CraFT (version 1.1) user guide

Hervé Moulinec, Fabrice Silva

November 24, 2020

Contents

1 Introduction

In this chapter, one will describe how to run CraFT, i.e. what sort of input data CraFT needs, how to specify it to CraFT, what sort of output data are to be created, and how these data (input and output) are organized.

To describe the mechanical problem, the user must:

- describe the geometry of the microstructure via an image telling which phase each pixel belongs to
- describe the mechanical behavior of each phase, this is done in CraFT via two files:
  - a file describing all materials present in the microstructure: the type of behavior they obey (linear elasticity, elastic-perfectly plastic behavior, ...), and their peculiar mechanical properties (e.g. Young's modulus and Poisson coefficient in the cas of isotropic linear elasticity)
  - a file telling for each phase which material it belongs to, and how the material is oriented in that phase
- describe the loading conditions,
- tell what outputs the user wants to store at the end of the computation, and the name she/he wants to give to them
- give some tuning parameters of the method:
  - how to choose the reference material C_0
  - precision required for convergence of the iterative process

2 CraFT Usage

The user can run CraFT with `-f` option followed by the name of a file describing the inputs required by CraFT:

```
craft -f inputfile
```

or, more simply, the user can type:

```
craft inputfile
```

See section 3.1, page 5 for more details on input file format.

An other possibility, is to specify every input parameters separately by using adequate input options: `-c -p -m -l -o -C -e`

These options must be used together. Invoking them exclude use of `-f` options (and vice versa).

Table (2.1) summarizes all possible options of `craft` command.

`-h`	display help (this information)
`-V`	displays CraFT version number
`-v`	verbose mode (default: non verbose)
`[-f] <file>`	read inputs in file <file>
`-c <file>`	characteristic function is given in file <file>
`-p <file>`	phases described in file <file>
`-m <file>`	materials described is given in file <file>
`-l <file>`	loading conditions described in file <file>
`-o <file>`	outputs described in file <file>
`-r <file>`	restores a previous result stored in <file>
`-C <line>`	C0 specified in command line <line>:
`-C auto`	C0 is computed by craft (default)
`-C param <λ> <μ>`	Lamé coefficients of C0 set to <λ> and <μ>
`-e <prec1[,prec2[,prec3]>`	precision required: * prec1: precision required for stress divergence * prec2: precision required for loading conditions * prec3: precision required for compatibility If prec3 is omitted, it is set to be equal to prec2 If prec2 is omitted, it is set to be equal to prec1
`-s <scheme> [<α> [<β>]]`	iterative scheme to be used: 0: basic scheme (Moulinec Suquet 1998) (default) 1: accelerated scheme (Eyre Milton 1999) 2: augmented lagrangian scheme (Michel et a 2000) 3: Monchiet-Bonnet scheme (Monchiet Bonnet 2011) α and β are valid only for Monchiet-Bonnet scheme if beta is omitted, it is set to be equal to α if α is omitted, it is set to be equal to 1.5
`-n <threads>`	number of threads used by OpenMP
`-t <thermal strain file>`	either an image of the thermal strain field (if any) supposed to be constant in time or a file giving, for each phase and for each time step, the value of the thermal strain which is supposed to be uniform in the phase. No thermal strain if option -t is omitted

Table 2.1: CraFT options

3 Entering specifications of a given problem

3.1 Input file

The input specifications of a problem can be given to CraFT in a file, the name of which is entered by `-f` option (`-f` can be omitted).

Example:

```
craft -f toto.in
```

or, equivalently:

```
craft toto.in
```

Input file can:

- either contains exactly this list, in the same order (obsolescent),
  - the name of an image file describing the microstructure
  - the name of a file describing the phases of the microstructure
  - the name of a file describing the materials the phases are made of
  - the name of a file describing the loading conditions
  - the name of a file in which the user can specify the outputs
  - how to choose the "reference material" C_0
  - the required accuracy (the accuracy at which iterative processes for convergence have to be stopped)
- or can use keywords to enter specifications; in that case the different specifications can be entered in any order.

The two formats for entering specs can not be mixed together.

In both cases, a line beginning with a # character is considered as a comment line. An empty line (i.e. a line containing nothing or just white spaces) is ignored.

Format of input file without keywords:(obsolescent)

In format without keywords, spec have to be entered strictly in the following order:

- the name of an image file describing the microstructure
- the name of a file describing the phases of the microstructure

keywords	arguments
`microstructure`	the name of an image file describing the microstructure
`phases`	the name of a file describing the phases of the microstructure
`materials`	the name of a file describing the materials the phases are made of
`loading`	the name of a file describing the loading conditions
`temperature`	the name of a file describing the loading conditions in temperature
`output`	the name of a file in which the user specify the outputs she/he wants
`C0`	how to choose the "reference material" C_0
`precision`	the required precision
`scheme`	iterative scheme used (basic scheme, Eyre-Milton scheme, augmented Lagrangian scheme or Monchiet-Bonnet scheme)
`thermal strain`	name of the file describing the thermal strain field

Table 3.1: Keywords available in input files

- the name of a file describing the materials the phases are made of
- the name of a file describing the loading conditions
- the name of a file in which the user specify the outputs she/he wants
- how to choose the "reference material" C_0
- the required precision (i.e. the accuracy at which iterative processes for convergence have to be stopped)

An example of an input file without keywords is given in annex A.1, page 37.

Format of input file with keywords:

An input file using keywords contains lines beginning with one of the available keywords (summarized in table 3.1). The keyword is followed by a = character, and then by the specification itself. Case distinction is ignored in keywords. Blanks are ignored.

An example of an input file with keywords is given in annex A.2, page 38.

For keywords accepting a file name as argument (i.e.: `microstructure` ,`phases`, `materials` , `loading` and `output`), it is also possible to directly enter the content of the file into the input file. In that case, the keyword is followed by the content of the file enclosed by braces (`{` and `}`).

Example A.3 in page 39 illustrates this case.

3.2 Scheme used

The iterative scheme used by craft can be one of the following:

- basic scheme
- Eyre-Milton accelerated scheme (see Eyre & Milton in [8] for more details)
- accelerated scheme based on augmented Lagrangian (see Michel et al in [10] and [13])
- Monchiet & Bonnet accelerated scheme (see Monchiet & Bonnet in [11])

As demonstrated in Moulinec & Silva in [17], Eyre-Milton scheme and Lagrangian scheme are particular cases of Monchiet-Bonnet scheme, the scheme of Eyre and Milton corresponding to the case when the parameters α and β of the scheme of Monchiet and Bonnet are equal to 2 ($\alpha = \beta = 2$), the augmented Lagrangian scheme of Michel et al corresponding to the case when $\alpha = \beta = 1$.

Important remark: we define the parameter β as the opposite ($-\beta$) of the one defined by Monchiet & Bonnet to insure it to be positive.

In the craft input file, the scheme used is specified by `scheme` keyword, followed by a number and by parameters α and β in the case of Monchiet & Bonnet scheme.

To choose the basic scheme, the user must write:

```
scheme=0
```

to choose the scheme of Eyre & Milton :

```
scheme=1
```

or, equivalently:

```
scheme=3 2.  2.
```

for the augmented Lagrangian scheme:

```
scheme=2
```

or, equivalently:

```
scheme=3 1.  1.
```

for Monchiet & Bonnet scheme:

```
scheme=3 α  β
```

where α and β must be floating values. If omitted, β is set to be equal to α. If α is omitted, it is set to be equal to `1.5` .

3.3 File describing the microstructure

The microstructure of the problem treated is described by an image file in CraFT format or in “simple legacy” VTK format.

```
#-------------------------------------------------------------
#  phase  material     phi1            Phi            phi2
#-------------------------------------------------------------
     0       0       3.8135413       1.8862685       1.1466009
     1       0       2.7503878       1.7827771       4.2127749
     2       0       2.0567105       1.6476569       3.3482159
     3       0       4.3410043       1.1427749        3.907608
     4       0       3.5043039       1.4998321       4.8580132
     5       0       4.4619361       1.6873032       6.1930471
     6       0        4.028708         2.07412       5.1103068
     7       0       2.2259622       1.4106619       1.8815486
     8       0       1.0618022       2.0650686      0.89195132
     9       0       4.6502682       1.5093459        5.475368
```

Figure 3.1: example of file describing the phases of a microstructure. All of the 10 phases are composed of the same material (whose id is 0) but do have different crystalline orientations.

In both format, each pixel of a microstructure image must contain the index of the phase it belongs to.

(see section (5), page 28, for more details on digital images).

See `www.vtk.org/VTK/img/file-formats.pdf` for details on simple legacy VTK file format.

3.4 File describing the phases

Phases in the microstructure are described in an ascii file, in which every phase in the microstructure is described by a line. First column gives the number of the phase, second column gives the number of the material the given phase is composed of, the next three colums give the orientation of the material in the given phase by three Euler angles ϕ_1, Φ, ϕ_2 (see figure 3.2 for details).

Empty lines and lines beginning with `#` character are ignored by CraFT.

Remarks:

- A given phase is uniquely described by an id number.
- Phases are not necessarily numbered from 0 to n: phase file has just to describe every phase present in image file of the microstructure, however the phases are numbered in the image.
- Phases in phase file can be more numerous that actual phases in microstructure image: the only prescription is that every phase in the microstructure must be described in phase file.

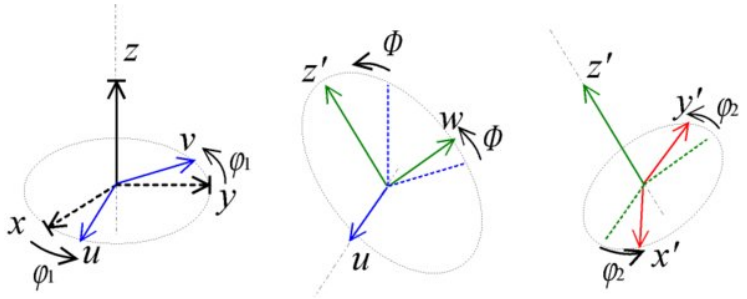


Figure 3.2: Euler angles with Bunge notations

3.5 File describing the materials

A unique file describes all phases of the microstructure. This is an ascii file composed of paragraphs each of which describing a given material.

Each paragraph begins with a line describing the id number of the material and the id number of the constitutive law that the material obeys.

The next lines give the value of the parameters of the constitutive law; their number, types and order depending on the constitutive law and how it has been implemented. For example, for an isotropic linear elastic behavior just two parameters has to be entered: Young's modulus and Poisson coefficient and for an elastic perfecftly plastic behavior, three parameters are required: Young's modulus, Poisson coefficient and yield stress.

Table 3.2 summarizes the ids of the different constitutive laws that have been implemented till now.

Details of parameters to be entered for each behavior are given in appendix (B).

Empty lines are ignored and lines beginning with # character are considered as comments by CraFT in material description files.

3.6 Thermal strain

3.6.1 Generalities about thermal strain in CraFT

A file describing the thermal strain can be optionally entered as input. In that case, the constitutive relations are applied on the strain from which the thermal strain has been previously subtracted.

If no thermal strain image is entered, it is supposed that there is no thermal strain in the considered problem.

In the case of linear elasticity, the constitutive relation in the presence of a thermal

id	constitutive law
0	void
(1)	(obsolete: use 10 instead) isotropic linear elasticity
(2)	(obsolete: use 4 instead) elastic perfectly plastic behavior (with isotropic elasticity)
(3)	(obsolete: use 10 instead) anisotropic linear elasticity
4	elastic plastic behavior (with isotropic elasticity)
10	linear elasticity
40	elasto visco-plastic behavior (von Mises plasticity, power law viscosity)
41	elasto visco-plastic behavior (von Mises plasticity, power law viscosity with linear kinematic hardening)
50	Voce-type elasto-visco-plastic behavior
60	Gurson-type behavior

Table 3.2: CraFT identification numbers of constitutive laws. Obsolete behavior ids are enclosed in parentheses.

strain denoted by $\boldsymbol{\varepsilon}^{th}(\boldsymbol{x})$ can be written as:

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{C} : (\boldsymbol{\varepsilon}(\boldsymbol{x}) - \boldsymbol{\varepsilon}^{th}(\boldsymbol{x}))$$

In the general case of a constitutive relation described in the form:

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{\varepsilon}(\boldsymbol{x}), ...)$$

it becomes, in the presence of thermal strain:

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{\varepsilon}(\boldsymbol{x}) - \boldsymbol{\varepsilon}^{th}(\boldsymbol{x}), ...) \ .$$

The file entered to specify the thermal strain can either be an image of the thermal strain fields, in which case the thermal strain field is supposed to be constant in time, or it can be a file in which a given value of the thermal strain is specified for each phase concerned, and for different time steps, in which case the thermal strain is supposed to be uniform per phase, but to be able to change with time.

3.6.2 The thermal strain is specified by an image file

The thermal strain image must be a VTK file of 2d order tensors, i.e. with 6 double precision scalar values per pixel/voxel (see 5.3).

The program `pico` is a useful tool for creating piecewise constant images of 2d order tensors; it is included in the CraFT package.

When an image of the thermal strain is entered as parameter of `-t` option, the thermal strain is supposed not to change with temperature.

3.6.3 The thermal strain is specified by a file giving ther thermal strain per phase and time step

The case of thermal strain being uniform per phase (i.e. the thermal strain has the same value in every point of a given phase) but which can change with time, can be chosen by the user by entering an ascii file organized as follows:

- the first line contains a list of phase identifiers. These phases are not necessarily those present in the microstructure structure, and the phases in the microstructure do not necessarily have to be present in the table.
- the following lines contains a time value in the first column, followed by a number of values equal to the number of phases enumerated in the first table, multiplied by 6, corresponding to the 6 components of the thermal strain of each phase in the order presented in the first line. Warning: the time values must be placed in ascending order. The 6 components of the thermal strain are organized as follows: $\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23}$.
- blank lines or lines beginning with a # character are ignored.

for a given time value t, the table of the thermal strain is explored and the 2 successive time values t_i and t_{i+1} surrounding t are determined, i.e.:

$$t_i \leq t < t_{i+1}$$

then the thermal strain for time t is evaluated by a linear interpolation of the thermal strain values at times t_i and t_{i+1}:

$$\boldsymbol{\varepsilon}^{th}(t) = \boldsymbol{\varepsilon}^{th}(t_i) + \frac{t - t_i}{t_{i+1} - t_i}\left(\boldsymbol{\varepsilon}^{th}(t_{i+1}) - \boldsymbol{\varepsilon}^{th}(t_i)\right)$$

Example: The following table

```
      17                  23                  38                  45
0.    0.1 0. 0. 0. 0. 0.  0. 0.1 0. 0. 0. 0.  0. 0. 0.1 0. 0. 0.  0. 0. 0. 0.1 0. 0.
10.   0.2 0. 0. 0. 0. 0.  0. 0.2 0. 0. 0. 0.  0. 0. 0.2 0. 0. 0.  0. 0. 0. 0.2 0. 0.
50.   0.3 0. 0. 0. 0. 0.  0. 0.3 0. 0. 0. 0.  0. 0. 0.3 0. 0. 0.  0. 0. 0. 0.3 0. 0.
```

means that phases number 17, 23, 38 and 45 are supposed to be subjected to a uniform thermal strain whose 6 components are given for times $t = 0,\ 10$ and 50. For example, phase 38 has a thermal strain of $\boldsymbol{\varepsilon}^{th} = (0., 0., 0.2, 0., 0., 0.)$ at time $t = 10$. Phases other than phase 17, 23, 38 and 45 are assumed not to be subject to any thermal strain ($\boldsymbol{\varepsilon}^{th} = \mathbf{0}$).

3.7 Loading specifications

Files specifiying loading conditions consist in two parts:

- loading condition (prescribed stress, prescribed strain or prescribed direction of stress)
- one or more lines describing every loading step(s)

3.7.1 loading condition

CraFT enables different loading conditions:

- prescribed macroscopic strain : macroscopic strain E is imposed
- prescribed macroscopic stress: macroscopic strain $\boldsymbol{\sigma}$ is imposed
- mixed conditions: some components of the macroscopic strain and the other components of the macroscopic stress are prescribed
- prescribed direction of stress + prescribed strain in that direction: macroscopic stress $\boldsymbol{\sigma}$ has to be colinear to prescribed direction of stress $\boldsymbol{\sigma}_0$ and the product of macroscopic stress and macroscopic strain, i.e. $\boldsymbol{\sigma} : E$, is prescribed.

Loading specification file begins with a line containing a letter:

- `D` : prescribed macroscopic strain
- `C` : prescribed macroscopic stress
- `S` : prescribed direction of stress

Mixed loading conditions are specified by entering 6 differents letters D or C (separated by blanks, commas or nothing) corresponding to the 6 components prescribed. For example, `DCCDDC` means that **strain** components E_{11}, E_{12} and E_{13} and **stress** components Σ_{22}, Σ_{33} and Σ_{23} will be prescribed.

3.7.2 Loading steps

Loading steps may be specified step-by-step: a given line describes one given step, or implied loops may be used to specify several steps in one line.

"Step-by-step" specification

A basic line of loading step specification comprises 8 values:

- the time value (for example in seconds), hereafter called t
- the 6 components of a symmetrical 2d order tensor, hereafter called $\mathbf{d}$ supposed to be entered in the following order: 11, 22, 33, 12, 13, 23
- a scalar, hereafter called k

k and $\mathbf{d}$ do have different meaning depending on loading condition:

- in the case of prescribed macroscopic strain : macroscopic strain $\mathbf{E}$ at time t is given by $\mathbf{E} = k.\mathbf{d}$

- in the case of prescribed macroscopic stress: macroscopic stress $\mathbf{\Sigma}$ at time t is given by $\mathbf{\Sigma} = k.\mathbf{d}$
- in the case of prescribed direction of stress, the macroscopic stress must be colinear to $\mathbf{d}$ (in other words: $\mathbf{d}$ is the direction of stress) and the product of the macroscopic strain by $\mathbf{d}$ must be equal to k: $\mathbf{E} : \mathbf{d} = k$

Important note: CraFT implies that at time $t = 0$, loading modulus k is null and direction d is useless.

For example:

```
#-----------------------------------
# prescribed strain
D
#-----------------------------------
# loading
#t      direction             k
#       11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - -
0.1      1. 0. 0. 0. 0. 0.     2.
#
```

In this file, macroscopic strain is prescribed, the loading consists in one step at time: $t = 0.1s$, the macroscopic strain $\mathbf{E}$ must be equal to $(E_{11}, E_{22}, E_{33}, E_{12}, E_{13}, E_{33}) = (2, 0, 0, 0, 0, 0)$

Implied loop specification

In the case of a monotonic loading, it can be tedious to enter the lines of every required time steps, where the time values t and the loading modulus k change regularly from one step to the next.

CraFT proposes an implied loop notation enabling to specify several time steps in one line.

Implied loops are specified at the beginning of a line (before time value specification). Two notations are possible:

- an integer value enclosed by `:` characters specifying the number of implied loops between the time step of the preceding line (not included) and the time step of the current line (included),
- a float value enclosed by `%` characters specifying the implied time steps between the time of the preceding line and the time step of the current line.

The time values t and the "loading modulus" k of the so-created loading steps are supposed to be linearly interpolated between their value in previous line ant in the current line; the directions d of the so-created loading steps are supposed to equal to the one in the current line.

For example:

```
#-----------------------------------
# prescribed strain
D
#-----------------------------------
# loading
#      t          direction       k
#             11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
:10:   1.    1. 0. 0. 0. 0. 0.   10.
#
```

will create 10 loading steps form $t = 0.1$ to $t = 1.$, with a modulus k varying form $k = 1.$ to $k = 10.$ (as the previous time step is implicitely considered as $t = 0$ and $k = 0$).

It would have been equivalently written as:

```
#-----------------------------------
# prescribed strain
D
#-----------------------------------
# loading
#      t          direction        k
#             11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
       0.1   1. 0. 0. 0. 0. 0.    1.
       0.2   1. 0. 0. 0. 0. 0.    2.
       0.3   1. 0. 0. 0. 0. 0.    3.
       0.4   1. 0. 0. 0. 0. 0.    4.
       0.5   1. 0. 0. 0. 0. 0.    5.
       0.6   1. 0. 0. 0. 0. 0.    6.
       0.7   1. 0. 0. 0. 0. 0.    7.
       0.8   1. 0. 0. 0. 0. 0.    8.
       0.9   1. 0. 0. 0. 0. 0.    9.
       1.0   1. 0. 0. 0. 0. 0.   10.
```

or:

```
#-----------------------------------
# prescribed strain
D
#-----------------------------------
# loading
#      t           direction     k
#             11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
%0.1%  1.    1. 0. 0. 0. 0. 0.   10.
#
```

as implied-time step is equal to 0.1, the number of implied loops between $t = 0$ (preceding line) and $t = 1$ (current line) is $1/0.1 = 10$

Examples of loading file are given in C.1.

3.8 Loading condition in temperature

The user can specify the evolution in time of the temperature conditions. The temperature at a given time is supposed to be uniform in the volume.

The temperature loading can be entered a file containing two columns for time and temperature and an arbitrary number of lines. For example:

```
# Loading conditions in temperature
#
# time                temperature
10.                   100.
20.                   200.
50.                   250.
100.                  260.
```

The values of time in this file do not have to match the time values present in the file of the mechanical loading conditions (described above in section 3.7). The temperature T at an arbitrary time t is calculated as follows:

- if t is lower than the lowest time value t_{min} in the temperature file, temperature T is set to the temperature corresponding to t_{min},
- if t is greater than the greatest time value t_{max} in the temperature file, temperature T is set to the temperature corresponding to t_{max},
- in the other case, T is linearly interpolated using the temperatures of the two values of time of the temperature file, between which t lays.

In other words, denoting t_i and T_i the successive values in temperature file, **sorted in ascending values of time**, with varying from 1 to n, one has:

$$\begin{array}{lll} T = T_1 & \text{if} & t < t_1 \\ T = T_n & \text{if} & t \geq t_n \\ T = T_i + (t - t_i)\frac{T_{i+1} - T_i}{t_{i+1} - t_i} & \text{if} & t_i \leq t < t_{i+1} \end{array} \tag{3.1}$$

Remark: The values in the file can be set in any order, the program will reorganize it in ascending values of time.

3.9 Output specifications

Output specification files contain lines beginning with a keyword, which can be composed of several words, followed by a = and one or several arguments.

Availables keywords are:

- `generic name`

- *xxx* `image` (where *xxx* is the name of a mechanical variable (stress, strain, ...) to be stored as an image
- *xxx* `moment` (where *xxx* is the name of a mechanical variable (stress, strain, ...) whose first and second moments have to be calculated and stored for each phase.

3.9.1 keyword: `generic name`

Argument following `generic name` is to be the lexical root of the names of all output files. For example, if output spec file contains line:

```
generic name=foo
```

craft will build a `foo.res` file that contains macroscopic results at each step of the loading path, a `foo.perf` file to display statistics about execution, etc ...

3.9.2 keyword: *xxx* `image`

Saving a field of a given variable as an image file

If the argument of keyword *xxx* `image` is `yes` (or if no argument is given), image(s) of *xxx* field are to be created. If argument is `no`, no images are created.

xxx is the name of a mechanical variable; it can be common to all possible mechanical behaviors; i.e. `strain` and `stress`, or it can be specific to a given behavior. A list of variables available for image storing is given for every behavior (see appendix B).

The equivalent part or the trace of any tensor field can be designated to be stored by preceding its name by keyword `equivalent` or `trace`, respectively. If the name of the field contains the "`strain`" (respectively "`stress`") substring, the equivalent part is considered as an equivalent strain (respectively stress).

In addition, the field of the rotation can be saved as an image of 3D vector. For more details see appendix (D).

Name of the image (output) file

The name of the image(s) to be created is built with the "generic name" followed by `_t=` and the time (in the loading path) at which the image has been captured, followed by the name of the variable and is ended by an extension depending on the image format (`.vtk` for VTK file format or `.i3d` for CraFT image format.

If the CraFT image format has been chosen, as it is not able to store any other field as scalar fields , each of the components of a given tensor or vector is stored is a separate file, whose name is ended by the number of the component: `11`, `22`, `33`, `12`, `13`, `23` for a tensor, `001`, `002`, `003`, ... for a vector.

For example, the following output spec file:

```
generic name=foo
stress image=yes
```

will create images of the 6 components of the stress field at the last time step of the loading path (let us say, at time 1s):

`foo_t=01.00000000e+00_stress11.i3d`,

`foo_t=01.00000000e+00_stress22.i3d`,

`foo_t=01.00000000e+00_stress33.i3d`,

`foo_t=01.00000000e+00_stress12.i3d`,

`foo_t=01.00000000e+00_stress13.i3d`,

`foo_t=01.00000000e+00_stress23.i3d`.

Saving images at different times of the loading path

If just `yes` is given argument, the image(s) is stored at the last step of the loading path.

Moreover one can give the time(s) at which images are to be stored by entering a list of time specifications separated by commas.

Time values can be filled either by their actual value (in seconds) or by the number of their step in the loading path, in which case this number is entered as an integer value preceded by an at sign character (`@`).

The time value of the first step of the loading path can be specified by `firt` or by `begin` (or by its actual time value).

The time value of the last step of the loading path can be specified by `last` or `end` (or by its actual time value).

When two time specifications are separated by a colon character (`:`), images are to be stored at each step of the loading path between these two extreme time values. If a second colon character is entered and followed by a time value, this last value is taken as a time step.

Examples

```
strain image = yes 10.,20, 30.:40.:@2, 45.:@100, @200
```

requires to store images of the strain tensor at times: $t = 10s$, $t = 20s$, once at every two time steps between $t = 30s$ and $t = 40s$, at every time steps between $t = 45s$ and the 100^{th} steps, and at the 200^{th} step of the loading path.

```
stress image=yes first:last
```

requires to store images of the stress tensor at every steps of the loading path (i.e. from the first step to the last one).

```
equivalent stress image=yes 180.:last:@2
```

requires to store an image of the equivalent stress once at every two steps, from time $t = 180s$ to the last step of the loading path.

```
rotation image=yes
```

requires to store an image of the rotation at the last step of the loading path. This is a 3d vector image (see (D).

3.9.3 keyword: `im_format`

The format of the images which have to be stored as results of the computation can be specified by keyword `im_format`.

- `im_format=vtk`

  simple legacy VTK file format is prescribed

- `im_format=i3d`

  CraFT image format is prescibed

- `im_format=all`

  every image to be stored will be saved under both formats (VTK and i3d).

3.9.4 keyword: *xxx* `moment`

First and second moments of *xxx* variable has to be stored during calculation.

The syntax is similar to the one for image storage. The only difference, is that a sole file will be created for each variable whose moments are required to be stored, even if several times for storage are given.

Example:

```
generic name=foo
strain moment = yes 10.:20
```

will create a file named `foo_strain.mom` containing the first and second moments of the strain field in every phase, at every time steps between 10s and 20s.

3.9.5 keyword: `variables`

All variables describing the mechanical state of the material can be stored at given times of the loading path using keyword `variables`.

The syntax is similar to the one for image storage.

See chapter 4 for more details.

Example:

```
generic name=foo
variables = yes 10.:20, @40
```

will create files like `foo_t=01.00000000e+00_variables.h5` containing the variables in every phase, at every time steps between 10s and 20s, and at the 40^th loading step.

3.10 Specification of reference material C_O

The reference material C_0 can be chosen:

- either automatically by entering: `auto` keyword (recommended)
- or explicitely by entering: `param` keyword followed the two Lamé coefficient of C_0 (C_0 is an istropic linear elastic material).

3.11 Required accuracy

The numerical method implemented in CraFT use an iterative process at each step of the loading path. The convergence is reached when:

1 the equilibrium has been reached,

2 the loading conditions are satisfied,

2 the strain field statisfies compatibility conditions.

In practice, these criteria are evaluated by means of test functions, the results of which are compared with values of the required accuracy.

3.11.1 Equilibrium test

Four different test functions are available to evaluate the deviation from equilibrium

0. Norm of the divergence of the stress

   The modulus of the divergence of the stress field is computed in Fourier space as:

$$\|div(\boldsymbol{\sigma})\| = \sqrt{\sum_{\boldsymbol{\xi}} |\boldsymbol{\xi}.\hat{\boldsymbol{\sigma}}(\boldsymbol{\xi})|^2} \tag{3.2}$$

and normalized by the Frobenius norm of the average stress as

$$\begin{aligned} \text{equilibrium deviation} &= \frac{\|div(\boldsymbol{\sigma})\|}{\| < \boldsymbol{\sigma} > \|} \\ &= \frac{\sqrt{\sum_{\boldsymbol{\xi}} |\boldsymbol{\xi}.\hat{\boldsymbol{\sigma}}(\boldsymbol{\xi})|^2}}{\sqrt{< \sigma_{ij} >:< \sigma_{ij} >}} \\ &= \frac{\sqrt{\sum_{\boldsymbol{\xi}} |\boldsymbol{\xi}.\hat{\boldsymbol{\sigma}}(\boldsymbol{\xi})|^2}}{\sqrt{\hat{\sigma}_{ij}(\mathbf{0}) : \hat{\sigma}_{ij}(\mathbf{0})}} \end{aligned} \tag{3.3}$$

(as the value at frequency 0 in Fourier space is equal to the average of the considered field).

This value is compared to a value entered by the user. Equilibrium is assumed to be reached when:

$$\frac{\|div(\boldsymbol{\sigma})\|}{\| < \boldsymbol{\sigma} > \|} < \quad required\_accuracy\_for\_divergence\_of\_stress$$

As explained in [19], this criterion is generally unnecessarily restrictive. To put it simply, after some iterations, no useful additional information is accessible through further iterations, as the spatial discretization is insufficient to capture the details needed to assess a better precision. Nevertheless, the criterion 0 based on the divergence of the stress field remains high and may decrease slowly with iterations.

1. Maximum value of the divergence of the stress (not recommended)

   Another possible test of equilibrium consits in comparing the maximum value of the divergence of the stress, in Fourier space, with a prescribed accuracy.

$$\begin{aligned} \text{equilibrium deviation} &= \frac{\max_{\xi} \|\hat{\boldsymbol{\sigma}}\|}{\| < \boldsymbol{\sigma} > \|} \\ &= \frac{\max_{\xi} \|\hat{\boldsymbol{\sigma}}\|}{\|\hat{\boldsymbol{\sigma}}(\mathbf{0})\|} \end{aligned} \tag{3.4}$$

   This criterion is even more restrictive than the previous one, and, for that reason, it is not recommended.

2. Monchiet and Bonnet equilibrium test (not recommended)

   In their article [11] propose an equilibrium test less restrictive than the two previous ones. Although not recommended by the author of these lines, this criterion can be used in CraFT. For more details, see [11] and section 4.2.3 of [18].

3. Bellis equilibrium test (recommended)

In his article of 2019 ([20]), Bellis introduced an equilibrium criterion based on an ad-hoc energetic principle, which is implemented in CraFT code as

$$\text{equilibrium deviation} = \frac{\|\boldsymbol{\Gamma_0\sigma} : \boldsymbol{C_0} : \boldsymbol{\Gamma_0\sigma}\|}{|<\boldsymbol{\sigma}>:<\boldsymbol{\epsilon}>|^{\frac{1}{2}}} \tag{3.5}$$

where $\boldsymbol{C_0}$ is the elastic modulus of the reference material (see 3.10), and where $\boldsymbol{\Gamma_0}$ is the associated Green operator.

This criterion can be evaluated either in Fourier space as

$$\text{equilibrium deviation} = \frac{\sum_{\boldsymbol{\xi}} \left(\left(\hat{\boldsymbol{\Gamma}}_{\mathbf{0}}(\boldsymbol{\xi}) : \hat{\boldsymbol{\sigma}}(\boldsymbol{\xi}) \right) : \boldsymbol{C_0} : \left(\hat{\boldsymbol{\Gamma}}_{\mathbf{0}}(\boldsymbol{\xi}) : \hat{\boldsymbol{\sigma}}(\boldsymbol{\xi}) \right) \right)^{\frac{1}{2}}}{|\hat{\boldsymbol{\sigma}}(\mathbf{0}) : \hat{\boldsymbol{\epsilon}}(\mathbf{0})|^{\frac{1}{2}}} \tag{3.6}$$

or in real space as

$$\text{equilibrium deviation} = \frac{<\boldsymbol{e} : C_0 : \boldsymbol{e}>^{\frac{1}{2}}}{|<\boldsymbol{\sigma}>:<\boldsymbol{\epsilon}>|^{\frac{1}{2}}} \tag{3.7}$$

(with $\boldsymbol{e} = \boldsymbol{\Gamma_0\sigma}$).

3.11.2 Test on loading conditions

Basically, the iterative process enables to prescribe macroscopic strain by forcing the strain field in Fourier space at null frequency to a given value.

Nevertheless, it is possible to prescribe macroscopic stress or to prescribe the direction of macroscopic stress via a secondary iterative scheme which proposes, at each iteration, a new macroscopic strain which is then imposed to the null frequency of the strain field (see section 3.7). Thus, it has to be verified, at each iteration, if prescribed loading conditions has been reached or not.

In the case of prescribed macroscopic stress, the iterative scheme is the following:

$$\mathbf{E}^{i+1} = \mathbf{E}^{i} + {\mathbf{C_0}}^{-1} : (\boldsymbol{\Sigma} - <\boldsymbol{\sigma}^{i}>)$$

where:

- $\mathbf{E}^i$ is the macroscopic strain at iteration i
- $\boldsymbol{\Sigma}$ is the prescribed macroscopic stress
- $<\boldsymbol{\sigma}^i>$ is the overall mean of the stress field at iteration i
- ${\mathbf{C_0}}^{-1}$ is the stiffness of reference material $\mathbf{C_0}$

and the convergence condition is:

$$\frac{\|\boldsymbol{\Sigma} - <\boldsymbol{\sigma}^{i}>\|}{\|\boldsymbol{\Sigma}\|} \quad < \quad required\_accuracy\_for\_loading\_conditions$$

In the case of prescribed direction of macroscopic stress, the iterative scheme is the following:

$$\begin{aligned} {\mathbf{C_0}}^{-1} : \mathbf{E}^{i+1} - k^{i+1}\mathbf{\Sigma_0} &= {\mathbf{C_0}}^{-1} : \mathbf{E}^{i} - < \boldsymbol{\sigma}^i > \\ \mathbf{E}^{i+1} : \mathbf{\Sigma_0} &= E(t) \end{aligned}$$

where:

- $\mathbf{\Sigma_0}$ is the prescribed direction of macroscopic stress
- $E(t)$ is the prescribed macroscopic strain in $\mathbf{\Sigma_0}$ direction (it is a scalar),
- $< \boldsymbol{\sigma}^i >$ is the overall mean of the stress field at iteration i
- k^{i+1} is a scalar to be computed

and the convergence condition is:

$$\frac{\| k^i \mathbf{\Sigma_0} - < \boldsymbol{\sigma}^i > \|}{\| k^i \mathbf{\Sigma_0} \|} \; < \; required\_accuracy\_for\_loading\_conditions$$

3.11.3 Compatibility test

Compatibility conditions are guaranted when the "basic scheme" is used, but they are not when one uses one of the "'accelerated schemes". In that case, one must test the deviation form compatibility.

There are six compatibility relations to be satisfied:

$$\begin{aligned} \frac{\partial^2 e_{11}}{\partial x_2^2} + \frac{\partial^2 e_{22}}{\partial x_1^2} - 2\frac{\partial^2 e_{12}}{\partial x_1 \partial x_2} &= 0 \, , \\ \frac{\partial^2 e_{22}}{\partial x_3^2} + \frac{\partial^2 e_{33}}{\partial x_2^2} - 2\frac{\partial^2 e_{23}}{\partial x_2 \partial x_3} &= 0 \, , \\ \frac{\partial^2 e_{33}}{\partial x_1^2} + \frac{\partial^2 e_{11}}{\partial x_3^2} - 2\frac{\partial^2 e_{13}}{\partial x_3 \partial x_1} &= 0 \, , \\ \frac{\partial^2 e_{11}}{\partial x_2 \partial x_3} - \frac{\partial^2 e_{13}}{\partial x_1 \partial x_2} - \frac{\partial^2 e_{12}}{\partial x_1 \partial x_3} + \frac{\partial^2 e_{23}}{\partial x_1 \partial x_1} &= 0 \, , \\ \frac{\partial^2 e_{22}}{\partial x_3 \partial x_1} - \frac{\partial^2 e_{12}}{\partial x_2 \partial x_3} - \frac{\partial^2 e_{23}}{\partial x_2 \partial x_1} + \frac{\partial^2 e_{13}}{\partial x_2 \partial x_2} &= 0 \, , \\ \frac{\partial^2 e_{33}}{\partial x_1 \partial x_2} - \frac{\partial^2 e_{23}}{\partial x_3 \partial x_1} - \frac{\partial^2 e_{13}}{\partial x_3 \partial x_2} + \frac{\partial^2 e_{12}}{\partial x_3 \partial x_3} &= 0 \, . \end{aligned}$$

The deviation from compatibility can be easily evaluated in Fourier space by

$$\epsilon_{compatibility} = \frac{\max_{\boldsymbol{\xi}}(\max_{j=1,\ldots,6}(|\widehat{c_j}(\boldsymbol{\xi})|))}{\sqrt{\sum_{\boldsymbol{\xi}} \widehat{e_{ij}}(\boldsymbol{\xi}) : \widehat{e_{ij}}^*(\boldsymbol{\xi})}} \tag{3.8}$$

with

$$\begin{aligned} \widehat{c_1}(\boldsymbol{\xi}) = \quad & -\xi_2\xi_2\widehat{e_{11}}(\boldsymbol{\xi}) - \xi_1\xi_1\widehat{e_{22}}(\boldsymbol{\xi}) + 2\xi_1\xi_2\widehat{e_{12}}(\boldsymbol{\xi}) \, , \\ \widehat{c_2}(\boldsymbol{\xi}) = \quad & -\xi_3\xi_3\widehat{e_{22}}(\boldsymbol{\xi}) - \xi_2\xi_2\widehat{e_{33}}(\boldsymbol{\xi}) + 2\xi_2\xi_3\widehat{e_{23}}(\boldsymbol{\xi}) \, , \\ \widehat{c_3}(\boldsymbol{\xi}) = \quad & -\xi_1\xi_1\widehat{e_{33}}(\boldsymbol{\xi}) - \xi_3\xi_3\widehat{e_{11}}(\boldsymbol{\xi}) + 2\xi_3\xi_1\widehat{e_{13}}(\boldsymbol{\xi}) \, , \\ \widehat{c_4}(\boldsymbol{\xi}) = \quad & -\xi_2\xi_3\widehat{e_{11}}(\boldsymbol{\xi}) + \xi_1\xi_2\widehat{e_{13}}(\boldsymbol{\xi}) + \xi_1\xi_3\widehat{e_{12}}(\boldsymbol{\xi}) - \xi_1\xi_1\widehat{e_{23}}(\boldsymbol{\xi}) \, , \\ \widehat{c_5}(\boldsymbol{\xi}) = \quad & -\xi_3\xi_1\widehat{e_{22}}(\boldsymbol{\xi}) + \xi_2\xi_3\widehat{e_{12}}(\boldsymbol{\xi}) + \xi_2\xi_1\widehat{e_{23}}(\boldsymbol{\xi}) - \xi_2\xi_2\widehat{e_{13}}(\boldsymbol{\xi}) \, , \\ \widehat{c_6}(\boldsymbol{\xi}) = \quad & -\xi_1\xi_2\widehat{e_{33}}(\boldsymbol{\xi}) + \xi_3\xi_1\widehat{e_{23}}(\boldsymbol{\xi}) + \xi_3\xi_2\widehat{e_{13}}(\boldsymbol{\xi}) - \xi_3\xi_3\widehat{e_{12}}(\boldsymbol{\xi}) \, . \end{aligned}$$

Finally, the test on compatibility conditions is:

$$\epsilon_{compatibility} < required\_accuracy\_for\_compatibility \tag{3.9}$$

3.11.4 How to enter required accuracy

CraFT user has to enter three values of accuracy for the test on equilibrium, on loading condtions and on compatibilty. These values must be entered separated by a comma.

If the user enter just 1 value, this value is applied to the 3 required accuracies. If the use enter just 2 value, the first is applied to the accuracy on equilibrium, the second one is applied to the accuracu on loading conditions and on compatibility.

Examples:

```
precision=1.e-4,1.e-5,2.e-4
```

The accuracies on equilibrium is set to 10^{-4}. The accuracy on loading conditions is set to 10^{-5}. The accuracy on compatibility is set to $2\ 10^{-4}$.

```
precision=1.e-4
```

The accuracies on equilibrium, loading conditions and compatibilty are set to 10^{-4}.

```
precision=1.e-4,1.e-5
```

The accuracies on equilibrium is set to 10^{-4}. The accuracy on loading conditions is set to 10^{-5}. The accuracy on compatibility is set to 10^{-5}.

Two further experimental options can be added to `precision` option:

- an integer value specifying the test to be used on equilibrium:
  - 0: test based on the quadratic norm of the divergence of the stress field (equation 3.3) (default case),
  - 1: test based on the maximum value of the divergence of the stress field, in Fourier space (equation 3.4),
  - 2: criterion proposed by Monchiet & Bonnet (see 2),
  - 3: Bellis stopping criterion (see 3.6).
- an integer value specifying the maximum number of iterations allowed for solving the Lippmann-Schwinger equations (the iterative process stops when the number of iterations reaches this value).

Example:

```
precision=1.e-4,1.e-5,2.e-4,3,50
```

The accuracies on equilibrium is set to 10^{-4}. The accuracy on loading conditions is set to 10^{-5}. The accuracy on compatibility is set to $2\ 10^{-4}$. The equilibrium test is the one given in (3.6). The maximum number of iterations allowed is 50.

Important remark: choice of the adequate equilibrium

The choice of the equilibrium is delicate and must be made carefully. The author of these guide has a preference for the "Bellis" test and for the test based on the stress divergence, but he suggests to the user to make his or her own choice by testing different criteria and different required accuracy and, by comparison, to determine the criteria best suited to the problem considered.

4 Restoring a calculation

CraFT enables to define save/restore points from which it is possible to restart a calculation.

A save/restore point consists in a file in which CraFT stores, at a given time of the loading path, all what it needed to restart a calculation from that time to the end of the loading (in practice: all the mechanical variables describing the state of the material at that time).

Creating save/restore point file can be a concise way to store all the mechanical fields in a single file; using program `var2images` being a simple way to extract an image of a given field from a save/restore file.

4.1 Defining save/restore points

The user can define save/restore points at given times of the loading path using a command line in the *output specification file* with `variables` keyword. The syntax is very similar to the one to specify the storage of images or moments of mechanical fields:

- `variables=yes`

  will create a restore point file at the las time of the loading path.

- `variables=no`

  no save/restore point file will be created.

- `variables=yes <time specification>`

  will create save/restore point files at every times specified as argument.

Examples:

```
variables=yes 10.,20, 30.:40.:@2, 45.:@100, @200
```

creates save/restore point files at times: $t = 10s$, $t = 20s$, once at every two time steps between $t = 30s$ and $t = 40s$, at every time steps between $t = 45s$ and the 100^{th} steps, and at the 200^{th} step of the loading path.

```
variables=yes begin:end:@10
```

creates save/restore point files once at every ten time steps between the beginning and the end of the loading path.

4.2 Restoring a calculation

The user can restart a calculation using a save/restore point file. The calculation will restart from the time of the save/restore point till the end of the loading path.

The syntax of the command line is the same as the one for a normal run of CraFT, except that the option `-r` followed by the name of the save/restore file has to be used.

Example:

```
craft -n 8 -f ex03.in -r ex03_t=08.02000000e-01_variables.h5
```

runs a calculation specified if input file `ex03.in` (option: `-f ex03.in`), using 8 threads (option `-n 8`), from a save/restore point stored in file:

`ex03_t=08.02000000e-01_variables.h5`.

4.3 Restore file format

4.3.1 file format

The format of the save/restore files can be either a format specific to craft or a format using HDF5 (see `http://www.hdfgroup.org/HDF5/` for more details).

The format to be used depends on the way `craft` has been compiled:

- hdf5 format file is used if `craft` has been compiled with:

  `use_HDF5=yes`

  set in `options.in` file
- a format file specific to CraFT is used if it has been compiled with:

  `use_HDF5=no`

  in `options.in` file

4.3.2 extracting data from a save/restore file

`var2image` program enables to extraxt an image file of a given mechanical field from a save/restore file (whatevec is the save/restore file format).

Syntax:

```
var2image [-o imagefilenameprefix] variablefile field1 [field2 ... ]
```

extracts given field(s) from a given save/restore file, and stores the created images into image files (default output file names are herited from the name of the save/restore file).

Example:

```
var2image ex03_t=05.00000000e-01_variables.dat stress
```

will extract the stress field from save/restore file:

`ex03_t=05.00000000e-01_variables.dat`

and store it into an image file called:

`ex03_t=05.00000000e-01_variables_stress.vtk`

4.3.3 creating a save/restore file

Inversely, program `images2var` enables to create a save/restore file from images of the mechanical fields.

Syntax:

```
images2var -c cfn -o vfn -t time --field1 spec1 [--field2 spec2 [...]]
```

Extracts fields from images file and store the created Variables structure, with time stamp set to the given value, in `vfn file`.

Each field may be specified by an imagefile name:

```
--stress example_stress.vtk
```

or, in order to create an uniform field, by an explicit definition

```
--plastic "SCALAR 0."
--displacement "VECTOR3D 0. 1. 0."
--stress "TENSOR2 1. 2. 3. 4. 5. 6."
--backstress "VECTOR 4 0. 0. 0. 0."
  (with the mention to the number of components)
```

Please note that the quote are mandatory.

It must be mentioned that all necessary fields must be put into the save/restore file. Otherwise, `craft` will not be able to restart, and it will immediately fail with an error message.

5 Digital images

5.1 Generalities

A digital image is a set of physical points, called "pixels" in 2d and "voxels" in 3d (although the author of this document does not like this word and prefers to use "pixel" in 2d and in 3d), placed at the nodes of a regular grid of the space.

Thus, pixels are organized as a set of $n_1 \times n_2 \times n_3$ points, each pixel being separated from its previous neighbour along the k-th direction $(k = 1, 2, 3)$ by a given $\mathbf{p}_k$ vector.

Remark: a 2d image can be considered as a 3d image whose third direction has a 1 pixel depth ($n_3 = 1$).

Hence, the volume described in that way is a parallelepiped (and not necessarily a cube nor even a rectangular parallelepiped as it is usually defined) as $\mathbf{p}_k$ $(k = 1, 2, 3)$ vectors are not necessarily orthogonal nor having same magnitude.

With the definition of the position $\mathbf{s} = (s_1, s_2, s_3)$ of the first pixel in the list, the coordinate in the euclidian space $\mathbf{x} = (x_1, x_2, x_3)$ of each pixel can be got from its position in the digital image $\mathbf{i} = (i_1, i_2, i_3)$ (with $i_1 = 0, 1, ..., n_1 - 1$, $i_2 = 0, 1, ..., n_2 - 1$, $i_3 = 0, 1, ..., n_3 - 1$

$$\begin{aligned} \mathbf{x} &= \mathbf{s} + \textstyle\sum_{k=1,2,3} i_k \times \mathbf{p}_k \\ x_l &= s_l + \textstyle\sum_{k=1,2,3} i_k \times p_{kl} \quad (k = 1, 2, 3) \end{aligned}$$

(p_{kl} being the l-th component of $\mathbf{p}_k$ vector).

The data stored at each pixel could theoritically be of any kind: a scalar value, an integer value, a vector, a tensor, ...

In practice, it depends on the way images are implemented.

5.2 CraFT "i3d" format of images

CraFT proposes a file format which allow to store images whose pixels contain scalar values which can be: can only be scalars of type:

- signed 1-byte integer (`char`),
- unsigned 1-byte integer (`unsigned char`),

- signed integer (`int`),
- unsigned integer (`unsigned int`),
- floating point in simple precision (`float`)
- floating point in double precision (`double`)

A file in this format is a **binary** file (IEEE 754 arithmetic). It consists in a header (the size of which depends on the case) followed by all the pixel values in the order of increasing x_1, x_2 and x_3.

The header comprises:

- 10 bytes describing the type of pixels the image contains:

  `HM2RS` : floating values in single precision (coded in 4 bytes)

  `HM2RD` : floating values in double precision (coded in 8 bytes)

  `HM2RI` : integer values

  `HM2RUI` : unsigned integer values

  `HM2RC` : character values

  `HM2RUC` : unsigned character values

  `HMRS` : old (obsolete?) format for floating values in simple precision

- 20 bytes giving endianness of data values:

  `Big Endian` : data values are coded in big endian format

  `Little Endian` : data values are coded in little endian format

  Following data in the header are supposed to be coded following the endianness which has been declared here.

- header size (in bytes) : **only in old** `HMRS` **format** total size of the header
- n_1 n_2 n_3 : the number of pixels in the 3 directions, given as integer values coded in binary
- s_1 s_2 s_3 : the coordinates of the first pixels of the image, given as double precision real values coded in binary
- p_{11} p_{12} p_{13} p_{21} p_{22} p_{23} p_{31} p_{32} p_{33} : the 3 components of the step vectores along the 3 directions, given as double precision real values coded in binary

  (Caution: In the case of old `HMRS` format, step vectors are supposed to be orthogonal, and just p_{11} p_{22} p_{33} are to be written here). the 3 components of the step vectores along the 3 directions, given as double precision real values coded in binary

Thus, except in the case of `HMRS` old format, the header comprises 138 bytes.

The pixel values are stored one after each other from the "first pixel" (i.e. the pixel with coordinates $\mathbf{x} = (s_1, s_2, s_3),\ \mathbf{i} = (0,0,0)$) to the last, i_1 coordinate varying the fastest, and i_3 the slowest. In other words, pixels are stored in the following order: $\mathbf{i} = (0,0,0)$, $\mathbf{i} = (1,0,0)$, $\mathbf{i} = (2,0,0)$, .. $\mathbf{i} = (n_1 - 1, 0, 0)$,

$\mathbf{i} = (0,1,0)$, $\mathbf{i} = (1,1,0)$, $\mathbf{i} = (2,1,0)$, .. $\mathbf{i} = (n_1 - 1, 1, 0)$,

$\mathbf{i} = (0,2,0)$, $\mathbf{i} = (1,2,0)$, ... $\mathbf{i} = (n_1 - 1, 2, 0)$,

...

$\mathbf{i} = (n_1 - 1, n_2 - 1, 0)$, $\mathbf{i} = (0,0,1)$, $\mathbf{i} = (1,0,1)$, ... $\mathbf{i} = (n_1 - 1, n_2 - 1, n_3 - 1)$

What some people could call Fortran-like indexing...

5.3 Simple legacy VTK file format

5.3.1 generalities on VTK files

CraFT is able to read and write images formatted in simple legacy VTK format with "STRUCTURED_POINTS" dataset.

The main advantage of using VTK file format instead of Craft format is its much more common use. For example, images of this format can be visualized via well known 3D visualization programs such as Paraview and Mayavi2.

Full details on simple legacy VTK format can be found in:

`www.vtk.org/VTK/img/file-formats.pdf`

(this document being taken from the VTK User's Guide (published by Kitware Inc)).

In few words:

- VTK files consist in a human-readable header followed by a set of pixels,
- the pixels of a VTK file image are placed on a regular grid whose axes are **orthogonal**. In other words, the 3 vectors defined in 5.1 giving the spaces between two consecutive pixels in each direction satisfy:

$$\mathbf{p}_1 = (p_{11}, 0, 0),\ \mathbf{p}_2 = (0, p_{22}, 0),\ \mathbf{p}_3 = (0, 0, p_{33})$$

- pixels can be stored either in ASCII or in binary format,
- pixels are stored in the same order as in CraFT format: pixels are ordered with x_1 increasing fastest, then x_2, then x_3,
- CraFT supposes that the data in VTK files are represented in IEEE 754 floating point standard with **big-endian byte ordering** (although it is not clearly explained in VTK documentation, according to the opinion of the author of the present document).

5.3.2 type of data

The pixel values of VTK files can be of the following types:

- `char`,
- `unsigned char`,
- `int`,
- `unsigned int`,
- `long int`,
- `unsigned long int`,
- `float`,
- `double`.

Remark: The case of complex values has not been taken into account by VTK format, but images of `double complex` can be stored as images of `double` using two double values per complex value (one for the real part and a second one for the imaginary part).

The so-called "dataset attribute" in VTK files determine what set of data is stored in each pixel.

dataset attribute: "scalars"

When using "scalars" attribute, there can be one or several scalar values (of the same type) in each pixels. According to VTK documentation, there should not be more than 4 scalars per pixel, nevertheless CraFT proposes an extension of this and allows the use of a larger number of scalar values per pixels.

dataset attribute: "texture coordinates"

In the special cases of:

- symmetrical second order tensors,
- n-dimensional vectors,
- 2d-arrays,

CraFT can generate image files with VTK dataset attribute of type "texture coordinates", as this attribute allows to store multidimensional values. In the case of symmetrical second order tensors (typically for strain or stress tensors), the components are stored in the following order:

$$11,\ 22,\ 33,\ 13,\ 23,\ 12$$

Remark: Dataset attribute "tensors" would not have been appropriate as it only allows 3x3 values, which is not adequate for symmetrical 2d order tensor, of

which only 6 components need to be stored. Dataset attribute “scalars” allows a number of components greater than 1, but this number theorically must not be greater than 4 (although Paraview seems to accept a number of components greater than 4).

dataset attribute: “vectors”

In the special cases of:

- 3d vectors,

CraFT can generate image files with VTK dataset attribute of type “vectors”. This format allows to store 3-dimensional vectors.

Remarks on multi-component images

In the end, there are different ways to record an image with several scalar components per pixel/voxel:

→ using “scalars” dataset attribute with more than one scalar,

→ using “texture coordinates” dataset attribute with multiple components,

→ using “vectors” dataset attribute in the case of 3 dimensional data.

5.3.3 example

An example of a simple Legacy VTK image in ascii is given in 5.1

```
# vtk DataFile Version 3.0
craft output
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 32 32 1
ORIGIN 0.000000 0.000000 0.000000
SPACING  0.031250 0.031250 1.000000
POINT_DATA 1024
SCALARS scalars float
LOOKUP_TABLE default
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 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 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 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 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 2 2 2 2 2 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 2 2 2 2 2 2 2 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 2 2 2 2 2 2 2 2 2 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 2 2 2 2 2 2 2 2 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 2 2 2 2 2 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 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 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 0 0
```

Figure 5.1: An example of a simple Legacy VTK image in ascii. This image contains two disks which are almost visible in the text file (although the image being seen upside down).

5.4 Tools for handling and processing image files

Several companion programs are distributed with craft code for handling and processing VTK or i3d image files.

5.4.1 Basic mathematical operations on images

The code `ciop` allow to apply some basic operations on i3d ou VTK images. These operations are:

- addition,
- substraction,
- multiplication,
- division,
- raise to a power,
- test equal to,
- test greater than or equal to,
- test lower than or equal to,
- test greater than,
- test lower than.

The syntax is the following:

```
ciop image1-or-scalar1  operator image2-or-scalar2  [result-image]
```

where `operator` can be:

`+` : for addition,

`-` : for substraction,

`*` or `x`: for multiplication,

`/` : for division,

`^` or `**` : for raise to a power.

`==` : test equal to,

`>=` : test greater than or equal to,

`<=` : test lower than or equal to,

`>` : test greater than,

`<` : test lower than.

The arguments of the operation may be scalars or images, but cannot be both scalars.

In the case where the name of the resulting image is not specified, the standard output is used for output image.

In the case where the input arguments are both images, they must have the same number of components but can be of different data types, except in the case when one image except in the case where one of the images has several components per pixel/voxel and the other has only one. In this case, the component of the second image is replicated in as many times as the first image has components, the operation is then applied. For example, if an image contains a unique scalar per pixel, when it is multiplied by a 6 component image (an image containing 2d order tensors, for example), each component of each pixel of the second image is multiplied by the scalar contained in the corresponding pixel of the first image. This is a quite natural implementation of the multiplications tensors by scalars. A less trivial example occurs when the operation applied is en addition: each component of each pixel of the second image is **added** to the scalar contained in the corresponding pixel of the first image. This is a quite natural implementation of the multiplications tensors by scalars.

The output image is created with the type of higest rank of the two input images. The order of priority is (from lower to higher priority):

`char` → `unsigned char` → `int` → `unsigned int` → `long int` →

`unsigned long int` → `float` → `double` → `double complex`

Images with the data type `vector3d`, `tensor2` or `vector` are equivalent to `double` images and are considered with the same priority.

Caution: in order to avoid that the shell interprets the character `*`, it must be enclosed in quotation marks.

Examples:

1. `ciop hello.vtk + bonjour.vtk salut.vtk`

   The pixels of images `hello.vtk` and `bonjour.vtk` are added and stored in resulting image `salut.vtk`.

2. `ciop hello.vtk + bonjour.vtk > salut.vtk`

   does the same (but it uses the standard output as output, and redirect it to `salut.vtk` file).

3. `ciop 2. '*' a.vtk  res.vtk`

   The pixels of image `a.vtk` are multiplied by 2. and stored in image `res.vtk`.

4. `ciop a.vtk ^ 0.5 res.vtk`

The pixels of image `a.vtk` are raised to the power of 0.5 and stored in image `res.vtk`.

5.4.2 Splitting a multi-component image into multiple scalar images

A VTK image of a multicomponent field, i.e.

- 2d order tensor field,
- 3D vector field,
- field of multiple scalars,

can be splitted into VTK images of scalar fields , one per component, using: `vtk_split` program.

Syntax:

```
vtk_split [-f format] image
```

(`format` can be either `i3d` or `vtk`, depending on the format required for the output images).

5.4.3 Conversion between CraFT format and simple legacy VTK file format

Two programs are available with craft distribution to convert from one format to the other:

- `i3dtovtk` : to convert from so-called"i3d" CraFT format to VTK format (either in binary or in ASCII format)
- `vtktoi3d` : to convert from VTK format to so-called"i3d" CraFT format

Type `i3dtovtk -h` and `vtktoi3d -h` in a unix terminal to get more details on how to use these programs.

Remark: `i3dtovtk` can also be used for converting a vtk file into a vtk file, from ASCII format to binary format or vice-versa.

5.4.4 Conversion of a VTK image to a ppm file

- `vtk2ppm` : to convert an image in VTK format to an image in PPM format.

This is useful to get images ready to use for a publication or a web page.

PPM images can easily be transformed in any common file format using programs available on the internet.

Appendix A How to run CraFT

Following examples shows different ways to run craft for the following problem:

- microsctructure described by image file: `micro01.ima`
- phases in microstructure described by file:`micro01.phases`
- material(s) in microstructure described by file: `micro01.mat`
- loading conditions described by file: `traction.dat`
- required outputs described by file: `micro01.output`
- reference material C_0 chosen by CraFT
- required precision: $1.\ 10^{-4}$

A.1 Case 1: inputs are described by configuration file `micro01a.in` in "without keywords" format

CraFT can be run using `micro01a.in` configuration file:

```
#-----------------------------------------------
# HM 30/12/2010
# file: micro01a.in
#
#-----------------------------------------------
# image file describing the microstructure:
micro01.ima

# file describing the different phases in the
# microstructure:
micro01.phases

# file describing the mechanical behavior of
# the different components of the material:
micro01.mat

# file describing loading conditions:
traction.dat

# file describing the selected outputs:
micro01.output
```

```
# choice of C0:
auto

# required precision:
1.E-4
#-----------------------------------------------
```

by typing following command line:

```
craft -f micro01a.in
```

or, alternately, by typing:

```
craft < micro01a.in
```

A.2 Case 2: inputs are described by configuration file `micro01b.in` in "keywords format"

CraFT can be run using `micro01b.in` configuration file:

```
#-----------------------------------------------
# HM 30/12/2010
# file: micro01b.in
#
#-----------------------------------------------
# file describing the mechanical behavior of the
# different components of the material:
Materials=micro01.mat

# file describing the different phases in the
# microstructure:
Phases=micro01.phases

# image file describing the microstructure:
Microstructure=micro01.ima

# file describing loading conditions:
loading=traction.dat

# choice of C0:
C0=auto

# required precision:
precision = 1.E-4

# file describing the selected outputs:
```

```
output = micro01.output

#-----------------------------------------------
```

by typing following command line:

```
craft -f micro01b.in
```

A.3 Case 3: inputs are described by configuration file `micro01c.in` in "keywords format", loading and output specified directly in the input file (instead of being described by files

```
#-----------------------------------------------
# HM 30/12/2010
# input file micro01c.in
#-----------------------------------------------
# file describing the mechanical behavior of the
# different components of the material:
Materials=micro01.mat

# file describing the different phases:
Phases=micro01.phases

# image file describing the microstructure:
Microstructure=micro01.ima

# loading conditions:
loading {
S
1.  1 0 0 0 0 0 1.
}

# choice of C0:
C0=auto

# required precision:
precision = 1.E-4

# file describing the selected outputs:
output {
generic name=micro01c
stress image = yes
strain image = no
}
#-----------------------------------------------
```

A.4 Case 4: problem specifications described one by one in a command line

```
craft -c micro01.ima -p micro01.phases -m micro01.mat \
-l traction.dat -o micro01.output -C auto -e 1.E-4
```

Appendix B File describing materials in CraFT

The first line of every given material specification consists in the identifier of this material (it is a integer value defining uniquely a given material) followed by the number describing its behavior (see table 3.2).

B.1 How to describe a void material

Behavior identifier: 0

No parameters to be entered for void materials.

Example:

```
#-------------------------------------------------------
# Mat. #17 is a void material
17   0
#no further parameters are required
#-------------------------------------------------------
```

B.2 How to describe a pressurized cavity

Behavior identifier: 70

The user enters a pressure P, which means that a value of $-P\boldsymbol{I}$ is set inside the cavity, where $\boldsymbol{I}$ is the identity tensor.

The pression in the cavity may evolve. So the pression is given through a table containing the time and the pression. A linear interpolation is done between two values. For any times after the last value entered in the table the pression, the pressure is considered to be constant and egal to the last value entered. Before to enter the table, the number of lines of the table has to be given.

Example:

```
#-------------------------------------------------------
# Mat. #1 is a pressurized cavity
1   70
#size of the table
3
#table containing the time and the pressure
0.0 0.0
1.0 0.5
2.0 0.75
#-------------------------------------------------------
```

B.3 How to describe a linear elastic material

Behavior identifier: 10

A linear elastic (either anisotropic or isotropic) material is described in CraFT through its stiffness matrix. However the full specification of its 21 independant coefficients may be avoided if the material has some symmetry, simplifying the input arguments.

Specification

The material can be specified by an integer value amongst the different possible values:

0 : the full stiffness matrix has to be entered

1 : isotropic case

2 : cubic symmetry

3 : hexagonal symmetry

4 : orthotropic symmetry

Case 0: stiffness matrix entirely specified The stiffness matrix has to be entered by its upper triangular part using Kelvin notations. I.e. if the stress tensor $\boldsymbol{\sigma}$ is represented as a vector of 6 components: $(\sigma_i)_{1\leq i\leq 6}$ with:

$$\left\{\begin{array}{ll}\sigma_1 &= \sigma_{11}\\ \sigma_2 &= \sigma_{22}\\ \sigma_3 &= \sigma_{33}\end{array}\right. \quad \& \quad \left\{\begin{array}{ll}\sigma_4 &= \sqrt{2}\sigma_{23}\\ \sigma_5 &= \sqrt{2}\sigma_{13}\\ \sigma_6 &= \sqrt{2}\sigma_{12}\end{array}\right.$$

and if the strain tensor $\boldsymbol{\varepsilon}$ is represented by a 6 component vector:

$$\left\{\begin{array}{ll}\varepsilon_1 &= \varepsilon_{11}\\ \varepsilon_2 &= \varepsilon_{22}\\ \varepsilon_3 &= \varepsilon_{33}\end{array}\right. \quad \& \quad \left\{\begin{array}{ll}\varepsilon_4 &= \sqrt{2}\varepsilon_{23}\\ \varepsilon_5 &= \sqrt{2}\varepsilon_{13}\\ \varepsilon_6 &= \sqrt{2}\varepsilon_{12}\end{array}\right.$$

The stiffness tensor can be represented as the matrix C as follows:

$$\begin{pmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \sigma_4\\ \sigma_5\\ \sigma_6\end{pmatrix} = \begin{pmatrix}C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16}\\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26}\\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36}\\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46}\\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56}\\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66}\end{pmatrix} \cdot \begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\\ \varepsilon_5\\ \varepsilon_6\end{pmatrix}$$

or:

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sqrt{2}\sigma_{23} \\ \sqrt{2}\sigma_{13} \\ \sqrt{2}\sigma_{12} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{pmatrix} \cdot \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \sqrt{2}\varepsilon_{23} \\ \sqrt{2}\varepsilon_{13} \\ \sqrt{2}\varepsilon_{12} \end{pmatrix}$$

The upper triangular part of the matrix is entered into CraFT like that:

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

Examples:

```
#-------------------------------------------------------
# Mat. #15 is an anisotropic Linear Elastic material
15 10
# its stiffness matrix will be entered:
0
# stiffness matrix:
13930.   7082   5765     0.     0.     0.
       13930.  5765     0.     0.     0.
              15010.    0.     0.     0.
                     6028.     0.     0.
                            6028.     0.
                                   6828.
#-------------------------------------------------------
```

Case 1: isotropic case In this case, the behavior of material is supposed to be isotropic linear elasticity. The user has to enter:

- the Young's modulus: E
- the Poisson coefficient: ν

The stiffness matrix is calculated as:

$$\begin{pmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{pmatrix}$$

where λ and μ are the Lamé coefficients which are related to the input parameters by

$$\mu = \frac{E}{2(1+\nu)} \text{ and } \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \,. \tag{B.1}$$

Example:

```
#-----------------------------------------------------------
# Mat. #32 is an linear elastic material
32   10
# ... and it is isotropic:
1
# Young's modulus:
10.
# Poisson coefficient:
0.23
#-----------------------------------------------------------
```

Case 2: cubic symmetry In the case cubic symmetry, the user has to enter the following parameters:

- bulk modulus K
- μ_1
- μ_2

the stiffness matrix being then calculated as:

$$\begin{pmatrix} (3K+4\mu_1)/3 & (3K-2\mu_1)/3 & (3K-2\mu_1)/3 & 0 & 0 & 0 \\ (3K-2\mu_1)/3 & (3K+4\mu_1)/3 & (3K-2\mu_1)/3 & 0 & 0 & 0 \\ (3K-2\mu_1)/3 & (3K-2\mu_1)/3 & (3K+4\mu_1)/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu_2 \end{pmatrix}$$

Case 3: hexagonal symmetry In the case hexagonal symmetry, the user has to enter the following parameters:

- bulk modulus K
- μ_t
- μ_l
- E_l
- ν_l

the stiffness matrix being then calculated as:

$$\begin{pmatrix} K+\mu_t & K-\mu_t & 2\nu_l K & 0 & 0 & 0 \\ K-\mu_t & K+\mu_t & 2\nu_l K & 0 & 0 & 0 \\ 2\nu_l K & 2\nu_l K & E_l + 4\nu_l^2 K & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu_l & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu_l & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu_t \end{pmatrix}$$

Case 4: orthotropic symmetry In the case of orthotropic symmetry, the user has to enter the following parameters:

- 3 Young'moduli: E_1, E_2, E_3
- 3 Poisson coefficients: ν_{12}, ν_{13}, ν_{23}
- 3 shear moduli: μ_{12}, μ_{13}, μ_{23}

the stiffness matrix being then calculated as

$$\begin{pmatrix} E_1(1-\nu_{23}\nu_{32})/k & E_1(\nu_{23}\nu_{31}+\nu_{21})/k & E_1(\nu_{21}\nu_{32}+\nu_{31})/k & 0 & 0 & 0 \\ E_2(\nu_{13}\nu_{32}+\nu_{12})/k & E_2(1-\nu_{13}\nu_{31})/k & E_2(\nu_{12}\nu_{31}+\nu_{32})/k & 0 & 0 & 0 \\ E_3(\nu_{12}\nu_{23}+\nu_{13})/k & E_3(\nu_{13}\nu_{21}+\nu_{23})/k & E_3(1-\nu_{12}\nu_{21})/k & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu_{13} & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu_{12} \end{pmatrix}$$

with:

$$k = 1 - \nu_{23}\nu_{32} - \nu_{12}\nu_{21} - \nu_{13}\nu_{31} - \nu_{12}\nu_{23}\nu_{31} - \nu_{21}\nu_{32}\nu_{13}$$
$$\nu_{21} = \nu_{12}E_2/E_1, \quad \nu_{32} = \nu_{23}E_3/E_2, \quad \nu_{31} = \nu_{13}E_3/E_1.$$

Implementation

Despite the apparent simplification of the stiffness matrix, the orientation of the phases may lead to a dense matrix for non trivial Euler angles, so that all but isotropic case are handled but applying the full matrix multiplication $\sigma_i = C_{ij}\varepsilon_j$, after rotation of the stiffness matrix according to the orientation of the phase.

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor

B.4 How to describe an elastic-perfectly-plastic von Mises material

Behavior identifier: 2

The case of an elastic-perfectly-plastic material with von Mises yield criterion (the linear elastic part is assumed to be isotropic) is governed by the equations

$$\begin{aligned} \sigma &= L : (\varepsilon - \varepsilon_p) \\ \dot{\varepsilon}_p &= \frac{3}{2}\dot{p}\frac{\sigma^d}{\sigma^{eq}} \text{ with } \dot{p} = \sqrt{\frac{2}{3}\dot{\varepsilon}_p : \dot{\varepsilon}_p} \\ \sigma^{eq} &\leq \sigma_0 \end{aligned} \tag{B.2}$$

where L is the isotropic linear elastic stiffness tensor (defined by E and ν), ε_p is the (deviatoric) plastic strain, $\dot{\varepsilon}_p$ its time derivative, σ^d and σ^{eq} are the deviatoric part of the stress and the von Mises stress, and p denotes the hardening parameter (that coincides with the cumulated plastic strain), respectively.

Specification

The user has thus to enter the following parameters:

- Young's modulus E
- Poisson coefficient ν
- Yield stress σ_0

Example:

```
#-------------------------------------------------------
# Mat. #8 is an elastic-perfectly-plastic material
8   2
# Young's modulus:
10.
# Poisson coefficient:
0.23
# Yield stress:
1.2
#-------------------------------------------------------
```

Implementation

The algorithm, given in full details in [1] (Appendix C) , is recalled here. The von Mises yield criterion suggests that the plastic strain only increases when the von Mises stress reaches a critical value known as the yield stress. The first step is then to check whether, for a varying load, the change in stress is only due to a change of the elastic strain or whether the plastic strain has increased. Using a backward

finite difference scheme at time step $n+1$, Eq. (B.8) leads to

$$\boldsymbol{\sigma_{n+1}} = \boldsymbol{\sigma_n} + \boldsymbol{L} : (\boldsymbol{\varepsilon_{n+1}} - \boldsymbol{\varepsilon_n}) - 2\mu\boldsymbol{\varepsilon}_{p(\dot{n+1})}\Delta t = \boldsymbol{\sigma_T} - 2\mu\Delta t\boldsymbol{\varepsilon}_{p(\dot{n+1})} \tag{B.3}$$

where σ_T denotes the elastic trial stress:

$$\boldsymbol{\sigma_T} = \boldsymbol{\sigma_n} + L : (\boldsymbol{\varepsilon_{n+1}} - \boldsymbol{\varepsilon_n}) \tag{B.4}$$

If this latter is within the yield surface ($\sigma_T^{eq} < \sigma_0$), the deformation is elastic and $\sigma_{n+1} = \sigma_T$. If not, Eq. (B.3) expands to

$$\sigma^d_{n+1} = \sigma^d_T - 2\mu\dot{\varepsilon}_{p(n+1)}\Delta t = \sigma^d_T - 3\mu\dot{p}\Delta t\frac{\sigma^d_{n+1}}{\sigma^{eq}_{n+1}} \tag{B.5}$$

$$\sigma^d_{n+1}\left(1 + 3\mu\dot{p}\Delta t/\sigma^{eq}_{n+1}\right) = \sigma^d_T \tag{B.6}$$

i.e. σ^d_{n+1} is colinear to σ_T, enabling the use of the radial return algorithm. While yielding (without hardening), the von Mises stress is limited to the yield stress, so $\sigma^d_{n+1} = (\sigma_0/\sigma^{eq}_T)\sigma_T$. The last step is to restore the hydrostatic component

$$\sigma_{n+1} = \sigma^d_{n+1} + \frac{1}{3}\operatorname{tr}(\sigma_{n+1})\mathbb{I} \text{ with } \operatorname{tr}(\sigma_{n+1}) = (3\lambda + 2\mu)\operatorname{tr}(\varepsilon_{n+1}). \tag{B.7}$$

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor
`previous stress`	stress tensor of the previous step
`previous strain`	strain tensor of the previous step

[1] H. Moulinec and P. Suquet, A numerical method for computing the overall response of nonlinear composites with complex microstructure, Comp. Meth. Appl. Eng. 157 (1998), pp. 69-94.

B.5 How to describe an elastic-plastic von Mises material

Behavior identifier: 4

The case of an elastic-plastic material with von Mises yield criterion and linear isotropic hardening is derived from the previous case, but with a yield surface expanding during the plastic flow:

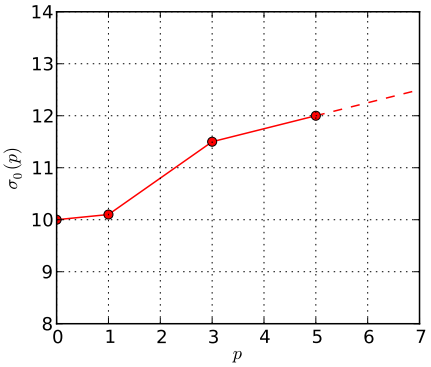
$$\begin{aligned} \sigma &= L : (\varepsilon - \varepsilon_p) \\ \dot{\varepsilon}_p &= \frac{3}{2}\dot{p}\frac{\sigma^d}{\sigma^{eq}} \text{ with } \dot{p} = \sqrt{\frac{2}{3}\dot{\varepsilon}_p : \dot{\varepsilon}_p} \\ \sigma^{eq} &\leq \sigma_0(p) \end{aligned} \tag{B.8}$$

Specification

The user has to enter the following parameters:

- Young's modulus
- Poisson coefficient
- A flag telling the hardening type:
  - 0: without hardening $\sigma_0(p) = \sigma_y$ (redundant with Sec. B.4). The user has then to enter the yield stress σ_y.
  - 1: with linear hardening $\sigma_0(p) = \sigma_y + Hp$. The yield stress σ_y and then the plastic modulus H have to be provided.
  - 2: with tabulated values of the hardening. The dependance of the isotropic hardening σ_0 with respect to the cumulated plastic strain p is specified through a table. The user has to give the number of points of the table, then a list of pairs of points defining the isotropic hardening criterion, for example:

```
# Number of points
4
# Pairs of points
# (p, sigma0)
0. 10.0
1. 10.1
3. 11.5
5. 12.0
```



The extrapolation beyond the last pair is based on the last segment.

Example:

```
#-------------------------------------------------------
```

```
# Mat. #8 is an elastic-plastic material
# with linear hardening
8   6   1
# Young's modulus:
10.
# Poisson coefficient:
0.23
# Yield stress:
10
# Plastic modulus:
0.3
#-------------------------------------------------------
```

Implementation

The algorithm is quite the same as the one of the elastic-perfectly plastic von Mises material, except that it is necessary to track the evolution of the yield surface using the cumulated plastic strain p. In fact the von Mises stress σ_{n+1}^{eq} is not bounded to a static value anymore, and the explicit value of p has to be calculed.

The trial von Mises stress σ_T^{eq} has then to be compared with $\sigma_0(p_n)$. If smaller, the yield surface is not reached and then does not grow: $p_{n+1} = p_n$ and $\sigma_{n+1} = \sigma_T$. If bigger, the increase of the plastic strain and the associated grow of the yield surface (null in the perfectly plastic case) have to be computed solving the implicit equation

$$\sigma_0(p_{n+1}) + 3\mu p_{n+1} = \sigma_T^{eq} + 3\mu p_n \tag{B.9}$$

according to the specified hardening law $\sigma_0(p)$. The determination of p_{n+1} allows to apply the radial return procedure

$$\sigma_{n+1}^d = \left(1 - 3\mu\Delta t \left(p_{n+1} - p_n\right)\right)\sigma_T^d \tag{B.10}$$

and then restore the hydrostatic component (same as in Sec. B.4).

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor
`plastic`	cumulated plastic strain (*)
`previous stress`	stress tensor of the previous step
`previous strain`	strain tensor of the previous step
`previous plastic`	cumulated plastic strain of the previous loading step (*)

(*): available only in presence of hardening but not in perfectly plastic case.

B.6 How to describe a power law Elastic Visco-Plastic material

Behavior identifier: 40

A power law Elastic Visco-Plastic material (the linear elastic part being supposed to be isotropic) is subjected to the equations

$$\begin{aligned} \sigma &= L : (\varepsilon - \varepsilon^{vp}), \\ \dot{\varepsilon}^{vp} &= \dot{\varepsilon}^0 \, \frac{3}{2} \left(\frac{\sigma^{eq}}{\sigma^0} \right)^N \frac{\sigma^d}{\sigma^{eq}}. \end{aligned}$$

In the following, we will consider that $\dot{\varepsilon}^0$ is equal to 1 , or, equivalently, that it has been incorporated into σ^0. Thus, the constitutive relation reads:

$$\begin{aligned} \sigma &= L : (\varepsilon - \varepsilon^{vp}), \\ \dot{\varepsilon}^{vp} &= \frac{3}{2} \left(\frac{\sigma^{eq}}{\sigma^0} \right)^N \frac{\sigma^d}{\sigma^{eq}}. \end{aligned} \tag{B.11}$$

Specification

the user has to enter the following parameters:

- Young's modulus E,
- Poisson coefficient ν,
- Yield stress σ^0
- exponent of the power N.

Example:

```
#-----------------------------------------------------------
# Mat. #86 is a power law Elastic Visco-Plastic material
86   40
# Young's modulus:
10.
# Poisson coefficient:
0.23
# Yield stress:
1.2
# exponent of the power law:
1.1
#-----------------------------------------------------------
```

Implementation

Denoting $\boldsymbol{\sigma_T} = \boldsymbol{\sigma_{n-1}} + L : (\boldsymbol{\varepsilon_n} - \boldsymbol{\varepsilon_{n-1}})$ the trial stress and s_T its deviatoric component, the use of a backward finite difference scheme at time n in Eq. (B.11) reads as

$$\frac{\boldsymbol{\sigma}_n - \boldsymbol{\sigma}_{n-1}}{\Delta t} = \boldsymbol{L} : \left(\frac{\boldsymbol{\varepsilon}_n - \boldsymbol{\varepsilon}_{n-1}}{\Delta t} \right) - \boldsymbol{L} : \dot{\varepsilon}_n^{vp} \tag{B.12}$$

which leads to

$$\sigma_n^d = s_T - 2\mu\Delta t\dot{\varepsilon}_n^{vp} = s_T - 3\mu\Delta t \left(\frac{\sigma_n^{eq}}{\sigma_n^0} \right)^N \frac{\sigma_n^d}{\sigma_n^{eq}}. \tag{B.13}$$

If the trial stress is hydrostatic (s_T^{eq}), the same applies to σ_n; otherwise the resolution of

$$\sigma_n^{eq} + 3\frac{\mu\Delta t}{\sigma_n^{0\,N}} \left(\sigma_n^{eq}\right)^N = s_T^{eq} \tag{B.14}$$

is performed using the Müller method[1] and gives σ_n^{eq}, the radial return method applies ($\sigma_n^d = (\sigma_n^{eq}/s_T^{eq}) s_T^d$). In either case, the hydrostatic part has to be restored (same as Eq. (B.7)).

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor
`previous stress`	stress tensor of the previous step
`previous strain`	strain tensor of the previous step

[1] D. Müller, A Method for Solving Algebraic Equations Using an Automatic Computer, Math. Tables Aids Comp. 10 No.56 (1956), pp. 208-215.

B.7 How to describe a power law Elastic Visco-Plastic material with kinematic linear hardening

Behavior identifier: 41

The case of a power law Elastic Visco-Plastic material with kinematic linear hardening whose constitutive law is described by:

$$\begin{aligned} \sigma &= L : (\varepsilon - \varepsilon^{vp}), \\ \dot{\varepsilon}^{vp} &= \frac{3}{2}\dot{p}\frac{(\sigma - X)^d}{(\sigma - X)^{eq}}, \text{ with } \dot{p} = \left\langle \frac{(\sigma - X)^{eq} - \sigma_l}{D} \right\rangle^N \\ X &= \frac{2}{3}H\varepsilon^{vp}. \end{aligned} \tag{B.15}$$

where $\langle . \rangle$ denotes the Macauley bracket and X is the backstress tensor (that is used to describe the kinematic hardening). An interesting review is given in [1]. The linear elastic part being supposed to be isotropic.

Specification

The user has to enter the following parameters in that order:

- Young's modulus E
- Poisson coefficient ν
- drag stress D
- exponent of the power law n
- linear hardening coefficient H
- yield stress σ_l

Example:

```
#-------------------------------------------------------
# Mat. #84 is a power law Elastic Visco-Plastic
# material with kinematic linear hardening
84   41
# Young's modulus and Poisson coefficient
5500. 0.33
# Drag stress
25.
# exponent of the power law:
7.
# linear hardening coefficient:
```

[1] J.L. Chaboche, A review of some plasticity and viscoplasticity constitutive theories, Int. J. Plasticity 24 No.10 (2008), pp. 1642-1693.

```
2200.
# Yield stress:
50.
#-------------------------------------------------------------
```

Implementation

The yielding is triggered according to the value of the tensor field $Y = \sigma - X$. The backward finite difference scheme at time n in Eq. (B.15) leads to

$$Y = L : \varepsilon - 2\mu\varepsilon^{vp} - X = L : \varepsilon - \frac{2}{3}(H + 3\mu)\varepsilon^{vp} \Rightarrow Y_n = \sigma_T - \frac{2}{3}(H + 3\mu)\Delta t\dot{\varepsilon}^{vp}$$

with the trial stress:

$$\sigma_T = \sigma_{n-1} - X_{n-1} + L : (\varepsilon_n - \varepsilon_{n-1}) \tag{B.16}$$

Expanding the viscoplastic strain, it appears that Y_n^d is colinear to the deviatoric part of the trial stress with

$$Y_n^{eq} + (H + 3\mu)\Delta t \left\langle \frac{Y_n^{eq} - \sigma_l}{D} \right\rangle^N = \sigma_T^{eq}. \tag{B.17}$$

The radial return method applies to Y_n^d. It should be noted that the storage of the backstress X can be avoided as $X = H/(3\mu)\left(2\mu\varepsilon^d - \sigma^d\right)$, so that

$$\sigma_n^d = \frac{3\mu}{H + 3\mu}\left(Y_n^d + \frac{2H}{3}\varepsilon_n^d\right), \tag{B.18}$$

and the last step is to restore the hydrostatic part (same as Eq. (B.7)).

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor
`previous stress`	stress tensor of the previous step
`previous strain`	strain tensor of the previous step

B.8 How to describe an elastic-plastic Gurson material

Behavior identifier: 60

The case of an elastic-perfectly-plastic material with Gurson yield criterion (the linear elastic part is assumed to be isotropic) is governed by the equations (were f is the porosity and P_b the pressure inside the cavities):

$$\begin{aligned} F(\sigma) &= q_3 \left(\frac{\sigma_{eq}}{\sigma_0}\right)^2 + 2q_1 f cosh\left(\frac{3\sigma_m}{2\sigma_0}\right) - 1 - (q_1 f)^2 \\ \sigma &= L : (\varepsilon - \varepsilon_p) \end{aligned} \tag{B.19}$$

Specification

The user has to enter the following parameters:

- Young's modulus
- Poisson coefficient
- Yield stress (σ_0)
- q_3 coefficient
- q_1 coefficient
- A flag telling the choice for the porosity evolution type:

  1: with no porosity evolution. The user has to enter the porosity value f, which will be kept constant during the whole calculation.

  2: with linear porosity evolution. The user has to enter the initial porosity value f_0 and the porosity slop f_1, the porosity f evolve linearly with the time $f = f_0 + f_1 t$.

  3: with tabulated values of the porosity. The evolution of porosity with the time is specified through a table. The user has to give the number of points of the table, then a list of pairs of points defining the porosity evolution with the time. Linear extrapolation is done between two points.

  5: with calculated values of the porosity. The evolution of porosity is linked to the trace of the plastic strain through the equation : $\dot{f} = (1 - f)\dot{\varepsilon}_{p,m}$. The user has to give the initial value of porosity.

- A flag telling the choice for the pressure evolution type:

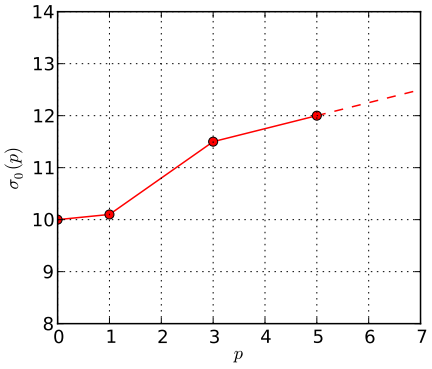
  0: without pressure $P_b = 0$

  1: with no pressure evolution. The user has to enter the pressure value P_b, which will be kept constant during the whole calculation.

2: with linear pressure evolution. The user has to enter the initial pressure value P_{b0} and the pressure slop P_{b1}, the pressure P_b evolve linearly with the time $P_b = P_{b0} + P_{b1}t$.

3: with tabulated values of the pressure. The evolution of pressure with the time is specified through a table. The user has to give the number of points of the table, then a list of pairs of points defining the pressure evolution with the time. Linear extrapolation is done between two points. for example:

```
# Number of points
4
# Pairs of points
# (t Pb)
0. 10.0
1. 10.1
3. 11.5
5. 12.0
```



The extrapolation beyond the last pair is based on the last segment.

Example:

```
#-----------------------------------------------------------
 material 0 : Gurson type plasticity:
0 60
# Young's Modulus
10.0
# Poisson coefficient
0.2
# limit stress
1.0
# coef q3
1.0
# coef q1
1.0
# porosity evolution model
5
#initial porosity
0.045
# bubble pressure evolution model : 1: pressure(t)=pressure(t=0)
1
#
#initial bubble pressure
0.0
#-----------------------------------------------------------
```

Implementation

An effective porosity is defined as $f_2 = q_1 f$.

If plasticity is activated (i.e. if $F(\sigma_T) > 0$, where σ_T is the trial stress) :

$$\dot{\sigma} = L : (\dot{\varepsilon} - \dot{\varepsilon}_p) = L : \left(\dot{\varepsilon} - \dot{\lambda}\frac{\partial F}{\partial \sigma}\right)$$

This equation is then split into a deviatoric and a hydrostatic part.

The deviatoric part of the equation enables the use of radial return :

$$\sigma_D \left(1 + \frac{6q_3\mu\dot{\lambda}\delta t}{\sigma_0^2}\right) = \sigma_{D,T} \tag{B.20}$$

The hydrostatic part enables to write $\dot{\lambda}\delta t$ as a function of σ_m (the dash t_0 indicates the value of the preceding time step):

$$\sigma_m - \sigma_m^{t_0} = 3k\delta t\dot{\varepsilon}_m - \frac{9k\dot{\lambda}f_2\delta t}{\sigma_0} sinh\left(\frac{3\sigma_m}{2\sigma_0}\right) \tag{B.21}$$

In the case of a fixed porosity, the following equation, where σ_m is the only unknown, is solved with a Müller algorithm :

$$F(\sigma) = \frac{3q_3}{2}\left(\frac{\sigma_{D,T} : \sigma_{D,T}}{\sigma_0^2}\right)\left(\frac{\sigma_0^2}{\sigma_0^2 + 6q_3\mu\left(\frac{\sigma_0(3k(\varepsilon_m - \varepsilon_m^{t_0}) - (\sigma_m - \sigma_m^{t_0}))}{sinh(\frac{3\sigma_m}{2\sigma_0})9kf_2}\right)}\right)^2 \tag{B.22}$$

$$+2f_2 cosh\left(\frac{3\sigma_m}{2\sigma_0}\right) - 1 - f_2^2 = 0$$

Plasticity dependent porosity

For plasticity dependent porosity, the porosity evolution is considered as an strain-hardening (or softening) mechanisms, which reduces (or expands) the plasticity surface. A new equation has to be introduced in order to identify the new porosity : the condition for the associated plastic flow is used :

$$\frac{\partial F}{\partial \sigma} : \dot{\sigma} + \frac{\partial F}{\partial f} : \dot{f} = 0 = \frac{3f}{\sigma_0} sinh\left(\frac{3\sigma_m}{2\sigma_0}\right)\dot{\sigma}_m + \frac{3\sigma_D}{\sigma_0^2} : \dot{\sigma}_D + \left(2cosh\left(\frac{3\sigma_m}{2\sigma_0}\right) - 2f\right)\dot{f} \tag{B.23}$$

Then, the system composed by the equations B.22 and B.23, with two unknowns, f and σ_m is solved through a Newton algorithm.

List of variables available for image storing

keywords	description
`stress`	stress tensor
`strain`	strain tensor
`cumulated plastic strain`	cumulated plastic strain
`previous stress`	stress tensor of the previous step
`previous strain`	strain tensor of the previous step
`porosity`	porosity (*)
`previous_porosity`	porosity of the previous loading step (*)

(*) : available only if porosity evolution model is 5.

B.9 How to describe a Voce law

Behavior identifier: 50

The Voce constitutive law implemented in CraFT can be described by:

$$\boldsymbol{\sigma} = \boldsymbol{L} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon^{vp}}), \quad \boldsymbol{\varepsilon^{vp}} = \sum_{k=1}^{M} \gamma^{(k)} \boldsymbol{\mu}^{(k)}, \quad \boldsymbol{\mu}^{(k)} = \boldsymbol{m}^{(k)} \otimes_s \boldsymbol{n}^{(k)} \tag{B.24}$$

where M is the number of slip systems, $\boldsymbol{\mu}^{(k)}$ is the Schmid tensor of the k-th slip system with slip plane normal $\boldsymbol{n}^{(k)}$ and slip direction $\boldsymbol{m}^{(k)}$ (which is orthogonal to $\boldsymbol{n}^{(k)}$), and $\otimes_s$ indicates the (symmetrical) dyadic product.

The slip-rate $\dot{\gamma}^{(k)}$ on the k-th system is related to the resolved shear stress $\tau^{(k)}$ on that system through:

$$\dot{\gamma}^{(k)} = \dot{\gamma}_0^{(k)} \left(\frac{\left|\tau^{(k)}\right|}{\tau_0^{(k)}} \right)^{n^{(k)}} \operatorname{sgn}\left(\tau^{(k)}\right), \quad \tau^{(k)} = \boldsymbol{\sigma} : \boldsymbol{\mu}^{(k)} \tag{B.25}$$

where $\tau_0^{(k)}$, the reference resolved shear stress on system k, depends on the activity of the other systems through:

$$\dot{\tau}_0^{(k)} = \left[\theta_{sta}^{(k)} + e^{-\Gamma a} \left(\theta_{ini}^{(k)} + a\theta_{sta}^{(k)} \, \Gamma - \theta_{sta}^{(k)} \right) \right] \dot{p}, \quad \dot{p} = \sum_{\ell=1}^{M} h^{(k,\ell)} \ \left|\dot{\gamma}^{(\ell)}\right|, \tag{B.26}$$

with a defined by:

$$a^{(k)} = \left| \frac{\theta_{ini}^{(k)}}{\tau_{sta}^{(k)} - \tau_{ini}^{(k)}} \right| \quad .$$

Γ is a scalar variable which cumulates the increments of all slip systems.

$$\dot{\Gamma} = \sum_{k=1}^{M} \left|\dot{\gamma}^{(k)}\right|$$

When all compontents of matrix h are equal to 1, Γ coincides to the term $\dot{p} = \sum_{\ell=1}^{M} h^{(k,\ell)} \ \left|\dot{\gamma}^{(\ell)}\right|$ appearing in the right member of equation (B.26).

Implementation

The constitutive relation can be formulated as a differential equation:

$$\dot{\boldsymbol{Y}} = \boldsymbol{F}(\boldsymbol{Y}, t)$$

where:

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{\sigma} \\ \Gamma \\ \tau_0^{(k)}\ (k=1,...M) \end{pmatrix}$$

and:

$$\boldsymbol{F}(\boldsymbol{Y},t) = \begin{pmatrix} \boldsymbol{L} : \left(\dot{\boldsymbol{\varepsilon}} - \sum_{k=1}^{M} \dot{\gamma}^{(k)}(\boldsymbol{Y}) \boldsymbol{\mu}^{(k)} \right) \\ \sum_{k=1}^{M} \left| \dot{\gamma}^{(k)}(\boldsymbol{Y}) \right| \\ \left[\theta_{sta}^{(k)} + e^{-\Gamma(\boldsymbol{Y})a} \left(\theta_{ini}^{(k)} + a\theta_{sta}^{(k)}\, \Gamma(\boldsymbol{Y}) - \theta_{sta}^{(k)} \right) \right] \quad \sum_{\ell=1}^{M} h^{(k,\ell)}\ \left| \dot{\gamma}^{(\ell)}(\boldsymbol{Y}) \right| \end{pmatrix}$$

where $\dot{\gamma}^{(k)}$ is given by relation B.26 .

This system is solved in CraFT by a Runge-Kutta method with fixed step size.

Specification

The user has to enter the following parameters in that order:

- elastic parameters, which has to be entered in the same manner as in the linear elastic case (see B.3)
- the visco-plastic parameters, i.e.:
  - the number M of slip systems,
  - for each slip system:
    - the vector $\boldsymbol{n}$ orthogonal to the plane of slip (3 coordinates),
    - the vector $\boldsymbol{m}$ of the direction of slip in the plane (3 coordinates)
  - for each slip system k:
    - $\tau_{ini}^{(k)}$
    - $\tau_{sta}^{(k)}$
    - $\theta_{ini}^{(k)}$
    - $\theta_{sta}^{(k)}$
    - $\dot{\gamma}_0^{(k)}$
    - power of the law: $n^{(k)}$
  - the matrix $h^{(k,\ell)}$ for $k = 1, ..., M$ and $\ell = 1, ..., M$

Example:

```
###############################################
# Copper material
#
###############################################
# material id: 50 = Voce law
1 50
#---------------------------------------------
# Elastic behavior:
#   cubic symmetry:
2
#
133336.666667
27625.
60980.
#---------------------------------------------
# number of slip systems
12
#
# n(i,1) n(i,2) n(i,3)   m(i,1) m(i,2) m(i,3)  i=1,nsyst
   1    1   -1      0    1    1
   1    1   -1      1    0    1
   1    1   -1      1   -1    0
   1   -1   -1      0    1   -1
   1   -1   -1      1    0    1
   1   -1   -1      1    1    0
   1   -1    1      0    1    1
   1   -1    1      1    0   -1
   1   -1    1      1    1    0
   1    1    1      0    1   -1
   1    1    1      1    0   -1
   1    1    1      1   -1    0
#
#---------------------------------------------
# visco-plastic parameters:
#- - - - - - - - - - - - - - - - - - - - - - -
# system #1:
#- - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - -
# system #2:
```

```
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# system #3:
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# system #4:
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# system #5:
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# system #6:
#- - - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
```

```
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - -
# system #7:
#- - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - -
# system #8:
#- - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - -
# system #9:
#- - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - - - -
# system #10:
#- - - - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
```

```
10.
#- - - - - - - - - - - - - - - - - - - - - - -
# system #11:
#- - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#- - - - - - - - - - - - - - - - - - - - - - -
# system #12:
#- - - - - - - - - - - - - - - - - - - - - - -
# tau_ini tau_sta:
10. 15.
# theta_ini theta_sta:
1000. 500.
# gamma_dot_0:
1.
# npower:
10.
#-----------------------------------------------
# H matrix:
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
###############################################
```

B.10 How to describe the behavior of UO2

Behavior identifier: 2021

B.10.1 Constitutive equations of UO2

The constitutive equations of UO2 are supposed to be

$$\begin{cases} \dot{\boldsymbol{\sigma}} = \boldsymbol{C} : (\dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}}^{vp}) \\ \dot{\boldsymbol{\varepsilon}}^{vp} = \sum_s \dot{\gamma}^s \boldsymbol{\mu}^s \\ \dot{\gamma}^s = \dot{\gamma}_0^s \ \exp\Big(- \dfrac{\Delta H_0^s}{k_0 T}\Big) \left(ch\Big(\dfrac{\tau_s}{\tau_0^s}\Big) - 1 \right) \text{sign}(\tau_s) \\ \tau_s = \boldsymbol{\mu}_s : \boldsymbol{\sigma} \\ \boldsymbol{\mu}_s = \dfrac{1}{2}(\boldsymbol{m}_s \otimes \boldsymbol{n_s} + \boldsymbol{n}_s \otimes \boldsymbol{m_s}) \end{cases} \tag{B.27}$$

with s being the index of the slip systems.

B.10.2 "Behavior" function

For each behaviour law, CraFT expects a function called "behavior" that returns the stress and takes as arguments the deformation and possible other local mechanical variables. Moreover CraFT imposes that the strain of the previous loading step and the increment of time between the previous and the current loading steps as arguments of behavior function. This can be noted as follows

$$\boldsymbol{\sigma}(t) = \mathcal{F}(\boldsymbol{\varepsilon}(t), \boldsymbol{\varepsilon}(t - \delta t), \delta t, ...) \tag{B.28}$$

In the case of UO2 behavior, equations B.27 must be reorganized in order to be presented as required by CraFT in B.28.

The stress rate is discretized in time as

$$\dot{\boldsymbol{\sigma}} = \frac{\boldsymbol{\sigma}(t) - \boldsymbol{\sigma}(t - \delta t)}{\delta t} \tag{B.29}$$

Thus

$$\begin{aligned} \boldsymbol{\sigma}(t) &= \boldsymbol{\sigma}(t - \delta t) + \delta t \boldsymbol{C} : \dot{\boldsymbol{\varepsilon}} - \delta t \boldsymbol{C} : \dot{\boldsymbol{\varepsilon}}^{vp} \\ &= \boldsymbol{\sigma}^T - \delta t \boldsymbol{C} : \sum_s \left[\dot{\gamma}_0^s \exp\Big(- \frac{\Delta H_0^s}{k_0 T}\Big) \left(ch\Big(\frac{\tau_s}{\tau_0^s}\Big) - 1 \right) \text{sign}(\tau_s) \boldsymbol{\mu}_s \right] \\ &= \boldsymbol{\sigma}^T - \delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(- \frac{\Delta H_0^s}{k_0 T}\Big) \left(ch\Big(\frac{\boldsymbol{\mu}_s : \boldsymbol{\sigma}}{\tau_0^s}\Big) - 1 \right) \text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma}) \, \boldsymbol{C} : \boldsymbol{\mu}_s \right] \end{aligned} \tag{B.30}$$

with the "elastic trial" defined as

$$\begin{aligned}\boldsymbol{\sigma}^T &= \boldsymbol{\sigma}(t-\delta t) + \delta t \boldsymbol{C} : \dot{\boldsymbol{\varepsilon}} \\ &= \boldsymbol{\sigma}(t-\delta t) + \boldsymbol{C} : \big(\boldsymbol{\varepsilon}(t) - \boldsymbol{\varepsilon}(t-\delta t)\big)\end{aligned} \tag{B.31}$$

The tensors $\boldsymbol{\sigma}(t-\delta t)$, $\boldsymbol{\varepsilon}(t)$, $\boldsymbol{\varepsilon}(t-\delta t)$ are supposed to be known, the unknow variable is $\boldsymbol{\sigma}(t)$ whose notation will be further simplified into $\boldsymbol{\sigma}$. Equation B.30 can be represented as

$$\boldsymbol{F}(\boldsymbol{\sigma}) = \boldsymbol{0} \tag{B.32}$$

with

$$\boldsymbol{F}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}(t) - \boldsymbol{\sigma}^T + \delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big) \left(ch\Big(\frac{\boldsymbol{\mu}_s : \boldsymbol{\sigma}}{\tau_0^s}\Big) - 1 \right) \text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma}) \ \boldsymbol{C} : \boldsymbol{\mu}_s \right] \tag{B.33}$$

We consider the stress $\boldsymbol{\sigma}$ and $\boldsymbol{F}(\boldsymbol{\sigma})$ as vectors of 6 components that we will note respectively σ_i and F_i with $i = 1, 2, ..., 6$. The Jacobian $\boldsymbol{J}$ of $\boldsymbol{F}$ is a 6×6 matrix given by

$$J_{ij} = \frac{\partial F_i}{\partial \sigma_j} = \delta_{ij} + \delta t \sum_s \left[\frac{\dot{\gamma}_0^s}{\tau_0^s} \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big) \left(sh\Big(\frac{\boldsymbol{\mu}_s : \boldsymbol{\sigma}}{\tau_0^s}\Big) \right) \text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma}) \ \big(\boldsymbol{C} : \boldsymbol{\mu}_s\big)_i \, \mu_j^s \right] \tag{B.34}$$

Equation B.32 will be solved using a Newton's method which reads

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k - \boldsymbol{J}^{-1}(\boldsymbol{\sigma}_k) : \boldsymbol{F}(\boldsymbol{\sigma}_k) \tag{B.35}$$

where $\boldsymbol{\sigma}_k$ is the k-th evaluation of the stress of the iterative process.

B.10.3 `solve_spc0e` function

In order to use accelerated schemes (Eyre-Milton scheme, augmented Lagrangian scheme, Monchiet-Bonnet scheme), an additional function is required by CraFT that must be able to solve

$$\begin{cases} \boldsymbol{\sigma}(t) = \mathcal{F}(\boldsymbol{\varepsilon}(t), \boldsymbol{\varepsilon}(t-\delta t), \delta t, ...) \\ \boldsymbol{\sigma}(t) + \boldsymbol{C^0} : \boldsymbol{\varepsilon}(t) = \boldsymbol{\tau}(t) \end{cases} \tag{B.36}$$

where $\boldsymbol{\tau}(t)$ is a known field of tensor and where the unknowns are $\boldsymbol{\sigma}(t)$ but also $\boldsymbol{\varepsilon}(t)$. $\boldsymbol{C^0}$ is the rigidity tensor of the reference material. Equations B.36 can be simply rewritten as

$$\begin{cases} \boldsymbol{\sigma}(t) = \mathcal{F}(\boldsymbol{\varepsilon}(t), \boldsymbol{\varepsilon}(t-\delta t), \delta t, ...) \\ \boldsymbol{\varepsilon}(t) = \boldsymbol{S^0} : \big(\boldsymbol{\tau}(t) - \boldsymbol{\sigma}(t)\big) \end{cases} \tag{B.37}$$

with $\boldsymbol{S^0} = \boldsymbol{C^0}^{-1}$ the compliance tensor of the reference material.

In the case of UO2 material, this reads

$$\left\{\begin{array}{lll} \boldsymbol{\sigma}(t) = \boldsymbol{\sigma}(t-\delta t) & +\boldsymbol{C} : \big(\boldsymbol{\varepsilon}(t) - \boldsymbol{\varepsilon}(t-\delta t)\big) & \\ & -\delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big)\left(ch\Big(\frac{\boldsymbol{\mu}_s:\boldsymbol{\sigma}}{\tau_0^s}\Big) - 1\right)\text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma})\, \boldsymbol{C} : \boldsymbol{\mu}_s\right] \\ \boldsymbol{\varepsilon}(t) = \boldsymbol{S^0} : \big(\boldsymbol{\tau}(t) & -\boldsymbol{\sigma}(t)\big) & \end{array}\right. \tag{B.38}$$

If one eliminates $\boldsymbol{\varepsilon}(t)$, one has

$$\begin{aligned} \boldsymbol{\sigma}(t) = \boldsymbol{\sigma}(t-\delta t) \quad & +\boldsymbol{C} : \big(\boldsymbol{S^0} : \big(\boldsymbol{\tau}(t) - \boldsymbol{\sigma}(t)\big) - \boldsymbol{\varepsilon}(t-\delta t)\big) \\ & -\delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big)\left(ch\Big(\frac{\boldsymbol{\mu}_s:\boldsymbol{\sigma}}{\tau_0^s}\Big) - 1\right)\text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma})\, \boldsymbol{C} : \boldsymbol{\mu}_s\right] \end{aligned} \tag{B.39}$$

$$\begin{aligned} \boldsymbol{\sigma}(t) + \boldsymbol{C} : \boldsymbol{S^0} : \boldsymbol{\sigma}(t) = \quad & \boldsymbol{\sigma}(t-\delta t) + \boldsymbol{C} : \big(\boldsymbol{S^0} : \boldsymbol{\tau}(t) - \boldsymbol{\varepsilon}(t-\delta t)\big) \\ & -\delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big)\left(ch\Big(\frac{\boldsymbol{\mu}_s:\boldsymbol{\sigma}}{\tau_0^s}\Big) - 1\right)\text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma})\, \boldsymbol{C} : \boldsymbol{\mu}_s\right] \end{aligned} \tag{B.40}$$

We define $\boldsymbol{\sigma}^T$ as

$$\boldsymbol{\sigma}^T = \boldsymbol{\sigma}(t-\delta t) + \boldsymbol{C} : \big(\boldsymbol{S^0} : \boldsymbol{\tau}(t) - \boldsymbol{\varepsilon}(t-\delta t)\big) \tag{B.41}$$

and $\boldsymbol{F}(\boldsymbol{\sigma})$ as

$$\boldsymbol{F}(\boldsymbol{\sigma}) = \boldsymbol{\sigma}(t) + \boldsymbol{C} : \boldsymbol{S^0} : \boldsymbol{\sigma}(t) - \boldsymbol{\sigma}^T + \delta t \sum_s \left[\dot{\gamma}_0^s \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big)\left(ch\Big(\frac{\boldsymbol{\mu}_s : \boldsymbol{\sigma}}{\tau_0^s}\Big) - 1\right)\text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma})\, \boldsymbol{C} : \boldsymbol{\mu}_s\right] \tag{B.42}$$

The constitutive relation will be solved when

$$\boldsymbol{F}(\boldsymbol{\sigma}) = \boldsymbol{0} \tag{B.43}$$

A Newton's method will used. One introduces the jacobian matrix in a similar way as above in B.34

$$J_{ij} = \frac{\partial F_i}{\partial \sigma_j} = \delta_{ij} + C_{ik} S^0_{kj} + \delta t \sum_s \left[\frac{\dot{\gamma}_0^s}{\tau_0^s} \exp\Big(-\frac{\Delta H_0^s}{k_0 T}\Big)\left(sh\Big(\frac{\boldsymbol{\mu}_s : \boldsymbol{\sigma}}{\tau_0^s}\Big)\right)\text{sign}(\boldsymbol{\mu}_s : \boldsymbol{\sigma})\ \big(\boldsymbol{C} : \boldsymbol{\mu}_s\big)_i\, \mu_j^s\right] \tag{B.44}$$

And B.43 is iteratively solved with

$$\left\{\begin{array}{ll} \boldsymbol{\sigma}_{k+1} & = \boldsymbol{\sigma}_k - \boldsymbol{J}^{-1}(\boldsymbol{\sigma}_k) : \boldsymbol{F}(\boldsymbol{\sigma}_k) \\ \boldsymbol{\varepsilon}_{k+1} & = \boldsymbol{S^0} : \big(\boldsymbol{\tau}(t) - \boldsymbol{\sigma}_{k+1}\big) \end{array}\right. \tag{B.45}$$

B.10.4 Specification

- Netons's method

- value of the convergence threshold (Newton's method)

- Thermo-elastic parameters
  - N_{temp} number of temperature values in the following table
  - values of temperature (their number must be equal to N_{temp})
  - values of C_{11} (their number must be equal to N_{temp})
  - values of C_{12} (their number must be equal to N_{temp})
  - values of C_{44} (their number must be equal to N_{temp})
- crystalline parameters
  - number of families of systems s
  - parameters of the 1st family:
    - number of system for the 1st family
    - 3 components of $\boldsymbol{n_s}$ and 3 components of $\boldsymbol{m_s}$ for the 1st system
    - 3 components of $\boldsymbol{n_s}$ and 3 components of $\boldsymbol{m_s}$ for the 2d system
    - ...
    - $\dot{\gamma}_0^s$ shear strain of the 1st family
    - ΔH_0^s energy of activation of the 1st family
    - τ_0^s reference shear stress of the 1st family
  - parameters of the 2d family:
    - number of system for the 2d family
    - 3 components of $\boldsymbol{n_s}$ and 3 components of $\boldsymbol{m_s}$ for the 1st system
    - 3 components of $\boldsymbol{n_s}$ and 3 components of $\boldsymbol{m_s}$ for the 2d system
    - ...
    - $\dot{\gamma}^s$ shear strain of the 1st family
    - ΔH_0^s energy of activation of the 1st family
    - τ_0^s reference shear stress of the 1st family
  - ...

B.10.5 Example

```
#-------------------------------------------------------
# Mat. #2021 is a ThermallyActivated law for UO2
0 2021
#-------------------------------------------------------
# value of the convergence criteria
```

```
1.e-6
#-------------------------------------------------------
# Cij vs Temperature
# nbElts of the following table
8
# line 1: T ;  line 2: C11 ; line 3: C12 ; line 4: C44(Kelvin):
186. 2341 2422 2547 2656 2771 2870 2946
3.93e+11 2.18e+11 2.17e+11 2.01e+11 1.88e+11 1.65e+11 1.43e+11 1.27e+11
1.25e+11 9.26e+10 9.14e+10 8.95e+10 8.79e+10 8.61e+10 8.46e+10 8.35e+10
1.33e+11 9.21e+10 8.82e+10 8.02e+10 7.23e+10 6.36e+10 5.58e+10 4.94e+10
#-------------------------------------------------------
# Number of system family
2
#-------------------------------------------------------
# Number of system for the first family
6
# Slip systems for the first family
## n(i,1) n(i,2) n(i,3) m(i,1) m(i,2) m(i,3)
   1    0   0  0      1      1
   1    0   0  0      1     -1
   0    1   0  1      0      1
   0    1   0  1      0     -1
   0    0   1  1     -1      0
   0    0   1  1      1      0
# Value of the reference shear strain (gampt0) for the first family
4.59e7
# Value of the activation energie (DH0) for the first family
9.16e-19
# Value of the reference shear stress (tau0) for the first family
1.35e6
#-------------------------------------------------------
# Number of system for the second family
6
# Slip systems for the second family
## n(i,1) n(i,2) n(i,3) m(i,1) m(i,2) m(i,3)
   0    1  -1  0      1      1
   0    1   1  0      1     -1
   1    0  -1  1      0      1
   1    0   1  1      0     -1
   1    1   0  1     -1      0
   1   -1   0  1      1      0
# Value of the reference shear strain (gampt0) for the second family
6.83e6
# Value of the activation energie (DH0) for the second family
8.36734693878e-19
# Value of the reference shear stress (tau0) for the second family
4.78e6
#-------------------------------------------------------
```

Appendix C Examples

C.1 Examples of loading files

C.1.1 example of creep loading

In the following example of loading specification file, creep loading is prescribed: macroscopic stress is prescribed (C in the first directive) to be $\Sigma_{11} = 10$, $\Sigma_{ij\neq 11} = 0$ at time $t = 0.1s$ and following time steps.

```
#-----------------------------------
# prescribed stress
C
#-----------------------------------
# loading
#      t          direction        k
#             11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
       0.1   1. 0. 0. 0. 0. 0.    10.
       0.2   1. 0. 0. 0. 0. 0.    10.
       0.3   1. 0. 0. 0. 0. 0.    10.
       0.4   1. 0. 0. 0. 0. 0.    10.
       0.5   1. 0. 0. 0. 0. 0.    10.
       0.6   1. 0. 0. 0. 0. 0.    10.
       0.7   1. 0. 0. 0. 0. 0.    10.
       0.8   1. 0. 0. 0. 0. 0.    10.
       0.9   1. 0. 0. 0. 0. 0.    10.
       1.0   1. 0. 0. 0. 0. 0.    10.
```

A more concise specification using implied loop notation would be:

```
#-----------------------------------
# prescribed stress
C
#-----------------------------------
# loading
#      t          direction        k
#             11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
       0.1   1. 0. 0. 0. 0. 0.    10.
:9:    1.0   1. 0. 0. 0. 0. 0.    10.
```

Remarks:

- The line for time $t = 0.1$ is necessary as CraFT implies that macroscopic stress is null at time $t = 0$.

- The number of implied loops between $t = 0.1$ (not included) and $t = 1$ (included) is 9, thus the implied time step is: $0.1 = (1 - 0.1)/9$.

Another possibility for the same loading conditions would be:

```
#-----------------------------------
# prescribed stress
C
#-----------------------------------
# loading
#      t          direction        k
#            11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
       0.1   1. 0. 0. 0. 0. 0.    10.
%0.1%  1.0   1. 0. 0. 0. 0. 0.    10.
```

Time steps of implied loops between $t = 0.1$ and $t = 1$ being prescribed to 0.1

C.1.2 example of simple traction

In this example, a simple traction is applied in 11 direction ($\Sigma_{11} \neq 0\ \Sigma ij \neq 11 = 0$) untill $\mathbf{E} : \mathbf{d} = 50$ (in other words, untill $E_{11} = 50$) at time $t = 20$; 1000 time steps are applied.

```
#-----------------------------------
# prescribed direction of stress
S
#-----------------------------------
# loading
#         t          direction        k
#               11 22 33 12 13 23
#- - - - - - - - - - - - - - - - - - -
:1000:    20.   1. 0. 0. 0. 0. 0.    50.
```

Appendix D Infinitesimal rotation tensor

The infinitesimal rotation tensor can be defined as:

$$\boldsymbol{\omega} = \frac{1}{2}(\boldsymbol{\nabla u} - \boldsymbol{\nabla u}^T)$$

which can be detailed as:

$$\boldsymbol{\omega} = \begin{pmatrix} 0 & \frac{1}{2}(\frac{\partial u_1}{\partial x_2} - \frac{\partial u_2}{\partial x_1}) & \frac{1}{2}(\frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}) \\ \frac{1}{2}(\frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}) & 0 & \frac{1}{2}(\frac{\partial u_2}{\partial x_3} - \frac{\partial u_3}{\partial x_2}) \\ \frac{1}{2}(\frac{\partial u_3}{\partial x_1} - \frac{\partial u_1}{\partial x_3}) & \frac{1}{2}(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}) & 0 \end{pmatrix}$$

$\boldsymbol{\omega}$ is an anti-symmetrical 2d order tensor with only 3 independent components γ_1, γ_2, γ_3, defined as:

$$\begin{cases} \gamma_1 &= \frac{1}{2}(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}) \\ \gamma_2 &= \frac{1}{2}(\frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}) \\ \gamma_3 &= \frac{1}{2}(\frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}) \end{cases}$$

Therefore, the rotation tensor can read as:

$$\boldsymbol{\omega} = \begin{pmatrix} 0 & -\gamma_3 & \gamma_2 \\ \gamma_3 & 0 & -\gamma_1 \\ -\gamma_2 & \gamma_1 & 0 \end{pmatrix}$$

One can defined a vector $\boldsymbol{\Omega}$ as:

$$\boldsymbol{\Omega} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix}.$$

Thus, the rotation applied to a given vector $\boldsymbol{u}$ can be expressed either as:

$$\boldsymbol{\omega u}$$

or as:

$$\boldsymbol{\Omega} \times \boldsymbol{u}$$

CraFT saves images of the rotation in the shape of 3d vector images containing the 3 components of $\boldsymbol{\Omega}$.

Bibliography

[1] Moulinec H., Suquet P. 1994. *A fast numerical method for computing the linear and nonlinear properties of composites*. *C. R. Acad. Sc. Paris II*, **318**, 1417–1423.

[2] Moulinec H., Suquet P. 1998. *A numerical method for computing the overall response of nonlinear composites with complex microstructure*. *Comp. Meth. Appl. Mech. Engng.*, **157**, 69–94.

[3] Müller W.H. 1996. *Mathematical versus experimental stress analysis of inhomogeneities in solids*. *J. Physique IV*, **6**:C1.139-C1-148

[4] Lebensohn R.A. 2001. *N-site modelling of a 3D viscoplastic polycrystal using Fast Fourier Transform*. *Acta Materialia*, **49**, 2723–2737.

[5] Vinogradov V., Milton G.W. 2008. *An accelerated FFT algorithm for thermoelastic and non linear composites*. *Int. J. Numer. Meth. Engng*, 76: 1678–1695.

[6] Zeman J., Vondrejc J., Novak J. and Marek I. 2010 *Accelerating a FFT-based solver for numerical homogenization of periodic media by conjugate gradients*. *Journal of Computational Physics*, **229**, 8065–8071

[7] Brisard S., Dormieux L. 2010. *FFT-based methods dor the mechanics of composites: A general variational framework*. *Computational Materials Science*, **49**, 663–671

[8] Eyre D.J., Milton G.W. 1999. *A fast numerical scheme for computing the response of composites using grid refinement*. *J. Physique III*, **6**, 41–47.

[9] Milton G.W. 2002. *The Theory of Composites* (1st edition). Cambridge University Press.

[10] Michel J.C, Moulinec H, Suquet P. 2000. *A computational method based on augmented Lagrangians and Fast Fourier Transforms for composites with high contrast Comput. Modelling Engng. Sc.*, **1** (2), 79–88.

[11] Monchiet V. and Bonnet G. (2012): *A polarization-based FFT iterative scheme for computing the effective properties of elastic composites with arbitrary contrast*. *Int. J. Numer. Meth. Engng*, 89: 1419-1436.

[12] Eshelby J.D. 1957. *The Determination of the Elastic Field of an Ellipsoidal Inclusion, and Related Problems Proc. R. Soc. Lond. A*, **241**, 376-396

[13] Michel J.C, Moulinec H, Suquet P. 2001. *A computational method for linear and nonlinear composites with arbitrary phase contrast Int. J. Numer. Meth. Engng* **52**, 139-160.

[14] Kröner, E.. 1972. *Statistical Continuum Mechanics*. Springer-Verlag.

[15] Bhattacharya K., Suquet P.M. 2005. *A Model Problem concerning Recoverable Strains of Shape-Memory Polycrystals* . *Proc. R. Soc. Lond. A*, **461**, 2797–2816

[16] Bilger N., Auslender F., Bornert M., Michel J.C., Moulinec H., Suquet P., Zaoui A. 2005. *Effect of a nonuniform distribution of voids on the plastic response of voided materials : a computational and statis- tical analysis*. *Int. J. Solids Structures*, **42**, 517-538

[17] Moulinec H., Silva F. 2013. *Comparison of three accelerated FFT-based schemes for computing the mechanical response of composite materials Int. J. Numer. Meth. Engng* (submitted)

[18] Kajetan Wojtacki, Pierre-Guy Vincent, Pierre Suquet, Hervé Moulinec, and Guylaine Boittin. A micromechanical model for the secondary creep of elasto-viscoplastic porous materials with two rate-sensitivity exponents: Application to a mixed oxide fuel. *International Journal of Solids and Structures*, 184:99 – 113, 2020. Physics and Mechanics of Random Structures: From Morphology to Material Properties.

[19] Hervé Moulinec, Pierre Suquet, and Graeme W. Milton. Convergence of iterative methods based on neumann series for composite materials: Theory and practice. *International Journal for Numerical Methods in Engineering*, 114(10):1103–1130, 2018.

[20] Cédric Bellis and Pierre Suquet. Geometric variational principles for computational homogenization. *Journal of Elasticity*, 137(2):119–149, Dec 2019.