Echoes

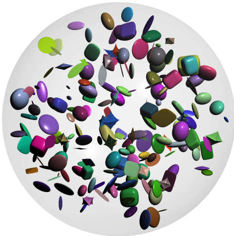
Extended Calculator of HOmogEnization Schemes

Jean-François Barthélémy

2025-01-10

Table of contents

Welcome



The library `echoes` allows to implement various mean-field homogenization schemes of random media involving different types of heterogeneities in the framework of elasticity, conductivity, viscoelasticity as well as nonlinear homogenization.

This book gathers tutorials presenting the main features of the library:

- elements of tensor calculus,
- Hill and Eshelby tensors and their derivatives with respect to reference medium moduli,
- concentration problems,
- RVEs and schemes in linear homogenization,
- extension to nonlinear homogenization,
- extension to linear time-dependent behaviors.

Download

The core of `echoes` has been developed in C++ and wrapped by a Python interface. Hence its use requires first the installation of a Python environment including `pip` executable (for instance Anaconda).

Wheel packages can be downloaded for various versions of Python under Windows or Linux by choosing the appropriate file for your configuration under the link

https://doi.org/10.5281/zenodo.10559657

Once in possession of the relevant `.whl` file, the package can be installed in a console (Anaconda console or any console allowing to run `pip`) by

```
pip install -U echoes-XYZ.whl
# replacing echoes-XYZ.whl by the correct path to the whl file
```

Citation

If you use `echoes`, please cite it as (Barthélémy, 2022) or in `bibtex` style

```
@software{echoes,
  title = {Echoes: {{Extended Calculator}} of {{HOmogEnization Schemes}}},
  shorttitle = {Echoes},
  author = {Barthélémy, Jean-François},
  date = {2022-11-22},
  doi = {10.5281/ZENODO.10559657},
  url = {https://zenodo.org/record/10559657},
  organization = {Zenodo},
  version = {v1.0.0},
}
```

All this work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License .

About the author

Jean-François Barthélémy is a researcher at Cerema in the research team UMR MCD.

- GitHub
- ResearchGate
- Google scholar
- □ HAL
- □ Web Of Science
- ORCID
- □ Email

Powered by Quarto.

Introduction

This book is aimed at researchers, engineers and students, knowing the fundamentals of mean-field theory to help them learn how to use the `echoes` library with some brief theoretical recalls when relevant. For a more exhaustive presentation of the theory of random medium homogenization, see (Bornert et al., 2001), (Milton, 2002), (Torquato, 2002) or (Kachanov and Sevostianov, 2018) among others.

The objectives of the library can be summarized as follows:

- simple and quick implementation of Eshelby problems and homogenization schemes,
- multi-physics and multi-scale homogenization,
- effects of microstructure changes by chemical, physical or mechanical process.

Features

- Eshelby problem solved at 2nd (conductivity) et 4th orders (elasticity)
- Isotropy and anisotropy
- Several types of inclusions including generic (user-defined) inclusion
- Large variety of schemes
- Derivatives of the macroscopic elasticity with respect to lower scale moduli
- Aging linear viscoelasticity
- Complex moduli

In this manual, some snippets of Python codes are presented. The `echoes` library can be imported as

```
from echoes import *
```

or, to avoid any ambiguity between libraries, as

```
import echoes as ec
```

A usual start of any tutorial could be the following

```
import numpy as np
from echoes import *
import matplotlib.pyplot as plt # if plots are needed

np.set_printoptions(precision=8, suppress=True)
# to display only 8 significant digits of array components
```

Whenever they are omitted, it is implicitly considered that these lines have previously been added.

Part I

Elements of tensor calculus

1 Kelvin-Mandel notation

Objectives

Before introducing the specific objects of `echoes` devoted to tensor calculations in isotropic or anisotropic contexts, this tutorial aims at providing the syntax allowing to represent second-order or fourth-order tensors under the form of matrices in the Kelvin-Mandel notation as detailed in Section A.2.

Download

- Python script
- Jupyter notebook

Imports

```
import numpy as np
from echoes import *
import math, random

np.set_printoptions(precision=8, suppress=True)
# to display only 8 significant digits of array components
```

A symmetric 3×3 second-order matrix can be transformed in a vector of $\mathbb{R}^6$ by the function `KM` consistently with A.14. The inverse is done by `invKM`.

```
α = np.random.rand(3, 3) ; ε = (α+α.T)/2
print("ε =\n",ε)
print("KM(ε) =\n",KM(ε))
assert np.allclose(invKM(KM(ε)), ε), "error"
```

```
ε =
 [[0.58771074 0.45094825 0.85199367]
 [0.45094825 0.58345257 0.77762682]
 [0.85199367 0.77762682 0.86478055]]
KM(ε) =
 [0.58771074 0.58345257 0.86478055 1.0997304  1.204901   0.63773713]
```

Given a $3 \times 3 \times 3 \times 3$ array `c` (of type `numpy.ndarray`) satisfying major and minor symmetries (see Section A.2), the corresponding 6×6 matrix `C` obtained by Kelvin-Mandel transform is calculated by `C = KM(c)`. Conversely, if `C` is a positive definite matrix, `c` is calculated by `c = invKM(C)`.

```python
A = np.random.rand(6,6)
C = A.T.dot(A) + np.eye(6) # generation of an arbitrary positive definite matrix
c = invKM(C)
print("C =\n",C)
print("c =\n",c)
assert np.allclose(KM(c), C), "error: KM(c) should be equal to C"
```

```
C =
 [[3.85644026 1.41095072 2.95853961 2.52847436 1.28150938 1.64440507]
 [1.41095072 2.48590364 1.72764219 0.99712666 0.73841788 0.64880257]
 [2.95853961 1.72764219 5.0025816  3.20820231 2.14993105 2.28459766]
 [2.52847436 0.99712666 3.20820231 3.90738234 1.79408665 2.1472566 ]
 [1.28150938 0.73841788 2.14993105 1.79408665 2.64546152 1.55864921]
 [1.64440507 0.64880257 2.28459766 2.1472566  1.55864921 2.84648975]]
c =
 [[[[3.85644026 1.16276998 0.90616397]
   [1.16276998 1.41095072 1.78790137]
   [0.90616397 1.78790137 2.95853961]]

  [[1.16276998 1.42324487 0.77932461]
   [1.42324487 0.4587727  1.0736283 ]
   [0.77932461 1.0736283  1.6154545 ]]

  [[0.90616397 0.77932461 1.32273076]
   [0.77932461 0.52214029 0.89704333]
   [1.32273076 0.89704333 1.52023083]]]


 [[[1.16276998 1.42324487 0.77932461]
   [1.42324487 0.4587727  1.0736283 ]
   [0.77932461 1.0736283  1.6154545 ]]

  [[1.41095072 0.4587727  0.52214029]
   [0.4587727  2.48590364 0.70507502]
   [0.52214029 0.70507502 1.72764219]]

  [[1.78790137 1.0736283  0.89704333]
   [1.0736283  0.70507502 1.95369117]
   [0.89704333 1.95369117 2.26854161]]]


 [[[0.90616397 0.77932461 1.32273076]
   [0.77932461 0.52214029 0.89704333]
```

```
 [1.32273076 0.89704333 1.52023083]]

[[1.78790137 1.0736283  0.89704333]
 [1.0736283  0.70507502 1.95369117]
 [0.89704333 1.95369117 2.26854161]]

[[2.95853961 1.6154545  1.52023083]
 [1.6154545  1.72764219 2.26854161]
 [1.52023083 2.26854161 5.0025816 ]]]]
```

2 Rotation matrices

Objectives

This tutorial presents the construction of rotation matrices in $\mathbb{R}^3$ in the convention proposed in Section A.3 for Euler angles.

Download

- Python script
- Jupyter notebook

Imports

```
import numpy as np
from echoes import *
import math, random

np.set_printoptions(precision=8, suppress=True)
# to display only 8 significant digits of array components
```

Given the Euler angles `θ, ϕ, ψ` defined in Section A.3 and more particularly in Fig. A.1, the rotation matrix (A.19) recalled here

$$\mathsf{R} = \begin{pmatrix} c_\theta c_\psi c_\phi - s_\psi s_\phi & -c_\theta c_\phi s_\psi - c_\psi s_\phi & c_\phi s_\theta \\ c_\theta c_\psi s_\phi + c_\phi s_\psi & -c_\theta s_\psi s_\phi + c_\psi c_\phi & s_\theta s_\phi \\ -c_\psi s_\theta & s_\theta s_\psi & c_\theta \end{pmatrix}$$

can be built with `rot3(θ, ϕ, ψ)`

```
π = math.pi
θ, ϕ, ψ = π/3, π/4, π/5
R = rot3(θ, ϕ, ψ)
print("R =\n",R)
```

```
R =
 [[-0.12959624 -0.77987487  0.61237244]
 [ 0.70165764  0.36424793  0.61237244]
 [-0.70062927  0.50903696  0.5       ]]
```

The vectors of the spherical basis correspond to the column of the rotation matrix for $\psi = 0$. They can individually be obtained by the function `es(i, θ, ϕ)`

> **Warning**
>
> Python numbering starts at 0 so `i` takes the values 0, 1 and 2.
> Besides vectors of the spherical basis are ordered as $\underline{e}_\theta, \underline{e}_\phi, \underline{e}_r$.

```
θ, ϕ = π/8, π/5
R = rot3(θ, ϕ)
print("R =\n",R)
for i in range(3): print(f"es({i}, θ, ϕ) = ", es(i, θ, ϕ), f" → e̲ {['θ','ϕ','r'][i]}")
```

```
R =
 [[ 0.74743424 -0.58778525  0.3095974 ]
 [ 0.54304276  0.80901699  0.22493568]
 [-0.38268343  0.          0.92387953]]
es(0, θ, ϕ) =  [ 0.74743424  0.54304276 -0.38268343]  → e̲ θ
es(1, θ, ϕ) =  [-0.58778525  0.80901699  0.         ]  → e̲ ϕ
es(2, θ, ϕ) =  [0.3095974  0.22493568 0.92387953]  → e̲ r
```

As shown in Section A.3 the rotation matrix applying on fourth-order tensors in Kelvin-Mandel notation can be deduced from the 3×3 rotation matrix from A.24

$$\begin{pmatrix} R_{11}^2 & R_{12}^2 & R_{13}^2 & \sqrt{2}R_{12}R_{13} & \sqrt{2}R_{11}R_{13} & \sqrt{2}R_{11}R_{12} \\ R_{21}^2 & R_{22}^2 & R_{23}^2 & \sqrt{2}R_{22}R_{23} & \sqrt{2}R_{21}R_{23} & \sqrt{2}R_{21}R_{22} \\ R_{31}^2 & R_{32}^2 & R_{33}^2 & \sqrt{2}R_{32}R_{33} & \sqrt{2}R_{31}R_{33} & \sqrt{2}R_{31}R_{32} \\ \sqrt{2}R_{21}R_{31} & \sqrt{2}R_{22}R_{32} & \sqrt{2}R_{23}R_{33} & R_{22}R_{33}+R_{23}R_{32} & R_{21}R_{33}+R_{23}R_{31} & R_{21}R_{32}+R_{22}R_{31} \\ \sqrt{2}R_{11}R_{31} & \sqrt{2}R_{12}R_{32} & \sqrt{2}R_{13}R_{33} & R_{12}R_{33}+R_{13}R_{32} & R_{11}R_{33}+R_{13}R_{31} & R_{11}R_{32}+R_{12}R_{31} \\ \sqrt{2}R_{11}R_{21} & \sqrt{2}R_{12}R_{22} & \sqrt{2}R_{13}R_{23} & R_{12}R_{23}+R_{13}R_{22} & R_{11}R_{23}+R_{13}R_{21} & R_{11}R_{22}+R_{12}R_{21} \end{pmatrix}$$

It can be directly obtained by `rot6(θ, ϕ, ψ)`.

```
θ, ϕ, ψ = π/3, π/4, π/5
R = rot3(θ, ϕ, ψ)
sboxtimes=lambda a,b:0.5*(np.einsum('ik,jl',a,b)+np.einsum('il,jk',a,b))
print("R⊠ˢR =\n",KM(sboxtimes(R,R)))

ℝ = rot6(θ, ϕ, ψ)
print("ℝ =\n",ℝ)
```

```
R⊠ˢR =
 [[ 0.01679518  0.60820482  0.375      -0.67539145 -0.11223363  0.14293294]
 [ 0.49232344  0.13267656  0.375       0.31544796  0.60765334  0.36144095]
 [ 0.49088137  0.25911863  0.25        0.35994349 -0.49541971 -0.50437388]
 [-0.69523004  0.26221734  0.4330127   0.49384417 -0.07821723  0.10196691]
```

```
 [ 0.12840906 -0.56142176  0.4330127  -0.07821723 -0.49384417  0.48043389]
 [-0.12859754 -0.40173255  0.53033009 -0.25451848  0.35031463 -0.59441032]]
ℝ =
 [[ 0.01679518  0.60820482  0.375      -0.67539145 -0.11223363  0.14293294]
 [ 0.49232344  0.13267656  0.375       0.31544796  0.60765334  0.36144095]
 [ 0.49088137  0.25911863  0.25        0.35994349 -0.49541971 -0.50437388]
 [-0.69523004  0.26221734  0.4330127   0.49384417 -0.07821723  0.10196691]
 [ 0.12840906 -0.56142176  0.4330127  -0.07821723 -0.49384417  0.48043389]
 [-0.12859754 -0.40173255  0.53033009 -0.25451848  0.35031463 -0.59441032]]
```

3 Special tensors

Objectives

This tutorial presents the matrix representation of isotropic tensors of second and fourth orders as well as Walpole tensors useful for transverse isotropy. It strongly relies on conventions of tensor algebra introduced in Appendix A especially in terms of products and contractions.

Download

- Python script
- Jupyter notebook

Imports

```
import numpy as np
from echoes import *
import math

np.set_printoptions(precision=6, suppress=True)
# to display only 6 significant digits of array components
```

3.1 Second-order identity

The second-order identity $\mathbf{1} = \delta_{ij}\underline{e}_i \otimes \underline{e}_j$ is given in Kelvin-Mandel notation (i.e. vector of $\mathbb{R}^6$) by the constant vector `Id2`

```
δ = Id2
print("1 (Kelvin-Mandel notation) =\n",δ)
print("1 =\n",invKM(δ))
```

```
1 (Kelvin-Mandel notation) =
 [1. 1. 1. 0. 0. 0.]
1 =
 [[1. 0. 0.]
 [0. 1. 0.]
```

```
 [0. 0. 1.]]
```

3.2 Fourth-order isotropic tensors

As detailed in Section A.4, the fourth-order identity tensor is

$$\mathbb{I} = \mathbf{1} \overset{s}{\boxtimes} \mathbf{1} = \frac{\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}}{2} \, \underline{e}_i \otimes \underline{e}_j \otimes \underline{e}_k \otimes \underline{e}_l$$

and the projectors

$$\mathbb{J} = \frac{1}{3}\mathbf{1} \otimes \mathbf{1} \quad \text{and} \quad \mathbb{K} = \mathbb{I} - \mathbb{J} = \mathbf{1} \overset{s}{\boxtimes} \mathbf{1} - \frac{1}{3}\mathbf{1} \otimes \mathbf{1}$$

which are provided in `echoes` by `Id4`, `J4` and `K4`

```python
for T in [Id4, J4, K4]:
    print(T)
```

```
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
[[0.333333 0.333333 0.333333 0.       0.       0.      ]
 [0.333333 0.333333 0.333333 0.       0.       0.      ]
 [0.333333 0.333333 0.333333 0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]]
[[ 0.666667 -0.333333 -0.333333  0.        0.        0.      ]
 [-0.333333  0.666667 -0.333333  0.        0.        0.      ]
 [-0.333333 -0.333333  0.666667  0.        0.        0.      ]
 [ 0.        0.        0.        1.        0.        0.      ]
 [ 0.        0.        0.        0.        1.        0.      ]
 [ 0.        0.        0.        0.        0.        1.      ]]
```

3.3 Walpole bases

The Walpole bases are useful to decompose transversely isotropic fourth-order tensors. They are presented in Section A.5. The Kelvin-Mandel representation of the i^{th} Walpole tensor oriented along an axis $\underline{n}$ is constructed by `W(i,n=e₃)` ($i \in \{0,..,5\}$ and the normal is by default oriented along the third axis). The symmetrized version is provided by `WS(i,n=e₃)` ($i \in \{0,..,4\}$).

> **Warning**
>
> Note again the shift in indices between the Python convention starting at 0 and the tensors presented in Section A.5.

```
for i in range(6):
    print("W"+str(i+1)+" =\n",W(i))

for i in range(5):
    print("Wˢ"+str(i+1)+" =\n",WS(i))
```

```
W1 =
 [[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
W2 =
 [[0.5 0.5 0.  0.  0.  0. ]
 [0.5 0.5 0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]]
W3 =
 [[0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.707107 0.707107 0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]]
W4 =
 [[0.       0.       0.707107 0.       0.       0.      ]
 [0.       0.       0.707107 0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]]
W5 =
 [[ 0.5 -0.5  0.   0.   0.   0. ]
 [-0.5  0.5  0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0. ]
```

```
 [ 0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   1. ]]
W6 =
 [[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0.]]
Wˢ1 =
 [[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
Wˢ2 =
 [[0.5 0.5 0.  0.  0.  0. ]
 [0.5 0.5 0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0. ]]
Wˢ3 =
 [[0.       0.       0.707107 0.       0.       0.      ]
 [0.       0.       0.707107 0.       0.       0.      ]
 [0.707107 0.707107 0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]
 [0.       0.       0.       0.       0.       0.      ]]
Wˢ4 =
 [[ 0.5 -0.5  0.   0.   0.   0. ]
 [-0.5  0.5  0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   1. ]]
Wˢ5 =
 [[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

4 The `tensor` object

Objectives

This tutorial presents the object `tensor` which is the main structure of `echoes` allowing to represent symmetric second-order or fourth-order tensors both in matrix and synthetic forms and containing information about anisotropy.

Download

- Python script
- Jupyter notebook

Imports

```
import numpy as np
from echoes import *
import math

np.set_printoptions(precision=6, suppress=True)
# to display only 6 significant digits of array components
```

4.1 Definition of the `tensor` object

The `tensor` object is a structure designed to represent **symmetric** second-order or fourth-order tensors. It gathers four members

- **the material symmetry**: `ISO` (isotropic), `TI` (transversely isotropic), `ORTHO` (orthotropic) or `ANISO` (anisotropic) [*other material symmetries exist but have not been implemented*]
- **a condensed vector of parameters** [*as shown below the size of this vector unambiguously defines the order of the tensor*]
- **a vector of angles (in radians)** [*if required*]
- **a matrix** [3×3 *for a second-order or* 6×6 *for a fourth-order tensor*]

> ⚠ Warning
>
> It is important to note that the `tensor` object is designed only to contain or retrieve information about a symmetric second-order or fourth-order tensor (for instance the material symmetry) and not to perform calculation. Although simple calculations such as addition, subtraction, multiplication by a scalar, inversion are available, most of other operations (products, contractions...) have not been implemented. To do so, it is necessary to extract the parameters or matrix and perform operations using usual `numpy` or `scipy` operations (`dot`, `einsum`...) before eventually building a new `tensor`.

If `C` is a python variable of `tensor` type, its members can be accessed by

- **material symmetry**: `C.sym`
- **parameters**: `C.param` or `C.p`
- **angles**: `C.angles`
- **matrix**: `C.array` or `C.a`

4.2 Second-order tensors

A symmetric second-order `tensor` object is designed by means of its three eigenvalues and the orientation of the eigenbasis through three Euler angles θ, ϕ and ψ (see Fig. A.1). It is built using one of the following constructors

- `T = tensor(T₁, T₂, T₃, θ=0, ϕ=0, ψ=0)` # default values of the angles are 0
- `T = tensor([T₁, T₂, T₃], angles=[0, 0, 0])`

> ⚠ Warning
>
> - The eigenvalues and eigenvectors are systematically reordered in decreasing order of eigenvalues and angles are recomputed accordingly.
> - The eigenvalues are analyzed in order to characterize the symmetry type between `ISO`, `TI` and `ORTHO`.
> - The Euler angles are not unique since unit eigenvectors are determined up to a multiplicative factor of ± 1.
> - Note that the six parameters of the constructor `tensor` should not be confused with the components of the Kelvin-Mandel representation of symmetric second-order tensors (see Section A.2).

```
T = tensor(2., 1., 3.) ; print(T)
```

```
Order 2 ORTHO tensor | Param(size=3)=[ 3 2 1 ] | Angles(size=3)=[ 1.5708 4.71239 3.14159 ]
[ 2 0 0
  0 1 0
  0 0 3 ]
```

```
π = math.pi
T = tensor(2., 1., 2., π/3, π/4, π/5) ; print(T)
```

```
Order 2 TI tensor | Param(size=3)=[ 2 2 1 ] | Angles(size=3)=[ 2.10486 5.84624 0.619825 ]
[ 1.3918 0.284068 0.396985
  0.284068 1.86732 -0.185416
  0.396985 -0.185416 1.74088 ]
```

```
T = tensor([2., 1., 3.], angles=[π/3, π/4, π/5]) ; print(T)
```

```
Order 2 ORTHO tensor | Param(size=3)=[ 3 2 1 ] | Angles(size=3)=[ 2.10486 5.84624 2.19062 ]
[ 1.7668 0.659068 0.703171
  0.659068 2.24232 0.120771
  0.703171 0.120771 1.99088 ]
```

A symmetric second-order `tensor` can also be built from a symmetric 3×3 matrix. In this case a diagonalization is performed so as to find the `param` vector (the eigenvalues) and the `angles` determining the orientation of the eigenvectors (see Fig. A.1)

```
M = np.array([ [1.7668, 0.659068, 0.703171],
               [0.659068, 2.24232, 0.120771],
               [0.703171, 0.120771, 1.99088] ])
T = tensor(M) ; print(T)
```

```
Order 2 ORTHO tensor | Param(size=3)=[ 3 2 1 ] | Angles(size=3)=[ 1.03673 2.70464 0.950971 ]
[ 1.7668 0.659068 0.703171
  0.659068 2.24232 0.120771
  0.703171 0.120771 1.99088 ]
```

4.3 Isotropic fourth-order tensors (`ISO`)

The special isotropic tensors $\mathbb{I}$, $\mathbb{J}$ and $\mathbb{K}$ are available under the global variables `tId4`, `tJ4` and `tK4`.

```
for T in [tId4, tJ4, tK4]:
    print(T)
```

```
Order 4 ISO tensor | Param(size=2)=[ 1 1 ] | Angles(size=0)=[ ]
[ 1 0 0 0 0 0
  0 1 0 0 0 0
  0 0 1 0 0 0
  0 0 0 1 0 0
```

```
  0 0 0 0 1 0
  0 0 0 0 0 1 ]

Order 4 ISO tensor | Param(size=2)=[ 1 0 ] | Angles(size=0)=[ ]
[ 0.333333 0.333333 0.333333 0 0 0
  0.333333 0.333333 0.333333 0 0 0
  0.333333 0.333333 0.333333 0 0 0
  0 0 0 0 0 0
  0 0 0 0 0 0
  0 0 0 0 0 0 ]

Order 4 ISO tensor | Param(size=2)=[ 1.11022e-16 1 ] | Angles(size=0)=[ ]
[ 0.666667 -0.333333 -0.333333 0 0 0
  -0.333333 0.666667 -0.333333 0 0 0
  -0.333333 -0.333333 0.666667 0 0 0
  0 0 0 1 0 0
  0 0 0 0 1 0
  0 0 0 0 0 1 ]
```

Any fourth-order tensor depends on two parameters α and β such that $\mathbb{T} = \alpha\mathbb{J} + \beta\mathbb{K}$ and can be built by one of these constructors

- `T = tensor(α, β)`
- `T = tensor([α, β])`
- `T = tensor(np.array([α, β]))`

```python
α, β = 7.2, 6.1
𝕋 = tensor(α, β)
print("𝕋 =\n", 𝕋)
print("𝕋.sym =", 𝕋.sym)
print("𝕋.a =\n", 𝕋.a)
print("𝕋.angles =", 𝕋.angles)
print("𝕋.p =", 𝕋.p)
```

```
𝕋 =
 Order 4 ISO tensor | Param(size=2)=[ 7.2 6.1 ] | Angles(size=0)=[ ]
[ 6.46667 0.366667 0.366667 0 0 0
  0.366667 6.46667 0.366667 0 0 0
  0.366667 0.366667 6.46667 0 0 0
  0 0 0 6.1 0 0
  0 0 0 0 6.1 0
  0 0 0 0 0 6.1 ]

𝕋.sym = ISO
𝕋.a =
 [[6.466667 0.366667 0.366667 0.        0.        0.       ]
```

```
 [0.366667 6.466667 0.366667 0.       0.       0.       ]
 [0.366667 0.366667 6.466667 0.       0.       0.       ]
 [0.       0.       0.       6.1      0.       0.       ]
 [0.       0.       0.       0.       6.1      0.       ]
 [0.       0.       0.       0.       0.       6.1      ]]
𝕋.angles = []
𝕋.p = [7.2 6.1]
```

```
𝕀, 𝕁, 𝕂 = tId4, tJ4, tK4 # just to make it look nice!

𝕋2 = α*𝕁 + β*𝕂
print("𝕋2 =\n", 𝕋2)
assert np.allclose(𝕋.a, 𝕋2.a), "error 𝕋.a and 𝕋2.a should be equal"
assert np.allclose(𝕋.p, 𝕋2.p), "error 𝕋.p and 𝕋2.p should be equal"
```

```
𝕋2 =
 Order 4 ISO tensor | Param(size=2)=[ 7.2 6.1 ] | Angles(size=0)=[ ]
[ 6.46667 0.366667 0.366667 0 0 0
  0.366667 6.46667 0.366667 0 0 0
  0.366667 0.366667 6.46667 0 0 0
  0 0 0 6.1 0 0
  0 0 0 0 6.1 0
  0 0 0 0 0 6.1 ]
```

A symmetric fourth-order tensor representing a stiffness tensor can also be built by means of

- *bulk and shear moduli* `stiff_kmu(k, μ)` $\rightarrow 3\,k\mathbb{J} + 2\,\mu\mathbb{K}$
- *Young modulus and Poisson ratio* `stiff_Enu(E, ν)` $\rightarrow \frac{E}{1-2\,\nu}\mathbb{J} + \frac{E}{1+\nu}\mathbb{K}$
- *Lamé moduli* `stiff_lambdamu(λ, μ)` $\rightarrow 3\,\lambda\mathbb{J} + 2\,\mu\mathbb{I}$

Some converters are also available (see conversion table)

- `E = E_from_kmu(k, μ)`
- `ν = nu_from_kmu(k, μ)`
- `k = k_from_Enu(E, ν)`
- `μ = mu_from_Enu(E, ν)`
- `E, ν = Enu_from_kmu(k, μ)`
- `k, μ = kmu_from_Enu(E, ν)`

```
k, μ = 72.5, 32.7
ℂ = stiff_kmu(k, μ)
print("ℂ =\n", ℂ)
print(f"E = {round(ℂ.E,2)} ; ν = {round(ℂ.nu,2)} ; λ = {round(ℂ.lamelambda,2)} ; μ = {round(ℂ
E, ν = Enu_from_kmu(k, μ)
assert np.allclose([k, μ], [*kmu_from_Enu(E, ν)]), "error kmu_from_Enu(E, ν) should return k,
```

```
ℂ =
 Order 4 ISO tensor | Param(size=2)=[ 217.5 65.4 ] | Angles(size=0)=[ ]
[ 116.1 50.7 50.7 0 0 0
  50.7 116.1 50.7 0 0 0
  50.7 50.7 116.1 0 0 0
  0 0 0 65.4 0 0
  0 0 0 0 65.4 0
  0 0 0 0 0 65.4 ]

E = 85.28 ; ν = 0.3 ; λ = 50.7 ; μ = 32.7
```

4.4 Symmetric transversely isotropic fourth-order tensors (`TI`)

Such a tensor depends on 5 parameters and 2 angles θ and ϕ defining the orientation of the normal to the isotropy plane (see Fig. A.1).

The decomposition on the symmetric Walpole tensors writes by means of five parameters $(t_i)_{i=1,\ldots,5}$. In the frame where $\underline{n} = \underline{e}_3$ the matrix is given by

$$\mathbb{T} = \sum_{i=1}^{5} t_i \mathbb{W}_i^s \mapsto \begin{pmatrix} \frac{t_2+t_4}{2} & \frac{t_2-t_4}{2} & \frac{\sqrt{2}\,t_3}{2} & 0 & 0 & 0 \\ \frac{t_2-t_4}{2} & \frac{t_2+t_4}{2} & \frac{\sqrt{2}\,t_3}{2} & 0 & 0 & 0 \\ \frac{\sqrt{2}\,t_3}{2} & \frac{\sqrt{2}\,t_3}{2} & t_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_4 \end{pmatrix}$$

A fourth-order transversely isotropic tensor can be built by one of these constructors

- `T = tensor(t1, t2, t3, t4, t5, θ=0, ϕ=0)`
- `T = tensor(param, angles=[0, 0])`

where `param` is a `list` or a `numpy.ndarray` of 5 items and `angles` is a `list` or a `numpy.ndarray` of 2 items `θ, ϕ` (if `angles` is omitted the angles are null and the axis is oriented along $\underline{e}_3$)

```
param = [3., 4., math.sqrt(2), 2., 1.]
angles = [π/3, π/4]
print(tensor(param))
print(tensor(param, angles))
```

```
Order 4 TI tensor | Param(size=5)=[ 3 4 1.41421 2 1 ] | Angles(size=2)=[ 0 0 ]
[ 3 1 1 0 0 0
  1 3 1 0 0 0
  1 1 3 0 0 0
  0 0 0 1 0 0
  0 0 0 0 1 0
```

```
  0 0 0 0 0 2 ]

Order 4 TI tensor | Param(size=5)=[ 3 4 1.41421 2 1 ] | Angles(size=2)=[ 1.0472 0.785398 ]
[ 2.53125 1.28125 1.1875 0.32476 -0.108253 -0.132583
  1.28125 2.53125 1.1875 -0.108253 0.32476 -0.132583
  1.1875 1.1875 2.625 -0.216506 -0.216506 0.265165
  0.32476 -0.108253 -0.216506 1.75 3.33067e-16 0.153093
  -0.108253 0.32476 -0.216506 3.33067e-16 1.75 0.153093
  -0.132583 -0.132583 0.265165 0.153093 0.153093 1.8125 ]
```

It is also possible to build a `TI` tensor by means of 5 C_{ijkl} moduli:

$$(T_{1111}, T_{1122}, T_{1133}, T_{3333}, T_{2323}) \mapsto \begin{pmatrix} T_{1111} & T_{1122} & T_{1133} & 0 & 0 & 0 \\ T_{1122} & T_{1111} & T_{1133} & 0 & 0 & 0 \\ T_{1133} & T_{1133} & T_{3333} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\,T_{2323} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\,T_{2323} & 0 \\ 0 & 0 & 0 & 0 & 0 & T_{1111}-T_{1122} \end{pmatrix}$$

by

```
T = stiff_TI(T₁₁₁₁, T₁₁₂₂, T₁₁₃₃, T₃₃₃₃, T₂₃₂₃, θ=0, ϕ=0)
```

```python
T = stiff_TI(6.2, 5.6, 1.3, 2.1, 2.3)
assert all(v > 0 for v in np.linalg.eigvals(T.a)), "one eigenvalue is negative"
# check that T is positive definite
print(T)
print(stiff_TI(6.2, 5.6, 1.3, 2.1, 2.3, π/3, π/4))
```

```
Order 4 TI tensor | Param(size=5)=[ 2.1 11.8 1.83848 0.6 4.6 ] | Angles(size=2)=[ 0 0 ]
[ 6.2 5.6 1.3 0 0 0
  5.6 6.2 1.3 0 0 0
  1.3 1.3 2.1 0 0 0
  0 0 0 4.6 0 0
  0 0 0 0 4.6 0
  0 0 0 0 0 0.6 ]

Order 4 TI tensor | Param(size=5)=[ 2.1 11.8 1.83848 0.6 4.6 ] | Angles(size=2)=[ 1.0472 0.785
[ 5.48281 1.88281 2.58437 -2.43028 -0.698233 -0.855157
  1.88281 5.48281 2.58438 -0.698233 -2.43028 -0.855157
  2.58437 2.58438 5.83125 -0.50879 -0.50879 -2.74446
  -2.43028 -0.698233 -0.50879 2.44375 0.84375 0.421006
  -0.698233 -2.43028 -0.50879 0.84375 2.44375 0.421006
  -0.855157 -0.855157 -2.74446 0.421006 0.421006 2.61563 ]
```

A engineering construction based on directional moduli and Poisson ratio is also available:

$$(E_1, E_3, \nu_{12}, \nu_{31}, G_{31}) \mapsto \begin{pmatrix} \frac{1}{E_1} & \frac{-\nu_{12}}{E_1} & \frac{-\nu_{31}}{E_3} & 0 & 0 & 0 \\ \frac{-\nu_{12}}{E_1} & \frac{1}{E_1} & \frac{-\nu_{31}}{E_3} & 0 & 0 & 0 \\ \frac{-\nu_{31}}{E_3} & \frac{-\nu_{31}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2\,G_{31}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2\,G_{31}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1+\nu_{12}}{E_1} \end{pmatrix}$$

by

```
S = comp_TI(E₁, E₃, ν₁₂, ν₃₁, G₃₁, θ=0, ϕ=0)
```

```
S = comp_TI(1., 4., 0.2, 0.1, 2.)
print(S)
print(inv(S))
print(comp_TI(1., 4., 0.2, 0.1, 2., π/3, π/4))
```

```
Order 4 TI tensor | Param(size=5)=[ 0.25 0.8 -0.0353553 1.2 0.25 ] | Angles(size=2)=[ 0 0 ]
[ 1 -0.2 -0.025 0 0 0
  -0.2 1 -0.025 0 0 0
  -0.025 -0.025 0.25 0 0 0
  0 0 0 0.25 0 0
  0 0 0 0 0.25 0
  0 0 0 0 0 1.2 ]

Order 4 TI tensor | Param(size=5)=[ 4.02516 1.25786 0.177888 0.833333 4 ] | Angles(size=2)=[
[ 1.0456 0.212264 0.125786 0 0 0
  0.212264 1.0456 0.125786 0 0 0
  0.125786 0.125786 4.02516 0 0 0
  0 0 0 4 0 0
  0 0 0 0 4 0
  0 0 0 0 0 0.833333 ]

Order 4 TI tensor | Param(size=5)=[ 0.25 0.8 -0.0353553 1.2 0.25 ] | Angles(size=2)=[ 1.0472
[ 0.53125 0.04375 -0.015625 0.205681 -0.205681 -0.251907
  0.04375 0.53125 -0.015625 -0.205681 0.205681 -0.251907
  -0.015625 -0.015625 0.6625 -0.248982 -0.248982 0.198874
  0.205681 -0.205681 -0.248982 0.75625 -0.20625 -0.107165
  -0.205681 0.205681 -0.248982 -0.20625 0.75625 -0.107165
  -0.251907 -0.251907 0.198874 -0.107165 -0.107165 0.7125 ]
```

4.5 Symmetric orthotropic fourth-order tensors (ORTHO)

Such a tensor depends on 9 parameters and 3 angles θ, ϕ and ψ defining the orientation of an orthonormal frame (see Fig. A.1).

The convention here relies on the components $T_{1111}, T_{1122}, T_{1133}, T_{2222}, T_{2233}, T_{3333}, T_{2323}, T_{3131}, T_{1212}$ as inputs. In the canonical frame the matrix is given by

$$(T_{1111}, T_{1122}, T_{1133}, T_{2222}, T_{2233}, T_{3333}, T_{2323}, T_{3131}, T_{1212}) \mapsto \begin{pmatrix} T_{1111} & T_{1122} & T_{1133} & 0 & 0 & 0 \\ T_{1122} & T_{2222} & T_{2233} & 0 & 0 & 0 \\ T_{1133} & T_{2233} & T_{3333} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\,T_{2323} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\,T_{3131} & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\,T_{1212} \end{pmatrix}$$

A fourth-order isotropic tensor can be built by one of these constructors

- `T = tensor(T₁₁₁₁, T₁₁₂₂, T₁₁₃₃, T₂₂₂₂, T₂₂₃₃, T₃₃₃₃, T₂₃₂₃, T₃₁₃₁, T₁₂₁₂, θ=0, ϕ=0, ψ=0)`
- `T = tensor(param, angles=[0, 0, 0])`

where `param` is a `list` or a `numpy.ndarray` of 9 items and `angles` is a `list` or a `numpy.ndarray` of 3 items `θ, ϕ, ψ` (if `angles` is omitted the frame is the canonical one).

```
T = tensor([9.,3.,5.,6.,3.,7.,1.,4.,3.])
assert all(v > 0 for v in np.linalg.eigvals(T.a)), "one eigenvalue is negative" # check that
print(T)
print(tensor([9.,3.,5.,6.,3.,7.,1.,4.,3.], [π/3, π/4, π/5]))
```

```
Order 4 ORTHO tensor | Param(size=9)=[ 9 3 5 6 3 7 1 4 3 ] | Angles(size=3)=[ 0 0 0 ]
[ 9 3 5 0 0 0
  3 6 3 0 0 0
  5 3 7 0 0 0
  0 0 0 2 0 0
  0 0 0 0 8 0
  0 0 0 0 0 6 ]

Order 4 ORTHO tensor | Param(size=9)=[ 9 3 5 6 3 7 1 4 3 ] | Angles(size=3)=[ 1.0472 0.785398
[ 5.83478 3.57431 3.79988 0.0416272 0.710013 -0.509607
  3.57431 9.74488 2.26743 -1.24237 -0.0801039 0.934783
  3.79988 2.26743 9.1371 -0.976365 1.31117 0.522827
  0.0416272 -1.24237 -0.976365 3.25126 -0.192475 -0.544059
  0.710013 -0.0801039 1.31117 -0.192475 5.36512 -1.53671
  -0.509607 0.934783 0.522827 -0.544059 -1.53671 4.66686 ]
```

4.6 Symmetric anisotropic fourth-order tensors (`ANISO`)

Such a tensor depends on 21 parameters, by convention here the 21 components of the Kelvin-Mandel notation. Components are considered line after line of the upper triangular part of the Kelvin-Mandel matrix

$$\begin{matrix}(T_{11}, & T_{12}, & T_{13}, & T_{14}, & T_{15}, & T_{16} \\ & T_{22}, & T_{23}, & T_{24}, & T_{25}, & T_{26} \\ & & T_{33}, & T_{34}, & T_{35}, & T_{36} \\ & & & T_{44}, & T_{45}, & T_{46} \\ & & & & T_{55}, & T_{56} \\ & & & & & T_{66})\end{matrix} \mapsto \begin{pmatrix} T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & T_{16} \\ T_{12} & T_{22} & T_{23} & T_{24} & T_{25} & T_{26} \\ T_{13} & T_{23} & T_{33} & T_{34} & T_{35} & T_{36} \\ T_{14} & T_{24} & T_{34} & T_{44} & T_{45} & T_{46} \\ T_{15} & T_{25} & T_{35} & T_{45} & T_{55} & T_{56} \\ T_{16} & T_{26} & T_{36} & T_{46} & T_{56} & T_{66} \end{pmatrix}$$

> **Warning**
>
> It is important to recall here that the parameters T_{IJ} with $1 \leq I, J \leq 6$ do not all coincide with the components T_{ijkl}. Indeed the convention presented in Section A.2 implies that $T_{11} = T_{1111}$ but $T_{14} = \sqrt{2}\, T_{1123}$ and $T_{44} = 2\, T_{2323}$.

Such a tensor is built by

```
T = tensor(param)
```

where `param` is a `list` or a `numpy.ndarray` of 21 items.

```
param = [ 2.66011, 1.26432, 0.662772, 1.9402, 1.54905, 1.10384, 3.6072, 1.78964, 2.0247, 1.270
T = tensor(param)
assert all(v > 0 for v in np.linalg.eigvals(T.a)), "one eigenvalue is negative" # check that t
print(T)
```

```
Order 4 ANISO tensor | Param(size=21)=[ 2.66011 1.26432 0.662772 1.9402 1.54905 1.10384 3.607
[ 2.66011 1.26432 0.662772 1.9402 1.54905 1.10384
  1.26432 3.6072 1.78964 2.0247 1.2701 1.1089
  0.662772 1.78964 2.743 1.3367 1.2962 0.897632
  1.9402 2.0247 1.3367 4.42684 2.05632 1.52686
  1.54905 1.2701 1.2962 2.05632 3.54431 1.3445
  1.10384 1.1089 0.897632 1.52686 1.3445 1.99356 ]
```

4.7 Construction of a fourth-order `tensor` from a 6×6 matrix and projections

Given a symmetric positive definite 6×6 matrix `M` (warning: these properties are not checked on construction), a tensor can be built by one of the function

- `T = tensor(M)`
- `T = tensor(M, angles=[θ, ϕ])`

By default, this function finds out whether `M` represents an `ISO`, `TI` or `ORTHO` tensor in the canonical frame if `angles` is not precised or in the frame defined by `angles`. If none of these symmetries is suitable, the tensor is considered `ANISO`.

```
angles = [π/3, π/4, π/5]
T = tensor([9.,3.,5.,6.,3.,7.,1.,4.,3.], angles)
M = T.a
print("M =\n", M) # the information about ORTHO symmetry has vanished in M
T2 = tensor(M, angles=angles)
print(T2)
```

```
M =
 [[ 5.834782  3.574311  3.799884  0.041627  0.710013 -0.509607]
 [ 3.574311  9.744881  2.267425 -1.242374 -0.080104  0.934783]
 [ 3.799884  2.267425  9.137098 -0.976365  1.311174  0.522827]
 [ 0.041627 -1.242374 -0.976365  3.25126  -0.192475 -0.544059]
 [ 0.710013 -0.080104  1.311174 -0.192475  5.36512  -1.536706]
 [-0.509607  0.934783  0.522827 -0.544059 -1.536706  4.666858]]
Order 4 ORTHO tensor | Param(size=9)=[ 9 3 5 6 3 7 1 4 3 ] | Angles(size=3)=[ 1.0472 0.785398
[ 5.83478 3.57431 3.79988 0.0416272 0.710013 -0.509607
  3.57431 9.74488 2.26743 -1.24237 -0.0801039 0.934783
  3.79988 2.26743 9.1371 -0.976365 1.31117 0.522827
  0.0416272 -1.24237 -0.976365 3.25126 -0.192475 -0.544059
  0.710013 -0.0801039 1.31117 -0.192475 5.36512 -1.53671
  -0.509607 0.934783 0.522827 -0.544059 -1.53671 4.66686 ]
```

Given a 6×6 matrix supposed to represent a tensor in the canonical frame, it is possible to find the closest tensor (in the sense of least-squares relatively to the euclidean matrix norm) among a family of given material symmetry. The syntax is simply `T = tensor(M, SYM)` where `SYM` denotes the chosen class of symmetry (`ISO`, `TI`, `ORTHO`).

For example the closest `ISO` tensor is calculated by

```
T = tensor(M, ISO)
```

In this case it boils down to the "isotropisation" of the tensor $\mathbb{T}$ given by (Bornert et al., 2001)

$$\mathrm{ISO}(\mathbb{T}) = (\mathbb{T} :: \mathbb{J})\ \mathbb{J} + \left(\frac{\mathbb{T} :: \mathbb{K}}{5}\right)\ \mathbb{K}$$

```
M = np.array([ [2.66011, 1.26432, 0.662772, 1.9402, 1.54905, 1.10384],
               [1.26432, 3.6072, 1.78964, 2.0247, 1.2701, 1.1089],
               [0.662772, 1.78964, 2.743, 1.3367, 1.2962, 0.897632],
               [1.9402, 2.0247 ,1.3367, 4.42684, 2.05632, 1.52686],
```

```
                [1.54905, 1.2701, 1.2962, 2.05632, 3.54431, 1.3445],
                [1.10384, 1.1089, 0.897632, 1.52686, 1.3445, 1.99356] ])

T = tensor(M,ISO)
print("T =\n", T)

cont4=lambda T1,T2:np.einsum('ij,ij',T1,T2) # T1 and T2 in KM notation so 2 indices
T2 = tensor(cont4(M,J4), cont4(M,K4)/5)
print("T2 =\n", T2)
```

```
T =
 Order 4 ISO tensor | Param(size=2)=[ 5.48126 2.69875 ] | Angles(size=0)=[ ]
[ 3.62625 0.927502 0.927502 0 0 0
  0.927502 3.62625 0.927502 0 0 0
  0.927502 0.927502 3.62625 0 0 0
  0 0 0 2.69875 0 0
  0 0 0 0 2.69875 0
  0 0 0 0 0 2.69875 ]

T2 =
 Order 4 ISO tensor | Param(size=2)=[ 5.48126 2.69875 ] | Angles(size=0)=[ ]
[ 3.62625 0.927502 0.927502 0 0 0
  0.927502 3.62625 0.927502 0 0 0
  0.927502 0.927502 3.62625 0 0 0
  0 0 0 2.69875 0 0
  0 0 0 0 2.69875 0
  0 0 0 0 0 2.69875 ]
```

Check a well-known result stating that the inverse of the closest isotropic tensor is in general **not** the closest isotropic tensor of the inverse.

```
M = np.array([ [2.66011, 1.26432, 0.662772, 1.9402, 1.54905, 1.10384],
               [1.26432, 3.6072, 1.78964, 2.0247, 1.2701, 1.1089],
               [0.662772, 1.78964, 2.743, 1.3367, 1.2962, 0.897632],
               [1.9402, 2.0247 ,1.3367, 4.42684, 2.05632, 1.52686],
               [1.54905, 1.2701, 1.2962, 2.05632, 3.54431, 1.3445],
               [1.10384, 1.1089, 0.897632, 1.52686, 1.3445, 1.99356] ])
T = tensor(M, ISO)
print("inv(ISO(M)) =\n", inv(tensor(M, ISO)))
print("ISO(inv(M)) =\n", tensor(np.linalg.inv(M), ISO))
```

```
inv(ISO(M)) =
 Order 4 ISO tensor | Param(size=2)=[ 0.18244 0.370542 ] | Angles(size=0)=[ ]
[ 0.307841 -0.0627006 -0.0627006 0 0 0
  -0.0627006 0.307841 -0.0627006 0 0 0
```

```
  -0.0627006 -0.0627006 0.307841 0 0 0
  0 0 0 0.370542 0 0
  0 0 0 0 0.370542 0
  0 0 0 0 0 0.370542 ]

ISO(inv(M)) =
 Order 4 ISO tensor | Param(size=2)=[ 0.416399 0.615133 ] | Angles(size=0)=[ ]
[ 0.548889 -0.0662448 -0.0662448 0 0 0
  -0.0662448 0.548889 -0.0662448 0 0 0
  -0.0662448 -0.0662448 0.548889 0 0 0
  0 0 0 0.615133 0 0
  0 0 0 0 0.615133 0
  0 0 0 0 0 0.615133 ]
```

The closest `TI` tensor in the frame defined by given angles is calculated by

`T = tensor(M, TI, angles=[θ, ϕ], epsrel=1.e-3)`

where `epsrel` is a relative stop criterion for the iterative least-square algorithm.

```
print(tensor(M, TI, angles=[π/3,π/4]))
```

```
Order 4 TI tensor | Param(size=5)=[ 9.90471 1.93007 0.947144 1.68894 1.88118 ] | Angles(size=
[ 3.29543 1.46231 1.08371 1.31151 1.39476 1.70822
  1.46231 3.29543 1.08371 1.39476 1.31151 1.70822
  1.08371 1.08371 2.59348 1.03685 1.03685 1.16792
  1.31151 1.39476 1.03685 3.04891 1.31191 1.57733
  1.39476 1.31151 1.03685 1.31191 3.04891 1.57733
  1.70822 1.70822 1.16792 1.57733 1.57733 3.69285 ]
```

One can also optimize with respect to the angles so as to find the best frame for a given symmetry

`T = tensor(M, TI, epsrel=1.e-3, optiangles=True)`

where `epsrel` is a relative stop criterion for the iterative least-square algorithm.

```
print(tensor(M, TI, epsrel=1.e-3, optiangles=True))
```

```
Order 4 TI tensor | Param(size=5)=[ 10.2696 1.81638 0.89854 1.70466 1.73988 ] | Angles(size=2
[ 2.52626 1.12576 1.07821 1.27354 1.04336 1.07127
  1.12576 3.30972 1.51116 1.80824 1.44496 1.49908
  1.07821 1.51116 3.17755 1.73199 1.39846 1.42041
  1.27354 1.80824 1.73199 3.77321 1.65983 1.70366
  1.04336 1.44496 1.39846 1.65983 3.05771 1.38017
  1.07127 1.49908 1.42041 1.70366 1.38017 3.13058 ]
```

Exercise

1. Find the closest `ORTHO` tensor `T` from the previous matrix `M` (also optimizing the angles)
2. Build a new tensor from the matrix of `T` (in the canonical frame)
3. Build a new tensor from the matrix of `T` and the angles of `T`
4. Find the tensor of best symmetry from the matrix of `T` with angle optimization

```
M = np.array([ [2.66011, 1.26432, 0.662772, 1.9402, 1.54905, 1.10384],
               [1.26432, 3.6072, 1.78964, 2.0247, 1.2701, 1.1089],
               [0.662772, 1.78964, 2.743, 1.3367, 1.2962, 0.897632],
               [1.9402, 2.0247 ,1.3367, 4.42684, 2.05632, 1.52686],
               [1.54905, 1.2701, 1.2962, 2.05632, 3.54431, 1.3445],
               [1.10384, 1.1089, 0.897632, 1.52686, 1.3445, 1.99356] ])
T = tensor(M, ORTHO, optiangles=True) ; print(T)
print(tensor(T.a))
print(tensor(T.a, angles = T.angles))
print(tensor(T.a, optiangles = True))
```

```
Order 4 ORTHO tensor | Param(size=9)=[ 2.06277 -0.00326333 -
0.375452 1.57933 1.64478 10.2695 1.03801 0.701183 0.792497 ] | Angles(size=3)=[ 0.925546 0.8
[ 2.80357 1.20599 1.14979 1.63413 1.50234 1.16022
  1.20599 3.38542 1.441 2.18362 1.51678 1.17623
  1.14979 1.441 2.6612 1.44697 1.1142 0.974349
  1.63413 2.18362 1.44697 4.52485 2.11525 1.42722
  1.50234 1.51678 1.1142 2.11525 3.37899 1.16562
  1.16022 1.17623 0.974349 1.42722 1.16562 2.22099 ]

Order 4 ANISO tensor | Param(size=21)=[ 2.80357 1.20599 1.14979 1.63413 1.50234 1.16022 3.38
[ 2.80357 1.20599 1.14979 1.63413 1.50234 1.16022
  1.20599 3.38542 1.441 2.18362 1.51678 1.17623
  1.14979 1.441 2.6612 1.44697 1.1142 0.974349
  1.63413 2.18362 1.44697 4.52485 2.11525 1.42722
  1.50234 1.51678 1.1142 2.11525 3.37899 1.16562
  1.16022 1.17623 0.974349 1.42722 1.16562 2.22099 ]

Order 4 ORTHO tensor | Param(size=9)=[ 2.06277 -0.00326333 -
0.375452 1.57933 1.64478 10.2695 1.03801 0.701183 0.792497 ] | Angles(size=3)=[ 0.925546 0.8
[ 2.80357 1.20599 1.14979 1.63413 1.50234 1.16022
  1.20599 3.38542 1.441 2.18362 1.51678 1.17623
  1.14979 1.441 2.6612 1.44697 1.1142 0.974349
  1.63413 2.18362 1.44697 4.52485 2.11525 1.42722
  1.50234 1.51678 1.1142 2.11525 3.37899 1.16562
  1.16022 1.17623 0.974349 1.42722 1.16562 2.22099 ]
```

```
Order 4 ORTHO tensor | Param(size=9)=[ 2.06277 -0.00326333 -
0.375452 1.57933 1.64478 10.2695 1.03801 0.701183 0.792497 ] | Angles(size=3)=[ 0.925546 0.8
[ 2.80357 1.20599 1.14979 1.63413 1.50234 1.16022
  1.20599 3.38542 1.441 2.18362 1.51678 1.17623
  1.14979 1.441 2.6612 1.44697 1.1142 0.974349
  1.63413 2.18362 1.44697 4.52485 2.11525 1.42722
  1.50234 1.51678 1.1142 2.11525 3.37899 1.16562
  1.16022 1.17623 0.974349 1.42722 1.16562 2.22099 ]
```

References

Abramowitz, M., Stegun, I.A., 1972. Handbook of Mathematical Functions. National Bureau of Standards - Applied Mathematics Series - 55, Washington D.C.

Barthélémy, J.-F., 2022. Echoes: Extended Calculator of HOmogEnization Schemes. https://doi.org/10.5281/ZENODO.10559657

Barthélémy, J.-F., 2020. Simplified approach to the derivation of the relationship between Hill polarization tensors of transformed problems and applications. International Journal of Engineering Science 154, 103326. https://doi.org/10.1016/j.ijengsci.2020.103326

Barthélémy, J.-F., 2009. Compliance and Hill polarization tensor of a crack in an anisotropic matrix. International Journal of Solids and Structures 46, 4064–4072. https://doi.org/10.1016/j.ijsolstr.2009.08.003

Barthélémy, J.-F., Sevostianov, I., Giraud, A., 2021. Micromechanical modeling of a cracked elliptically orthotropic medium. International Journal of Engineering Science 161, 103454. https://doi.org/10.1016/j.ijengsci.2021.103454

Bornert, M., Bretheau, T., Gilormini, P., 2001. Homogénéisation en mécanique des matériaux. Hermes science.

Brisard, S., 2014. Tensor algebra section in Sébastien Brisard's blog [WWW Document]. URL https://sbrisard.github.io/category/tensor-algebra.html

Dormieux, L., Kondo, D., Ulm, F.-J., 2006. Microporomechanics. John Wiley & Sons, Chichester, West Sussex, England ; Hoboken, NJ. https://doi.org/10.1002/0470032006

Eshelby, J.D., 1957. The determination of the elastic field of an ellipsoidal inclusion, and related problems. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 241, 376–396. https://doi.org/10.1098/rspa.1957.0133

Gavazzi, A.C., Lagoudas, D.C., 1990. On the numerical evaluation of Eshelby's tensor and its application to elastoplastic fibrous composites. Computational Mechanics 7, 13–19. https://doi.org/10.1007/BF00370053

Ghahremani, F., 1977. Numerical evaluation of the stresses and strains in ellipsoidal inclusions in an anisotropic elastic material. Mechanics Research Communications 4, 89–91. https://doi.org/10.1016/0093-6413(77)90018-0

Kachanov, M., Sevostianov, I., 2018. Micromechanics of Materials, with Applications, Solid Mechanics and Its Applications. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-76204-3

Kellogg, O.D., 1929. Potential theory. Berlin : Springer-Verlag.

Masson, R., 2008. New explicit expressions of the Hill polarization tensor for general anisotropic elastic solids. International Journal of Solids and Structures 45, 757–769. https://doi.org/10.1016/j.ijsolstr.2007.08.035

Milton, G.W., 2002. The Theory of Composites, Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511613357

Morin, L., Gilormini, P., Derrien, K., 2020. Generalized Euclidean Distances for Elasticity Tensors. Journal of Elasticity 138, 221–232. https://doi.org/10.1007/s10659-019-09741-z

Mura, T., 1987. Micromechanics of Defects in Solids, Second Edition. Kluwer Academic. https://doi.org/10.1002/zamm.19890690204

Torquato, S., 2002. Random Heterogeneous Materials, Interdisciplinary Applied Mathematics. Springer New York, New York, NY. https://doi.org/10.1007/978-1-4757-6355-3

Walpole, L.J., 1984. Fourth-rank tensors of the thirty-two crystal classes: Multiplication tables. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences 391, 149–179. https://doi.org/10.1098/rspa.1984.0008

Willis, J.R., 1977. Bounds and self-consistent estimates for the overall properties of anisotropic composites. Journal of the Mechanics and Physics of Solids 25, 185–202. https://doi.org/10.1016/0022-5096(77)90022-9

Withers, P.J., 1989. The determination of the elastic field of an ellipsoidal inclusion in a transversely isotropic medium, and its relevance to composite materials. Philosophical Magazine A 59, 759–781. https://doi.org/10.1080/01418618908209819

A Tensor algebra

A.1 Conventions of tensor algebra

This appendix presents some conventions regarding tensor algebra in the usual three-dimensional euclidean space $E = \mathbb{R}^3$. In the sequel, tensor components are associated to an orthonormal frame $(\underline{e}_i)_{i=1,2,3}$ so that introducing the notion of tensor variance is useless here. The following presentation relies on the prior knowledge of the definition of tensors as multilinear operators and the classical isomorphism between the euclidean space and its dual through the scalar product

$$\begin{aligned} \phi : E &\longrightarrow E^* \\ \underline{v} &\longmapsto \underline{v} \cdot \bullet \end{aligned} \tag{A.1}$$

which allows to identify vectors and linear forms.

Consider two tensors $\mathcal{T}$ and $\mathcal{T}'$ of respective orders p and q. The tensor product $\mathcal{T} \otimes \mathcal{T}'$ is the $(p+q)$ order tensor decomposed as

$$\mathcal{T} \otimes \mathcal{T}' = \mathcal{T}_{i_1,\dots,i_p} \mathcal{T}'_{i_{p+1},\dots,i_{p+q}} \underline{e}_{i_1} \otimes \dots \otimes \underline{e}_{i_{p+q}} \tag{A.2}$$

where Einstein convention of immplicit summation over repeated indices is adopted and $\underline{e}_{i_1} \otimes \dots \otimes \underline{e}_{i_{p+q}}$ is the multilinear form such that[1]

$$(\underline{e}_{i_1} \otimes \dots \otimes \underline{e}_{i_{p+q}})(\underline{e}_{j_1}, \dots, \underline{e}_{j_{p+q}}) = \delta_{i_1,j_1} \dots \delta_{i_{p+q},j_{p+q}} \tag{A.3}$$

The notation $\mathcal{T} \overset{s}{\otimes} \mathcal{T}'$ indicates a tensor product followed by a symmetrization over the last index of $\mathcal{T}$ and the first of $\mathcal{T}'$, i.e.

$$\mathcal{T} \overset{s}{\otimes} \mathcal{T}' = \frac{\mathcal{T}_{i_1,\dots,i_p} \mathcal{T}'_{i_{p+1},\dots,i_{p+q}} + \mathcal{T}_{i_1,\dots,i_{p+1}} \mathcal{T}'_{i_p,\dots,i_{p+q}}}{2} \underline{e}_{i_1} \otimes \dots \otimes \underline{e}_{i_{p+q}} \tag{A.4}$$

It follows that

$$\underline{u} \overset{s}{\otimes} \underline{v} = \frac{\underline{u} \otimes \underline{v} + \underline{v} \otimes \underline{u}}{2} \tag{A.5}$$

and an example of generalization involving a second-order tensor $\mathbf{a}$ and vectors $\underline{u}$ and $\underline{v}$

$$\underline{u} \overset{s}{\otimes} \mathbf{a} \overset{s}{\otimes} \underline{v} = \frac{u_i\, a_{jk}\, v_l + u_i\, a_{jl}\, v_k + u_j\, a_{ik}\, v_l + u_j\, a_{il}\, v_k}{4} \underline{e}_i \otimes \underline{e}_j \otimes \underline{e}_k \otimes \underline{e}_l \tag{A.6}$$

The simple dot product or contracted product between $\mathcal{T}$ and $\mathcal{T}'$ involves by convention a contraction between the last index of $\mathcal{T}$ and the first of $\mathcal{T}'$, which leads to the $(p+q-2)$ order tensor

$$\mathcal{T} \cdot \mathcal{T}' = \mathcal{T}_{i_1,\dots,i_{p-1},k} \mathcal{T}'_{k,i_p,\dots,i_{p+q-2}} \underline{e}_{i_1} \otimes \dots \otimes \underline{e}_{i_{p+q-2}} \tag{A.7}$$

[1] $\delta_{ij} = 1$ if $i = j$ and 0 if $i \neq j$ (Kronecker symbol)

As regards the double dot product, the classical convention consists in consuming the indices going up from the extremities, which means that a first contraction acts as in the simple dot product then a second contraction is performed between the penultimate index of $\mathcal{T}$ and the second one of $\mathcal{T}'$. However and alternate convention adopted here is proposed in (Brisard, 2014)[2], which somehow consists in considering that the double contraction operates over the two last indices of $\mathcal{T}$ as a pair and the two first indices of $\mathcal{T}'$ as the corresponding pair. In other words, this operation is such that if $\mathbf{a}$ and $\mathbf{b}$ are two second-order tensors and $\mathbb{T}$ is a fourth-order tensor

$$\mathbf{a} : \mathbf{b} = a_{ij} b_{ij} \quad \text{and} \quad \mathbb{T} : \mathbf{a} = T_{ijkl}\, a_{kl}\, \underline{e}_i \otimes \underline{e}_j \tag{A.8}$$

and the transpose tensor ${}^t\mathbb{T}$ is consistently defined by

$${}^t\mathbb{T} : \mathbf{a} = \mathbf{a} : \mathbb{T} \quad \Leftrightarrow \quad \left({}^t\mathbb{T}\right)_{ijkl} = \left(\mathbb{T}\right)_{klij} \tag{A.9}$$

The quadruple dot product is introduced as a scalar product between fourth-order tensors as

$$\mathbb{T} :: \mathbb{T}' = T_{ijkl} T'_{ijkl} \tag{A.10}$$

Another useful operator introduced in (Brisard, 2014)[3] is the modified tensor product denoted by $\boxtimes$. The fourth-order tensor $\mathbf{a} \boxtimes \mathbf{b}$ (where $\mathbf{a}$ and $\mathbf{b}$ are two second-order tensors) is defined by its operation over any second-order tensor $\mathbf{p}$ and by its components

$$\begin{aligned} (\mathbf{a} \boxtimes \mathbf{b}) : \mathbf{p} &= \mathbf{a} \cdot \mathbf{p} \cdot {}^t\mathbf{b} = a_{ik}\, p_{kl}\, b_{jl}\, \underline{e}_i \otimes \underline{e}_j \\ (\mathbf{a} \boxtimes \mathbf{b})_{ijkl} &= a_{ik}\, b_{jl} \end{aligned} \tag{A.11}$$

A symmetrized version of $\boxtimes$ denoted by $\overset{s}{\boxtimes}$ can also be introduced. It operates as

$$\begin{aligned} (\mathbf{a} \overset{s}{\boxtimes} \mathbf{b}) : \mathbf{p} &= (\mathbf{a} \boxtimes \mathbf{b}) : \left(\frac{\mathbf{p} + {}^t\mathbf{p}}{2}\right) = \mathbf{a} \cdot \left(\frac{\mathbf{p} + {}^t\mathbf{p}}{2}\right) \cdot {}^t\mathbf{b} \\ (\mathbf{a} \overset{s}{\boxtimes} \mathbf{b})_{ijkl} &= \frac{a_{ik}\, b_{jl} + a_{il}\, b_{jk}}{2} \end{aligned} \tag{A.12}$$

It follows from these definitions that the fourth-order identity, as an operator over second-order tensors, writes $\mathbb{1} = \mathbf{1} \boxtimes \mathbf{1}$ where $\mathbf{1}$ is the second-order identity. The fourth-order operator allowing to extract the symmetric part of a second-order tensor writes $\mathbb{I} = \mathbf{1} \overset{s}{\boxtimes} \mathbf{1}$. The latter tensor, which obviously complies with the conditions of minor symmetries, is classically used to play the role of fourth-order identity operating over symmetric second-order tensors.

[2]see https://sbrisard.github.io/posts/20140219-on_the_double_dot_product.html
[3]see https://sbrisard.github.io/posts/20140226-decomposition_of_transverse_isotropic_fourth-rank_tensors.html

Some remarkable relationships result from the previous definitions

$$
\begin{array}{ll}
(\mathbf{a} \boxtimes \mathbf{b}) : (\underline{u} \otimes \underline{v}) = (\mathbf{a} \cdot \underline{u}) \otimes (\mathbf{b} \cdot \underline{v}) & (a) \\
(\mathbf{a} \boxtimes \mathbf{b}) : (\mathbf{c} \boxtimes \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c}) \boxtimes (\mathbf{b} \cdot \mathbf{d}) & (b) \\
(\mathbf{a} \overset{s}{\boxtimes} \mathbf{b}) : (\mathbf{c} \overset{s}{\boxtimes} \mathbf{d}) = \dfrac{(\mathbf{a} \cdot \mathbf{c}) \overset{s}{\boxtimes} (\mathbf{b} \cdot \mathbf{d}) + (\mathbf{a} \cdot \mathbf{d}) \overset{s}{\boxtimes} (\mathbf{b} \cdot \mathbf{c})}{2} & (c) \\
(\mathbf{a} \overset{s}{\boxtimes} \mathbf{a}) : (\mathbf{b} \overset{s}{\boxtimes} \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b}) \overset{s}{\boxtimes} (\mathbf{a} \cdot \mathbf{b}) & (d) \\
{}^t(\mathbf{a} \boxtimes \mathbf{b}) = {}^t\mathbf{a} \boxtimes {}^t\mathbf{b} & (e) \\
{}^t(\mathbf{a} \overset{s}{\boxtimes} \mathbf{a}) = {}^t\mathbf{a} \overset{s}{\boxtimes} {}^t\mathbf{a} \;\text{ but }\; {}^t(\mathbf{a} \overset{s}{\boxtimes} \mathbf{b}) \neq {}^t\mathbf{a} \overset{s}{\boxtimes} {}^t\mathbf{b} & (f) \\
(\mathbf{a} \boxtimes \mathbf{b})^{-1} = \mathbf{a}^{-1} \boxtimes \mathbf{b}^{-1} & (g) \\
(\mathbf{a} \overset{s}{\boxtimes} \mathbf{a})^{-1} : \mathbf{p} = (\mathbf{a}^{-1} \overset{s}{\boxtimes} \mathbf{a}^{-1}) : \mathbf{p} \;\text{ if }\; {}^t\mathbf{p} = \mathbf{p} \;\text{ but }\; (\mathbf{a} \overset{s}{\boxtimes} \mathbf{b})^{-1} \neq \mathbf{a}^{-1} \overset{s}{\boxtimes} \mathbf{b}^{-1} & (h)
\end{array}
\tag{A.13}
$$

A.2 Kelvin-Mandel notation

The Kelvin-Mandel notation allows to write the matrix of a symmetric second-order tensor in a given orthonormal frame $(\underline{e}_i)_{i=1,2,3}$ under the form of a vector of $\mathbb{R}^6$

$$
\mathrm{Mat}(\boldsymbol{\varepsilon}, (\underline{e}_i)) = \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{31} \\ \varepsilon_{12} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{23} & \varepsilon_{33} \end{pmatrix} \mapsto \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \sqrt{2}\,\varepsilon_{23} \\ \sqrt{2}\,\varepsilon_{31} \\ \sqrt{2}\,\varepsilon_{12} \end{pmatrix} \tag{A.14}
$$

The vector of $\mathbb{R}^6$ in (A.14) corresponds to the components of the second-order tensor $\boldsymbol{\varepsilon}$ in the basis ordered as

$$
\mathcal{B} = \left(\underline{e}_1 \otimes \underline{e}_1, \underline{e}_2 \otimes \underline{e}_2, \underline{e}_3 \otimes \underline{e}_3, \sqrt{2}\,\underline{e}_2 \otimes \underline{e}_3, \sqrt{2}\,\underline{e}_3 \otimes \underline{e}_1, \sqrt{2}\,\underline{e}_1 \otimes \underline{e}_2\right) \tag{A.15}
$$

The tensors of the basis (A.15) form an orthonormal frame spanning the space of symmetric second-order tensors equipped with the double contraction ":" as scalar product. It follows that the double contraction between symmetric second-order tensors is no other than the classical scalar product of the corresponding vectors of $\mathbb{R}^6$ written according to the convention (A.14).

> **i Note**
>
> Note the effect of $\sqrt{2}$ on the off-diagonal components. Unlike Voigt convention, the choice of Kelvin-Mandel convention throughout all the library allows to address strain and stress tensors under the same synthetic representation.

A fourth-order tensor with minor symetries ($C_{jikl} = C_{ijlk} = C_{ijkl}$), which can be seen as a linear operator acting over symmetric second-order tensors by double contraction, writes in the same convention under the form of a 6×6 square matrix (the solid lines separate blocks affected by different factors whereas the colored components highlight a central block playing a major role in the sequel)

$$\mathrm{Mat}(\mathbb{C},\mathcal{B}) = \left(\begin{array}{ccc|ccc} C_{1111} & C_{1122} & C_{1133} & \sqrt{2}\,C_{1123} & \sqrt{2}\,C_{1131} & \sqrt{2}\,C_{1112} \\ C_{2211} & C_{2222} & C_{2233} & \sqrt{2}\,C_{2223} & \sqrt{2}\,C_{2231} & \sqrt{2}\,C_{2212} \\ C_{3311} & C_{3322} & C_{3333} & \sqrt{2}\,C_{3323} & \sqrt{2}\,C_{3331} & \sqrt{2}\,C_{3312} \\ \hline \sqrt{2}\,C_{2311} & \sqrt{2}\,C_{2322} & \sqrt{2}\,C_{2333} & 2\,C_{2323} & 2\,C_{2331} & 2\,C_{2312} \\ \sqrt{2}\,C_{3111} & \sqrt{2}\,C_{3122} & \sqrt{2}\,C_{3133} & 2\,C_{3123} & 2\,C_{3131} & 2\,C_{3112} \\ \sqrt{2}\,C_{1211} & \sqrt{2}\,C_{1222} & \sqrt{2}\,C_{1233} & 2\,C_{1223} & 2\,C_{1231} & 2\,C_{1212} \end{array}\right) \tag{A.16}$$

The result of $\mathbb{C} : \varepsilon$ writes as a classical matrix-vector product of (A.16) by (A.14).

However another way of ordering the tensors of (A.15) which proves useful for the calculation of crack compliance is based on a gathering of one set of three in-plane and another one of three out-of-plane tensors (the latter involving $\underline{n} = \underline{e}_3$ assumed to be the normal of the crack and the former not)

$$\mathcal{B}^* = \Big(\underbrace{\underline{e}_1 \otimes \underline{e}_1, \underline{e}_2 \otimes \underline{e}_2, \sqrt{2}\,\underline{e}_1 \overset{s}{\otimes} \underline{e}_2,}_{\text{in-plane}} \quad \underbrace{\underline{e}_3 \otimes \underline{e}_3, \sqrt{2}\,\underline{e}_2 \overset{s}{\otimes} \underline{e}_3, \sqrt{2}\,\underline{e}_3 \overset{s}{\otimes} \underline{e}_1}_{\text{out-of-plane}} \Big) \tag{A.17}$$

such that the matrix of $\mathbb{C}$ in $\mathcal{B}^*$ is now obtained by permutations of lines and columns of (A.16) to give

$$\mathrm{Mat}(\mathbb{C},\mathcal{B}^*) = \left(\begin{array}{ccc|ccc} C_{1111} & C_{1122} & \sqrt{2}\,C_{1112} & C_{1133} & \sqrt{2}\,C_{1123} & \sqrt{2}\,C_{1131} \\ C_{2211} & C_{2222} & \sqrt{2}\,C_{2212} & C_{2233} & \sqrt{2}\,C_{2223} & \sqrt{2}\,C_{2231} \\ \sqrt{2}\,C_{1211} & \sqrt{2}\,C_{1222} & 2\,C_{1212} & \sqrt{2}\,C_{1233} & 2\,C_{1223} & 2\,C_{1231} \\ \hline C_{3311} & C_{3322} & \sqrt{2}\,C_{3312} & C_{3333} & \sqrt{2}\,C_{3323} & \sqrt{2}\,C_{3331} \\ \sqrt{2}\,C_{2311} & \sqrt{2}\,C_{2322} & 2\,C_{2312} & \sqrt{2}\,C_{2333} & 2\,C_{2323} & 2\,C_{2331} \\ \sqrt{2}\,C_{3111} & \sqrt{2}\,C_{3122} & 2\,C_{3112} & \sqrt{2}\,C_{3133} & 2\,C_{3123} & 2\,C_{3131} \end{array}\right) \tag{A.18}$$

One may notice that the bottom right 3×3 block of (A.18) exactly corresponds to the colored block in (A.16).

A.3 Rotation of tensors

Recalling that the set of second-order tensors is isomorphic to the set of endomorphism in an euclidean space, it is natural to define a rotation tensor as an element of the special orthogonal group, i.e. tensors $\mathbf{R}$ such that ${}^t\mathbf{R} \cdot \mathbf{R} = \mathbf{1}$ and $\det \mathbf{R} = 1$. In $\mathbb{R}^3$ equipped with an orthonormal frame $(\underline{e}_i)_{i=1,2,3}$, these tensors can be defined by three Euler angles $(\theta, \phi, \psi) \in [0, \pi] \times [0, 2\pi] \times [0, 2\pi]$ (see Fig. A.1) such that[4]

$$\mathrm{Mat}(\mathbf{R}, (\underline{e}_i)) = \begin{pmatrix} c_\theta c_\psi c_\phi - s_\psi s_\phi & -c_\theta c_\phi s_\psi - c_\psi s_\phi & c_\phi s_\theta \\ c_\theta c_\psi s_\phi + c_\phi s_\psi & -c_\theta s_\psi s_\phi + c_\psi c_\phi & s_\theta s_\phi \\ -c_\psi s_\theta & s_\theta s_\psi & c_\theta \end{pmatrix} \tag{A.19}$$

A rotated frame $(\underline{e}'_i)_{i=1,2,3}$ is obtained by application of the rotation $\mathbf{R}$ on the vectors $\underline{e}_i$ (see Fig. A.1)

$$\forall i \in \{1,2,3\} \quad \underline{e}'_i = \mathbf{R} \cdot \underline{e}_i \tag{A.20}$$

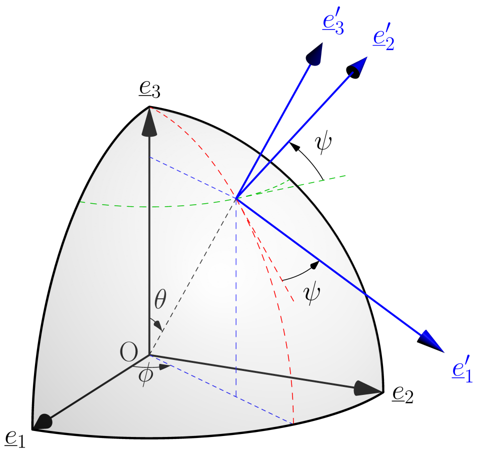


Figure A.1: Eurler angles

The rotation of Eurler angles (θ, ϕ, ψ) applies on a p order tensor $\mathcal{T}$ as

$$\mathcal{T} = \mathcal{T}_{i_1,\ldots,i_p}\, \underline{e}_{i_1} \otimes \ldots \otimes \underline{e}_{i_p} \overset{\mathbf{R}}{\longmapsto} \mathbf{R}(\mathcal{T}) = \mathcal{T}_{i_1,\ldots,i_p}\, (\mathbf{R}\cdot\underline{e}_{i_1}) \otimes \ldots \otimes (\mathbf{R}\cdot\underline{e}_{i_p}) \tag{A.21}$$

The application of (A.21) to a second-order tensor $\mathbf{a}$ therefore gives

$$\mathbf{R}(\mathbf{a}) = \mathbf{R}\cdot\mathbf{a}\cdot{}^t\mathbf{R} \tag{A.22}$$

and to a fourth-order tensor $\mathbb{T}$

$$\mathbf{R}(\mathbb{T}) = (\mathbf{R} \overset{s}{\boxtimes} \mathbf{R}) : \mathbb{T} : {}^t(\mathbf{R} \overset{s}{\boxtimes} \mathbf{R}) \tag{A.23}$$

where the fourth-order rotation tensor $\mathbb{R} = \mathbf{R} \overset{s}{\boxtimes} \mathbf{R}$ can be expressed in Kelvin-Mandel notation (A.16) by means of the components of $\mathbf{R}$ defined in (A.19)

$$\mathrm{Mat}(\mathbb{R}, \mathcal{B}) = \begin{pmatrix} R_{11}^2 & R_{12}^2 & R_{13}^2 & \sqrt{2}R_{12}R_{13} & \sqrt{2}R_{11}R_{13} & \sqrt{2}R_{11}R_{12} \\ R_{21}^2 & R_{22}^2 & R_{23}^2 & \sqrt{2}R_{22}R_{23} & \sqrt{2}R_{21}R_{23} & \sqrt{2}R_{21}R_{22} \\ R_{31}^2 & R_{32}^2 & R_{33}^2 & \sqrt{2}R_{32}R_{33} & \sqrt{2}R_{31}R_{33} & \sqrt{2}R_{31}R_{32} \\ \sqrt{2}R_{21}R_{31} & \sqrt{2}R_{22}R_{32} & \sqrt{2}R_{23}R_{33} & R_{22}R_{33}+R_{23}R_{32} & R_{21}R_{33}+R_{23}R_{31} & R_{21}R_{32}+R_{22}R_{31} \\ \sqrt{2}R_{11}R_{31} & \sqrt{2}R_{12}R_{32} & \sqrt{2}R_{13}R_{33} & R_{12}R_{33}+R_{13}R_{32} & R_{11}R_{33}+R_{13}R_{31} & R_{11}R_{32}+R_{12}R_{31} \\ \sqrt{2}R_{11}R_{21} & \sqrt{2}R_{12}R_{22} & \sqrt{2}R_{13}R_{23} & R_{12}R_{23}+R_{13}R_{22} & R_{11}R_{23}+R_{13}R_{21} & R_{11}R_{22}+R_{12}R_{21} \end{pmatrix} \tag{A.24}$$

It results that (A.23) can be seen as a classical rotation operation involving matrix multiplications in $\mathbb{R}^6$.

A.4 Fourth-order isotropic tensors

This paragraph concerns fourth-order tensors operating over symmetrical second-order tensors, which allows to impose that they satisfy the minor symmetries ($C_{jikl} = C_{ijlk} = C_{ijkl}$). The identity operator is given by

$$\mathbb{I} = \mathbf{1} \overset{s}{\boxtimes} \mathbf{1} = \frac{\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}}{2}\, \underline{e}_i \otimes \underline{e}_j \otimes \underline{e}_k \otimes \underline{e}_l \tag{A.25}$$

[4]with the simplifying writing convention $c_\theta = \cos\theta$, $s_\theta = \sin\theta$, $c_\phi = \cos\phi$, $s_\phi = \sin\phi$, $c_\psi = \cos\psi$, $s_\psi = \sin\psi$

of identity matrix in Kelvin-Mandel convention

$$\mathrm{Mat}(\mathbb{I},\mathcal{B}) = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \tag{A.26}$$

It is classically proven that any minor-symmetrical fourth-order tensor invariant by rotation (A.23) write as a linear combination on the two projectors $\mathbb{J}$ and $\mathbb{K}$ which respectively extract the spherical and deviatoric part of any symmetrical second-order tensor

$$\mathbb{J} : \mathbf{a} = \frac{1}{3}\operatorname{tr}\mathbf{a}\,\mathbf{1} \quad \text{and} \quad \mathbb{K} : \mathbf{a} = \mathbf{a} - \frac{1}{3}\operatorname{tr}\mathbf{a}\,\mathbf{1} \tag{A.27}$$

In other words, $\mathbb{J}$ and $\mathbb{K}$ are defined by

$$\mathbb{J} = \frac{1}{3}\mathbf{1}\otimes\mathbf{1} \quad \text{and} \quad \mathbb{K} = \mathbb{I} - \mathbb{J} = \mathbf{1} \overset{s}{\boxtimes} \mathbf{1} - \frac{1}{3}\mathbf{1}\otimes\mathbf{1} \tag{A.28}$$

their components by

$$J_{ijkl} = \frac{1}{3}\delta_{ij}\delta_{ik} \quad \text{and} \quad K_{ijkl} = \frac{\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}}{2} - \frac{1}{3}\delta_{ij}\delta_{ik} \tag{A.29}$$

and their matrices in Kelvin-Mandel notation relatively to any orthonormal frame by

$$\mathrm{Mat}(\mathbb{J},\mathcal{B}) = \left(\begin{array}{ccc|ccc} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{A.30}$$

$$\mathrm{Mat}(\mathbb{K},\mathcal{B}) = \left(\begin{array}{ccc|ccc} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} & 0 & 0 & 0 \\ \frac{-1}{3} & \frac{2}{3} & \frac{-1}{3} & 0 & 0 & 0 \\ \frac{-1}{3} & \frac{-1}{3} & \frac{2}{3} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \tag{A.31}$$

The following relationships are easily obtained

$$\mathbb{J}:\mathbb{J}=\mathbb{J} \quad ; \quad \mathbb{K}:\mathbb{K}=\mathbb{K} \quad ; \quad \mathbb{J}:\mathbb{K}=\mathbb{0} \quad ; \quad \mathbb{J}::\mathbb{J}=1 \quad ; \quad \mathbb{K}::\mathbb{K}=5 \tag{A.32}$$

The isotropisation of any fourth-order tensor $\mathbb{T}$ is defined by (Bornert et al., 2001)

$$\mathrm{ISO}(\mathbb{T}) = (\mathbb{T}::\mathbb{J})\,\mathbb{J} + \left(\frac{\mathbb{T}::\mathbb{K}}{5}\right)\mathbb{K} \tag{A.33}$$

It is easy to show that (A.33) is no other than the closest isotropic tensor to $\mathbb{T}$ if the distance is chosen as the euclidean one i.e. associated to the scalar product (A.10). Note however that this isotropisation relying on euclidean distance to the set of isotropic tensors does not lead to the same result if $\mathbb{T}'$ is considered instead of $\mathbb{T}$. Other distance definitions such as log-Euclidean distance fullfilling invariance by inversion are analyzed in (Morin et al., 2020).

A.5 Fourth-order transversely isotropic tensors and Walpole basis

The Walpole basis ((Walpole, 1984), (Brisard, 2014)[5]) allowing to write any fourth-order transversely isotropic relatively to a an axis oriented by the unit vector $\underline{n}$ is composed of the six following tensors built from $\mathbf{1}_n = \underline{n} \otimes \underline{n}$ and $\mathbf{1}_T = \mathbf{1} - \mathbf{1}_n$

$$\begin{array}{ll} \mathbb{W}_1 = \mathbf{1}_n \otimes \mathbf{1}_n \quad ; \quad \mathbb{W}_2 = \dfrac{\mathbf{1}_T \otimes \mathbf{1}_T}{2} & ; \mathbb{W}_3 = \dfrac{\mathbf{1}_n \otimes \mathbf{1}_T}{\sqrt{2}} \quad ; \quad \mathbb{W}_4 = \dfrac{\mathbf{1}_T \otimes \mathbf{1}_n}{\sqrt{2}} \quad (a) \\ \mathbb{W}_5 = \mathbf{1}_T \overset{s}{\boxtimes} \mathbf{1}_T - \dfrac{\mathbf{1}_T \otimes \mathbf{1}_T}{2} & ; \mathbb{W}_6 = \mathbf{1}_T \overset{s}{\boxtimes} \mathbf{1}_n + \mathbf{1}_n \overset{s}{\boxtimes} \mathbf{1}_T \quad (b) \end{array} \tag{A.34}$$

Any transversely isotropic fourth-order tensor can be decomposed as

$$\mathbb{L} = \ell_1 \, \mathbb{E}_1 + \ell_2 \, \mathbb{E}_2 + \ell_3 \, \mathbb{E}_3 + \ell_4 \, \mathbb{E}_4 + \ell_5 \, \mathbb{E}_5 + \ell_6 \, \mathbb{E}_6 \tag{A.35}$$

The six parameters can be conveniently gathered in a triplet composed of a 2×2 matrix containing the four first parameters ℓ_i $(1 \leq i \leq 4)$ and the two last parameters ℓ_5 and ℓ_6

$$\mathbb{L} \equiv (L, \ell_5, \ell_6)\,, \quad L = \begin{pmatrix} \ell_1 & \ell_3 \\ \ell_4 & \ell_2 \end{pmatrix} \tag{A.36}$$

Such a synthetic notation allows simple calculations of products and inverses which consist in classical matrix or scalar products and inverses

$$\begin{array}{ll} \mathbb{L} : \mathbb{M} \equiv (LM, \ell_5 m_5, \ell_6 m_6) & (a) \\ \mathbb{L}^{-1} \equiv \left(L^{-1}, \dfrac{1}{\ell_5}, \dfrac{1}{\ell_6} \right) & (b) \end{array} \tag{A.37}$$

A symmetrized version of the Walpole also exists; it is composed of the following tensors

$$\begin{array}{ll} \mathbb{W}^s_1 = \mathbf{1}_n \otimes \mathbf{1}_n \quad ; \quad \mathbb{W}^s_2 = \dfrac{\mathbf{1}_T \otimes \mathbf{1}_T}{2} & ; \mathbb{W}^s_3 = \dfrac{\mathbf{1}_n \otimes \mathbf{1}_T}{\sqrt{2}} + \dfrac{\mathbf{1}_T \otimes \mathbf{1}_n}{\sqrt{2}} \quad (a) \\ \mathbb{W}^s_4 = \mathbf{1}_T \overset{s}{\boxtimes} \mathbf{1}_T - \dfrac{\mathbf{1}_T \otimes \mathbf{1}_T}{2} & ; \mathbb{W}^s_5 = \mathbf{1}_T \overset{s}{\boxtimes} \mathbf{1}_n + \mathbf{1}_n \overset{s}{\boxtimes} \mathbf{1}_T \quad (b) \end{array} \tag{A.38}$$

[5]see https://sbrisard.github.io/posts/20140226-decomposition_of_transverse_isotropic_fourth-rank_tensors.html

B Hill polarization tensor in elasticity

This section recalls some results about the calculation of the Hill polarization tensors related to a matrix of stiffness $\mathbb{C}$ and an ellipsoid $\mathcal{E}_{\mathbf{A}}$ of equation

$$\underline{x} \in \mathcal{E}_{\mathbf{A}} \quad \Leftrightarrow \quad \underline{x} \cdot ({}^t\mathbf{A} \cdot \mathbf{A})^{-1} \cdot \underline{x} \leq 1$$

where $\mathbf{A}$ is an invertible second-order tensor so that ${}^t\mathbf{A} \cdot \mathbf{A}$ is a positive definite symmetric tensor associated to 3 radii (eigenvalues $a \geq b \geq c$ possibly written $\rho_1 \geq \rho_2 \geq \rho_3$ for convenience) and 3 angles (orientation of the frame of eigenvectors $\underline{e}_1, \underline{e}_2, \underline{e}_3$)

$$ {}^t\mathbf{A} \cdot \mathbf{A} = a^2 \underline{e}_1 \otimes \underline{e}_1 + b^2 \underline{e}_2 \otimes \underline{e}_2 + c^2 \underline{e}_3 \otimes \underline{e}_3 = \sum_{i=1}^{3} \rho_i \underline{e}_i \otimes \underline{e}_i \tag{B.1}$$

B.1 General expression

A general expression of the elastic polarization tensor is derived in (Willis, 1977) (see also (Mura, 1987))

$$\begin{aligned} \mathbb{P}(\mathbf{A}, \mathbb{C}) &= \frac{1}{4\pi} \int_{\|\underline{\zeta}\|=1} (\mathbf{A}^{-1} \cdot \underline{\zeta}) \overset{s}{\otimes} \Big((\mathbf{A}^{-1} \cdot \underline{\zeta}) \cdot \mathbb{C} \cdot (\mathbf{A}^{-1} \cdot \underline{\zeta}) \Big)^{-1} \overset{s}{\otimes} (\mathbf{A}^{-1} \cdot \underline{\zeta}) \, \mathrm{d}S_{\zeta} \\ &= \frac{\det \mathbf{A}}{4\pi} \int_{\|\underline{\xi}\|=1} \frac{\underline{\xi} \overset{s}{\otimes} (\underline{\xi} \cdot \mathbb{C} \cdot \underline{\xi})^{-1} \overset{s}{\otimes} \underline{\xi}}{\|\mathbf{A} \cdot \underline{\xi}\|^3} \, \mathrm{d}S_{\xi} \end{aligned} \tag{B.2}$$

When $\mathbb{C}$ is arbitrarily anisotropic, it is necessary to resort to numerical cubature to estimate $\mathbb{P}$ as proposed in (Ghahremani, 1977), (Gavazzi and Lagoudas, 1990) or (Masson, 2008). However in some cases of anisotropy, analytical solutions are available ((Withers, 1989), (Barthélémy, 2020)). The case of isotropic matrix is particularly developed in the next section.

B.2 Isotropic matrix

In this section, the matrix is assumed isotropic so that its stiffness tensor writes by means of a bulk k and shear μ or Lamé λ and μ moduli or even Young modulus E and Poisson ratio ν with $k = \frac{E}{3(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$.

$$\begin{aligned} \mathbb{C} =& 3k\mathbb{J} + 2\mu\mathbb{K} = 3\lambda\mathbb{I} + 2\mu\mathbb{K} \\ &\text{with} \quad J_{ijkl} = \frac{\delta_{ij}\delta_{kl}}{3}, I_{ijkl} = \frac{\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}}{2} \text{ and } \mathbb{K} = \mathbb{I} - \mathbb{J} \end{aligned} \tag{B.3}$$

Introducing (B.3) in (B.2) leads to after some algebra

$$\mathbb{P} = \frac{1}{\lambda + 2\,\mu}\mathbb{U} + \frac{1}{\mu}(\mathbb{V} - \mathbb{U})$$

where the tensors $\mathbb{U}$ and $\mathbb{V}$, depending only on the ellipsoidal tensor $\mathbf{A}$ of (B.1), are given by (see (Barthélémy, 2020))

$$\begin{aligned}\mathbb{U} &= \frac{\det \mathbf{A}}{4\pi} \int_{\|\underline{\xi}\|=1} \frac{\underline{\xi} \otimes \underline{\xi} \otimes \underline{\xi} \otimes \underline{\xi}}{\|\mathbf{A} \cdot \underline{\xi}\|^3}\,\mathrm{d}S_\xi \\ &= \frac{1}{4\pi} \int_{\|\underline{\zeta}\|=1} \frac{(\mathbf{A}^{-1} \cdot \underline{\zeta}) \otimes (\mathbf{A}^{-1} \cdot \underline{\zeta}) \otimes (\mathbf{A}^{-1} \cdot \underline{\zeta}) \otimes (\mathbf{A}^{-1} \cdot \underline{\zeta})}{\|\mathbf{A}^{-1} \cdot \underline{\zeta}\|^4}\,\mathrm{d}S_\zeta\end{aligned}$$

and

$$\begin{aligned}\mathbb{V} &= \frac{\det \mathbf{A}}{4\pi} \int_{\|\underline{\xi}\|=1} \frac{\underline{\xi} \overset{s}{\otimes} \mathbf{1} \overset{s}{\otimes} \underline{\xi}}{\|\mathbf{A} \cdot \underline{\xi}\|^3}\,\mathrm{d}S_\xi \\ &= \frac{1}{4\pi} \int_{\|\underline{\zeta}\|=1} \frac{(\mathbf{A}^{-1} \cdot \underline{\zeta}) \overset{s}{\otimes} \mathbf{1} \overset{s}{\otimes} (\mathbf{A}^{-1} \cdot \underline{\zeta})}{\|\mathbf{A}^{-1} \cdot \underline{\zeta}\|^2}\,\mathrm{d}S_\zeta\end{aligned}$$

For an arbitrary ellipsoid defined by (B.1), the components of $\mathbb{U}$ and $\mathbb{V}$ write

$$\begin{aligned}U_{iiii} &= \frac{3(I_i - \rho_i^2 I_{ii})}{2} \quad \forall\, i \in \{1,2,3\} \\ U_{iijj} = U_{ijij} = U_{ijji} &= \frac{I_j - \rho_i^2 I_{ij}}{2} = \frac{I_i - \rho_j^2 I_{ij}}{2} \quad \forall\, i \neq j \in \{1,2,3\}\end{aligned}$$

and

$$\begin{aligned}V_{iiii} &= I_i \quad \forall\, i \in \{1,2,3\} \\ V_{ijij} = V_{ijji} &= \frac{I_i + I_j}{4} \quad \forall\, i \neq j \in \{1,2,3\}\end{aligned}$$

where the coefficients I_i and I_{ij} are given by (note that I_i and I_{ij} are adapted from those provided in (Kellogg, 1929) and (Eshelby, 1957): they differ by a factor of $4\pi/3$ for I_{ij} with $i \neq j$ and by 4π for the others)

- if $a > b > c$

$$\begin{aligned}I_1 &= \frac{a\,b\,c}{(a^2 - b^2)\sqrt{a^2 - c^2}}\,(F - E) \\ I_3 &= \frac{a\,b\,c}{(b^2 - c^2)\sqrt{a^2 - c^2}}\left(\frac{b\sqrt{a^2 - c^2}}{a\,c} - E\right) \\ I_2 &= 1 - I_1 - I_3 \\ I_{ij} &= \frac{I_j - I_i}{\rho_i^2 - \rho_j^2} \quad \forall\, i \neq j \in \{1,2,3\} \\ I_{ii} &= \frac{1}{3}\left(\frac{1}{\rho_i^2} - \sum_{j \neq i} I_{ij}\right) \quad \forall\, i \in \{1,2,3\}\end{aligned}$$

where $F = F(\theta, \kappa)$ and $E = E(\theta, \kappa)$ are respectively the elliptic integrals of the first and second kinds (see (Abramowitz and Stegun, 1972)) of amplitude and parameter

$$\theta = \arcsin\sqrt{1 - \frac{c^2}{a^2}} \quad ; \quad \kappa = \sqrt{\frac{a^2 - b^2}{a^2 - c^2}}$$

- if $a > b = c$ (prolate spheroid)

$$\begin{aligned}
I_2 = I_3 &= a\,\frac{a\sqrt{a^2 - c^2} - c^2\ \mathrm{arcosh}\,(a/c)}{2\,(a^2 - c^2)^{3/2}} \\
I_1 &= 1 - 2\,I_3 \\
I_{1i} = I_{i1} &= \frac{I_i - I_1}{a^2 - \rho_i^2} \quad \forall\, i \in \{2, 3\} \\
I_{ij} &= \frac{1}{4}\left(\frac{1}{c^2} - I_{31}\right) \quad \forall\, i, j \in \{2, 3\} \\
I_{11} &= \frac{1}{3}\left(\frac{1}{a^2} - 2\,I_{31}\right)
\end{aligned}$$

- if $a = b > c$ (oblate spheroid)

$$\begin{aligned}
I_1 = I_2 &= c\,\frac{a^2\ \arccos\,(c/a) - c\sqrt{a^2 - c^2}}{2\,(a^2 - c^2)^{3/2}} \\
I_3 &= 1 - 2\,I_1 \\
I_{3i} = I_{i3} &= \frac{I_3 - I_i}{\rho_i^2 - c^2} \quad \forall\, i \in \{1, 2\} \\
I_{ij} &= \frac{1}{4}\left(\frac{1}{a^2} - I_{31}\right) \quad \forall\, i, j \in \{1, 2\} \\
I_{33} &= \frac{1}{3}\left(\frac{1}{c^2} - 2\,I_{31}\right)
\end{aligned}$$

- if $a = b = c$ (sphere)

$$\begin{aligned}
I_1 = I_2 = I_3 &= \frac{1}{3} \\
I_{ij} &= \frac{1}{5\,a^2} \quad \forall\, i, j \in \{1, 2, 3\}
\end{aligned}$$

In this last case of spherical inclusion ($\mathbf{A} = \mathbf{1}$), $\mathbb{U}$ and $\mathbb{V}$ are simply decomposed as

$$\mathbb{U} = \frac{1}{3}\mathbb{J} + \frac{2}{15}\mathbb{K} \quad \text{and} \quad \mathbb{V} = \frac{1}{3}\mathbb{I}$$

B.3 Case of cracks

The case of cracks corresponds to ellipsoids for which the smallest radius is very small compared to the two others, in other words the characteristic tensor $\mathbf{A}$ (B.1) can be written here

$$\mathbf{A} = \underline{\ell} \otimes \underline{\ell} + \eta\,\underline{m} \otimes \underline{m} + \omega\,\underline{n} \otimes \underline{n} \quad \text{with} \quad \eta = \frac{b}{a} \quad \text{and} \quad \omega = \frac{c}{a}$$

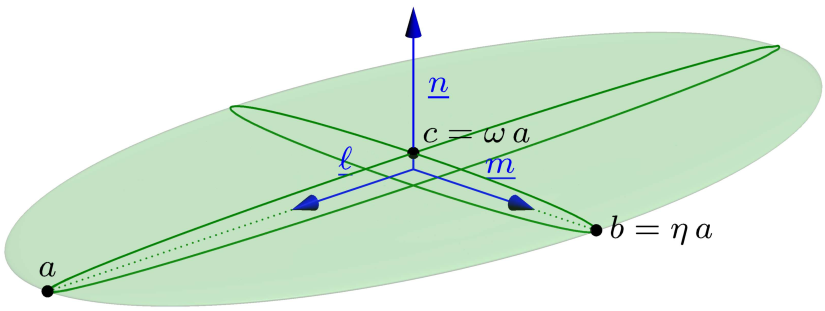


Figure B.1: Ellipsoidal crack

In the case of cracks, it is useful to introduce the second Hill polarization tensor defined as

$$\mathbb{Q} = \mathbb{C} - \mathbb{C} : \mathbb{P} : \mathbb{C}$$

and in particular $\lim_{\omega \to 0} \omega\, \mathbb{Q}^{-1}$ in which it is recalled that $\mathbb{P}$ and thus $\mathbb{Q}$ depend on ω such that the components $\left(\mathbb{Q}^{-1}\right)_{nijk}$ (with n corresponding to the crack normal) behave as $1/\omega$ when ω tends towards 0. The analytical expressions of this limit are fully detailed in (Barthélémy et al., 2021) which recalls in particular that $\mathbb{L}$ actually derives from a symmetric second-order tensor $\mathbf{B}$ as

$$\mathbb{L} = \lim_{\omega \to 0} \omega\, \mathbb{Q}^{-1} = \frac{3}{4}\, \underline{n} \overset{s}{\otimes} \mathbf{B} \overset{s}{\otimes} \underline{n} \tag{B.4}$$

For an arbitrarly anisotropic matrix, an algorithm allowing to estimate the limit (B.4) is proposed in (Barthélémy, 2009) whereas in the isotropic case $\mathbf{B}$ writes

$$\mathbf{B} = B_{nn}\, \underline{n} \otimes \underline{n} + B_{mm}\, \underline{m} \otimes \underline{m} + B_{\ell\ell}\, \underline{\ell} \otimes \underline{\ell}$$

with

$$\begin{aligned} B_{nn} &= \frac{8\,\eta\,(1-\nu^2)}{3\,E}\, \frac{1}{\mathcal{E}_\eta} \\ B_{mm} &= \frac{8\,\eta\,(1-\nu^2)}{3\,E}\, \frac{1-\eta^2}{\left(1-(1-\nu)\,\eta^2\right)\, \mathcal{E}_\eta - \nu\,\eta^2\, \mathcal{K}_\eta} \\ B_{\ell\ell} &= \frac{8\,\eta\,(1-\nu^2)}{3\,E}\, \frac{1-\eta^2}{(1-\nu-\eta^2)\, \mathcal{E}_\eta + \nu\,\eta^2\, \mathcal{K}_\eta} \end{aligned}$$

where $\mathcal{K}_\eta = \mathcal{K}(\sqrt{1-\eta^2})$ and $\mathcal{E}_\eta = \mathcal{E}(\sqrt{1-\eta^2})$ are the complete elliptic integrals of respectively the first and second kind (see (Abramowitz and Stegun, 1972)). If the crack is circular, the components of $\mathbf{B}$ become

$$B_{nn} = \frac{16\,(1-\nu^2)}{3\,\pi\,E} \quad ; \quad B_{mm} = B_{\ell\ell} = \frac{B_{nn}}{1-\nu/2}$$

B.4 Application of Hill calculation

Definition of the matrix tensor

```
C = stiff_Enu(1.,0.2) ; print(C)
```

```
Order 4 ISO tensor | Param(size=2)=[ 1.66667 0.833333 ] | Angles(size=0)=[ ]
[ 1.11111 0.277778 0.277778 0 0 0
  0.277778 1.11111 0.277778 0 0 0
  0.277778 0.277778 1.11111 0 0 0
  0 0 0 0.833333 0 0
  0 0 0 0 0.833333 0
  0 0 0 0 0 0.833333 ]
```

Calculation of the crack compliance $\mathbb{L} = \lim_{\omega \to 0} \omega\, \mathbb{Q}^{-1}$

Note that in *Echoes* it is necessary to provide an aspect ratio ω for the crack even if the crack compliance is actually calculated as a limit (not depending on ω)

```
ω = 1.e-4
L = crack_compliance(spheroidal(ω), C) ; print(L)
```

```
[[0.          0.          0.          0.          0.          0.          ]
 [0.          0.          0.          0.          0.          0.          ]
 [0.          0.          1.22230996 0.          0.          0.          ]
 [0.          0.          0.          0.67906109 0.          0.          ]
 [0.          0.          0.          0.          0.67906109 0.          ]
 [0.          0.          0.          0.          0.          0.          ]]
```

Checking the aspect ratio for which $\omega\, \mathbb{Q}^{-1} \approx \lim_{\omega \to 0} \omega\, \mathbb{Q}^{-1}$ is acceptable

```
tω = np.logspace(-5,1,20)
tabδ = []
for ω in tω:
    Q = hill_dual(spheroidal(ω), C)
    Lω = ω*np.linalg.inv(Q)
    δL = np.linalg.norm(Lω-L)/np.linalg.norm(L)
    tabδ.append(δL)
plt.figure(figsize=(8,3))
plt.loglog(tω,tabδ,'+-')
plt.xlabel(r"$\omega$")
```

```
plt.ylabel(r"$\frac{||\mathbb{L}-\omega\,\mathbb{Q}^{-1}||}{||\mathbb{L}||}$")
plt.grid(True,which='both')
plt.show()
```

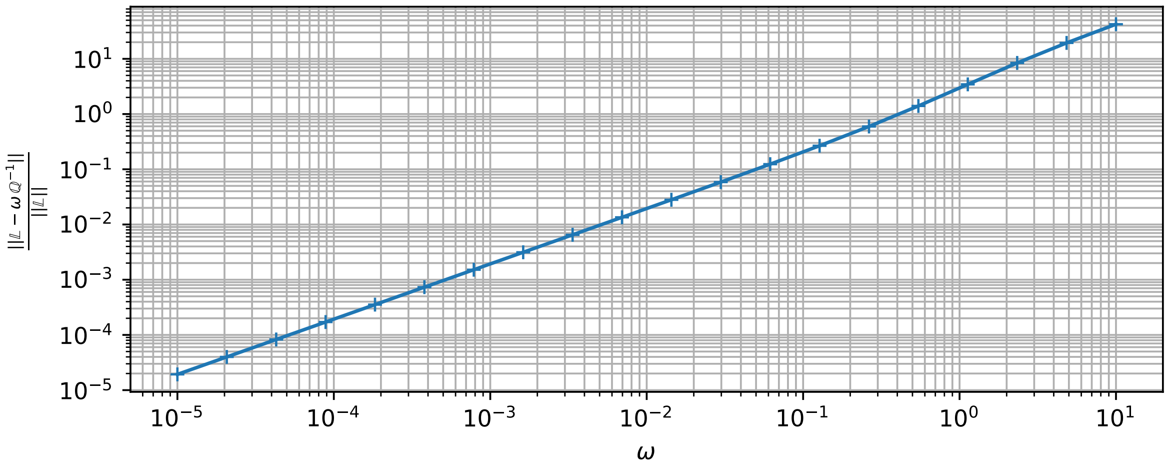


Figure B.2: Influence of the aspect ratio on the contribution tensor

C Basic problem of elasticity homogenization

C.1 System of equations

Consider a representative volume element (RVE) Ω composed of a heterogeneous material. Neglecting body forces in a problem posed at the scale of a RVE is consistent with the fact that the order of magnitude of mechanical effects induced by body forces is in general much lower than that of the macroscopic strain $\mathbf{E}$ or stress $\mathbf{\Sigma}$ effects accounting for interactions with particles surrounding the RVE (see (Dormieux et al., 2006)). The hypothesis of quasi-static equilibrium is also invoked here to write the balance law involving the Cauchy stress field $\boldsymbol{\sigma}$

$$\underline{\mathrm{div}}\,\boldsymbol{\sigma} = \underline{0} \quad (\Omega) \tag{C.1}$$

In the sequel, the small perturbation hypothesis is adopted so that the strain field $\boldsymbol{\varepsilon}$ derives from the displacement one $\underline{u}$ as the symmetrical part of its gradient

$$\boldsymbol{\varepsilon} = \frac{\mathbf{grad}\,\underline{u} + {}^{t}\mathbf{grad}\,\underline{u}}{2} \quad (\Omega) \tag{C.2}$$

In the framework of random media homogenization, two types of conditions applied at the boundary $\partial\Omega$ of a RVE Ω are usually considered:

- *homogeneous strain boundary conditions* corresponding to prescribed displacements $\underline{u}^g$ at $\partial\Omega$

  $$\underline{u}^g = \mathbf{E}\cdot\underline{x} \quad (\partial\Omega) \tag{C.3}$$

  It is noticeable that in this case the divergence theorem implies the following relationship between the microscopic and macroscopic strain tensors

  $$< \boldsymbol{\varepsilon} >_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{\varepsilon}\,\mathrm{d}\Omega = \frac{1}{|\Omega|}\int_{\partial\Omega} \underline{u}\overset{s}{\otimes}\underline{n}\,\mathrm{d}S = \mathbf{E} \tag{C.4}$$

  where the spatial average over a domain ω is denoted by $< \bullet >_\omega$ and $\underline{n}$ is the unit outward normal at the boundary. The macroscopic stress tensor is then simply defined as the average

  $$\mathbf{\Sigma} = < \boldsymbol{\sigma} >_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{\sigma}\,\mathrm{d}\Omega \tag{C.5}$$

- *homogeneous stress boundary conditions* corresponding to prescribed surface tractions $\underline{T}^g$ at $\partial\Omega$

  $$\underline{T}^g = \mathbf{\Sigma}\cdot\underline{n} \quad (\partial\Omega) \tag{C.6}$$

Now owing to the remarkable identity $(x_i\sigma_{jk})_{,k} = \sigma_{ij}$ resulting from (C.1) and the symmetry of $\boldsymbol{\sigma}$, the relationship between the microscopic and macroscopic stress is ensured by the divergence theorem

$$< \boldsymbol{\sigma} >_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{\sigma}\,\mathrm{d}\Omega = \frac{1}{|\Omega|}\int_{\partial\Omega} \underline{x}\overset{s}{\otimes}(\boldsymbol{\sigma}\cdot\underline{n})\,\mathrm{d}S = \boldsymbol{\Sigma} \tag{C.7}$$

The macroscopic stress tensor is then simply defined as the average

$$\mathbf{E} =< \boldsymbol{\varepsilon} >_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{\varepsilon}\,\mathrm{d}\Omega \tag{C.8}$$

Hill lemma

Note that whatever the choice of boundary conditions between (C.3) and (C.6), the consistency between the microscopic and macroscopic works is ensured by

$$< \boldsymbol{\sigma} : \boldsymbol{\varepsilon} >_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\varepsilon}\,\mathrm{d}\Omega = \frac{1}{|\Omega|}\int_{\partial\Omega} \underline{u}\cdot\boldsymbol{\sigma}\cdot\underline{n}\,\mathrm{d}S = \boldsymbol{\Sigma} : \mathbf{E} \tag{C.9}$$

which results from the application of the divergence theorem to $(u_i\sigma_{ij})_{,k} = u_{i,j}\sigma_{ij} = \varepsilon_{ij}\sigma_{ij}$.

The set of equations defining the problem posed on the RVE is finally completed by the local constitutive law relating the strain and stress fields. The hypothesis of linear elasticity is adopted in this part so that

$$\boldsymbol{\sigma} = \mathbb{c} : \boldsymbol{\varepsilon} \quad (\Omega) \tag{C.10}$$

where $\mathbb{c}(\underline{x})$ denotes the heterogeneous (positive definite fourth-order) stiffness tensor field satisfying the conditions of minor ($c_{jikl} = c_{ijlk} = c_{ijkl}$) and major ($c_{klij} = c_{ijkl}$) symmetries. The compliance tensor field is introduced as the inverse $\mathbb{s} = \mathbb{c}^{-1}$ in the sense of fourth-order tensors operating over symmetrical second-order tensors.

In short, the system of equations posed on the RVE is given by (C.1), (C.2), (C.3) or (C.6) and (C.10).

C.2 Macroscopic stiffness or compliance tensors

Whatever the boundary condition of homogeneous strain or stress type (C.3) or (C.6), the linearity of the problem allows to invoke the existence of concentration tensors relating the microscopic strain $\boldsymbol{\varepsilon}$ and stress $\boldsymbol{\sigma}$ fields to the macroscopic strain $\mathbf{E}$ or stress $\boldsymbol{\Sigma}$ tensors

$$\begin{aligned} \boldsymbol{\varepsilon} &= \mathbb{A}_E : \mathbf{E} \\ \boldsymbol{\sigma} &= \mathbb{B}_E : \mathbf{E} \quad \text{with} \quad \mathbb{B}_E = \mathbb{c} : \mathbb{A}_E \\ \boldsymbol{\sigma} &= \mathbb{B}_\Sigma : \boldsymbol{\Sigma} \\ \boldsymbol{\varepsilon} &= \mathbb{A}_\Sigma : \boldsymbol{\Sigma} \quad \text{with} \quad \mathbb{A}_\Sigma = \mathbb{s} : \mathbb{B}_\Sigma \end{aligned} \tag{C.11}$$