualize
model=d_piezo('MeshULBplate');  % creates the model
model=fe_case(model,'FixDof','Cantilever','x==0'); % Clamp plate
% Set modal default zeta = 0.01
```

```
model=stack_set(model,'info','DefaultZeta',0.01);
%% Step 2 - Define actuators and sensors and visualize
nd=feutil('find node x==463 & y==100',model);
model=fe_case(model,'SensDof','Tip',{[num2str(nd) ':z']}); % Displ sensor
i1=p_piezo('TabInfo',model);i1=i1.Electrodes(:,1);
model=fe_case(model,'DofSet','V-Act',struct('def',1,'DOF',i1(1)+.21, ...%Act
    'Elt',feutil('selelt proid 104',model))); % Elt defined for display
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-Act',i1(1)),model); % Charge sensors
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S1',i1(2)),model);
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S2',i1(3)),model);
model=p_piezo(sprintf('ElectrodeSensQ  %i Q-S3',i1(4)),model);
% Fix ElectrodeSensQ dofs to measure resultant (charge)
model=fe_case(model,'FixDof','SC*S1-S3',i1(2:end)+.21);

%% Step 3 Compute dynamic response full/state-space and compare
model=stack_set(model,'info','oProp',mklserv_utils('oprop','CpxSym'));
f=linspace(1,100,400); % in Hz

% Full model
d1=fe_simul('dfrf',stack_set(model,'info','Freq',f(:))); % direct refer frf
sens=fe_case(model,'sens'); C1=fe_case('SensObserve -DimPos 2 3 1',sens,d1);
C1=sdsetprop(C1,'PlotInfo','sub','magpha','scale','xlin;ylog');

% state-space model
[s1,TR1]=fe2ss('free 5 10 0 -dterm',model); %
C2=qbode(s1,f(:)*2*pi,'struct');C2.name='SS';

% Compare the two curves
ci=iiplot;
iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C2.name,C2});
iicom('submagpha'); d_piezo('setstyles',ci)




% End of script
```

Figures 6.6 and 6.7 show that there is an excellent match between the responses computed with the full model and the state-space model with 10 modes and static corrections.

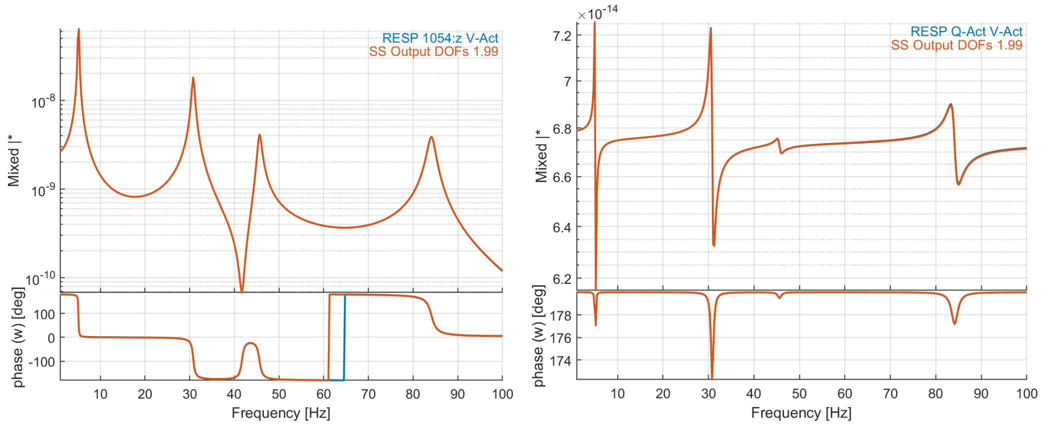


Figure 6.6: Open-loop transfer function between V-Act and tip displacement (left), Q-Act sensor (right), comparison between, full model and reduced state-space model (10 modes + static corrections - dterm)

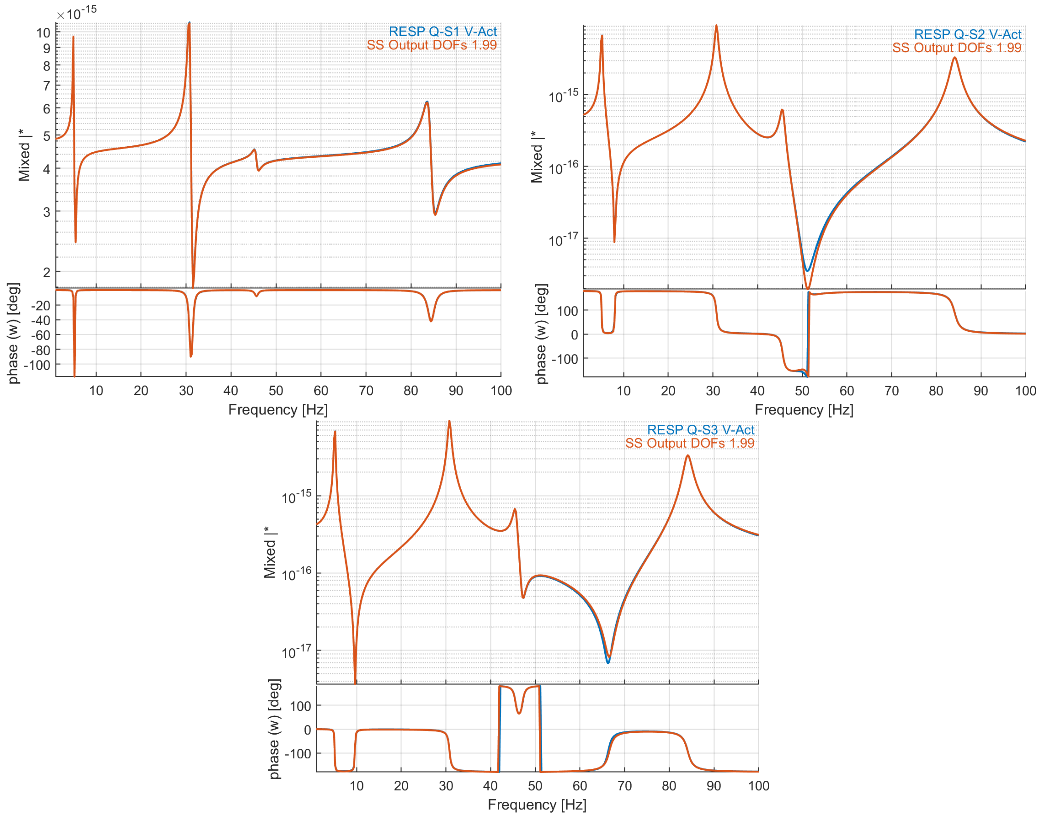


Figure 6.7: Open-loop transfer function between V-Act and Q-S1 sensor (top-left),Q-S2 sensor (top-right), Q-S3 sensor (bottom), comparison between, full model and reduced state-space model (10 modes + static corrections - dterm)

6.4 State-space models and Craig-Bampton model reduction

For coupled problems linked to model substructuring, it is traditional to state the problem in terms of imposed displacements rather than loads. Assuming that the imposed displacements correspond to DOFs, one seeks solutions of problems of the form

$$\begin{bmatrix} Z_{II} & Z_{IC} \\ Z_{CI} & Z_{CC} \end{bmatrix} \begin{Bmatrix} q_I \\ q_C \end{Bmatrix} = \begin{Bmatrix} R_I \\ 0 \end{Bmatrix} \tag{6.38}$$

where the displacement q_I are given and the reaction forces R_I are non-zero. The exact response to an imposed harmonic displacement q_I is given by

$$\{q\} = \begin{bmatrix} I \\ -Z_{CC}^{-1} Z_{CI} \end{bmatrix} \{q_I\} \tag{6.39}$$

The first level of approximation is to use a quasistatic evaluation of this response , that is to use $Z(0) = K$). Model reduction on this basis is known as **static or Guyan condensation** [**?**].

This reduction does not fulfill the requirement of validity over a given frequency range. Craig and Bampton [**?**] thus complemented the static reduction basis by **fixed interface modes** : normal modes of the structure with the imposed boundary condition $q_I = 0$. These modes correspond to singularities in Z_{CC} so their inclusion in the reduction basis allows a direct control of the range over which the reduced model gives a good approximation of the dynamic response.

The Craig-Bampton reduction basis takes the special form

$$\left\{ \begin{array}{c} q_I \\ q_C \end{array} \right\} = \left[\begin{array}{cc} I & 0 \\ -K_{CC}^{-1} K_{CI} & \phi_C \end{array} \right] \{q_R\} \tag{6.40}$$

where the fact that the additional fixed interface modes have zero components on the interface DOFs is very useful to allow direct coupling of various component models. `fe_reduc` provides a solver that directly computes the Craig-Bampton reduction basis.

A major reason of the popularity of the Craig-Bampton (CB) reduction basis is the fact that the interface DOFs q_I appear explicitly in the generalized DOF vector q_R, and is a major reason why the use of Craig-Bampton reduction methods is very popular in the industry.

The major finite elements softwares such as Nastran, Abaqus or Ansys all propose a CB reduction. In the industry, it is a common practice to share models via CB reduced matrices, therefore keeping the details of the geometry and material properties confidential. Generally, the degrees of freedom which are kept in the model are the essential ones, i.e. the ones where forces are applied, or where the magnitude of displacement, velocity or acceleration needs to be assessed, as well as interface degrees of freedom if the structure needs to be coupled to one or several others. The question we address in the next section is how to build an accurate state-space model based on these reduced matrices only.

6.4.1 State-space models with imposed displacements using CB matrices

As detailed in Section section 6.3.1 , when dealing with a full FE model, the imposed displacement can be replaced by a force equal to $-[T_C][K][T_I]\{q_I\}$, neglecting the term related to acceleration. It turns out that this assumption is not valid after aCB reduction.

It can be easily understood if we consider that only the imposed degrees of freedom are retained in the CB basis. In that case, $[T_C][K][T_I] = [K_{CI}]$ which, according to (6.23) is equal to zero, and would result in no load applied to the system. In this specific case, the intertial term is the most important one, while the stiffness term is zero, hence it is not possible to build a state-space model based on the standard approach presented before. This is illustrated below:

```
% See full example as MATLAB code in d_piezo('ScriptTutoTowerSSUimpCB1')
d_piezo('DefineStyles');

%% Step 1 : Build Model with imposed displacement
model=d_avc('meshtower');
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
model=fe_case(model,'DOFSet','UImp',[1.01]); % Leave x free for imposed displ

%% Step 2 : reduce model using CB
% Build matrices
[model,Case] = fe_case('assemble NoT -matdes 2 1 Case -SE',model) ;

% Build CB matrices
CB    = fe_reduc('craigbampton 5 5',model);
TR = CB.TR;

KCB    = feutilb('tkt',TR.def,model.K); % CB
K0 = feutilb('tkt',Case.T,model.K); % Full-model

F = -Case.T'*model.K{2}*Case.TIn;
F1=-KCB{2}(2:end,1);
F2=-KCB{1}(2:end,1);
w=logspace(-2,2,2048);

for i=1:length(w)
    U0(:,i)=Case.T*((K0{2}*(1+0.02*1i)-w(i)^2*K0{1})\F); % full model
    U1r(:,i)=((KCB{2}(2:end,2:end)*(1+0.02*1i)-w(i)^2*KCB{1}(2:end,2:end))\(F1-w(i)^2*F
    U2r(:,i)=((KCB{2}(2:end,2:end)*(1+0.02*1i)-w(i)^2*KCB{1}(2:end,2:end))\(F1)); % CB s
    U3r(:,i)=((KCB{2}(2:end,2:end)*(1+0.02*1i)-w(i)^2*KCB{1}(2:end,2:end))\(-w(i)^2*F2)
end

CTA = fe_c(model.DOF,21.01); u0 = CTA*U0;
U1=TR.def*[ones(1,length(w)); U1r]; u1 = CTA*U1;
U2=TR.def*[ones(1,length(w)); U2r]; u2 = CTA*U2;
U3=TR.def*[ones(1,length(w)); U3r]; u3 = CTA*U3;
```

```
% Change output format to be compatible with iicom
C1=d_piezo('BuildC1',w'/(2*pi),u0.','Tip displ','Uimp'); C1.name='full model';
C2=d_piezo('BuildC1',w'/(2*pi),u1.','Tip displ','Uimp'); C2.name='CB K and M';
C3=d_piezo('BuildC1',w'/(2*pi),u2.','Tip displ','Uimp'); C3.name='CB K';
C4=d_piezo('BuildC1',w'/(2*pi),u3.','Tip displ','Uimp'); C4.name='CB M';
ci=iiplot;iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C2.name,C2;'curve',C3.name,C
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.8 illustrates clearly the fact that the forcing term related to acceleration is the most important one to represent the response.

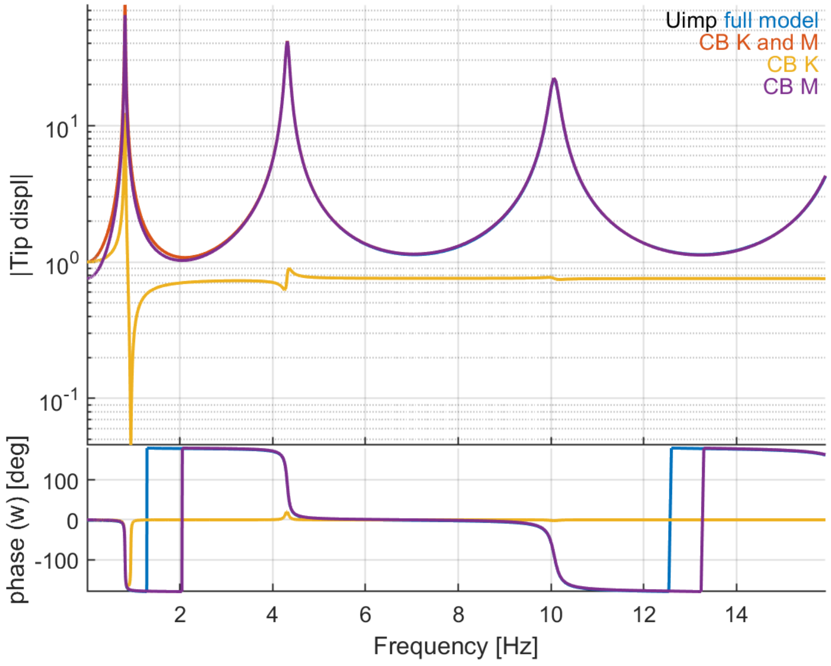


Figure 6.8: Transfer function between horizontal displacement at the bottom and the absolute displacement at the top of the beam

In practice however, the DOFS retained with Craig-Bampton are not limited to the imposed displacements. As an illustrative example, we consider that the translation DOF at the top of the tower is also kept in the CB reduction. Based on these reduced matrices, a state-space model is built with fe2ss the response is compared to the full model.

```
%% Step 2: keep bottom and top translation in the CB basis
```

```
% Mesh and set dofs to be kept
model=d_avc('meshtower');
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
SET.DOF=[1.01; 21.01]; SET.def=eye(2); % Top and bottom DOF to be kept in CB reduction
model=fe_case(model,'DOFSet','UImp',SET); % To retain in CB matrices input and output

% Build CB matrices
model = stack_set(model,'info','EigOpt',[5 5 0]);
model.DOF = feutil('getdof',model);
SE1= fe_reduc('CraigBampton -SE -matdes 2 1 3 4 ',model); % Do not use -USEDOF
% DOFS are numbered with -1.001, which is not a supported format for fe_eig, so you nee
SE1.DOF(SE1.DOF<0) = 1000.99+(1:numel(SE1.DOF(SE1.DOF<0))); %
SE1 = rmfield(SE1,{'Node','Elt','il','pl','Stack','mdof','TR'}) ; % To keep only the ma

% Initialize the reduced model (using super-elements to define matrices)
SE0 = struct('Node',[],'Elt',[]);
model2 = fesuper('SEAdd 1 -1 -unique -initcoef -newID se1',SE0,SE1) ;

% Define input/output
model2=fe_case(model2,'SensDOF','Output',21.01);
model2=fe_case(model2,'DOFSet','UImp',[1.01]); % Leave x free for imposed displ

% Build state-space model based on reduced CB matrices
sys=fe2ss('free 5 5 0 -dterm',model2); u4=freqresp(sys(1,1),w); u4 = u4(:);
C5=d_piezo('BuildC1',w'/(2*pi),u4,'Tip-displ','Uimp'); C5.name='fe2ss CB';
ci=iiplot;iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C5.name,C5});

iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.9 shows that using fe2ss after a CB reduction leads to a very bad reduced state-space representation of the dynamics of the beam with imposed displacement.

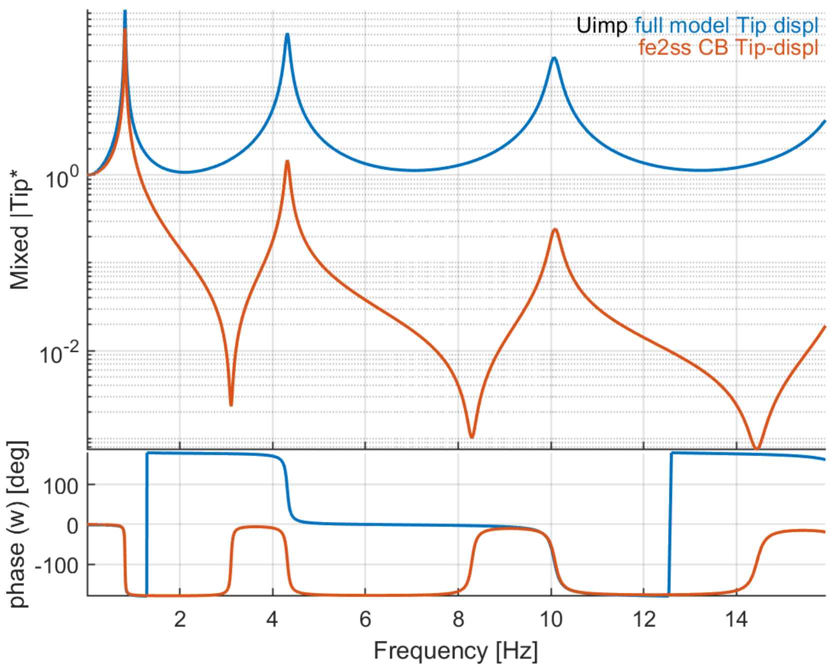


Figure 6.9: Transfer function between horizontal displacement at the bottom and the absolute displacement at the top of the beam

An alternative is to use a transformation of the CB mass and stiffness matrices in order to come back to a situation where the inertial term in the excitation is zero, and only the stiffness term is non-zero, so that fe2ss can be applied to build the state-space model. The transformation is applied so that the retained DOFs are untouched (so that they represent physical DOFs and can be used to simply apply force and measure output quantities), as proposed in [?]. The transform takes the form:

$$\{\tilde{q_C}\} = -\left[M_{CC}\right]^{-1}\left[M_{CI}\right]\{q_I\} + \{q_c\} \tag{6.41}$$

Therefore the matrices can be transformed with the following projection matrix

$$\left[\tilde{T}\right] = \begin{bmatrix} I & 0 \\ -\left[M_{CC}\right]^{-1}\left[M_{CI}\right] & I \end{bmatrix} \tag{6.42}$$

It is straightfowraed to show that after applying the transformation, matrix $\left[\tilde{M_{CI}}\right] = 0$, so that now only the stiffness term appears in the excitation term. After transforming the matrices, fe2ss can be used in a straightforward manner to build an accurate reduced state-space model. This is illustrated below

```
%% Step 3 : Apply Raze transform before making the state-space model

% Transform the initial CB matrices
KCB=SE1.K;
N=size(KCB{1},1);
T1=[ eye(2) zeros(2,N-2) ; - KCB{1}(3:end,3:end)\KCB{1}(3:end,1:2) eye(N-2)];

Mr=T1'*(KCB{1})*T1;
Kr=T1'*(KCB{2})*T1;

% Replace matrices in the CB model
model3=model2;
model3.Stack{1,3}.K{1}=Mr;
model3.Stack{1,3}.K{2}=Kr;

sys2=fe2ss('free 5 5 0 -dterm',model3); u5=freqresp(sys2(1,1),w);
u5 = u5(:);

C6=d_piezo('BuildC1',w'/(2*pi),u5,'Tip-displ','Uimp'); C6.name='fe2ss CB+Raze';
ci=iiplot;iicom(ci,'curveinit',{'curve',C1.name,C1;'curve',C6.name,C6});

iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.10 shows that using `fe2ss` after a CB and the transformation given by (6.42) leads to an accurate model in the frequency band of interest

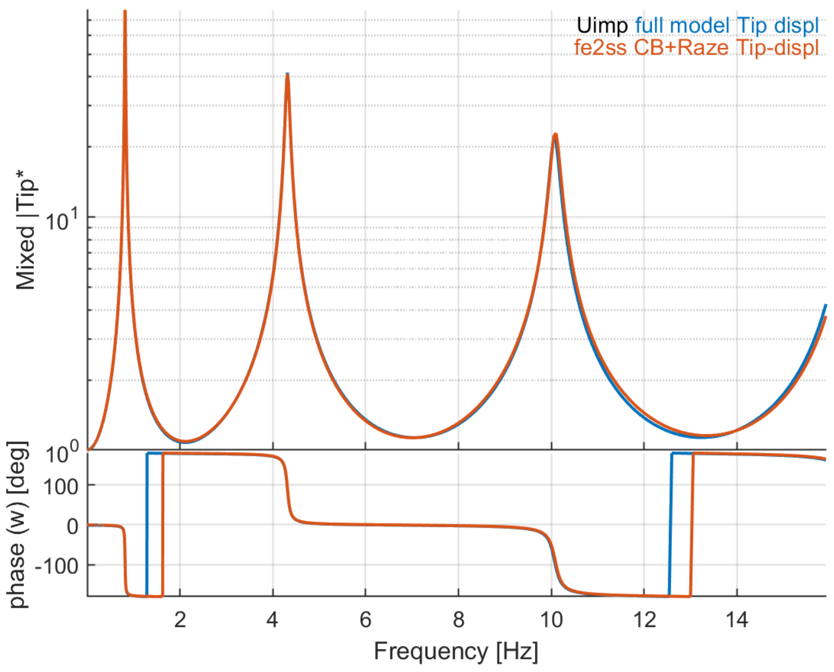


Figure 6.10: Transfer function between horizontal displacement at the bottom and the absolute displacement at the top of the beam

6.4.2 State-space models with imposed accelerations

The case of imposed acceleration is much simpler to treat as `fe2ss` can be used directly after CB reduction in the case of relative displacement. If one is interested in absolute displacement, `nor2ss` can be applied after computing the modeshapes based on the reduced CB matrices and having orthonormalized them. The first script illustrates the construction of a state-space model for relative displacement, using `fe2ss` and the definition of an equivalent load.

```
% See full example as MATLAB code in d_piezo('ScriptTutoTowerSSAimpCB')
d_piezo('DefineStyles');

%% Step 1: reference solution
% build model
model=d_avc('meshtower');
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
model=fe_case(model,'DOFSet','UImp',[1.01]); % Leave x free for imposed displ
```

```
[model,Case] = fe_case('assemble NoT -matdes 2 1 Case -SE',model) ;
K0 = feutilb('tkt',Case.T,model.K); % Full-model

CB    = fe_reduc('craigbampton 5 5',model);
TIn= CB.TR.def(:,1);
F = -Case.T'*model.K{1}*TIn;

w=logspace(-2,2,2048);

for i=1:length(w)
    U0(:,i)=Case.T*((K0{2}*(1+0.02*1i)-w(i)^2*K0{1})\F);
end

CTA = fe_c(model.DOF,21.01); u0 = CTA*U0;
% Change output format to be compatible with iicom
C0=d_piezo('BuildC1',w'/(2*pi),u0.','ur-top','Aimp'); C1.name='Full';

%% Step 2: State-space model using a dofload (relative displ)

% Create super-element type model
model=d_avc('meshtower');
model=fe_case(model,'FixDof','Clamped',[1.06]); % Leave x free for imposed displ
model=fe_case(model,'Remove','F-top'); % Remove point force
SET.DOF=[1.01; 21.01]; SET.def=eye(2);
model=fe_case(model,'DOFSet','UImp',SET); % To retain in CB matrices input and output
model = stack_set(model,'info','EigOpt',[5 5 0]);
model.DOF = feutil('getdof',model);
SE1= fe_reduc('CraigBampton -SE -matdes 2 1 3 4 ',model); % Ne pas utiliser -USEDOF

SE1.DOF(SE1.DOF<0) = 1000.99+(1:numel(SE1.DOF(SE1.DOF<0))); % round(SE1.DOF(SE1.DOF<0)+
SE1 = rmfield(SE1,{'Node','Elt','il','pl','Stack','mdof','TR'}) ; % To keep only the ma

% Initialize the Model
SE0 = struct('Node',[],'Elt',[]);
model2 = fesuper('SEAdd 1 -1 -unique -initcoef -newID se1',SE0,SE1) ;
model0=model2; % Save super-element model without any load BC (2 dofs retained)

% Extract matrices and compute static response to imposed displacement at the base
```

```
KCB=model2.Stack{1,3}.K;
TIn= [1; -KCB{2}(2:end,2:end)\KCB{2}(2:end,1)];

% Compute forcing vector
FCB= -KCB{1}*TIn; FCB=FCB(2:end);

% Block 1.01 and define a DofLoad and a new observation matrix instead
model2=fe_case(model2,'FixDOF','BC',1.01); % BC for relative displacement
SET.DOF=model2.Stack{1,3}.DOF(2:end); SET.def= FCB;
model2=fe_case(model2,'DOFLoad','AImp',SET); % Equivalent load
model2=fe_case(model2,'SensDOF','Output',SET.DOF(1));

sys=fe2ss('free 5 5 0 -dterm',model2);
w=logspace(-2,2,2048);u1=freqresp(sys(1,1),w);
u1 = u1(:);
C1=d_piezo('BuildC1',w'/(2*pi),u1,'ur-top','Aimp'); C1.name='fe2ss 5md+st';
ci=iiplot;iicom(ci,'curveinit',{'curve',C0.name,C0;'curve',C1.name,C1});
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.11 shows that using `fe2ss` with applied load leads to an accurate model in the frequency band of interest

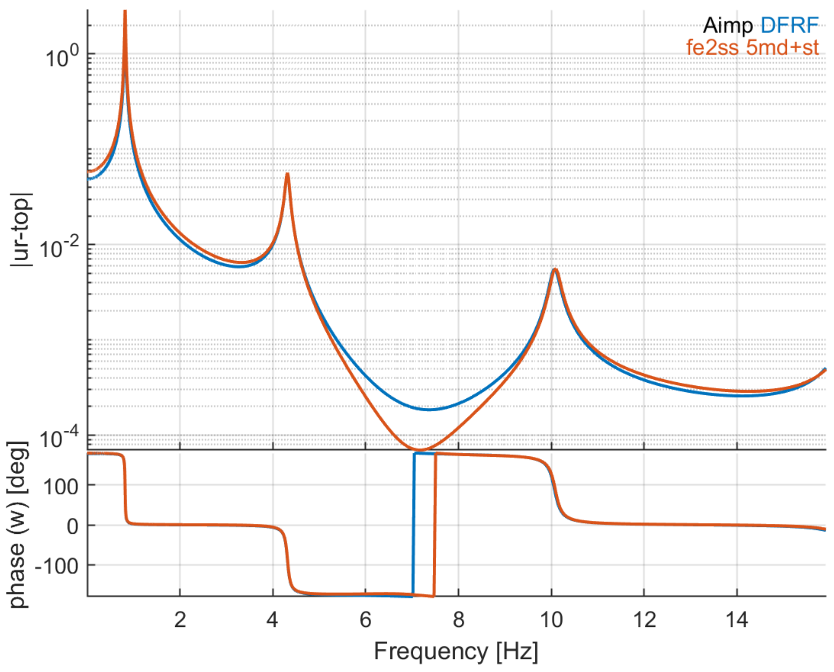


Figure 6.11: Transfer function between horizontal acceleration at the bottom and the relative displacement at the top of the beam

Absolute displacement response can be computed by using `nor2ss`

```
%%Step 3: State-space with absolute displacement

model3=model0; % Initial model without BCs
TR2=fe_eig(model3,[5 7 0]); % Compute modes with CB matrices
SET.DOF=[1.01]; SET.def=eye(1); %
model3=fe_case(model3,'DOFSet','UImp',SET); % Impose acc at bottom
model3=fe_case(model3,'SensDOF','Output',21+.01); % sensor

sys2= nor2ss(TR2,model3) ;
u2=freqresp(sys2(1,1),w); u2=u2(:);
C2=d_piezo('BuildC1',w'/(2*pi),u2,'u-top','Aimp'); C2.name='nor2ss';

% convert reference solution to absolute displacement
u0a=u0-1./w.^2;
C0a=d_piezo('BuildC1',w'/(2*pi),u0a.','u-top','Aimp'); C0a.name='Full';
ci=iiplot;iicom(ci,'curveinit',{'curve',C0a.name,C0a;'curve',C2.name,C2});
```

```
iicom('submagpha')
d_piezo('setstyle',ci);
```

Figure 6.12 shows that using `nor2ss` with imposed acceleration leads to an accurate model in the frequency band of interest

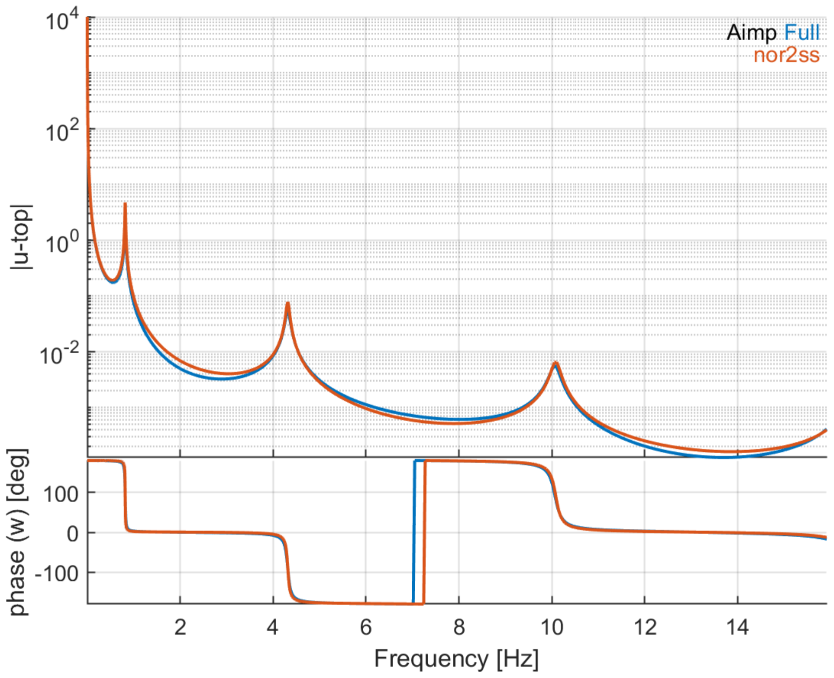


Figure 6.12: Transfer function between horizontal acceleration at the bottom and the absolute displacement at the top of the beam

6.4.3 State-space models with imposed voltage (piezoelectric actuators)

7 Function reference

Contents

Piezo related functions	
d_piezo	support for demonstration of piezo capabilities
p_piezo	piezoelectric volume and shell property handling
m_piezo	piezoelectric material property handling

m_piezo

Purpose

Material function for piezoelectric solids

Syntax

```
mat= m_piezo('database name')
pl = m_piezo('dbval MatId  -elas 12 Name');
```

See section **??** for tutorial calls. Accepted commands are

`[ Database, Dbval] [-unit TY] [,MatiD]] Name`

`m_piezo` contains a number of defaults obtained with the `database` and `dbval` commands which respectively return a structure or an element property row. You can select a particular entry of the database with using a name matching the database entries.
Piezoelectric materials are associated with two material identifiers, the main defines the piezoelectric properties and contains a reference `ElasMatId` to an elastic material used for the elastic properties of the material (see `m_elastic` for input formats).

```
m_piezo('info') % List of materials in data base
% database piezo and elastic properties
pl=m_piezo('dbval 3 -elas 12 SONOX_P502_iso')
```

Theoretical details on piezoelectric materials are given in chapter **??**. The `m_piezo Const` and `BuildConstit` commands support integration constant building for piezo electric volumes integrated in the standard volume elements. Element properties are given by `p_solid` entries, while materials formats are detailed here.

`Patch`

Supports the specification of a number of patches available on the market. The call uses an option structure with fields

- `.name` of the form `ProId`*`val+patchName`*. For example `ProId1+SmartM.MFC-P1.2814`.
- `MatId` value for the initial `MatId`.

`m_piezo('patch')` lists currently implemented geometries. In particular

- `Disk.Material.Geometry` is used for generic circular patches. (Legacy `Noliac.Material.Geometry` is used for circular patches by Noliac). The list of materials in the database is obtained with `m_piezo('info')`. Fields for the geometry are

  - `OD` outer diameter (mm). This can be replaced by `RC`.
  - `TH` Thickness (mm). To specify a milimiter fraction replace the `.` by and `_`. For example `TH0_7` is used for `TH=0.7 mm`.
  - `ID` inner diameter (mm) (optional for piezo rings).
  - `LC` characteristic length for meshing (when mesher has hability to use the information)

- `Rect.Material.Geometry` is used for generic rectangular patches. `SmartM.Material.Geometry` is used for rectangular patches by Smart Materials. Fields for the geometry are
  - `WI` width (mm)
  - `LE` length (mm)
  - `TH` Thickness (mm). To specify a milimiter fraction replace the `.` by and `_`. For example `TH0_7` is used for `TH=0.7 mm`.
  - `LC` characteristic length for meshing (when mesher has hability to use the information)

  Thus `WI28LE14TH_2` is a 28 by 14 by 0.2 mm patch. The geometry can also be coded as `2814TH_2`.

The piezoelectric constants can be declared using the following sub-types

`1:Simplified 3D piezoelectric properties`

`[ProId Type ElasMatId d31 d32 d33 eps1T eps2T eps3T EDType]`
These simplified piezoelectric properties (3.5) can be used for PVDF, but also for PZT if shear mode actuation/sensing is not considered ($d_{24} = d_{15} = 0$). For `EDType==0` on assumes d is given. For `EDType==1`, e is given. Note that the values of ε^T (permitivity at zero stress) should be given (and not ε^S).

`2:General 3D piezo`

`[ProId Type ElasMatId d_1:18 epsT_1:9]`
`d_1:18` are the 18 constants of the $[d]$ matrix (see section 2.1), and `epsT_1:9` are the 9 constants of the $\left[\varepsilon^T\right]$ matrix. One reminds that strains are stored in order xx, yy, zz, yz, zx, yx.

`3 : General 3D piezo, e matrix`

`[ProId Type ElasMatId e_1:18 epsT_1:9]`
`e_1:18` are the 18 constants of the $[d]$ matrix, and `epsT_1:9` are the 9 constants of the $\left[\varepsilon^T\right]$ matrix in the constitutive law (see section 2.1).

See also

`p_piezo`.

p_piezo

Purpose

Property function for piezoelectric shells and utilities associated with piezoelectric models.

Syntax

```
mat= m_piezo('database name')
pl = m_piezo('dbval MatId  -elas 12 Name');
```

See section **??** for tutorial calls. Accepted commands are

`ElectrodeMPC`

`[model,InputDOF(end+1,1)]=p_piezo('ElectrodeMPC Name',model,'z==5e-5');` defines the isopotential constraint as a case entry `Name` associated with `FindNode` command `z==5e-5`. An illustration is given in section **??** .
Accepted command options are

- `-Ground` defines a fixed voltage constraint `FixDof`,`V=0 on` *`Name`*.
- `-Input"InName"` defines an enforced voltage `DofSet`,*`InName`* entry for voltage actuation.
- `MatId`*`i`* is used to define a resultant sensor to measure the charge associated with the electrode. Note that the electrode surface must not be inside the volume with `MatId`*`i`*. If that is the case, you must arbitrarily decompose your mesh in two parts with different `MatId`. You can also generate this sensor a posteriori using `ElectrodeSensQ`, which attempts to determine the `MatId`*`i`* based on the search of a piezoelectric material connected to the MPC.

`ElectrodeSensQ`

`model=p_piezo('ElectrodeSensQ 1682 Q-Base',model);` adds a charge sensor (`resultant`) called `Q-Base` on node `1682`. (See (3.10) for theory).
For **shells**, the node number is used to identify the `p_piezo` shell property and thus the associated elements. It is reminded that `p_piezo` entries must be duplicated when multiple patches are used.
For **volumes**, the `p_piezo ElectrodeMPC` should be first defined, so that it can be used to obtain the electrode surface information.
Note that the command calls `fe_case('SensMatch')` so that changes done to material properties after this call will not be reflected in the observation matrix of this sensor.
To obtain sensor combinations (add charges of multiple sensors as done with specific wiring), specify a data structure with observation `.cta` at multiple `.DOF` as illustrated below.
For a voltage sensor, you can simply use a DOF sensor
`model=fe_case(model,'SensDof','V-Base',1682.21)`.

```
model=d_piezo('meshULBPlate cantilever');  % creates the model
% If you don't remember the electrode node numbers
r2=p_piezo('ElectrodeDOF',model)
% Combined charge
r1=struct('cta',[1 1],'DOF',[r2{1:2,2}]+.21,'name','QS2+3');
model=p_piezo('ElectrodeSensQ',model,r1);
sens=fe_case(model,'sens');
% Combined voltage
r1=struct('cta',[1 1],'DOF',[r2{3:4,2}]+.21,'name','VS2+3');
model=fe_case(model,'SensDof',r1.name,r1);
sens=fe_case(model,'sens');sens.lab
```

ElectrodeDOF

`p_piezo('ElectrodeDof Bottom',model)` returns the DOF the bottom electrode. With no name for selection `p_piezo('ElectrodeDof',model)` the command returns the list of electrode DOFs based on MPC defined using the `ElectrodeMPC` command or `p_piezo` shell entries. Use `ElectrodeDof.*` to get all DOFs.

ElectrodeView ...

`p_piezo('electrodeview',cf)` outlines the electrodes in the model and prints a clear text summary of electrode information. To only get the summary, pass a model `model` rather than a pointer `cf` to a `feplot` figure.

`p_piezo('electrodeviewCharge',cf)` builds a `StressCut` selection allowing the visualization of charge density. You should be aware that only resultant charges at nodes are known. For proper visualization a transformation from charge resultant to charge density is performed, this is known to have problem in certain cases so you are welcome to report difficulties.

Electrode2Case

`Electrode2Case` uses electrode information defined in the obsolete `Electrode` stack entry to generate appropriate case entries : `V_In` for enforced voltage actuators, `V_Out` for voltage measurements, `Q_Out` for charge sensors.

ElectrodeInit

`ElectrodeInit` analyses the model to find electric master DOFs in piezo-electric shell properties or in MPC associated with volume models.

Tab

`Tab` commands are used to generate tabulated information about model contents. The calling format is `p_piezo('TabDD',model)`. With no input argument, the current `feplot` figure is used. Currently generated tabs are

- `TabDD` constitutive laws
- `TabPro` material and element parameters shown as java tables.

`View`

`p_piezo('ViewDD',model)` displays information about piezoelectric constitutive laws in the current model.
`p_piezo('ViewElec ...',model)` is used to visualize the electrical field. An example is given in section **??** . Command options are `DefLen`*`val`* to specify the arrow length, `EltSel`*`val`* for the selection of elements to be viewed, `Reset` to force reinit of selection.
`ViewStrain` and `ViewStress` follow the same calling format.

`Shell element properties`

Piezo shell elements with electrodes are declared by a combination of a mechanical definition as a layered composite, see `p_shell 2`, and an electrode definition with element property rows of the form
`[ProId Type MecaProId ElNodeId1 LayerId1 UNU1 ElNodeId2...]`

- `Type` typically `fe_mat('p_piezo','SI',1)`
- `MecaProId` : `ProId` for mechanical properties of element `p_shell 2` composite entry. The `MatId`*`i`* for piezo layers must be associated with piezo electric material properties.
- `ElNodId1` : `NodeId` for electrode 1. This needs to be a node declared in the model but its position is not used since only the value of the electric potential (DOF 21) is used. You may use a node of the shell but this is not necessary.
- `LayerId` : layer number as declared in the composite entry.
- `UNU1` : currently unused property (angle for polarization)

The constitutive law for a piezoelectric shell are detailed in section 3.2 . The following gives a sample declaration.

```
model=femesh('testquad4'); % Shell MatId 100 ProdId 110

% MatId 1 : steel, MatId 12 : PZT elastic prop
```

```
model.pl=m_elastic('dbval 1  Steel');
% Sonox_P502 piezo material, sdtweb m_piezo('Sonox_P502')
model.pl=m_piezo(model.pl,'dbval 3 -elas 12 SONOX_P502');

% ProId 111 : 3 layer composite (mechanical properties)
model.il=p_shell(model.il,['dbval 111 laminate ' ...
    '3 1e-3 0 ' ...  % MatID 3 (PZT),   1 mm piezo, 0
    '1 2e-3 0  ' ... % MatID 1 (Steel), 2 mm
    '3 1e-3 0']);    % MatID 3 (PZT),   1 mm piezo, 0
% ProId 110 : 3 layer piezo shell with electrodes on nodes 1682 and 1683
model.il=p_piezo(model.il,'dbval 110 shell 111 1682 1 0 1683 3 0');

p_piezo('viewdd',model) % Details about the constitutive law
p_piezo('ElectrodeInfo',model) % Details about the layers
```

d_piezo

Purpose

Support function for piezoelectric demos

Syntax

```
sdtweb('_taglist','d_piezo')  % display contents
```

Accepted commands are

Script

These commands group sample scripts. Use `d_piezo` to display tag list and see available contents.

MeshPlate

Meshing utilities for the placement of piezoelectric patches on a supporting structure (flat plate for now).
The options are specified in a structure with fields

- `.list` : defines a list of features to be introduced with columns giving `name,LamSpec,Geo`, name laminate specification and shape options.
- `.unit` : gives the model unit (needed since patch dimensions are always given in mm).

The laminate specification string is composed of the following

- `BaseId`*i* gives the `ProId` of the base laminate which is then used to figure out the position of patches.
- `+Patch` or `-Patch` to place a patch above or below the base laminate.
- `Patch` itself is a specification of a patch material and geometry. The list of implemented patch can be obtained using `m_piezo Patch`
- `.In` to specify that the patch has an enforced voltage.

The geometry/position specification string `Geo` can be

- a specification of the patch corner and orientation such as `xc=.03 yc=.05 ang=30` if the patch geometry is specified using the laminate specification.
- `rect` shapes `[xc lx nx yc ly ny alpha MatId ProId]` where `MatId` and `ProId` are filled automatically if not provided.

- circ shapes [xc yc rc lc (MatId ProId)]
- global lx=.4 ly=.3 lc=.02 -Sens. The option -Sens generates a sensor entry corresponding to normal displacement of the initial mesh. Alternatively you can add a sensor configuration SensDOF entry see sdtweb('sensor#scell') or sdtweb('sensor#sstruct').

```
% Start by defining properties of the underlying laminate
mdl=struct('Node',[],'Elt',[], ... % empty model
    'pl', ... % composite layer property
      [1 fe_mat('m_elastic','SI',1) 42.5e9 .042 1490 3.35e9 .01], ...
     'il', ... % laminate definition (6 layers at 0,90,0,90,0,90)
     p_shell(['dbval 1 laminate    1 2.167e-4 0    1 2.167e-4 90 ' ...
         '1 2.167e-4 0 1 2.167e-4 90 1 2.167e-4 0 1 2.167e-4 90']), ...
     'unit','SI');
RG=struct;
RG.list={'Name','Lam','shape'
   'Main_plate', mdl,'global lx=.4 ly=.3 lc=.02'
   'Act1','BaseId1 +SmartM.MFC-P1.2814 -SmartM.MFC-P1.2814.in','xc=.35 yc=.25 ang=30'
   'Sen2','BaseId1 +SmartM.MFC-P1.2814','xc=.03 yc=.05 ang=30'
   'Sen3','BaseId1 +Noliac.NCE51.OD25TH1','xc=.05 yc=.25'
   };
cf=feplot;d_piezo('MeshPlate',RG);cf.mdl.name='Plate with piezo';
p_piezo('electrodeinfo',cf.mdl.GetData)
matgui('jil',cf);matgui('jpl',cf); % Display properties
```

The following illustrates transient simulation to a load on a specific piezo

```
model=cf.mdl.GetData; model=p_piezo('electrode2case',model);
opt=fe_time('TimeOpt Newmark .25 .5 0 .3e-6 200');
opt.AssembleCall='assemble -fetime Load';
%opt.FinalCleanupFcn='out.DOF=model.DOF;
%out.def=[Case.T*out.def+Case.TIn*[0 ft(1:end-1)'']];';
model=stack_set(model,'info','Rayleigh',[0 2*.0025/200e3]);
def=fe_time(opt,cf.mdl);def.name=model.name;
cf.def=def; fecom('colordataevalRadZ-edgealpha0');fecom('scc1e-10');
```

Mesh

Patch Simple volume patch.

`Plate` Generic script for arbitrary placement of patches on a flat plate. A list of shapes can be given as a cell array. This is considered as a demo since it currently only supports a rectangular base plate.
`GammaS` build a weighting for surface control.

Bibliography

[1] J. Yang, *An introduction to the theory of piezoelectricity.* Springer, 2010.

[2] “IEEE standards on piezoelectricity, ans n° 176-187, IEEE,” 1988.

[3] A. Deraemaeker and H. Nasser, “Numerical evaluation of the equivalent properties of Macro Fiber Composite (MFC) transducers using periodic homogenization,” *International Journal of Solids and Structures*, vol. 47, pp. 3272–3285, 2010.

[4] “Multiple-support seismic analysis of large structures,” *Computers and Structures*, vol. 36, no. 6, pp. 1153–1158, 1990.

[5] R. Guyan, “Reduction of mass and stiffness matrices,” *AIAA Journal*, vol. 3, p. 380, 1965.

[6] R. J. Craig and M. Bampton, “Coupling of substructures for dynamic analyses,” *AIAA Journal*, vol. 6, no. 7, pp. 1313–1319, 1968.

[7] G. Raze, C. Dumoulin, and A. Deraemaeker, “Reduced-order state-space models of structures with imposed displacements and accelerations,” *Mechanical Systems and Signal Processing*, vol. 191, p. 110156, 2023.