Viscoelastic Vibration Toolbox

For Use with MATLAB®

User's Guide

Version 1.1

Etienne Balmes

How to Contact SDTools

33 +1 44 24 63 71	Phone
SDTools 44 rue Vergniaud 75013 Paris (France)	Mail

https://www.sdtools.com	Web
https://www.openfem.net	An Open-Source Finite Element Toolbox

support@sdtools.com	Technical support
suggest@sdtools.com	Product enhancement suggestions
info@sdtools.com	Sales, pricing, and general information

Contents

1 Modeling viscoelastic materials **5**

1.1 Introduction . . . 6

1.2 Representing complex modulus . . . 6

1.2.1 Non parametric (tabular) representations . . . 7

1.2.2 Simple rheological models . . . 9

1.2.3 High order rational models . . . 11

1.2.4 Fractional derivative models . . . 13

1.2.5 3D constitutive laws . . . 13

1.3 Environmental factors . . . 14

1.3.1 Influence of temperature . . . 14

1.3.2 Other environment factors . . . 18

1.4 Determining the complex modulus . . . 19

1.5 Conclusion . . . 20

2 Viscoelastic FEM models **23**

2.1 Viscous and structural damping . . . 24

2.1.1 Properties of the damped 1 DOF oscillator . . . 25

2.1.2 Real modes and modal damping . . . 27

2.1.3 Selection of modal damping coefficients . . . 29

2.2 Viscoelastic models . . . 31

2.2.1 Frequency domain representation with variable coefficients . . . 31

2.2.2 State-space representations . . . 32

2.2.3 Second order models with internal states . . . 34

2.2.4 Fractional derivatives . . . 35

2.3 Spectral decomposition and reduced models . . . 36

2.3.1 Complex modes of analytical models . . . 36

2.3.2 Complex mode eigenvalue problems with constant matrices . . . 37

2.3.3 Model reduction methods . . . 39

2.3.4 Equivalent viscous damping . . . 41

2.3.5 Case of viscoelastic models . . . 42
2.4 Meshing of sandwich models . . . 43
2.4.1 Mesh convergence and non conformity . . . 47
2.5 Thermal considerations . . . 47
2.5.1 Thermal model . . . 48
2.5.2 Heat source due to viscoelastic behavior . . . 48
2.5.3 Cantilever plate example . . . 49
2.5.4 Thermo-elastic damping . . . 50

3 Composite FEM modeling **53**
3.1 Material models . . . 54
3.1.1 Classical lamination theory . . . 54
3.1.2 3D anisotropy . . . 57
3.2 Ply generation example . . . 57
3.3 Material orientation maps . . . 58
3.4 Topology optimization tools . . . 58

4 Toolbox tutorial **61**
4.1 Download and installation procedures . . . 63
4.2 Representing viscoelastic materials . . . 63
4.2.1 Introducing your own nomograms . . . 64
4.2.2 Selecting a material for your application . . . 65
4.2.3 Selective components in constitutive law . . . 65
4.3 Viscoelastic device meshing tools . . . 66
4.3.1 Generation of sandwich models . . . 66
4.3.2 Meshing foam fillings . . . 67
4.3.3 Exporting submeshes to NASTRAN . . . 68
4.4 Parametric models, structure reference . . . 68
4.4.1 Parametric models, zCoef . . . 68
4.4.2 Parametric models, zCoef . . . 72
4.4.3 Input definitions . . . 74
4.4.4 Sensor definitions . . . 74
4.4.5 MVR Format reference . . . 75
4.4.6 Response post-processing options . . . 75
4.5 Sample setup for parametric studies . . . 76
4.5.1 Performance in modulus/loss plane . . . 76
4.5.2 Illustration of pole range computations . . . 78
4.5.3 Model parameterization . . . 78
4.5.4 Sample parametric study in SDT (full solver, Upcom superelement) . . . 79
4.5.5 Parametric model generated within NASTRAN (fo_by_set DMAP) . . . 80

4.5.6 Parametric model from NASTRAN element matrices 82
4.6 Fluid/structure coupling 83
4.6.1 Summary of theory 83
4.6.2 Acoustic stiffness on a loudspeaker 84
4.7 Rayleigh integral computations 85
4.7.1 Summary of theory 85
4.7.2 Diffuse field and transmission loss 85
4.8 NASTRAN Generation of the parametric model 86
4.9 Advanced connection models 87
4.9.1 Screw models 87
4.9.2 Physical point with rotations 88
4.10 External links 88

5 Toolbox Reference **89**
fe2xf 90
fevisco 101
m_visco, mvisco_3m 109
nasread 116
naswrite 121

Bibliography **127**

Index **130**

1

Modeling viscoelastic materials

Contents

1.1 Introduction 6

1.2 Representing complex modulus 6

1.2.1 Non parametric (tabular) representations 7

1.2.2 Simple rheological models 9

1.2.3 High order rational models 11

1.2.4 Fractional derivative models 13

1.2.5 3D constitutive laws 13

1.3 Environmental factors 14

1.3.1 Influence of temperature 14

1.3.2 Other environment factors 18

1.4 Determining the complex modulus 19

1.5 Conclusion 20

1.1 Introduction

Linear viscoelasticity assumes [1] that stress is a function of strain history. This translates into the existence of a relaxation function $h(t)$ given by

$$\sigma(t) = \int_0^{\infty} \varepsilon(t-\tau)h(\tau)d\tau \tag{1.1}$$

Using Laplace transform, one sees that this hypothesis is equivalent to the existence of a complex modulus $\Lambda(s)$ (transform of $h(t)$) such that

$$\sigma(s) = \Lambda(s)\varepsilon(s) = (\Lambda'(s) + i\Lambda''(s))\varepsilon(s) \tag{1.2}$$

From a practical point of view, one can solve viscoelasticity problems as elasticity problems with a complex modulus that depends on frequency. This property is known as the **elastic/viscoelastic equivalence principle** [1].

For a strain tensor, the number of independent coefficients in Λ is identical to that in Hooke's law for an elastic material (for the same reasons of material invariance). For homogeneous and isotropic materials, one thus considers a Young's modulus and a Poisson coefficient that are complex and frequency dependent.

The separate measurement of $E(\omega)$ et $\nu(\omega)$ is however a significant experimental challenge [2] that has no well established solution. Practice is thus to measure a compression $E(\omega)$ or a shear modulus $G(\omega)$ and to assume a constant Poisson's ratio, although this is known to be approximate.

Section 1.2 analyzes the main representations of complex moduli. Section 1.3 lists representations used to account for the effect of environmental factors (temperature, pre-stress, ...). Domains of applicability for different models are given at the end of the chapter.

For more details, Ref. [1] a detailed account of viscoelasticity theory. Refs [2, 3] present most practical representations of viscoelastic behavior.

1.2 Representing complex modulus

Even though, viscoelastic constitutive laws are fully characterized by the measurement of a complex modulus. As will be detailed in the next chapter, numerical solvers are often associated with specific analytical representations of moduli. This section illustrates the main classes of modulus representations.

1.2.1 Non parametric (tabular) representations

At each frequency, the complex modulus describes an elliptical stress/strain relation. For $\epsilon = \text{Re}(e^{i\omega t})$, one has $\sigma(t) = \text{Re}(E_s(1+i\eta)e^{i\omega t}) = E_s(\cos\omega t - \eta\sin\omega t)$ as shown in figure 1.1. One calls, **storage modulus** the real part of the complex modulus $E_s = \text{Re}(E)$ and **loss factor** the ratio of the imaginary and real parts $\eta = \text{Im}(E)/\text{Re}(E)$. The *loss factor* can also be computed as the ratio of the power dissipated over a cycle, divided by 2π by the maximum strain energy

$$\eta(X,T) = \frac{\frac{1}{2\pi}\int_0^T \sigma(X,t)\dot{\epsilon}(X,t)dt}{\frac{1}{2}\max_0^T\left(\sigma(X,t)\epsilon(X,t)\right)} \tag{1.3}$$

This definition in terms of dissipated energy extends the complex modulus definition to cases with small non linearities. In such cases, the dissipation may be a structural rather than a material effect and may depend on amplitude or history so that its use as a constitutive parameters may not be relevant.

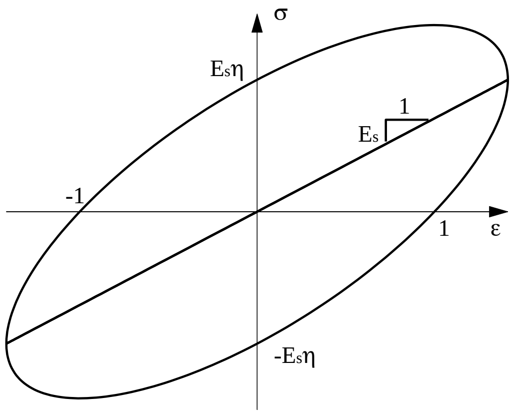


Figure 1.1: Elliptical stress/strain cycle

The raw measure of viscoelastic characteristics gives a complex modulus which typically has the characteristics shown in figure 1.2.

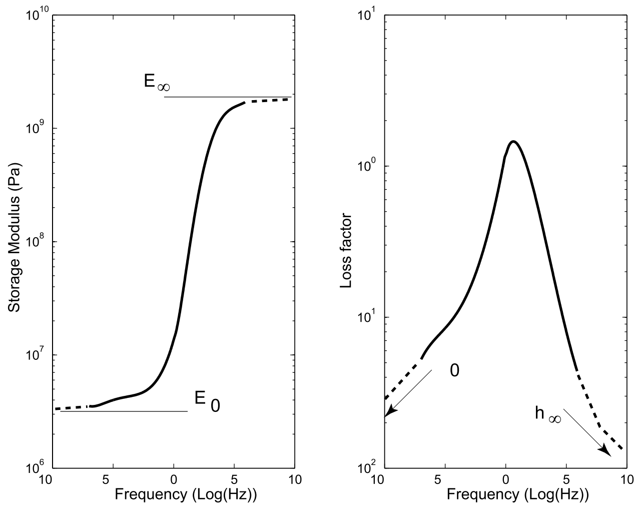


Figure 1.2: Sample constitutive law for a viscoelastic material

The solution of frequency domain problems (frequency response functions, complex mode extraction) requires the interpolation of measurements between frequency points (shown as a solid line) and the extrapolation in unmeasured low and high frequency areas (shown in dotted lines in the figure).

For interpolation, a moving average, weighted for 2 or 3 experimental points, is easily implemented and fast to compute. For extrapolation at low frequencies, one will assume a real asymptote $E(0)$ (i.e. $\eta(0) = 0$) from basic principles : since the relaxation function is real, its Fourier transform is even and thus real at 0. For high frequencies, one uses a complex asymptote E_∞, η_∞. There are mathematical limitations, on possible values for this asymptote but in practice, there are also other physical dissipation mechanisms that are not represented by the viscoelastic law so that the real objective is to avoid numerical pathologies when pushing the model outside its range of validity.

The main advantage of non parametric representations of constitutive laws is to allow fully general accounting of behavior that are strongly dependent on frequency, temperature, pre-stress, ... Moreover, the direct use of experimental data avoids standard steps of selecting a representation and identifying the associated parameters. Since good software to perform those steps is not widespread, avoiding them is really useful.

Disadvantages are very few and really mostly linked to the extraction of complex modes and as a consequence time response simulations using modal bases. And even these difficulties can be circumvented.

1.2.2 Simple rheological models

The classical approach in rheology is to represent the stress/strain relation as a series of springs and viscous dashpots. Figure 1.3 shows the simplest models [3].

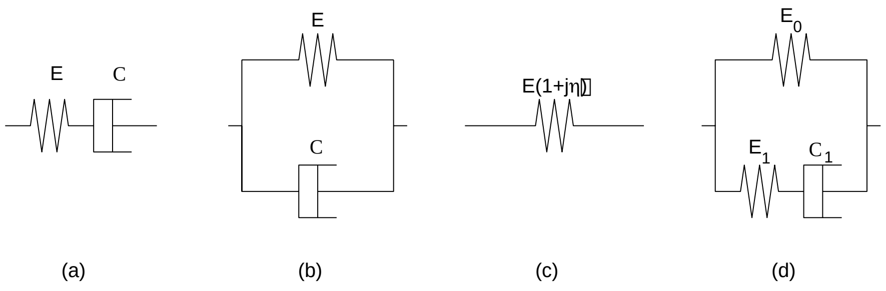


Figure 1.3: Material damping models with 2 or 3 parameters : (a) Maxwell's model, (b) viscous damping (Kelvin-Voigt's model), (c) structural damping, (d) standard viscoelastic solid

Maxwell's model is composed of a spring and a dashpot in series. Each element carries the same load while strains are added. The stress / strain relation is, in the frequency domain, written as

$$\varepsilon = \left(\frac{1}{E} + \frac{1}{Cs}\right) = \left(\frac{Es}{s + E/C}\right)^{-1} \sigma \tag{1.4}$$

where

$$\tau = \frac{C}{E} \tag{1.5}$$

is a settling time that is characteristic of the model and is associated to a pole at $\omega = -E/C = -1/\tau$.

The viscous model, called *Kelvin Voigt Model*, is composed of a spring and a dashpot in parallel. Each element undergoes the same elongation but strains add. Damping is then represented by the addition of strains proportional to deformation and deformation velocity. In the frequency domain the modulus has the form

$$E(s) = E_0(1 + \beta s) \tag{1.6}$$

Structural damping, also called hysteretic damping, is introduced by considering a complex stiffness that is frequency independent. The complex modulus is thus characterized by constant storage moduli and loss factors

$$E(s) = E_0(1 + j\eta) \tag{1.7}$$

The standard viscoelastic solid has a constitutive law given by

$$\sigma(s) = \left(E_0 + \left(\frac{1}{E_1} + \frac{1}{C_1 s}\right)^{-1}\right)\varepsilon = \frac{E_0 E_1 + (E_0 + E_1)C_1 s}{E_1 + C_1 s}\varepsilon = E_0 \frac{1 + s/z}{1 + s/p}\varepsilon \tag{1.8}$$

leading to a time domain representation

$$E_1 \sigma(t) + C_1 \dot{\sigma}(t) = E_0 \varepsilon + E_1 C_1 \dot{\varepsilon} \tag{1.9}$$

Figure 1.4 shows the frequency domain properties of these models. Maxwell's model is only valid in the high frequency range since its static stiffness is zero. The viscous damping model is unrealistic because the loss factor goes to infinity at high frequency (there deformation locking).

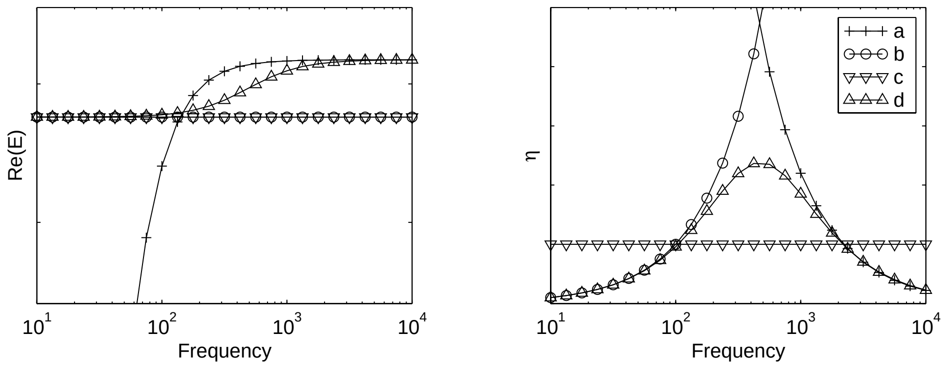


Figure 1.4: Complex modulus associated to standard models: (a) Maxwell's model, (b) viscous damping (Kelvin-Voigt's model), (c) structural damping, (d) standard viscoelastic solid

The structural damping model approximates the modulus by a complex constant. Although it does not represent accurately any viscoelastic material, such as the one shown in figure 1.2, it leads to a good approximation when the global behavior of the structure is not very sensitive to the evolution of the complex modulus with frequency. This model is generally adapted for materials with load damping (metals, concrete, ...).

The standard viscoelastic model has the main characteristics of real materials : low and high frequency asymptotes, maximum dissipation at the point of highest slope of the real part of the modulus.

This model is characterized a nominal level E_0 and two pairs of related quantities: a pole

$$p = E_1/C_1 = \omega_m \left(\eta_m + \sqrt{1 + \eta_m^2}\right), \tag{1.10}$$

where the modulus flattens out, and a zero

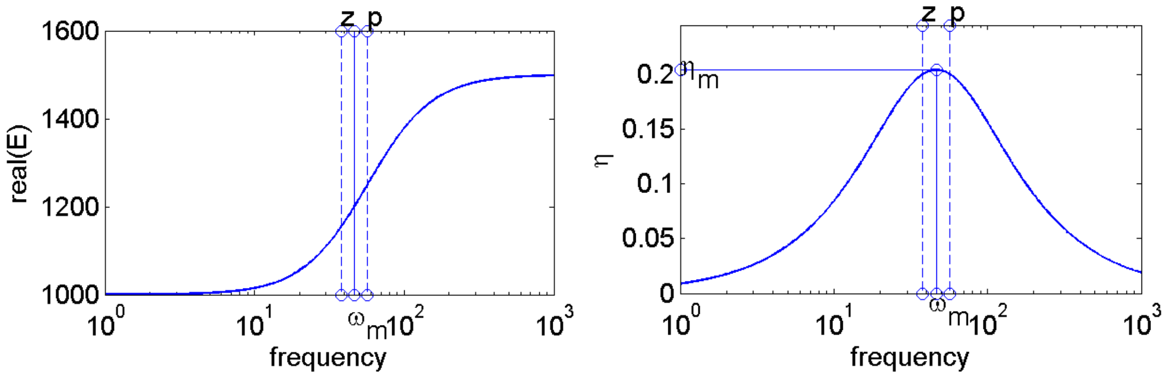


Figure 1.5: Complex modulus associated to standard viscoelastic solid model

$$z = (E_0 E_1)/((E_0 + E_1)C_1) = \frac{\omega_m}{\eta_m + \sqrt{1+\eta_m^2}} \tag{1.11}$$

giving the inflection point where the modulus starts to augment; or the maximum loss factor

$$\eta_m = \frac{(p-z)}{2\sqrt{pz}} = \frac{1}{2}\frac{E_1}{\sqrt{E_0(E_0+E_1)}} \tag{1.12}$$

and the frequency where this maximum is reached (between the pole and the zero)

$$\omega_m = \sqrt{pz} = \frac{E_1}{C_1}\sqrt{\frac{E_0}{E_0+E_1}} \tag{1.13}$$

The maximum loss factor is higher when the pole and zero are well separated and the low and high frequency moduli differ. This is consistent with the fact that materials that dissipate well also have storage moduli that are strongly frequency dependent.

Figure 1.6 illustrates problems encountered with three parameter models. The parameters of the standard viscoelastic model where chosen to match the low and high frequency asymptote and the frequency of the maximum in the loss factor. One clearly sees that the slope is not accurately represented and more importantly that the loss factor maximum and evolution in frequency are incorrect. These limitations have motivated the introduction of more complex models detailed in the following sections.

1.2.3 High order rational models

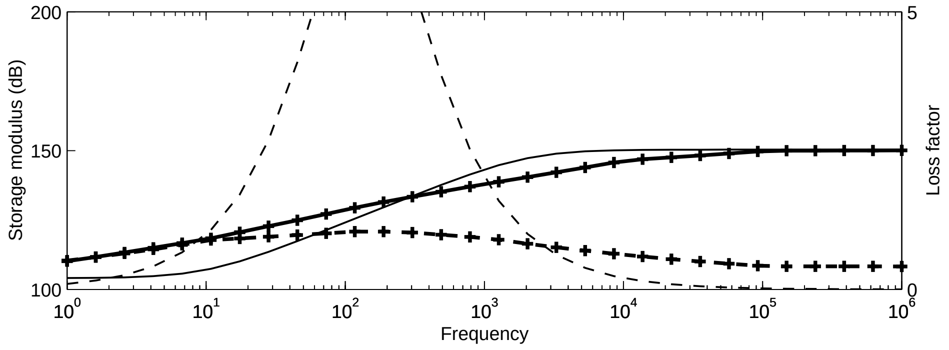


Figure 1.6: E amplitude (—) and loss factor (- - -) of ISD112 (+) overlaid with a 3 parameter model

The first generalization is the introduction of higher order models in the form of rational fractions. All classical representations of rational fractions have been considered in the literature. Thus, fractions of high order polynomials, decompositions in products of first order polynomials giving poles and zeros, sums of low order fractions

$$E(s) = E_0 \frac{1+\sum\limits_{i=1}^{n}\beta_i(s)^i}{1+\sum\limits_{i=1}^{n}\alpha_i(s)^i} = E_\infty \frac{\prod\limits_{i=1}^{n}(s-z_i)}{\prod\limits_{i=1}^{n}(s-\lambda_i)} = E_0 + \sum_{i=1}^{n}\frac{E_i s}{s+E_i/C_i} \tag{1.14}$$

are just a few of various equivalent representations.

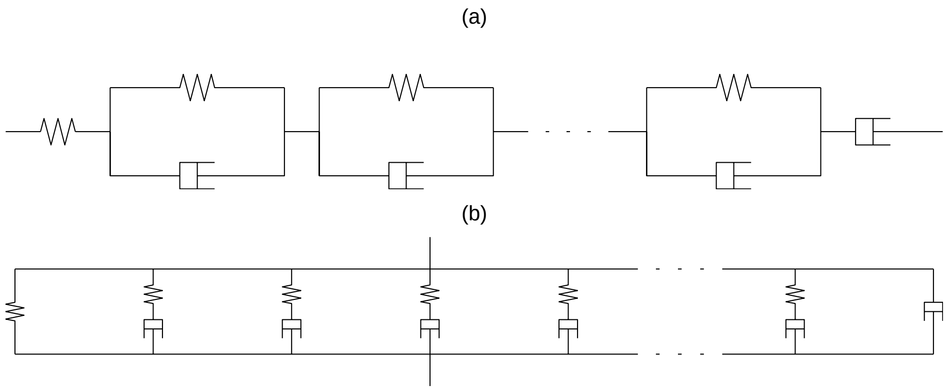


Figure 1.7: Examples of generalized damping models (a) Kelvin's chain, (b) generalized Maxwell model

Figure 1.7 shows two classical representations. Maxwell generalized model introduces a pole ($\lambda_j = E_j/C_j$, associated to settling time $\tau_j = C_j/E_j$) and high frequency stiffness E_j for each spring/dashpot branch.

The physical meaning of each representation is identical. However, each representation may have advantages when posing equations for implementation in a numerical solver. For example, section 2.2.3 will discuss models using particular representations of the modulus of order 2 (so called GHM models for Golla, Hughes, McTavish [4, 5]) and order 1 (so called ADF models for Anelastic Displacement Field [6, 7, 8]). This section will also show how these models correspond to the formalism of materials with internal states that are commonly used in non linear mechanics.

Rational fractions have a number of well known properties. In particular, the modulus slope at a given frequency $s = i\omega$ is directly characterized by the position of poles and zeros in the complex plane (see Bode diagram building rules in any course in controls [9]). The modulus slopes found in true materials are generally such that it is necessary to introduce at least one pole per decade to accurately approximate experimental measurements (see [10] for a discussion of possible identification strategies). This motivated the use of fractional derivatives described in the next section.

1.2.4 Fractional derivative models

The use of non integer fractions of s allows a frequency domain representation with an arbitrary slope. A four parameter model is thus proposed in [11]

$$E(s) = E_{inf}(g_0 - \sum_i \frac{g_i}{1 + (s/\omega_i)^{\alpha^i}} \tag{1.15}$$

where the high E_{inf} and low $E_{min} = g_0 * E_{inf}$ frequency moduli are readily determined and the ω and α coefficients are used to match the frequency of the maximum loss factor and its value. Properties of this model are detailed in [2].

One can of course use higher order fractional derivative models. To formulate constant matrix models one is however bound to use rational exponents (see section 2.2.4).

The main interest of fractional derivative models is to allow fairly good approximations of realistic material behavior with a low number of parameters. The main difficulty is that their time representation involves a convolution product. One will find in Ref. [12] an analysis of the thermomechanical properties of fractional derivative models as well as a fairly complete bibliography.

1.2.5 3D constitutive laws

It is often necessary to split constitutive laws. This is done in `fevisco MatSplit`. For isotropic materials the usual law

$$D = \begin{bmatrix} \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \end{bmatrix} \tag{1.16}$$

with at nominal $G = E/(2(1+\nu))$. But in reality the bulk modulus $K = E./(3*(1-2\nu))$ is not very damped and does not decrease notably with temperature while G is much more sensitive.

$$D = K \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + G \begin{bmatrix} 4/3 & -2/3 & -2/3 & 0 & 0 & 0 \\ -2/3 & 4/3 & -2/3 & 0 & 0 & 0 \\ -2/3 & -2/3 & 4/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

1.3 Environmental factors

For usual damping materials, the complex modulus depends not only of frequency but also of other environmental factors such as temperature, prestress, etc. The following sections discuss these factors and the representation in constitutive laws.

1.3.1 Influence of temperature

Temperature is the environmental factor that has the most influence on viscoelastic material characteristics [2]. At various temperatures, these materials typically have four different regions shown in figure 1.8: glassy, transition, rubberlike and fluid. Depending on the considered material, the operating temperature can be in any of the four regions. For polymer blends, each polymer can be in a different region.

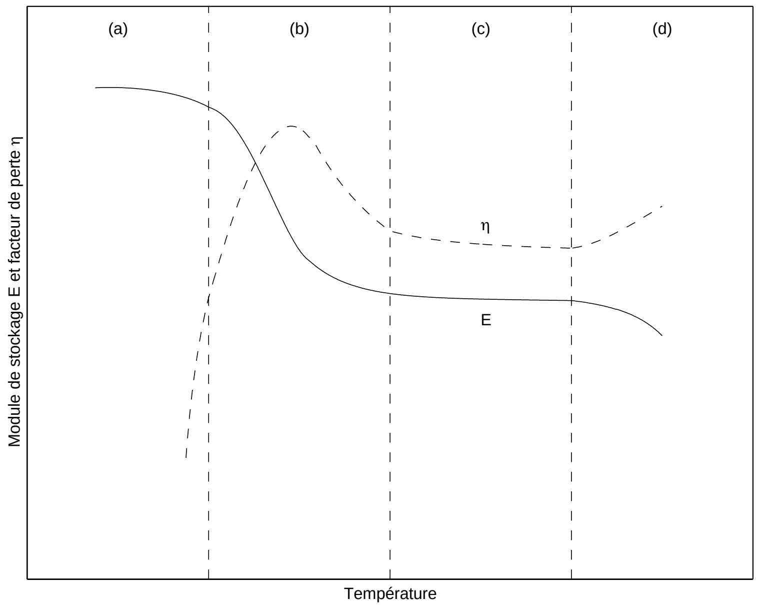


Figure 1.8: Evolution of complex modulus with temperature at a fixed frequency. Regions : (a) vitrous, (b) transition, (c) rubberlike, (d) fluid.

In the first region, associated with low temperatures, the material in its glassy state is characterized by a storage modulus that reaches its maximum value and has low variations with temperature. The loss factor is very small and diminishes with temperature. Material deformations are then small.

The transition region is characterized by a modulus decreasing with temperature and a loss factor peaking in the middle of the region. Typically the maximum corresponds to the point of maximum slope for the storage modulus. The associated temperature is called transition temperature. Note that this can be confusing since the transition temperature depends on the frequency used to generate figure 1.8.

In the rubberlike region storage modulus and loss factor are both characterized by relatively small values and low temperature dependence. The fourth region corresponds to a fluid state. This region is rarely considered because of its inherent instability.

For damping applications, one typically uses viscoelastic materials in the transition region. This choice is motivated by the fact that the loss factors presents a maximum in this area, thus allowing an efficient use of the material damping properties.

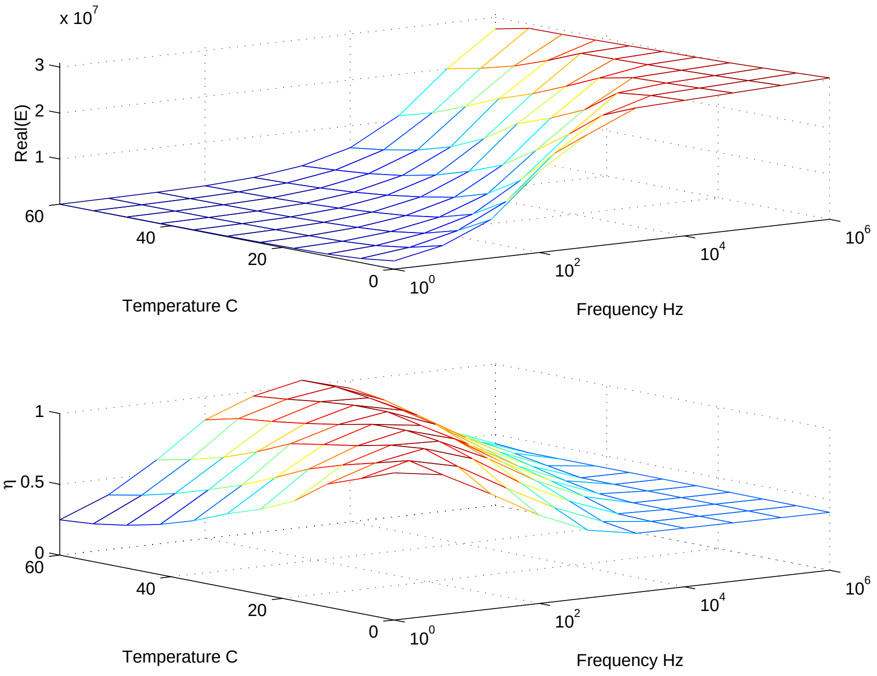


Figure 1.9: Variations of ISD 112 modulus in the frequency / temperature domain

In the frequency / temperature domain, figure 1.9 illustrates the existing inverse relation of the effects of temperature and frequency. Experimentally, one often finds that by shifting isotherm curves along the frequency axis by a given factor α_T, one often has good superposition.

This property motivates the introduction of the *reduced frequency* $\alpha(T)\omega$ and a description of the complex modulus under the form

$$E(\omega, T) = \hat{E}(\alpha(T)\omega) \tag{1.18}$$

The validity of this representation is called the *frequency temperature superposition principle* [2, 13] and the curve $\hat{E}$ is called a master curve.

Various authors have given thermodynamic justifications to the *frequency temperature superposition principle*. These are limited to unique polymers. For polymer blends, which have significant advantages, it is not justified.

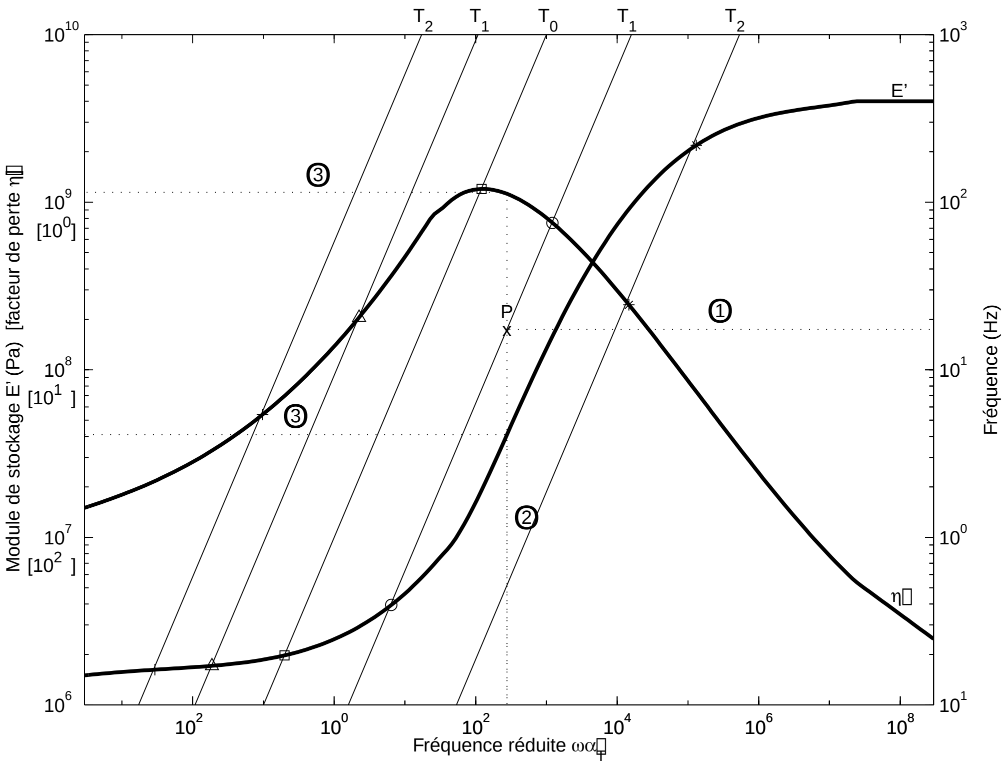


Figure 1.10: Reduced frequency nomogram

The superposition principle is used to build a standard representation called *nomogram* which simplifies the analysis of properties as a function of temperature T and frequency ω. The product $\omega\alpha_T$ corresponds to an addition on a logarithmic scale. One thus defines *true* frequencies on the right vertical axis, and isotherm lines allowing to read the reduced frequency graphically on the horizontal axis. For a frequency ω_j and a temperature T_k, one reads the nomogram in three step shown in figure 1.10

- (1) one seeks the intersection P of the horizontal ω_j line and the of sloped T_k line,
- (2) the abscissa of point P gives the reduced frequency $\omega_j\alpha_T(T_k))$,
- (3) the intersection of the vertical line at that reduced frequency with the storage modulus E and loss factor η master curves gives their respective values at ω_j, T_k.

Various parametric expression have been proposed to model the temperature shift factor α_T. The empirical equation of Williams-Landel-Ferry [14], called WLF equation,

$$\mathrm{Log}(\alpha_T) = \frac{-C_1(T-T_0)}{C_2+T-T_0} \tag{1.19}$$

is often used. Various papers state that $C_1 = 17.4$ et $C_2 = 51.6$ are realistic values for many materials with changing T_0 but this claims seems mostly unfounded.

On should also cite the α_T model based on Arrhenius equation used in thermodynamics to quantify the relation between the rate of a chemical reaction and it's temperature

$$\mathrm{Log}(\alpha_T) = \frac{E_a}{R}\left(\frac{1}{T}-\frac{1}{T_0}\right) \tag{1.20}$$

where T is the temperature in degree Kelvin, $R = 8.314 \times 10^{-3} kJmol^{-1}K^{-1}$ is constant of perfect gas and Ea corresponds to the activation energy of the reaction. This relation is less used than the WLF equation or other models uniquely based on ΔT, but the reason is probably only linked to easiness in the determination of parameters.

Figure 1.11 shows typical α_T curves and their expression as a function of ΔT (difference between T and a given reference temperature T_0). One clearly sees that these curves mostly differ in their low temperature behavior. In practical applications, one can usually adjust parameters of any law to be appropriate. The solution preferred here is to simply interpolate between points of an α_T table.

1.3.2 Other environment factors

Between the other environmental factors influencing the behavior, one essentially distinguishes non linear effects (static and dynamic) and history effects (exposition to oil, high temperatures, vacuum, ...).

Non linear dynamic effects are very hard to characterize since high amplitude variations of the induced strain are typically correlated with significant energy dissipation and thus temperature changes. The effects of level and temperature changes are then coupled for materials of interest which are typically in the transition region. Experimentally, such non linear studies are thus limited to the rubber like region. The effects are similar to those of temperature although of smaller magnitude.

Non linear static effects, that is effects linked to a static prestress assumed to be constant, are significant and easier to characterize. It is known to be essential when considering machinery suspensions or constrained viscoelastic sandwiches where press-forming induced significant pre-stress [15].

History effects are generally associated with extreme solicitation that one seeks to avoid but whose probability of occurrence is non zero.

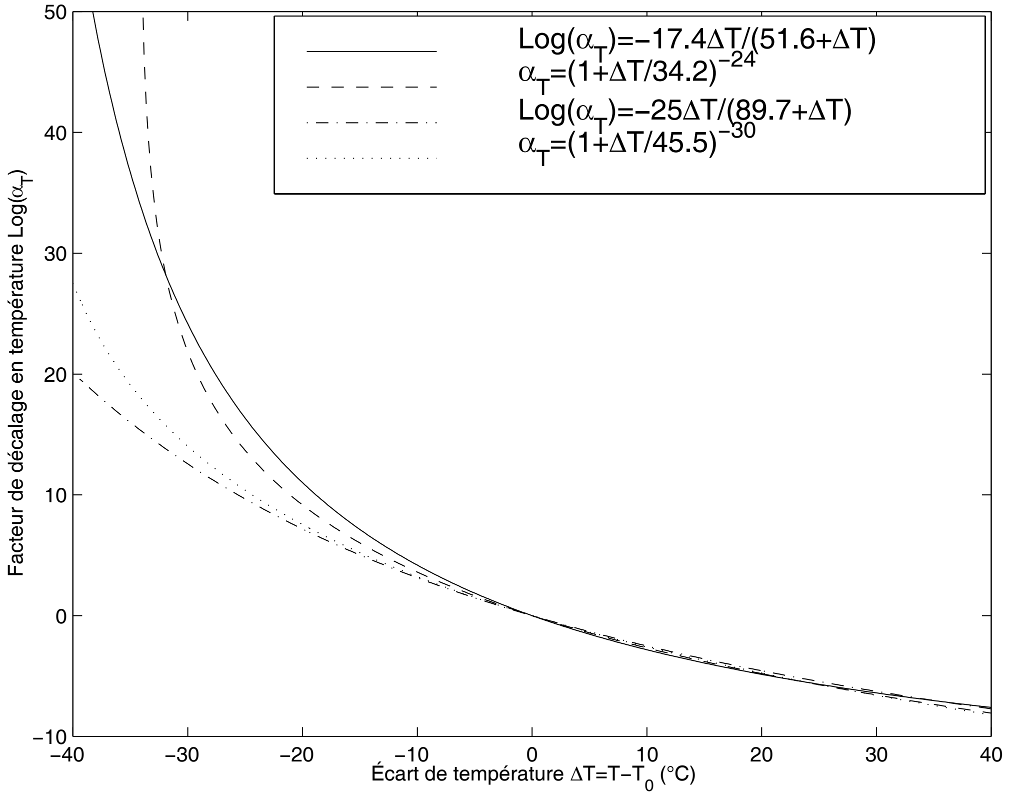


Figure 1.11: Frequency shift factor α_T and reduced frequency

As for temperature, one generally represent the effect of other environmental factors as shift factors [2], although the superposition hypothesis may not be as well verified [15].

1.4 Determining the complex modulus

A number of methods and a few standards [16], exist for the experimental determination of the complex modulus. Only the main test categories will be listed here

- traction/compression tests under sinusoidal excitation are used to measure the properties of materials that are sufficiently stiff to allow testing without combination of the material sample with metallic components. With significant experimental precautions, this technique has also been applied to films. Depending on the experimental setup, one will determine the complex modulus directly on isofrequencies or isotherms.

- Oberst [17] and sandwich beams are used to determine the properties of a viscoelastic layer glued onto a metallic beam. Stiff materials work in traction compression in a free layer, while soft ones work in shear in a metal/visco/metal sandwich. Vibration analysis of the beam gives resonance frequencies and associated damping ratio. By inverse analysis of analytical [18] or numerical solutions, one determines the complex modulus at the resonances under the modal damping assumption (see section 2.1.3). Using changes in the beam/plate dimensions and temperatures, one can obtain a large number of points on the nomogram.

- shear tests allow the direct determination of the complex modulus of films between two rigid surfaces. This test is more representative of material solicitation for sandwich structures. It is the only one applicable to test the effect of pre-stress in sandwich structures.

Building a test rig with no perturbing modes being quite difficult, modulus characterization is always performed on fairly narrow frequency bands. The frequency-temperature superposition hypothesis is thus made to create master curves over a wide frequency band. The next possible step is the determination of the coefficients of an analytical representation. Identification tools developed in control theory (ARMA models, ...) are suited for rational fraction models. It is however difficult to enforce a good reproduction of quantities that are typically judged as important : low and high frequency moduli, maximum loss factor, ... In other case a non linear optimization is readily implemented using optimization tools available in MATLAB.

1.5 Conclusion

For a given complex modulus, it is difficult to validate the fact that general thermodynamic principles are verified. Ref. [12] gives an analysis based on the fact that each component of a generalized maxwell model must verify the second principle of thermodynamics and thus be associated with positive stiffness and damping coefficients.

In practice, the criteria that are used to judge the quality of a constitutive model are the accurate reproduction of experimental complex modulus measurements within the tested frequency/temperature range, and the likelihood of extrapolations. Figure 1.12 illustrates the fact that with rational fractions one can fairly easily satisfy the first objective (reproduce the modulus in a narrow band in bold), but extrapolations can be difficult (thin lines are here of poor quality).

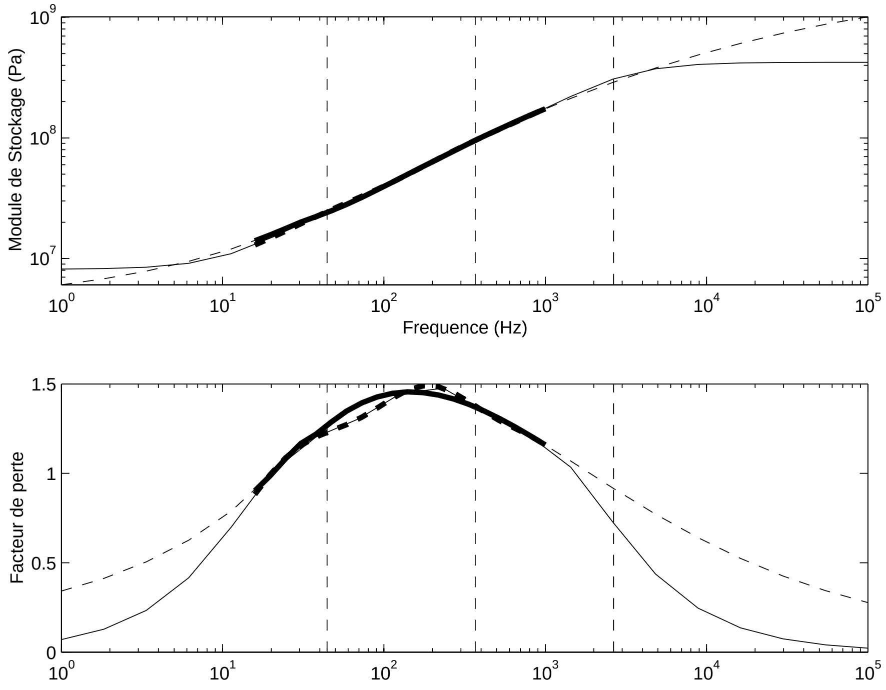


Figure 1.12: Complex modulus of the TA viscoelastic (—) and estimation with a 3 pole model (- - -) whose frequencies are indicated as vertical lines

On the other hand, one should note that the determination of the complex modulus is often difficult and that the frequency / temperature superposition principle is an hypothesis. It just happens to be reasonably valid in many cases.

In conclusion, for the material stand-point, accurate representations are either tabulated (our point of view is that this is actually the most practical representation), rational fractions with a sufficient degree or fractional derivative models. The following chapters will address difficulties in using these representations for modeling damped structures.

2

Viscoelastic FEM models

Contents

2.1 Viscous and structural damping **24**
2.1.1 Properties of the damped 1 DOF oscillator 25
2.1.2 Real modes and modal damping 27
2.1.3 Selection of modal damping coefficients 29
2.2 Viscoelastic models **31**
2.2.1 Frequency domain representation with variable coefficients 31
2.2.2 State-space representations 32
2.2.3 Second order models with internal states 34
2.2.4 Fractional derivatives 35
2.3 Spectral decomposition and reduced models **36**
2.3.1 Complex modes of analytical models 36
2.3.2 Complex mode eigenvalue problems with constant matrices 37
2.3.3 Model reduction methods 39
2.3.4 Equivalent viscous damping 41
2.3.5 Case of viscoelastic models 42
2.4 Meshing of sandwich models **43**
2.4.1 Mesh convergence and non conformity 47
2.5 Thermal considerations **47**
2.5.1 Thermal model 48
2.5.2 Heat source due to viscoelastic behavior 48
2.5.3 Cantilever plate example 49
2.5.4 Thermo-elastic damping 50

The design of viscoelastic treatments is typically composed of a series of steps which are outlined in this section

- definition of dynamic objectives
- potential area localization by combination of technological constraints on placement and sensitivity analysis on treatment potential
-

This chapter analyzes properties of models used to represent damped structures. In the case of linear viscoelasticity these models have the general form

$$\begin{array}{c}[Z(s,T)]\,\{q(s)\} = [b]_{N\times NA}\,\{u(s)\}_{NA\times 1}\\ \{y(s)\}_{NS\times 1} = [c]_{NS\times N}\,\{q(s)\}_{N\times 1}\end{array} \tag{2.1}$$

where physical loads $[b]\,\{u(s)\}$ are decomposed into vectors b, characterizing spatial localization, and $u(s)$, characterizing time/frequency responses and vector $\{y\}$ represents outputs (physical quantities to be predicted, assumed to depend linearly in DOFs q.

Section 2.1 details the case of models with viscous and structural damping. Although one saw in the last chapter that these models did not allow a correct representation of material behavior over a wide frequency band, they are more easily accessible and can be used to illustrate the difference in needs when modeling vibration damping in a structure and dissipation in a material.

Section 2.2 details strategies that can be used to create models with realistic representation of viscoelastic materials.

Section 2.3 deals with issues of spectral decompositions and model reduction which are central to the analysis of the vibratory behavior of damped structures.

The objective of this chapter is to introduce equations used to solve viscoelastic problems. The next chapter will focus on numerical techniques used for the resolution.

2.1 Viscous and structural damping

In this section one analyses the properties of classical models with viscous and structural damping of the form

$$\begin{array}{c}\left[Ms^2 + Cs + K + iD\right]_{N\times N}\{q(s)\}_{N\times 1} = [Z(s)]\,\{q(s)\} = [b]_{N\times NA}\,\{u(s)\}_{NA\times 1}\\ \{y(s)\}_{NS\times 1} = [c]_{NS\times N}\,\{q(s)\}_{N\times 1}\end{array} \tag{2.2}$$

where matrices are assumed to be constant.

This type of models does not allow a correct representation of the local behavior of damping treatments (constitutive laws detailed in the preceding chapter). At the level of a complete structure, it is however often possible to represent the effect of various damping mechanisms by a viscous or structural model. One then uses a global *behavior* model. It does not necessarily have a local mechanical meaning, but this does not lower its usefulness.

The main results introduced in this section are

- properties of the damped oscillator;
- the conditions of validity of the `modal damping assumption` which leads to models where the response is decomposed in a sum of independent oscillators;
- techniques used to estimate equivalent viscous damping models.

2.1.1 Properties of the damped 1 DOF oscillator

This section illustrates the properties of the single degree of freedom oscillator with a viscoelastic stiffness shown in figure 2.1.

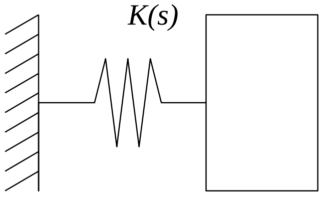


Figure 2.1: Oscillator with a viscoelastic stiffness

For a viscous damping $K(s) = k + cs$, the load to displacement transfer is given by

$$H_{visc}(s) = \frac{1}{ms^2 + cs + k} = \frac{1/m}{(s-\lambda)(s-\bar{\lambda})} \tag{2.3}$$

whose poles (root of the transfer denominator) λ are

$$\begin{array}{ccc} \lambda = -\zeta\omega_n \pm i\omega_d & , & \omega_d = \omega_n\sqrt{1-\zeta^2} \\ \omega_n = \sqrt{k/m} & , & \zeta = \frac{c}{2\sqrt{km}} \end{array}$$

For structural damping $K(s) = k(1+i\eta)$, the load to displacement transfer is given by

$$H_\eta(s) = \frac{1}{ms^2 + k(1+i\eta)} = H_{visc} - \frac{ik\eta - cs}{(ms^2+cs+k)(ms^2+k(1+i\eta))} \tag{2.4}$$

its pole with a positive imaginary part is identical to that of the viscous model for

$$\eta = 2\zeta \tag{2.5}$$

which leads to a difference $H_\eta - H_{visc}$ that is zero at resonance ω_n. The pole with a negative imaginary part is unstable (positive real part) which is a classical limitation of the structural damping model.

Damping is also defined by a *quality factor* which can be measured in a shaking table as the ratio between the acceleration of the mass at resonance and the acceleration of its base. The value is approximately

$$Q = \frac{1}{2\zeta} \tag{2.6}$$

Under the assumption that the strain energy is sufficiently uniform to be represented as a spring, Q^-1 the inverse of the quality factor corresponds to a loss factor.

For a damper following the 3 parameter law of a standard viscoelastic solid (1.8), the system has a pair of complex poles $\lambda, \bar{\lambda}$ and one real pole β

$$H(s) = \frac{1}{ms^2 + k\frac{1+s/z}{1+s/p}} = \frac{1 + s\frac{1+2\zeta\omega/\beta}{\beta}}{m(s^2+2\zeta\omega s+\omega^2)(1+\frac{s}{\beta})} \tag{2.7}$$

and the model characteristics depend on those poles as follows

$$p = \frac{\beta}{1+\frac{2\zeta\omega}{\beta}} \quad z = \frac{\beta}{1+\frac{2\zeta\beta}{\omega}} \quad k = m\frac{\omega^2}{1+\frac{2\zeta\omega}{\beta}} \tag{2.8}$$

For low damping of the conjugate pair of poles (that is $\zeta \ll 1$) and β and ω in the same frequency range, p and z are close which leads to a small maximum loss factor and a response that is very similar to that of the oscillator with viscous damping (2.3).

Figure 2.2 shows that for viscous, structural and viscoelastic dynamic stiffness models for the oscillator (c), the dynamic flexibilities (a-b) are almost exactly overlaid. This is linked to the fact that the real parts of the dynamic stiffness coincide naturally since they are they not (viscous or structural) or little (viscoelastic) influenced by the damping model and the imaginary parts (d) are equal at resonance.

This equivalence principle is the basis for the Modal Strain Energy (MSE) method that will be detailed in section 2.1.3.

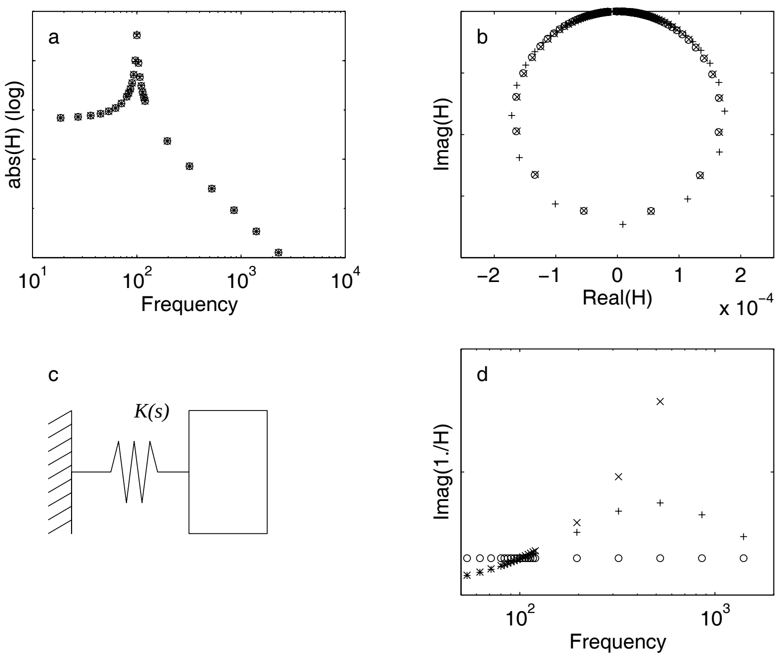


Figure 2.2: Low sensitivity of the dynamic flexibility to the damping model ($\times$ viscous, o structural, $+$ standard viscoelastic)

2.1.2 Real modes and modal damping

For an elastic model, **normal modes** are solution of the eigenvalue problem (see ref [19] for more details)

$$-[M]\{\phi_j\}\omega_j^2 + [K]_{N\times N}\{\phi_j\}_{N\times 1} = \{0\}_{N\times 1} \tag{2.9}$$

associated with elastic properties (sometimes called the underlying conservative problem). They verify two orthogonality conditions in mass

$$\{\phi_j\}^T[M][\phi_k] = \left[{}^{\backslash}\mu_j\delta_{jk}{}_{\backslash}\right] \tag{2.10}$$

and stiffness

$$\{\phi_j\}^T[M][\phi_k] = \left[{}^{\backslash}\mu_j\omega_j^2\delta_{jk}{}_{\backslash}\right] \tag{2.11}$$

There are different standard scaling for normal modes, and one will assumed that they are scale so as to obtain $\mu_j = 1$ which greatly simplifies equation writing. The other standard scaling, often used in experimental modal analysis, sets one DOF (node, direction) of ϕ_j to unity, μ_j is then called the generalized mass at this DOF.

The basis of normal modes is classically used to build reduced model by congruent transformation (2.57) with $\{q\} = [T]\{q_r\} = [\phi_1 ... \phi_{NM}]\{q_r\}$. In the resulting coordinates, called principal coordinates, the mass and stiffness matrices are diagonal ((2.10)-(2.11) conditions). But this is not the case for the viscous and hysteretic damping matrices $T^T C T$ and $T^T D T$.

The **modal damping** assumption (also called **Basile's hypothesis** in French terminology) consists in an approximation of the response where the off-diagonal terms of the damping matrices in principal coordinates are neglected. In practice, one even further restricts the model to viscous damping in principal coordinates since the result can then be integrated in the time domain. On thus has

$$\left[s^2[I] + s\left[\backslash 2\zeta_j\omega_{j\backslash}\right] + \left[\backslash \omega_{j\backslash}^2\right]\right]\{p\}(s) = [\phi]^T\{F(s)\} + \{F_d\} \tag{2.12}$$

where the approximation is linked to the fact of neglecting coupling terms described by

$$\{F_d\} = \left[s\left[\backslash 2\zeta_j\omega_{j\backslash}\right] - \left[\phi^T\right][Cs + D][\phi]\right]\{p\}(s) \tag{2.13}$$

Damping is exactly modal ($F_d = 0$) if the matrices C and/or D are linear combinations of products of M and K [20]

$$[C] = \sum \alpha_{kl}[M]^k[K]^l \tag{2.14}$$

Rayleigh damping [21], typically called **proportional damping**, where

$$[C] = \alpha[M] + \beta[K], \tag{2.15}$$

is thus often used. This is a behavior model, that can be easily adjust to predict correct damping levels for two modes since

$$[\zeta_j] = \alpha\frac{1}{2\omega_j} + \beta\frac{\omega_j}{2} \tag{2.16}$$

Over a wide frequency band, it is however very unrealistic since $\frac{1}{2\omega_j}$ gives high damping ratio for low frequencies ω_j and $\frac{\omega_j}{2}$ for high frequencies. Rayleigh damping is thus really inappropriate to accurate damping studies.

When one starts from a local description of dissipation in the materials or interfaces, modal damping is an approximation whose validity needs to be understood. To do so, one should analyze whether the coupling force (2.13) can be neglected. To validate the domain of validity of this hypothesis, one considers a two DOF system [22, 23]

$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} s^2 + \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} s + \begin{bmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{bmatrix}\right) \begin{Bmatrix} p_1 \\ p_2 \end{Bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \{u(s)\} \tag{2.17}$$

from which one determines an expression of the response of mode 1 given by

$$p_1 = (1+e_1)^{-1} \frac{b_1 u}{\left(s^2 + \gamma_{11} s + \omega_1^2\right)} + e_2 \tag{2.18}$$

with

$$e_1 = \frac{\gamma_{12}\gamma_{21} s^2}{(s^2 + \gamma_{11} s + \omega_1^2)(s^2 + \gamma_{22} s + \omega_2^2)} \tag{2.19}$$

and

$$e_2 = \frac{\gamma_{12} s b_2 u}{(s^2 + \gamma_{11} s + \omega_1^2)(s^2 + \gamma_{22} s + \omega_2^2)} \tag{2.20}$$

The Γ matrix being positive definite, one has $\gamma_{12}\gamma_{21}/(\gamma_{11}\gamma_{22}) < 1$. Thus, the e_1 term can be close to 1 and coupling be significant, if and only if both factors in the denominator are small simultaneously, that is

$$\min(\zeta_1\omega_1, \zeta_2\omega_2)/|\omega_1 - \omega_2| \ll 1 \tag{2.21}$$

Similarly, the e_2 ter, is only significant in special cases of closely spaced modes or such that $b_1 \ll b_2$.

In practice, the validity of the modal damping assumption is thus linked to the frequency separation criterion (2.21) which is more easily verified for small damping levels.

In cases where modal damping is not a good approximation (that is criterion (2.21) is not verified), a generalization of the modal damping model is the use of a viscous damping matrix that is non diagonal in principal coordinates. One the generally talks of non proportional damping since condition (2.14) is not verified. The use of non proportionally damped models will be discussed in section 2.3.4.

For groups of modes that do no verify condition (2.21), one can still use an assumption of modal damping by block [24] where the off diagonal terms of $\phi_j C \phi_k$ are considered for j and k in the same group of modes. Between modes of different groups, the reasoning developed above remains valid and the error induced by neglecting damping coupling terms is small.

2.1.3 Selection of modal damping coefficients

Modal Strain Energy method

The Modal Strain Energy method (MSE [25]) is a classical approximation base on the choice of an equivalent viscous damping coefficient chosen by evaluating the loss factor for a cycle of forced response along a particular real mode shape. As we saw for the particular case of section 2.1.1, this is an appropriate choice since it leads to a near perfect superposition of the transfer function for an isolated mode.

A general approach, that can also be considered for a weakly non linear system, is to compute the ratio, called *loss factor*, of energy dissipated over an enforced motion cycle $q_j(t) = \{\phi_j\}\cos(\omega t)$divided by 2π times the maximum elastic energy associated to that deformation

$$\eta_j(\omega) = \frac{\int_\Omega \int_0^{2\pi/\omega} \sigma(q_j) : \dot{\varepsilon}(q_j) dt}{\pi \{\phi_j\}^T [K] \{\phi_j\}} \tag{2.22}$$

and to impose at each resonance frequency ω_j of the elastic problem, the equality of this loss factor with that of the model with modal damping (2.12)

$$\zeta_j = \frac{\eta_j(\omega_j)}{2} \tag{2.23}$$

For a model where a viscous and/or structural damping model is associated with each component/element (m), the loss factor of mode j is thus obtained as a weighted sum of loss factors in each component

$$\eta_j = \frac{\sum_{(m)} \{\phi_j\}^T [D]^{(m)} \{\phi_j\} + \omega_j \{\phi_j\}^T [C]^{(m)} \{\phi_j\}}{\sum_{(m)} \{\phi_j\}^T [K]^{(m)} \{\phi_j\}} \tag{2.24}$$

For non linearities, there exists classical results of equivalent damping for dry friction [26, 23], dissipation associated with drag in a viscous fluid [26], a plastic spring [23], or small impacts [23].

While the modal strain energy method is typically associated with the modal damping assumption, it can be easily extended to account for frequency dependent non diagonal (one says non-proportional) damping matrices. Thus using $\{q\} = [T]\{q_r\} = [\phi_1 ... \phi_{NM}]\{q_r\}$, leads to a model of the form

$$\left[s^2 [I] + [\phi]^T [Im(Z(s))] [\phi] + \left[\backslash \omega_j^2 \backslash\right]\right] \{p\}(s) = [\phi]^T \{F(s)\} + \{F_d\} \tag{2.25}$$

This is typically referred to as a modal solution. In NASTRAN for example, you will find modal complex eigenvalue (SOL110), frequency response (SOL111), transient (SOL112).

For more general viscoelastic models, it is always possible to define pseudo normal modes solutions of

$$\left[-\omega_j^2 [M] + \mathrm{Re}\left(K(\omega_j)\right)\right] \left\{\tilde{\phi}_j\right\} = \{0\} \tag{2.26}$$

to normalize the using a condition similar to (2.47) and to define an equivalent damping ratio at resonance by

$$\zeta(\tilde{\omega}_j) = \frac{1}{2}\frac{\left\{\tilde{\phi}_j\right\}^T[\mathrm{Im}(K(\omega_j))]\left\{\tilde{\phi}_j\right\}}{\left\{\tilde{\phi}_j\right\}^T[\mathrm{Re}(K(\tilde{\omega}_j))]\left\{\tilde{\phi}_j\right\}} \tag{2.27}$$

Experimental and design damping ratio

The other classical approach is to use modal damping ratio determined experimentally. Identification techniques of experimental modal analysis [27, 28] give methods to determine these ratios.

For correlated modes (when a one to one match between test and analysis is established), one thus uses a damping ratio ζ_{jTest} while typically preserving the analysis frequency.

For uncorrelated modes, one uses values determined as design criteria. $\zeta = 10^{-3}$ for a pure metallic component, 10^{-2} for an assembled metallic structure, a few percent in the medium frequency range or a civil engineering structure. Each industry typically has rules for how to set these values (for example [29] for nuclear plants).

2.2 Viscoelastic models

This section details models of structures used to account more precisely for the constitutive behavior of various viscoelastic materials.

For frequency response computations, section section 2.2.1 shows how the complex dynamic stiffness is built as a weighted sum of constant matrices associated with the various materials.

For eigenvalue computations or time responses on the full model, the introduction of state space models (section 2.2.2) or second order models with internal states (section 2.2.3) allow constant matrix computations. Such formulations could be used for frequency domain solutions but they are higher order and the increase in DOF count limits their usefulness. For the case of fractional derivative models (section 2.2.4), only modeshape computations are accessible.

2.2.1 Frequency domain representation with variable coefficients

For a structure composed of elastic and viscoelastic materials frequency domain computations only require the knowledge of the complex modulus $E_i(s, T, \sigma_0)$ for each material. By using the fact that stiffness matrices depend linearly on the constitutive law coefficients, one can represent the dynamic stiffness of a viscoelastic model as a linear combination of constant matrices (for independent complex moduli in the same material their may be more than one matrix associated to a given material)

$$[Z(E_i,s)] = \left[Ms^2 + K_e + iK_{ei} + sC + \sum_i E_i(s,T,\sigma_0)\frac{K_{vi}(E_0)}{E_0}\right] \tag{2.28}$$

This representation is the basis for the development of solvers adapted for structures with viscoelastic materials.

For frequency domain computations, it is rather inefficient to reassemble Z at each operating point E_i, s. Two solutions can be implemented easily. On can store the various $M, K_e, K_{ei}, C, K_{vi}$ matrices and evaluate the weighted sum (2.28) at each operating point, or store element matrices and reassemble with a weighing coefficient associated with the material property of each element.

2.2.2 State-space representations

One discusses here state space representations associated with analytical representations of the complex modulus discussed in section 1.2.3.

State space models are first order differential equations, assumed here with constant coefficients, with the standard form

$$\begin{aligned}\{\dot{x}(t)\} &= [A]\{x(t)\} + [B]\{u(t)\}\\ \{y(t)\} &= [C]\{x(t)\} + [D]\{u(t)\}\end{aligned} \tag{2.29}$$

The matrices are called : A transfer, B input, C observation and D direct feed-trough. The first equation is the evolution equation while the second is called the observation equation.

The usual technique for time integration of mechanical models is to define a state vector combining displacements and velocities, leading to a model of the form

$$\begin{aligned}\left\{\begin{array}{c}\dot{q}\\ \ddot{q}\end{array}\right\} &= \left[\begin{array}{cc}0 & I\\ -M^{-1}K & -M^{-1}C\end{array}\right]\left\{\begin{array}{c}q\\ \dot{q}\end{array}\right\} + \left[\begin{array}{c}0\\ M^{-1}b\end{array}\right]\{u(t)\}\\ \{y(t)\} &= [c\phi \;\; 0]\left\{\begin{array}{c}q\\ \dot{q}\end{array}\right\}\end{aligned} \tag{2.30}$$

This model is rarely used as such because M is often singular or, for a consistent mass, leads to matrices $M^{-1}K$ and $M^{-1}C$ that are full. When using generalized coordinates, one can however impose a unit mass matrix and this easily use the model form. Some authors [30] also prefer this form to define eigenvalue problems but never build the matrices explicitly so that the full matrices are not built.

The other standard representation is the generalized state-space model

$$\begin{bmatrix} C & M \\ M & 0 \end{bmatrix} \begin{Bmatrix} \dot{q} \\ \ddot{q} \end{Bmatrix} + \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix} \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \{u\}$$
$$\{y\} = \begin{bmatrix} c & 0 \end{bmatrix} \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix} \tag{2.31}$$

which preserves symmetry and allow the definition of simple orthogonality conditions on complex modes.

One will now introduced generalized state space models for the case of viscoelastic constitutive laws. A rational fraction, that does not go to infinity at high frequencies and having distinct poles, can be represented by a sum of first order rational fractions

$$E(s) = K^0 \prod_{i=1}^{N} \frac{1 - {}^{s}/_{z^i}}{1 - {}^{s}/_{p^i}} = E^\infty \left(1 + \sum_{i=1}^{N} \frac{-g^i}{{}^{s}/_{\omega^i} + 1} \right) = E^\infty \left(g^0 + \sum_{i=1}^{N} g^i \frac{s}{s + \omega^i} \right). \tag{2.32}$$

By introducing the intermediate (relaxation) field $q_v^i = -\frac{E^\infty g^i}{(s+\omega^i)} q$, one can rewrite (2.28) as a state-space model of higher size

$$\left[\begin{bmatrix} M & 0 & \dots & 0 \\ 0 & M & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & M \end{bmatrix} s + \right.$$
$$\left. \begin{bmatrix} 0 & -M & 0 & 0 & 0 \\ K_e + E_\infty \sum K_v^i & 0 & K_{v1} & \dots & K_{vn} \\ g^1 E^\infty M & 0 & \omega_1 M & \dots & 0 \\ \dots & & & & \\ g^n E^\infty M & 0 & 0 & \dots & \omega_n M \end{bmatrix} \right] \begin{Bmatrix} q \\ sq \\ q_{v1} \\ \vdots \\ q_{vn} \end{Bmatrix} = \begin{Bmatrix} 0 \\ F \\ 0 \\ 0 \end{Bmatrix} \tag{2.33}$$

One saw in the previous chapter that real constitutive laws could be represented with rational fractions of relatively high order. This representation is practical only for reduced models ($M_R = T^T M T$, ...) where the strategy for the selection of reduction basis T will be detailed in section **??**.

In practice for compatibility with non-linear hyperelastic behavior, it is preferable to use a stress rate relaxation equation, where the evolution equation is given by

$$\dot{F}_g^i + \omega^i F^i = \frac{g^i}{g^0} \dot{F}^0(u_g) = \frac{g^i}{g^0} \left. \frac{\partial F^0}{\partial u} \right|_{u_g} \dot{u}_g \tag{2.34}$$

which, in linearized matrix form, using a rank N_v decomposition of $K_v = [B_v][B_v]^T$, the viscous force given by $F_v^i = [B_v]\, q_v^i$ and the viscous states q_v^i evolution equation

$$\{\dot{q}_v^i\} + \omega^i \{q_v^i\} = \frac{g^i}{g^0}\left[B_v^T\right]\{\dot{q}\} \tag{2.35}$$

Assuming a standard mass normalized reduction basis, so that for the chosen states the mass is equal to identity, the resulting state space model is

$$s\,[I] + \begin{bmatrix} 0 & -I & \dots & 0 \\ K_e + K_v^0 & 0 & \dots & B_v^i \\ \vdots & \vdots & & \\ 0 & \frac{g^i}{g^0} B_v^{iT} & \dots & \omega^i \end{bmatrix} \begin{Bmatrix} q \\ sq \\ \vdots \\ q_v^i \end{Bmatrix} = \begin{Bmatrix} 0 \\ F \\ 0 \\ 0 \end{Bmatrix} \tag{2.36}$$

which has $2N_q + N_{cell} \times N_v$ states. Building of such state-space models is implemented in `nl_solve Reduc2ss` which requires as `SDT-nlmodal` token.

For time domain representations, `SDT-nlsim` (see more details in see `sdtweb nlfu#uMaxw`) allows the use of stress or strain relaxation to provide adapted non-linear coupling.

2.2.3 Second order models with internal states

If the state space form is more compact a priori, the operators available in a given FEM code may make its manipulation more difficult. A classical solution is thus to build a second order model of the form usual in mechanics and thus easily manipulated with a mechanically oriented code. The implementation of internal states, however requires the definition of multiple fields at the same node which is not easily implemented in all software packages.

The *Anelastic Displacement field* [31]) method considers a modulus representation of the form (2.32), which leads to a model of the form

$$\begin{aligned} &\left[s^2 \begin{bmatrix} M & 0 & \dots \\ 0 & 0 & \\ \vdots & & \ddots \end{bmatrix} + s \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \frac{K_{v1}}{E_1} & & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \frac{K_{vn}}{E_n} \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} K_e - E_\infty \sum K_{vi} & K_{v1} & \dots & K_{vn} \\ K_v 1 & \frac{\omega_1}{E_1} K_{v1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \frac{\omega_n}{E_n} K_{vn} \end{bmatrix} \right] \begin{Bmatrix} q \\ q_{v1} \\ \vdots \\ q_{vn} \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \end{Bmatrix} \end{aligned} \tag{2.37}$$

The absence of a mass associated with internal states q_{vi} can lead to problems with certain solvers. An alternative is the GHM [32] method which represents the modulus as

$$E(s) = E_\infty \left(1 + \sum_{j=1}^{n} \frac{\alpha^i}{s^2 + 2\zeta^i \omega^i s + (\omega^i)^2} \right) \tag{2.38}$$

and defines internal states by $q_{vj} = \frac{\alpha^i}{s^2+2\zeta^i\omega^i s+(\omega^i)^2} q$. One will note that not all rational fractions can be represented in the form (2.38).

The introduction of q_{vi} fields in the previous section corresponds to the classical thermodynamics theory of materials with augmented potential including internal states. The first row in equation (2.37) indeed corresponds to the representation of stress in viscoelastic materials in the form

$$\sigma = E_\infty \varepsilon(q) + E_\infty \sum_i \varepsilon(q_{vi}) \tag{2.39}$$

In practice, the q_v are only non zero for viscoelastic elements. The direct use of matrices assembled following (2.37) must thus be done with solvers capable of eliminating unused DOFs. When using reduced models, the problem does not normally occur since reduced basis vectors are typically non zero over the whole structure and thus lead to none zero internal states.

In the time domain, this formalism is more easily dealt with because the time evolution of internal states can easily be computed by time integration of the relation between the q_{vi} and q. For a model of form (2.32), the evolution of the internal state is thus given by

$$\{\dot{q}_{vi}\} = -\omega^i \{q_{vi}\} - E^i \{q\} \tag{2.40}$$

which can be easily integrated (this approach is used in ABAQUS [33] for example). This formalism corresponds to the separate treatment of bloc rows in (2.33) or (2.37) which is simple in the time domain but leads to a non-linear problem in the frequency domain, unless operators are defined implicitly as in Ref [30].

2.2.4 Fractional derivatives

The internal state formalism can also be used to represent fractional derivative constitutive laws if one uses non integer but rational derivatives. For a common denominator p, one will use a modulus of the form

$$E(s) = E_{max} - \sum_{k=1}^{p} \frac{E_k}{s^{k/p} + \omega_k} \tag{2.41}$$

and build a state space model of the form

$$\{x(s)\}\, s^{1/p} = [A]\,\{x(s)\} + [B]\,\{u(s)\} \tag{2.42}$$

where the state vector will combine fractional derivatives of the displacement $s^{k/p}q$ where $k = 1 : 2p-1$ and of the internal state $q_{vk} = -\frac{E_k}{(s^{k/p}+\omega_k)}q$ (see [34] for example).

In practice, the number of blocs in the state vector being proportional to p, constant matrix representations are thus limited to small values of p. This limits practical uses of fractional derivative models to frequency domain response and non-linear eigenvalue computations (see section 2.3.1).

2.3 Spectral decomposition and reduced models

For an input $[b]\,\{u(s)\}$ characterized by the frequency domain characteristics of $u(s)$ and spatial content of b, component mode synthesis and substructuring methods provide approximations of the solution of problems (2.2), (2.33), or (2.37).

For damped problems, one should distinguish

- exact spectral decompositions using complex modes as treated in sections 2.3.1 and 2.3.2;
- model reduction methods which only seek to approximate the transfer spectrum by projecting the model on bases built using solutions of problems that are simpler to solve than the complex eigenvalue problem. These approaches are detailed in sections **??**, 2.3.4 and 2.3.5.

2.3.1 Complex modes of analytical models

All the problems that where introduced earlier can, in the frequency domain, be represented as frequency response computations of the form

$$\{y(s)\} = [H(s)]\,\{u(s)\} = [c]\,\{q(s)\} = [c]\,[Z(s)]^{-1}\,[b]\,\{u(s)\} \tag{2.43}$$

which involve the inverse of the dynamic stiffness $Z(s)$.

In the very general case where complex moduli are supposed to be analytic functions in the complex plane (locally regular), the dynamic stiffness $Z(s)$ is also an analytic function. Observation and input matrices c and b being constant, the poles (i.e. singularities) of $H(s)$ correspond to non zero solutions of

$$[Z(\lambda_j)]\,\{\psi_{jD}\} = \{0\} \quad \textit{and} \quad \{\psi_{jG}\}^T\,[Z(\lambda_j)] = \{0\} \tag{2.44}$$

which defines the generalized non linear eigenvalue problem associated with a viscoelastic model.

Near a given pole, analytic functions have a unique Laurent's development

$$f(z) = \sum_{k=-\infty}^{+\infty} \frac{a_k}{(s-\lambda)^k} \quad avec \quad a_k = \frac{1}{2\pi i}\int_\gamma f(s)(s-\lambda)^{k+1} ds \tag{2.45}$$

where γ is a arbitrary closed direct contour of the singularity λ.

As a result, near an isolated pole λ_j one has

$$[Z(s)]^{-1} = \frac{\{\psi_{jD}\}\{\psi_{jG}\}^T}{\alpha_j(s-\lambda_j)} + O(1) \tag{2.46}$$

where the normalization coefficient α_j depends on the choice of a norm when solving (2.44) and is determined by

$$\alpha_j = \{\psi_{jG}\}^T \left[\frac{\partial\left[Z(s)\right]}{\partial s}\right] \{\psi_{jD}\} \tag{2.47}$$

To simplify writing, it is desirable to use $\alpha_j = 1$ which si the usual scaling for constant matrix eigenvalue problems described in the next section.

Having determined the set of poles in a given frequency band, having normalized the associated modes so that $\alpha_j = 1$, one obtains a first order development in s

$$H(s) = [c]\,[Z(s)]^{-1}\,[b] = \sum_j \frac{s}{\lambda_j}\frac{\{c\psi_{jD}\}\left\{\psi_{jG}^T b\right\}}{s-\lambda_j} + [c]\,[Z(0)]^{-1}\,[b] \tag{2.48}$$

where the $[c]\,[Z(0)]^{-1}\,[b]$ terms correspond to the exact static response to loads associated with the input shape matrix b. This static correction term is well known in component mode synthesis applications and is analyzed in section **??**.

It is often useful to consider a representation using residual flexibility

$$H(s) = [c]\,[Z(s)]^{-1}\,[b] = \sum_{j=1}^{2NM} \frac{\{c\psi_{jD}\}\left\{\psi_{jG}^T b\right\}}{s-\lambda_j} + \left([c]\,[Z(0)]^{-1}\,[b] - \sum_{j=1}^{2NM}\frac{c\psi_{jD}\psi_{jG}^T b}{-\lambda_j}\right) \tag{2.49}$$

2.3.2 Complex mode eigenvalue problems with constant matrices

The solution of the non linear eigenvalue problem (2.44) is difficult (see section 2.3.1). Solution algorithms are thus greatly simplified by restating the problem as a classical first order eigenvalue problem with constant matrices.

For models with viscous and structural damping (2.2) or viscoelastic models of form (2.37), on thus generally solves the eigenvalue problem associated with (2.31), that is

$$\left(\begin{bmatrix} C & M \\ M & 0 \end{bmatrix}\lambda_j + \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}\right)\{\theta_j\} = \{0\} \tag{2.50}$$

Because of the block form of this problem, one can show that

$$\{\theta_j\} = \left\{ \begin{array}{c} \psi_j \\ \psi_j\lambda_j \end{array} \right\} \tag{2.51}$$

and one thus gives the name *complex mode* both to θ_j and ψ_j.

The existence of $2N$ eigenvectors that diagonalize the matrices of (2.50) is equivalent to the verification of two orthonormality conditions

$$\begin{array}{lll} [\theta]^T \begin{bmatrix} C & M \\ M & 0 \end{bmatrix} [\theta] & = \psi^T C\psi + \Lambda\psi^T M\psi + \psi^T M\psi\Lambda & = \left[\backslash I \backslash\right]_{2N} \\ [\theta]^T \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix} [\theta] & = \psi^T K\psi - \Lambda\psi^T M\psi\Lambda & = -\left[\backslash \Lambda \backslash\right]_{2N} \end{array} \tag{2.52}$$

For a model represented in is state space form, such as (2.33), one solves the left and right eigenvalue problems

$$([E]\,\lambda_j + [A])\,\{\theta_{jD}\} = \{0\} \quad \text{et} \quad \{\theta_{jG}\}^T\,([E]\,\lambda_j + [A]) = \{0\} \tag{2.53}$$

and uses standard orthonormality conditions

$$\{\psi_{jG}\}\,[E]\,\{\theta_{kD}\} = \delta_{jk} \quad \text{et} \quad \{\psi_{jG}\}\,[A]\,\{\theta_{kD}\} = -\lambda_j\delta_{jk} \tag{2.54}$$

In the complex plane, one should distinguish complex poles associated with vibration modes and real poles which can be related with a damping ratio above 1 (supercritical damping that is not common) or to material relaxation (be linked to poles of the complex modulus). To understand this distinction one considers the viscoelastic oscillator (2.8). The viscoelastic behavior leads to one real pole β, whereas supercritical damping corresponds to $\zeta > 1$ and leads to two real poles

$$\lambda = -\zeta\omega \pm \omega\sqrt{\zeta^2 - 1} \tag{2.55}$$

In a viscoelastic computation where the constitutive law contains real poles the root locus of the solution is similar to that shown in figure 2.3. One must thus distinguish the classical spectrum of vibration modes and the real poles associated with material relaxation

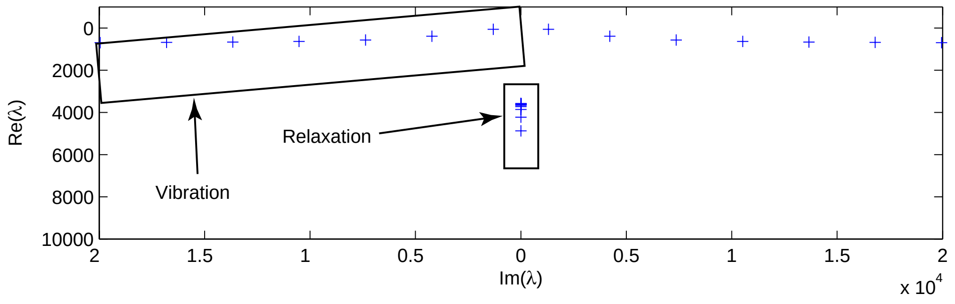


Figure 2.3: Poles of a viscoelastic beam in traction

It is important to note that the number of real poles is directly associated with the number of DOFs in the internal states q_v. The number of these poles will the increase with mesh refinement. In practice, one thus cannot expect to compute the real poles and associated modes in the area of relaxation poles. This selection of convergence area is the aspect that needs to be accounted for in the development of partial eigenvalue solvers for damped problems.

A second consequence of the increase in the number of real poles is the potential impossibility to compute modes with supercritical damping. The author's experience is that this limitation is mostly theoretical since in practice modes with supercritical damping are rare.

The complex modulus, being the Fourier transform of a real valued relaxation function, should be symmetric in frequency ($E(-\omega) = (E(\omega)^*)$. For a modulus representation that does not verify this hypothesis, as is the case of structural damping, only poles with positive imaginary parts have a meaning. For response synthesis, one will thus take the conjugates λ_j^*, ψ_j^* of modes computed with positive imaginary parts.

The computation of complex modes can be used to approach transfer functions using the developments (2.46) and (2.48). Orthogonality conditions given above correspond to the $\alpha_j = 1$ normalization.

2.3.3 Model reduction methods

To simulate the dynamic response it is not useful and rarely possible from a numerical cost standpoint to use the full model (2.2) for direct time simulations (see section **??**). Model reduction methods (modal analysis, substructuring, component mode synthesis, ...) seek an approximate solution within a restricted subspace. One thus assumes

$$\{q\}_{N\times 1} = [T]_{N\times NR} \{q_R\}_{NR\times 1} \tag{2.56}$$

and seek solution of (2.1) whose projection on the one the dual subspace T^T is zero (this congruent transformation corresponds to a Ritz-Galerkin analysis). Transfer functions are the approximated by

$$[H(s)] = [c]\,(Z(s))^{-1}\,[b] \approx [cT]\,\left[T^T Z(s) T\right]^{-1}\left[T^T b\right] \tag{2.57}$$

One can note that for a non-singular transformation T (when $\{q\} = [T]\,\{q_R\}$ is bijective) the input u / output y relation is preserved. One says that the transfer functions are objective quantities (they are physical quantities that are uniquely defined) while DOFs q are generally not objective.

Classical bases used for model reduction combine modes and static responses to characteristic loads [35]. One distinguishes

- bases containing free modes and static responses to applied loads $\{b\}$

$$[T] = \left[[\phi_1 \ldots \phi_{NM}] \quad \left[[K]^{-1}[b] - \sum_{j=1}^{NR} \frac{[c]\,\{\phi_j\}\,\{\phi_j\}^T\,[b]}{\omega_j^2}\right]\right] \tag{2.58}$$

  For component mode synthesis (component model reduction to prior to a coupled system prediction), free modes have been used by [36], MacNeal [37], and many others.

- bases containing static displacements associated with displacement enforced on an interface and fixed interface modes

$$[T] = \left[\begin{bmatrix} 0 \\ \phi_{1:NM,c} \end{bmatrix} \quad \begin{bmatrix} I \\ K_{cc}^{-1} K_{ci} \end{bmatrix}\right] \tag{2.59}$$

  For CMS, the use of static terms only is called Guyan condensation [38]. Adding fixed interface modes leads to the Craig Bampton method [39].

- damped modes can be considered as elastic models with an external damping load. Static correction for the effects of damping loads can then be incorporated. The first order correction proposed in [40] (see also [41])

$$[T] = \left[[T_0] \quad \left[[K]_0^{-1}\,[K_{vi}\,[\phi_{1:NM}]]\right]\right] \tag{2.60}$$

- multi-model reduction where modal bases for multiple design points are considered. In particular, the typical approach used by `fe2xf Build` is to combine low and high modulus computations.

2.3.4 Equivalent viscous damping

Section 2.1.2 showed that the modal damping assumption was justified under the hypothesis of modal separation (2.21). When dealing with a real basis, it is often possible to compute an equivalent viscous damping model approximating the damped response with good accuracy.

To build this equivalence, one distinguishes in reduction bases (2.58), (2.59), or other, blocs T_m associated with modes whose resonance is within the frequency band of interest and T_r associated with residual flexibility terms. As shown in figure 2.4, one is interested in the modal contribution of the first while for the others only the asymptotic contribution counts.

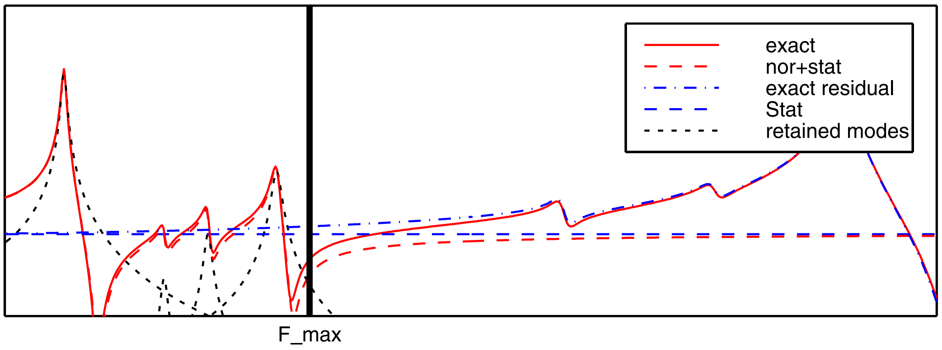


Figure 2.4: Transfer function decomposition into modal contributions and residual terms

For a real basis reduction, one is thus interested in an approximation of the form

$$[H(s)] \approx [cT] \begin{bmatrix} T_m^T Z(s) T_m & T_m^T Z(s) T_r \\ T_r^T Z(s) T_m & T_r^T Z(s) T_r \end{bmatrix}^{-1} [T^T b] \tag{2.61}$$

The objective in distinguishing T_m and T_r is to guarantee that $T_r^T Z(s) T_r$ does not present singularities within the band of interest (it represents residual flexibility not resonances).

To validate this hypotheses, one defines a reference elastic problem characterized by a stiffness K_0 (one will use $K_0 = \mathrm{Re}(Z(\omega_0)) + M\omega_0^2$ avec ω_0 with typically better results for a higher frequency value).

By computing modes in a subspace generated by T_r of the representative elastic problem

$$\left[T_r^T K_0 T_r - \omega_{jr}^2 T_r^T M T_r\right] \phi_{jr} = 0 \tag{2.62}$$

one can verify this uncoupling: if ω_{1r} is within the band of interest the decoupling is not verified.

One can always choose bases T_m and T_r so as to diagonalize the reference problem. Furthermore, by writing $\Delta Z(s) = Z(s) - K - s^2 M$, the transfer function is approximated by

$$[H(s)] \approx [cT] \begin{bmatrix} \left[\backslash s^2 + \omega_{jm \backslash}\right] + T_m^T \Delta Z(s) T_m & T_m^T Z(s) T_r \\ T_r^T Z(s) T_m & \left[\backslash s^2 + \omega_{jr \backslash}\right] + T_r^T \Delta Z(s) T_r \end{bmatrix}^{-1} \left[T^T b\right] \tag{2.63}$$

The Modal Strain Energy method (MSE) with damping ratio given by (2.27), corresponds to the following approximation

$$T_m^T \Delta Z(s) T_m \approx s\,[\Gamma] = s \left[\backslash 2\zeta_j \omega_{j \backslash}\right] \tag{2.64}$$

One can easily generalize this approximation by building an equivalent viscous damping matrix by enforcing

$$\left(T_m^T \Delta Z(i\omega) T_m\right)_{jk} = i\omega \Gamma_{jk} \tag{2.65}$$

for a characteristic frequency ω_{jr} for diagonal terms $(j = k)$ $(\omega_{jr} + \omega_{kr})/2$ otherwise. For a viscous damping model, it is a simple projection (computation of $T_m^T C T_m$). For a structural damping model, there is a degree of approximation.

The validity of this approximation is discussed in [42] where it is shown that building the equivalence in generalized coordinates and using term by term characteristic frequencies is efficient.

For coupling terms $T_m^T Z(s) T_r$, damping only has low influence (see the discussion on non proportional damping in section 2.1.2) and can thus be neglected.

For residual terms $T_r^T \Delta Z(s) T_r$ damping can be neglected for frequency analyses. For transient analyses, the presence of high frequency undamped modes (linked to the ω_{jr}) induces non physical oscillations since these modes are introduced to approximate low frequency contributions and not high frequency resonances. It is thus good practice to introduce a significant modal damping for residual modes. For example one uses $\zeta = 1/\sqrt{(2)}$, which leads to assume

$$T_r^T \Delta Z(i\omega) T_r \approx s \left[\backslash \sqrt{2} \omega_{jr \backslash}\right] \tag{2.66}$$

2.3.5 Case of viscoelastic models

The reduction is also applicable for viscoelastic models detailed in section 2.2. Indeed all matrices used in the formulation of the dynamic stiffness (2.28) can be projected. State-space or second order representations can be generated by replacing each matrix by its reduced version M by $T^T M T$, etc. This reduction form will be used for eigenvalue solvers discussed in section **??** .

The equivalent viscous damping model building strategy detailed in the previous section cannot be generalized since the real part of $T_m^T \Delta Z(s) T_m$ undergoes significant variations as the storage modulus changes with frequency. In other terms, the ω_{jm} associated with K_0 differ, possibly significantly, for frequencies of the non linear eigenvalue problem $\left[\mathrm{Re}\left(K(\omega_j)\right) - \omega_j^2 M\right]\{\phi_j\} = 0$. For modal synthesis or transient response computations, one will thus prefer representations associated with the spectral decomposition (2.49) as detailed in section section **??**.

2.4 Meshing of sandwich models

Two main strategies have been considered to model sandwich structures: building higher order shell models [43] or connecting multiple elements. The main problem with the higher order element approach is that developing good shell elements is very difficult so that most developments for sandwiches will not perform as well as state of the art shell elements. The multiple element strategy is also the only available for immediate implementation into industrial FEM software.

To properly account for shear effects in the viscoelastic layer, the offsets between the neutral fiber and the shell surface are most of the time essential. Rather than defining offsets for shell elements [44], rigid links between the shell nodes and the volume element are used here as shown in figure 2.5. Although this generates additional nodes (4 node layers for a single constrained layer model), this strategy accommodates all possible layer configurations. During resolution, the model is smaller since all viscoelastic volume nodes are constrained.

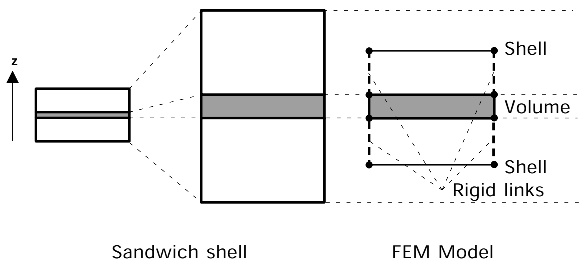


Figure 2.5: Shell/volume/shell model for sandwiches

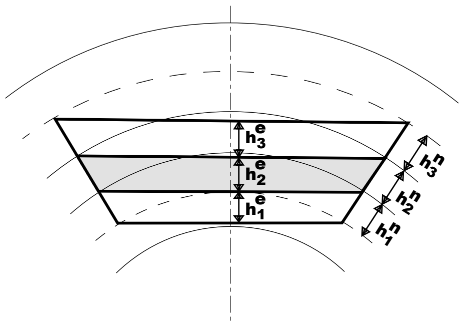


Figure 2.6: Problems with thickness definitions in shells with significant curvature

Automated layer mesh generation from a selected area of a nominal shell model is a basic need (supported by the `fevisco`MakeSandwich commands). Figure 2.6 illustrates the fact that for curved shells, the use of flat elements generates a distinction between layer thicknesses along the element normal h_i^e or along the normal at nodes h_i^n. This distinction is important for relatively coarse meshes of press formed parts (as the floor panel of figure **??**). Advanced options meshing options, let you preserve thickness either at element center or nodes and possibly control the normal map used as a meshing support.

For stiff layers, shells are preferred over volumes, because volume element formulations are sensitive to shear locking when considering high aspect ratio (dimensions of the element large compared to thickness).

For soft layers, the use of a volume element both necessary, because shell elements will typically not correctly represent high shear through the thickness, and acceptable, because almost all their energy is associated with shear so that they will not lock in bending [40]. Note that shear corrections used in some FEM codes to allow bending representation with volumes may have to be turned off to obtain appropriate results. Finally there are doubts on how to properly model the through the layer compression stiffness of a very thin viscoelastic layer (this can have significant effects on curved layers).

The demo `basic_sandwich` generates curves for the validation of shell/volume/shell model used to represent constrained layer treatments as first discussed in [40]. The idea is to vary the properties of a central volume layer between a very soft modulus and the skin modulus.

For a very soft value, figure 2.7 shows convergence to the asymptotic value of a single skin plate. For a modulus of the viscoelastic core equal to that of the skins, one should converge to the frequency of a plate model with thickness equal to the sum of skins + viscoelastic core. Figure 2.7 shows that the high modulus asymptote is slightly higher for the shell / volume / shell model. This is due to shear locking in the very thin volume layer (well known and documented problem that low order volumes cannot represent bending properly). This difficulty can be limited by using volume element with shear locking protection. The figure also shows that damping is optimal somewhere between the low and high modulus values.

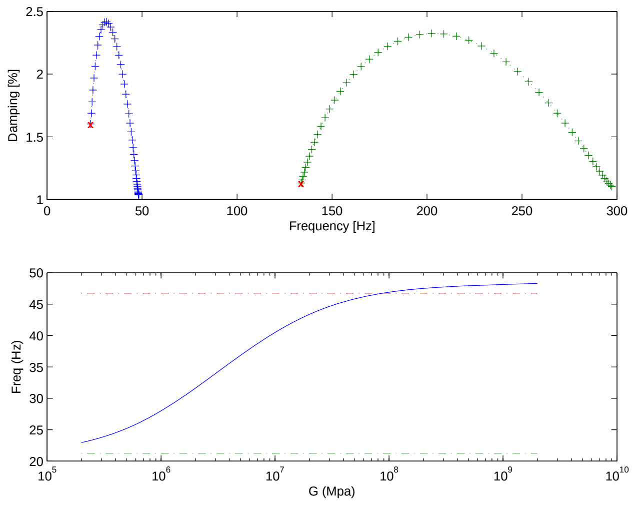


Figure 2.7: Constrained layer model validity.

Figure 2.8 illustrates the validity of a shell/volume model as compared to a single shell based on composite shell theory. Figure **??** illustrates that the results are nearly identical, provided that volume elements with proper shear locking protection are used. For a standard isoparametric volume, a shell/volume model tends to be be to stiff (shear locking associated with bending).

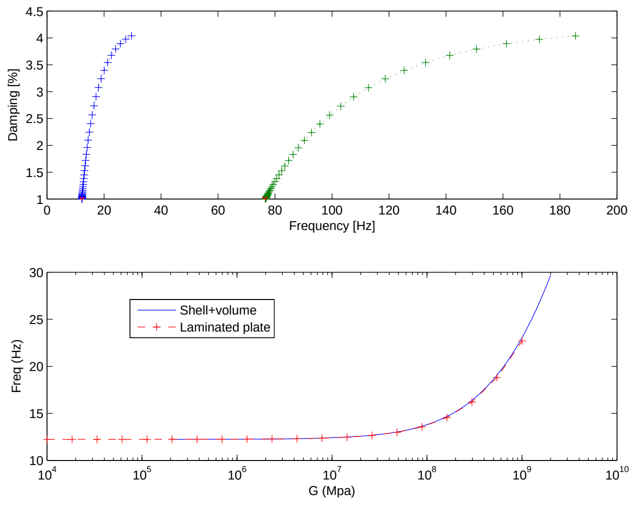


Figure 2.8: Free layer model validity.

The element degree does not seem critical to obtain accurate predictions of the response. The use of multiple elements through the viscoelastic layers has also been considered by some authors but the motivation for doing so is not understood.

For press formed sandwiches, there are further unknowns in how the forming process affects the core thickness and material properties. In particular, most materials used for their high damping properties are also very sensitive to static pre-stress. For a simple folded plate, figure 2.9 illustrates how the modal frequencies and energy distribution in the viscoelastic layer are modified if the shear modulus is multiplied by 10 in the fold. Such behavior was found in tests and motivated the study in Ref. [15], where the effect of static pre-stress is measured experimentally. Overall, predicting the effects press forming or folding sandwiches is still a very open issue.

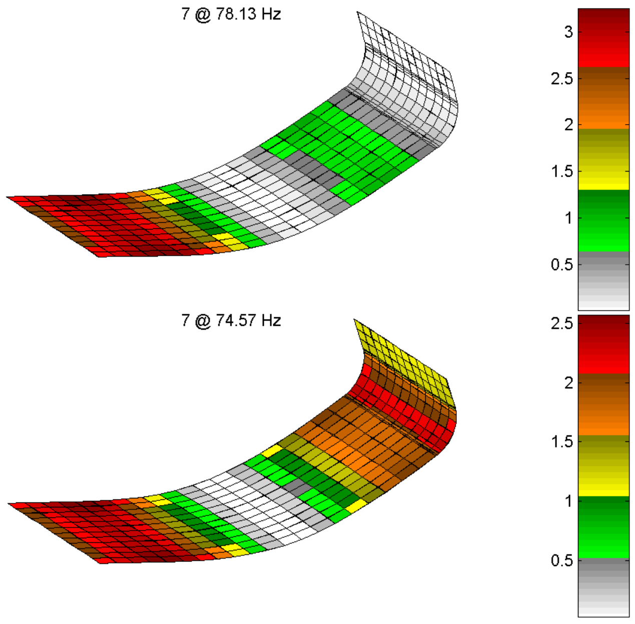


Figure 2.9: Energy density in the viscoelastic layer of a simple folded sandwich plate. (Top) High stiffness viscoelastic in the fold. (Bottom) equal stiffness in the fold and elsewhere.

A final difficulty is to deal properly with boundary conditions of the skin layers. Since differential motion of the skins plays a major role in the effectiveness of the core, the boundary conditions of each layer has to be considered separately. This is easily illustrated by the generation of cuts in constraining layers [45, 44] (and `cut_optim` demo).

2.4.1 Mesh convergence and non conformity

As illustrated in figure **??** the dissipation if often localized on a fairly small sub-part of the structure. It is thus quite important to validate the accuracy of predictions obtained with various mesh refinements. Figure 2.10 illustrates a convergence study where a constrained layer damping treatment is refined and one compares the strain energy density maps for two levels of refinement. The strain energy maps, clearly indicate edge effects, which are typical of constrained layer treatments. In such studies the author's have usually found, that the distribution of constraints is well predicted and energy fractions (strain energy in the viscoelastic compared to total strain energy in the model) predicted with the fine and coarse meshes do not show significant differences.

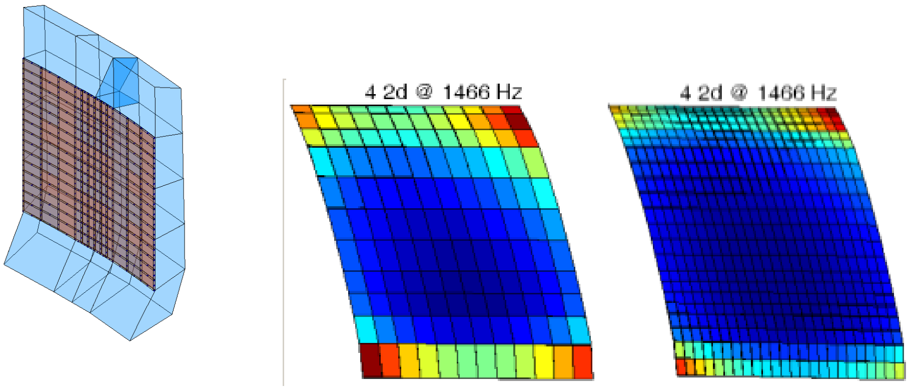


Figure 2.10: Zoom on the refined mesh of a constrained layer damping treatment placed on a volume model. Comparisons of strain energy maps for two levels of refinement.

When considering free placement of damping devices (see section **??**), one is rapidly faced with the problem of incompatible meshes. For discrete connections, where loads are transmitted at isolated points with at most one point on a given element of the supporting structure, the problem is very much related to that of the representation of weld spots and strategies that use the underlying shape functions are most effective (see SDT `feutilb` `MpcFromMatch` command).

2.5 Thermal considerations

In this section one addresses models needed to evaluate the steady state temperature field for a forced harmonic response.

2.5.1 Thermal model

xxx detail xxx

Exchange on free structure interfaces.

Exchange at the internal viscoelastic/metal interfaces.

2.5.2 Heat source due to viscoelastic behavior

Assuming a forced harmonic response, one has

$$\epsilon\left(X,t\right)=\Re\left(\epsilon\left(X,\omega\right)e^{i\omega t}\right)\quad\sigma\left(X,t\right)=\Re\left(\sigma\left(X,\omega\right)e^{i\omega t}\right)\tag{2.67}$$

where stress is related to strain through the complex modulus

$$\sigma\left(X,\omega\right)=\Lambda\left(X,\omega\right)\epsilon\left(X,\omega\right)\tag{2.68}$$

or in the time domain

$$\sigma\left(X,t\right)=\Re\left(\Lambda\left(\omega\right)\epsilon\left(\omega\right)e^{i\omega t}\right)\tag{2.69}$$

Integrating the power dissipated over a period of the forced response, one obtains the spatial distribution of dissipated power

$$\begin{aligned}p(X,\omega)\quad&=\int_0^{\frac{2\pi}{\omega}}\sigma\left(X,t\right)\dot{\epsilon}\left(X,t\right)dt\\&=\int_0^{\frac{2\pi}{\omega}}\Re\left(\Lambda\left(\omega\right)\epsilon\left(\omega\right)e^{i\omega t}\right)^T\Re\left(i\omega\epsilon\left(X,\omega\right)e^{i\omega t}\right)\\&=\pi\left(\Im(\epsilon)^T\Im(\Lambda)\Im(\epsilon)+\Re(\epsilon)^T\Im(\Lambda)\Re(\epsilon)\right)\end{aligned}\tag{2.70}$$

Note that the maximum strain energy during the cycle is given by

$$\begin{aligned}e(X,\omega)\quad&=max_0^{\frac{2\pi}{\omega}}\left(\epsilon(X,t)^T\Re(\Lambda)\epsilon(X,t)\right)\\&=\Im(\epsilon)^T\Re(\Lambda)\Im(\epsilon)+\Re(\epsilon)^T\Re(\Lambda)\Re(\epsilon)\end{aligned}\tag{2.71}$$

which gives a simple way to estimate the loss factor (1.3) associated with the local stress/strain state.

Despite the local nature of dissipation, one may want to verify that the integral over the volume of the dissipated energy is equal to the input power. Noting $H(\omega)$ the transfer collocated with the input, the equivalent power input in the structure is given by

$$\begin{aligned} p(\omega) = & \int_0^{\frac{2\pi}{\omega}} \Re(F(\omega)e^{i\omega t})\Re(i\omega H(\omega)F(\omega)e^{i\omega t}) = -\pi\Im(H(\omega)) \\ = & \int_0^{\frac{2\pi}{\omega}} \left(q(t)^T \Im(K) q(t)\right) \\ = & \ \pi \left(\Im(q(\omega))^T \Im(K) \Im(q(\omega)) + \Re(q(\omega))^T \Im(K) \Re(q(\omega))\right) \end{aligned} \tag{2.72}$$

In general, there is no direct way to measure the maximum strain energy in a structure and thus no experimental definition of a global or system loss factor.

For models, the energy can be computed but there is not particular reason for the local energy (2.73) to reach a maximum value at all points simultaneously. One can thus search for the maximum strain energy in the system using

$$\begin{aligned} e(q(\omega)) & = max_0^{\frac{2\pi}{\omega}} \left(q(t)^T \Re(K) q(t)\right) \\ & = \Im(q(\omega))^T \Re(K) \Im(q(\omega)) + \Re(q(\omega))^T \Re(K) \Re(q(\omega)) \end{aligned} \tag{2.73}$$

In the case of normal modes, the strain energy is equal to the square of the mode pulsation so that the MSE methods can be meaningful (see section 2.1.3).

An illustration of global loss factor use for uniform and non-uniform material loss can be found in in `t_visco('ThermoPower')`

2.5.3 Cantilever plate example

To perform coupled thermoelastic modes computation, 3 commands are available in `comp12` :

- `Thermo1Assemble` assembles structure matrices, thermal matrices and coupled thermoelastic matrices.
- `Thermo1Build` builds MVR reduced model.
- `Thermo1Modes` computes associated complex modes.

First a finite element model (`mdl` must be created, with element associated to structure properties (`m_elastic` and `p_solid`). Thermal properties (`m_heat` and `p_heat`) must be present in model material and element property stack.

Matrices must be assembled as following:
`RO=comp12('Thermo1Assemble',mdl,RO);`
`RO` is the coupling data structure with fields

- `.MatId [Matid_struct MatId_therm]`, first is the `MatId` of the structure material property, and second the thermal one.
- `.ProId [ProId_struct ProId_therm T0 Matid_struct MatId_therm]`. `T0` is the thermal coupling (C, Inf if no coupling).

Then reduced MVR can be build using:
`MVR=comp12('Thermo1Build-reduce',mdl,RO);` If command option `-reduce` is given model is reduced according to real modeshapes. Then eig options must be given in `RO.EigOpt`.

Complex coupled thermoelastic modes can then be computed using
`def=comp12('Thermo1Modes',MVR);`

2.5.4 Thermo-elastic damping

To perform coupled thermoelastic modes computation, 3 commands are available in `comp12` :

- `Thermo1Assemble` assembles structure matrices, thermal matrices and coupled thermoelastic matrices.
- `Thermo1Build` builds MVR reduced model.
- `Thermo1Modes` computes associated complex modes.

First a finite element model (`mdl` must be created, with element associated to structure properties (`m_elastic` and `p_solid`). Thermal properties (`m_heat` and `p_heat`) must be present in model material and element property stack.

Matrices must be assembled as following:
`RO=comp12('Thermo1Assemble',mdl,RO);`
`RO` is the coupling data structure with fields

- `.MatId [Matid_struct MatId_therm]`, first is the `MatId` of the structure material property, and second the thermal one.
- `.ProId [ProId_struct ProId_therm T0 Matid_struct MatId_therm]`. `T0` is the thermal coupling (C, Inf if no coupling).

Then reduced MVR can be build using:
`MVR=comp12('Thermo1Build-reduce',mdl,RO);` If command option `-reduce` is given model is reduced according to real modeshapes. Then eig options must be given in `RO.EigOpt`.

Complex coupled thermoelastic modes can then be computed using
`def=comp12('Thermo1Modes',MVR);`

Following full example can be found in comp12('numeric'). It illustrates the computation of thermo-elastic damping in a simple rectangular plate example.

```
T0=20; % coupling temperature
% Build model: - - -
Nz=5;
mdl=femesh(sprintf('testhexa8 divide 10 10 %i',Nz));
h=2e-3;
mdl.Node(:,5:7)=mdl.Node(:,5:7)*diag([.304 .192 h]);
mdl.Elt=feutil('set groupall matid 1',mdl);
mdl.Elt=feutil('set groupall proid 1',mdl);

% define material properties: - - -
mdl.pl=m_elastic('dbval 1 Aluminum');  % structure
mdl=feutil('setmat 1 alpha=22e-6 T0=20',mdl);% XXX alpha at what temperature ?
mdl.pl=m_heat(mdl.pl,'dbval 2 Aluminum'); % therm
% define element properties:
mdl.il=p_solid('dbval 1 d3 -3'); % structure
mdl.il=p_heat(mdl.il,'dbval 2 d3 -3'); % therm
mdl=feutil('lin2quad',mdl);
RO=struct; % build options: - - -
RO.MatId=[1 2]; % structure | therm
RO.ProId=[1 2   T0    1 2]; % structure | therm | Temperature(if coupling, Inf if not)

% Assemble elementary matrices according to coupling defined in RO :
RO=comp12('Thermo1Assemble',mdl,RO);

% Build MVR with coupling :
RO.EigOpt=[5 40 1e3];
MVR=comp12('Thermo1Build-reduce',mdl,RO);

% Compute associated complex modes:
def=comp12('Thermo1Modes',MVR); % compute modes
cf=feplot(mdl); cf.def=def;
```

3

Composite FEM modeling

Contents

3.1 Material models **54**

3.1.1 Classical lamination theory 54

3.1.2 3D anisotropy 57

3.2 Ply generation example **57**

3.3 Material orientation maps **58**

3.4 Topology optimization tools **58**

This is a first attempt at describing composite related tools in SDT.

3.1 Material models

3.1.1 Classical lamination theory

Both isotropic and orthotropic materials are considered. In these cases, the general form of the 3D elastic material law is

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ & C_{22} & C_{23} & 0 & 0 & 0 \\ & & C_{33} & 0 & 0 & 0 \\ & & & C_{44} & 0 & 0 \\ & (s) & & & C_{55} & 0 \\ & & & & & C_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{Bmatrix} \tag{3.1}$$

Plate formulation consists in assuming one dimension, the thickness along x_3, negligible compared with the surface dimensions. Thus, vertical stress $\sigma_{33} = 0$ on the bottom and upper faces, and assumed to be neglected throughout the thickness,

$$\sigma_{33} = 0 \Rightarrow \epsilon_{33} = -\frac{1}{C_{33}} \left(C_{13}\epsilon_{11} + C_{23}\epsilon_{22} \right), \tag{3.2}$$

and for isotropic material,

$$\sigma_{33} = 0 \Rightarrow \epsilon_{33} = -\frac{\nu}{1-\nu} \left(\epsilon_{11} + \epsilon_{22} \right). \tag{3.3}$$

By eliminating σ_{33}, the plate constitutive law is written, with engineering notations,

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 \\ 0 & 0 & Q_{66} & 0 & 0 \\ 0 & 0 & 0 & Q_{44} & 0 \\ 0 & 0 & 0 & 0 & Q_{55} \end{bmatrix} \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{13} \end{Bmatrix}. \tag{3.4}$$

The reduced stiffness coefficients Q_{ij} (i,j = 1,2,4,5,6) are related to the 3D stiffness coefficients C_{ij} by

$$Q_{ij} = \begin{cases} C_{ij} - \dfrac{C_{i3}C_{j3}}{C_{33}} & \text{if i,j=1,2,} \\ C_{ij} & \text{if i,j=4,5,6.} \end{cases} \tag{3.5}$$

The reduced elastic law for an isotropic plate becomes,

$$\left\{ \begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \tau_{12} \end{array} \right\} = \frac{E}{(1-\nu^2)} \left[\begin{array}{ccc} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{array} \right] \left\{ \begin{array}{c} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{array} \right\}, \tag{3.6}$$

and

$$\left\{ \begin{array}{c} \tau_{23} \\ \tau_{13} \end{array} \right\} = \frac{E}{2(1+\nu)} \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \left\{ \begin{array}{c} \gamma_{23} \\ \gamma_{13} \end{array} \right\}. \tag{3.7}$$

Under Reissner-Mindlin's kinematic assumption the linearized strain tensor is

$$\epsilon = \left[\begin{array}{ccc} u_{1,1} + x_3\beta_{1,1} & \frac{1}{2}(u_{1,2} + u_{2,1} + x_3(\beta_{1,2} + \beta_{2,1})) & \frac{1}{2}(\beta_1 + w_{,1}) \\ & u_{2,2} + x_3\beta_{2,2} & \frac{1}{2}(\beta_2 + w_{,2}) \\ (s) & & 0 \end{array} \right]. \tag{3.8}$$

So, the strain vector is written,

$$\{\epsilon\} = \left\{ \begin{array}{c} \epsilon_{11}^m + x_3\kappa_{11} \\ \epsilon_{22}^m + x_3\kappa_{22} \\ \gamma_{12}^m + x_3\kappa_{12} \\ \gamma_{23} \\ \gamma_{13} \end{array} \right\}, \tag{3.9}$$

with ϵ^m the membrane, κ the curvature or bending, and γ the shear strains,

$$\epsilon^m = \left\{ \begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \end{array} \right\}, \ \kappa = \left\{ \begin{array}{c} \beta_{1,1} \\ \beta_{2,2} \\ \beta_{1,2} + \beta_{2,1} \end{array} \right\}, \ \gamma = \left\{ \begin{array}{c} \beta_2 + w_{,2} \\ \beta_1 + w_{,1} \end{array} \right\}, \tag{3.10}$$

Note that the engineering notation with $\gamma_{12} = u_{1,2} + u_{2,1}$ is used here rather than the tensor notation with $\epsilon_{12} = (u_{1,2} + u_{2,1})/2$. Similarly $\kappa_{12} = \beta_{1,2} + \beta_{2,1}$, where a factor 1/2 would be needed for the tensor.

The plate formulation links the stress resultants, membrane forces $N_{\alpha\beta}$, bending moments $M_{\alpha\beta}$ and shear forces $Q_{\alpha 3}$, to the strains, membrane ϵ^m, bending κ and shearing γ,

$$\left\{ \begin{array}{c} N \\ M \\ Q \end{array} \right\} = \left[\begin{array}{ccc} A & B & 0 \\ B & D & 0 \\ 0 & 0 & F \end{array} \right] \left\{ \begin{array}{c} \epsilon^m \\ \kappa \\ \gamma \end{array} \right\}. \tag{3.11}$$

The stress resultants are obtained by integrating the stresses through the thickness of the plate,

$$N_{\alpha\beta} = \int_{hb}^{ht} \sigma_{\alpha\beta}\, dx_3, \quad M_{\alpha\beta} = \int_{hb}^{ht} x_3\, \sigma_{\alpha\beta}\, dx_3, \quad Q_{\alpha 3} = \int_{hb}^{ht} \sigma_{\alpha 3}\, dx_3, \tag{3.12}$$

with $\alpha, \beta = 1, 2$.

Therefore, the matrix extensional stiffness matrix $[A]$, extension/bending coupling matrix $[B]$, and the bending stiffness matrix $[D]$ are calculated by integration over the thickness interval $[hb \;\; ht]$

$$\begin{array}{cc} A_{ij} = \int_{hb}^{ht} Q_{ij}\, dx_3, & B_{ij} = \int_{hb}^{ht} x_3\, Q_{ij}\, dx_3, \\ D_{ij} = \int_{hb}^{ht} x_3^2\, Q_{ij}\, dx_3, & F_{ij} = \int_{hb}^{ht} Q_{ij}\, dx_3. \end{array} \tag{3.13}$$

An improvement of Mindlin's plate theory with transverse shear consists in modifying the shear coefficients F_{ij} by

$$H_{ij} = k_{ij} F_{ij}, \tag{3.14}$$

where k_{ij} are correction factors. Reddy's 3^{rd} order theory brings to $k_{ij} = \frac{2}{3}$. Very commonly, enriched 3^{rd} order theory are used, and k_{ij} are equal to $\frac{5}{6}$ and give good results. For more details on the assessment of the correction factor, see [46].

For an isotropic symmetric plate ($hb = -ht = h/2$), the in-plane normal forces N_{11}, N_{22} and shear force N_{12} become

$$\left\{ \begin{array}{c} N_{11} \\ N_{22} \\ N_{12} \end{array} \right\} = \frac{Eh}{1-\nu^2} \left[\begin{array}{ccc} 1 & \nu & 0 \\ & 1 & 0 \\ (s) & & \frac{1-\nu}{2} \end{array} \right] \left\{ \begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \end{array} \right\}, \tag{3.15}$$

the 2 bending moments M_{11}, M_{22} and twisting moment M_{12}

$$\left\{\begin{array}{c} M_{11} \\ M_{22} \\ M_{12} \end{array}\right\} = \frac{Eh^3}{12(1-\nu^2)} \left[\begin{array}{ccc} 1 & \nu & 0 \\ & 1 & 0 \\ (s) & & \frac{1-\nu}{2} \end{array}\right] \left\{\begin{array}{c} \beta_{1,1} \\ \beta_{2,2} \\ \beta_{1,2}+\beta_{2,1} \end{array}\right\}, \tag{3.16}$$

and the out-of-plane shearing forces Q_{23} and Q_{13},

$$\left\{\begin{array}{c} Q_{23} \\ Q_{13} \end{array}\right\} = \frac{Eh}{2(1+\nu)} \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right] \left\{\begin{array}{c} \beta_2 + w_{,2} \\ \beta_1 + w_{,1} \end{array}\right\}. \tag{3.17}$$

One can notice that because the symmetry of plate, that means the reference plane is the mid-plane of the plate ($x_3(0) = 0$) the extension/bending coupling matrix $[B]$ is equal to zero.

Using expression (3.13) for a constant Q_{ij}, one sees that for a non-zero offset, one has

$$A_{ij} = h\,[Q_{ij}] \quad B_{ij} = x_3(0)h\,[Q_{ij}] \quad C_{ij} = (x_3(0)^2h + h^3/12)\,[Q_{ij}] \quad F_{ij} = h\,[Q_{ij}] \tag{3.18}$$

where is clearly appears that the constitutive matrix is a polynomial function of h, h^3, $x_3(0)^2h$ and $x_3(0)h$. If the ply thickness is kept constant, the constitutive law is a polynomial function of $1, x_3(0), x_3(0)^2$.

3.1.2 3D anisotropy

The constitutive equation in Voigt notation is of the form

$$\left\{\begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{array}\right\} = [C] \left\{\begin{array}{c} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{array}\right\} \Leftrightarrow \left\{\begin{array}{c} \sigma_{1} \\ \sigma_{2} \\ \sigma_{3} \\ \sigma_{4} \\ \sigma_{5} \\ \sigma_{6} \end{array}\right\} = \left[\begin{array}{cccccc} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & & & & C_{55} & C_{56} \\ \text{symm.} & & & & & C_{66} \end{array}\right] \left\{\begin{array}{c} \epsilon_{1} \\ \epsilon_{2} \\ \epsilon_{3} \\ \epsilon_{4} \\ \epsilon_{5} \\ \epsilon_{6} \end{array}\right\} \tag{3.19}$$

3.2 Ply generation example

fevisco('CompThick',cf);

xxx

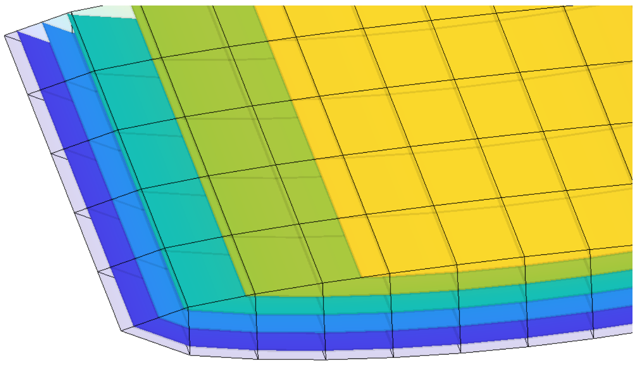


Figure 3.1: Definition of ply surfaces from distance to face

3.3 Material orientation maps

Computation of fiber orientation has been the object of much work [?]. Draping commercial codes are available: QUICK-FORM, CATIA CPD, MSC Laminate Modeller, FiberSIM. SDT only seeks to allow incorporation of results of such software

- incorporating material orientations in element properties `coordm` requires definition of a property per element.
- element wise `EltOrient` stack entry is most relevant for thin plies since orientation changes at inter-ply boundaries. This is typically stored as a structure with `.bas` and `.EltId` fields. Usually the transform from material to global orientation `TGL=reshape(bas(7:15),3,3)`. `TensorT=feval(m_elastic('MechaTensorT'),TGL)` returns a structure with tensor transforms. Thus `DDL=r4.tLG*DDG*r4.tGL` corresponds to the coordinate transform.

  `d_visco('ScriptEltOrient')` provides a sample script.
- node wise `NodeOrient` stack entry can be used to model configurations with smooth property gradients.

3.4 Topology optimization tools

xxx rewrite formula (9) of [?] .

The parametrization of mass and stiffness is of the form DD(comp,orientMap,orientIncrement) : to represent viscoelastic deviatoric `kvd`, viscous isochore `kvi`, contraining layer `kcl`, base structure `ke`

$$\begin{array}{c} K(p_g, DD) = K_0 + \sum_e \left((\epsilon_\kappa + (1-\epsilon_\kappa) g_{ke}(p_g)\right) K_e \\ M(p) = M_0 + \sum_e \left((\epsilon_\kappa + (1-\epsilon_\kappa) g_{me}(p_g)\right) M_e \end{array} \tag{3.20}$$

where parameters p may link multiple elements while $g_{ke}(p)$ designates the coefficient associate with a given integration point. The matrix derivatives are given by

$$\begin{array}{c} \frac{\partial K}{\partial p} = \sum_e (1-\kappa) \frac{\partial g_{ke}}{\partial p} K_e \\ \frac{\partial M}{\partial p} = \sum_e (1-\kappa) \frac{\partial g_{me}}{\partial p} M_e \end{array} \tag{3.21}$$

The MSE approximation of loss is given by

$$\eta_j = \frac{\phi_j^T K_v \phi_j}{\omega_j^2} \tag{3.22}$$

but one minizes the objective function combining multiple modes with positive weights μ_j

$$J = \sum_j \frac{\mu_j}{\eta_j} \tag{3.23}$$

whose derivative is given by

$$\frac{\partial J}{\partial p} = \sum_j \mu_j \frac{\partial \eta_j^{-1}}{\partial p} = \sum_j \mu_j \frac{-1}{\eta_j^2} \frac{\partial \eta_j}{\partial p} \tag{3.24}$$

with the detailed expression of the contribution of each mode (a negative sensitivity corresponds to more added damping)

$$\frac{\partial \eta_j^{-1}}{\partial p} = \frac{-1}{\eta_j^2} \left[-\frac{\partial \omega_j^2}{\partial p} \frac{1}{\omega_j^4} \phi_j^T K_v \phi_j + \frac{1}{\omega_j^2} \phi_j^T \frac{\partial K_v}{\partial p} \phi_j + \frac{2}{\omega_j^2} \phi_j^T K_v \frac{\partial \phi_j}{\partial p} \right] \tag{3.25}$$

where the squared frequency sensitivity is

$$\frac{\partial \omega_j^2}{\partial p} = \phi_j^T \left(\frac{\partial K}{\partial p} - \omega_j^2 \frac{\partial M}{\partial p} \right) \phi_j \tag{3.26}$$

and a negative frequency sensitivity corresponds to an added damping (because in (3.22) decreasing ω_j will increase damping). and the shape derivative is found as

$$\frac{\partial \phi_j}{\partial p} = \left[K - \omega_j^2 M\right]^{-1} \left[\frac{\partial K}{\partial p} - \omega_j^2 \frac{\partial M}{\partial p} - \frac{\partial \omega_j^2}{\partial p} M \right] \{\phi_j\} \tag{3.27}$$

4

Toolbox tutorial

Contents

4.1 Download and installation procedures . . . 63
4.2 Representing viscoelastic materials . . . 63
4.2.1 Introducing your own nomograms . . . 64
4.2.2 Selecting a material for your application . . . 65
4.2.3 Selective components in constitutive law . . . 65
4.3 Viscoelastic device meshing tools . . . 66
4.3.1 Generation of sandwich models . . . 66
4.3.2 Meshing foam fillings . . . 67
4.3.3 Exporting submeshes to NASTRAN . . . 68
4.4 Parametric models, structure reference . . . 68
4.4.1 Parametric models, zCoef . . . 68
4.4.2 Parametric models, zCoef . . . 72
4.4.3 Input definitions . . . 74
4.4.4 Sensor definitions . . . 74
4.4.5 MVR Format reference . . . 75
4.4.6 Response post-processing options . . . 75
4.5 Sample setup for parametric studies . . . 76
4.5.1 Performance in modulus/loss plane . . . 76
4.5.2 Illustration of pole range computations . . . 78
4.5.3 Model parameterization . . . 78
4.5.4 Sample parametric study in SDT (full solver, Upcom superelement) . . . 79
4.5.5 Parametric model generated within NASTRAN (fo_by_set DMAP) . . . 80
4.5.6 Parametric model from NASTRAN element matrices . . . 82
4.6 Fluid/structure coupling . . . 83
4.6.1 Summary of theory . . . 83
4.6.2 Acoustic stiffness on a loudspeaker . . . 84
4.7 Rayleigh integral computations . . . 85

4.7.1 Summary of theory 85
4.7.2 Diffuse field and transmission loss 85
4.8 NASTRAN Generation of the parametric model 86
4.9 Advanced connection models 87
4.9.1 Screw models 87
4.9.2 Physical point with rotations 88
4.10 External links 88

The viscoelastic modeling tools provide packaged solutions to address the following problems

- Generation of sandwich structures
- Handling of tabulated and analytical representations of viscoelastic constitutive laws.
- Parameterization of FEM models to allow multiple viscoelastic materials.
- Approximate solutions for the viscoelastic response
- Vibroacoustic response generation

These tools are not distributed with SDT. Please contact SDTools for licensing information.

4.1 Download and installation procedures

- you must first install the latest SDT release. Unless we suggest otherwise, you should use the pre-release version at `https://www.sdtools.com/sdtcur_dis.html`).
- You should use a 64 bit MATLAB (it is not realistic to use viscoelastic tools on a 32 bit application)
- install the viscoelastic tools by patching your SDT.
  - download `https://www.sdtools.com/distrib/beta/visco_patch_dis.p`) into a temporary directory (not the SDT directory)
  - in MATLAB, move to that directory and install the patch simply by using `visco_patch_dis`. Note that you must have write permission to the SDT directory so that if you have installed SDT in the "Program files" directory you should run MATLAB as an Administrator, see `https://www.sdtools.com/faq/Release.html#install`.
- check that the documentation has been installed using `sdtweb('visctoc')`.
- if you intend to use larger model install the SDT/MKL server using `sdtcheck('PatchMkl')`.

4.2 Representing viscoelastic materials

4.2.1 Introducing your own nomograms

You can introduce your own nomograms in the `m_visco` database. By simply defining an `mvisco_*.m` file (mvisco_3m.m serves as a prototype. The data structure defines a reference elastic material in `mat.pl`, complex modulus and shift factor tables, an finally additional properties stored in `mat.nomo` (which will be better documented later).

```
mat.pl=[1 fe_mat('m_elastic','SI',1) 1e6*2*1.49 .49 1500 1e6];
mat.name='ISD112 (1993)';
mat.type='m_visco';
mat.unit='SI';
mat.T0=[0];

mat.G=[ % Freq, Re(G) Im(G)
        1  1.72688e+004  3.51806e+003
       10  2.33865e+004  5.35067e+003
      100  3.49390e+004  8.25596e+003
     1000  5.76323e+004  1.67974e+004
    10000  1.03151e+005  5.72383e+004
   1e+005  2.10295e+005  1.79910e+005
   1e+006  6.59947e+005  6.57567e+005
   1e+007  2.06023e+006  1.95406e+006
   1e+008  5.83327e+006  3.57017e+006
   1e+009  1.48629e+007  5.60247e+006
   1e+010  3.25633e+007  7.33290e+006
   1e+011  6.16925e+007  5.40189e+006
   1e+012  1.01069e+008  2.48077e+006
];

mat.at=[ % T, at
      -10  1.32885e+007
        0  9.16273e+005
       10  1.14678e+005
       20  2.45660e+004
       30  9.00720e+003
       40  2.99114e+003
       50  1.27940e+003
       60  7.10070e+002
       70  2.88513e+002
       80  1.96644e+002
```

```
        90  1.37261e+002
       100  1.03674e+002
       110  6.84906e+001
       120  4.66815e+001
];
mat.nomo={'w',[-1 0 12],'Eeta',[4 9 2],'unit','SI', ...
          'www','www.3m.com', ...
          'file','ISD_112_93.png','Rect',[145    35   538   419], ...
          'type','G'};
```

4.2.2 Selecting a material for your application

```
cf=feplot; m_visco('database',cf); % select all materials
m_visco('info',cf);

m_visco('deffreq',cf) % set frequencies vector for all the material
m_visco('defT',cf)    % set temperature vector for all the material

cf.Stack{'info','Freq'}=logspace(3,log10(15e3),300); % range of interest
cf.Stack{'info','Range'}=20; % Temperature of interest
m_visco('nomo',cf) % list with all nomograms

Freq=stack_get(cf.mdl,'info','Freq','getdata');
T=stack_get(cf.mdl,'info','Range','getdata');
Mat=stack_get(cf.mdl,'mat');

m_visco('nomo',cf)
```

4.2.3 Selective components in constitutive law

General anisotropic elastic materials are described by a constitutive law of the form

$$\{\sigma\} = [\Lambda]\{\epsilon\} \tag{4.1}$$

where for orthotropic material the expression of $[\Lambda]$ is given by:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{Bmatrix} = [\Lambda]\{\epsilon\} = \begin{bmatrix} \frac{1-\nu_{yz}\nu_{zy}}{E_yE_z\Delta} & \frac{\nu_{yx}+\nu_{zx}\nu_{yz}}{E_yE_z\Delta} & \frac{\nu_{zx}+\nu_{yx}\nu_{zy}}{E_yE_z\Delta} & 0 & 0 & 0 \\ \frac{\nu_{yx}+\nu_{zx}\nu_{yz}}{E_yE_z\Delta} & \frac{1-\nu_{xz}\nu_{zx}}{E_xE_z\Delta} & \frac{\nu_{yz}+\nu_{yx}\nu_{xz}}{E_xE_y\Delta} & 0 & 0 & 0 \\ \frac{\nu_{zx}+\nu_{yx}\nu_{zy}}{E_yE_z\Delta} & \frac{\nu_{yz}+\nu_{yx}\nu_{xz}}{E_xE_y\Delta} & \frac{1-\nu_{xy}\nu_{yx}}{E_xE_y\Delta} & 0 & 0 & 0 \\ 0 & 0 & 0 & G_{yz} & 0 & 0 \\ 0 & 0 & 0 & 0 & G_{xz} & 0 \\ 0 & 0 & 0 & 0 & 0 & G_{xy} \end{bmatrix} \begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix} \tag{4.2}$$

with $\Delta = \frac{1-\nu_{xy}\nu_{yx}-\nu_{yz}\nu_{zy}-\nu_{zx}\nu_{xz}-2\nu_{yx}\nu_{zy}\nu_{xz}}{E_xE_yE_z}$. The expression of the flexibility allows easier identification of terms:

$$[\Lambda]^{-1} = \begin{bmatrix} 1/E_x & -\nu_{xy}/E_x & -\nu_{zx}/E_z & 0 & 0 & 0 \\ -\nu_{yx}/E_y & 1/E_y & -\nu_{yz}/E_y & 0 & 0 & 0 \\ -\nu_{xz}/E_x & -\nu_{zy}/E_z & 1/E_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{yz} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{xz} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{xy} \end{bmatrix} \tag{4.3}$$

If one has no particular interest in the engineering constants, the orthotropic constitutive law (4.2) can also be written:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{31} \\ \sigma_{12} \end{Bmatrix} = [\Lambda]\{\epsilon\} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} & \Lambda_{13} & 0 & 0 & 0 \\ \Lambda_{12} & \Lambda_{22} & \Lambda_{23} & 0 & 0 & 0 \\ \Lambda_{13} & \Lambda_{23} & \Lambda_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & \Lambda_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & \Lambda_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & \Lambda_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_1 = \epsilon_{11} \\ \epsilon_2 = \epsilon_{22} \\ \epsilon_3 = \epsilon_{33} \\ \epsilon_4 = 2\epsilon_{23} \\ \epsilon_5 = 2\epsilon_{31} \\ \epsilon_6 = 2\epsilon_{12} \end{Bmatrix} \tag{4.4}$$

In the `fevisco` `MatSplit` command. `ortho` allows the decomposition in the 9 non-zero terms shown above, while `EG` groups C_{44}, C_{55}, C_{66} as G and other terms as E.

4.3 Viscoelastic device meshing tools

4.3.1 Generation of sandwich models

Starting from an undamped structure without treatment, you often want to generate models for viscoelastic patches applied to the structure. This is done using `fevisco MakeSandwich` commands. Modeling issues associated with this meshing are discussed in section 2.4 .

Sample meshes are listed with `fevisco('Test')`.

For example, the generation of a three layer sandwich with the original layer 0.01 thick (leading to a 0.005 offset), a volume of thickness 0.002, and a second 0.01 thick shell looks like

```
model=femesh('testquad4 divide 10 12');
model.Elt=feutil('orient 1 n 0 0 1',model);
sandCom=['makesandwich shell 0 0 .005 ' ...
         'volume 101 .002 shell 102 -.005 .005'];
treated='withnode{x>.5 & y>.5}';
model=fevisco(sandCom,model,treated);
cf=feplot;cf.model=model;fecom('colordatamat');
```

You can also specify normals using a map.

```
model=femesh('testquad4 divide 3 4');
model.Elt=feutil('orient 1 n 0 0 1',model);
sandCom=['makesandwich shell 0 0 .005 volume 101 .2 '];
% use a normal map that specify the direction of extrusion
MAP=feutil('getnormal map node',model); MAP.normal(:,1)=2;
model=fevisco(sandCom,model,'withnode{x>.5 & y>.5}',MAP);
cf=feplot;cf.model=model;fecom(';colordatamat;view 1');
```

4.3.2 Meshing foam fillings

When meshing foam filling of geometrically complex parts, generating a mesh of the part and foam can be difficult. The problem is solved through the following steps

1. mesh the part (usually done in a CAD environment then imported into SDT), and mesh approximate foam volume.

2. defines a foam expansion map giving an expansion direction for external nodes of the approximate foam volume. In this phase, one typically defines the connected surface of the foam model, defines normals on this surface and possibly corrects these normals. The result is a vector map structure with field `.ID` node numbers of connected nodes and `.normal` giving the associated directions (see `fe_case map`).

3. expand the foam to touch the initial part mesh and generate MPC connections to connect the foam and the underlying mesh. This is done automatically with the `ConnectionStickThenSolid` command.

Example

```
femesh('reset');
model=femesh('testubeam');
model=feutil('objecthexa 101 101',model, ...
  [-.3 -.5 0;.6 0 0;0 .8 0;0 0 2.5],3,3,12);
cf=feplot(2);cf.model=model;fecom('colordatamat -alpha.1')
% Build normal MAP for connected foam surface
MAP=feutil('getnormal node MAP',model.Node, ...
 feutil(['selelt matid==101 &selface& ' ...
   'withnode {y>-.49 & z>0 & z<2.5}',model));
% Define the facing elements
FacingSurface='matid 1 & selface & facing >0 0 0 1';
%
mo2=fevisco('ConnectionStickThenSolid',model,'Foam',MAP,FacingSurface);
cf.model=mo2;fecom('colordatamat -alpha .3')
```

4.3.3 Exporting submeshes to NASTRAN

Once the sandwich model generated it can be exported to NASTRAN. There two typical strategies, rewriting the whole model (using `naswrite('FileName',model)`) or generating an include file.

The second strategy is more adapted when testing multiple viscoelastic configurations since it is more robust at preserving all options of the original file. The `fevisco WriteInclude` command is meant for that purpose. It lets you select newly meshed viscoelastic parts using any selection (their `MatId` for example) and generates the NASTRAN bulk containing the associated nodes, elements, material and element property cards, RBE2 entries connected to the selected elements.

4.4 Parametric models, structure reference

4.4.1 Parametric models, zCoef

Different major applications use families of structural models. *Update problems*, where a comparison with experimental results is used to update the mass and stiffness parameters of some elements or element groups that were not correctly modeled initially. *Structural design problems*, where component properties or shapes are optimized to achieve better performance. *Non-linear problems* where the properties of elements change as a function of operating conditions and/or frequency (viscoelastic behavior, geometrical non-linearity, etc.).

A *family of models* is defined (see [47] for more details) as a group of models of the general second order form (**??**) where the matrices composing the dynamic stiffness depend on a number of *design parameters* p

$$[Z(p,s)] = \left[M(p)s^2 + C(p)s + K(p)\right] \tag{4.5}$$

Moduli, beam section properties, plate thickness, frequency dependent damping, node locations, or component orientation for articulated systems are typical p parameters. The dependence on p parameters is often very non-linear. It is thus often desirable to use a model description in terms of other parameters α (which depend non-linearly on the p) to describe the evolution from the initial model as a linear combination (called `zCoef` in SDT)

$$[Z(p,s)] = \sum_{j=1}^{NB} \alpha_j(p) \, [Z_{j\alpha}(s)] \tag{4.6}$$

with each $[Z_{j\alpha}(s)]$ having constant mass, damping and stiffness properties.

Plates give a good example of p and α parameters. If p represents the plate thickness, one defines three α parameters: t for the membrane properties, t^3 for the bending properties, and t^2 for coupling effects.

p parameters linked to elastic properties (plate thickness, beam section properties, frequency dependent damping parameters, etc.) usually lead to low numbers of α parameters so that the α should be used. In other cases (p parameters representing node positions, configuration dependent properties, etc.) the approach is impractical and p should be used directly.

par

SDT handles parametric models where various areas of the model are associated with a scalar coefficient weighting the model matrices (stiffness, mass, damping, ...). The first step is to define a set of parameters, which is used to decompose the full model matrix in a linear combination.

The elements are grouped in non overlapping sets, indexed m, and using the fact that element stiffness depend linearly on the considered moduli, one can represent the dynamic stiffness matrix of the parameterized structure as a linear combination of constant matrices

$$[Z(G_m, s)] = s^2 [M] + \sum_m p_m [K_m] \tag{4.7}$$

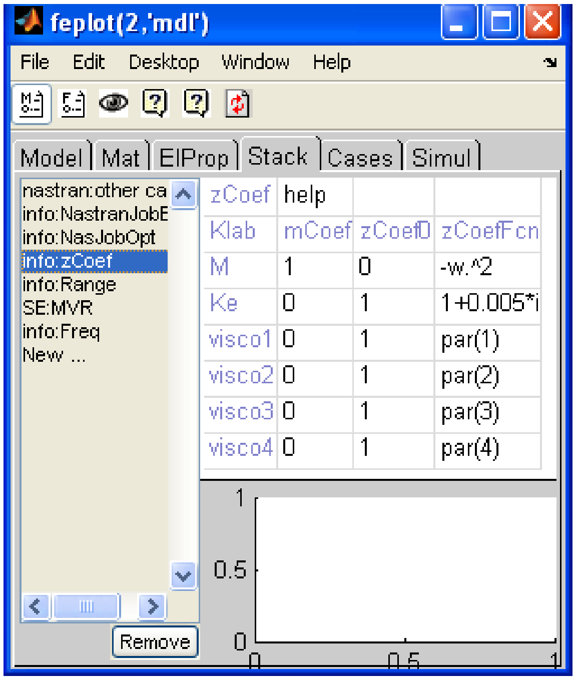
Parameters are case stack entries defined by using fe_case par commands (which are identical to upcom Par commands for an upcom superelement).

A parameter entry defines a element selection and a type of varying matrix. Thus

```
model=demosdt('demoubeam');
model=fe_case(model,'par k 1 .1 10','Top','withnode {z>1}');
fecom('proviewon');fecom('curtabCase','Top') % highlight the area
```

zCoef

The weighting coefficients in (4.7) are defined formally using the cf.Stack{'info','zCoef'} cell array viewed in the figure and detailed below.



nastran:other ca
info:NastranJobE
info:NasJobOpt
info:zCoef
info:Range
SE:MVR
info:Freq
New ...

zCoef	help		
Klab	mCoef	zCoef0	zCoefFcn
M	1	0	-w.^2
Ke	0	1	1+0.005*i
visco1	0	1	par(1)
visco2	0	1	par(2)
visco3	0	1	par(3)
visco4	0	1	par(4)

The columns of the cell array, which can be modified with the feplot interface, give

- the matrix labels Klab which must coincide with the defined parameters

- the values of coefficients in (4.7) for the nominal mass (typically `mCoef=[1 0 0 ... ]`)
- the real valued coefficients `zCoef0` in (4.7) for the nominal stiffness K_0
- the values or strings `zCoefFcn` to be evaluated to obtain the coefficients for the dynamic stiffness (4.7).

Given a model with defined parameters/matrices, `model=fe_def('zcoef-default',model)` defines default parameters.

`zcoef=fe_def('zcoef',model)` returns weighting coefficients for a range of values using the frequencies (see `Freq`) and design point stack entries

Frequencies are stored in the model using a call of the form `model=stack_set(model,'info','Freq',w_hertz_colum)`. Design points (temperatures, optimization points, ...) are stored as rows of the `'info','Range'` entry, see `fevisco Range` for generation.

When computing a response, `fe_def zCoef` starts by putting frequencies in a local variable `w` (which by convention is always in rd/s), and the current design point (row of `'info','Range'` entry or row of its `.val` field if it exists) in a local variable `par`. `zCoef2:end,4` is then evaluated to generate weighting coefficients `zCoef` giving the weighting needed to assemble the dynamic stiffness matrix (4.7). For example in a parametric analysis, where the coefficient `par(1)` stored in the first column of `Range`. One defines the ratio of current stiffness to nominal $Kvcurrent = par(1) * Kv(nominal)$ as follows

```
% external to fexf
 zCoef={'Klab','mCoef','zCoef0','zCoefFcn';
        'M'    1       0        '-w.^2';
        'Ke'   0       1        1+i*fe_def('DefEta',[]);
        'Kv'   0       1        'par(1)'};
model=struct('K',{cell(1,3)});
model=stack_set(model,'info','zCoef',zCoef);
model=stack_set(model,'info','Range', ...
   struct('val',[1;2;3],'lab',{{'par'}}));

%Within fe2xf
w=[1:10]'*2*pi;  % frequencies in rad/s
Range=stack_get(model,'info','Range','getdata');
for jPar=1:size(Range.val,1)
 Range.jPar=jPar;zCoef=fe2xf('zcoef',model,w,Range);
 disp(zCoef)
  % some work gets done here ...
end
```

4.4.2 Parametric models, zCoef

The viscoelastic tools handle parametric models where various areas of the model are associated with a scalar coefficient weighting the model matrices (stiffness, mass, damping, ...). The first step is to define a set of parameters, which is used to decompose the full model matrix in a linear combination. The elements are grouped in non overlapping sets, indexed m, and using the fact that element stiffness depend linearly on the considered moduli, one can represent the dynamic stiffness matrix of the parameterized structure as a linear combination of constant matrices

$$[Z(G_m, s)] = s^2\,[M] + [K_e] + \sum_m p_m\,[K_{vm}] \tag{4.8}$$

By convention, K_e represents the stiffness of all elements not in any other set. While the architecture is fully compatible there is no simplified mechanism to parameterize the mass.

The first step of a study is thus to define parameters. For all models this can be done using `fe_case par` commands (or `upcom Par` commands for an `upcom` superelement).

```
model=demosdt('demoubeam');
model=fe_case(model,'par','Top','withnode {z>1}');
fecom('proviewon');fecom('curtabCase','Top') % highlight the area
```

If the parameters correspond to viscoelastic materials, one needs to declare which, of the initially elastic materials, are really viscoelastic. This is done using `fevisco AddMat` calls which associate particular `MatId` values with viscoelastic materials selected in the `m_visco` database.

```
Up=d_visco('MeshPlate upreset');cf=feplot(Up);Up=cf.mdl;
Up=stack_rm(Up,'mat');
Up = fevisco('addmat 101',Up,'First area','ISD112 (1993)');
Up = fevisco('addmat 103',Up,'Second area','ISD112 (1993)');
cf.Stack{'info','Range'}=[20];
cf.Stack{'info','Freq'}=logspace(1,3,30);
%reset default zCoef Fcn and display
fe2xf('zCoef-default',cf);fe2xf('zCoef',cf)
```

Viscoelastic materials are then considered as parameters by `fevisco`.

The full constant matrices M, K_e, K_{vm} can be assembled using `fevisco MakeModel` or with a NASTRAN DMAP `fevisco Step12` (for implementations with other software such as ABAQUS, ANSYS or SAMCEF please contact us).

The normal mode of operation is to display your full model in a `feplot` figure. Reduced models can be generated with `fe2xf direct` commands, or with a NASTRAN DMAP `step12`, reduced matrices are stored in the model stack as an `'SE','MVR'` entry.

xxx

The problems handled by fe2xf and computations at multiple frequencies and design points. Frequencies are either stored in the model using a call of the form model=stack_set(model,'info','Freq',w
or given explicitly as an argument (the unit is then rad/s).

Design points (temperatures, optimization points, ...) are stored as rows of the 'info','Range' entry, see fevisco Range for generation.

When computing a response, fe2xf zCoef starts by putting frequencies in a local variable w (which by convention is always in rd/s), and the current design point (row of 'info','Range' entry or row of its .val field if it exists) in a local variable par. zCoef2:end,4 is then evaluated to generate weighting coefficients zCoef giving the weighting needed to assemble the dynamic stiffness matrix (4.7). For example in a parametric analysis, where the coefficient par(1) stored in the first column of Range. One defines the ratio of current stiffness to nominal $Kvcurrent = par(1) * Kv(nominal)$ as follows

```
% external to fexf
 zCoef={'Klab','mCoef','zCoef0','zCoefFcn';
        'M'    1       0        '-w.^2';
        'Ke'   0       1        1+i*fe_def('DefEta',[]);
        'Kv'   0       1        'par(1)'};
model=struct('K',{cell(1,3)});
model=stack_set(model,'info','zCoef',zCoef);
model=stack_set(model,'info','Range', ...
   struct('val',[1;2;3],'lab',{{'par'}}));

%Within fe2xf
w=[1:10]'*2*pi;  % frequencies in rad/s
Range=stack_get(model,'info','Range','getdata');
for jPar=1:size(Range.val,1)
 Range.jPar=jPar;zCoef=fe2xf('zcoef',model,w,Range);
 disp(zCoef)
  % some work gets done here ...
end
```

To use a viscoelastic material, you can simply declare it using an AddMat command and use a '_visc' entry in the zCoefFcn column (the stack name for the material and the zCoef matrix name must match).

```
cf=d_visco('MeshPlateLoadMV feplot');
% define material with a unit conversion
mat=m_visco('convert INSI',m_visco('database Soundcoat-DYAD609'));
```

```
%                  MatId for original, name of parameter
cf.mdl = fevisco('addmat 101',cf.mdl,'Constrained 101',mat);
cf.Stack{'zCoef'}(4,4)={'_visc'};
% Third coefficient will use material with name
cf.Stack{'zCoef'}{4,1}
fe2xf('zcoef',cf,500:10:4000,struct('val',5,'lab',{{'T'}}));
```

At time of computation, the matrix coefficient in (4.7) is found as $\frac{G_m(s,T,\sigma_0)}{G_{m0}}$ where the reference modulus is found in the `cf.Stack{'Constraint 101'}.pl` entry. The choice of E or G is based on the existence of a `'type','E'` or `'type','G'` entry in the `mat.nomo` field. For such materials one assumes Poisson's ratio to be real and the considered viscoelastic material to have its stiffness dominated by either shear or compression. Based on this assumption, one considers the frequency dependence of either the shear or Young's modulus. The error associated to neglecting the true variation of other moduli is assumed to be negligible (methodologies to treat cases where this is not true are not addressed here).

4.4.3 Input definitions

Inputs at DOFs are declared as `fe_case` entries.

If you have stored full basis vectors when building a reduced model (`MVR.TR` field), you may often be interested in redefining your inputs. To do so, you can use `fe_case` commands to define your loads, and `MVR=fe_sens('br&lab',MVR,model)`, to redefine the reduced inputs accordingly.

4.4.4 Sensor definitions

```
MV=d_visco('MeshPlate');
MV=fe_case(MV,'SensDof','BasicAtDof',[1;246]+.03);
% Surface velocity
'xxx'
% NEED TO BE REVISED WITH STRESS CUT 4 strain sensors along a line
%pos=linspace(.5,1,4)';pos(:,2)=.75; pos(:,3)=.006;
%data=struct('Node',pos,'dir',ones(size(pos,1),1)*[1 0 0]);
%MV=fe_case(MV,'SensStrain','ShearInLayer',data)
Sens=fe_case(MV,'sens')
```

If you have stored full basis vectors when building a reduced model (`MVR.TR` field), you may often be interested in redefining your sensors. To do so, you can use the `fe_case` commands to set your sensors, and `fe_sens('cr&lab',cf)` to redefine the reduced sensors accordingly (this calls

`Sens=fe_case('sens',model)` to generate the data structure combining the observation equations of all your sensors).

```
[MV,cf]=d_visco('MeshPlateLoadMVR');cf.Stack{'MVR'}
% redefine sensors
i1=feutil('findnode x==0 & z==0 epsl1e-3',MV);
MV=fe_case(MV,'SensDof','Sensors',i1+.03);
% reset reduced sensor representation
fe_sens('cr&lab',cf);cf.Stack{'MVR'}
```

4.4.5 MVR Format reference

A viscoelastic model (see section 4.5.6 for the typical generation procedure), stores the information needed to compute (4.7) in a data structure stored as a `SE,MVR` stack entry. The format corresponds to a generic type 1 superelement (handled by `fe_super`) but the following fields are specifically of interest.

`.Opt`	Options characterizing the model matrices. In particular the second row describes the type of each matrix in `model.K` (1 for stiffness, 2 for mass, ...)
`.K`	a cell array of matrices giving the constant matrices in (4.7). One normally uses $M, K_e, K_{vm}, \ldots$. These matrices should be real and only the combination coefficients should be complex.
`.Klab`	a cell array giving a labels for matrices in the `.K` field.
`.DOF`	DOF definition vector associated with matrices in `.K`
`.Stack`	standard model stack where one defines viscoelastic materials (see `fevisco AddMat` and `m_visco database`), frequencies, `zcoef`, reduced model, range (see `fevisco Range`) ... as well as the case that contains descriptions for loads, sensors, parameters, ...

Additional fields used in some solutions are

`.br`	input shape matrix for reduced model, generated by `fe_sens('br&lab',MVR)`.
`.cr`	input shape matrix for reduced model, generated by `fe_sens('cr&lab',MVR)`.
`.kd`	optional static preconditioner (nominally equal to `ofact(k0)`) use to compute static response based on a known residue. Deprecated and replaced by `fe2xf Build` calls.

4.4.6 Response post-processing options

For a given reduced (or full) model you may want to post-process the computed frequency responses before saving them. This is in particular important to analyze responses on large sensor sets (panel velocities, stresses, ...) which would require a lot of storage space if saved at all frequencies.

The list `MifDes` xxx

4.5 Sample setup for parametric studies

This section provides tutorial studies based on `SDT-visc` only.

4.5.1 Performance in modulus/loss plane

This tutorial uses the cantilever constrained plate shown in the figure below.

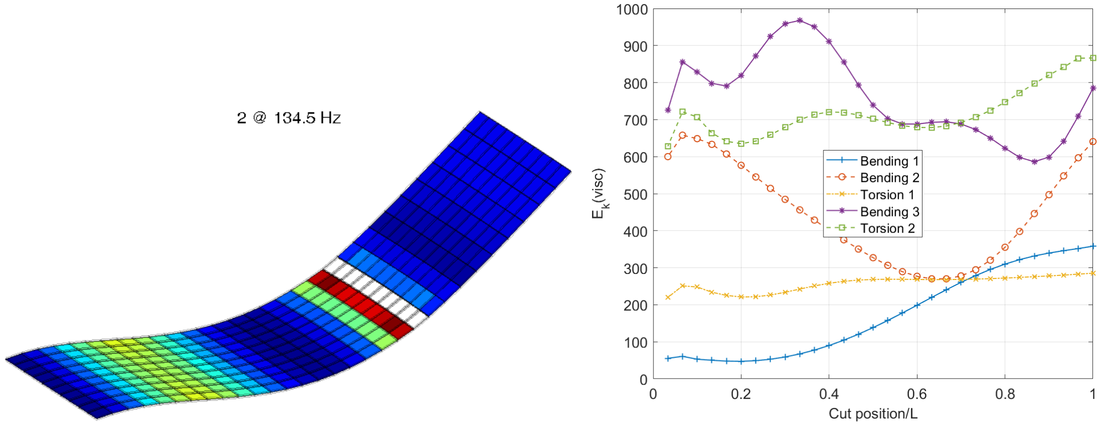


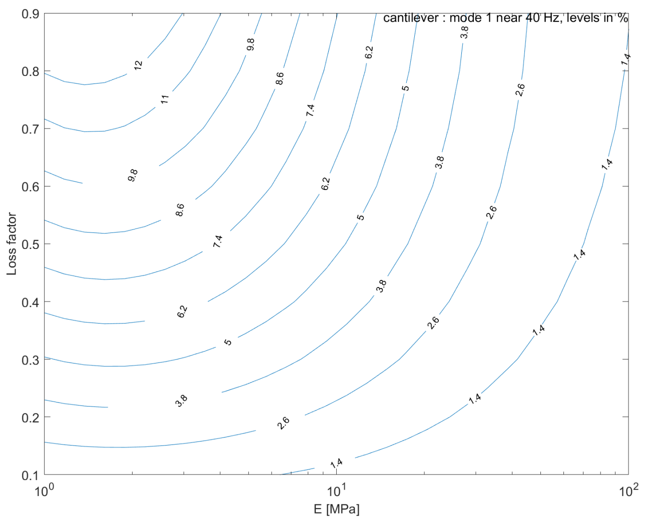
Figure 4.1: Constrained layer model validity.

`d_visco('TutoEhPerf-s1')` meshes a cantilever constrained plate.

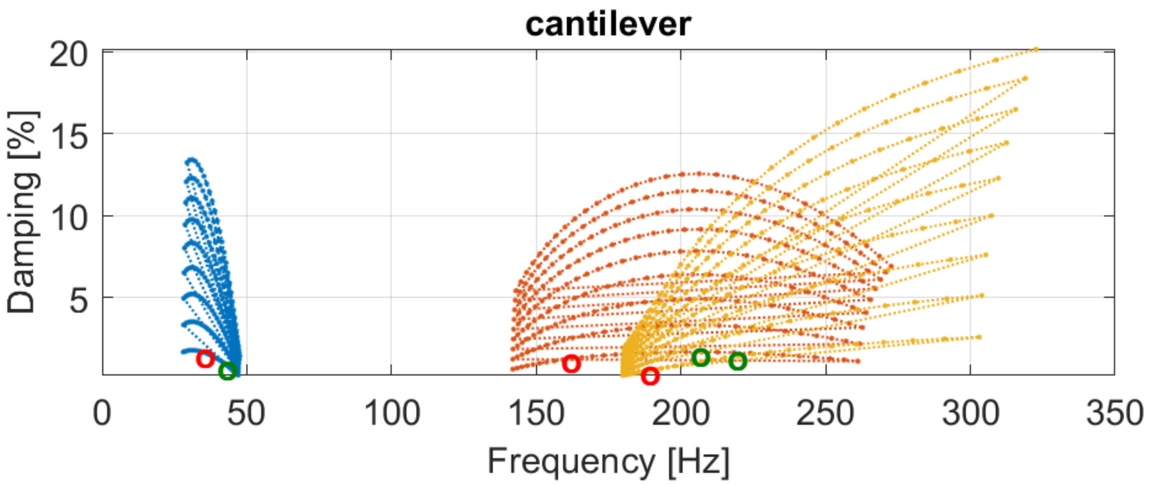
step 2 builds a reduced model. One starts by defining a viscoelastic material, and then uses an `fe2xf Build` call to build the reduced model by learning a soft and a stiff modulus point.

step 3 generates the standard frequency/damping curves for the first 3 modes.

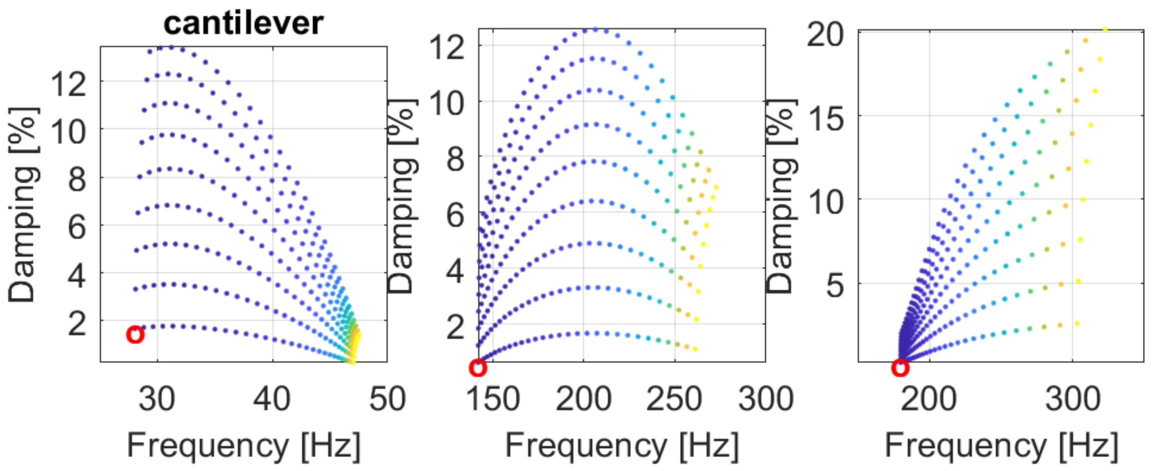
With `EhPerf` one obtains the damping level as function of modulus and loss factor map. In the present case one sees that the best performance is obtained for a modulus around 2 MPa.



The data used for the E, η performance map is a grid which for each mode can be shown in the frequency damping plane as illustrated below



It may also be interesting to split those maps by mode as shown here.



Older demos are `basic_sandwich` which generates curves for the validation of shell/volume/shell model used to represent constrained layer treatments as first discussed in section 2.4 .

4.5.2 Illustration of pole range computations

`d_visco('TutoPoleRange-s1')` uses a beam with viscoelastic rotational spring example.

4.5.3 Model parameterization

`fevisco` handles parametric models with variable stiffness expressed as linear combinations of constant matrices (4.7). The initial step of all parametric studies is to define the parameter sets.

For example, a selection based on `ProId`.

```
model=fe_case(model,'par','FrontStruts','-k ProId 168';
                    'par','BackStruts' ,'-k ProId 304'};
```

The PREDIT_mat model XXX

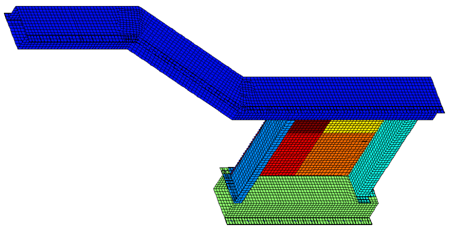
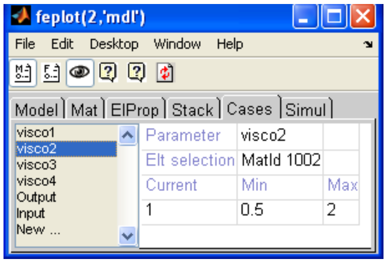


Figure 4.2: The PREDIT_mat model

For example, in the PREDIT_mat model, one can define each of the 4 squares of viscoelastic materials on the pane as a parameter using ProId

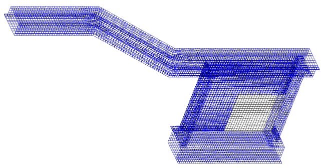
```
cf=feplot(model); MV=cf.mdl; %XXX
```

One can then visualize each parameter using feplot model properties windows, within the Case tab


feplot(2,'mdl')
File Edit Desktop Window Help
Model Mat ElProp Stack Cases Simul

visco1
visco2
visco3
visco4
Output
Input
New ...

Parameter	visco2	
Elt selection	MatId 1002	
Current	Min	Max
1	0.5	2



4.5.4 Sample parametric study in SDT (full solver, Upcom superelement)

Typically, one starts predictions with the a first order approximation to generate a reduced model MVR which is then used to approximate the frequency response functions Rred

```
Up=d_visco('meshplate upreset');
%Up=fevisco('testplate up');
Up=stack_rm(Up,'mat');
Up = fevisco('addmat 101',Up,'Area1','ISD112 (1993)');
Up = fevisco('addmat 103',Up,'Area2','ISD112 (1993)');
MV=fevisco('makemodel matid 101 103',Up);
cf=feplot(MV);
cf.Stack{'info','Range'}=[10;30];
cf.Stack{'info','Freq'}=[30:.5:500]';    % Target frequencies Hz
cf.Stack{'info','EigOpt'}=[6 40 -(2*pi*30)^2 11];
fe2xf('directfirst zCoef0',cf);
ci=iiplot(cf.Stack{'curve','RESP'});
iicom('challDesign point')

% This is a call to the new strategy for pole tracking as function of temp
RO=struct('IndMode',7:20,'Temp',20:10:100,'Freq',logspace(1,4,20)');
Po=fevisco('poletemp',cf,{'Area1';'Area2'},RO);
fe2xf('plotpolesearch',Po)
```

This is a sample direct computation of the viscoelastic response at 3 frequencies.

```
d_visco('meshplate matrix');cf=feplot;MV=cf.mdl;
cf.Stack{'info','Freq'}=[5:5:40]';
 [Rfull,def]=fe2xf('directfull',MV);
 cf.def=def; % NEED REVISE : iiplot(Rfull);
```

Given the reduced model `MVR` you can then track poles through your parametric range using

```
 MVR.Range=[0:10:50]';
 Hist=fe2xf('frfpolesearch',MVR);
 fe2xf('plotpolesearch',Hist) % Generate standard plot of result
```

4.5.5 Parametric model generated within NASTRAN (fo_by_set DMAP)

The following gives a simple example of a beam separated in two parts. The `step12` calls run NASTRAN with the `fo_by_set` DMAP where an eigenvalue computation is used to generate a modal basis that is then enriched (so called step 1) before generating a parametric reduced model (step 2).

The steps of the procedure are the following

- load the initial model into MATLAB using a `nasread` command.
- Generate through `naswrite` commands, or manual editing, a `*RootName*_bulk.bdf` file that contains bulk data information, EIGRL and and possibly PARAM cards.

  NOTE elements that will be parameterized should have the loss factor of their material set to zero.
- Define element groups associated with parameters (see section 4.5.3).

  These can be easily checked using `feplot` (open the `Edit:Model properties` menu and go to the `Case` tab). `cf.sel=*selection*{*1,2*}` commands. Once the job is written, elements sets will be written in a `parameter_sets.bdf` file (using `nas2up WriteSetC` commands).
- Sensors using `SensDof` case entries, see section 4.4.4 .
- Generate the `*RootName*_step12.dat` file and run the job using

  ```
  cf=feplot; % the model should be displayed in feplot
  cf.mdl=nas2up('JobOpt',cf.mdl); % Init NasJobOpt entry to its default
  fevisco('writeStep12 -run','RootName',cf)
  fecom(cf,'Save','FileName'); % save your model for reload
  ```

where the write command edits the nominal job files found in `sdtdef('FEMLink.DmapDir')`).

If you need to edit the bulk file, for job specific aspects of the `Case Control Section` (definition of MPC and SPC for example), omit the `-run`, do your manual edits, then run the job (for example `nas2up('joball memory=8GB','RootName_step12.dat'`).

You can also pre-specify a series of `EditBulk` entries so that your job can run automatically. For example

```
edits={'insert',   'SOL 103'         ,'', 'GEOMCHECK NONE';
       'replace', 'SPC        = 10','', 'SPC        = 1'};
model=stack_set(model,'info','EditBulk',edits);
```

- Once the NASTRAN job done, you should have locally the files `RootName_mkekvr.op4` (reduced matrices), `RootName_USETT.op2` degree of freedom set and info needed to build `Case.T`, `RootName_TR.op4` basis vectors defined on DOFs that are needed (sensor and input DOFs).

  You are now ready to build the reduced parameterized model using

  ```
  cf=feplot;fecom(cf,'Load','FileName');
  MVR=fevisco('BuildStep12','RootName',model)
  ```

  If the files are not automatically copied from the NASTRAN server machine, the `BuildStep12 -cpfrom` makes sure the result file are copied back.

Here is a complete example of this procedure

```
cd(sdtdef('tempdir')); if ~isunix; return;end % Don't test on windows
wd=sdtdef('FEMLink.DmapDir-safe',fullfile(fileparts(which('nasread')),'dmap'));
copyfile(fullfile(wd,'fo*.dmp'),'.')
copyfile(fullfile(wd,'sdt*.dmp'),'.')
!rm ubeam_step12.MASTER ubeam_step12.DBALL ubeam_*.[0-9]

model=demosdt('demoubeam');cf=feplot;
model=fe_case(model,'dofload','Input', ...
 struct('DOF',[349.01;360.01;241.01;365.03],'def',[1;-1;1;1],'ID',100));
model=fe_case(model,'par','Top','withnode {z>1}');
model=fe_case(model,'sensdof','Sensors',[360.01]);
cf.Stack{'info','EigOpt'}=[5 20 1e3 11];
model=nas2up('JobOpt',model); % Init NasJobOpt entry to its default
cf.Stack{'info','Freq'}=[20:2:150];
fevisco('writeStep12 -write -run','ubeam',model);
fecom('save','ubeam_param.mat'); % save before MVR is built
```

```
% you may sometimes need to quit Matlab here if NASTRAN is long
cf=feplot('Load','ubeam_param.mat'); % reload if you quit matlab
fevisco('BuildStep12','ubeam',cf)
fecom('save','ubeam_param.mat'); % save after MVR is built
```

When dealing with a model that would have been treated through a SOL108 (full order direct frequency response), the parametric model returns a B matrix. This is actually related to the K_e and K_{vi} matrices by

$$B = \eta_{glob} K_e + \sum_i \eta_{i,nom} K_{vi} \tag{4.9}$$

where η_{glob} is defined in NASTRAN using `PARAM,G` and the loss factors for the viscoelastic parts using the `GE` values of the respective components.

For viscoelastic analysis NASTRAN requests a complex curve $T(\omega)$ given as two tables. NASTRAN applies the `PARAM,G` to all elements (the matrix called K^1_{dd} in NASTRAN lingo is equal to $K_e + \sum_i K_{vi}$) as a result the variable viscoelastic stiffness which are here proportional to $G(\omega)/G_{i,nom} = 1 + i\eta_{glob} + T(\omega)\eta_{i,nom}$ hence, given the complex modulus, the table is given by

$$T = \frac{1}{\eta_{i,nom}} \left(\frac{Re(G(\omega))}{G_{i,nom}} - 1 \right) + i \left(\frac{Im(G(\omega))}{G_{i,nom}} - \eta_{glob} \right) \tag{4.10}$$

Generation of variable coefficients to be used in the toolbox from NASTRAN input is obtained with `MVR.zCoefFcn=fevisco('nastranzCoefv1',model)`.

4.5.6 Parametric model from NASTRAN element matrices

An alternative to the `step12` procedure Viscoelastic computations are performed on an `upcom` model typically imported from NASTRAN after a SOL103 (real eigenvalue) run with `PARAM,POST,-4` to export the element matrices. Thus for a NASTRAN run `UpModel.dat` which generated `UpModel.op2` and where the `upcom` superelement is saved in `UpModel.mat` a typical script begins with `Up=nasread('UpMo`

In some cases, one may want to use reduced basis vectors that are generated by an external code (typically NASTRAN). The problem is then to generate the reduced model `MVR` without asking `fevisco` to generate the appropriate basis. You can simply call the `DirectReduced` command

```
Up=d_visco('meshplate upreset');
Up=stack_rm(Up,'mat');
Up = fevisco('addmat 101',Up,'First area','ISD112 (1993)');
Up = fevisco('addmat 103',Up,'Second area','ISD112 (1993)');
```

```
def=fe_eig(Up,[6 10 1e3]);
MVR = fevisco('makemodel matid 101 103',Up,def);
MVR=stack_set(MVR,'info','Freq',[20:.5:90]');
RESP=fe2xf('directreduced',MVR);  % Result in XF(1)
```

4.6 Fluid/structure coupling

This section illustrates the use of `fevisco Fluid` commands.

4.6.1 Summary of theory

One considers fluid structure interaction problems of the form [48]

$$s^2 \begin{bmatrix} M & 0 \\ C^T & K_p \end{bmatrix} \begin{Bmatrix} q \\ p \end{Bmatrix} + \begin{bmatrix} K(s) & -C \\ 0 & F \end{bmatrix} \begin{Bmatrix} q \\ p \end{Bmatrix} = \begin{Bmatrix} F^{ext} \\ 0 \end{Bmatrix} \tag{4.11}$$

with q the displacements of the structure, p the pressure variations in the fluid and F^{ext} the external load applied to the structure, where

$$\begin{array}{c} \int_{\Omega_S} \sigma_{ij}(u)\epsilon_{ij}(\delta u)dx \Rightarrow \delta q^T K q \\ \int_{\Omega_S} \rho_S u.\delta u dx \Rightarrow \delta q^T M q \\ \frac{1}{\rho_F} \int_{\Omega_F} \nabla p \nabla \delta p dx \Rightarrow \delta p^T F p \\ \frac{1}{\rho_F c^2} \int_{\Omega_F} p \delta p dx \Rightarrow \delta p^T K_p p \\ \int_{\Sigma} p \delta u.n dx \Rightarrow \delta q^T C p \end{array}$$

Full order equations for the coupled problem are impractical. One thus considers model reduction for both the solid and fluid parts of the model. Given a reduction basis T_s for the structure, one builds a reduction basis containing fluid modes within the bandwidth of interest and static corrections for the effects of vectors in T_s. Thus the resulting basis for the fluid model is

$$[T_f] = \left[\phi_{f,1:NM} \quad [F]^{-1} [C]^T [T_s] \right] \tag{4.12}$$

Similar equations can of course be developed for applications where the fluid is represented using boundary elements [49].

4.6.2 Acoustic stiffness on a loudspeaker

The procedure is divided in the following steps:

- declare the fluid as a superelement in the structure model. A typical call would take the form

  ```
  [modelS,modelF]=fevisco('fluidtest'); % this generates demo models
  cf=feplot(fevisco('fluidmerge',modelS,modelF));
  fecom('curtabStack','SE:fluid')
  cf.sel(1)={'innode {y>=0}& eltname~=SE','colordatamat'}
  ```

  Note that the solid model `modelF` will often be read from NASTRAN as modeshapes (PARAM,POST 2) or `upcom` superelement (PARAM,POST,-4). If node numbers in the fluid and solid are coincident, the call automatically shifts fluid node numbers.

- assembly of the fluid/structure coupling and fluid matrices with the `FluidMatrix` command

  ```
  fevisco('fluidmatrix',cf, ...
    'SelElt selface', ...    % Fluid interface (in fluid SE)
    'SelElt EltName quad4'); % Solid interface (in structure)
  cf.Stack{'fsc'} % see the coupling superelement
  ```

  Arguments of the command are

  – a pointer to the `feplot` figure containing the solid model and fluid as a superelement

  – a series of `femesh` commands that allows the selection of the fluid interface for which a fluid structure property is defined. In the example, one selects the fluid (`eltname flui`) and keeps its exterior boundary (`SelFace`).

    If the generation of this interface cannot be performed using a single `feutil` command, then the element matrix should be provided instead of a selection command.

  – a selection for the solid interface. In the example, one retains all elements but in practical applications, one will often eliminate parts of the model (weld points, stiffeners, ...). You can again provide the selection as an element matrix.

  The result is stored as a `fsc` superelement. It is obtained by estimating translations at the fluid interface nodes trough MPCs (see `ConnectionSurface`) and standard integration of the fluid/structure coupling elements. Its DOFs combine DOFs of the solid model and pressure DOFs on the fluid superelement (the MPCs are eliminated).

- The final step is to generate a reduced coupled model. This requires defining, applied loads, required sensors, and a modal model structure in vacuum (defined trough its modes `def` as done here, or a parametric reduced model `MVR`. You can start by adding additional pressure sensors then use the call

```
cf.Stack{'fluid'}=stack_set(cf.Stack{'fluid'},'Info','EigOpt',[6 10 -1e3]);
cf.mdl=fe_case(cf.mdl,'SensDof','Sensors',[65.01;1.19], ...
 'DofLoad','Point load',[65.01]);
dsol=fe_eig(modelS,[6 20 1e3]);
cf.Stack{'info','FluidEta'}=.1;      % Fluid loss set to 10 %
cf.Stack{'info','DefaultZeta'}=.01;  % Structure loss set to 2 %
fevisco('fluidMakeReduced',cf.mdl,dsol);
```

Acoustic loads are not yet considered in this procedure.

- Reduced basis computations are then performed using low level fe2xf calls

```
[ci,XF]=iiplot
cf.Stack{'info','Freq'}=linspace(10,250,1024)';
ci.Stack={'curve','coupled',fe2xf('frfzr',cf)};
cf.Stack{'zCoef'}(4:6,4)={'-w.^2*1e-3';1e-3;1e-3};
ci.Stack{'curve','no fluid'}=fe2xf('frfzr',cf);
iicom('iixonly',{'coupled','no fluid'})
```

4.7 Rayleigh integral computations

4.7.1 Summary of theory

The Rayleigh integral provides an approximation of the pressure at point M as

$$p(M) = -\frac{\rho\omega^2}{2\pi}\sum_g \frac{\{n_g\}^T\{v(x_g)\}\,w_g J}{R(x)} e^{k(\omega)R(x)} \tag{4.13}$$

with $R(x,M)=|x-M|$ the distance between emission and target, $k=i\omega/c_0$ the wave number, ρ_0 the fluid density, c_0 the fluid celerity.

This integral is implemented in `fsc` and also in `fe2xf frfzr` in an optimized form.

4.7.2 Diffuse field and transmission loss

Sound transmission loss corresponds to the ratio between incident and radiated power in a panel.

The incident power is associated with acoustic loading assumed to be due to a diffuse field. This loading is classically estimated using the power spectral density at integration points of the excited surface

$$[S_F(\omega)]_{A,B} = S_{ref}(\omega)\left[w(g_A)J(g_A)w(g_B)J(g_B)\frac{sin(k\|A-B\|)}{k\|A-B\|}\right] \tag{4.14}$$

where $w(g_A)J(g_A)$ is the surface associated with the the integration rule at Gauss point A, $k=\omega/c$ the acoustic wavenumber, $\gamma_{AB}=\frac{sin(k\|A-B\|)}{k\|A-B\|}$ the spatial correlation of pressures between two points, $S_{ref}(\omega)$ the blocking pressure. [?] also addresses grazing incidences. The function $\frac{sin(x)}{x}$ is also called `sinc` in the code.

When using a reduced model, where a kinematic description of motion is of the form $\{q\}=[T]\{q_R\}$, it is more efficient to use the projection of the PSD defined in surface nodes onto the generalized coordinates q_R. Thus

$$[S_{FR}(\omega)]_{A,B} = S_{ref}(\omega)\,[T]^T\left[w(g_A)J(g_A)w(g_B)J(g_B)\frac{sin(k\|A-B\|)}{k\|A-B\|}\right][T] \tag{4.15}$$

xxx order of operations xxx

An alternative to the definition of a diffuse field through a spectral density matrix, is the use a sum of plane waves, each defined by an origin M, a direction $\{d_M\}$ and an amplitude $a_M(\omega)$. The pressure applied on any point of structure subjected to this wave is given by

$$p_M(x,\omega) = a_M(\omega)exp\left(-\frac{i\omega}{c_0}\{d_M\}^T\{x-M\}\right) = a(\omega)exp\left(-k(\omega)\{d\}^T\{x-M\}\right) \tag{4.16}$$

From a series of sources, and a loaded surface described by its Gauss points, the computation of the pressure load is done as

$$\{p_i(\omega)\} = \sum_M\sum_g w_gJN_i(x_g)a(M,\omega)exp\left(-k(\omega)\{d_M\}^T\{x_g-M\}\right) \tag{4.17}$$

In order to compute the radiation response, the next step is to compute the transfer from pressure forces to responses. xxx

Radiation from a surface into a cavity

4.8 NASTRAN Generation of the parametric model

The viscoelastic vibration toolbox is distributed with a set of NASTRAN DMAP and sample files that are used to implement some solutions steps within the NASTRAN environment. The following parameters are used in those DMAP

ViNset	number of sets defining the element selections.
ViOut	type of desired output. 0 returns the full matrices of the linear combination. 1 returns a first order reduced model.
ViSet1	ID for the first set. Sets are assumed to be written in sequential order.
ViIOSET	ID for set containing the list of nodes that need to be saved for input/output calculations.
ViShift	Frequency shift used to handle cases with rigid body modes. By default the shift is the complex double precision number 10.

fo_by_set

The objective of this DMAP (fo_by_set.dmp associated with sample job file vfirst_step12.dat) and the associated fevisco WriteStep12 and BuildStep12 commands is to generate a first order reduction of a model where parameters are coefficients on the stiffness matrix of elements within a given selection (NASTRAN set).

nas_sdtserv

This is used to overload basic *SDT* functionality through NASTRAN calls. You must first set local preferences

```
sd_pref('set','SDT','ExternalEig', ... % Callback used by fe_eig method 50
 'mode=fevisco(''nastran eig'',m,k,model.DOF,opt,model,Case);');
```

4.9 Advanced connection models

4.9.1 Screw models

Proper representation of screws is a classical difficulty that can be much alleviated using automated procedures. The models used here consider the screw as a circular beam with diameter equal to that of the hole.

Rigid screw models use rings of rigid connections. One only takes a few nodes on the screw to be masters, and big lumps of model nodes are forced to follow the associated rigid body motion.

Flexible screw models assume that the responses of the nodes on the beam are dependent on the response of a number of other nodes in the model. This dependency is represented as a weighted sum, which in terms of loads corresponds to a distribution of loads on a number of nodes.

4.9.2 Physical point with rotations

This type of connection finds the element a given node is connected to (see `feutilb` `match`) and uses shape functions of underlying element to estimate motion at the node. Rotations are determined using local derivatives of the shape functions at the physical point.

4.10 External links

References to external documents. In SDT use `sdtweb('ref')` to open the page.

- `ParamEdit` standard SDT parameter extraction `cingui`
- `def` shape at DOF `def`.
- `feplot` display mesh `feplot`.
- `staticNewton` non-linear static computation `fe_time`.
- `newmark` linear Newmark solver, `NLNewmark` nonlinear Newmark solver`fe_time`.
- `m_elastic` material property function , `m_elastic`
- `fe_caseg` assembly and GUI complement `fe_caseg`
- `ConnectionScrew`, `Assemble` assembly and GUI complement `fe_caseg`
- end

5

Toolbox Reference

Contents

fe2xf 90
fevisco 101
m_visco, mvisco_3m 109
nasread 116
naswrite 121

fe2xf

Purpose

Main function of SDT/ZParam for reduced parametric dynamic analysis This function is part of the viscoelastic tools.

The other purpose is the direct computation of frequency response functions.

Syntax

```
cases = fe2xf('command',model,case);
```

Description

frfzr [-file *FileName*]

This command supports direct frequency response computations and post-processing for reduced models. For a given input, the response is fully characterized by the response at DOFs $\{q\}$ (state vector in control theory), which dependence on the load is defined by an evolution equation (equation of dynamics in mechanics)

$$[Z(\omega)]_{N\times N}\{q(\omega)\}_{N\times 1} = [b]_{N\times NA}\{u(\omega)\}_{NA} \tag{5.1}$$

Unit inputs $[b]$ describe the spatial content of loads (see more details in section 4.4.3). The frequency content is described at each frequency by specifying an input $\{u\}$ or the associated covariance matrix Σ_{uu}. Cases with enforced displacements require a few additional manipulations but leads to similar equation forms.

Outputs are characterized by a vector $\{y\}$ that is supposed to be linearly related to DOFs through an observation equation (see more details in section 4.4.4)

$$[y(\omega)]_{NS} = [c]_{NS\times N}\{q(\omega)\}_N \tag{5.2}$$

where the y components can be translations, rotations, normal velocities, strains, ...

The first step of an analysis is to define the input shape matrix b and possibly the inputs u. The second step is to define the outputs.

`fe2xf` provides a fairly large set of response processing options for full and reduced models, through specification of a `'info','MifDes '` entry in the model stack. This entry should contain a cell array, each row describing a response processing with `{post_name,copt}`. `post_name` is the string identifier of the post (`'frf'`, `'svd'`, ...) and `copt` contains the options related to the post (see below). For example

```
MVR=fevisco('TestPlateLoadMVR');
MVR=stack_set(MVR,'info','Freq',[30:1:500]');
MVR=stack_set(MVR,'info','MifDes',{'frf',[]});
RESP=fe2xf(horzcat('frfzr -file',nas2up('tempname RESP.mat')),MVR);
R1=RESP(1);R1.xf=abs(R1.xf);fe_curve('plot -fig11 ylog xtight',R1);
```

These low level commands are used, given a reduced model, to compute FRFs for a frequency/parameter range. The reduced model uses the standard fields used to describe parametric models (see section 4.4.5). It must at least contain the following fields `.Range`, `.cr,.lab_out .br,.lab_in`, `zCoefFcn`, `.K`.

Given a *FileName*, results are saved every 30 seconds to allow post-processing during the evaluation.

Without output argument, one can specify the identifier of an `iiplot` or `feplot` figure (whose stack will store results of computation) using a `-cf` *i* command option.

Accepted post-processing options given in an `info,MifDes` entry

- `frf` computes the transfer function for unit inputs

$$[H(\omega)]_{NS\times NA} = [c]_{NS\times N} \, [Z]^{-1}_{N\times N} \, [b]_{N\times NA} \tag{5.3}$$

  Possibly select inputs and outputs by their indices with `copt.in` and `copt.out`.

- `frfu` computes the response to specified loads

$$\{y\}_{NS} = [H(\omega)]_{NS\times NA} \{u(\omega)\}_{NA} \tag{5.4}$$

  Possibly select outputs by their indices with `copt.out`.

- `svd` computes the singular values of H (the first one is sometimes called the spectral radius)

$$\sigma_k([H(\omega)]_{NS\times NA}) \tag{5.5}$$

  Possibly select of a subset of singular values with `copt.ind`.

- `svdu` computes the singular values of H weighted by the input level given in $\{u(\omega)\}$

$$\sigma_k([H(\omega)]_{NS\times NA} \left[{}^{\backslash} u(\omega)_{\backslash}\right]_{NA\times NA}) \tag{5.6}$$

  Possibly select of a subset of singular values with `copt.ind`. Only vectors $\{u(\omega)\}$ (one load) are supported at this time.

- `usvd` computes the singular values of $\{y\} = [H]\{u(\omega)\}$ which corresponds to the quadratic norm of the response.

$$\sigma_k(\{[H(\omega)]_{NS\times NA} \{u(\omega)\}_{NA}\}) \tag{5.7}$$

- ener[k, m] computes the strain energy (enerk) or the kinetic energy (enerm) in a selection of elements after recovering the physical displacement to associated DOFs. The recovery basis must fit in memory. copt must give a model.

- mean computes the mean quadratic response of one or multiple sensor sets. A sensor set is defined by a vector of indices in the cell array copt.group which each row has the form {SetName,Indices}.
  Following example computes the mean quadratic response along the node normal of previous model using the fe2xf function, and correlates it to FRF as an illustration

- Rayleigh computes the Rayleigh integral for a radiating surface.

```
cf=fevisco('TestPlateLoadMVR feplot'); MV=cf.mdl;
MAP=feutil('getnormalnode map',cf.mdl);
ind=find(ismember(MAP.ID,feutil('getnode MatId1',cf.mdl)));
MAP.ID=MAP.ID(ind); MAP.normal=MAP.normal(ind,:);
tdof=[MAP.ID MAP.ID MAP.normal];
cf.mdl=fe_case(cf.mdl,'remove','Sensors','SensDof','Sensors',tdof);
cf.Stack{'info','MifDes'}={ ...
  'mean -vel',struct('group',{{'main',[1:size(tdof,1)]'}}); % mean velocity
  'frf -vel',[]}; % all response
i1=feutil('findnode x==0 & z==0 epsl1e-3',MV);
MV=fe_case(MV,'DofLoad','Inputs',i1(1:2)+.03);
fe_sens('cr&lab',cf); fe_sens('br&lab',cf)
R1=fe2xf('frfzr',cf);
% compare mean and frf
r1=squeeze(mean(abs(R1(2).Y).^2,2)); r2=squeeze(R1(1).Y);
if norm(r1./r2-1)>1e-10;error('Mismatch in mean computation');end
```

- impe computes the impedance for an input $\{u(\omega)\}$.

$$\{u(\omega)\}^T [H(\omega)] \{u(\omega)\} \tag{5.8}$$

Following example computes a set of responses

```
cf=fevisco('TestPlateLoadMVR feplot'); MV=cf.mdl;
% redefine sensors
i1=feutil('findnode x==0 & z==0 epsl1e-3',MV);
MV=fe_case(MV,'SensDof','Sensors',i1+.03);
MV=fe_case(MV,'DofLoad','Inputs',i1(1:2)+.03);
```

```
MV=fe_case(MV,'setcurve','Inputs',{'input';'input'});
% reset reduced sensor representation
fe_sens('cr&lab',cf)
fe_sens('br&lab',cf)
MV=stack_set(MV,'info','Freq',[30:.5:32]');   % Target frequencies
% define the time-dependent load
data.X{1}=linspace(0,200*1e-4,201);
data.Xlab{1}=fe_curve('datatypecell','freq');
data.Y=ones(1,length(data.X{1}));
MV=fe_curve(MV,'set','input',data);
% define model selection for energy computation
mo1=feutil('rmfield',MV.GetData,'file','wd','Opt','Stack');
mo1.Elt=feutil('selelt matid103',mo1);
% define responses to be computed
MV=stack_set(MV,'info','MifDes',{'frf',[];
  'svd',struct('ind',2);'svdu',1;'usvd',[];
  'frf -acc',struct('out',1:2); 'frf -vel',struct('out',2);
  'frfu',[];
  'enerk',mo1;'enerm',mo1;
  'mean -vel',struct('group',{{'all',[1:10]}});
  'impe',[]});
% compute responses
fe2xf('frfzr',cf);
```

The commands `fe_sens('br&lab')` and `fe_sens('cr&lab')` are used to compute respectively the reduced load matrix br and the reduced observation matrix cr, and their associated labels. If a load is time dependent, the associated curve must be linked to it before calling `fe_sens('br&lab')`. However there is no need to call again `fe_sens('br&lab')` if the curve is changed provided that the name of the curve stays the same.

`direct [Full,First [,zCoef0], Iter, Reduced]`

`direct` commands are used for direct frequency response computations. It is assumed that loads and sensors are defined using entries in the `model` case.

`[XF,def]=fe2xf('DirectFull',model)` calls a direct full order complex sparse solver. With no output argument, the FRFs are displayed in `iiplot`.

`model` is a structure array that describes the impedance of the model whose response is to be computed. It is typically generated using the `fevisco MakeModel` command which describes the fields used by `fe2xf direct` commands (see section **??**).

DirectFirst builds a reduction basis containing nominal normal modes and a first order correction for damping effects. DirectFirst zCoef0 generates zCoef0 based on the actual contents of zCoef rather than the values stored in the info:zCoef stack entry.

DirectReduced assumes the model already contains a reduced model, as illustrated in section 4.5.6 .

These commands are illustrated in section 4.5.4 .

FrfPoleSearch

`[xf,po]=fe2xf('frfpolesearch',rmodel,w,ind)`

This low level command is used, given a reduced model, to track poles in the same range. The reduced model uses the standard fields used to describe parametric models (see section 4.4.5). It must at least contain the following fields `.Range`, `.cr,.br`, `zCoefFcn`, `.K`.

The pole tracking algorithm assumes that the modeshape is well approximated by the reduction vector with the same index.

PoleEh

Performance map of a single mode in the E,h plane is detailed in section **??**.

PoleTemp

Interpolation based search for dependence of poles on temperature (the FrfPoleSearch is a slower alternative). xxx example needed xxx

```
RO=struct('ind',1:10, ...  % selected modes for output
    'Temp',50:110, ...  % temperatures
    'Freq',logspace(log10(1000),log10(20e3),30)'); % target range of frequencies
Po=fe2xf('poletemp',MV2,'Visco',RO);
hist=fe2xf('PoleRange',MVR,ind);
fe2xf('plotpolesearch',hist);
```

PoleRange ...

This command supports a parametric study on the poles of frequency invariant damped model. Variable coefficients must be defined by a `zCoef` stack entry. The mass column in particular should be filled correctly and matrix types should be found in `model.Opt(2,:)` so that the stiffness can be found by setting to zero coefficients of mass matrices in `zCoef`.

If the provided superelement has an `nmap` field with key `Map:CurPar` or `CurParHandle` defining an `nmap`, the pointed `nmap` will be set with keys `Range` the current DoE and `jPar` the current design point. This allows implicit matrices using parameter interpolation to be updated during the procedure.

Options can be specified in an option structure `RR` in the example below or set in the command string

- `Real` specifies that real and not complex modes should be computed
- `-irange` specifies indices of design points to be used in the experiment specified by the `info,Range` entry.
- `-frac` adds energy fraction output for each mode in the history
- `.IndMode` indices of target modes. `r`b+(1:10) can be used to skip target modes.
- `-EigOpt`*`val`* can be used to specify an eigenvalue solver if non full is to be used.
- `-reduce` can be used to perform the second reduction layer for complex mode reduction in a real mode subspace, enhanced with non symmetric stiffness terms.
- `-optim`*`i`* in combination with `-reduce` to avoid recomputing the subspace at each step, if set to *`1`* enhancement is performed from the initial basis at each time step. If set to *`2`* no operation is performed on the reduction basis.
- `-redFcn"`*`call`*`"` in combination with `-reduce` and `-optim`*`2`* allows external computation of the real mode basis, must output `mdr`, `fr`, `k0`.

- `d_visco('TutoPoleRange-s2')` performs a basic range computation with result in the `Hist` data structure.
- `d_visco('TutoPoleRange-s3')` opens an feplot with data tip linking.

`PlotPoleSearch`

Is the general interface to display pole histories. Accepted options are

- `-cf` specifies the figure to use.
- `-khz` uses kHz units.
- `-nomind i` specified design point numbers to be highlighted by a marker.

- `-ShowDef` displays in feplot, the mode shapes associated with design points in `.NomInd` in a call of the form `fe2xf('plotpolesearch -cf1 -nomind 15 -leg E -showdef',hist,MVR,cf)`.
- `-leg s` specifies a legend type. Currently `e` is accepted for a modulus range legend.

If a `hist.faces` is defined, these are used to connect points of the parametric study.

For an example, see `fe2xf PoleRange`.

`zCoef`

This command is used to generate the weighting coefficients in (4.7), more details are given in section 4.4.2 .

`model=fe2xf('zcoef -default',model);` is used to analyze the model and define a default `info,zCoef` entry. `fe2xf('zcoef',model);` displays the values. If a parameter range is defined, you can specify the parameter to use with the `-jpar i` option and all parameters with `-jpar -1`.

`DefList`

When reading shapes for the purpose of generating reduced models, `def=fe2xf('DefList','root*.mat')` reads all files matching `root*.mat` and combines the shapes in a `v_handle` `def.def` field. Selection of diameters and frequency range during the read process is performed using

```
RO=struct('Fmax',8000);
d1=fe_cyclicb('DefList','root*.mat',RO);
```

Optional argument `RO` can have following fields

- `.Fmax` defines maximal frequency of retained vectors.
- `.Fmin` defines minimal frequency of retained vectors.
- `.First6RB` defines upper frequency tolerance of rigid body modes. Then only the first 6 occurrences of rigid bodies are kept and next removed.

`Build[,TrEnh,ListR]`

Utilities for reduced parametric model (MVR) generation. The parametric model is assumed to be a pre-assembled parametric model. It consists in a model containing assembled matrices, the assembled matrices can be split into different contributions relative to specific parameters, see `fe_case par` for parameter definition and `mattyp` for reference on split assemblies.

The multi-model reduction then considers an orthonormalized basis concatenating the real modal bases of a subset of chosen design points, possibly enhanced with inelastic contributions.

- `BuildList,`*R*

  Command `BuildList` packages the input to command `Build`. It aims at defining the subset of design points used for the modal basis reduction. The set of design points is in the `Range` format (see `sdtweb fe_range`).

  The default syntax is `[RunOpt,zCoef]=fe2xf('BuildList',model,RunOpt)'`, with `model` a SDT model with pre-defined matrices and resolved `Case`. `RunOpt` is a structure providing options, with at least field `.EigOpt` that defined the real mode basis computation strategy. By default, this corresponds to the options vector of `fe_eig` , but it can also be the content of the `Mode` tab (`sdtroot initMode`), or a cell array for callback definition.

  The `Range` data must be defined, either with the direct `Range` format or as a cell array of parameters in the `fe_range Vect` format. This data can exist at several places, it is sequentially searched as

  - the `.Range` field in `RunOpt`
  - the `info,Range` stack entry in `model`
  - the `info,Range0` stack entry in `model`, this entry usually containing the cell array of `RangeVect` parameters.

  By default, it is assumed that the provided `Range` data is the parameter subset so that all points will be used to generate the reduction list.

  Command `BuildListR`*stra*, will consider a reduction of a global `Range` entry by generating new points based on the parameter variations. The syntax is the same, `RunOpt=fe2xf('BuildListRM`
  and *stra* is a strategy token to define chosen design points. This is based on `fe_range BuildR` functionality. Command option `-given` forces the use of the provided `Range` for the list. Command option `-flip` will flip the design points list. As for some procedures the list first point defines the mass and stiffness used for normalization this can have an impact on the reduction basis.

  Among all the generated design points (1+the number of varying parameters) only the ones with a variation on elastic matrices (types `2, 20 1, 5`, see `mattyp` are kept. Command option `allp` forces to keep all design points is implicit variations are present.

  Command option `-First`*typ* will add to the end of the list a series of first order corrections for the provided matrix types in *ty*.

  The output is the input structure `RunOpt` with additional field `.list` compatible with `Build` command. The second output is the `zCoef` data for the model, see `sdtweb zCoef`.

- `Build`

  This command generates a multi-model reduction basis, and can generate the associated reduced model (`MVR`).

  To work with `feplot` the syntax is `fe2xf('build',cf,RO)'`, with `cf` is a handle to the feplot figure that contains the initial model. In such case, the built reduced model is stored in `cf.Stack{'SE','MVR'}`. If `cf` is replaced by a standard SDT model, the output is directly the reduced model: `MVR=fe2xf('build',model,RO)'`.

  `RO` is a data structure with fields defining the options

  - `.list` cell array that defines reduction operations (see below). One can also rebuild MVR from existing def providing `RO.list` as a cell array of `'file` *`fname`*`'` strings, or cell array of def data structures.
  - `.Range` is an alternative way to define the list of reduction operation, see `BuildList`. As many mode computations (according to options defined in the `RO.EigOpt` field) as rows in `RO.Range.val` are performed. See `fe_range` `BuildR` and `fevisco` `Range` for how to define a range data structure (field `.val` with as many rows as design points and as many columns as matrices, field `.lab` is a cell array with string labels for each column).
  - `.ListR` if no `.list` is present it is possible to call `BuilListR` on-the-fly by providing this field. If set to one, the nominal `ListR` is called, but one can also provide a string command, in which case a command is generated with `[BuildListR RO.ListR]`.
  - `.EigOpt` defines mode computation options (see `fe_eig`).
  - `.Save` contains 1 if mode computation results are to be saved during the reduction process. If 2 the MVR is not built and modes are stored for delayed build call. If 0 or absent modes are not saved.
  - `.Root` is a string that begins the filenames where modes are saved (if `RO.Save=` 1 or 2).
  - `.NoT` if set to 1, the reduction basis is output, and no model projection is performed. This allows for further reduction basis handling, such as `BuildTrEnh`.
  - `.setjPar` (in combination with presence of `.Range`) will look in the model `nmap` for key `Map:CurPar` or `CurParHandle` that defines an `nmap` object used by implicit matrices. If that is the case, incrementation in the list will also trigger parameter changes in the `nmap`.

  Basis building calls in `.list`, the cell can contain

  - `eig EigOpt` entries in first column and the coefficients to be used to generate the current stiffness in the second column.
  - `first` entries in first column to generate a first order correction based on the last eigenvalue problem solved and stiffness defined by coefficients given in the second column.

- `file fname` reads subspace from `def.def` in `fname`. This can be used in conjunction with a first run where data is saved using `RO.Root=fullfile(sdtdef('tempdir'),'tpMVR');RO.Sa`
- `def` precomputed def.

For use with `fe_reduc` use an `info,Fe2xfBuild` entry.

Following example builds a reduced model using `RO.list`:

```
model=d_visco('MeshCantilever'); cf=feplot(model);
mat=m_visco('convert INSI',m_visco('database Soundcoat-DYAD609'))
cf.mdl=fevisco('addmat 101',cf.mdl,'visco',mat);
RO=struct('list',{{'eig 5 10 0',[0 1  1];
                   'eig 5 10 0',[0 1 .1];
                   'first',     [0 0  1]}});
fe2xf('build',cf,RO);
```

Another example with a call using a parameter Range:

```
model=d_visco('MeshCantilever'); cf=feplot(model)
mat=m_visco('convert INSI',m_visco('database Soundcoat-DYAD609'))
cf.mdl=fevisco('addmat 101',cf.mdl,'visco',mat);
Range=struct('val',[0 1 1;0 1 .1],'lab',{{'m' 'k' 'visco'}})
RO=struct('EigOpt',[5 10 0],'Range',Range);
fe2xf('build',cf,RO);
```

When starting from a model without pre-defined matrices (`cf.mdl.K` does not exist), one assembles mass, stiffness and parameter matrices `assemble -matdes 2 1 -1 -se NoT` and removes the nominal parameter matrices from the base stiffness. The nominal model is thus associated with coefficients 1 for all parameters.

- `BuildTrEnh`

  This command handles reduction basis enhancement adapted to inelastic contributions. Starting from a reduction basis T, one generate the enhanced basis

$$T_1 = \left[T K_S^{-1} K_{i1} T ... K_S^{-1} K_{it} T\right]_{Orth} \tag{5.9}$$

  where K_{it} is the sum of all matrices found of type t (or each matrix with option `diffMat`, K_S is a symmetric positive definite matrix (usually a shifted stiffness matrix), and orthonormalization uses a `fe_norm` call with a mass and stiffness matrix.

  The syntax is `TR=fe2xf('BuildTrEnh',TR,model,Case,RO,kd,K)'`, with `TR` a reduction basis in the format, `model` a SDT model with pre-defined matrices (see command `Build`), `Case` the resolved case entry associated to the assembled modal, `RO` if an option structure with field

`.enhtyp` containing the matrix types to be considered as a line vector, see `mattyp`, `kd` is a matrix factor (see `ofact`) representing K_S^{-1}, and `K` is a 2x1 cell array containing a mass and stiffness matrix for normalization.

Entries `kd` and `K` can be omitted. In such case, `kd` will be generated as the shifted elastic matrix of the model, the shift being used as found in either the `info,EigOpt` model entry (see `fe_eig`) or 1000 if undefined. If `K` is undefined, `K1` will be the system mass matrix (types 2 or 20) and `K2` will be elastic stiffness matrix (types 1 or 5).

Structure `RO` can contain other optional fields that if undefined can also be passed as command options,

- `.clearKd`, if non-null the provided `kd` factor will be cleared in the procedure.
- `.norm`, set to 0 by default, the generated basis will not be orthonormalized, set to 1 orthonormalization will be performed sequentially after enhancement of each matrix type, set to 2 one global orthonormalization will be performed after the total reduction basis concatenation.
- `.diffMat` set to 0 by default. Set to 1 consider orthonormalization for each matrix separately instead of summing each type (necessary when handling matrices to interpolate).
- `.MVR`, to directly generate the reduced model based on `model` as an output. The reduction basis is then stored in `MVR.TR`. *e.g.* `MVR=fe2xf('BuildTrEnh-MVR-norm',TR,model,Case,4);`.
- `.NoT`, to output a reduction basis on `Case.DOF` instead of `model.DOF`.

See also

`fe2ss`, `fevisco`, section 4.4.5

fevisco

Purpose

User interface function for support of viscoelastic materials

This function is part of the viscoelastic tools.

Syntax

```
out = fevisco('command',model);
```

Description

The solid and fluid models typically used for viscoelastic studies are described in section **??** , where sample applications are treated.

AddMat,Par2Visc

`model=fevisco('addmat MatId',model,'NameForMatId',mat);` is used to properly append a viscoelastic material with the given ***MatId*** to the model. During this definition, one sets the reference elastic material in `model.pl`, the viscoelastic material definition in `model.Stack{'mat','NameForMatId'}` and a parameter definition in `model.CStack{'par','NameForMatId'}`.

If `mat.pl` does not contain a reference material, one is created using a call to `m_visco RefMat`.

The tag `NameForMatId` will be used for parametric model generation and `fe2xf zCoef` calls, if you change material you should just modify the `model.Stack{'mat','NameForMatId'}` data but not the parameter or the reference material. `fevisco('Par2Visc',cf)` uses the `NameForMatId` tag to check the existence of viscoelastic materials associated with matrices and modifies the `zCoefFcn` entries accordingly.

```
cf=d_visco('MeshplateLoadMV feplot');
cf.Stack{'info','Freq'}=[500:10:4000]';
cf.Stack{'info','Range'}=[10 20]';
% define material with a unit conversion
mat=m_visco('convert INSI',m_visco('database Soundcoat-DYAD609'));
%                 MatId for original, name of parameter
cf.mdl = fevisco('addmat 101',cf.mdl,'Constrained 101',mat);
fevisco('par2visc',cf);cf.Stack{'zCoef'}
fe2xf('zcoef',cf)
```

The command can also be used with reduced models, so set a viscoelastic parameter variation (`_visc` entry in `zcoef`) for a stiffness parameter that is already existing.

`Fluid [merge,matrix,makereduced, ...]`

These commands are meant to allow the generation of a coupled fluid/structure model based on a reduced model used for viscoelastic structure predictions (result of a `fe2xf direct` command for example).

A complete example is treated in section 4.6.2 . The steps are

- add the fluid as a superelement to the solid model
  `mCP=fevisco('fluidmerge',modelS,modelF)`.
- assembly of the fluid/structure coupling and fluid matrices with the `FluidMatrix` command `fevisco('FluidMatrix',cf, 'FluidInterfaceSel','SolidInterfaceSelection')`. The fluid interface selection applies to the elements of the `fluid` superelement renumbered consistently for the combined model.

  The result is stored in the `feplot` model stack as two assembled superelements which you can access with `cf.Stack{'fluid'}` and `cf.Stack{'FSC'}`. When built with `fevisco` one also has `coup.Node,coup.Elt` with the fluid interface as `fsc` elements and the solid interface as `quad4` elements to allow verification of the matching.

  Note that the `coup` matrix can also be imported from an external reference code that often exists in the automotive industry. `fevisco('FluidMatrix',cf,coup)` with `coup.K, coup.sdof, coup.fdof` describing the coupling matrix, solid and fluid DOFs. Solid DOFs correspond to rows of `coup.K`, fluid DOFs to columns.

  The `fevisco('fluidCheckCon',cf)` can be used to check the connectivity of the coupling matrix. The distance between each coupled fluid node and structural nodes involved in the coupling is computed and the associated links are shown for verification.
- compute the fluid modes and generate a reduced coupled model

  `[mRCoup,dflu]=fevisco('fluidMakeReduced',mCP,dsol);`

  which can then be used for predictions with `fe2xf` as illustrated in section 4.6.2 .

`Feplot`

`fevisco('feplotEnerK','matid1001','matid1 2 3');` displays the strain energy of the viscoelastic part of the structure (first selection here `MatId 1001`) as one object and the rest of the structure (element selection in third argument) as a wire-frame.

`feplotStress` provides stress cuts within volume. Accepted options are

- `MatId` desired material

- DefLen max arrow length
- Rule integration rule
- v element direction : 1 for x, 2 y, 3 z,4 normal for z revolution layer.

```
PA=d_visco('Script Comp13Cantilever');
comp13('solveEFracNom');  % comgui imwrite
cf=feplot(PA.MVR,PA.MVR.TR);
RA=struct('MatId','101','DefLen',.01,'rule',-1,'v',3, ...
 'Arrow','Arrow');
fevisco('feplotstress',PA.MVR,RA)
```

MakeModel [Matid i]

A parametric model is described by a type 1 superelement as detailed in section 4.4.5 .

The MakeModel command generates this model based on a type 3 superelement where parameters are set (see upcom Par commands, one then generates a model for parametric studies) or where viscoelastic materials have been properly declared using fevisco addmat commands (note that you can start with parameters and use the par2visc command to declare those parameters to correspond to viscoelastic materials, see section 4.4.2). You can also parameterize elastic materials using 'MakeModel MatId *i*' where the additional elastic materials are given by *i*. The example given in section 4.5.6 , uses commands

```
d_visco('MeshPlate up');cf=feplot;Up=cf.mdl;% This is a simple test case
Up=stack_rm(Up,'mat');
Up = fevisco('addmat 101',Up,'First area','ISD112 (1993)');
Up = fevisco('addmat 103',Up,'Second area','ISD112 (1993)');
MV = fevisco('makemodel matid 101 103',Up);
```

If you have a reduced basis in def, MVR = fevisco('makemodel matid 101 103',Up,def); is used to generate a reduced model. See section 4.5.6 .

The resulting data structure MV contains all the stack entries needed for viscoelastic response computations (see section 4.4.5) : parameters defined in the case stack and viscoelastic materials defined as mat entries and as elastic materials in MV.pl (this is needed to define the reference modulus value, the selection of a reference E or G is based on the content of mat.nomo.

`[Write,Build] Step12`

This command is used to generate parameterized reduced models in one NASTRAN run as detailed in section 4.8 . From the command line, you can use `fevisco('WriteStep12 -run',cf.mdl)`, to start the job and `fevisco('BuildStep12',cf.mdl)` to build the parametric model.

With the command option `-write`, `writestep12` command writes the bulk model before starting job.

For in MATLAB operation you can use the `fe2xf DirectFirst` command.

`MakeSandwich` *`layers`*

Tools for the generation of multi-layer sandwich models.

```
fevisco('makesandwich (layer generation)',FEnode,FEel0,treated,MAPN);
```

For each layer, the `makesandwich` command specifies

- element nature (shell or volume)
- desired material property for the meshed layer (for the starting layer use `0`).
- thickness of volumes, and distances of faces to neutral fiber for shells. For shells, one has two distances from the bottom layer to the neutral axis and from the neutral axis to the top layer.

The supporting model can be specified by its nodes and elements as shown above or using a model data structure. `Treated` is an optional FindNode command that allows generation of a sandwich for a part of the original model only.

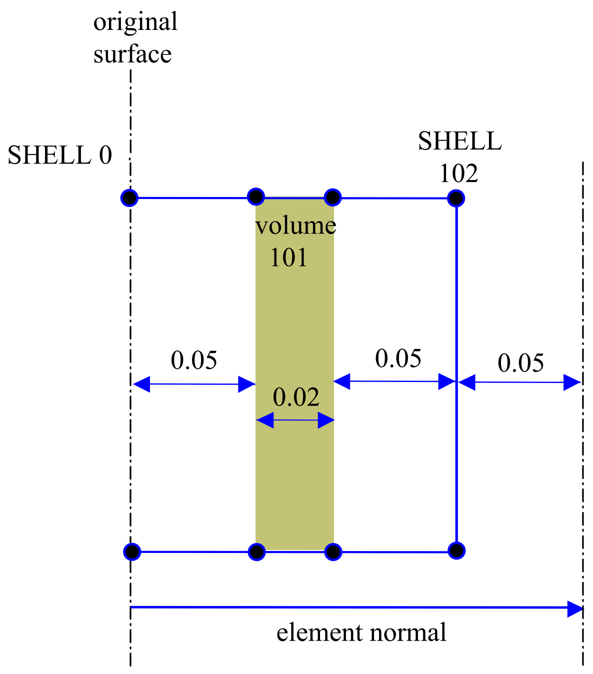


Figure 5.1: Example of a MakeSandwich command

For example, the generation of a three layer sandwich with the original layer 0.01 thick (leading to a 0.005 offset), a volume of thickness 0.002, and a second 0.01 thick shell looks like

```
femesh('reset');
femesh(';testquad4');FEel0=feutil('orient 1 n 0 0 1',FEnode,FEel0);
femesh('divide',linspace(0,1,10),linspace(0,1,12));
sandCom='makesandwich shell 0 0 .005 volume 101 .002 shell 102 -.005 .005';
treated='withnode{x>.5 & y>.5}';
[FEnode,FEelt]=fevisco(sandCom,FEnode,FEel0,treated);
femesh('plotelt');fecom('colordatamat');
```

Offsets are handled using rigid links between the shell neutral fiber and upper/lower surfaces. By default element normals at the center are used to define thickness, you can also use normals at node by inserting a `-node` option in the command.

Notes on `sandCom` format:

The first layer uses `0` material property : it means that the original material property of the first layer is used. For the volume layer, `101` is used and only the thickness needs to be specified (`.002`). The last layer is a shell (property `102`) with a thickness of `.001` with an offset of `.005`.

The command allows more accurate control of normals used for the sandwich generation. Nominally the normals are generated using the commands

```
MAPE = feutil('get normalmap',node,elt);
MAPN = feutil('get normalmap node',node,elt);
```

where `elt` is the element selection for the sandwich generation. You can provide your own maps using with a call of the form `fevisco(sandCom,FEnode,FEel0,treated,MAP)`. It is then expected that the provided `MAP` has an `MAP.opt` field where `opt(1)==1` leads to sandwich generation with offset at element center (element normal map) and `opt(1)==2` uses a normal map at nodes.

MatSplit

`model=fevisco('MatSplit MatId -type "val"',model);`
Split material *MatId* into multiple materials corresponding. Elements are then repeated.

- `svd` decompose constitutive law into 6 main directions
- `ortho` use 9 components of orthotropic material, `orthu` uses unit components so that coefficients to be used are the physical values
- `iso` uses a single coefficient for the material. `isoe` defines the constitutive law divided by E.
- `rot` split in radial and tangential components
- `EG` split compression and shear (1.16)
- `KG` split bulk and shear modulus (1.17)
- `MB` split membrane and bending in a shell
- `-ParType=1` used to define a type 1 (stiffness) parameter for each constitutive component in `ortho` and `rot`. `-ParType=5` for assembly of geometric stiffness (needed for material orientation handling).
- `.Group={`
  `zs'Comp',[1:3 7:9];'Shear',4:6}}` can be used to group components into a single parameter.
- `m-ortho` is used to define an additional homogeneous material for all elements. The associated `MatId` and `ProId` are assumed equal and should not be used by any initial element.

```
mo2=d_mesh('rve1fiber',struct('vf',.5,'mat','BerthCarDx','y',3,'ny',1));
mo2=fe_case(mo2,'remove','Matrix');
mo3=fevisco('matsplit MatId 1 -partype1 -type ortho',mo2);
[mo2,C2]=fe_case(mo3,'assemble -matdes -1 -SE -NoT');mo2
```

Utilities for energy display are available with `MatESplit -MatId` *`i`* `-cut` *`val`* .

Thermo

Computation of the power dissipated within a viscoelastic layer for forced response at a selected number of frequencies.

StrainMap

This is an experimental command to compute membrane strains with an offset to a given shell surface. This result can be used for placement.

```
    smap = fevisco('StrainMap OffSet',model,def,SurfaceSelection)
    fevisco('StrainPlot',smap,def)
```

Test

A few test cases are provided with `fevisco` to allow example documentation.

`TestPlate` generates the model of a square plate with partial coverage using constrained layer damping on one side and free layer damping on the other side.

`TestCantilever` generates the model of a cantilevered constrained layer damping treatment.

WriteInclude

`fevisco('WriteInclude` *`Selection`*`',model,FileName)` is used to generate a NASTRAN bulk with name `FileName` containing

- the associated nodes,
- the elements selected by *`Selection`*
- the rigid links and other case information connected to the nodes used by these elements
- the material and element property information used by the selected elements

This command is used to write the bulk for sandwich structures generated by MakeSandwich commands. Usually specific MatId are used to identify materials for the sandwich so that the selection is simply a list of material identifiers (for example 'WriteInclude MatId 1001 1002'). When called with a selection, an attempt to select the case information associated with the nodes of the selection is performed.

As an alternate format fevisco('WriteInclude',NewModel,OldModel,FileName) writes elements, nodes and properties of NewModel that are not in OldModel. Case entries of the NewModel are also written.

For example

```
model=d_visco('MeshPlate');
model=fe_case(model,'remove','Drilling');
tname=nas2up('tempname include.dat');
fevisco('writeinclude MatId 101 102',model,tname);
```

ceig, direct

The ceig command supports advanced complex eigenvalue solutions for constant viscous and/or hysteretic damping. It is normally called through fe_ceig where the solvers are documented.

The direct command supports various direct frequency response solvers. It is normally called through fe2xf where the solvers are documented.

NASTRAN

Utilities for interfacing with NASTRAN.

MVR.zCoefFcn=fevisco('nastranzCoefv1',model,MVR) builds the proper .zCoefFcn to reproduce jobs that would otherwise run as SOL108 with one viscoelastic material.

fevisco('nastranEtaGlob',model) return the global loss factor (known PARAM,G for NASTRAN) with proper default handling.

NastranEig implements a call to NASTRAN as fe_eig method 50. To use this capability, first use the call fevisco('NastranEig') that will set preferences.

See also

fe2xf, m_visco, section **??**

m_visco, mvisco_3m

Purpose

Material function for support of viscoelastic materials

This function is part of the viscoelastic tools.

Syntax

```
out = m_visco('command',model);
```

Description

Viscoelastic materials are described by a structure with fields

`.pl`	Reference elastic properties used for initial model assembly. This should include the loss factor.
`.name`	Reference name for the material.
`.type`	Always set to `m_visco` since this is the material handling function.
`.unit`	Unit system see `fe_mat convert`.
`.T0`	Reference temperature.
`.G`	Description of the modulus as a function of frequency. This can be a matrix with three columns giving frequencies, real and imaginary parts of the modulus, or a string that will be evaluated to determine the modulus. To sort frequencies use `mat.G=sortrows(mat.G)`.
`.at`	Description of the frequency shift as a function of temperature. This can be a matrix with two columns giving temperature and frequency shift α_T, or a string that will be evaluated to determine the shift factor.
`.nomo`	is a cell array containing information need to create the nomogram. See details under the `nomo` command.

Accepted commands are

`database,default,info,dbval`

These commands are used to select materials. `Info` lists available materials. `Default` gives the first material. `m_visco` searches its own database and for `mvisco_*.m` functions in the MATLAB search path. The `mvisco_3m` function is given to illustrate the contents of a user database file.

To start a selection process use `cf=feplot;m_visco('database',cf)`. This will give you a list of materials in the database. In the `Mat` tab, select materials you want to see, use the delete key (or the

`Remove material ...` context menu) to remove entries. You can also add materials to the stack manually with

```
m_visco('database -unit TM Densil',cf);
m_visco('database -unit TM Smactane 50_G',cf);
```

By default, the maximum loss factor and associated modulus and frequency are displayed for the material reference temperature. You can order the list by giving a target `InfoTarg freq temp sort` (frequency, temperature, sort by modulus `1` or loss factor `2`). Thus

```
>> m_visco('info wtarg 30 ttarg=20')
     CorningGlass0010   (G 2.87e+009 Pa e 0.02 3.00e+001 Hz 20 C)
ChicagoVitreous-CV325   (E 1.03e+008 Pa e 0.06 3.00e+001 Hz 20 C)
    Soundcoat-DYAD609   (G 6.21e+008 Pa e 0.08 3.00e+001 Hz 20 C)
   BarryControl-H-326   (G 5.90e+006 Pa e 0.08 3.00e+001 Hz 20 C)
...
```

You can also select your material by opening an `feplot` figure with all materials and selecting graphically. The `InfoTarg` command can then be used to select materials in a plots. For example giving a target frequency and a range of temperatures is achieved with

```
cf=feplot;cf.mdl='reset';
m_visco('database',cf);% open FEPLOT figure with database
RO=struct('wtarg',20,'ttarg',[-10 5 20 40 60 80]');
mat=[stack_get(cf.mdl,'mat','#ISD'); % Get selected materials
     stack_get(cf.mdl,'mat','#Smac')];
cf.mdl.Stack=mat;
% Once you have all desired entries, redraw
m_visco('info -modmin 1e6 -etamin .2 -cf1',cf,RO);
```

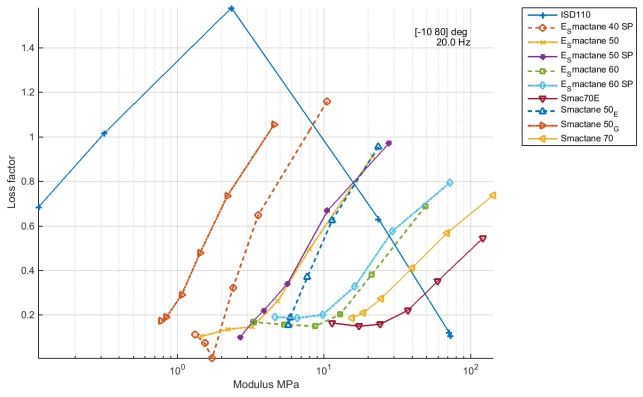


Figure 5.2: Material selection diagram.

`mat=m_visco('database Name')` tries to find an optimal match for *Name* in the database and returns the associated material. `mat=m_visco('database -unit TM Name')` converts to unit system TM (see `fe_mat Convert`). Command option `-matid` can be used to specify a `MatId` for the material. Without output argument material is added to the stack of the feplot figure (one can give the `cf pointer` as a second input argument). If *Name* is not given, all materials are returned.

`mat=m_visco('dbval')` is used to generate equivalent elastic materials. This is not currently functional.

The `3ParWLF` material is a standard viscoelastic model with WLF type temperature dependence. Its constitutive law is the characterized by

$$G = G_0 \frac{1 + i\omega_R/z}{1 + i\omega_R/p} \text{ and } \omega_R = \omega\alpha_T = \omega 10^{-c_1*(T-T_0)/(c2+T-T_0)} \tag{5.10}$$

The parameters retained to characterize this model are `G0, etamax, wmax, c1, c2, T0`.

DispMat

`m_visco('dispmat',mat)` is used to generate a text display of a viscoelastic material describe by the `mat` data structure.

RefMat

`m_visco('refmat',mat,model)` uses data on temperature range and frequency in the model to create a

nomo

The `nomo` command generates a standard nomogram plot. The following entries in the cell array `mat.nomo` are used. If using a `mat.G` table that contains raw experimental data that is non smooth, `mat=m_visco('nomo -smooth',mat)` will generate a smooth table.

Eeta	gives the log10 of Emin,Emax,etamax. Note that for units in Pa, a shift to MPa is performed in the plot.
w	gives the log10 of Wmin,WRmin,WRmax. Increasing Wmin shifts the show isotherms right.
file	gives the name of the bitmap file used to digitize the nomogram. When this file is in the current directory, the nomogram and the bitmap are overlaid. This allows editing of tabular values as described under fevisco handnomo.
T	is used to specify isotherm positions in the nomogram.

For example

```
m_visco('nomo','isd112 (1999)',4)
mat=m_visco('isd112 (1993)');
mat=sdsetprop(mat,'nomo','w',[-1 0 12],'Eeta',[4 8 2],'type','G', ...
 'unit','SI','T',[-20:20:120]);
m_visco('nomo',mat)
```

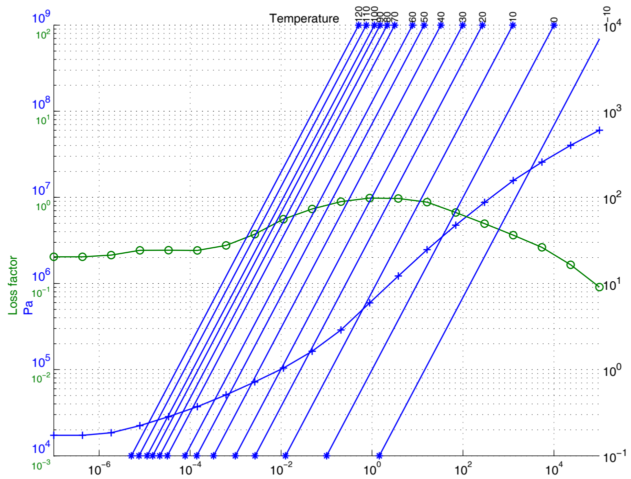


Figure 5.3: Nomogram of the ISD112.

at

The at command is used to generate shift factor using calls of the form

```
at=m_visco('at',T,mat)
```

where mat can be a data structure or a string matched against the m_visco database.

`interp [,-unit SI] [,showrange]`

The `interp` command is used to generate the modulus at arbitrary frequency-temperature design points using calls of the form `G=m_mvisco('interp',w,T,mat)` where `mat` can be a data structure or a string matched against the `m_visco` database. Accepted options are

- `-unit SI` an optional unit conversion code (see `fe_mat('convert')` can be given when needed.
- `ToE` uses the assumption that $G = \frac{E}{E(1+\nu)}$ to estimate the Young modulus.
- `-rel` generates the relative coefficient such that $G(\omega, T) = \text{coef} G_0$.
- `ShowRange` generates a standard display giving modulus variations for a given frequency/temperature range. This allows the user to validate the usefulness of a particular material for the operating conditions given by this range.
- `cf=1` selects the figure for display
- `yscale=log,lin` selects linear or log scale for display.

For example

```
w=logspace(1,3,100);  % default : cf.Stack{'info','Freq'} in Hz
T=[0:10:50];          % default : cf.Stack{'info','Range'}
mat=m_visco('isd112 (1993)');
G=m_visco('interp -unit MM',w,T(:),mat);
m_visco('interp ShowRange',w,T(:),mat);
%m_visco('interp',cf,'MatName') is also acceptable
```

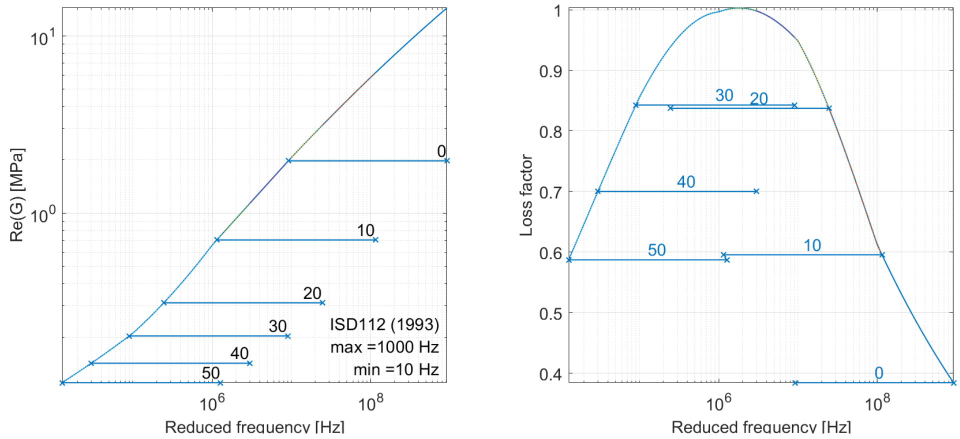


Figure 5.4: Modulus vs. frequency and temperature. Horizontal blue lines show the physical frequency range (indicated in the legend) at the temperature shown by the text.

WriteNastran

This utility is meant to help users generate complex modulus tables for use in NASTRAN. The tables are generated as two TABLED1 bulk entries if only one ID is specified in the command, the real and imaginary parts are written with sequential numbers. The command specifies the table identifier (default 1) and file name. Note that this only supports a single temperature, since NASTRAN does not handle multi dimensional tables.

For more details on writing TABLED1 entries see naswrite WriteCurve.

```
fname=horzcat(nas2up('tempname'),'.blk');
m_visco(horzcat('WriteNastran 101 201',fname), ...
  10:10:100,10,'isd112 (1993)');
type(fname)
```

HandNomo FileName

Displays the bitmap image of a nomogram and guides the user through the process needed to obtain a tabular version suitable for inclusion in m_visco. The main steps of the procedure are as follows

- Rect : define the location of the axis in the bitmap by selecting two of its corners.
- Elim : click on the text of ytick labels on the left of the axis to define the modulus range and the maximum axis value for the loss factor range.

- `wlim` : click on the text of `ytick` labels on the right of the axis to define the minimum frequency displayed on the true frequency axis (left y axis) and the min and max values for the reduced frequency (x axis).
- `at` : click on the isotherm that is always generated to enter the isotherm edition mode. You can then use the following keys to edit the `at` values. **n** to add new isotherm values. Left and right arrows to move the offset of the current isotherm. Up and down arrow to select a different isotherm.

  If the isotherm slope is not correct, check the `Elim` and `wlim` values and possibly adjust the axis rectangle using : **uU** to move the rectangle along -x or +x; **vV** to move the rectangle along -y or +y; **xXyY** to resize the rectangle.
- `eta` click on the green line with circle markers. Adjust values with your mouse. Use left and right arrows to select the points and `q` to exit.
- `G` click on the blue line with circle markers. Adjust values with your mouse. Use left and right arrows to select the points and `q` to exit.

At any point during the procedure, you can press the **i** key to generate a screen printout of the material. This can then be included in `m_visco` or a `mvisco_*.m` database file.

With a material saved in a database, can superpose the original figure and the nomogram using `m_visco('HandNomo MatName')` while having the image file in the current directory.

See also

`fe2xf`, `m_visco`, section **??**

nasread

Purpose

Read results from outputs of the MSC/NASTRAN finite element code. This function is part of FEMLink.

Syntax

```
out = nasread('FileName','Command')
```

Description

`nasread` reads bulk data deck (NASTRAN input), direct reading of model and result information in OUTPUT2 and OUTPUT4 files generated using NASTRAN `PARAM,POST,-i` cards. This is the most efficient and accurate method to import NASTRAN results for post-processing (visualization with `feplot`, normal model handling with `nor2ss`, ...) or parameterized model handling with `upcom`. Results in the `.f06` text file (no longer supported).

Available commands are

`Bulk file`

`model=nasread('FileName','bulk')` reads NASTRAN bulk files for nodes (grid points), element description matrix, material and element properties, and coordinate transformations, MPC, SPC, DMIG, SETS, ...

Use `'BulkNo'` for a file with no `BEGIN BULK` card. Unsupported cards are displayed to let you know what was not read. You can omit the `'bulk'` command when the file name has the `.dat` or `.bdf` extension.

Each row of the `bas.bas` output argument contains the description of a coordinate system.

The following table gives a *partial conversion list*. For an up to date table use `nas2up('convlist')`

NASTRAN	*SDT*
CELAS1, CELAS2, RBAR	celas
RBE2	rigid
RBE3	rbe3 in Case
CONROD	bar1
CBAR, CBEAM, CROD	beam1
CBUSH	cbush
CSHEAR	quad4
CONM1, CONM2	mass2
CHEXA	hexa8, hexa20
CPENTA	penta6, penta15
CTETRA	tetra4, tetra10
CTRIA3, CTRIAR	tria3
CTRIA6	tria6
CQUAD4, CQUADR	quad4
CQUAD8	quadb

Details on properties are given under naswrite WritePLIL. NASTRAN Scalar points are treated as standard SDT nodes with the scalar DOF being set to DOF .01 (this has been tested for nodes, DMIG and MPC).

OUTPUT2 binary

model=nasread('FileName','output2') reads output2 *binary output format* for tables, matrices and labels. You can omit the output2 command if the file names end with 2. The output model is a model data structure described in section **??** . If deformations are present in the binary file, the are saved OUG(i) entries in the stack (see section **??**). With no output argument, the result is shown in feplot.

Warning: do not use the FORM = FORMATTED in the eventual ASSIGN OUTPUT2 statement.

The optional `out` argument is a cell array with fields the following fields

`.name`	Header data block name (table, matrix) or label (label)
`.dname`	Data block name (table, matrix) or NASTRAN header (label)
`.data`	cell array with logical records (tables), matrix (matrix), empty (label)
`.trl`	Trailer (7 integers) followed by record 3 data if any (for table and matrix), date (for label)

Translation is provided for the following tables

`GEOM1`	nodes with support for local coordinates and output of nodes in global coordinates
`GEOM2`	elements with translation to SDT model description matrix (see `bulk` command).
`GEOM4`	translates constraints (`MPC`, `OMIT`, `SPC`) and rigid links (`RBAR`, `RBE1`, `RBE2`, `RBE3`, `RROD`, ...) to SDT model description matrix
`GPDT`	with use of `GPL` and `CSTM` to obtain nodes in global coordinates
`KDICT`	reading of element mass (`MDICT`, `MELM`) and stiffness (`KDICT`, `KELM`) matrix dictionaries and transformation of a type 3 superelement handled by `upcom`. This is typically obtained from NASTRAN with `PARAM,POST,-4`. To choose the file name use `Up.file='FileName';Up=nasread(Up,'Output2.op2');`
`MPT`	material property tables
`OUG`	transformation of shapes (modes, time response, static response, ...) as `curve` entries in the stack (possibly multiple if various outputs are requested). Note : by default deformations are in the SDT global coordinate system (basic in NASTRAN terminology). You may switch to output in the local (global in NASTRAN terminology) using `PARAM,OUGCORD,GLOBAL`. To avoid *Out of Memory* errors when reading deformations, you can set use a smaller buffer `sdtdef('OutOfCoreBufferSize',10)` (in MB). When too large, `def.def` is left in the file and read as a `v_handle` object that lets you access deformations with standard indexing commands. Use `def.def=def.def(:,:)` to load all. To get the deformation in the stack use calls of the form `def=stack_get(model,'curve','OUG(1)','get')`
`OEE`	tables of element energy
`OES`	tables of element stresses or strains.

This translation allows direct reading/translation of output generated with NASTRAN `PARAM,POST` commands simply using `out=nasread('FileName.op2')`. For model and modeshapes, use `PARAM,POST,-1`. For model and element matrices use `PARAM,POST,-4` or `PARAM,POST,-5` (see `BuildUp` command below).

BuildUp,BuildOrLoad

A standard use of FEMLink is to import a model including element matrices to be used later with `upcom`. You must first run NASTRAN SOL103 with `PARAM,POST,-4` to generate the appropriate `.op2` file (note that you must include the geometry in the file, that is not use `PARAM,OGEOM,NO`). Assuming that you have saved the bulk file and the `.op2` result in the same directory with the same name (different extension), then

```
Up=nasread('FileName.blk','buildup')
```

reads the bulk and `.op2` file to generate a superelement saved in `FileName.mat`.

It is necessary to read the bulk because linear constraints are not saved in the `.op2` file during the NASTRAN run. If you have no such constraints, you can read the `.op2` only with `Up=upcom('load FileName);Up=nasread(Up,'FileName.op2')`.

The `BuildOrLoad` command is used to generate the `upcom` file on the first run and simply load it if it already exists.

```
nasread('FileName.blk','BuildOrLoad') % result in global variable Up
```

OUTPUT4 binary

`out=nasread('FileName','output4')` reads `output4` *binary output format* for matrices (stiffness, mass, restitution matrices ...). The result `out` is a cell array containing matrix names and values stored as MATLAB sparse matrices.

All double precision matrix types are now supported. If you encounter any problem, ask for a patch which will be provided promptly.

`Output4` text files are also supported with less performance and no support for non sequential access to data with the SDT v_handle object.

Supported options

- `-full` : assumes that the matrix to be read should be stored as full (default sparse).
- `-transpose` : transpose data while reading.
- `-hdf` : save data in a hdf file. Reading is performed using buffer (`sdtdef('OutOfCoreBufferSize',100)` for a 100MB buffer). It is useful to overcome the 2GB limit on 32 bit Matlab: see `sdthdf` for details about how to build v_handle on hdf file.

In NASTRAN to generate matrices you should output to OP4 files. You should have a card to specify the output file `ASSIGN INPUTT4='job_kv.op4',STATUS=NEW,UNIT=40,DELETE` then in your DMAP alter `OUTPUT4 KGG,MGG,,,//0/40/1 $ Output matrices`

`.f06 output (obsolete)`

ASCII reading in `.f06` files is slow and often generates round-off errors. You should thus consider reading binary OUTPUT2 and OUTPUT4 files, which is now the only supported format. You may try reading matrices with `nasread('FileName','matprt')`, tables with `nasread('F','tabpt')` and real modes with

```
[vector,mdof]=nasread('filename','vectortype')
```

Supported vectors are displacement (`displacement`), applied load vector (`oload`) and grid point stress (`gpstress`).

See also

`naswrite`, `FEMLink`

naswrite

Purpose

Formatted ASCII output to MSC/NASTRAN bulk data deck. This function is part of FEMLink.

Syntax

```
naswrite('FileName',node,elt,pl,il)
naswrite('FileName','command', ...)
naswrite('-newFileName','command', ...)
naswrite(fid,'command', ...)
```

Description

naswrite appends its output to the file FileName or creates it, if it does not exist. Use option -newFileName to force deletion of an existing file. You can also provide a handle fid to a file that you opened with fopen. fid=1 can be used to have a screen output.

EditBulk

Supports bulk file editing. Calls take the form
nas2up('EditBulk',InFile,edits,Outfile), where InFile and OutFile are file names and edits is a cell array with four columns giving command, BeginTag, EndTag, and data. Accepted commands are

Before	inserts data before the BeginTag.
Insert	inserts data after the EndTag.
Remove	removes a given card. Warning this does not yet handle multiple line cards.
Set	used to set parameter and assign values. For example

```
edits={'Set','PARAM','POST','-2'};
rootname='my_job';
f0={'OUTPUT4',sprintf('%s_mkekvr.op4',rootname),'NEW',40,'DELETE',
    'OUTPUT4',sprintf('%s_TR.op4',rootname),'NEW',41,'DELETE'};
edits(end+1,1:4)={'set','ASSIGN','',f0}
```

When writing automated solutions, the edits should be stored in a stack entry info,EditBulk.

model

`naswrite('FileName',model)` the nominal call, it writes everything possible : nodes, elements, material properties, case information (boundary conditions, loads, etc.). For example `naswrite(1,femesh('testquad4'))`.

The following information present in model stack is supported

- curves as `TABLED1` cards if some curves are declared in the `model.Stack` see `fe_curve` for the format).
- Fixed DOFs as `SPC1` cards if the model case contains `FixDof` and/or `KeepDof` entries. `FixDof,AutoSPC` is ignored if it exists.
- Multiple point constraints as `MPC` cards if the model case contains `MPC` entries.
- coordinate systems as `CORDi` cards if `model.bas` is defined (see `basis` for the format).

The obsolete call `naswrite('FileName',node,elt,pl,il)` is still supported.

`node,elt`

You can also write nodes and elements using the low level calls but this will not allow fixes in material/element property numbers or writing of case information.

```
femesh('reset');
femesh('testquad4')
fid=1 % fid=fopen('FileName');
naswrite(fid,'node',FEnode)
naswrite(fid,'node',FEnode)
%fclose(fid)
```

Note that `node(:,4)` which is a group identifier in SDT, is written as the SEID in NASTRAN. This may cause problems when writing models from translated from other FEM codes. Just use `model.Node(:,4)=0` in such cases.

`dmig`

`DMIG` writing is supported through calls of the form `naswrite(fid,'dmigwrite NAME',mat,mdof)`. For example

```
naswrite(1,'dmigwrite KAAT',rand(3),[1:3]'+.01)
```

A `nastran,dmig` entry in `model.Stack`, where the data is a cell array where each row gives name, DOF and matrix, will also be written. You can then add these matrices to your model by adding cards of the form `K2GG=KAAT` to you NASTRAN case.

`job`

NASTRAN job handling on a remote server from the MATLAB command line is partially supported. You are expected to have `ssh` and `scp` installed on your computer. On windows, it is assumed that you have access to these commands using CYGWIN. You first need to define your preferences

```
setpref('FEMLink','CopyFcn','scp');
setpref('FEMLink','RunNastran','nastran');
setpref('FEMLink','RemoteShell','ssh');
setpref('FEMLink','RemoteDir','/tmp2/nastran');
setpref('FEMLink','RemoteUserHost','user@myhost.com')
setpref('FEMLink','DmapDir',fullfile(fileparts(which('nasread')),'dmap'))
```

You can define a job handler customized to your needs and still use the `nas2up` calls for portability by defining `setpref('FEMLink','NASTRANJobHandler', 'FunctionName')`.

You can then run a job using `nas2up('joball','BulkFileName.dat')`. Additional arguments can be passed to the `RunNastran` command by simply adding them to the `joball` command. For example `nas2up('joball','BulkFileName.dat',struct('RunOptions','memory=1GB'))`.

It is possible provide specific options to your job handler by storing them as a `info,NasJobOpt`entry in your `model.Stack`. `nas2up('JobOptReset')` resets the default. The calling format in various functions that use the job handling facility is then

```
model=stack_set('info','NasJobOpt',nas2up('jobopt'));
nas2up('joball','step12.dat',model);
```

`RunOpt.RunOptions` stores text options to be added to the `nastran` command. `RunOpt.BackWd` can be used to specify an additional relative directory for the `JobCpFrom` command. `RunOpt.RemoteRelDir` can be used to specify the associated input for the `JobCpTo` command.

`nas2up('JobCpTo', 'LocalFileName', 'RemoteRelDir')` puts (copies) files to the remote directory or to `fullfile(RemoteDir,RemoteRelativeDir)` if specified.

`nas2up('JobCpFrom', 'RemoteFileName')` fetches files. The full remote file name is given by `fullfile(RemoteDir,RemoteFileName)`. Any relative directory is ignored for the local directory.

Here is a simple script that generates a model, runs NASTRAN and reads the result

```
wd=sdtdef('tempdir');

model=demosdt('demoubeam-2mat'); cf=feplot;
model=fe_case(model,'dofload','Input', ...
  struct('DOF',[349.01;360.01;241.01;365.03],'def',[1;-1;1;1],'ID',100));
model=nas2up('JobOpt',model);
```

```
model=stack_set(model,'info','Freq',[20:2:150]);

% write bulk but do not include eigenvalue options
naswrite(['-new' fullfile(wd,'ubeam.bdf')],stack_rm(model,'info','EigOpt'))

% generate a job by editing the reference mode.dat file
fname='ubeam.dat';
edits={'Set','PARAM','POST','-2';
 'replace','include ''model.bdf''','','include ''ubeam.bdf''';
 'replace','EIGRL','',nas2up('writecard',-1,[1 0 0 30],'ijji','EIGRL')};
nas2up('editbulk','mode.dat',edits,fullfile(wd,fname));
cd(wd);type(fname)
nas2up('joball',fname,model)
cg=feplot(4);mo1=nasread('ubeam.op2');
```

Wop4

Matrix writing to OUTPUT4 format. You provide a cell array with one matrix per row, names in first column and matrix in second column. The optional byte swapping argument can be used to write matrices for use on a computer with another binary format.

```
kv=speye(20);
ByteSwap=0;  % No Byte Swapping needed
nas2up('wop4','File.op4',{'kv',kv},ByteSwap);
```

For ByteSwap you can also specify ieee-le for little endian (Intel PC) or ieee-be depending on the architecture NASTRAN will be running on. You can omit specifying ByteSwap at every run by setting

```
setpref('FEMLink','OutputBinaryType','ieee-le')
```

WriteFreqLoad

edits=naswrite('Target.bdf','WriteFreqLoad',model) (or the equivalent nas2up call when the file is already open as show below) writes loads defined in model (and generated with Load=fe_load(model)) as a series of cards. FREQ1 for load frequencies, TABLED1 for the associated curve, RLOAD1 to define the loaded DOFs and DAREA for the spatial information about the load. The return edits argument is the entry that can be used to insert the associated subcase information in a nominal bulk.

The identifiers for the loads are supposed to be defined as Case.Stack{j1,end}.ID fields.

```
% Generate a model with sets of point loads
model=demosdt('Demo ubeam dofload noplot')
% Define the desired frequencies for output
model=stack_set(model,'info','Freq', ...
   struct('ID',101,'data',linspace(0,10,12)));
fid=1 % fid=fopen('FileName');
edits=nas2up('writefreqload',fid,model);
fprintf('%s\n',edits{end}{:}); % Main bulk to be modified with EditBulk
%fclose(fid)
```

Write[Curve,Set,SetC,Uset]

Write commands are used to WriteCurve lets you easily generate NASTRAN curve tables.

WriteSet lets you easily generate NASTRAN node and elements sets associated with node and element selection commands.

WriteSetC formats the sets for use in the case control section rather than the bulk.

WriteUset generates DOFs sets.

```
model=demosdt('demogartfe');
fid=1; % display on screen (otherwise use FOPEN to open file)
nas2up('WriteSet',fid,3000,model,'findnode x>.8');
selections={'zone_1','group 1';'zone_2','group 2:3'};
nas2up('WriteSet',fid,2000,model,selections);
st=nas2up('WriteSet',-1,2000,model,selections);

curves={'curve','Sine',fe_curve('testEval -id1 sin(t)',linspace(0,pi,10)) ; ...
        'curve','Exp.',fe_curve('testEval -id100 exp(-2*t)',linspace(0,1,30))};
nas2up('WriteCurve',fid,curves)
DOF=feutil('getdof',model);
nas2up('WriteUset U4',fid,DOF(1:20))
```

WritePLIL

The WritePLIL is used to resolve identifier issues in MatId and ProId (elements in SDT have both a MatId and an ProID while in NASTRAN they only have a ProId with the element property information pointing to material entries). While this command is typically used indirectly while writing a full model, you may want to access it directly. For example

```
model=demosdt('demogartfe');
nas2up('Writeplil',1,model);
```

- p_solid properties are implemented somewhat differently in NASTRAN and SDT, thus for a il row giving [ProID type Coordm In Stress Isop Fctn]

  In NASTRAN In is either a string or an integer. If it is an integer, this property is the same in il. If it is a string equal to resp. TWO or THREE, this property is equal to resp. 2 or 3 in il.

  In NASTRAN Stress is either a string or an integer. If it is an integer, this property is the same in il. If it is a string equal GAUSS, this property is equal to 1 in il.

  In NASTRAN, Isop is either a string or an integer. If it is an integer, this property is the same in il. If it is a string equal FULL, this property is equal to 1 in il.

  If Fctn is equal to FLUID in the NASTRAN Bulk file, it is equal to 1 in il and elements are read as flui* elements.

- MAT9 and m_elastic 3 differ by the order of shear stresses yz, zx, Gxy in SDT and xy, yz, zx in NASTRAN. The order of constitutive values is thus different, which is properly corrected in SDT 6.5.

See also

nasread, ufread, ufwrite

Bibliography

[1] J. Salençon, *Viscoélasticité.* Presse des Ponts et Chaussés, Paris, 1983.

[2] A. Nashif, D. Jones, and J. Henderson, *Vibration Damping.* John Wiley and Sons, 1985.

[3] C. Bert, "Material damping: An introductory review of mathematical models, measures, and experimental techniques," *Journal of Sound and Vibration*, vol. 29, no. 2, pp. 129–153, 1973.

[4] G. Golub and C. Van Loan, *Matrix computations.* Johns Hopkins University Press, 1983.

[5] D. McTavish and P. Hugues, "Finite element modeling of linear viscoelastic structures," *ASME Biennal Conference on Mechanical Vibration and Noise*, sep 1987.

[6] G. Lesieutre and E. Bianchini, "Time domain modeling of linear viscoelasticity using augmenting thermodynamic fields," *SDM Conference, AIAA paper 93-1550-CP*, pp. 2101–2109, 1993.

[7] E. Bianchini and G. Lesieutre, "Viscoelastic constrained-layer damping - time domain finite element modeling and experimental results," *SDM Conference, AIAA paper 94-1652-CP*, pp. 2666–2676, 1994.

[8] G. Lesieutre and E. Bianchini, "Time domain modeling of linear viscoelasticity using augmenting thermodynamic fields," *J. Vibration and Acoustics*, vol. 117, pp. 424–430, 1995.

[9] J. D'Azzo and C. Houpis, *Linear Control System Analysis and Design.* MacGraw Hill Book Company, 1988.

[10] F. Renaud, J. L. Dion, G. Chevallier, I. Tawfiq, and R. Lemaire, "A new identification method of viscoelastic behavior: Application to the generalized maxwell model," *Mechanical Systems and Signal Processing*, vol. 25, no. 3, pp. 991–1010, 2011.

[11] F. Schwartzl *Physica*, pp. 830–923, 1951.

[12] A. Lion, "On the thermodynamics of fractional damping elements," *Continuum Mech. Thermodyn.*, vol. 9, pp. 83–96, 1997.

[13] J. Ferry, *Viscoelastic Properties of Polymers.* Wiley, 2nd ed., 1970.

[14] Zilson, D. Kiureghian, and Bayo, "A replacement of the srss method in seismic analysis," *Earthquake Engineering and Structural Dynamics*, vol. 9, pp. 187–194, 1981.

[15] G. Kergourlay, *Mesure et prédiction de structures viscoélastiques - Application à une enceinte acoustique.* PhD thesis, Ecole Centrale de Paris, 2004.

[16] American Society for Testing and Materials, *E756-98 Standard Test Method for Measuring Vibration-Damping Properties of Materials*, 1998.

[17] H. Oberst and K. Frankenfeld, "Über die dämpfung der biegeschwingungen dünner bleche durch festhaftende beläge," *Acustica*, vol. 2, pp. 181–194, 1952.

[18] D. Ross, E. Ungar, and E. Kerwin, "Damping of plate flexural vibrations by means of viscoelastic laminates," *ASME*, vol. 51, 1959.

[19] E. Balmes, *Methods for vibration design and validation.* Course notes ENSAM/Ecole Centrale Paris, 1997-2012.

[20] T. Caughey, "Classical normal modes in damped linear dynamic systems," *ASME J. of Applied Mechanics*, pp. 269–271, 1960.

[21] J. Rayleigh, *The Theory of Sound.* Dover Publications, New-York, NY, 1945 (reedition).

[22] E. Balmes, *Modèles analytiques réduits et modèles expérimentaux complets en dynamique des structures.* Mémoire d'habilitation à diriger des recherches soutenue à l'Université Pierre et Marie Curie, 1997.

[23] R.-J. Gibert, *Vibrations des Structures.* Editions Eyrolles, Paris, 1988.

[24] E. Balmes, "New results on the identification of normal modes from experimental complex modes," *Mechanical Systems and Signal Processing*, vol. 11, no. 2, pp. 229–243, 1997.

[25] L. Rogers, C. Johnson, and D. Kienholz, "The modal strain energy finite element method and its application to damped laminated beams," *Shock and Vibration Bulletin*, vol. 51, 1981.

[26] D. Inman, *Engineering Vibration.* Prentice-Hall, Englewood Cliffs, N.J., 1994.

[27] W. Heylen, S. Lammens, and P. Sas, *Modal Analysis Theory and Testing.* KUL Press, Leuven, Belgium, 1997.

[28] D. Ewins, *Modal Testing: Theory and Practice.* John Wiley and Sons, Inc., New York, NY, 1984.

[29] EDF, *RCC-G : Règles de conception et de construction des îlots nucléaires REP*. EDF - Direction de l'Equipement Edition, Juillet 1988.

[30] L. Komzsik, "Implicit computational solutions of generalized quadratic eigenvalue problems," *Finite Elements In Analysis and Design*, vol. 37, pp. 799–810, 2001.

[31] G. Lesieutre and E. Bianchini, "Time domain modeling of linear viscoelasticity using augmenting thermodynamic fields," *SDM Conference, AIAA paper 93-1550-CP*, pp. 2101–2109, 1993.

[32] D. Golla and P. Hughes, "Dynamics of viscoelastic structures – a time domain finite element formulation," *Journal of Applied Mechanics*, vol. 52, pp. 897–906, 1985.

[33] ABAQUS/Standard, *User's Manual*, vol. 1. Hibbit, Karlsson, Sorhensen, Inc.

[34] L. Bagley and P. Torvik, "Fractional calculus - a different approach to the analysis of viscoelastically damped structures," *AIAA Journal*, vol. 21, no. 5, pp. 741–748, 1983.

[35] E. Balmes, "Modes and regular shapes. how to extend component mode synthesis theory.," *Proceedings of the XI DINAME - Ouro Preto - MG - Brazil*, March 2005.

[36] S. Rubin, "Improved component-mode representation for structural dynamic analysis," *AIAA Journal*, vol. 13, no. 8, pp. 995–1006, 1975.

[37] R. MacNeal, "A hybrid method of component mode synthesis," *Computers and structures*, vol. 1, no. 4, pp. 581–601, 1971.

[38] R. Guyan, "Reduction of mass and stiffness matrices," *AIAA Journal*, vol. 3, p. 380, 1965.

[39] R. J. Craig and M. Bampton, "Coupling of substructures for dynamic analyses," *AIAA Journal*, vol. 6, no. 7, pp. 1313–1319, 1968.

[40] A. Plouin and E. Balmes, "A test validated model of plates with constrained viscoelastic materials," *International Modal Analysis Conference*, pp. 194–200, 1999.

[41] B. Groult, *Extension d'une méthode de modification structurale pour la conception de dispositifs dissipatifs intégrant des matériaux viscoélastiques*. PhD thesis, École Centrale Paris 2008-14, 2008.

[42] E. Balmes, "Model reduction for systems with frequency dependent damping properties," *International Modal Analysis Conference*, pp. 223–229, 1997.

[43] T. Kant and S. K., "Free vibration of isotropic, orthotropic and multilayer plates based on higher order refined theories," *Journal of Sound and Vibration*, vol. 241, no. 2, pp. 319–327, 2001.

[44] E. Balmes and A. Bobillot, “Analysis and design tools for structures damped by viscoelastic materials,” *International Modal Analysis Conference*, February 2002.

[45] D. Mead and S. Markus, “The forced vibration of a three layer, damped sandwich beam with arbitrary boundary conditions,” *Journal of Sound and Vibration*, vol. 10, no. 2, pp. 163–175, 1969.

[46] J.-M. Berthelot, *Materiaux composites - Comportement mecanique et analyse des structures.* Masson, 1992.

[47] A. K. Pickett, G. Creech, and P. De Luca, “Simplified and advanced simulation methods for prediction of fabric draping,” *Revue Européenne des Éléments Finis*, vol. 14, pp. 677–691, Jan. 2005.

[48] M. Couet, J.-F. Deü, L. Rouleau, F. Thouverez, and M. . Gruin, “Topology optimization of constrained layer damping on fan blades,” in *ISMA, Leuven*, 2024.

[49] E. Balmes, “Parametric families of reduced finite element models. theory and applications,” *Mechanical Systems and Signal Processing*, vol. 10, no. 4, pp. 381–394, 1996.

[50] H. J.-P. Morand and R. Ohayon, *Fluid Structure Interaction.* J. Wiley & Sons 1995, Masson, 1992.

[51] G. Kergourlay, E. Balmes, and D. Clouteau, “Interface model reduction for efficient fem/bem coupling,” *International Seminar on Modal Analysis, Leuven*, pp. 1167–1174, September 2000.

[52] B. Van den Nieuwenhof, G. Lielens, and J. Coyette, “Modeling acoustic diffuse fields: Updated sampling procedure and spatial correlation function eliminating grazing incidences,” in *ISMA*, p. 14, 2010.