Software Tools for the Characterisation of Coated and Uncoated Mechanical Resonators

F. Fabrizi

July 13, 2022

Contents

1 Introduction

Thermal noise plays a major rôle in limiting the sensitivity curve of modern and next-generation Gravitational Wave detectors. In the quest for high-quality, low-dissipation mirror coatings and suspensions, operating either at room temperature or in cryogenic conditions, measurements of mechanical losses performed in the laboratory have become of paramount importance, to identify and pursue the characterisation of promising candidate materials.

In this context, standard procedures have been defined according to metrology considerations. For instance, investigating coating properties involves the comparison of the mechanical dissipation in two measurements, on the same sample, before and after coating deposition. The mechanical properties of the material employed for the coating may then be extracted.

To accurately interpret such data, experimental care is required in performing the two measurements under the same procedure and conditions. However, an equal emphasis should be placed on an accurate understanding of the many physical factors contributing to the dissipation, and of their mutual interplay, so that the comparison can be done meaningfully. For example, the mere presence of the coating may affect the dissipation within the substrate, so that simply subtracting the data collected on the uncoated sample from the data collected on the coated one, in order to obtain the dissipation within the coating, would potentially be misleading.

This highlights the importance of having a reliable modeling tool at one's disposal. In this respect, consider the effect of thermoelastic noise, which is particularly relevant for crystalline materials (such as silicon, sapphire) possessing large thermal conductivity and a high coefficient of thermal expansion. Modeling of the thermoelastic effect has been tackled both via analitycal and numerical approaches. As for the former, the main analytical models are those put forward by [1, 2, 3, 4]. They can be tailored on the specific geometrical properties of the sample under consideration (a thin two-dimensional disc; a layered structure; a long fiber) [3]. When it comes to constructing the overall loss angle deriving from several physical mechanisms (thermoelastic dissipation in the substrate, plus other sources of dissipation in the substrate, plus dissipation in the coating), the introduction of dilution factors expressing the relative importance of the various contributions becomes important. The models can thus be improved by introducing accurate dilution factors which are dependent on the specific resonant modes being excited [2]. As for the numerical approaches, most research groups rely on the commercially available tools for Finite Element Analysis computations, most notably Ansys and COMSOL.

This variety of theoretical methods and of software tools makes it somewhat challenging to design a campaign of experiments, to contrast simulations carried out at different labratories, and to integrate seamlessly the different potential sources of thermal noise into one model.

We present here an all-in-one software tool for the simulation of mechanical losses, adapted to the sample geometries and the experimental equipments most commonly employed (coated and uncoated discs measured on a GeNS system; fibers measured in clamped configuration). The software aims to provide a direct estimation of the quantities returned by such laboratory experiments. Firstly, leaving aside the dissipation, the information on the resonant frequency of the mechanical modes, and their associated stress and strain fields, is calculated by invoking the Ansys software in a completely automated and standardised manner. Then, dissipation is introduced. The contributions from substrate and coating, and from different physical mechanisms, are considered and weighted accordingly. In particular, the contribution from the thermoelastic effect is calculated analytically, making use of the models mentioned above, whereas contributions from other effects must be estimated by the user. The program then brings the various contributions together, using accurate mode-dependent dilution factors, to give the total value expected from measurements. We hope to facilitate the implementation of a standardised procedure and to alleviate the metrology concerns described above, possibly by extending this modularly-conceived software, in the future, to deal with more sample geometries, experimental conditions, and physical mechanisms underlying the dissipation process.

2 Basic Equations and Definitions used by the Software

The types of mechanical resonator that we consider have the shape of a disc (a cylinder with height << base diameter) or a fiber (a cylinder with height >> base diameter). The disc can be flat or slightly distorted (as if curved on top of a sphere). Moreover, the flat disc can be uncoated (bare substrate) or coated with an identical coating film on both base surfaces, whereas the curved disc and the fiber are always assumed to be uncoated.

The main dissipation processes at work in these resonators are:

- The thermoelastic dissipation in the substrate, which depletes the dilatation energy stored in the substrate;
- The "intrinsic" dissipation in the substrate, which depletes the total elastic energy (dilatation + shear) stored in the substrate. By "intrinsic" we mean here any dissipation mechanism other than thermoelastic, be it due to structural relaxation associated with crystal defects, Two-Level Systems (TLS) in amorphous materials, etc.
- The "intrinsic" dissipation in the coating, which depletes the total elastic energy (dilatation + shear) stored in the coating, and is the quantity we are chiefly interested in measuring experimentally.

Recall that the elastic energy is in fact the sum of two contributions, dilatation and shear:

$$E^{elastic} = E^{dil} + E^{shear}$$

which are related to stress (σ) and strain (ϵ) tensors by

$$U^{elastic} = \frac{1}{2}\,\sigma^{ij}\,\epsilon_{ij}$$

and

$$U^{dil} = \frac{1}{6}\,trace(\sigma)\,trace(\epsilon)$$

$$U^{shear} = \frac{1}{2}\,\sigma_D^{ij}\,\epsilon_{D\,ij}$$

where U are the energy densities (E divided by volume), and the symmetrical tensors σ and ϵ have been decomposed into their spherical (= isotropic) part, and their deviator (= symmetrical traceless) part:

$$\sigma = \sigma_M\,I + \sigma_D = \begin{bmatrix} \sigma_M & 0 & 0 \\ 0 & \sigma_M & 0 \\ 0 & 0 & \sigma_M \end{bmatrix} + \begin{bmatrix} \sigma_{11}-\sigma_M & \sigma_{21} & \sigma_{31} \\ \sigma_{21} & \sigma_{22}-\sigma_M & \sigma_{32} \\ \sigma_{31} & \sigma_{32} & \sigma_{23}-\sigma_M \end{bmatrix}$$

$$I = \text{identity matrix}; \qquad \sigma_M = \frac{1}{3}\,trace(\sigma)$$

and the same for ϵ.

The dilatation energy is thus directly related to the volume change, since the trace of the strain tensor represents the fractional volume change before (V_0) and after (V) distortion:

$$trace(\epsilon) = \frac{\Delta V}{V_0} = \frac{V - V_0}{V_0}$$

Linear eleasticity theory furthermore introduces the notion of stress and strain being linearly related, in the case of small distortion, through the compliance and stiffness tensors, which characterise the material (see Section 3.4.3, Part III, for more details). In finite element analysis, one can find the stress and strain fields defined at each elemental volume of the whole sample, and reconstruct the energies above as the sum over all elements.

Let us introduce the dilution factors bewteen dilatation energy and total elastic energy, in the substrate:

$$D^{TE} = \frac{E_s^{dil}}{E_s^{dil} + E_s^{shear}} \tag{1}$$

and between total elastic energy in the coating and total elastic energy in the whole resonator:

$$D_c = \frac{E_c^{dil} + E_c^{shear}}{E_s^{dil} + E_s^{shear} + E_c^{dil} + E_c^{shear}} \tag{2}$$

from which it follows

$$1 - D_c = \frac{E_s^{dil} + E_s^{shear}}{E_s^{dil} + E_s^{shear} + E_c^{dil} + E_c^{shear}} \tag{3}$$

Then, we define the thickness of the coating (if present) as t (cf. Fig. 1):

t = coating thickness (on **one** side)

so that $t = 0$ corresponds to the case of an uncoated resonator.

With this, we can express the measured loss angles for an uncoated and a coated resonator as the sum of the various contributions mentioned above, in the following way:

$$\begin{cases} \phi_{meas}(0) = \phi_s^{intr} + D^{TE}(0)\phi_s^{TE}(0) & \text{for } t = 0 \text{ (no coating)} \\ \phi_{meas}(t) = \left[1 - D_c(t)\right]\left[\phi_s^{intr} + D^{TE}(t)\phi_s^{TE}(t)\right] + D_c(t)\phi_c^{intr} & \text{for } t \neq 0 \text{ (coating)} \end{cases} \tag{4}$$

where the dependence of the various terms on t has been made explicit. The fact that the coating is on two sides is taken into account within the definition of $D_C(t)$, when calculating the energies in Eqs. (2) and (3). Note that the first equation in (4) represents the case of bare substrate, and the second one the case of a coated sample.

One may wish to express the loss angle of a coated substrate as a function of the loss angle of that same bare substrate, plus the thermoelastic shift $\Delta\phi^{TE}$ (the variation of thermoelastic dissipation *in the substrate* due to the presence *of the coating*), which yields:

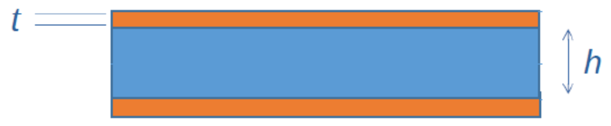


Figure 1: A coated disc, shown in side view. The two layers of coating on the two base faces are assumed to be identical; each one has thickness t. The thickness of the substrate is h, with $h >> t$.

$$\begin{cases} \phi_{meas}(0) \;=\; \phi_s^{intr} + D^{TE}(0)\phi_s^{TE}(0) \\ \phi_{meas}(t) \;=\; \left[1 - D_c(t)\right]\phi_{meas}(0) + \Delta\phi^{TE}(t) + D_c(t)\phi_c^{intr} \\ \Delta\phi^{TE}(t) \;=\; \left[1 - D_c(t)\right]\left[D^{TE}(t)\phi_s^{TE}(t) - D^{TE}(0)\phi_s^{TE}(0)\right] \end{cases} \tag{5}$$

The purpose of the software described in these notes is to provide a calculation of the quantities above, for each mechanical mode, together with some additional parameters such as the resonant frequency, the stress and strain fields and the energy fields.

2.1 Combining multiple dissipation processes

The purpose of this section is to elucidate the way that dilution factors enter into the definition of the loss angles, and therefore to justify the equations presented in the previous section.

Throughout this section, we shall indicate with ψ the loss angles multiplied by the dilution factor, and with ϕ the loss angles proper, related to one specific physical mechanism of dissipation.

If only one physical mechanism is involved, and it depletes the total energy E of the system, then ψ and ϕ coincide, and

$$\psi = \frac{1}{2\pi}\frac{\Delta E}{E}$$

where ΔE is the energy depleted in each cycle (taken with its absolute value, i.e. with a positive sign).

If many physical mechanisms are involved, each depleting energy from a different and separated part of the system, then the total energy is the sum of all these separate parts

$$E = E_1 + E_2 + E_3 ...$$

and the loss angles related to each part are

$$\phi_1 = \frac{1}{2\pi}\frac{\Delta E_1}{E_1}; \qquad \phi_2 = \frac{1}{2\pi}\frac{\Delta E_2}{E_2}; \qquad ...$$

or equivalently, the energies dissipated are

$$\Delta E_1 = 2\pi E_1 \phi_1; \qquad \Delta E_2 = 2\pi E_2 \phi_2; \qquad ...$$

In total, then, one finds that:

$$\psi \;=\; \frac{1}{2\pi}\frac{\Delta E}{E} \;=\; \frac{1}{2\pi}\;\frac{1}{E_1 + E_2 + ...}\;\left(2\pi E_1\phi_1 + 2\pi E_2\phi_2 + ...\right) \;=$$

$$=\; \frac{E_1}{E}\phi_1 + \frac{E_2}{E}\phi_2 + ... \;=$$

$$=\; D_1\phi_1 + D_2\phi_2 + ...$$

where the dilution factors related to the energy stored in each part have been introduced.

If many physical mechanisms are involved, depleting the same kind of stored energy, then the total energy E is being dissipated simultaneously by many dissipation processes:

$$\Delta E \;=\; \Delta E^{m1} + \Delta E^{m2} + ...$$

each related to a specific loss angle:

$$\phi_1 = \frac{1}{2\pi}\frac{\Delta E^{m1}}{E}; \qquad \phi_2 = \frac{1}{2\pi}\frac{\Delta E^{m2}}{E}; \qquad ...$$

or equivalently

$$\Delta E^{m1} = 2\pi E\phi_1; \qquad \Delta E^{m2} = 2\pi E\phi_2; \qquad ...$$

In total:

$$\psi \;=\; \frac{1}{2\pi}\frac{\Delta E}{E} \;=\; \frac{1}{2\pi}\frac{1}{E}\;\left(\Delta E^{m1} + \Delta E^{m2} + ...\right) \;=$$

$$=\; \frac{1}{2\pi}\;\frac{1}{E}\;\left(2\pi E\phi_1 + 2\pi E\phi_2 + ...\right) \;=$$

$$=\; \phi_1 + \phi_2 + ...$$

so that the sum can be done directly over the loss angles ϕ.

Our most general case (disc with coating) is slightly more complicated, involving two separate parts (substrate and coating) and, within the substrate part, two physical mechanisms, one depleting a subset of the energy depleted by the other.

Our total energy is

$$E \;=\; E_s^{dil} + E_s^{shear} + E_c^{dil} + E_c^{shear}$$

and the individual loss angles for each physical mechanism (thermoelastic in the substrate; intrinsic in the substrate; intrinsic in the coating), defined in the previous section, are as follows:

$$\phi_s^{TE} \;=\; \frac{1}{2\pi}\;\frac{\Delta E_s^{dil\;TE}}{E_s^{dil}}$$

$$\phi_s^{intr} = \frac{1}{2\pi}\frac{\Delta\left(E_s^{dil}+E_s^{shear}\right)^{intr}}{E_s^{dil}+E_s^{shear}}$$

$$\phi_c^{intr} = \frac{1}{2\pi}\frac{\Delta\left(E_c^{dil}+E_c^{shear}\right)^{intr}}{E_c^{dil}+E_c^{shear}}$$

To find the total loss, then, one considers

$$\psi = \frac{1}{2\pi}\frac{\Delta E}{E}$$

where the energies are being depleted as follows

$$\psi = \frac{1}{2\pi E}\left(\Delta E_c^{dil\ intr}+\Delta E_c^{shear\ intr}+\Delta E_s^{dil\ intr}+\Delta E_s^{dil\ TE}+\Delta E_s^{shear\ intr}\right) =$$

$$= \frac{1}{2\pi E}\left[2\pi\left(E_c^{dil}+E_c^{shear}\right)\phi_c^{intr}+2\pi\left(E_s^{dil}+E_s^{shear}\right)\phi_s^{intr}+2\pi E_s^{dil}\phi_s^{TE}\right] =$$

$$= D_c\phi_c^{intr}+(1-D_c)\phi_s^{intr}+(1-D_c)D^{TE}\phi_s^{TE}$$

introducing the dilution factors as defined in the previous section, and considering that

$$(1-D_c)D^{TE} = \frac{E_s^{dil}}{E_s^{dil}+E_s^{shear}}\frac{E_s^{dil}+E_s^{shear}}{E} = \frac{E_s^{dil}}{E}$$

2.2 Modeling the Thermoelastic Effect

The purpose of this section is to illustrate how the specific loss angle for the thermoelastic dissipation in the substrate, ψ^{TE}, is being calculated by the program, in the various configurations available (coated disc, uncoated disc, fiber).

Object of cylindrical symmerty, any aspect ratio, uncoated (= only substrate)
As shown in [2, 3], the thermoelastic loss angle is represented as a series of Debye peaks, with a common pre-factor:

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} = D^{TE}\frac{(3\alpha_L)^2BT}{C_v}\sum_{nmk}^{\infty}C_{nmk}\frac{\omega\,\omega_{peak\ nmk}}{\omega^2+\omega_{peak\ nmk}^2}$$

where B is the elastic bulk modulus, $C_v = \rho c_v$ is the volumetric heat capacity (heat capacity per volume, or specific heat multiplied by the density), and α_L is the linear thermal expansion ($3\alpha_L = \alpha_v$ is the volumetric thermal expansion, considering here an isotropic material). The coefficients C_{nmk} of the series and the positions $\omega_{peak\ nmk}$ of the Debye peaks depend on the geometry and on the physical properties of the sample.

D^{TE} is the dilution factor defined as the ratio between dilatation energy and elastic energy *in the same part of the sample*, consistent with our definition in the previous sections involving only the substrate.

Considering only the first Debye mode as dominant, one assumes that for this mode $C_{nmk} \approx 1$ and all other coefficients in the series are negligible. Calling the peak position of the first mode $\omega_{peak\ nmk} = \omega_{peak}$, we have:

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} \approx D^{TE}\frac{(3\alpha_L)^2 BT}{C_v}\frac{\omega\,\omega_{peak}}{\omega^2+\omega_{peak}^2}$$

Generally speaking, the dilution factor D^{TE} depends on the stress and strain fields induced in the sample, and is therefore dependent on the mechanical modes being excited.

Fiber (= one-dimensional, circular section), uncoated (only substrate)

The results above can be particularised to the case of a fiber, i.e. a cylinder whose aspect ratio is such that heigth >> base diameter. It has been shown by [3] that in this case,

$$D^{TE} = \frac{E}{9B} \quad \text{for every mode}$$

and

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} \approx \frac{\alpha_L^2 ET}{C_v}\frac{\omega\,\omega_{peak}}{\omega^2+\omega_{peak}^2}$$

where α_L is again the linear thermal expansion and E is Young's modulus (if the material is anisotropic, the parameters involved should be along the fiber axis), and ω_{peak} is:

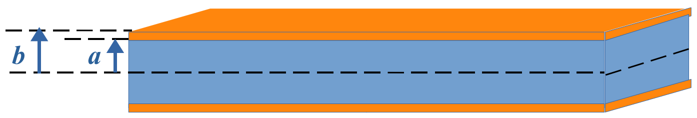
$$\omega_{peak} = (2\cdot 1.841)^2\frac{k}{C_v d^2}$$

where d is the diameter of the base of the fiber, and k is the thermal conductivity (transverse to the fiber axis if the material is anisotropic, since in this geometry it is assumed that there is small to no heat flow along that axis).
The expression above is the one implemented by this program in the case of resonator type CantileverFiber.

Beam (= one-dimensional, rectangular section), coated

We turn now to a different model.

Vengallatore's model [1] has the major advantage of being applicable to coated specimens. It considers a one-dimensional (beam) layered structure, in which coating has been deposited on two sides, as in figure:



The substrate has thickness $2a$, and each coating layer has thickness $b-a$. The model is derived as a simplification of Bishop and Kinra [4]. It suffers however from the simplifying assumption of considering Poisson's effect to be negligible. It also makes the assumption that both materials are isotropic, and we will make some simple adjustments here to extend the results to anisotropic ones.

The dissipation in this model is again expressed as a series of Debye peaks:

$$\psi_{s+c}^{TE} = D^{TE}\phi_{s+c}^{TE} = = \psi_{01} \frac{1}{\frac{1}{3}\left[\frac{a^2}{b^2} + \frac{E_2}{E}\left(1 - \frac{a^2}{b^2}\right)\right]} \sum_{n=1}^{\infty} Q_n \frac{\omega\, \omega_{peak,n}}{\omega^2 + \omega_{peak,n}^2}$$

where

$$\omega_{peak,n} = \frac{k}{C_v a^2}\gamma_n^2$$

E and E_2 are the Young's moduli of substrate and coating, respectively.
k and C_v are the thermal conductivity and the volumetric heat capacity of the substrate.
ψ_{01}, Q_n and γ_n depend on the physical properties and the geometry of the structure.
Since in our case the thickness of the substrate is much greater than the thickness of the coating, we will make the assumption that ψ_{01} can be reduced to the Zener's modulus of the sole substrate material [5, 6]:

$$\psi_{s+c}^{TE} = D^{TE}\phi_{s+c}^{TE} = \frac{\alpha_L^2 ET}{C_v} \frac{1}{\frac{1}{3}\left[\frac{a^2}{b^2} + \frac{E_2}{E_1}\left(1 - \frac{a^2}{b^2}\right)\right]} \sum_{n=1}^{\infty} Q_n \frac{\omega\, \omega_{peak,n}}{\omega^2 + \omega_{peak,n}^2}$$

where the linear thermal expansion α_L, the volumetric heat capacity C_v, and the Young's modulus E in the pre-factor all refer to the substrate.

Coated Beam –> Coated Disc

We want to use Vengallatore's model because it allows us to model a coated sample, but we want to adapt it from a 1-dimensional structure (a beam) to a 2-dimensional one (a thin disc).

To do so, consider that implicit in Vengallatore's formula is the assumption that the dilution factors are

$$D^{TE} = \frac{E}{9B} \quad \text{for every mode}$$

as is the case for a 1-dimensional structure (such as the fiber, seen above).

Therefore, we adjust the model simply by dividing the expression by $E/9B$ and re-multiplying it by the proper dilution factors D^{TE} of a two-dimensional disc (intended to be calculated numerically for each resonant mode).

We obtain:

$$\psi_{s+c}^{TE} = D^{TE}\phi_{s+c}^{TE} = D^{TE}\frac{(3\alpha_L)^2 BT}{C_v} \frac{1}{\frac{1}{3}\left[\frac{a^2}{b^2} + \frac{E_2}{E_1}\left(1 - \frac{a^2}{b^2}\right)\right]} \sum_{n=1}^{\infty} Q_n \frac{\omega\, \omega_{peak,n}}{\omega^2 + \omega_{peak,n}^2}$$

Moreover, we consider that, in the case of anisotropic materials, the only relevant heat flow direction is the one along the disc axis, and we assume that the physical values of k are selected accordingly along this direction; conversely, α_L and E are selected to be transverse to the disc axis.

Note that, rigorously speaking, the thermoelastic dissipation calculated here occurs in all layers of the structure; however, in our case of thin coating, we can assume that the contribution of the coating to the thermoelastic dissipation is negligible. All of the thermoelastic loss occurs in the substrate, although its magnitude *in the substrate* is influenced by the properties *of the coating*, whose presence changes the boundary conditions.

The above expression is the one implemented by this program in the case of resonator type Doubly-CoatedDisc.

Beam (= one-dimensional, rectangular section), uncoated

The same Vengallatore's model can be used to model an uncoated sample, in the limit where the coating thickness vanishes (b –> a).

In this case:

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} = \frac{\alpha_L^2 ET}{C_v} \, 6 \sum_{n=1}^{\infty} \frac{\omega\,\omega_{peak,n}}{\omega^2 + \omega_{peak,n}^2} \frac{1}{\gamma_n^4}$$

with simplified expressions for γ_n:

$$\gamma_n = (2n-1)\frac{\pi}{2}\,; \qquad n = 1, 2, 3...$$

This series converges rapidly. Considering only the first term, one finds

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} = \frac{\alpha_L^2 ET}{C_v} \, \frac{96}{\pi^4} \, \frac{\omega\,\omega_{peak,1}}{\omega^2 + \omega_{peak,1}^2}$$

$$\approx \frac{\alpha_L^2 ET}{C_v} \, \frac{\omega\,\omega_{peak,1}}{\omega^2 + \omega_{peak,1}^2}$$

since

$$\frac{96}{\pi^4} \approx 1$$

and

$$\omega_{peak,1} = \frac{k}{C_v (2a)^2}\pi^2$$

which reduces to the usual formula for a beam found by Zener.

Unoated Beam –> Uncoated Disc

Again, we make the same correction in the implicit dilution factors, to account for our two-dimensional geometry. We obtain:

$$\psi_s^{TE} = D^{TE}\phi_s^{TE} = D^{TE}\,\frac{(3\alpha_L)^2 BT}{C_v} \, 6 \sum_{n=1}^{\infty} \frac{\omega\,\omega_{peak,n}}{\omega^2 + \omega_{peak,n}^2} \frac{1}{\gamma_n^4}$$

with simplified expressions for γ_n:

$$\gamma_n = (2n-1)\frac{\pi}{2}\,; \qquad n = 1, 2, 3...$$

where D^{TE} are the mode-dpendent dilution factors of a thin disc, to be calculated numerically.

For an anisotropic material, the same considerations stated above still hold.

The expression above is the one implemented by this program in the case of resonator type UncoatedDisc.

3 Software

3.1 Overview

What the program does

The purpose of this software is to streamline the calculation of mechanical losses, such as those measured by GeNS systems in the laboratory. Specifically, the user can select a geometry among a number of options typically associated with GeNS experiments or similar measuring apparatus (uncoated disc, doubly coated disc, clamped fiber, etc.); the software then calculates the resonance frequency of the various mechanical modes, the associated stress and strain fields and their elastic energy, and the mechanical loss due to thermoelastic effect.

In particular, with respect to the quantities in Eqs. (4) and (5), one finds that:

- the thermoelastic loss angle of a bare substrate $\phi_s^{TE}(0)$ and the mode-dependent dilution factor $D^{TE}(0)$ can be calculated by the program;

- the intrinsic loss angle ϕ_s^{intr} is considered negligible in this first implementation of our program, given the high-quality crystalline materials chiefly under consideration;

- the dilution factor between coating and whole resonator (substrate + coating) $D_c(t)$ can be calculated;

- the thermoelastic loss angle of the substrate in the presence of coating $\phi_s^{TE}(t)$ and the corresponding dilution factor $D^{TE}(t)$ can in principle be calculated, however, they are dependent on the physical parameters of the coating which may not be known accurately;

- the intrinsic dissipation in the coating ϕ_c^{intr} is the main quantity of interest that typically one wishes to extract from the measured quantities $\phi_{meas}(0)$ and $\phi_{meas}(t)$. It cannot be calculated by the program, however, if its order of magnitude is known and given as an input, the program will then incorporate it to return the overall uncertainty expected from the measurement, which may serve as a guide to define the experimental strategy.

The program aims to provide a complete simulation tool dealing with all steps of the calculation mentioned above, with results that can be readily compared with experiments. This "all-in-one" approach has several advantages, namely:

. it ensures consistency of definitions and conventions throughout the various steps,
. it provides easy to use tools for non-expert users, and
. it is practical and quick, especially in the case of repeated calculations, such as within the context of fitting and optimisation routines.

To expand on these points, the input required from the user is minimal, since the program is already focused on the samples and procedures typically employed in research on gravitational wave detectors. Another key feature is that this software is designed to be fully automatic, in that it does not require user input beyond the initial settings. This makes it suitable to repeated, automatic calls, such as parameter fitting. Finally, the software has been developed in a modular fashion, by using an object-oriented approach, and assigning an object to each type of resonator and to each step of the calculation. This is to make it easily extendable, in the future, to the new requirements imposed by the evolution of the experimental techniques and the materials under investigation, to new sample geometries, and to new models of mechanical dissipation, beyond the thermoelastic effect which has been our main focus so far.

How the program is constructed

The program is comprised of a series of routines written in MatLab [7]. They are organised in a hierarchichal way, with one main variable containing as separate attributes the input data, the calculation settings and the results.

For the first part of the calculation, focusing on finding the mechanical vibration modes without considering the dissipation, the program makes use of the software Ansys [8], a commercial package intended for engineering simulation and 3D design.

The calculation flow

The whole calculation flow, as depicted in detail in Fig. 2, is articulated in three steps:

- Create a software object containing the information pertaining to:
  - sample geometry (define the type of resonator: uncoated/coated disc, fiber, etc.; specify the dimensions of each part)
  - material of the different parts
  - calculation settings as required by the user
- Calculate the vibration modes with Ansys, in particular, for each mode:
  - resonant frequency
  - stress and strain fields (for each element and in total for that mode)
  - elastic energy, separated into its dilatation and shear contributions
  - the dilution factor D^{TE}
- Calculate the thermoelastic dissipation
  - thermoelastic loss angle (mode independent) as a function of frequency: ϕ^{TE}
  - thermoelastic loss angle as a function of frequency, inclusive of the mode-dependent dilution factors: $D^{TE}\phi^{TE}$
- Optional, stand-alone calculations on material properties (outside of this workflow)

The main MatLab variable

All of the calculation is contained in a single, self-consistent MatLab variable. This includes the input data, the settings required for the computation, and the results.

This main MatLab variable is structured as an object with various attributes and methods. Many of the attributes are also object themselves, arranged in a hierarchichal fashion, as per the scheme in Fig. 3. In particular, some of the most important attributes are:
- The **Body** object, which contains information on the different parts of the resonator (substrate, coating if present). Each part is itself divided into objects, one containing information on the material, the others on the geometrical shape of the part.
- The **Calculation Settings** and **Mesh Settings**, which contain the parameters according to which the calculation should be done.
- The **Calculation Results**, which is automatically populated with the findings from the computations as they are run.

The details of the object structure outlined above can somewhat change, depending on the type of resonator that is being considered. All available options for the type are:

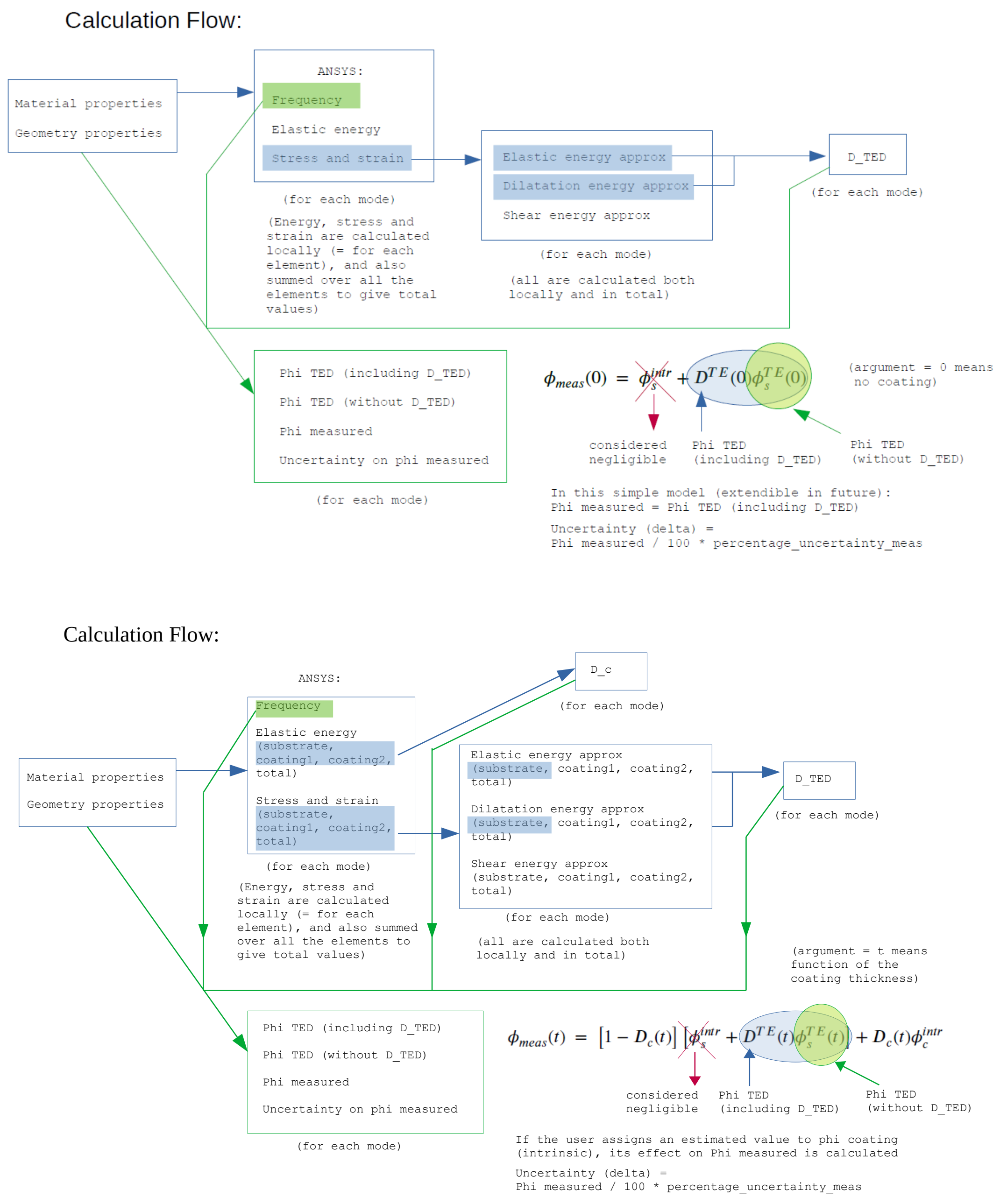

Calculation Flow:
Material properties
Geometry properties
ANSYS:
Frequency
Elastic energy
Stress and strain
(for each mode)
(Energy, stress and strain are calculated locally (= for each element), and also summed over all the elements to give total values)
Elastic energy approx
Dilatation energy approx
Shear energy approx
(for each mode)
(all are calculated both locally and in total)
D_TED
(for each mode)
Phi TED (including D_TED)
Phi TED (without D_TED)
Phi measured
Uncertainty on phi measured
(for each mode)
$\phi_{meas}(0) = \cancel{\phi_s^{intr}} + D^{TE}(0)\phi_s^{TE}(0)$
(argument = 0 means no coating)
considered negligible
Phi TED (including D_TED)
Phi TED (without D_TED)
In this simple model (extendible in future):
Phi measured = Phi TED (including D_TED)
Uncertainty (delta) =
Phi measured / 100 * percentage_uncertainty_meas
Calculation Flow:
ANSYS:
Frequency
Elastic energy (substrate, coating1, coating2, total)
Stress and strain (substrate, coating1, coating2, total)
(for each mode)
(Energy, stress and strain are calculated locally (= for each element), and also summed over all the elements to give total values)
Material properties
Geometry properties
D_c
(for each mode)
Elastic energy approx (substrate, coating1, coating2, total)
Dilatation energy approx (substrate, coating1, coating2, total)
Shear energy approx (substrate, coating1, coating2, total)
(for each mode)
(all are calculated both locally and in total)
D_TED
(for each mode)
(argument = t means function of the coating thickness)
Phi TED (including D_TED)
Phi TED (without D_TED)
Phi measured
Uncertainty on phi measured
(for each mode)
$\phi_{meas}(t) = [1 - D_c(t)]\,[\cancel{\phi_s^{intr}} + D^{TE}(t)\phi_s^{TE}(t)] + D_c(t)\phi_c^{intr}$
considered negligible
Phi TED (including D_TED)
Phi TED (without D_TED)
If the user assigns an estimated value to phi coating (intrinsic), its effect on Phi measured is calculated
Uncertainty (delta) =
Phi measured / 100 * percentage_uncertainty_meas

Figure 2: Panel above: The overall calculation flow performed by the program, for the case of an uncoated resonator. Panel below: The same, for the case of a coated resonator. The first step of the calculations (mechanical modes with Ansys software) corresponds to the blue arrows, the second step (dissipation) to the green arrows.

- **Disc** A disc whose thickness can be considered small, compared to its diameter. It is made of a single material and has no coating.
- **DoublyCoatedDisc** Same as Disc, but with the addition of two thin layers of coating on the two base surfaces. The coatings share the same thickness and are made of the same material, which can be different from the material of the substrate.
- **CurvedDisc** Same as Disc, but the geometry has a slight spherical curvature. It is intended to mimic possible deformation effects induced by thermal annealing, aging, etc.
- **CantileverFiber** A fiber whose diameter can be considered small, compared to its length. This is the opposite approximation with respect to the Disc. One extremity of the fiber is constrained to be fixed in space (contrary to the objects above, which are left free). This is meant to replicate the typical measurement conditions in the laboratory for a long fiber: as opposed to the objects above, which can be measured on a GeNS apparatus, a fiber is most easily measured in a cantilever configuration.

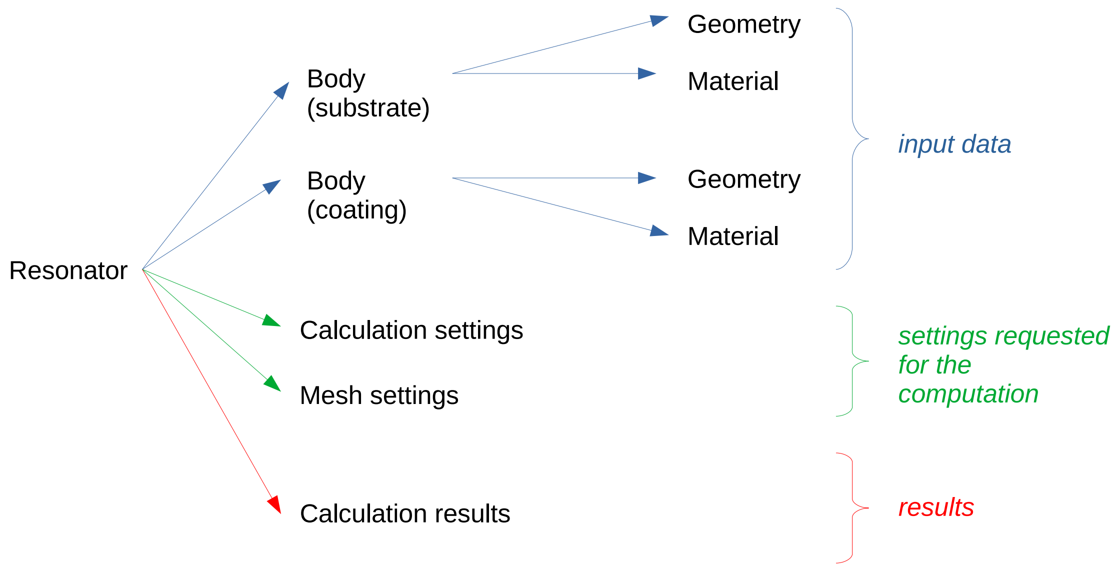


Figure 3: Scheme of a typical structure of the main MatLab variable.

How to use the program

What follows is a user's guide giving step-by-step instructions on how to use the program.

First, we present four tutorials (one for each resonator type) which can be used as quick templates for one's own programs. Some basic comments and explanations are included within the code.

Then, in the following section, we present a more detailed tutorial, in which all the available options for the various steps are discussed in depth. For the sake of simplicity, the resonator type "uncoated disc" has been chosen for this.

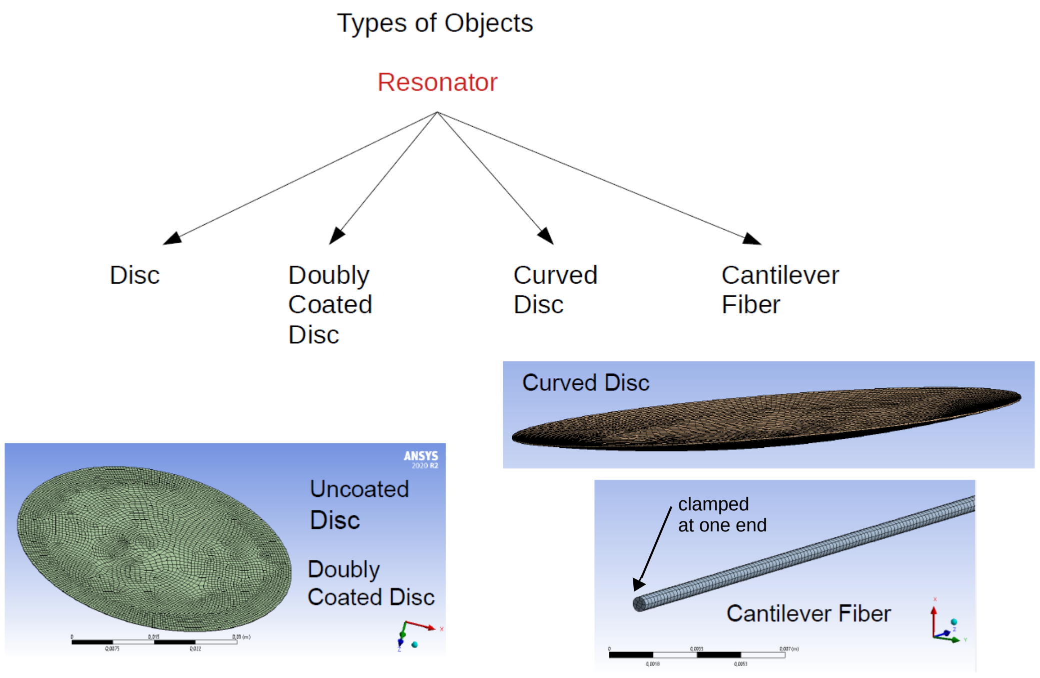


Figure 4: The types of mechanical resonator handled by the program.

3.2 Getting started

Before running the program for the first time on a local machine, the user should perform the following steps:

1) the file *Resonator.m* (in folder *software_core_routines*) should be edited to point to the location of the Ansys executables installed on the machine. That is, the lines:

```
path_ansys_exe = 'C:\Program Files\ANSYS Inc\v202\ansys\bin\winx64\ANSYS202.exe';
path_workbench_exe = 'C:\Program Files\ANSYS Inc\v202\Framework\bin\Win64\RunWB2.exe';
```

should be edited to point to the actual locations where the Ansys MAPDL solver and interpreter, and the Ansys WorkBench environment, can be found.

2) the main folder *software_core_routines*, containing the program files, should be added to the MatLab path:

```
addpath('..\software_core_routines');
```

where the line above should be edited to match the actual location chosen for the folder on the machine.

3.3 Quick Tutorials to be used as Templates

3.3.1 Example 1. Uncoated Disc

This script is available in the file *script_demo1.m*, within the folder *software_core_routines*.

Step 1: create a disc; define its geometry properties.

An uncoated disc, of thickness = 300 micron and diameter = 1 inch.

```
% Create object of type Disc, with associated folder
mydisc1 = Disc();
mydisc1 = mydisc1.set_folder('..\myfolder1');
mydisc1 = mydisc1.reinitialise();   % in case you run this script more than once

% Set properties:

% 1) Geometry properties (thickness = 300 micron and diameter = 1 inch)

mydisc1.substrate.diameter = 0.0254;
mydisc1.substrate.thickness = .3e-3;
```

Step 2: define the material properties. Here are templates for several different cases.

2-a The material properties are loaded from the **database** for silica. The material is in **amorphous** state.

```
% 2) Material properties

mydisc1.substrate.material = Material('silica');

mydisc1.substrate.material.flag_anisotropic = 'amorph';  % already the default
```

2-b The material properties are loaded from the **database** for silicon. The material is in **single crystal** state. The crystallographic axis (1 1 1) is oriented perpendicularly to the disc surface. We give this direction in a Cartesian frame, which coincides with the crystallographic frame, since the crystal symmetry is cubic.

```
% 2) Material properties

mydisc1.substrate.material = Material('silicon');

mydisc1.substrate.material.flag_anisotropic = 'single_crystal';
mydisc1.substrate.material.orientation_axis_1 = [1 1 1];
```

2-c The material properties are loaded from the **database** for silicon. The material is in **polycrystalline** state.

```
% 2) Material properties

mydisc1.substrate.material = Material('silicon');

mydisc1.substrate.material.flag_anisotropic = 'poly';
```

2-d The material properties are **inserted manually**, rather than loaded from the database. The material is in amorphous state; however, this can be done for a material in any state.

```
% 2) Material properties

mydisc1.substrate.material = Material();

mydisc1.substrate.material.flag_anisotropic = 'amorph';  % already the default

mydisc1.substrate.material.density = 2200;  % [kg m^-3]
mydisc1.substrate.material.specific_heat_capacity = 770;  % [J kg^-1 K^-1]
mydisc1.substrate.material.thermal_conductivity_amorph = 1.38;  % [W m^-1 K^-1]
mydisc1.substrate.material.thermal_linear_expansion_amorph = 0.5e-6;  % [K^-1]
mydisc1.substrate.material.young_amorph = 73e9;  % [Pa]
mydisc1.substrate.material.poisson_amorph = 0.16;  % [dimensionless]
```

Step 3: Set the temperature. Here are templates for two different cases.

3-a: The temperature is 300 K.

```
% 3) Set the temperature

mydisc1.substrate.material = mydisc1.substrate.material.set_temperature(300);
```

3-b: The temperature is very low (10 K).

If this is out of range for some of the material properties in the database, we instruct the program to automatically load the value at the lowest temperature available.

```
% 3) Set the temperature

mydisc1.substrate.material = mydisc1.substrate.material.extend_to_zero_temperature();
mydisc1.substrate.material = mydisc1.substrate.material.set_temperature(10);
```

Step 4: Configure the mesh settings.

4-a Use the sweep method (uniform layers across the disc thickness). Recommended for thermoelastic calculations.

Determine the number of layers (optional) and the element size (optional).

```
% 4) Mesh settings

mydisc1.mesh_settings.method = ’sweep’;
mydisc1.mesh_settings.divisions = 5;  % no. of elements along the thickness

mydisc1.mesh_settings.fineness = mydisc1.substrate.diameter/20;  % element size
```

4-b Use the non-sweep method (Ansys’s default).

Leave the element size to be determined by Ansys.

(Alternatively, you can define the element size with the parameter *fineness*, as in the example 4-a).

```
% 4) Mesh settings

mydisc1.mesh_settings.method = ’non-sweep’;

mydisc1.mesh_settings.fineness = 0;  % already the default
```

Step 5 to 10: Configure the calculation settings; perform the (non-dissipative) mode calculation with Ansys; examine the detailed results for one particular mode (stress and strain fields; etc.); calculate the thermoelastic dissipation according to Vengallatore's model; display the results; save the MatLab variable so that it can be uploaded again in a later session.

```
% 5) Calculation settings

% Let Ansys create a list of modes, according to the following constraints
mydisc1.calculation_settings.max_number_of_modes = 20;
mydisc1.calculation_settings.max_frequency = 100000;
mydisc1.calculation_settings.min_frequency = 100;  % already the default

% Create a vector-list of the modes of interest, whose results will be saved.
% (The numbers in this vector-list are referred to the position occupied in the
% Ansys's list above; i.e. mode_list = [2,4] selects the 2nd and the 4th mode
% found by Ansys, within the constraints specified)

mydisc1.calculation_settings.mode_list = [2,4];  % modes of interest

% The results for each mode will be saved in a sub-folder labeled by
% the mode number, e.g. "2" and "4" in this case

% 6) Calculate with Ansys

workbench_out = mydisc1.ansys_workbench_calculate();  % prepare input file
mydisc1 = mydisc1.ansys_apdl_calculate();  % edit input file and execute

% 7) Inspect Ansys results for one specific mode (Mode 4)

% Refer to the mode by the position occupied in the Ansys's list above;
% i.e. mode_number = 4 selects the 4th mode found by Ansys,
% and stored in sub-folder "4",
% corresponding to the second mode in the vector-list mode_list
% and to the second position in all vectorial results loaded into mydisc1
% (frequency_modes; elastic_energy_modes; etc.)

mode_number = 4;

element_volumes = mydisc1.get_element_volume(mode_number);
element_volume_changes = mydisc1.get_element_volumechange(mode_number);

[strainXX, strainYY, strainZZ, strainXY, strainYZ, strainXZ] = ...
    mydisc1.get_element_strain(mode_number);
[stressXX, stressYY, stressZZ, stressXY, stressYZ, stressXZ] = ...
    mydisc1.get_element_stress(mode_number);
```

```
elastic_energy = mydisc1.get_element_elastic_energy(mode_number);
[elastic_energy_approx,dilatation_energy_approx,shear_energy_approx] = ...
    mydisc1.get_element_energy_approx(mode_number);

% 8) Calculate Thermoelastic Dissipation according to Vengallatore's model

mydisc1 = mydisc1.calculate_TED_vengallatore();
mydisc1 = mydisc1.calculate_dissipation_TED();

% 9) Inspect results

disp(mydisc1.calculation_results);

% 10) Save the MatLab variable "mydisc1" (within the project folder)

mydisc1.save();
```

3.3.2 Example 2. Doubly Coated Disc

This script is available in the file *script_demo2.m*, within the folder *software_core_routines*.

A doubly coated disc, of subtstrate thickness = 200 micron, radius = 1 inch, and coating thickness = 1 micron.

The material properties of the substrate are loaded from the database for silicon, oriented with crystallographic axis (0 0 1) perpendicular to the surface. The material properties of the substrate are loaded from the database for silica.

The temperature is 200 K.

All the variations in the possible input presented for an Uncoated Disc can still be applied here, even if not explicitly stated.

```
% Create object of type DoublyCoatedDisc, with associated folder
mydisc2 = DoublyCoatedDisc();
mydisc2 = mydisc2.set_folder('..\myfolder2');
mydisc2 = mydisc2.reinitialise();   % in case you run this script more than once

% Set properties:

% 1) Geometry properties

mydisc2.substrate.diameter = 0.0254*2;
mydisc2.substrate.thickness = .2e-3;

mydisc2.coating.thickness = 1e-6;
```

```
% 2) Material properties:

mydisc2.substrate.material = Material('silicon');
mydisc2.substrate.material.flag_anisotropic = 'single_crystal';
mydisc2.substrate.material.orientation_axis_1 = [0 0 1];

mydisc2.coating.material = Material('silica');

% 3) Set the temperature

mydisc2.substrate.material = mydisc2.substrate.material.set_temperature(200);
mydisc2.coating.material = mydisc2.coating.material.set_temperature(200);

% 4) Mesh settings

mydisc2.mesh_settings.method = 'sweep';
mydisc2.mesh_settings.divisions = 5;  % no. of elements along the thickness
mydisc2.mesh_settings.fineness = mydisc2.substrate.diameter/20;  % element size

% 5) Calculation settings

mydisc2.calculation_settings.max_number_of_modes = 10;
mydisc2.calculation_settings.mode_list = [2:5];

% 6) Calculate with Ansys

workbench_out = mydisc2.ansys_workbench_calculate();
mydisc2 = mydisc2.ansys_apdl_calculate();

% 7) Inspect Ansys results for one specific mode (Mode 4)

% Refer to the mode by the position occupied in the Ansys's list above;
% i.e. mode_number = 4 selects the 4th mode found by Ansys,
% and stored in sub-folder "4",
% corresponding to the third mode in the vector list mode_list
% and to the third position in all vectorial results loaded into mydisc2
% (frequency_modes; elastic_energy_modes; etc.)

mode_number = 4;

element_volumes = mydisc2.get_element_volume(mode_number);
element_volume_changes = mydisc2.get_element_volumechange(mode_number);

[strainXX, strainYY, strainZZ, strainXY, strainYZ, strainXZ] = ...
```

```
    mydisc2.get_element_strain(mode_number);
[stressXX, stressYY, stressZZ, stressXY, stressYZ, stressXZ] = ...
    mydisc2.get_element_stress(mode_number);

elastic_energy = mydisc2.get_element_elastic_energy(mode_number);
[elastic_energy_approx,dilatation_energy_approx,shear_energy_approx] = ...
    mydisc2.get_element_energy_approx(mode_number);

% 8) Calculate Thermoelastic Dissipation with Vengallatore's model

mydisc2 = mydisc2.calculate_TED_vengallatore(2);
mydisc2 = mydisc2.calculate_dissipation_TED_plus_coating();

% 9) Inspect results

disp(mydisc2.calculation_results);

% 10) Save the MatLab variable "mydisc2" (within the project folder)

mydisc2.save();
```

3.3.3 Example 3. Curved Uncoated Disc

This script is available in the file *script_demo3.m*, within the folder *software_core_routines*.

An uncoated disc, of thickness = 300 micron, and diameter = 1 inch. The disc is not flat, but curved as if cut out from a sphere; in other words, it follows the curved surface of a spherical cap. This sphere is characterised by a radius of 1 m (a flat disc would correspond to an infinite radius).

The material properties are loaded from the database for silica.

The temperature is 200 K.

All the variations in the possible input presented for an Uncoated Disc can still be applied here, even if not explicitly stated.

```
% Create object of type CurvedDisc, with associated folder

mydisc3 = CurvedDisc();
mydisc3 = mydisc3.set_folder('..\myfolder3');
mydisc3 = mydisc3.reinitialise();   % in case you run this script more than once

% Set properties:

% 1) Geometry properties:

mydisc3.substrate.diameter = 0.0254;
```

```
mydisc3.substrate.thickness = .3e-3;

mydisc3 = mydisc3.set_curvature(1);  % Curvature radius (supposed to be large)

% 2) Material properties:

mydisc3.substrate.material = Material('silica');

% 3) Set the temperature:

mydisc3.substrate.material = mydisc3.substrate.material.set_temperature(300);

% 4) Mesh settings

mydisc3.mesh_settings.method = 'sweep';
mydisc3.mesh_settings.divisions = 5;  % no. of elements along the thickness
mydisc3.mesh_settings.fineness = mydisc3.substrate.diameter/20;  % element size

% 5) Calculation settings

% Let Ansys create a list of modes, according to the following constraints
mydisc3.calculation_settings.max_number_of_modes = 20;
mydisc3.calculation_settings.max_frequency = 100000;
mydisc3.calculation_settings.min_frequency = 100;  % already the default

% Create a vector-list of the modes of interest, whose results will be saved.
% (The numbers in this vector-list are referred to the position occupied in the
% Ansys's list above; i.e. mode_list = [10:15] selects the 10th to 15th among the modes
% found by Ansys, within the constraints specified)

mydisc3.calculation_settings.mode_list = [10:15];  % modes of interest

% The results for each mode will be saved in a sub-folder labeled by
% the mode number, e.g. "10" to "15" in this case

% 6) Calculate with Ansys

workbench_out = mydisc3.ansys_workbench_calculate();
mydisc3 = mydisc3.ansys_apdl_calculate();

% 7) Save the MatLab variable "mydisc3" (within the project folder)

mydisc3.save();
```

Now let us show how the previous calculation on a curved sample can be used for a comparison against another calculation, performed on the same sample, but flat. The aim is to find the shifts in the resonant frequencies, and to classify them according to the kind of modes being considered.

The script to do so is as follows (it should be appended to the previous script):

```
% Create object of type Disc (flat), with associated folder,
% and assign to it the same properties as the previous object

mydisc0 = Disc();
mydisc0 = mydisc0.set_folder('..\myfolder3-flat');
mydisc0 = mydisc0.reinitialise();

mydisc0.substrate.diameter = mydisc3.substrate.diameter;
mydisc0.substrate.thickness = mydisc3.substrate.thickness;

material_key = mydisc3.substrate.material.keyword;
temperature = mydisc3.substrate.material.temperature;

mydisc0.substrate.material = Material(material_key);
mydisc0.substrate.material = mydisc0.substrate.material.set_temperature(temperature);

mydisc0.mesh_settings.method = mydisc3.mesh_settings.method;
mydisc0.mesh_settings.divisions = mydisc3.mesh_settings.divisions;
mydisc0.mesh_settings.fineness = mydisc3.mesh_settings.fineness;

mydisc0.calculation_settings.max_number_of_modes = ...
    mydisc3.calculation_settings.max_number_of_modes;
mydisc0.calculation_settings.max_frequency = mydisc3.calculation_settings.max_frequency;
mydisc0.calculation_settings.min_frequency = mydisc3.calculation_settings.min_frequency;

mydisc0.calculation_settings.mode_list = mydisc3.calculation_settings.mode_list;

workbench_out0 = mydisc0.ansys_workbench_calculate();
mydisc0 = mydisc0.ansys_apdl_calculate();

mydisc0.save();

%%%%%%%%%%%%%%%%%%%%%%%%%%

% Identify the kind of mechanical modes found by Ansys.
% To do this, you can open Ansys in interactive mode, and run in DesignModeler
% the file "myfolder3\support_files\geom.js", generated by this program.
% Then, in Mechanical, set the material properties and run the macro
% "myfolder3\support_files\mech_edited.py", generated by this program.
% Check that the kind of modes follows the same sequence for both
% curved and flat objects; if not, make the necessary adjustments to this script.

% With the parameters assigned in this script:

butterfly_mode_numbers = [1,2,4,5,8,9,10,11,15,16];
mixed_circ1_mode_numbers = [6,7,12,13,17,18];
mixed_circ2_mode_numbers = [19,20];
drum_mode_numbers = [3,14];

% Convert the mode number in Ansys's list into the position occupied by the
% mode in the result vectors, e.g. the 10th Ansys mode is the first one of the
% vector-list (= the first mode we asked for, in defining the mode_list).

butterfly_mode_indexes = ...
    find(ismember(mydisc3.calculation_settings.mode_list,butterfly_mode_numbers)==1);
mixed_circ1_mode_indexes = ...
    find(ismember(mydisc3.calculation_settings.mode_list,mixed_circ1_mode_numbers)==1);
mixed_circ2_mode_indexes = ...
    find(ismember(mydisc3.calculation_settings.mode_list,mixed_circ2_mode_numbers)==1);
drum_mode_indexes = ...
    find(ismember(mydisc3.calculation_settings.mode_list,drum_mode_numbers)==1);

flat_butterfly   = mydisc0.calculation_results.frequency_modes(butterfly_mode_indexes);
curved_butterfly = mydisc3.calculation_results.frequency_modes(butterfly_mode_indexes);

flat_mixed_circ1   = mydisc0.calculation_results.frequency_modes(mixed_circ1_mode_indexes);
curved_mixed_circ1 = mydisc3.calculation_results.frequency_modes(mixed_circ1_mode_indexes);

flat_mixed_circ2   = mydisc0.calculation_results.frequency_modes(mixed_circ2_mode_indexes);
curved_mixed_circ2 = mydisc3.calculation_results.frequency_modes(mixed_circ2_mode_indexes);

flat_drum   = mydisc0.calculation_results.frequency_modes(drum_mode_indexes);
curved_drum = mydisc3.calculation_results.frequency_modes(drum_mode_indexes);

% Calculate and plot the difference in resonant frequency
% between curved and flat objects

butterfly = (curved_butterfly - flat_butterfly)./flat_butterfly*100;
mixed_circ1 = (curved_mixed_circ1 - flat_mixed_circ1)./flat_mixed_circ1*100;
mixed_circ2 = (curved_mixed_circ2 - flat_mixed_circ2)./flat_mixed_circ2*100;
drum = (curved_drum - flat_drum)./flat_drum*100;

figure();
hold on;
```

```
frequencies = {flat_butterfly,flat_mixed_circ1,flat_mixed_circ2,flat_drum};
differences = {butterfly,mixed_circ1,mixed_circ2,drum};
names = ["butterfly","mixed (1)","mixed (2)","drum"];
colours = ['b','r','m','g'];
h = [];
for jj =1:4
    if ~isempty(frequencies{jj})
        h(end+1) = plot(frequencies{jj},differences{jj},'o','MarkerFaceColor',colours(jj),...
            'MarkerEdgeColor',colours(jj),'DisplayName',names(jj));
    end
end
h(end+1) = yline(0,'-k','DisplayName','');
xlabel('Frequency (flat) [Hz]');
ylabel('Frequency Variation [%]')
legend(h(1:end));
```

3.3.4 Example 4. Cantilever Fiber

This script is available in the file *script_demo4.m*, within the folder *software_core_routines*.

A fiber clamped at one extremity, of length = 335 mm, and diameter = 2 mm.

This resonator type is suitable for cylindrical objects with length >> diameter, whereas it was the opposite case for the discs seen so far. Note that the length of the fiber is still called “thickness” in the context of this program, to keep consistency with the name given to the attributes of the other resonator types (discs).

The material properties are loaded from the database for silica.

The temperature is 300 K.

All the variations in the possible input presented for an Uncoated Disc can still be applied here, even if not explicitly stated.

```
% Create object of type CantileverFiber, with associated folder

myfiber = CantileverFiber();
myfiber = myfiber.set_folder('..\myfolder4');
myfiber = myfiber.reinitialise();   % in case you run this script more than once

% Set properties:

% 1) Geometry properties:

myfiber.substrate.diameter = 2e-3;     % [m] 2 mm
myfiber.substrate.thickness = 335e-3;  % [m] 335 mm

% 2) Material properties:

myfiber.substrate.material = Material('silica');
```

```
% 3) Set the temperature:

myfiber.substrate.material = myfiber.substrate.material.set_temperature(300);

% 4) Mesh settings

myfiber.mesh_settings.method = 'non-sweep';
myfiber.mesh_settings.fineness = 0;  % already the default

% 5) Calculation settings

% Let Ansys create a list of modes, according to the following constraints
myfiber.calculation_settings.max_number_of_modes = 20;
myfiber.calculation_settings.max_frequency = 100000;
myfiber.calculation_settings.min_frequency = 100;  % already the default

% Create a vector-list of the modes of interest, whose results will be saved.
% (The numbers in this vector-list are referred to the position occupied in the
% Ansys's list above; i.e. mode_list = [2,4] selects the 2nd and the 4th mode
% found by Ansys, within the constraints specified)

myfiber.calculation_settings.mode_list = [2,4];  % modes of interest

% The results for each mode will be saved in a sub-folder labeled by
% the mode number, e.g. "2" and "4" in this case

% 6) Calculate with Ansys

workbench_out = myfiber.ansys_workbench_calculate();
myfiber = myfiber.ansys_apdl_calculate();

% 8) Calculate Thermoelastic Dissipation according to Zener's model

myfiber = myfiber.calculate_TED_zener();
myfiber = myfiber.calculate_dissipation_TED();

% 9) Inspect results

disp(myfiber.calculation_results);

% 10) Save the MatLab variable "myfiber" (within the project folder)

myfiber.save();
```

3.4 Tutorial with Detailed Instructions

The first thing to do is to initialise an object. To do so, simply create an instance of the type of resonator you wish to use. For example, to create an instance of type "Disc" called *mydisc*, type:

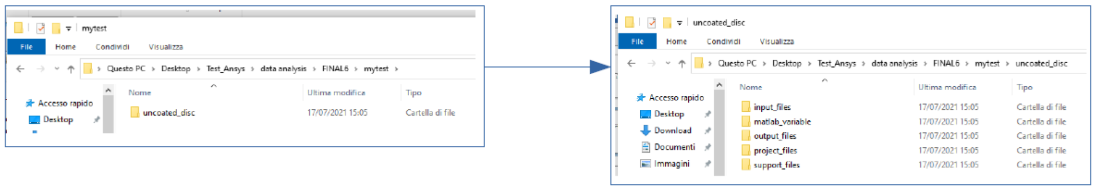
```
mydisc = Disc();
```

This will create an empty MatLab object, with the structure of a Disc (see Fig. 3).

Immediately after initialising the variable *mydisc*, the next thing to do is to assign a folder to it, which will contain the structure depicted in Fig. 5. Either an absolute or a relative path can be used. The name of the folder can be chosen by the user, and does not have to match the name given to the MatLab variable. For instance, the variable *mydisc* can be associated with the folder *uncoated_disc*, placed in the current MatLab working directory, in the following way:

```
mydisc = mydisc.set_folder('.\uncoated_disc');
```

This will create the folder named *uncoated_disc* and also the various subfolders within it. It will not populate them with files as of yet. Their purpose is:
1) to store the MatLab variable *mydisc*, which will contain the input parameters and the main results of the calculations.
2) to store separately the other results of the calculations, which are too onerous to be conveniently included within the MatLab variable *mydisc* (such as element-by-element values for the stress and strain fields). They still can be easily accessed in case of need.
3) to keep a record of the files automatically created by the program and given as an input to the Ansys software, and of Ansys's log and output files.


mytest
File
Home
Condividi
Visualizza
Questo PC › Desktop › Test_Ansys › data analysis › FINAL6 › mytest ›
Accesso rapido
Desktop
uncoated_disc
File
Home
Condividi
Visualizza
Questo PC › Desktop › Test_Ansys › data analysis › FINAL6 › mytest › uncoated_disc
Accesso rapido
Desktop
Download
Documenti
Immagini

Nome	Ultima modifica	Tipo
uncoated_disc	17/07/2021 15:05	Cartella di file

Nome	Ultima modifica	Tipo
input_files	17/07/2021 15:05	Cartella di file
matlab_variable	17/07/2021 15:05	Cartella di file
output_files	17/07/2021 15:05	Cartella di file
project_files	17/07/2021 15:05	Cartella di file
support_files	17/07/2021 15:05	Cartella di file

Figure 5: The folder created by the program for this project, together with its default subfolders.

This concludes the initialisation. Once the type of resonator and the associated folder have been defined, the next step is to set up the calculation by defining the parameters of the resonator (dimensions and materials) and the requirements of the computation (precision of the mesh, frequency range of interest, etc.).

However, if one simply wishes to become familiarised with the program, by making use of a sample initialised with some common, plausible values, one can use the following function to get something quick to work with:

```
mydisc = mydisc.set_defaults();
```

If, at any point, one wishes to re-start from scratch, keeping only the resonator type, the variable name (*mydisc*), and the associated folder, one may type:

```
mydisc = mydisc.reinitialise();
```

Note that the previous command will **delete all of the old data**. That includes all content generated so far and saved in the subfolders. The MatLab variable itself (*mydisc*) will keep its structure and its resonator type, but will have all its attributes emptied. Only the association with the assignated folder will remain.

Finally, if one wishes to save the MatLab variable, one can use the command:

```
mydisc.save();
```

The variable will be saved in the dedicated subfolder "*matlab_variable*", placed within the associated folder *uncoated_disc*. It can be re-loaded in a subsequent MatLab session with:

```
myvar = load('path\to\object_folder\matlab_variable\resonator.mat').resonator;
```

The structure and contents of the saved variable get uploaded into a new variable, named *myvar* in this example (the name may be the same or it may different from the one used previously).

3.4.1 Setting the Resonator Properties

After the initialisation is complete, the next step is to define 1) the resonator properties (geometry and material) and 2) the required settings for the calculations.

All physical properties are meant to be given in SI standard units.

In this section, we will describe how to define the resonator properties.

In the case of a Disc object, there is only one body to be defined, the substrate.

3.4.2 Geometry properties

In the case of a variable of type Disc, only two geometrical parameters are needed for the substrate: the diameter and the thickness.

Note that the calculations for the thermoelastic dissipation will be done under the assumption that the thisckness is orders of magnitude smaller than the diameter. If this is not the case, you may consider using another cylindrical resonator type, such as CantileverFiber.

To define the dimensions, use the following commands. All dimensions must be given in meters. For instance:

```
mydisc.substrate.diameter = 0.0254*2;
mydisc.substrate.thickness = .1e-3;
```

defines a disc of 2 inches diameter (radius is 1 inch, or 2.54 cm) and 100 micron thickness.

3.4.3 Material properties

Part I - How they are defined

In order to define the material, we need to distinguish between **physical properties** and **state of the material**. The meaning of these terms within the context of the program is explained in the following paragraphs.

The **physical properties** of the material are stored in the attribute *material* of the substrate. They can be grouped into three categories:

- the *isotropic properties* collect the properties of the material when it is found in amorphous state (e.g. SiO_2 in silica form) and in polycrystalline state (completely random grain orientation). These values are all scalars.

- the *anisotropic properties* collect the fully tensorial properties of the material when it is found in the state of a single crystal (or at least with one strongly preferred grain orientation). These values are tensors and are referred to a specific Material Reference Frame, described below.

- the *directional properties* are to be used when the material is single crystal, as in the point above, but the full tensor of the property is not known. The properties measured along *some* of the crystallographic directions can be stored here (see the variable structure below). In this case, the program uses the most suitable values it can find, according to procedures which will be explained in the following.

The list of possible properties, grouped by category, is represented in Fig. 6. There is no need to define all of the properties: those that are not admissible, known, or relevant for the physics of a given material, can be left empty.

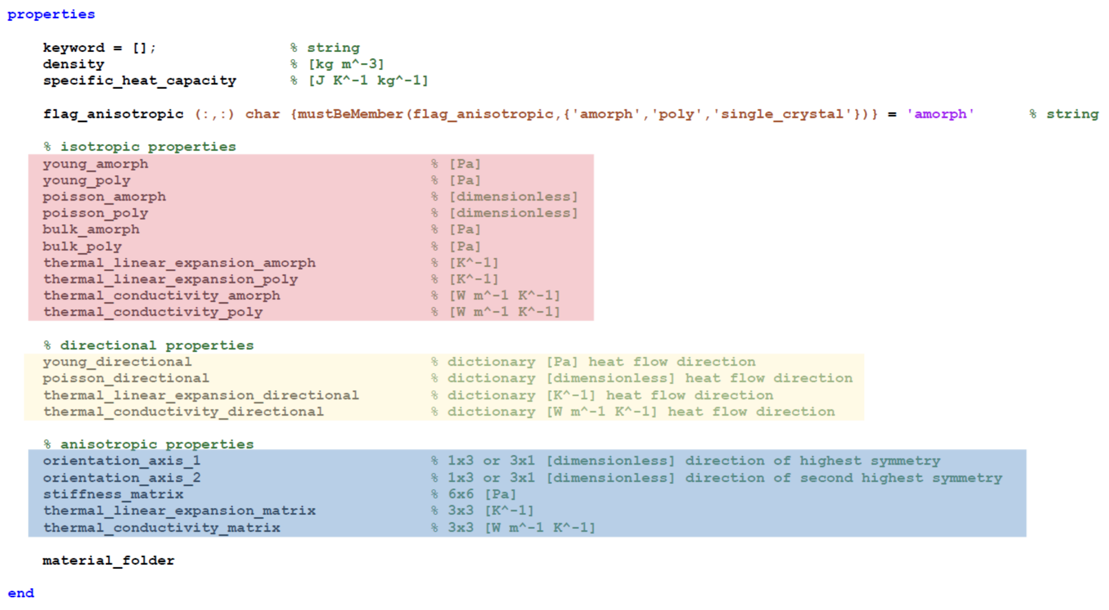
Each directional property is organised as a MatLab "struct" (a dictionary), whose name fields are keywords formed by "*direction_*" and the indexes of the specific direction, in coordinates of the Material Reference Frame. For instance, a Young's modulus directional variable is as shown in Fig. 7. A negative coordinate is expressed in the keyword by prefacing it with the "m" letter. It is expected that each coordinate does not exceed one (signed) digit. The values can be accessed and set by commands such as this:

```
mydisc2.substrate.material.young_directional.direction_001
```

Once the relevant properties are set, one needs to specify in which state the material is found. The program can work with three **states of material**:

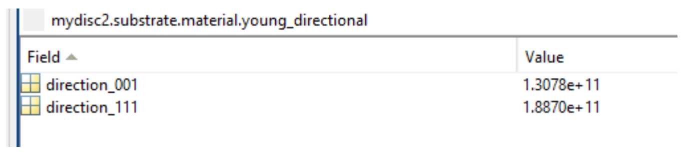
- *amorphous* material (e.g. silica).
- *polycrystalline* material (completely random grain orientation)
- *single crystal* material (or at least with one strongly preferred grain orientation)

You can define what kind of material you are working with by setting the attribute *flag_anisotropic* to one of the three possible keywords *amorph*, *poly* or *single_crystal*, in the following way (when initialising a new object, the flag is *amorph* by default):



```
properties

    keyword = [];                   % string
    density                         % [kg m^-3]
    specific_heat_capacity          % [J K^-1 kg^-1]

    flag_anisotropic (:,:) char {mustBeMember(flag_anisotropic,{'amorph','poly','single_crystal'})} = 'amorph'      % string

    % isotropic properties
    young_amorph                                    % [Pa]
    young_poly                                      % [Pa]
    poisson_amorph                                  % [dimensionless]
    poisson_poly                                    % [dimensionless]
    bulk_amorph                                     % [Pa]
    bulk_poly                                       % [Pa]
    thermal_linear_expansion_amorph                 % [K^-1]
    thermal_linear_expansion_poly                   % [K^-1]
    thermal_conductivity_amorph                     % [W m^-1 K^-1]
    thermal_conductivity_poly                       % [W m^-1 K^-1]

    % directional properties
    young_directional                               % dictionary [Pa] heat flow direction
    poisson_directional                             % dictionary [dimensionless] heat flow direction
    thermal_linear_expansion_directional            % dictionary [K^-1] heat flow direction
    thermal_conductivity_directional                % dictionary [W m^-1 K^-1] heat flow direction

    % anisotropic properties
    orientation_axis_1                              % 1x3 or 3x1 [dimensionless] direction of highest symmetry
    orientation_axis_2                              % 1x3 or 3x1 [dimensionless] direction of second highest symmetry
    stiffness_matrix                                % 6x6 [Pa]
    thermal_linear_expansion_matrix                 % 3x3 [K^-1]
    thermal_conductivity_matrix                     % 3x3 [W m^-1 K^-1]

    material_folder

end
```

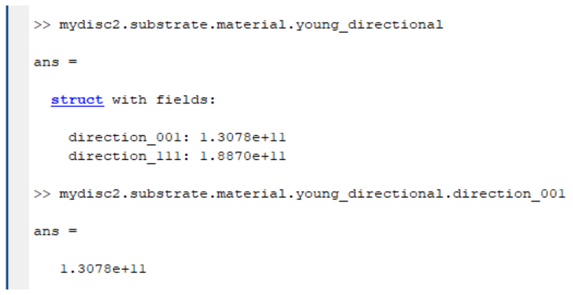
Figure 6: List of attributes in the MatLab object *material*, representing the physical properties. They are divided in three groups (emphasised by colour).



mydisc2.substrate.material.young_directional

Field	Value
direction_001	1.3078e+11
direction_111	1.8870e+11

From the MatLab “Command Window”:



```
>> mydisc2.substrate.material.young_directional

ans = 

  struct with fields:

    direction_001: 1.3078e+11
    direction_111: 1.8870e+11

>> mydisc2.substrate.material.young_directional.direction_001

ans =

   1.3078e+11
```

Figure 7: Structure of a directional property in *material*: how it appears when displayed in the Variables window (above), and the way to access it from the Command Window (below).

```
mydisc.substrate.material.flag_anisotropic = 'single_crystal';
```

Depending on the state of the material that has been specified, the program chooses the appropriate category for the properties to be used in the calculations. However, **there isn't necessarily a one-to-one match between the state of the material and the category of the properties that will be used. Given the state, the program will look for the best possible option, restricted to the properties which are not empty.** For instance, if the state has been specified as *amorph*, then the program looks for the Young's modulus in the *young_amorph* attribute of the isotropic properties. However, if there are no data available for this attribute, the program then selects the next best property: for instance, when the *young_amorph* variable is empty, it might use the *young_poly* variable instead. The exact rules with which the properties are selected at each step of the calculations are detailed in the following sections.

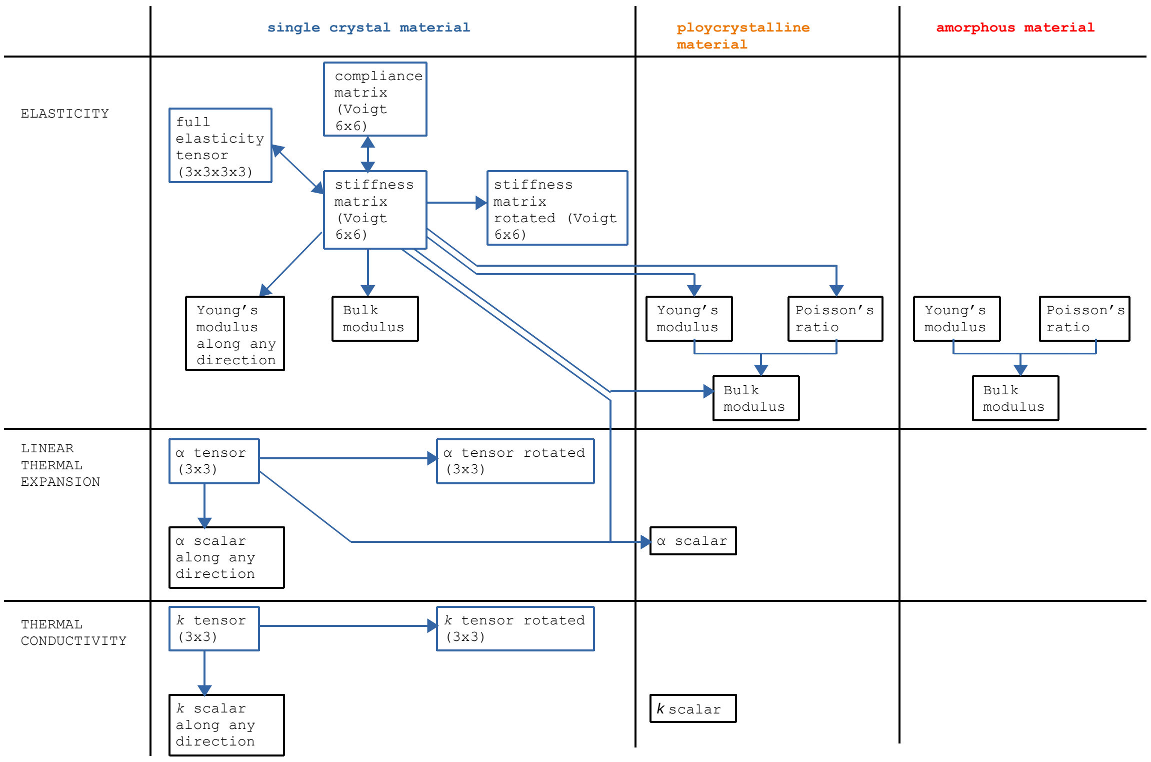


Figure 8: Table of the functions implemented by the program, to derive one property from another(s). They are used automatically during the calculations, but can also be called indipendently as stand-alone functions by the user (see section "Other useful functions", below).

Reference frames for single crystals
In the case of a single crystal, the tensorial properties must be referred to a specific reference frame. There are three reference frames that can be useful to consider:
1) The crystal reference frame (CRF). The axes are dictated by the symmetry of the crystal, according to

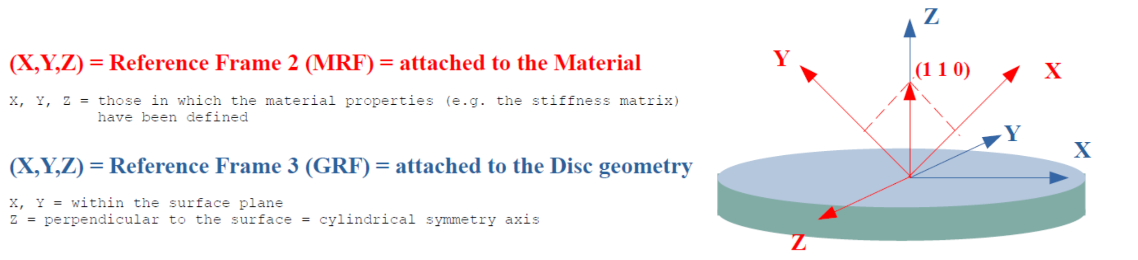

(X,Y,Z) = Reference Frame 2 (MRF) = attached to the Material
X, Y, Z = those in which the material properties (e.g. the stiffness matrix) have been defined
(X,Y,Z) = Reference Frame 3 (GRF) = attached to the Disc geometry
X, Y = within the surface plane
Z = perpendicular to the surface = cylindrical symmetry axis
Z
Y
(1 1 0)
X
Y
X
Z

Figure 9: Reference frames attached to the material properties (MRF) and to the geometray (GRF). Their resepective orientation, in this example, is given to the program by setting orientation_axis_1 = [1 1 0].

space group rules. The frame is not necessarily orthonormal. This frame is not uused by the program itself, however, the user may wish to keep it in mind when assigning values to the properties in frame 2. If the properties are not assigned by the user, but loaded from a database, then the orientation between frames 1 and 2 is recorded in the folder dedicated to that material (see Part II below).

2) The material reference frame (MRF). The orientation of the axes is fixed with respect to the crystallographic axes of reference frame 1, however, this frame is Cartesian (that is, orthonormal). It can be thought of as an ortho-normalisation of frame 1. This is the frame in which the anisotropic material properties are given to the program, and stored as attributes of the MatLab variable.

3) The geometry reference frame (GRF). This is also orthonormal, but in this case, the orientation of the axes is fixed with respect to the geometrical shape of the resonator. The third axis (Z) is always defined as the axis of maximum symmetry; in the case of a Disc resonator type, it coincides with the axis of cylindrical symmetry. The second axis (Y) lies in the plane perpendicular to the first axis, and is oriented along the direction of maximum symmetry within that plane. If no such preferred orientation selected by symmetry exists (as is the case for a Disc in the plane perpendicular to the cylindrical axis), then an arbitrary direction within the plane is chosen by the program. The first axis (X) is simply defined in order to complete the right-handed orthonormal triad (X, Y, Z).

The important orientation to be specified to the program is the one between reference frames 2 and 3. This is done in the following way:

```
mydisc.substrate.material.orientation_axis_1 = [1 1 0];
```

The attribute *orientation_axis_1* is the direction of the third basis vector (Z) of frame 3 (the Geometry Frame), expressed in coordinates of frame 2 (the Material Frame). In the case of a resonator of type Disc, it is the direction perpendicular to the surface, expressed in the reference frame of the material. The attribute *orientation_axis_2* does not need to be specified for the class Disc, because of cylindrical symmetry. The coordinates do not need to be normalised to match a vector of unitary length.

For instance, setting orientation_axis_1 = [1 1 0] will set up the orientation depicted in Fig. 9.

This information will be used by both Ansys and by the routines for thermoelastic calculations.

A final word should be given about the **elastic properties**.

To set all of the elastic properties for a single crystal, only the stiffness matrix and its orientation need to be given.

The stiffness matrix must be given in Voigt notation (dimensions are 6x6). As all material properties, it is referred to the orthonormal frame 2 (the Material Frame), which does not necessarily correspond to the crystallographic axes. Its symmetry properties can be taken into account by the user when specifying the values. They will simplify the matrix by constraining some of the values to be equal, or zero [9, 10]. However, the crystallographic symmetry is not taken into account by the MatLab program, which simply performs the calculations numerically, so that they are valid for any symmetry.

Part II - How they are set

There are two ways to set the properties of a material:

- For a few materials (identified by keywords, such as ‘silicon’, ‘silica’, ...) the user can load the values from a database. The database is stored in folder *temperature_dependent_material_properties*, which must be located at the same level as the folder *software_core_routines* containing the MatLab code.
- For materials that are not in the database (or if you want to use specific, accurate values for your sample), the properties can be set by hand.

Let us now examine each way in turn.

Load from database

Templates of how to format the data are given in the folder *temperature_dependent_material_properties*, for two materials (an amorphous one and a single crystal). The database is meant to be expanded by the user with the materials they deem useful.

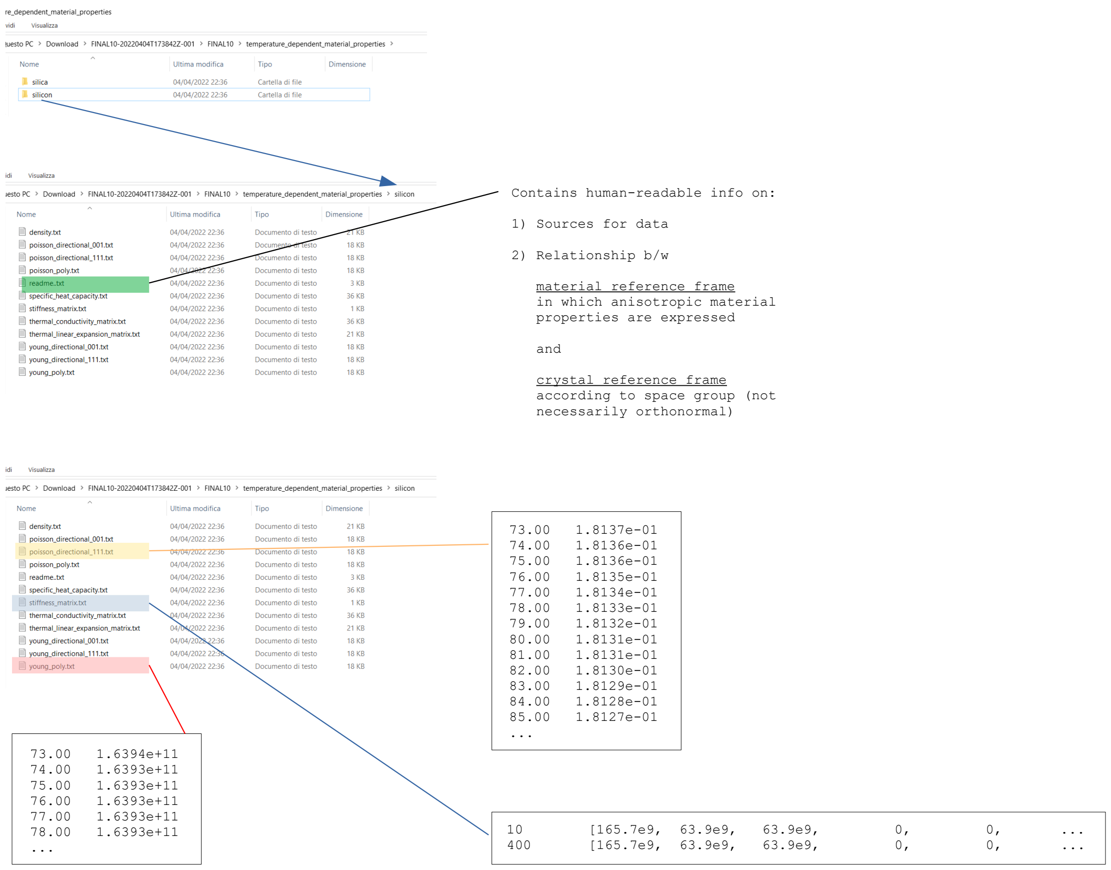
The file structure of the available database is organised in the following way. There is one folder per material; the name of the folder corresponds to the keyword used to identify that material, such as: silica; silicon; etc.

The contents of the folder are a series of text files, that will now be briefly described.

The file *readme.txt* is the only one that is not meant to be uploaded automatically by the program. It contains human-readable information on: a) the sources used for the data (such as the compilation of experimental data collected by Touloukian, etc.); b) the relative orientation between the Material Reference Frame, in which the anisotropic properties are expressed, and the Crystal Reference Frame, defined accordingly to space group (not necessarily orthonormal).

The data files, meant to be uploaded by the program, have the same name as the property they contain. (In the case of a directional property with a negative coordinate, the negative sign "-" may be replaced with the "m" letter in the name of the file, but this is not mandatory; it will be replaced anyway in the variable name by the program, upon uploading). The data files all have the following structure. They are organised in columns and rows; the separator at the end of each row must be the newline character (the ‘Enter’ key); the separator between columns within the same row can be white space or a comma, a semicolon, or a tab. Multiple separators next to each other get grouped together and treated as one.

The values in the first column represent the temperatures (in Kelvin degrees); for each temperature, the subsequent items within the same row give the values of the physical property (in standard SI units). A *scalar* property will only have two relevant columns, one for the temperature and one for the property. If additional columns are present in this case, they do not get uploaded by the program, but are only preserved in the file



```
re_dependent_material_properties
vidi    Visualizza
uesto PC > Download > FINAL10-20220404T173842Z-001 > FINAL10 > temperature_dependent_material_properties >
```

Nome	Ultima modifica	Tipo	Dimensione
silica	04/04/2022 22:36	Cartella di file	
silicon	04/04/2022 22:36	Cartella di file	

```
idi    Visualizza
uesto PC > Download > FINAL10-20220404T173842Z-001 > FINAL10 > temperature_dependent_material_properties > silicon
```

Nome	Ultima modifica	Tipo	Dimensione
density.txt	04/04/2022 22:36	Documento di testo	21 KB
poisson_directional_001.txt	04/04/2022 22:36	Documento di testo	18 KB
poisson_directional_111.txt	04/04/2022 22:36	Documento di testo	18 KB
poisson_poly.txt	04/04/2022 22:36	Documento di testo	18 KB
readme..txt	04/04/2022 22:36	Documento di testo	3 KB
specific_heat_capacity.txt	04/04/2022 22:36	Documento di testo	36 KB
stiffness_matrix.txt	04/04/2022 22:36	Documento di testo	1 KB
thermal_conductivity_matrix.txt	04/04/2022 22:36	Documento di testo	36 KB
thermal_linear_expansion_matrix.txt	04/04/2022 22:36	Documento di testo	21 KB
young_directional_001.txt	04/04/2022 22:36	Documento di testo	18 KB
young_directional_111.txt	04/04/2022 22:36	Documento di testo	18 KB
young_poly.txt	04/04/2022 22:36	Documento di testo	18 KB

```
Contains human-readable info on:

1) Sources for data

2) Relationship b/w

   material reference frame
   in which anisotropic material
   properties are expressed

   and

   crystal reference frame
   according to space group (not
   necessarily orthonormal)
```

```
idi    Visualizza
uesto PC > Download > FINAL10-20220404T173842Z-001 > FINAL10 > temperature_dependent_material_properties > silicon
```

Nome	Ultima modifica	Tipo	Dimensione
density.txt	04/04/2022 22:36	Documento di testo	21 KB
poisson_directional_001.txt	04/04/2022 22:36	Documento di testo	18 KB
poisson_directional_111.txt	04/04/2022 22:36	Documento di testo	18 KB
poisson_poly.txt	04/04/2022 22:36	Documento di testo	18 KB
readme.txt	04/04/2022 22:36	Documento di testo	3 KB
specific_heat_capacity.txt	04/04/2022 22:36	Documento di testo	36 KB
stiffness_matrix.txt	04/04/2022 22:36	Documento di testo	1 KB
thermal_conductivity_matrix.txt	04/04/2022 22:36	Documento di testo	36 KB
thermal_linear_expansion_matrix.txt	04/04/2022 22:36	Documento di testo	21 KB
young_directional_001.txt	04/04/2022 22:36	Documento di testo	18 KB
young_directional_111.txt	04/04/2022 22:36	Documento di testo	18 KB
young_poly.txt	04/04/2022 22:36	Documento di testo	18 KB

```
73.00   1.8137e-01
74.00   1.8136e-01
75.00   1.8136e-01
76.00   1.8135e-01
77.00   1.8134e-01
78.00   1.8133e-01
79.00   1.8132e-01
80.00   1.8131e-01
81.00   1.8131e-01
82.00   1.8130e-01
83.00   1.8129e-01
84.00   1.8128e-01
85.00   1.8127e-01
...
```

```
73.00   1.6394e+11
74.00   1.6393e+11
75.00   1.6393e+11
76.00   1.6393e+11
77.00   1.6393e+11
78.00   1.6393e+11
...
```

```
10        [165.7e9,  63.9e9,   63.9e9,          0,         0,        ...
400       [165.7e9,  63.9e9,   63.9e9,          0,         0,        ...
```

Figure 10: Structure of the text files contained in the material database. The material shown here is identified by the keyword *silicon*. In each data file, the first column contains the temperatures, the other columns contain the values of the physical property. These values can be a scalar or a matrix (given row by row).

for documentation purposes, e.g. for storing values in the original units from the source (if not SI units). A *tensorial* property will have n+1 columns, the first for temperature and the remaining n ones for the n elements of the tensor. For instance for a tensor of order 2, expressible as a 3x3 matrix, the values at each temperature will be represented by 9 elements in this order: the 3 elements from the first row, then the 3 elements from the second row, then the 3 elements from the third row. The use of square brackets and commas as separators is optional. However, if the file for a 3x3 matrix is found to contain only one scalar value, the program assumes that this is a shorthand notation for an isotropic matrix, and automatically converts it into a matrix upon uploading.

The commands to select and upload one material from the database are the following:

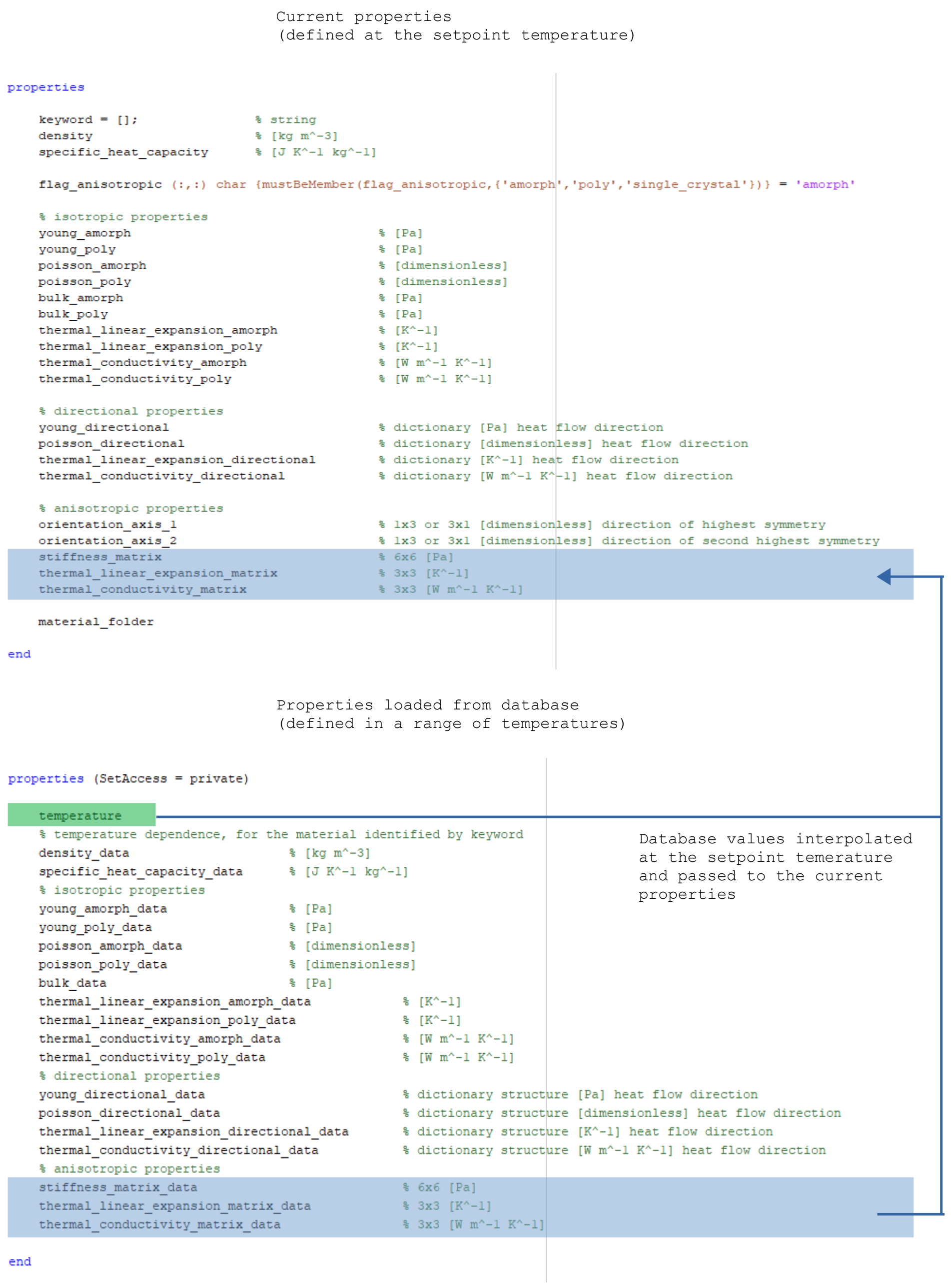
```
mydisc.substrate.material = Material('silica');
mydisc.substrate.material = mydisc.substrate.material.extend_to_zero_temperature();
mydisc.substrate.material = mydisc.substrate.material.set_temperature(300);
```

The first line initialises the *material* object as an attribute of the MatLab variable *mydisc*. The material kewyword ('silica') is passed as an argument to the constructor. The effect of this command is to upload the entire set of data (at all temperatures) into the *material* attribute. For each property, the data get stored as a matrix, in a variable named after the name of the property with the suffix *_data* added at the end (e.g.: *young_amorph_data*). These *_data* variables are classified as "private access" according to MatLab rules, which means that they cannot be directly modified or set by the user, only uploaded from the available database of files.

Regarding the second line, it is an optional command which can be used in the following circumstances. Some of the properties in the database may not be defined for very low temperatures, due to lack of accurate experimental data. As a first approximation, you may wish to extend their value, taken from the lowest available temperature, as a constant down to 0 K.

The third line sets the current temperature. The corresponding values for the material properties will then be taken from the data, interpolated at the requested temperature, and set as the current material properties (in an attribute with the name of the property, without the suffix *_data*). This process is shown in the scheme in Fig. 11.

The constructor in the first line can also be called without any input argument (see next paragraphs). In this case all attributes stay empty. However, whenever you call the constructor and pass it a keyword, as it was done in this section, the object *material* gets re-instantiated from scratch, to avoid leftover data from previous assignments.


Current properties
(defined at the setpoint temperature)
Properties loaded from database
(defined in a range of temperatures)
Database values interpolated
at the setpoint temerature
and passed to the current
properties

```
properties

    keyword = [];               % string
    density                     % [kg m^-3]
    specific_heat_capacity      % [J K^-1 kg^-1]

    flag_anisotropic (:,:) char {mustBeMember(flag_anisotropic,{'amorph','poly','single_crystal'})} = 'amorph'

    % isotropic properties
    young_amorph                                % [Pa]
    young_poly                                  % [Pa]
    poisson_amorph                              % [dimensionless]
    poisson_poly                                % [dimensionless]
    bulk_amorph                                 % [Pa]
    bulk_poly                                   % [Pa]
    thermal_linear_expansion_amorph             % [K^-1]
    thermal_linear_expansion_poly               % [K^-1]
    thermal_conductivity_amorph                 % [W m^-1 K^-1]
    thermal_conductivity_poly                   % [W m^-1 K^-1]

    % directional properties
    young_directional                           % dictionary [Pa] heat flow direction
    poisson_directional                         % dictionary [dimensionless] heat flow direction
    thermal_linear_expansion_directional        % dictionary [K^-1] heat flow direction
    thermal_conductivity_directional            % dictionary [W m^-1 K^-1] heat flow direction

    % anisotropic properties
    orientation_axis_1                          % 1x3 or 3x1 [dimensionless] direction of highest symmetry
    orientation_axis_2                          % 1x3 or 3x1 [dimensionless] direction of second highest symmetry
    stiffness_matrix                            % 6x6 [Pa]
    thermal_linear_expansion_matrix             % 3x3 [K^-1]
    thermal_conductivity_matrix                 % 3x3 [W m^-1 K^-1]

    material_folder

end
```

```
properties (SetAccess = private)

    temperature
    % temperature dependence, for the material identified by keyword
    density_data                    % [kg m^-3]
    specific_heat_capacity_data     % [J K^-1 kg^-1]
    % isotropic properties
    young_amorph_data               % [Pa]
    young_poly_data                 % [Pa]
    poisson_amorph_data             % [dimensionless]
    poisson_poly_data               % [dimensionless]
    bulk_data                       % [Pa]
    thermal_linear_expansion_amorph_data            % [K^-1]
    thermal_linear_expansion_poly_data              % [K^-1]
    thermal_conductivity_amorph_data                % [W m^-1 K^-1]
    thermal_conductivity_poly_data                  % [W m^-1 K^-1]
    % directional properties
    young_directional_data                          % dictionary structure [Pa] heat flow direction
    poisson_directional_data                        % dictionary structure [dimensionless] heat flow direction
    thermal_linear_expansion_directional_data       % dictionary structure [K^-1] heat flow direction
    thermal_conductivity_directional_data           % dictionary structure [W m^-1 K^-1] heat flow direction
    % anisotropic properties
    stiffness_matrix_data                           % 6x6 [Pa]
    thermal_linear_expansion_matrix_data            % 3x3 [K^-1]
    thermal_conductivity_matrix_data                % 3x3 [W m^-1 K^-1]

end
```

Figure 11: The properties in the database (at all tempreatures) are loaded into the "data" attributes (below); the values at the current temperature are then derived and loaded in the current attributes (above), used for the calculation.

Set by hand

To set the material properties by hand rather than upload them from the database, proceed in the following way:

```
mydisc.substrate.material = Material();
mydisc.substrate.material.thermal_conductivity_amorph = 140;
mydisc.substrate.material = mydisc.substrate.material.set_temperature(300);
```

In the first line, to initialise the material, call the constructor method without any arguments.

Then, as shown in the second line, you can set each property individually. Similar lines should be repeated for every property that needs to be set. (The attributes with *_data* suffix will stay empty, since no database has been uploaded; only the current properties are being defined).

Finally, in the third line, the current temperature is set. This won't affect the materal properties that have been set manually, but you need to **set the temperature anyway**, because it may get used explicitly in later calculations, especially in the computation of the thermoelastic effect by Vengallatore's model.

Here is an example with fully anisotropic properties:

```
mydisc.substrate.material = Material();
mydisc.substrate.material.thermal_conductivity = 140;

mydisc.substrate.material = mydisc.substrate.material.set_temperature(300);

mydisc.substrate.material.flag_anisotropic = 'single_crystal';

silicon_stiffness_matrix = [...
[165.7e9,  63.9e9,   63.9e9,        0,         0,        0 ]; ...
[ 63.9e9, 165.7e9,   63.9e9,        0,         0,        0 ]; ...
[ 63.9e9,  63.9e9,  165.7e9,        0,         0,        0 ]; ...
[      0,       0,        0,   79.6e9,         0,        0 ]; ...
[      0,       0,        0,        0,    79.6e9,        0 ]; ...
[      0,       0,        0,        0,         0,   79.6e9 ]];

% from: L. Zhang et al., J. Synchrotron Rad. 21 (2014) 507-517

quartz_stiffness_matrix =  [...
[ 86.6e9,   6.7e9,   12.6e9,  -17.8e9,         0,        0 ]; ...
[  6.7e9,  86.6e9,   12.6e9,   17.8e9,         0,        0 ]; ...
[ 12.6e9,  12.6e9,  106.1e9,        0,         0,        0 ]; ...
[-17.8e9,  17.8e9,        0,   57.8e9,         0,        0 ]; ...
[      0,       0,        0,        0,    57.8e9,  -17.8e9 ]; ...
[      0,       0,        0,        0,   -17.8e9,  39.95e9 ]];

% from: F.M.J. Naus-Thijssen et al., Journal of Structural Geology 32 (2010) 330-341
```

```
mydisc.substrate.material.stiffness_matrix = silicon_stiffness_matrix;

mydisc.substrate.material.orientation_axis_1 = [1 1 0];
```

The assignment lines should be repeated for every property that needs to be set. Notice that the anisotropic flag has been set to *single_crystal*, against the default.

Part III - Other useful functions

There are a number of functions defined within the class *material*, that may be useful to characterise the material at hand, independently of the calculation flow described so far. They are here briefly described. Please refer to Fig. 8 for a visual scheme of how these functions work.

Starting with the functions dealing with linear **elasticity**, it should first be recalled that the stiffness and compliance tensors are defined as tensors of order 4, related to stress (σ^{ij}) and strain (ϵ_{ij}):

$$\sigma^{ij} = C^{ijkl}\,\epsilon_{kl} \qquad \text{(stiffness)}$$

$$\epsilon_{ij} = S_{ijkl}\,\sigma^{kl} \qquad \text{(compliance)}$$

They are therefore expressed by a 3x3x3x3 array of values, of which only 21 are independent (in the most general case, triclinic symmetry).

However, a more compact form is found in Voigt's notation, which exploits the physical symmetry causing the skew-symmetric part of stress and strain to be null, in order to reduce the order of the representation [11, 12]:

$$\begin{bmatrix}\sigma^{11}\\ \sigma^{22}\\ \sigma^{33}\\ \sigma^{23}\\ \sigma^{13}\\ \sigma^{12}\end{bmatrix} = \begin{bmatrix} C^{1111} & C^{2211} & C^{3311} & C^{2311} & C^{3111} & C^{1211}\\ C^{2211} & C^{2222} & C^{3322} & C^{2322} & C^{3122} & C^{1222}\\ C^{3311} & C^{3322} & C^{3333} & C^{2333} & C^{3133} & C^{1233}\\ C^{2311} & C^{2322} & C^{2333} & C^{2323} & C^{3123} & C^{1223}\\ C^{3111} & C^{3122} & C^{3133} & C^{3123} & C^{3131} & C^{1231}\\ C^{1211} & C^{1222} & C^{1233} & C^{1223} & C^{1231} & C^{1212}\end{bmatrix} \begin{bmatrix}\epsilon_{11}\\ \epsilon_{22}\\ \epsilon_{33}\\ 2\epsilon_{23}\\ 2\epsilon_{13}\\ 2\epsilon_{12}\end{bmatrix}$$

This is the form in which the property *stiffness_matrix* is passed by the user to our program, and the one accepted by Ansys [13]. It contains 36 parameters, of which again 21 are independent (as emphasised by the indexing). Note that the index order we use in our program is the one most commonly found in the literature, rather than the unusual one implemented by Ansys [13] (the necessary adjustments will be made automatically, whenever the parameters get passed to Ansys).

The analogous compliance matrix is

$$\begin{bmatrix}\epsilon_{11}\\ \epsilon_{22}\\ \epsilon_{33}\\ 2\epsilon_{23}\\ 2\epsilon_{13}\\ 2\epsilon_{12}\end{bmatrix} = \begin{bmatrix} S_{1111} & S_{2211} & S_{3311} & 2S_{2311} & 2S_{3111} & 2S_{1211}\\ S_{2211} & S_{2222} & S_{3322} & 2S_{2322} & 2S_{3122} & 2S_{1222}\\ S_{3311} & S_{3322} & S_{3333} & 2S_{2333} & 2S_{3133} & 2S_{1233}\\ 2S_{2311} & 2S_{2322} & 2S_{2333} & 4S_{2323} & 4S_{3123} & 4S_{1223}\\ 2S_{3111} & 2S_{3122} & 2S_{3133} & 4S_{3123} & 4S_{3131} & 4S_{1231}\\ 2S_{1211} & 2S_{1222} & 2S_{1233} & 4S_{1223} & 4S_{1231} & 4S_{1212}\end{bmatrix} \begin{bmatrix}\sigma^{11}\\ \sigma^{22}\\ \sigma^{33}\\ \sigma^{23}\\ \sigma^{13}\\ \sigma^{12}\end{bmatrix}$$

and it can also be written in terms of the generalised Young's moduli E, shear moduli G, Poisson-like ratios ν, and other coefficients as (in the most general, triclinic symmetry case) [14, 10]:

$$S = \left[\begin{array}{ccc:ccc}
\frac{1}{E_1} & -\frac{\nu_{12}}{E_2} & -\frac{\nu_{13}}{E_3} & \frac{\eta_{14}}{G_4} & \frac{\eta_{15}}{G_5} & \frac{\eta_{16}}{G_6} \\
-\frac{\nu_{21}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_3} & \frac{\eta_{24}}{G_4} & \frac{\eta_{25}}{G_5} & \frac{\eta_{26}}{G_6} \\
-\frac{\nu_{31}}{E_1} & -\frac{\nu_{32}}{E_2} & \frac{1}{E_3} & \frac{\eta_{34}}{G_4} & \frac{\eta_{35}}{G_5} & \frac{\eta_{36}}{G_6} \\
\hdashline
\frac{\eta_{41}}{E_1} & \frac{\eta_{42}}{E_2} & \frac{\eta_{43}}{E_3} & \frac{1}{G_4} & \frac{\mu_{45}}{G_5} & \frac{\mu_{46}}{G_6} \\
\frac{\eta_{51}}{E_1} & \frac{\eta_{52}}{E_2} & \frac{\eta_{53}}{E_3} & \frac{\mu_{54}}{G_4} & \frac{1}{G_5} & \frac{\mu_{56}}{G_6} \\
\frac{\eta_{61}}{E_1} & \frac{\eta_{62}}{E_2} & \frac{\eta_{63}}{E_3} & \frac{\mu_{64}}{G_4} & \frac{\mu_{65}}{G_5} & \frac{1}{G_6}
\end{array}\right]$$

(The order of the indexes in coefficients ν, μ and η found in the literature may vary, depending on how they are defined with respect to the cause and the effect of the deformation).

In the original tensors of order 4, since the basis is orthonormal, the distinction between contravariant and covariant coordinates is superfluous. However, it has been shown [11] that the Voigt contraction equates to a representation in a tensorial basis that is othogonal but not normalised. The distinction then becomes significant, and multiplication factors of 2 and 4 arise when switching from the contra- to the co-variant form, since the metric tensor does not correspond to identity. This is seen in the expressions above for compliance S and strain ϵ. Importantly, the distinction also affects how the components transform under rotation [15, 16].

Taking all this into account, the program offers a series of methods to convert the elastic properties from one notation to the other, and to rotate them in a way consistent with the notation in which they are given.

The commands to switch notation are the following. These functions are static methods, i.e. they are associated with a class, but not with specific instances of that class. In other words, they do not require an object of a resonator type (such as *mydisc*) or of a material type to be created, as they can work as stand-alone functions.

```
compliance_voigt = compliance_tensor_to_voigt(compliance_tensor);
stiffness_voigt = stiffness_tensor_to_voigt(stiffness_tensor);
compliance_tensor = compliance_voigt_to_tensor(compliance_voigt);
stiffness_tensor = stiffness_voigt_to_tensor(stiffness_voigt);
```

where the properties with suffix `_tensor` are in the proper tensor notation, and those with suffix `_voigt` are in the compact notation.

To go from stiffness to compliance and vice versa in Voigt's notation, one can simply invert the matrix [11]:

```
stiffness_voigt = inv(compliance_voigt);
compliance_voigt = inv(stiffness_voigt);
```

The rest of the functions presented in the following require an instance of class *material* to be crated. For the sake of brevity, let us assign:

```
  mymaterial = mydisc.substrate.material;
```

The function to rotate the properties (in Voigt's notation) is:

```
stiffness_voigt_rotated = mymaterial.rotate_stiffness_matrix(eulerZ,eulerY,eulerX);
```

where the angles (given in degrees) may be taken to refer to a series of intrinsic rotations about the axes Z, Y' (= Y after 1st rotation) and X" (= X after 1st and 2nd rotation), in this order. A more exhaustive explanation of all the conventions used for rotations is found in the comments at the end of file *material.m*.

The function to find the Young's modulus along an arbitrary direction (in the Material Reference Frame, MRF) is

```
direction_vector = [1 1 2];
young_directional = mymaterial.young_directional_from_stiffness_matrix(direction_vector);
```

where [1 1 2] are coordinates in the MRF, and the direction vector does not need to be normalised. The correctedness of this function has been tested against the results found in [17] for the case of a silicon crystal.

To conclude with the elasticity utilities, there are a few functions extracting the isotropic properties (cf. Fig. 8)

```
young_poly = mymaterial.young_poly_from_stiffness_matrix();
poisson_poly = mymaterial.poisson_poly_from_stiffness_matrix();
bulk_poly = mymaterial.bulk_poly_from_stiffness_matrix();
bulk_single_crystal = mymaterial.bulk_single_crystal_from_stiffness_matrix();
bulk_amorph = mymaterial.bulk_amorph_from_young_amorph_and_poisson_amorph();
bulk_poly = mymaterial.bulk_poly_from_young_poly_and_poisson_poly();
```

where the models in [18, 19, 20] have been employed.

Turning now to the properties related to the **thermal conductivity** ("k") and the **linear thermal expansion** ("alpha"), they are tensors of order 2. The functions to rotate them and extract their value along an arbitrary direction are

```
k_matrix_rotated = mymaterial.rotate_k_matrix(eulerZ,eulerY,eulerX);
k_directional = mymaterial.k_directional_from_k_matrix(direction_vector);
```

```
alpha_matrix_rotated = mymaterial.rotate_alpha_matrix(eulerZ,eulerY,eulerX);
alpha_directional = mymaterial.alpha_directional_from_alpha_matrix(direction_vector);
```

with the same conventions and definitions used for the elasticity.

Finally, the isotropic α for a polycrystalline material can be derived as

```
alpha_poly = mymaterial.alpha_poly_from_alpha_matrix_and_stiffness_matrix();
```

following the model in [19].

3.4.4 Mesh settings

In the context of finite element analysis with Ansys, the Element Type chosen for the substrate is "SOLID186", a 3D 20-node structural solid element, suitable for stress-strain analysis [13].

In the case of a thin layer such as the coating, however, as will be seen in Section 3.4.8, the object is modeled as a "3D surface body", and the Element Type chosen is "SHELL281", an 8-node structural shell type. As stated in [21, 22]: when a real world body is thin and subject to bending, it is generally a good candidate for a 3D surface body. There are advantages to using these models: 1) Creating surface models is usually easier than creating solid models; 2) The problem becomes simpler to solve for Ansys compared to using solid models (due to the efficiency of shell elements). This results in a much faster, and a more accurate computation. Ansys will mesh surface bodies with shell elements. They are designed to decouple the deformation on the surface and the deformation in the normal direction, allowing for a simple and efficient simulation of a thin structure. Specifically, SHELL281 is well-suited for linear, large rotation, and/or large strain nonlinear applications.

In our program, the user can define the parameters according to which an automated mesh of the sample will be performed. The mesh method can be set to 'sweep' or 'non-sweep' (see Fig. 12). When the goal of the calculations is to determine the thermoelastic noise, it is recommended that the sweep method be selected. This will create a series of equally spaced divisions along the Disc thickness, so that the expansion or contraction in the volume of each element, as a function of its position along the thickness of the disc, can be captured. This is the physical mechanism at the core of the thermoelastic effect.

The parameter *fineness* defines the dimensions of each element in the mesh. For instance,

```
mydisc.mesh_settings.fineness = 1e-3;
```

requires a mesh to be 1 mm in element size, whereas

```
mydisc.mesh_settings.fineness = mydisc.substrate.diameter/20;
```

requires that there are at least 20 elements along the diameter of the Disc. To let Ansys autonomously decide the element size, simply set the parameter *fineness* to 0 (which is the default).

The parameter *divisions* specifies, in the case of a sweep mesh, how many layers along the thickness must be created. Obviously, the higher the number of divisions, the more precise the computation of the thermoelastic effect will be, at the expense of slowing down the calculation. Generally, a number of at least

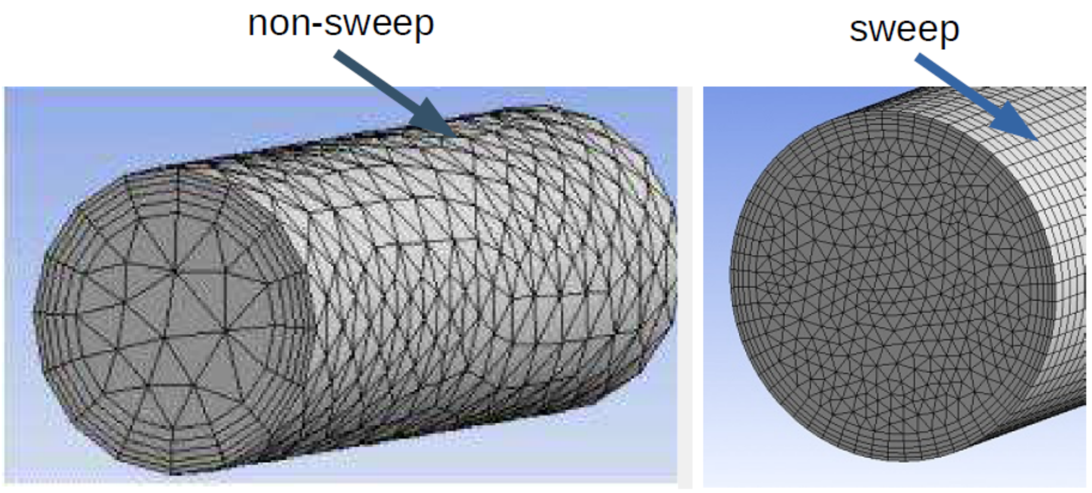


Figure 12: The two main approaches to meshing made available by the program, shown on a cylindrical object. In the sweep method, the two-dimensional base surface is meshed first, then the pattern is repeated across the length of the object in a series of equally-spaced layers, thus creating the three-dimensional mesh elements. The sweep method is particularly suitable for computations of thermoelastic dissipation.

5 or 6 divisions is recommended. If the selected mesh method is set to non-sweep, the value assigned to this parameter is meaningless and will be ignored.

3.4.5 Calculation settings

With the object *calculation_settings*, the user can specify the mechanical modes and the range of frequencies that Ansys will employ. A typical input can look like this:

```
mydisc.calculation_settings.max_number_of_modes = 10;
mydisc.calculation_settings.max_frequency = 5000;
mydisc.calculation_settings.min_frequency = 0;
mydisc.calculation_settings.mode_list = [2,5,6];
```

The first three lines refer to which modes will be calculated by Ansys. The fourth line (the mode list) refer to which subset of those modes will be saved by the program in a series of dedicated folders.

The modes in the "mode list" are numbered according to Ansys: the numbers (starting from 1 onwards) refer to the modes effectively calculated within the parameters of the current specific calculation, and they are ordered in increasing order of resonant frequency.

To verify what type each mode is (butterfly; drum; mixed; breathing; ...) it can be useful to run manually Ansys once, via the GUI (the files stored in folder *support_files* can be used to speed up the set-up and ensure consistency with the automated runs). After that, the fully automated program can be used for repeated calls. The goal of this preliminary run is to create the mode list, which contains the numbers of all modes of interest to the user. This information will then be passed to the program as seen above, and will be used to determine which mode results need to be saved. A subfolder for each mode will be created within *output_results*. For instance, if the mode list indicates that modes 2, 5 and 6 are of interest, then three subfolders named ‘2’, ‘5’ and ‘6’ will be created. The mode list will also be used to upload the results from the text files stored in these folders, to the MatLab variable *mydisc*, and to upload the detailed element-by-element results into MatLab, if requested by the user. Note that when the mode folders are created, any other content already present in *output_files* from previous runs is **deleted**, to avoid contamination with leftover data from previous calculations, which may have been made with different inputs and settings.

The way to affect which modes are calculated by Ansys, and thus to speed up the computation, is to set the following parameters:

- The parameters max_frequency and min_frequency set limits on the range of resonant frequencies of the modes. Any mode whose resonant frequency falls outside of this range will not be calculated. The mode numbers assigned by Ansys and used in the *mode_list* are dependent on the frequency range: for instance, mode No. 1 is the first mode that falls within the acceptable frequency range. By default, the maximum frequency is left undefined, whereas the minimum frequency is set at 100 Hz, to avoid including the low-frequency rigid body motions.

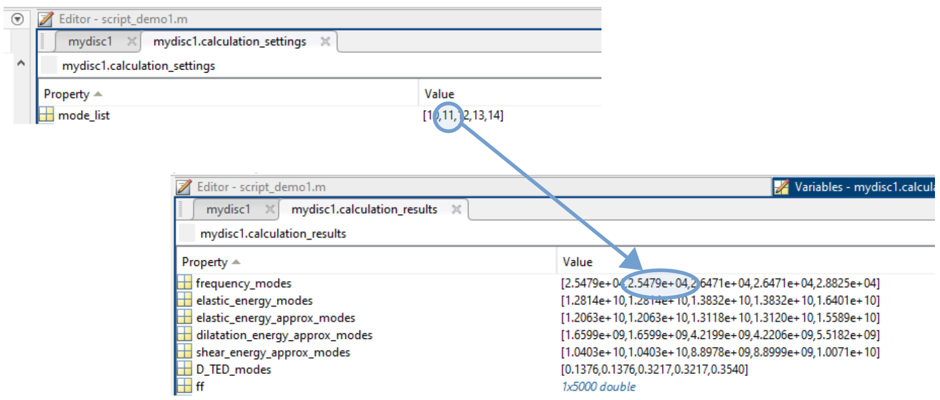
- Similarly, the parameter max_number_of_modes cuts off the calculation once the mode with the designated number has been reached.

The calculation by Ansys gets stopped once the limitation imposed by either max_number_of_modes or by max_frequency is hit (whichever limitation is encountered first).

Note that while the numbers in mode_list always correspond to Ansys’s convention, once the modes are loaded into the MatLab object they become a simple vector, indexable as a sequence.

For instance in the Figure below, the mode numbered as 11th by Ansys (given the settings of this particular calculation) has been requested as the second mode of interest, within the mode_list of *calculation_settings*. Therefore, all results pertaining to this mode occupy the second position in the vectors of *calculation_results*: for example, its frequency is the second value stored in frequency_modes, as shown.

Finally, note that in *calculation_settings* one can also find useful pointers to the principal folders used by the program, in which the computation results, the MatLab variable, and the auxiliary files to be fed to Ansys are stored.


Editor - script_demo1.m
mydisc1
mydisc1.calculation_settings
mydisc1.calculation_settings

Property	Value
mode_list	[10,11,12,13,14]

Editor - script_demo1.m
Variables - mydisc1.calcul
mydisc1
mydisc1.calculation_results
mydisc1.calculation_results

Property	Value
frequency_modes	[2.5479e+04,2.5479e+04,2.6471e+04,2.6471e+04,2.8825e+04]
elastic_energy_modes	[1.2814e+10,1.2814e+10,1.3832e+10,1.3832e+10,1.6401e+10]
elastic_energy_approx_modes	[1.2063e+10,1.2063e+10,1.3118e+10,1.3120e+10,1.5589e+10]
dilatation_energy_approx_modes	[1.6599e+09,1.6599e+09,4.2199e+09,4.2206e+09,5.5182e+09]
shear_energy_approx_modes	[1.0403e+10,1.0403e+10,8.8978e+09,8.8999e+09,1.0071e+10]
D_TED_modes	[0.1376,0.1376,0.3217,0.3217,0.3540]
ff	1x5000 double

3.4.6 Calculating the vibration mechanical modes with Ansys

The first stage of the calculation calls the Ansys software to compute, for each mode, the resonant frequency, the stress and strain fields, and the elastic energy.

This calculation is performed by calling the following two commands:

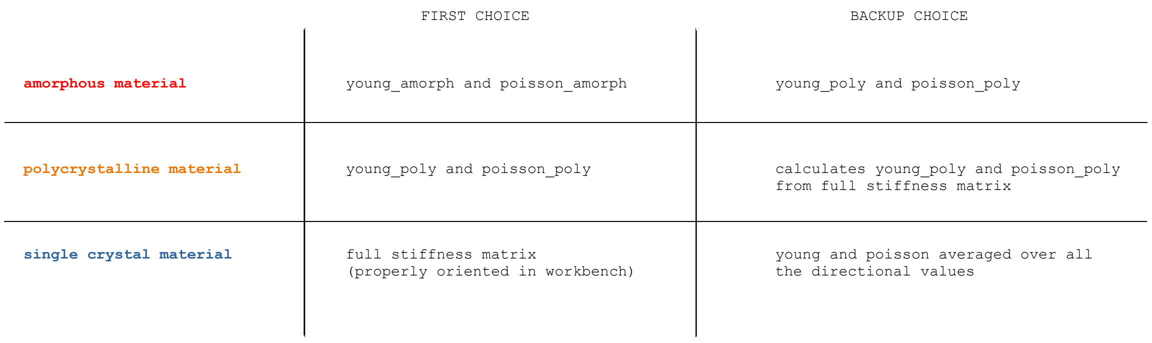
```
workbench_out = mydisc.ansys_workbench_calculate();
mydisc = mydisc.ansys_apdl_calculate();
```

of which, the first one prepares the input file to be given to Ansys, and the second one runs the Ansys MAPDL processor to execute the calculation.

You can find the file as *input_file_edited.dat* in subfolder *input_files* of the main folder assigned to your variable.

The data and parameters defined in the previous sections get passed to Ansys in order to perform the computation. In particular, as far as the material properties are concerned, the relevant ones are 1) the density (a scalar) and 2) the elasticity (a tensorial property in the most general case). Regarding the elasticity, Ansys can work with the full stiffness matrix or, in the isotropic cases, with the two parameteres: Young's modulus and Poisson's ratio.

As anticipated in section "Material Properties" (Part I), the program will try to find the best parameters to pass to Ansys, based on the state of the material (amorphous, polycrystalline, single crystal) and the available data. Below is a complete chart of how this assignment is done. For instance, if the material is designated as single crystal, the program first looks for the full stiffness matrix within the MatLab variable *mydisc*, specifically in the attribute *mydisc.substrate.material.stiffness_matrix*. If it can find it (the attribute is not empty), then the matrix gets rotated to the proper orientation and passed to Ansys. If, instead, the attribute is empty, it looks for the directional values of the elastic parameters along specific crystallographic axes, and averages them.



	FIRST CHOICE	BACKUP CHOICE
amorphous material	young_amorph and poisson_amorph	young_poly and poisson_poly
polycrystalline material	young_poly and poisson_poly	calculates young_poly and poisson_poly from full stiffness matrix
single crystal material	full stiffness matrix (properly oriented in workbench)	young and poisson averaged over all the directional values

Figure 13: Chart of the choices performed by the program to pass the most suitable material parameters to Ansys. The backup choice is made if the properties specified as first choice are found to be empty.

The calculation will be performed in two steps, according to the two commands given. The first step (called by *ansys_workbench_calculate*) will briefly open the Workbench GUI in order to create the Ansys input file; however, no interaction on the part of the user is required. The starting and finishing times for each step will be displayed in the MatLab window. A successful calculation will return the value 0, which will also be displayed onscreen in the MatLab window.

The overall results for each mode (the sum of all the element-by-element values) are automatically loaded into *calculation_results* and become part of the MatLab variable *mydisc*. After the calculation is completed, the attribute *mydisc.calculation_results*, which was previously storing empty values, should be partially populated with the results obtained so far. The fields that remain empty pertain to the physics of thermal dissipation and will be populated at a subsequent time, after the thermoelastic calculations.

On the other hand, the detailed results (the element-by-element values that are too numerous to be conveniently uploaded into the MatLab variable) are saved as text files in the folders dedicated to each mode. They can still be explicitly retrieved and imported in MatLab with the "get" methods, as will be shown at the end of this section.

Let us now inspect the values loaded into *calculation_results* more closely. They should look similar to these (calculated with five modes in the mode list):

```
              frequency_modes:   [239.5960 239.6134 360.6332 550.0708 550.1143]
          elastic_energy_modes:   [1.1332e+06 1.1333e+06 2.5672e+06 5.9726e+06 5.9736e+06]
   elastic_energy_approx_modes:   [1.0061e+06 1.0062e+06 2.2799e+06 5.3014e+06 5.3017e+06]
dilatation_energy_approx_modes:   [7.0072e+04 7.0089e+04 1.0589e+06 5.8408e+05 5.8402e+05]
     shear_energy_approx_modes:   [9.3606e+05 9.3614e+05 1.2210e+06 4.7173e+06 4.7177e+06]
                    D_TED_modes:   [0.0696 0.0697 0.4645 0.1102 0.1102]
                             ff:   [ ]
      ff_phi_TED_without_D_TED:   [ ]
                  phi_TED_modes:   [ ]
  phi_TED_without_D_TED_modes:   [ ]
                 phi_MEAS_modes:   [ ]
           Delta_phi_MEAS_modes:   [ ]
```

The first six attributes have been populated. By examining each attribute in turn, we find:

1. The resonant modal frequencies as calculated by Ansys, for each mode. They are stored in one vector, of the same length as the number of modes being saved. They are ordered in order of increasing frequency.

2. The elastic energy for each mode. The results are stored in one vector, of the same length as the number of modes being saved. The modes are ordered in the same way as 1. The values correspond to the variable "sene" from Ansys (which, in this context, represents the elastic energy). For each mode, the value is calculated simply by summing all the element-by-element values.

3-4-5. The (approximated) values of the elastic, dilatation and shear energy, respectively. As already stated in Section 2, the elastic energy is in fact the sum of two contributions:

$$E^{elastic} = E^{dil} + E^{shear}$$

which can be derived from the stress (σ) and strain (ϵ) tensors.

It should be remarked that the off-diagonal "strains" outputted by Ansys, although denoted by ϵ, are in fact defined as the engineering shear strains, albeit named against the usual convention. From the manual [13]: "*The shear strains (ϵ_{xy}, ϵ_{yz} and ϵ_{xz}) are the engineering shear strains, which are twice the tensor shear strains. The ϵ notation is commonly used for the tensor shear strains, but is used here as engineering* *shear strains for simplicity of output*" (p. 7). Recall that the the shear tensor strains are defined from the displacements u and positions x as

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad \text{with } i \neq j$$

whereas the engineering shear tensor strains are

$$\gamma_{ij} = \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} = \epsilon_{ij} + \epsilon_{ji} = 2\epsilon_{ij} \qquad \text{with } i \neq j$$

and the on-diagonal strains are always defined without ambiguity as

$$\epsilon_{ii} = \frac{\partial u_i}{\partial x_i}$$

The energies returned by our program for these entries are calculated from the "stress", "strain" and "volume" fields provided by Ansys, for each mode. The results are stored in one vector, of the same length as the number of modes being saved. The modes are ordered in the same way as 1. For each mode, the value is calculated simply by summing all the element-by-element values.

Note that these numerical values will be less accurate than the ones obtained from 2. The reason is that the "stress" and "strain" returned by Ansys are merely the values at the centroid of each element, whereas the "sene" values are properly integrated over the element volume. This is why the elastic energy calculated with this approach has been called the "approximated" elastic energy. Still, this approach is necessary because the "sene" values provided by Ansys do not allow one to separate the dilatation and shear contribution. By deriving the energies anew from stress and strain, we are able to do so. The sum of the dilatation and shear energy gives the total elastic energy (approximated), which is supposed to coincide with the more exact value obtained from 2. The finer the mesh, the more the approximated value should converge to the "sene" one.

One can therefore use the two values for the total elastic energy, derived from points 2 and 3, to judge the quality of the mesh. If the mesh is fine enough, elastic_energy_approx should converge to elastic_energy, and in that case, one can use the dilatation and shear values as reasonable estimates. This means that the dilution factor found in 6 is also reliable.

6. The dilatation and shear contributions to the elastic energy, from point 5, are used to calculate the dilution factors, according to:

$$D^{TE} = \frac{E_s^{dil}}{E_s^{dil} + E_s^{shear}}$$

or, in terms of the results found in 5:

```
D_TED_modes =  dilatation_energy_approx_modes./elastic_energy_approx_modes;
```

The results are stored in one vector, of the same length as the number of modes being saved. The modes are ordered in the same way as 1.

The detailed element-by-element results are not uploaded into the MatLab variable *mydisc*, but they are kept stored as text files within the subfolder of that mode, and can be imported into a MatLab variable in the following way:

```
mode_number = 2;   % number as given by Ansys

element_volumes = mydisc.get_element_volume(mode_number);
element_volume_changes = mydisc.get_element_volumechange(mode_number);

[strainXX, strainYY, strainZZ, strainXY, strainYZ, strainXZ] = ...
mydisc.get_element_strain(mode_number);
[stressXX, stressYY, stressZZ, stressXY, stressYZ, stressXZ] = ...
mydisc.get_element_stress(mode_number);

elastic_energy = mydisc.get_element_elastic_energy(mode_number);
[elastic_energy_approx,dilatation_energy_approx,shear_energy_approx] = ...
mydisc.get_element_energy_approx(mode_number);
```

3.4.7 Calculating the thermoelastic dissipation

The second stage of the calculations involves finding the thermoelastic dissipation, from Vengallatore's or similar analytical models [1, 4, 2] .

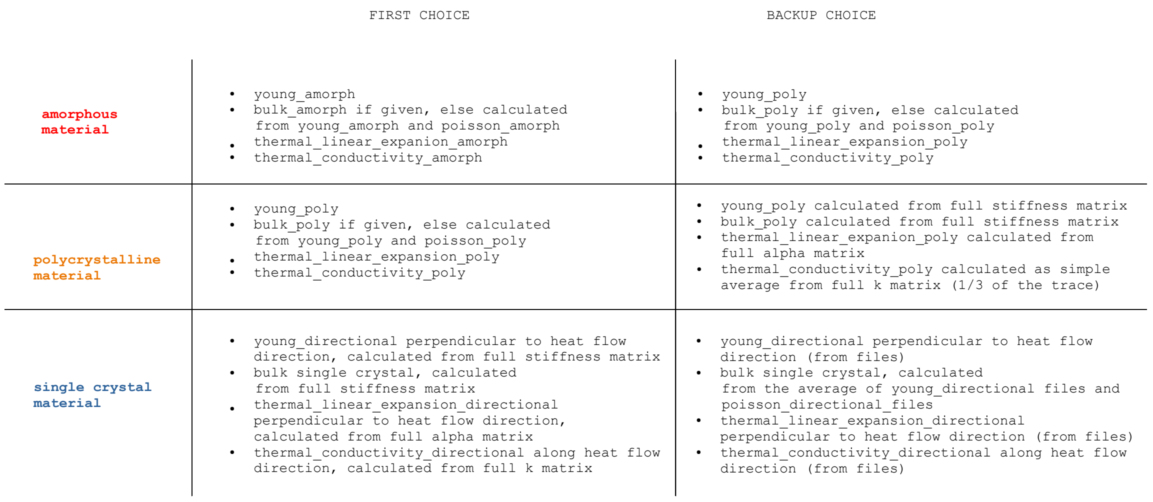
Specifically, Vengallatore's model is used for computations on resonators of type Disc and DoublyCoated-Disc, since it is more accurate and extendable to the case of a layered structure, which makes it particularly useful in comparing the results of coated and uncoated samples and in finding the thermoelastic shift between the two [1, 2]. For a resonator of type CantileverFiber, however, the model is not suitable, since the geometrical approximation of thin structure is not applicable. Therefore, a version of Zener's simpler model is employed, suitable for a cylindrical object whose length is much greater than its diameter [3]. Finally, no method for thermoelastic calculation is associated with a sample of type CurvedDisc, since the curvature does not fit rigorously into either model, and this resonator type is mostly meant for comparison of the resonant frequencies between curved and uncurved samples.

The command to do the calculation is (in the case of a type Disc):

```
mydisc = mydisc.calculate_TED_vengallatore(2);

mydisc = mydisc.calculate_dissipation_TED();
```

The first command performs the calculation of the thermoelastic dissipation. Its argument, which is optional, represents the number of terms in Vengallatore's series that will be used. That is, if the argument is 2, the series will be truncated after the first two terms. If no argument is given, by default, only the first term of the series will be calulated.

As was the case in the first stage of the calculations done with Ansys, there are a number of parameters being passed to the function doing the computation, whose purpose is to describe the physical properties of the material. Again, the program tries to use the optimal set of parameters, given the state of the material specified by the keyword (amorphous, polycrystalline, single crystal). If the optimal information is not available from the data, then a second-best choice is made. The complete chart of the physical parameters and how they are selected is in Fig. 14.



	FIRST CHOICE	BACKUP CHOICE
amorphous material	• young_amorph • bulk_amorph if given, else calculated from young_amorph and poisson_amorph • thermal_linear_expanion_amorph • thermal_conductivity_amorph	• young_poly • bulk_poly if given, else calculated from young_poly and poisson_poly • thermal_linear_expansion_poly • thermal_conductivity_poly
polycrystalline material	• young_poly • bulk_poly if given, else calculated from young_poly and poisson_poly • thermal_linear_expansion_poly • thermal_conductivity_poly	• young_poly calculated from full stiffness matrix • bulk_poly calculated from full stiffness matrix • thermal_linear_expanion_poly calculated from full alpha matrix • thermal_conductivity_poly calculated as simple average from full k matrix (1/3 of the trace)
single crystal material	• young_directional perpendicular to heat flow direction, calculated from full stiffness matrix • bulk single crystal, calculated from full stiffness matrix • thermal_linear_expansion_directional perpendicular to heat flow direction, calculated from full alpha matrix • thermal_conductivity_directional along heat flow direction, calculated from full k matrix	• young_directional perpendicular to heat flow direction (from files) • bulk single crystal, calculated from the average of young_directional files and poisson_directional_files • thermal_linear_expansion_directional perpendicular to heat flow direction (from files) • thermal_conductivity_directional along heat flow direction (from files)

Figure 14: Chart of the choices performed by the program to select the most suitable material parameters for the calculation of the thermoelastic dissipation. The backup choice is made if the properties specified as first choice are found to be empty.

Turning now to a closer inspection of the results, one can see that the remaining fields of the *calculation_results* have been populated.

```
             frequency_modes:   [239.5960 239.6134 360.6332 550.0708 550.1143]
          elastic_energy_modes:   [1.1332e+06 1.1333e+06 2.5672e+06 5.9726e+06 5.9736e+06]
      elastic_energy_approx_modes:   [1.0061e+06 1.0062e+06 2.2799e+06 5.3014e+06 5.3017e+06]
   dilatation_energy_approx_modes:   [7.0072e+04 7.0089e+04 1.0589e+06 5.8408e+05 5.8402e+05]
        shear_energy_approx_modes:   [9.3606e+05 9.3614e+05 1.2210e+06 4.7173e+06 4.7177e+06]
                    D_TED_modes:   [0.0696 0.0697 0.4645 0.1102 0.1102]
                             ff:   [1×5000 double]
         ff_phi_TED_without_D_TED:   [1×5000 double]
                  phi_TED_modes:   [3.2116e-07 3.2120e-07 1.6355e-06 2.7270e-07 2.7264e-07]
      phi_TED_without_D_TED_modes:   [4.6114e-06 4.6113e-06 3.5213e-06 2.4752e-06 2.4750e-06]
                 phi_MEAS_modes:   [3.2116e-07 3.2120e-07 1.6355e-06 2.7270e-07 2.7264e-07]
           Delta_phi_MEAS_modes:   [3.2116e-09 3.2120e-09 1.6355e-08 2.7270e-09 2.7264e-09]
```

Recall that, for the case of an **uncoated sample**, as seen in Eq. 4, the measured loss angle is:

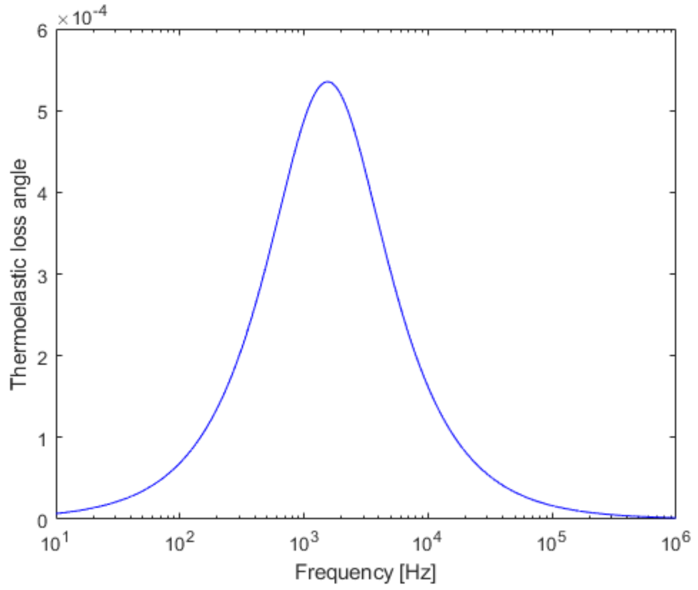
$$\phi_{meas}(0) \; = \; \phi_s^{intr} + D^{TE}(0)\phi_s^{TE}(0) \qquad \text{for } t = 0 \text{ (no coating)} \tag{6}$$

where ϕ_s^{intr} has been considered negligible, $D^{TE}(0)$ is the mode-dependent dilution factor (called `D_TED` by the program, see point 6 of the results populated in the previous section), and $\phi_s^{TE}(0)$ is the thermoelastic loss in the substrate, which does not depend directly on the mode, but does depend on its frequency (called `phi_TED_without_D_TED` by the program, see points 8 and 10 below). The product $D^{TE}(0)\phi_s^{TE}(0)$ corresponds to the actual measured loss, comprehensive of the dilution factor (called `phi_TED` by the program, see points 9 and 11 below).

Here is a description of the meaning of each field which has been newly populated, in the results shown above. For each field (except No. 7 and 8), the results are stored in one vector, of the same length as the number of modes being saved. The modes are ordered by increasing resonant frequency (stored in point 1).

7 and 8. An array of frequencies and the corresponding values of $\phi_s^{TE}(0)$ (without multiplying by $D^{TE}(0)$), meant to be plotted together with a command such as:

```
figure();
semilogx(mydisc.calculation_results.ff,...
    mydisc.calculation_results.ff_phi_TED_without_D_TED,'-b');
xlabel('Frequency [Hz]');
ylabel('Thermoelastic loss angle');
```



These values lack the mode-dependent information given by the diluition factors, but they allow one to plot the general frequency dependence of the thermoelastic noise, so that it may be judged where the resonant frequencies of different modes are located with respect to the peak of the thermoelastic noise.

9. `phi_TED_modes` corresponds to $D^{TE}(0)\phi_s^{TE}(0)$.

The thermoelastic loss for each mode, comprehensive of the mode-dependent dilution factor.

Recall from Section 2.2 that our calculation involves dividing the results from Vengallatore's model by the D_TED factors that are implicit in that model, and substituting them with the ones previously found with Ansys, and stored in field 6 of *calculation_results*.

10. `phi_TED_without_D_TED_modes` corresponds to $\phi_s^{TE}(0)$.

It is the same results of line 9, but divided by $D^{TE}(0)$, for each mode.

11. phi_MEAS_modes corresponds to $\phi_{meas}(0)$. In the simple case of an uncoated resonator, the results in lines 9 and 11 should coincide. For a coated resonator, this is no longer the case, as will be detailed briefly in Section 3.4.8.

12. Delta_phi_MEAS_modes corresponds to phi_MEAS_modes/100*percentage_uncertainty_meas. It is simply an estimation of the uncertainty of the measured values, for each mode. The uncertainty on each single measurement, percentage_uncertainty_meas, is an attribute of the MatLab variable *mydisc*, set by default at 1. It can be changed by the user, depending on their specific experimental setup.

3.4.8 Extension to the case of other resonator types

The procedure described so far, in the tutorial with detailed instructions, can be applied essentially unaltered also to the case of a **CantileverFiber**. The only difference will be that the calculation of the thermoelastic dissipation will assume the heat flow direction to be perpendicular to the cylindrical axis, and the geometry of the sample will be approximated as being quasi-1-dimensional. These adjustments are transparent to the user and do not require any correction on their part, except for the fact that, to emphasise these peculiarities in the theoretical model, the function performing the thermoelastic calculations has been named differently:

```
myfiber = myfiber.calculate_TED_zener();
myfiber = myfiber.calculate_dissipation_TED();
```

In the case of a **CurvedDisc**, the only difference in the set-up part is that the user also has to specify the curvature radius, as shown in the tutorial 3.3.3. Regarding the calculation part, however, the function to calculate the thermoelastic dissipation has not been implemented, since this case does not fit squarely in any of the theoretical models described at the beginning. The main usefulness of this type of resonator lies in the calculation of the resonant frequencies, and in their comparison with those of an equivalent non-curved disc.

In the case of a **DoublyCoatedDisc**, the differences are more substantial and we shall briefly highlight them.

In the setup phase of the procedure, the user must define the geometrical properties and the material properties for both the substrate and the coating, separately. In examining the calculation results, after the first stage of the computation (done with Ansys), the fields are again partially populated. They present as follows:

```
                 element_number_substrate:   726
                 element_number_coating1:   242
                 element_number_coating2:   242
         elastic_energy_substrate_modes:   [7.0792e+06 1.0433e+07 1.9773e+07 4.5662e+07 4.5665e+07]
          elastic_energy_coating1_modes:   [6.3777e+04 6.3734e+04 1.4495e+05 3.3971e+05 3.3973e+05]
          elastic_energy_coating2_modes:   [6.3777e+04 6.3734e+04 1.4495e+05 3.3971e+05 3.3973e+05]
  elastic_energy_approx_substrate_modes:   [6.2720e+06 9.2469e+06 1.7538e+07 4.0043e+07 4.0045e+07]
   elastic_energy_approx_coating1_modes:   [6.4809e+04 6.4620e+04 1.4613e+05 3.4153e+05 3.4161e+05]
   elastic_energy_approx_coating2_modes:   [6.5360e+04 6.5118e+04 1.4616e+05 3.4641e+05 3.4645e+05]
dilatation_energy_approx_substrate_modes:   [3.0210e+05 5.1237e+05 6.2120e+06 3.5447e+06 3.5454e+06]
 dilatation_energy_approx_coating1_modes:   [3.2691e+03 5.5450e+03 6.7239e+04 3.8366e+04 3.8374e+04]
 dilatation_energy_approx_coating2_modes:   [3.2691e+03 5.5450e+03 6.7239e+04 3.8366e+04 3.8374e+04]
     shear_energy_approx_substrate_modes:   [5.9700e+06 8.7345e+06 1.1326e+07 3.6498e+07 3.6499e+07]
      shear_energy_approx_coating1_modes:   [6.1540e+04 5.9075e+04 7.8890e+04 3.0316e+05 3.0324e+05]
      shear_energy_approx_coating2_modes:   [6.2091e+04 5.9573e+04 7.8917e+04 3.0804e+05 3.0807e+05]
                               D_c_modes:   [0.0177 0.0121 0.0144 0.0147 0.0147]
                         frequency_modes:   [604.2330 731.4254 1.0082e+03 1.5322e+03 1.5323e+03]
                    elastic_energy_modes:   [7.2067e+06 1.0560e+07 2.0063e+07 4.6341e+07 4.6345e+07]
             elastic_energy_approx_modes:   [6.4022e+06 9.3766e+06 1.7830e+07 4.0731e+07 4.0733e+07]
          dilatation_energy_approx_modes:   [3.0863e+05 5.2346e+05 6.3464e+06 3.6214e+06 3.6222e+06]
               shear_energy_approx_modes:   [6.0936e+06 8.8532e+06 1.1483e+07 3.7109e+07 3.7111e+07]
                             D_TED_modes:   [0.0482 0.0554 0.3542 0.0885 0.0885]
                                      ff:   [ ]
                 ff_phi_TED_without_D_TED:   [ ]
                           phi_TED_modes:   [ ]
             phi_TED_without_D_TED_modes:   [ ]
                          phi_MEAS_modes:   [ ]
                    Delta_phi_MEAS_modes:   [ ]
```

As can be seen, most of the fields are divided into contributions from the substrate and from the coatings. Recall that, for the case of a **coated sample**, as seen in Eq. 4, the measured loss angle is:

$$\phi_{meas}(t) \;=\; \left[1 - D_c(t)\right] \left[\phi_s^{intr} + D^{TE}(t)\phi_s^{TE}(t)\right] + D_c(t)\phi_c^{intr} \quad \text{for } t \neq 0 \text{ (coating)} \tag{7}$$

where t is the thickness of the coating (supposed to be equal on both sides), ϕ_s^{intr} is again considered negligible, $D^{TE}(t)$ is the mode-dependent thermoelastic dilution factor (called `D_TED` by the program), and $\phi_s^{TE}(t)$ is the thermoelastic loss in the substrate, which does not depend directly on the mode, but does depend on its frequency (called `phi_TED_without_D_TED` by the program). The product $D^{TE}(t)\phi_s^{TE}(t)$ (called `phi_TED` by the program) is the thermoelastic loss, comprehensive of the dilution factor.

$D_c(t)$ is a different dilution factor, related to energy in the two coatings vs in the substrate. ϕ_c^{intr} is the dissipation in the coating, and is not derived from physical principles by the program, but simply estimated at an order of magnitude of 10^{-4}. It can be changed by the user, by setting the property `phi_coating` of the main MatLab variable *mydisc2*. This is to allow one to judge how large the contribution from the coatings must be, to be measured with an acceptable uncertainty. Finally, $\phi_{meas}(t)$ corresponds to the actual measured loss (called `phi_MEAS` by the program), including all contributions mentioned so far.

The meaning of the fields in the *calculation_results* is as follows:

1 to 3. The first three fields refer to the number of elements into which the substrate and the coatings have been divided, to perform the Finite Element Analysis. In the case of the coatings, Ansys is instructed to model them as surface bodies, and mesh them with shell elements, as detailed in 3.4.4.

4 to 15 and 18 to 21. The subsequent fields, pertaining to the energies, have the same meaning as those discussed in the case of a resonator of type Disc. They are simply divided into contributions from the substrate and the two coatings. By symmetry, the values pertaining to coating 1 and 2 are expected to be close to identical.

Whenever the location of the contribution is not specified, i.e. the words "coating" or "substrate" are not present in the name of the field, then the energies are intended as the sum total of all contributions. For instance:

```
shear_energy_approx_modes = shear_energy_approx_substrate_modes +
                            shear_energy_approx_coating1_modes +
                            shear_energy_approx_coating2_modes;
```

and so on.

Finally, recall that the (total) elastic energy is the sum of dilatation + shear. This is true for each location (substrate, coating1, coating2), and globally for the whole sample. However, this separation can be done only on the approximated energies. For instance:

```
elastic_energy_approx_substrate_modes = dilatation_energy_approx_substrate_modes +
                                        shear_energy_approx_substrate_modes;
```

and so on.

16. The next field, `D_c`, is the coating dilution factor for each mode, as defined in Eq. 2. It should correspond to

```
D_c_modes = (elastic_energy_coating1_modes + elastic_energy_coating2_modes) ./
            (elastic_energy_coating1_modes + elastic_energy_coating2_modes
             + elastic_energy_substrate_modes)  =

           = (elastic_energy_coating1_modes + elastic_energy_coating2_modes) ./
           elastic_energy_modes;
```

17. Resonant frequencies for each mode (same as in the case of Disc).

22. The dilution factors `D_TED` are calculated only for the substrate, as the thermoelastic effect in the coating is supposed to be negligible. Therefore, they correspond to:

```
D_TED_modes = dilatation_energy_approx_substrate_modes ./
              elastic_energy_approx_substrate_modes  =

            = dilatation_energy_approx_substrate_modes ./
              (dilatation_energy_approx_substrate_modes + shear_energy_approx_substrate_modes);
```

The second step of the calculation involves the thermoelastic effect. In this case, Vengallatore's model is employed as it can be used for a layered symmetrical structure, such as is a doubly coated sample. Note that the function performing the calculation of the total loss angle has been named differently, to emphasise the specificity of the model being used:

```
mydisc2 = mydisc2.calculate_TED_vengallatore();
mydisc2 = mydisc2.calculate_dissipation_TED:plus_coating();
```

After the computation, the remaining fields of *calculation_results* are automatically populated:

```
                еlement_number_substrate:   726
                 element_number_coating1:   242
                 element_number_coating2:   242
          elastic_energy_substrate_modes:   [7.0792e+06 1.0433e+07 1.9773e+07 4.5662e+07 4.5665e+07]
           elastic_energy_coating1_modes:   [6.3777e+04 6.3734e+04 1.4495e+05 3.3971e+05 3.3973e+05]
           elastic_energy_coating2_modes:   [6.3777e+04 6.3734e+04 1.4495e+05 3.3971e+05 3.3973e+05]
   elastic_energy_approx_substrate_modes:   [6.2720e+06 9.2469e+06 1.7538e+07 4.0043e+07 4.0045e+07]
    elastic_energy_approx_coating1_modes:   [6.4809e+04 6.4620e+04 1.4613e+05 3.4153e+05 3.4161e+05]
    elastic_energy_approx_coating2_modes:   [6.5360e+04 6.5118e+04 1.4616e+05 3.4641e+05 3.4645e+05]
dilatation_energy_approx_substrate_modes:   [3.0210e+05 5.1237e+05 6.2120e+06 3.5447e+06 3.5454e+06]
 dilatation_energy_approx_coating1_modes:   [3.2691e+03 5.5450e+03 6.7239e+04 3.8366e+04 3.8374e+04]
 dilatation_energy_approx_coating2_modes:   [3.2691e+03 5.5450e+03 6.7239e+04 3.8366e+04 3.8374e+04]
     shear_energy_approx_substrate_modes:   [5.9700e+06 8.7345e+06 1.1326e+07 3.6498e+07 3.6499e+07]
      shear_energy_approx_coating1_modes:   [6.1540e+04 5.9075e+04 7.8890e+04 3.0316e+05 3.0324e+05]
      shear_energy_approx_coating2_modes:   [6.2091e+04 5.9573e+04 7.8917e+04 3.0804e+05 3.0807e+05]
                               D_c_modes:   [0.0177 0.0121 0.0144 0.0147 0.0147]
                         frequency_modes:   [604.2330 731.4254 1.0082e+03 1.5322e+03 1.5323e+03]
                    elastic_energy_modes:   [7.2067e+06 1.0560e+07 2.0063e+07 4.6341e+07 4.6345e+07]
             elastic_energy_approx_modes:   [6.4022e+06 9.3766e+06 1.7830e+07 4.0731e+07 4.0733e+07]
          dilatation_energy_approx_modes:   [3.0863e+05 5.2346e+05 6.3464e+06 3.6214e+06 3.6222e+06]
               shear_energy_approx_modes:   [6.0936e+06 8.8532e+06 1.1483e+07 3.7109e+07 3.7111e+07]
                             D_TED_modes:   [0.0482 0.0554 0.3542 0.0885 0.0885]
                                      ff:   [1×5000 double]
                ff_phi_TED_without_D_TED:   [1×5000 double]
                           phi_TED_modes:   [8.5681e-06 1.1766e-05 9.9735e-05 3.4283e-05 3.4289e-05]
             phi_TED_without_D_TED_modes:   [1.7789e-04 2.1234e-04 2.8157e-04 3.8728e-04 3.8729e-04]
                          phi_MEAS_modes:   [1.0338e-05 1.2973e-05 1.0118e-04 3.5749e-05 3.5755e-05]
                    Delta_phi_MEAS_modes:   [1.0338e-07 1.2973e-07 1.0118e-06 3.5749e-07 3.5755e-07]
```

23 and 24. An array of frequencies and the corresponding values of $\phi_s^{TE}(t)$ (without multiplying by $D^{TE}(t)$), meant for plotting, same as in the case of Disc.

25. `phi_TED_modes` corresponds to $D^{TE}(t)\phi_s^{TE}(t)$.

The thermoelastic loss for each mode, in the substrate, comprehensive of the mode-dependent dilution factor.

Recall from Section 2.2 that, in our calculation, we divide the results from Vengallatore's model by the D_TED factors of the substrate that are implicit in that model, and substitute them with the ones previously found with Ansys, and stored in field 22 of *calculation_results*.

26. `phi_TED_without_D_TED_modes` corresponds to $\phi_s^{TE}(t)$.
It contains the same results of line 25, but divided by $D^{TE}(t)$, for each mode.

27. `phi_MEAS_modes` corresponds to $\phi_{meas}(t)$.
It is calculated from the results found in the previous fields, following Eq. 7.

28. `Delta_phi_MEAS_modes` corresponds to `phi_MEAS_modes`/100*`percentage_uncertainty_meas`.
It is simply an estimation of the uncertainty of the measured values, for each mode. The uncertainty on each single measurement, `percentage_uncertainty_meas`, is an attribute of the MatLab variable *mydisc2*, set by default at 1. It can be changed by the user, depending on their specific experimental setup.

References

[1] S. Vengallatore. "Analysis of thermoelastic damping in laminated composite micromechanical beam resonators". In: *Journal of Micromechanics and Microengineering* 15 (2005), p. 2398. DOI: `10.1088/0960-1317/15/12/023`.

[2] G. Cagnoli et al. "Mode-dependent mechanical losses in disc resonators". In: *Physics Letters A* 382.33 (2018), pp. 2165–2173. DOI: `10.1016/j.physleta.2017.05.065`.

[3] F. Piergiovanni and F. Martelli. "Study of the thermoelastic effect in a cylinder". In: *Virgo TDS* VIR-0468A-22 (2010).

[4] J.E. Bishop and V.K. Kinra. "Elastothermodynamic Damping in Laminated Composites". In: *International Journal of Solids and Structures* 34 (1997), pp. 1075–1092. DOI: `10.1016/S0020-7683(96)00085-6`.

[5] C. Zener. "Internal Friction in Solids. I. Theory of Internal Friction in Reeds". In: *Physical Review* 52 (1937), p. 230. DOI: `10.1103/PhysRev.52.230`.

[6] C. Zener. "Internal Friction in Solids II. General Theory of Thermoelastic Internal Friction". In: *Physical Review* 53 (1938), p. 90. DOI: `10.1103/PhysRev.53.90`.

[7] *MATLAB*. The MathWorks, Inc. Natick, Massachusetts (USA).

[8] *ANSYS*. ANSYS, Inc. Canonsburg, Pennsylvania (USA).

[9] L. Anand and S. Govindjee. *Continuum Mechanics of Solids*. Oxford University Press, UK, 2020. Chap. 7. ISBN: 978–0–19–886472–1. DOI: `10.1093/oso/9780198864721.001.0001`.

[10] J.J. Skrzypek and A.W. Ganczarski. *Mechanics of Anisotropic Materials*. Springer International Publishing, Switzerland, 2015. Chap. 1. ISBN: 978-3-319-17160-9. DOI: `10.1007/978-3-319-17160-9`.

[11] R.M. Brannon. *Rotation, Reflection, and Frame Changes - Orthogonal tensors in computational engineering mechanics*. IOP Publishing, Bristol, UK, 2018. Chap. 26. ISBN: 978-0-7503-1454-1. DOI: `10.1088/978-0-7503-1454-1`.

[12] P. Helnwein. "Some remarks on the compressed matrix representation of symmetric second-order and fourth-order tensors". In: *Computer Methods in Applied Mechanics and Engineering* 190 (2001), pp. 2753–2770.

[13] *Theory Reference for the Mechanical APDL and Mechanical Applications*. ANSYS, Inc. 2009 (Release 12.0).

[14] P.P. Teodorescu. *Treatise on Classical Elasticity - Theory and Related Problems*. Springer, Dordrecht, 2013. Chap. 14. ISBN: 978-94-007-2616-1. DOI: 10.1007/978-94-007-2616-1.

[15] D. Healy, N.E. Timms, and M.A. Pearce. “The variation and visualisation of elastic anisotropy in rock-forming minerals”. In: *Solid Earth* 11 (2020), pp. 259–286. DOI: 10.5194/se-11-259-2020.

[16] M. Bao. *Analysis and Design Principles of MEMS Devices*. Elsevier Science, 2005. Chap. 6. ISBN: 978-0-444-51616-9. DOI: 10.1016/B978-0-444-51616-9.X5000-0.

[17] L. Zhang et al. “Anisotropic elasticity of silicon and its application to the modelling of X-ray optics”. In: *Journal of Synchrotron Radiation* 21 (2014), pp. 507–517. DOI: 10.1107/S1600577514004962.

[18] R. Hill. “The Elastic Behaviour of a Crystalline Aggregate”. In: *Proceedings of the Physical Society. Section A* 65.5 (1952), pp. 349–354. DOI: 10.1088/0370-1298/65/5/307.

[19] R. deWit. “Elastic constants and thermal expansion averages of a nontextured polycrystal”. In: *Journal of Mechanics of Materials and Structures* 3.2 (2008), pp. 195–212. DOI: 10.2140/jomms.2008.3.195.

[20] Y.-C. Zhou et al. “Theoretical Prediction and Experimental Investigation on the Thermal and Mechanical Properties of Bulk beta-Yb2Si2O7”. In: *Journal of the American Ceramic Society* 96.12 (2013), pp. 3891–3900. DOI: 10.1111/jace.12618.

[21] H.-H. Lee. *Finite Element Simulations with ANSYS Workbench 12*. Schroff Development Corporation, 2010. Chap. 6. ISBN: 978-1-58503-604-2.

[22] *ANSYS Mechanical APDL Element Reference*. ANSYS, Inc. 2011 (Release 14.0).