Symbolic computations in viscoelasticity and anisotropic elasticity

Fabio Cavallini

Géza Seriani

Viscoelastic constitutive laws arising from series and parallel connections of springs and dashpots can be easily formulated in symbolic form using the simple *Mathematica* code presented here. Viscoelastic quantities as the Q−factor and the relaxation function are also computed from symbolic (if there is no need to factor higher−order polynomials), or numeric data. The general symbolic form of the elasticity tensor in a given crystallographic class (e.g., monoclinic, cubic, and transversely isotropic) is obtained here from its symmetry properties. One−dimensional viscoelastic models, when applied to the eigenvalues of the stiffness operator of an elastic anisotropic medium, yield realistic 3−D viscoelastic models.

■ 1. Introduction

> Perfecta elasticitas esse nequit: nam partes subjaciuntur attritu, hoc attritu vis quaedam perit. (*P. V. Musschenbroek, 1762)*

As suggested by the quotation (Ianniello, 1993), viscoelastic constitutive laws are to be used, in geophysics and materials engineering, to model intrinsic attenuation of mechanical waves that propagate in dissipative media (Carcione, 2001). Although the first general theoretical formulation of viscoelasticity dates back to Ludwig Boltzmann (1874), this topic is still an active field of research (Fabrizio and Morro, 1992).

The main causes of anisotropy in the earth are: microstructural anisotropy of the minerals that form the rock, fine layering, and fracturing: indeed, homogenization theory allows researchers to model isotropic inhomogeneous media with equivalent, but simpler, homogeneous anisotropic media (Crampin, 1981). Nowadays, anisotropic data acquisition and processing is becoming an industrial standard in geophysical exploration and in reservoir engineering (Helbig, 1994).

One−dimensional viscoelastic models can be embedded into anisotropic elastic models to yield three−dimensional viscoelastic rheologies able to describe adequately the mechanical behaviour of realistic dissipative materials (Carcione and Cavallini, 1994).

In this paper we give some examples of how *Mathematica* may be profitably used to perform the symbolic computations needed to work out a suitable constitutive law, which is a necessary step preliminary to any numerical simulation of wave propagation.

■ 2. One−dimensional viscoelasticity

Classical viscoelastic rheological models (e.g., Kelvin−Voigt, Maxwell, Zener and Burgers) arise from the series/parallel connection of two basic building blocks: the spring and the dashpot. The resulting constitutive laws may be expressed in time domain through a convolution integral (Boltzmann superposition principle) or, equivalently, as an ordinary differential equation. By Fourier transforming, one obtains the frequency−domain representation of the constitutive model, which is often useful in the applications. Hence, various wave−related quantities may be computed, e.g., complex velocity, slowness, wavenumber, attenuation, and quality factor.

□ 1. Basic concepts

1. Stress and strain

In 1−D viscoelasticity, stress `s(x,t)` and strain `e(x,t)` are real functions of position `x` and time `t`. Strain `e(x,t)` is related to displacement `u(x,t)` through the kinematic equation

$$e(x, t) = \frac{\partial}{\partial x} u(x, t) \tag{1}$$

while stress `s(x,t)` with mass density ρ and displacement `u(x,t)` satisfy the momentum balance

$$\frac{\partial}{\partial x} s(x, t) = \rho \frac{\partial^2}{\partial t^2} u(x, t) \tag{2}$$

which is the counterpart in continuum mechanics of Newton's second law in particle dynamics. For convenience, the dependence on the space variable `x` will usually be understood.

2. Constitutive laws

In general, a linear causal and time−invariant stress−strain relationship is expressed by the *Boltzmann superposition principle*

$$s = C' * e \tag{3}$$

where `C` is the *relaxation function*, the prime `(')` denotes time differentiation, and the asterisk `(*)` indicates time convolution.

In the frequency domain, the constitutive law is suitably expressed through the *complex modulus*, which is the ratio between the Fourier transforms of the stress and the strain.

Of special importance are the so−called *differential models*, which have the form

$$\begin{aligned} a_0 s(t) + a_1 s'(t) + \ldots + a_m s^{(m)}(t) = \\ b_0 e(t) + b_1 e'(t) + \ldots + b_n e^{(n)}(t) \end{aligned} \tag{4}$$

and so they may be represented with the pair of their coefficient sequences

$$\{\{a_0, a_1, \ldots, a_m\}, \{b_0, b_1, \ldots, b_n\}\} \tag{5}$$

Conversely, we may pass from (5) to (4) by using

```
ConstitutiveLaw[model_, e_, s_, t_] :=
Sum[model⟦1⟧⟦k⟧ * Derivative[k - 1][s][t],
    {k, 1, Length[model⟦1⟧]}] ==
Sum[model⟦2⟧⟦k⟧ * Derivative[k - 1][e][t],
    {k, 1, Length[model⟦2⟧]}];
```

as will be shown in the Section on classical rheologies.

The complex modulus of differential models is simply given by

```
ComplexModulus[model_, ω_] :=
With[{diff = I * ω},
Sum[model⟦2⟧⟦k⟧ * diff^(k - 1), {k, 1, Length[model⟦2⟧]}] /
Sum[model⟦1⟧⟦k⟧ * diff^(k - 1), {k, 1, Length[model⟦1⟧]}]];
```

Noting that the relaxation function coincides with the stress if the strain is a unit step, we may compute the relaxation function with the following routine:

```
RelaxationFunction[model_, s_, t_] :=
If[Length[model⟦1⟧] == 1
,
  s[t] /. Flatten@Solve[ConstitutiveLaw[model, UnitStep, s, t]
, s[t]], s[t] /. Flatten@DSolve[
{ConstitutiveLaw[model, UnitStep, s, t],
       Table[Derivative[k][s][-1] == 0 // Unevaluated,
        {k, 0, Length[model⟦1⟧] - 1}]
} // Flatten, s[t], t]]
```

3. Energy−related quantities

By introducing the *particle velocity* $v := \partial u/\partial t$, the *Umov−Poynting variable* $p := s\ v$, the *kinetic energy* $E_K := 1/2\ \rho\ v^2$, and the *potential energy* $E_P := 1/2\ s\ e$, from the momentum balance (2) we get the energy balance

$$\frac{\partial p}{\partial x} = \frac{\partial}{\partial t}\ (E_K + E_P) + P_D \tag{6}$$

where $P_D := \partial E_P/\partial t - e\ \partial s/\partial t$ is the *dissipated power*.

For periodic motions of period τ, the amount of energy dissipation is usually expressed by the *quality factor Q*, defined as

$$Q := 4\,\pi\,\frac{\langle E_P \rangle}{E_D}$$

where $E_D := \tau\ \langle s\ \partial e / \partial t \rangle$ is *dissipated energy*, with angular brackets representing time average.

A straightforward computation shows that, for monochromatic waves, the quality factor Q is given by the ratio between the real and imaginary parts of the complex modulus :

```
QualityFactorQ[model_, ω_] :=
 ComplexModulus[model, ω] // (Re[#] / Im[#] &)
```

4. Wave propagation

The proper ansatz for investigating viscoelastic wave propagation is

$$u = U\,\mathrm{Exp}[i\ (k\,x - \omega\,t)\ ] \tag{7}$$

where $k = \kappa + i\ \alpha$, with κ and α the real wavenumber and attenuation, respectively (e.g., Hayes, 1980). Substituting (7) into (1)–(3) yields the propagation condition

$$V := \frac{\omega}{k} = \sqrt{\frac{M}{\rho}} \tag{8}$$

where V is the *complex velocity*, and M is the complex modulus. To describe wave propagation, the most important parameter is the *phase velocity* $V_P := \omega / \kappa$, which can be computed through

```
PhaseVelocity[model_, ρ_, ω_] :=
1 / Re[Sqrt[ρ / ComplexModulus[model, ω]]]
```

□ 2. Classical rheologies and beyond

The simplest and most used differential viscoelastic rheological models, which we call *classical*, are obtained by connecting "springs" and "dashpots" either in series or in parallel. A *parallel* connection of two models implies that they both undergo the same strain and that the total stress is the sum of the two stresses, so that the resulting model is given by

```
InParallel[model1_, model2_] := Module[
   {A1, B1, A2, B2, A1p, B1p, A2p, B2p, x}, {A1, B1} = model1;
   {A2, B2} = model2; A1p = A1.Table[x^k, {k, 0, Length[A1] - 1}];
   B1p = B1.Table[x^k, {k, 0, Length[B1] - 1}];
   A2p = A2.Table[x^k, {k, 0, Length[A2] - 1}];
   B2p = B2.Table[x^k, {k, 0, Length[B2] - 1}];
   {CoefficientList[A1p * A2p, x],
    CoefficientList[A1p * B2p + A2p * B1p, x]}];
```

while a *series* connection of two models implies that they both have the same stress and that the total strain is the sum of the two strains, so that the resulting model is given by

```
InSeries[model1_, model2_] := Module[
   {A1, B1, A2, B2, A1p, B1p, A2p, B2p, x}, {A1, B1} = model1;
   {A2, B2} = model2; A1p = A1.Table[x^k, {k, 0, Length[A1] - 1}];
   B1p = B1.Table[x^k, {k, 0, Length[B1] - 1}];
   A2p = A2.Table[x^k, {k, 0, Length[A2] - 1}];
   B2p = B2.Table[x^k, {k, 0, Length[B2] - 1}];
   {CoefficientList[A1p * B2p + A2p * B1p, x],
    CoefficientList[B1p * B2p, x]}];
```

1. The spring

The *spring* model obeys Hooke's law:

$$s(t) = \mu\, e(t)$$

and hence, according to (5), it may be represented as

```
Off[General::"spell1"]
Spring[μ_] := {{1}, {μ}}
On[General::"spell1"]
```

while its phase velocity is simply the sound speed:

```
PhaseVelocity[Spring[μ], ρ, ω] //
 FullSimplify[#, {μ > 0, ρ > 0, ω > 0}] &
```

$$\sqrt{\frac{\mu}{\rho}}$$

2. The dashpot

The *dashpot* model behaves as a Newtonian fluid (frictional stress proportional to displacement rate):

$$s(t) = \eta\, e'(t)$$

and hence, according to (5), it may be represented as

```
Dashpot[η_] := {{1}, {0, η}}
```

while its phase velocity is:

```
PhaseVelocity[Dashpot[η], ρ, ω] // ComplexExpand //
 FullSimplify[#, {η > 0, ρ > 0, ω > 0}] &
```

$$\sqrt{2}\ \sqrt{\frac{\eta\,\omega}{\rho}}$$

3. Kelvin−Voigt model

Kelvin−Voigt model consists in a parallel connection of a spring and a dashpot:

```
KelvinVoigt[μ_, η_] := InParallel[Spring[μ], Dashpot[η]]
```

which yields the stress−strain relationship

```
ConstitutiveLaw[KelvinVoigt[μ, η], e, s, t]
```

$$\mathtt{s[t]} == \mu\,\mathtt{e[t]} + \eta\,\mathtt{e'[t]}$$

and hence the relaxation function

```
RelaxationFunction[KelvinVoigt[μ, η], s, t]
```

$$\eta\,\mathtt{DiracDelta[t]} + \mu\,\mathtt{UnitStep[t]}$$

Moreover, the complex modulus is

```
ComplexModulus[KelvinVoigt[μ, η], ω]
```

$$\mu + i\,\eta\,\omega$$

and hence the quality factor Q follows:

```
QualityFactorQ[KelvinVoigt[μ, η], ω] // ComplexExpand
```

$$\frac{\mu}{\eta\,\omega}$$

Finally, the phase velocity is:

```
PhaseVelocity[KelvinVoigt[μ, η], ρ, ω] //
  ComplexExpand[#, TargetFunctions -> {Re, Im}] & //
 FullSimplify[#, {μ > 0, η > 0, ρ > 0, ω > 0}] &
```

$$\frac{(\mu^2 + \eta^2\,\omega^2)^{1/4}\ \mathtt{Sec}\left[\frac{1}{2}\,\mathtt{ArcTan}\left[\frac{\eta\,\omega}{\mu}\right]\right]}{\sqrt{\rho}}$$

4. Maxwell model

Maxwell model consists in a series connection of a spring and a dashpot:

```
Maxwell[μ_, η_] := InSeries[Spring[μ], Dashpot[η]]
```

which yields the stress−strain relationship

```
ConstitutiveLaw[Maxwell[μ, η], e, s, t]
```

$$\mu\,\mathtt{s[t]} + \eta\,\mathtt{s'[t]} == \eta\,\mu\,\mathtt{e'[t]}$$

and hence the relaxation function

```
RelaxationFunction[Maxwell[μ, η], s, t]
```

$$e^{-\frac{t\,\mu}{\eta}}\ \mu\,\mathtt{UnitStep[t]}$$

Moreover, the complex modulus is

```
ComplexModulus[Maxwell[μ, η], ω]
```

$$\frac{i\,\eta\,\mu\,\omega}{\mu + i\,\eta\,\omega}$$

and hence the quality factor Q follows:

```
QualityFactorQ[Maxwell[μ, η], ω] // ComplexExpand
```

$$\frac{\eta\,\omega}{\mu}$$

Finally, the phase velocity is:

```
PhaseVelocity[Maxwell[μ, η], ρ, ω] //
  ComplexExpand[#, TargetFunctions -> {Re, Im}] & //
 FullSimplify[#, {μ > 0, η > 0, ρ > 0, ω > 0}] &
```

$$\frac{\text{Sec}\left[\frac{1}{2}\,\text{ArcCot}\left[\frac{\eta\,\omega}{\mu}\right]\right]}{\left(\rho^2\,\left(\frac{1}{\mu^2} + \frac{1}{\eta^2\,\omega^2}\right)\right)^{1/4}}$$

5. Zener model

Zener model (also called *standard linear solid*) consists in a series connection of a spring and a Kelvin–Voigt model:

```
Zener[μS_, {μK_, ηK_}] :=
InSeries[Spring[μS], KelvinVoigt[μK, ηK]]
```

which yields the stress–strain relationship

```
ConstitutiveLaw[Zener[μS, {μK, ηK}], e, s, t]
```

$$(\mu K + \mu S)\,s[t] + \eta K\,s'[t] == \mu K\,\mu S\,e[t] + \eta K\,\mu S\,e'[t]$$

and hence the relaxation function

```
RelaxationFunction[Zener[μS, {μK, ηK}], s, t] // FullSimplify
```

$$\frac{\mu S\,\left(\mu K + e^{-\frac{t\,(\mu K+\mu S)}{\eta K}}\,\mu S\right)\,\text{UnitStep}[t]}{\mu K + \mu S}$$

Moreover, the complex modulus is

```
ComplexModulus[Zener[μS, {μK, ηK}], ω]
```

$$\frac{\mu K\,\mu S + i\,\eta K\,\mu S\,\omega}{\mu K + \mu S + i\,\eta K\,\omega}$$

and hence the quality factor Q follows:

```
QualityFactorQ[Zener[μS, {μK, ηK}], ω] // ComplexExpand //
 FullSimplify
```

$$\frac{\mu K\,(\mu K + \mu S)}{\eta K\,\mu S\,\omega} + \frac{\eta K\,\omega}{\mu S}$$

Finally, the phase velocity is:

```
PhaseVelocity[Zener[μS, {μK, ηK}], ρ, ω] //
   ComplexExpand[#, TargetFunctions -> {Re, Im}] & //
  FullSimplify[#, {μS > 0, μK > 0, ηK > 0, ρ > 0, ω > 0}] & // Timing
```

$$\left\{6.60262\,\text{Second},\ \sqrt{\frac{\mu S}{\rho}}\,\left(\frac{\mu K^2 + \eta K^2\,\omega^2}{(\mu K + \mu S)^2 + \eta K^2\,\omega^2}\right)^{1/4}\right.$$
$$\left.\text{Sec}\left[\frac{1}{2}\,\text{ArcTan}\left[\frac{\eta K\,\mu S\,\omega}{\mu K\,(\mu K + \mu S) + \eta K^2\,\omega^2}\right]\right]\right\}$$

6. Burgers model

Burgers model consists in a series connection of a Maxwell and a Kelvin–Voigt element:

```
Burgers[{μM_, ηM_}, {μK_, ηK_}] :=
 InSeries[Maxwell[μM, ηM], KelvinVoigt[μK, ηK]]
```

which yields the stress–strain relationship

```
ConstitutiveLaw[Burgers[{μM, ηM}, {μK, ηK}], e, s, t]
```

$$\mu K\, \mu M\, s[t] + (\eta M\, \mu K + \eta K\, \mu M + \eta M\, \mu M)\, s'[t] + \eta K\, \eta M\, s''[t] == \eta M\, \mu K\, \mu M\, e'[t] + \eta K\, \eta M\, \mu M\, e''[t]$$

and hence the relaxation function

```
RelaxationFunction[Burgers[{μM, ηM}, {μK, ηK}], s, t] //
 FullSimplify
```

$$\left(e^{-\frac{t\left(\eta K\, \mu M+\eta M\, (\mu K+\mu M)+\sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M+(\eta K\, \mu M+\eta M\, (\mu K+\mu M))^2}\right)}{2\, \eta K\, \eta M}}\, \mu M \left(-\eta M\, \mu K + \eta K\, \mu M + \eta M\, \mu M + e^{\frac{t\sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M+(\eta K\, \mu M+\eta M\, (\mu K+\mu M))^2}}{\eta K\, \eta M}}\, (\eta M\, \mu K - (\eta K + \eta M)\, \mu M) + \sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M + (\eta K\, \mu M + \eta M\, (\mu K + \mu M))^2} + e^{\frac{t\sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M+(\eta K\, \mu M+\eta M\, (\mu K+\mu M))^2}}{\eta K\, \eta M}}\, \sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M + (\eta K\, \mu M + \eta M\, (\mu K + \mu M))^2}\right) \text{UnitStep}[t]\right) \Big/ \left(2\sqrt{-4\, \eta K\, \eta M\, \mu K\, \mu M + (\eta K\, \mu M + \eta M\, (\mu K + \mu M))^2}\right)$$

Moreover, the complex modulus is

```
ComplexModulus[Burgers[{μM, ηM}, {μK, ηK}], ω]
```

$$\frac{i\, \eta M\, \mu K\, \mu M\, \omega - \eta K\, \eta M\, \mu M\, \omega^2}{\mu K\, \mu M + i\, (\eta M\, \mu K + \eta K\, \mu M + \eta M\, \mu M)\, \omega - \eta K\, \eta M\, \omega^2}$$

and hence the quality factor Q follows:

```
QualityFactorQ[Burgers[{μM, ηM}, {μK, ηK}], ω] //
  ComplexExpand // FullSimplify
```

$$\frac{\eta M\, \omega\, \left(\mu K\, (\mu K + \mu M) + \eta K^2\, \omega^2\right)}{\mu M\, \left(\mu K^2 + \eta K\, (\eta K + \eta M)\, \omega^2\right)}$$

Finally, the phase velocity is:

```
PhaseVelocity[Burgers[{μM, ηM}, {μK, ηK}], ρ, ω] //
   ComplexExpand[#, TargetFunctions -> {Re, Im}] & //
  FullSimplify[#, {μM > 0, μK > 0, ηM > 0, ηK > 0, ρ > 0, ω > 0}] & //
 Timing
```

$$\Big\{15.9539\ \text{Second},\ \sqrt{\frac{\eta M\, \mu M\, \omega}{\rho}}\, \left(\frac{\mu K^2 + \eta K^2\, \omega^2}{(\eta K\, \mu M + \eta M\, (\mu K + \mu M))^2\, \omega^2 + (\mu K\, \mu M - \eta K\, \eta M\, \omega^2)^2}\right)^{1/4} \text{Sec}\Big[\frac{1}{2}\, \text{ArcCot}\Big[\frac{\eta M\, \omega\, \left(\mu K\, (\mu K + \mu M) + \eta K^2\, \omega^2\right)}{\mu M\, \left(\mu K^2 + \eta K\, (\eta K + \eta M)\, \omega^2\right)}\Big]\Big]\Big\}$$

7. ... and beyond

Of course, more complex differential viscoelastic rheologies can be devised and analyzed along the lines seen above.

■ 3. Anisotropic elasticity

The classification of anisotropic materials relies upon the invariance properties of the strain energy with respect to symmetry groups of rotations and reflections (Landau, 1990). The computation of the corresponding elasticity tensor requires tricky and lengthy procedures. With *Mathematica*'s symbolic tools, this goal may be achieved just using the very definitions, so avoiding to handle the cumbersome Bond's transformation (Auld, 1990).

□ 1. Anisotropic Hooke's law

The anisotropic Hooke's law states that stress S and strain E are linearly related through the stiffness operator C :

$$S = C\, E \tag{9}$$

or, in component notation,

$$s_{ij} = c_{ijkl}\ e_{kl} \tag{10}$$

so that the stiffness operator C is also called the stiffness tensor.

The stiffness tensor has $3^4 = 81$ components, but at most (6×6−6)/2+6=21 of them are independent, because stress, strain and stiffness operator are all symmetric. This yields

```
stiffnessTensorC =
Table[ToExpression@StringJoin[ToString /@
      Flatten[{c, Sort[{Sort[{i, j}], Sort[{k, l}]}]}]]
, {i, 3}, {j, 3}, {k, 3}, {l, 3}];
stiffnessTensorC // MatrixForm
% // Flatten // Union
% // Length
```

$$\begin{pmatrix} \begin{pmatrix} c1111 & c1112 & c1113 \\ c1112 & c1122 & c1123 \\ c1113 & c1123 & c1133 \end{pmatrix} & \begin{pmatrix} c1112 & c1212 & c1213 \\ c1212 & c1222 & c1223 \\ c1213 & c1223 & c1233 \end{pmatrix} & \begin{pmatrix} c1113 & c1213 & c1313 \\ c1213 & c1322 & c1323 \\ c1313 & c1323 & c1333 \end{pmatrix} \\ \begin{pmatrix} c1112 & c1212 & c1213 \\ c1212 & c1222 & c1223 \\ c1213 & c1223 & c1233 \end{pmatrix} & \begin{pmatrix} c1122 & c1222 & c1322 \\ c1222 & c2222 & c2223 \\ c1322 & c2223 & c2233 \end{pmatrix} & \begin{pmatrix} c1123 & c1223 & c1323 \\ c1223 & c2223 & c2323 \\ c1323 & c2323 & c2333 \end{pmatrix} \\ \begin{pmatrix} c1113 & c1213 & c1313 \\ c1213 & c1322 & c1323 \\ c1313 & c1323 & c1333 \end{pmatrix} & \begin{pmatrix} c1123 & c1223 & c1323 \\ c1223 & c2223 & c2323 \\ c1323 & c2323 & c2333 \end{pmatrix} & \begin{pmatrix} c1133 & c1233 & c1333 \\ c1233 & c2233 & c2333 \\ c1333 & c2333 & c3333 \end{pmatrix} \end{pmatrix}$$

```
{c1111, c1112, c1113, c1122, c1123, c1133, c1212,
 c1213, c1222, c1223, c1233, c1313, c1322, c1323,
 c1333, c2222, c2223, c2233, c2323, c2333, c3333}

21
```

For convenience, we shall use the abbreviated index notation

```
singleIndex[i_, j_] := {{1, 6, 5}, {6, 2, 4}, {5, 4, 3}}[[i, j]];
```

so that the general stiffness tensor is

```
stiffnessTensorC =
Table[ToExpression@StringJoin[ToString /@ Flatten[
       {c, Sort[{singleIndex[i, j], singleIndex[k, l]}]}]],
   {i, 3}, {j, 3}, {k, 3}, {l, 3}];
stiffnessTensorC // MatrixForm
entriesOfC = stiffnessTensorC // Flatten // Union;
```

$$\begin{pmatrix} \begin{pmatrix} c11 & c16 & c15 \\ c16 & c12 & c14 \\ c15 & c14 & c13 \end{pmatrix} & \begin{pmatrix} c16 & c66 & c56 \\ c66 & c26 & c46 \\ c56 & c46 & c36 \end{pmatrix} & \begin{pmatrix} c15 & c56 & c55 \\ c56 & c25 & c45 \\ c55 & c45 & c35 \end{pmatrix} \\ \begin{pmatrix} c16 & c66 & c56 \\ c66 & c26 & c46 \\ c56 & c46 & c36 \end{pmatrix} & \begin{pmatrix} c12 & c26 & c25 \\ c26 & c22 & c24 \\ c25 & c24 & c23 \end{pmatrix} & \begin{pmatrix} c14 & c46 & c45 \\ c46 & c24 & c44 \\ c45 & c44 & c34 \end{pmatrix} \\ \begin{pmatrix} c15 & c56 & c55 \\ c56 & c25 & c45 \\ c55 & c45 & c35 \end{pmatrix} & \begin{pmatrix} c14 & c46 & c45 \\ c46 & c24 & c44 \\ c45 & c44 & c34 \end{pmatrix} & \begin{pmatrix} c13 & c36 & c35 \\ c36 & c23 & c34 \\ c35 & c34 & c33 \end{pmatrix} \end{pmatrix}$$

Moreover, Voigt notation allows for the compact representation

```
doubleIndex[k_] :=
  {{1, 1}, {2, 2}, {3, 3}, {2, 3}, {1, 3}, {1, 2}}⟦k⟧;
FromTo["c3333", "cVoigt"][c3333_] :=
  Table[c3333⟦doubleIndex[I]⟦1⟧, doubleIndex[I]⟦2⟧,
    doubleIndex[J]⟦1⟧, doubleIndex[J]⟦2⟧⟧,
   {I, 1, 6}, {J, 1, 6}];
FromTo["c3333", "cVoigt"][stiffnessTensorC] // MatrixForm
```

$$\begin{pmatrix} c11 & c12 & c13 & c14 & c15 & c16 \\ c12 & c22 & c23 & c24 & c25 & c26 \\ c13 & c23 & c33 & c34 & c35 & c36 \\ c14 & c24 & c34 & c44 & c45 & c46 \\ c15 & c25 & c35 & c45 & c55 & c56 \\ c16 & c26 & c36 & c46 & c56 & c66 \end{pmatrix}$$

If we subject the reference system to a linear transformation F, then the stiffness tensor C becomes

$$(C')_{ijkl} = C_{mnpq}\, F_{im}\, F_{jn}\, F_{kp}\, F_{lq}$$

which can be coded as

```
NewStiffnessTensorCprime[F_] :=
NewStiffnessTensorCprime[F] =
Table[Sum[stiffnessTensorC⟦m, n, p, q⟧ *
      F⟦i, m⟧ * F⟦j, n⟧ * F⟦k, p⟧ * F⟦l, q⟧
         , {m, 3}, {n, 3}, {p, 3}, {q, 3}
         ]
       , {i, 3}, {j, 3}, {k, 3}, {l, 3}
      ];
```

□ 2. Reflections and rotations

Reflections and rotations are easily coded using the tensor (or dyadic) product of two vectors.

1. Tensor product

The tensor (or dyadic) product of two vectors, u and v, is the linear operator

$$u \otimes v : x \mapsto (v \cdot x)\, u$$

whose matrix entries are

$$(u \otimes v)_{ij} = u_i\, v_j$$

In *Mathematica* it can be coded as

```
TensorProduct[vec1_?VectorQ, vec2_?VectorQ] :=
 Outer[Times, vec1, vec2]
```

2. Reflections

A mirror reflection with respect to a plane is a linear operator whose matrix is given by

```
ReflectionMatrix[versor_] :=
 IdentityMatrix[3] - 2 * TensorProduct[versor, versor]
```

where `versor` is the unit vector normal to the symmetry plane.

3. Rotations

A rotation is a linear operator whose matrix is given by

```
RotationMatrix[versor_, angle_] :=
With[{vTv = TensorProduct[versor, versor]}, (vTv
+ Sin[angle] * Skew[versor]
+ Cos[angle] * (IdentityMatrix[3] - vTv))]
```

where

```
Skew[{v1_, v2_, v3_}] := {{0, -v3, v2}, {v3, 0, -v1}, {-v2, v1, 0}}
```

For example, the rotation of an angle ϕ about the vertical axis is described by

```
RotationMatrix[{0, 0, 1}, ϕ] // MatrixForm
```

$$\begin{pmatrix} \text{Cos}[\phi] & -\text{Sin}[\phi] & 0 \\ \text{Sin}[\phi] & \text{Cos}[\phi] & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

□ 3. Classes of anisotropic stiffness operators

In this section we give a few examples of how the previous arguments may be applied to obtain the general form of the stiffness tensor in a given crystallographic class.

1. Monoclinic

Monoclinic stiffness tensors are invariant with respect to mirror reflection about a given plane. For example, the coordinate plane y−z is normal to versor {1,0,0} and identifies the reflection matrix

```
ReflectionMatrix[{1, 0, 0}] // MatrixForm
```

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The corresponding monoclinic stiffness tensor is obtained by solving the equations generated by the invariance condition, and we note that at most 13 parameters are needed to describe a monoclinic medium:

```
Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
     ReflectionMatrix[{1, 0, 0}]]] // Thread // Union

Off[Solve::svars]
Solve[%%, entriesOfC] // Flatten
On[Solve::svars]

stiffnessTensorC /. %%;
% // FromTo["c3333", "cVoigt"] // MatrixForm
% // Flatten // Union // Rest
% // Length
```

$$\{\text{True}, c15 == -c15, c16 == -c16, c25 == -c25, c26 == -c26,\ c35 == -c35, c36 == -c36, c45 == -c45, c46 == -c46\}$$

$$\{c15 \to 0, c16 \to 0, c25 \to 0,\ c26 \to 0, c35 \to 0, c36 \to 0, c45 \to 0, c46 \to 0\}$$

$$\begin{pmatrix} c11 & c12 & c13 & c14 & 0 & 0 \\ c12 & c22 & c23 & c24 & 0 & 0 \\ c13 & c23 & c33 & c34 & 0 & 0 \\ c14 & c24 & c34 & c44 & 0 & 0 \\ 0 & 0 & 0 & 0 & c55 & c56 \\ 0 & 0 & 0 & 0 & c56 & c66 \end{pmatrix}$$

$$\{c11, c12, c13, c14, c22, c23, c24, c33, c34, c44, c55, c56, c66\}$$

```
13
```

2. Transversely isotropic

Transversely isotropic stiffness tensors are invariant with respect to all rotations about a given axis. For example, the transversely isotropic stiffness tensors with vertical axis of symmetry are obtained from the invariance condition:

```
Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
      RotationMatrix[{0, 0, 1}, ϕ]]] // Thread // Union;

Off[Solve::svars]
Solve[%%, entriesOfC] // Flatten
On[Solve::svars]

stiffnessTensorC /. %%;
% // FromTo["c3333", "cVoigt"] // MatrixForm

% // Flatten // Union // Rest
Count[%, _?AtomQ]
```

$$\{c34 \to 0, c35 \to 0, c13 \to c23, c44 \to c55, c36 \to 0, c45 \to 0,\ c11 \to c12 + 2\,c66, c14 \to 0, c15 \to 0, c22 \to c12 + 2\,c66,\ c24 \to 0, c25 \to 0, c46 \to 0, c56 \to 0, c16 \to 0, c26 \to 0\}$$

$$\begin{pmatrix} c12 + 2\,c66 & c12 & c23 & 0 & 0 & 0 \\ c12 & c12 + 2\,c66 & c23 & 0 & 0 & 0 \\ c23 & c23 & c33 & 0 & 0 & 0 \\ 0 & 0 & 0 & c55 & 0 & 0 \\ 0 & 0 & 0 & 0 & c55 & 0 \\ 0 & 0 & 0 & 0 & 0 & c66 \end{pmatrix}$$

$$\{c12, c23, c33, c55, c66, c12 + 2\,c66\}$$

```
5
```

whence we note that at most 5 parameters are needed to describe a transversely isotropic medium.

This procedure may be applied even if the symmetry axis is oblique, although some numerical trick is here advisable in order to speed up computations:

```
Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
       RotationMatrix[{1, 1, 1} / Sqrt[3], Pi / 11
]]] // Thread // Union // N;

Off[Solve::svars]
Solve[%%, entriesOfC] // Flatten // Chop
On[Solve::svars]

stiffnessTensorC /. %%;
% // FromTo["c3333", "cVoigt"]

% // Flatten // Union;
Cases[%, _?AtomQ]
% // Length
```

```
{c11 → 1. c33, c12 → 1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66,
 c13 → 1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66,
 c14 → 1. c36, c15 → 1. c35, c16 → 1. c35, c22 → 1. c33,
 c23 → 1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66,
 c24 → 1. c35, c25 → 1. c36, c26 → 1. c35, c34 → 1. c35,
 c44 → 1. c66, c45 → 1. c56, c46 → 1. c56, c55 → 1. c66}
```

```
{{1. c33, 1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66,
  1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66, 1. c36, 1. c35, 1. c35},
 {1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66, 1. c33,
  1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66, 1. c35, 1. c36, 1. c35},
 {1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66,
  1. c33 - 1. c35 + 1. c36 + 2. c56 - 2. c66, c33, 1. c35, c35, c36},
 {1. c36, 1. c35, 1. c35, 1. c66, 1. c56, 1. c56},
 {1. c35, 1. c36, c35, 1. c56, 1. c66, c56},
 {1. c35, 1. c35, c36, 1. c56, c56, c66}}
```

```
{c33, c35, c36, c56, c66}
```

```
5
```

In this case, all the components of the transversely isotropic stiffness tensor are nonzero.

3. Isotropic

Isotropic stiffness tensors are invariant with respect to all rotations about any axis. Actually, it is enough to require transverse isotropy with respect to two coordinate axes, and the invariance condition now yields that two parameters are sufficient to describe an isotropic medium:

```
{Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
        RotationMatrix[{1, 0, 0}, ϕ]]] // Thread // Union
, Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
        RotationMatrix[{0, 1, 0}, ϕ]]] // Thread // Union
} // Flatten;

Off[Solve::svars]
Solve[%%, entriesOfC] // Flatten
On[Solve::svars]

stiffnessTensorC /. %%;
% // FromTo["c3333", "cVoigt"] // MatrixForm

% // Flatten // Union // Rest
Count[%, _?AtomQ]
```

$$\{c11 \to c23 + 2\,c55,\ c22 \to c23 + 2\,c55,\ c12 \to c23,\ c66 \to c55,\ c16 \to 0,\ c26 \to 0,\ c14 \to 0,\ c25 \to 0,\ c46 \to 0,\ c56 \to 0,\ c33 \to c23 + 2\,c55,\ c36 \to 0,\ c13 \to c23,\ c15 \to 0,\ c24 \to 0,\ c44 \to c55,\ c45 \to 0,\ c34 \to 0,\ c35 \to 0\}$$

$$\begin{pmatrix} c23 + 2\,c55 & c23 & c23 & 0 & 0 & 0 \\ c23 & c23 + 2\,c55 & c23 & 0 & 0 & 0 \\ c23 & c23 & c23 + 2\,c55 & 0 & 0 & 0 \\ 0 & 0 & 0 & c55 & 0 & 0 \\ 0 & 0 & 0 & 0 & c55 & 0 \\ 0 & 0 & 0 & 0 & 0 & c55 \end{pmatrix}$$

$$\{c23,\ c55,\ c23 + 2\,c55\}$$

```
2
```

4. Cubic

Cubic stiffness tensors are invariant with respect to rotations of $\pi/2$ about each princpal coordinate axis. Accordingly, the invariance condition now yields that three parameters are sufficient to describe a cubic medium:

```
{Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
        RotationMatrix[{1, 0, 0}, Pi / 2]]] // Thread // Union
, Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
        RotationMatrix[{0, 1, 0}, Pi / 2]]] // Thread // Union
, Flatten[stiffnessTensorC] ==
Flatten[NewStiffnessTensorCprime[
        RotationMatrix[{0, 0, 1}, Pi / 2]]] // Thread // Union
} // Flatten;

Off[Solve::svars]
Solve[%%, entriesOfC] // Flatten
On[Solve::svars]

stiffnessTensorC /. %%;
cubicC = % // FromTo["c3333", "cVoigt"];
% // MatrixForm

% // Flatten // Union // Rest
Count[%, _?AtomQ]
```

$$\{c11 \to c33,\ c12 \to c23,\ c13 \to c23,\ c14 \to 0,\ c15 \to 0,\ c16 \to 0,\ c22 \to c33,\ c24 \to 0,\ c25 \to 0,\ c26 \to 0,\ c34 \to 0,\ c35 \to 0,\ c36 \to 0,\ c44 \to c66,\ c45 \to 0,\ c46 \to 0,\ c55 \to c66,\ c56 \to 0\}$$

$$\begin{pmatrix} c33 & c23 & c23 & 0 & 0 & 0 \\ c23 & c33 & c23 & 0 & 0 & 0 \\ c23 & c23 & c33 & 0 & 0 & 0 \\ 0 & 0 & 0 & c66 & 0 & 0 \\ 0 & 0 & 0 & 0 & c66 & 0 \\ 0 & 0 & 0 & 0 & 0 & c66 \end{pmatrix}$$

$$\{c23, c33, c66\}$$

```
3
```

■ 4. Anisotropic viscoelasticity

Using the correspondence principle, one can modify the eigenvalues of an elasticity operator while retaining its original eigenvectors: in this way one gets a fully 3−D viscoelastic anisotropic constitutive model. This may be accomplished in three steps:

1. Write the stiffness tensor in canonical form, C_{can} .
2. Compute the eigenstiffnesses, i.e., the eigenvalues of C_{can} .
3. In the principal−axis decomposition of C_{can} , substitute the eigenstiffnesses with complex viscoelastic moduli.

Step 1 may be obtained with the function

```
FromTo["cVoigt", "cCanonical"][cVoigt_] :=
 Module[{a, b, F}, a = 1 / Sqrt[2]; b = 1 / 2; F = {{1, 1, 1, a, a, a},
    {1, 1, 1, a, a, a}, {1, 1, 1, a, a, a}, {a, a, a, b, b, b},
    {a, a, a, b, b, b}, {a, a, a, b, b, b}}; cVoigt / F]
```

so that the eigenstiffnesses of, e.g., a cubic stiffness operator are given with their multiplicities (Step 2) by

```
Eigenvalues[FromTo["cVoigt", "cCanonical"][cubicC]]
```

$$\{-c23 + c33,\ -c23 + c33,\ 2\,c23 + c33,\ 2\,c66,\ 2\,c66,\ 2\,c66\}$$

in perfect agreement with Mehrabadi and Cowin (1990).

For further details and applications, see Carcione and Cavallini (1994).

■ 5. Conclusion

The numerical modelling of wave propagation in anisotropic viscoelastic media (Carcione, 2001; Carcione et al., 1992) requires preliminary symbolic computations that, although conceptually trivial, are tedious and error−prone if performed by hand. Using the built−in *Mathematica* facilities for symbolic computation (including algebraic manipulations, ODE solvers and Fourier transforms), one can produce software that greatly simplifies this task and yields deeper geometric and physical insight. Our treatment of viscoelastic rheologies may be adapted to simulate any network (e.g., in electrical engineering or systems theory) of linear constituents connected in series and/or in parallel.

■ References

Auld, B. A., 1990: Acoustic fields and waves in solids, Vol. 1, 2.nd edition, Robert Krieger Publishing Co.

Boltzmann, L., 1874: Zur Theorie der elastichen Nachwirkung, Sitz. Kgl. Akad. Wiss. Wien (Math−Naturwiss Klasse) 70, 275−306.

Carcione, J., 2001: Wave fields in real media (Handbook of geophysical exploration: seismic exploration), Pergamon.

Carcione, J. M., and Cavallini, F., 1994: A rheological model for anelastic media with applications to seismic wave propagation. Geophys. J. Internat. 119(1)338−348.

Carcione, J. M., Seriani, G., and Priolo, E., 1992: Wave simulation in three−dimensional anisotropic−viscoelastic media, 62nd Ann. Internat. Mtg: Soc. of Expl. Geophys., 1251−1254.

Crampin, S., 1981: A review of wave motion in anisotropic and cracked elastic media, Wave Motion, 3, 343−391.

Fabrizio, M., and Morro, A., 1992: Mathematical Problems in Linear Viscoelasticity, SIAM.

Hayes. M., 1980: Energy flux for trains of inhomogeneous plane waves, Proc. R. Soc. London A 370, 417−429.

Helbig, K., 1994: Foundations of anisotropy for exploration seimics (Handbook of geophysical exploration: seismic exploration), Pergamon.

Ianniello, M. G., 1993: Elastic *Nachwirkung*, Brownian motion and the tide against determinism: 1835−1920, HSPS 24(1), 41−100.

Landau, L., and Lifchitz, E., 1990: Théorie de l'elasticité, Mir.

Mehrabadi, M. M. and Cowin, S. C., 1990: Eigentensors of linear anisotropic elastic materials, Q. Jl. Mech. & Appl. Math. 43(1), 15−41.

About the Authors

Fabio Cavallini is a tenure senior researcher at OGS, and a member of the Scientific Board of OGS. He has authored articles, and professional software in the area of seismic and electromagnetic wave propagation, oceanography, hydrology, and ecology. He holds an Italian Laurea in physics (cum laude) from the University of Trieste, Italy.

Géza Seriani is vice−Director of the Geophysics department at OGS and responsible of the geophysical modelling group in the same department . He has authored articles, and professional software in the area of seismic wave propagation, in numerical wave modeling for complex

geological structures and large scale parallell computing. He holds an Italian Laurea in physics from the University of Trieste, Italy.

Fabio Cavallini
Department of Geophysics of the Lithosphere
Istituto Nazionale di Oceanografia e di Geofisica Sperimentale − OGS
Borgo Grotta Gigante 42/C
I−34010 Sgonico, Trieste, Italy
fcavallini@ogs.trieste.it

Géza Seriani
Department of Geophysics of the Lithosphere
Istituto Nazionale di Oceanografia e di Geofisica Sperimentale − OGS
Borgo Grotta Gigante 42/C
I−34010 Sgonico, Trieste, Italy
gseriani@ogs.trieste.it