Object-Oriented Programming Applied to the Finite Element Method
Part II. Application to Material Behaviors

Jacques Besson * — Rodolphe Leriche *,***
Ronald Foerch ** — Georges Cailletaud *

* *Ecole des Mines de Paris, Centre des Matériaux*
CNRS URA 866, F-91003 Evry cedex

** *North West Numerics Inc., 444 N.E. Ravenna Blvd*
Suite 301-A, Seattle, WA 98115, USA

*** *Institut de Mécanique de Rouen*
Avenue de l'Université, Le Madrillet, F-76300 Saint-Etienne-du-Rouvray

ABSTRACT. *This paper examines the application of C++ design patterns to the development of material constitutives equations to be used in finite element simulation softwares. All material behaviors use the same generic interface. Accordingly, the same governing principles can be applied to simple elasto-(visco)-plastic materials as well as single crystals, polycrystals and multiphased materials. As these models use numerous parameters, a generic optimization tools was also developed to adjust these parameters. Finally, a specific pre-processor can be used to quickly implement and test new material behaviors. The result (i.e., interface + optimizer + pre-processor) is an integrated approach to the development of new constitutive equations for structural computations.*

RÉSUMÉ. *Cet article examine l'application de schémas de programmation C++ au développement de lois constitutives pouvant être utilisées dans les calculs par éléments finis. Tous les comportements peuvent utiliser la même interface générique. Par conséquent, les mêmes principes peuvent être appliqués aux matériaux simples de type élasto-(visco)-plastique, aux monocristaux, aux polycristaux ainsi qu'aux matériaux multiphasés. Comme ces modèles comportent un grand nombre de paramètres, un outil d'optimisation a également été développé afin d'ajuster ces paramètres. Enfin, un pré-processeur spécifique peut être utilisé afin d'implanter et de tester rapidement de nouveaux comportements. Le résultat (i.e., interface + optimiseur + pré-processeur) est une approche intégrée du développement des nouvelles lois constitutives pour le calcul de structures.*

KEY WORDS : *object-oriented languages, finite element method, optimization, constitutive equations.*
MOTS-CLÉS : *langages orientés objet, méthode des éléments finis, optimisation, équations de comportement.*

1. Introduction

Problems encountered in the field of engineering computations tend to handle an increasing number of degrees of freedom and also more complex constitutive equations to describe the material behavior. In an accompanying paper [BES 98], general design patterns based on object-oriented programming have been presented. They can be used to ease, secure and quicken the implementation of new functionalities.

In this second paper, the proposed programming concepts are applied to simplify the implementation of complex material behaviors. This is achieved in particular by isolating material description from all other aspects of the code, especially those related to the finite element method. New behavior laws are derived from existing objects whose formulation is based on physics, mathematical relations, and pragmatics. Such a careful implementation of basic programming patterns has allowed the implementation of various, increasingly complex behaviors with a reduced amount of programming. Current features of Zébulon include large deformations, damage, multi-mechanisms elastoviscoplastic materials with isotropic and kinematic hardening, micro-macro models for single and polycrystals, and multi-materials (for example, metal matrix composites). Diffusion and conduction law calculations are also implemented.

One of the principal difficulties associated with complex models is the number of parameters to be handled. Besides numerical modeling, recent experimental practice also contributes to the increasing difficulty of parameter determination: the development of tests on large scale structures with unhomogeneous three dimensional stress states hide the effect of the material parameters in the global quantities that are measured. Both modeling and experiments call for better identification tools. To answer this need, Zébulon has a general identification and optimization module. It features a simple interfacing mechanism based on templates and various optimization methods.

2. Material Objects

C++ classes have been created for handling material-related variables in a physical and pragmatic way. The material classes receive a `primal` variable from the finite elements, and return an associated `dual` variables [BES 98]. Other types of variables exist as well, and are listed in Figure 1. A `MAT_DATA` class holds and manages the variables of Figure 1 at each Gauss point. The software does not make any difference between internal variables (`IV`) which have a thermodynamic meaning, and internal parameters (`IP`). They are both handled as integrated variables: $\mathcal{V}_{\text{int}}$ =`IP`+`IV`. It is also important to note the use of generic input/output (`primal`/`dual`) variables that allow to define a generic interface for all behaviors. This interface can be used by any object handling behaviors. This is indeed the case of elements; this also provides an easy way to implement multi-material behaviors (e.g., composites). The

`primal`	Input from elements or any object handling a behavior	Primary imposed problem variable over which the behavior is integrated. e.g. $\underset{\sim}{\varepsilon}$, $\underset{\sim}{\mathbf{F}}$, $(T, \underline{\nabla T})$.
`dual`	Output from the behavior integration	Returned variable defining the internal force. Corresponding e.g. $\underset{\sim}{\sigma}$, $\underset{\sim}{\mathbf{S}}$, $(\underline{\mathbf{q}}, q_v)$.
`EP`	External Parameters	Values used as "loading" in the problem definition. Set by the user, and thus always known. e.g. temperature already calculated, humidity, grain size, etc.
`IV`	Internal Variables	Define the thermodynamic state of the material and have associated forces. e.g. $\underset{\sim}{\varepsilon}_{el}$–$\underset{\sim}{\sigma}$, $\underset{\sim}{\alpha}$–$\underset{\sim}{\mathbf{X}}$, s–T.
`IP`	Internal Parameters	Variables providing information about the material state, but lacking an associated thermodynamic force. e.g. $p = \int (\epsilon : \epsilon)^{0.5}\, dt$.
$\mathcal{V}_{\text{int}}$	Integrated variables	The model variables which are to be integrated over a given time increment in order to arrive at the new `dual`. In general will contain variables from both `IV` and `IP`.
$\mathcal{V}_{\text{aux}}$	Auxiliary Variables	Interesting information which does not directly define the material state (output).
`CO`	Coefficients	Coefficients used to define a specific material within a general behavior. May depend on `EP`, $\mathcal{V}_{\text{aux}}$ and $\mathcal{V}_{\text{int}}$.

Figure 1. *List of problem variables which define a constitutive equation*

composite object handles several base materials and evaluates their behaviors using the general purpose interface. The `BEHAVIOR` class reads and maintains the coefficients, provides access to the variables stored in `MAT_DATA`, has an integration function for the material law, and handles output of local data. All these classes make full use of the programming patterns presented in Part I.

An important instance of behavior concerns phenomenological elasto–(visco)–plastic models which are implemented through the `GEN_EVP` class. `GEN_EVP` permits the assemblage of a material law using predefined elementary blocks. Different inelastic mechanisms are superposed in terms of strain rates. Each inelastic deformation mechanism is defined by a plasticity criterion, a flow rule, and an arbitrary number of hardening mechanisms (isotropic or kinematic). Interactions between mechanisms are handled through the `INTERACTION` class whose content stems from the definition of a quadratic free energy potential. The structure of `GEN_EVP` enables new behaviors to be dynamically build through the superposition of inelastic mechanisms.

Materials are often tested on axisymmetrical specimen and simulated on unit volume element. In general purpose finite element codes, simulations are performed on one element with 4 or 8 nodes and 4 or 9 Gauss points. In order to gain in numerical efficiency when simulating materials, a representative volume element (RVE) has been developed: it has a single Gauss point (no mesh is required). The degrees of freedom are the strains; associated forces are stresses. Loading can be any valid combination of strains and stresses.

3. Optimization Tools

Zébulon has an optimization module that can handle any kind of constrained optimization problem with continuous design variables. It is, however, particularly adapted to parameter identification problems. Optimization problems derive from the general `BASE_PROBLEM` object of Zébulon. They therefore inherit from the read, load, verification, and error handling methods. The module is based on an abstract base class, `OPTIMIZER`, a utility class, `COMPARISON`, and two container classes, `OPTIMIZED_VARIABLES` and `CONSTRAINT`.

`OPTIMIZER` is the parent class of all particular implementations of optimization methods (evolutionary algorithm, SQP, simplex, simulated annealing, BFGS, see Figure 2).

`COMPARISON` offers tools for calculating the distance between curves (which typically correspond to experiments and simulations). `COMPARISON` objects can interpolate data when necessary. Comparisons can also be made with analytical functions.

`OPTIMIZED_VARIABLES` contains the values of the parameters, their names, their upper and lower bounds, reference values used for norming the design variables, and the corresponding norming method to norm the design variables. The purpose of this framework is to make the addition of new optimization tools (sensitivity calculation or optimization methods) easier. A representation of the classes involved in the optimization module is given in Figure 2.

3.1. *Interfacing Strategy*

Communication with the optimization module is achieved through a file interface. A generic optimization interface is shown in Figure 3. The optimization method is specified through the `****optimize` command. Any external program that is necessary to the calculation of the distance minimized can be executed, at each set of variables, through the `***shell` instruction. Unknowns are declared using a template mechanism, which makes it straightforward to declare as variable any part of a file.

Unknowns are simple ASCII strings preceded by a "?" symbol. Unknowns are located in the files declared after `***files`, with the extension `.tmpl` added. Unknowns are further specified in the optimization file after the `values` instruction. At each evaluation of the distance to minimize (the "cost function"), Zébulon replaces, in the `.tmpl` files, the "?+unknown name" by a numerical value. This template translation yields the same file without `tmpl` extension and with unknowns replaced by trial values. Such a mechanism is very versatile and can easily modify FE–related files like material files.

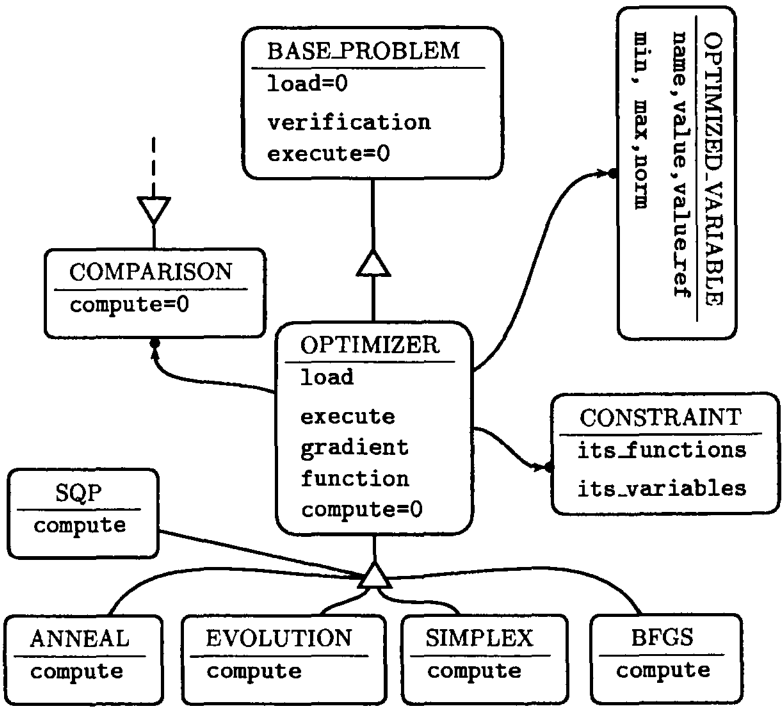


Figure 2. *Structure of the optimization module*

3.2. *Optimization Methods*

Five optimization methods are currently available in Zébulon, a simplex [NEL 65], a SQP [ZHO 97], a BFGS [BRO 70], an evolutionary algorithm [BÄC 96] and a simulated annealing algorithm. This panel of methods encompasses most aspects of optimization algorithms (local versus global, constrained versus unconstrained optimization).

On the one hand, there are three local, rapidly converging methods, namely the simplex, the SQP, and the BFGS algorithms. The evolutionary and the simulated annealing algorithms are, on the other hand, stochastic methods global in scope, i.e., they have a chance of escaping local optima. As such, they offer additional robustness when dealing with non-convex, multi-modal optimization problems. It is worth noting that most identification and optimization problems of practical interest are non-convex optimization problems with many local optima. This is a consequence of the increasing level of complexity of the models: the number of parameters is rapidly growing (increasing number of kinematic hardening variables, increasing number of

```
****optimize optimizer name (sqp, evolution, simplex, anneal, bfgs)
  ***shell external program
  ***files files with unknowns
  ***compare simulation and experiment files to compare
  ***values
      name init. value min max
  ***constraint function of the unknowns
  ***convergence
      parameters of the optimizer
****return
```

Figure 3. *Optimizer Interface*

parameters in micro–macro models,etc.) for the same single macroscopic structural response, so that many combinations of parameters can produce reasonable responses. Globality in evolutionary and simulated annealing is achieved at the expense of a larger number of evaluations of the cost function.

Among the local optimizers, simplex is a 0 order method (it does not require gradient calculation). It is a robust optimizer, especially for low slope functions. BFGS is a quasi–Newton method, i.e., a second–order method, where an approximation of the Hessian is calculated during the search. SQP, which stands for sequential quadratic programming, works by sequentially approximating the optimization problem by a quadratic programming problem, which is then solved by *ad–hoc* methods [SCH 86]. It is the only method among those proposed here to explicitly handle optimization constraints. [Explicitly means by a strategy other than penalization: if the optimization problem is formulated as $\min_{x \in S} f(x)$ such that $g(x) \leq 0.$, penalization would replace this original constrained problem by an unconstrained problem of the form $\min_{x \in S} f(x) + H(r,x)$, where $H(r,x) = 0$ when $g(x) \leq 0$ and $H(r,x) > 0$ otherwise. $H(r,x)$ is an increasing function of r, $r > 0$.] In the version included in the module (CFSQP), quasi-feasible iterates only are generated.

The evolutionary algorithm of the module is a standard steady state algorithm, with real numbers encoding, a linear cross–over operator and a Gaussian mutation. It presents two specificities:

1. constraints are handled through a dynamically adjusting penalty function ($dr \propto max(g(\bar{x}), 0.)$ where $\bar{x}$ is the best solution so far in terms of the penalized objective function).

2. The set of the best n_f feasible and n_i infeasible points distant of at least Δx (user specified) are collected during the search. This set is a collection of alternative good solutions, different from the final population.

4. Examples

Examples of identification (model implementation, interface with the optimizer, and results) will be provided for three problems: first a phenomenological elastoviscoplastic model identified using a representative volume element, then a behavior with damage identified on a structure, and finally a micro–macro self–consistent model for polycrystals.

4.1. *Identification of a Phenomenological Behavior*

The parameters of a phenomenological elastoviscoplastic behavior are identified in this example. The model includes both isotropic and kinematic hardening [CHA 89]. The equations of the model are,

stress: $\underset{\sim}{\sigma}$, elastic strain: $\underset{\sim}{\varepsilon}_e$	$\underset{\sim}{\sigma} = \underset{\approx}{C}\underset{\sim}{\varepsilon}_e$
$\underset{\sim}{X}$: back stress, $\underset{\sim}{\alpha}$: associated IV	$\underset{\sim}{X} = \frac{2}{3}C\underset{\sim}{\alpha}$
yield function	$f = J(\underset{\sim}{\sigma} - \underset{\sim}{X}) - R$
$\dot{v}$: creep rate	$\dot{v} = \left\langle \frac{J(\underset{\sim}{\sigma} - \underset{\sim}{X}) - R}{K} \right\rangle^n$
Von Mises norm	$J(\underset{\sim}{A}) = \sqrt{\frac{3}{2}\underset{\sim}{A} : \underset{\sim}{A}}$
$\dot{\underset{\sim}{\varepsilon}}_p$: viscoplastic strain rate	$\dot{\underset{\sim}{\varepsilon}}_p = \dot{v}\,\underset{\sim}{n}$
flow direction	$\underset{\sim}{n} = \frac{\partial f}{\partial \underset{\sim}{\sigma}}$
	$\dot{\underset{\sim}{\alpha}} = \dot{\underset{\sim}{\varepsilon}}_p - \frac{3}{2}\frac{D}{C}\underset{\sim}{X}\dot{v}$
isotropic hardening	$R = Q(1 - \exp(-bv)) + R_0$

The material coefficients are the elastic tensor $\underset{\approx}{C}$, the coefficient for kinematic hardening (C and D), the coefficients for isotropic hardening (Q and b) and the initial yield stress R_0. A tension test and a cyclic fatigue test are used in the set of experiments (although they were numerical experiments in this particular example). An example of data files for this identification is given in Figure 4. Identification is done on C, D, Q and b; all other coefficients are assumed to be known. The identification example was purposely designed so as to have a local optimum, and thus show the respective advantages and drawbacks of local and global optimizers. The material to determine has only kinematic hardening. For the tension test however, another material with only isotropic hardening perfectly fits the data. This material with isotropic hardening material is the local optimum. The difference between both materials can be seen on the cyclic fatigue test. The optimal material has $C = 2500$ MPa,

$D = 10$, $Q = 0$ MPa and b does not matter. The locally optimal isotropic material has $C = 0$, D does not matter, $Q = 250$ MPa, and $b = 10$.

file `optimize.inp`

```
****optimize sqp
  ***shell z7 rvee_trac
           z7 rvee_cyc
  ***files material
  ***compare
    t_file_file
      rvee_t.test 1 3 rvee_t.exp 1 3
    t_file_file
       rvee_c.test 1 3 rvee_c.exp 1 3
  ***values
   xC 1.   min  0.    max  400000.
   xD 40.  min 1.     max  3000.
   xQ 250. min -500. max 500.
   xb 10.6 min 0.2    max  20.0
  ***exponent  2.
  ***convergence ...
****return
```

file `material.tmpl`

```
***behavior gen_evp
 **elasticity isotropic
       young    169000.
       poisson   0.3
 **potential gen_evp ev
  *criterion mises
 *flow norton
       n      2.
       K   622.
 *kinematic nonlinear
       C  ?xC
       D  ?xD
 *isotropic nonlinear
       R0  645.
       Q  ?xQ
       b  ?xb
***return
```

Figure 4. *Data files for identifying an elastoviscoplastic behavior using the SQP algorithm. The* `***shell` *command runs two problems: the simulation of the tension test (*`rvee_trac`*) and the simulation of the cyclic test (*`rvee_cyc`*). These problems generate output files (*`rvee_trac.test` *and* `rvee_cyc.test`*) which are compared to experiments (*`rvee_t.exp` *and* `rvee_c.exp`*). Data relative to the convergence of the algorithm (*`***convergence`*) have been omitted.*

One identification is given using the SQP algorithm, and compared to an identification performed using the evolutionary algorithm. The SQP algorithm is started at $C = 5$ MPa, $D = 50$, $Q = 253$ MPa, and $b = 10.6$ (purposely near the local optimum). The distance between simulation and experiments for this set of parameters is $L_2 = 575$. SQP converges to $C = 5.54$ MPa, $D = 639$, $Q = 57.7$ MPa, and $b = 19.99$ where $L_2 = 205$ after 280 simulations. Clearly, SQP diminished the isotropic hardening, but it could not really escape the local optimum. The same identification was performed using the evolutionary algorithm with population size=200, mutation probability=0.1/real parameter, cross–over probability=0.7, search stopped after 6000 simulations. The optimum point was $C = 3813$ MPa, $D = 57$, $Q = -17$ MPa, and $b = 18$, where $L_2 = 7.6$. 6 other points such that $L_2 < 15$ were located during the run. Although the performance of a stochastic optimizer should never be assessed from a single test, this one clearly shows the ability of evolutionary algorithms to approach global optima, although at a high numerical cost. The convergences of SQP and evolutionary algorithms are further compared in Figure 5.

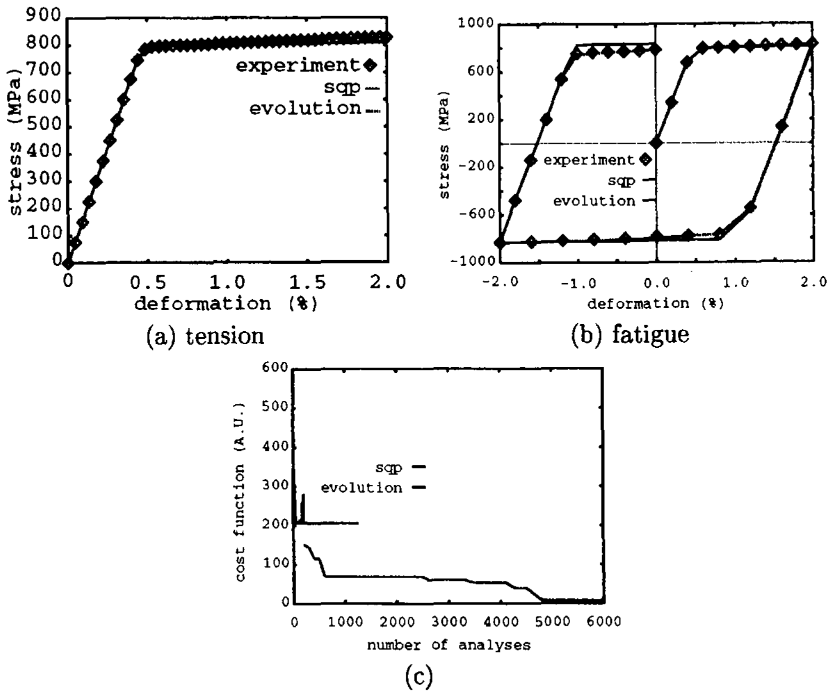


Figure 5. *Convergence of SQP vs. convergence of an evolutionary algorithm. (a) comparison of simulations on tensile tests, (b) comparison of simulations on cyclic tests, (c) convergence rates of both algorithms*

4.2. *Simultaneous Identification of a Ceramic in Tension and Bending*

This example shows how identification tools were used in the development of a model for the damage behavior of a ceramic (reaction bonded silicon carbide: RBSC) at high temperature (1250°C) [GUT 97]. At such a temperature, uniaxial compressive tests are particularly difficult to perform. It is nevertheless mandatory to test the material both in tension and compression because these ceramics are known to behave differently under tensile or compressive stress states. The initial experimental data base (tension) has therefore been enriched with 3-and 4-point bending tests where the global structural response of the structure (force and displacements) was recorded. The identification involves both volume element and finite elements calculations of bending bars. The following model was chosen to describe the non–symmetrical behavior of the ceramic. It is based on the previous model for which isotropic hardening is not considered; in addition, it includes a damage variable ω whose evolution is governed by the internal back stress $\underset{\sim}{\mathbf{X}}$.

stress: $\underset{\sim}{\sigma}$, elastic strain: $\underset{\sim}{\varepsilon}_e$	$\underset{\sim}{\sigma} = \underset{\approx}{\mathbf{C}}\underset{\sim}{\varepsilon}_e$
strain partition	$\underset{\sim}{\varepsilon}_t = \underset{\sim}{\varepsilon}_e + \underset{\sim}{\varepsilon}_p$
$\underset{\sim}{\mathbf{X}}$: back stress, $\underset{\sim}{\alpha}$: associated IV	$\underset{\sim}{\mathbf{X}} = \frac{2}{3}C\underset{\sim}{\alpha}$
$\dot{v}$: creep rate	$\dot{v} = \left\langle \frac{J(\underset{\sim}{\sigma} - \underset{\sim}{\mathbf{X}})}{K(1-\omega)} \right\rangle^n$
$\dot{\underset{\sim}{\varepsilon}}_p$: viscoplastic strain rate	$\dot{\underset{\sim}{\varepsilon}}_p = \dot{v}\,\underset{\sim}{\mathbf{n}}$
yield function	$f = J(\underset{\sim}{\sigma} - \underset{\sim}{\mathbf{X}})$
flow direction	$\underset{\sim}{\mathbf{n}} = \frac{\partial f}{\partial \underset{\sim}{\sigma}}$
evolution of $\underset{\sim}{\alpha}$	$\dot{\underset{\sim}{\alpha}} = \dot{\underset{\sim}{\varepsilon}}_p - \frac{3}{2}\frac{D/C}{1-\omega}\underset{\sim}{\mathbf{X}}\dot{v}$
damage evolution	$\dot{\omega} = \left(\frac{X_I - X_0(1-\omega)}{A} \right)^r (1-\omega)^{-k}$

Strain is divided into an elastic ($\underset{\sim}{\varepsilon}_e$) and a viscoplastic term ($\underset{\sim}{\varepsilon}_p$). The viscoplastic strain rate is based on the cumulated inelastic strain rate $\dot{v}$. There is no elasticity domain, but the flow rate is evaluated from a neutral state such as described by the kinematic hardening variable $\underset{\sim}{\mathbf{X}}$. The flow rate is also a function of damage, ω. Flow rate is therefore controled by 2 scalars (v and ω) and 2 tensors ($\underset{\sim}{\sigma}$ and $\underset{\sim}{\mathbf{X}}$ or $\underset{\sim}{\varepsilon}_e$ and $\underset{\sim}{\alpha}$). Damage behavior is non–symmetrical and driven by the largest principal internal stress, X_I. For uniaxial testing, the deviatoric internal stress is written as:

$$\underset{\sim}{\mathbf{X}} = \begin{pmatrix} \frac{2X}{3} & & \\ & -\frac{X}{3} & \\ & & -\frac{X}{3} \end{pmatrix}$$

In compression X_I is twice as large as it is in tension, and it always remains positive since $\underset{\sim}{\mathbf{X}}$ is deviatoric.

Figure 6 shows the interface with the optimizer for this problem. Results of the identification are given in Figure 7. Some of them are derived with structure bending tests while others are not. Note that, in order to involve damage behavior, the identification not only concerns strain/stress behavior but also time to fracture. Values of the model parameters after identification are:

n	K (MPa.s$^{1/n}$)	C (MPa)	D	X_0 (MPa)	A (MPa)	r	k
3.43	11070	375500	2570	70	480	9	1

file `optimize.inp`

```
****optimize sqp
  ***shell z7 rvee
  ***files material
  ***compare t_file_file
  rvee.test 1 3 rvee.exp 1 3
  ***values
   xC 200000. min 3000. max 300000.
   xD 2000. min 500. max 5000.
   xA 1600. min 1000. max 2000.
   xr 6. min 5. max 15.
   xk 30. min 10. max  40.
   xX0 70. min 60. max 80.
  ***constraint xC / xD - 200. ;
  ***exponent  1.
  ***convergence
****return
```

(a)

file `material.tmpl`

```
 ***behavior RBSC
  **elasticity isotropic
        young    400000.
        poisson   0.28
  **model_coef
        n      2.
        K   5000.
        C  ?xC
        D  ?xD
        A  ?xA
        r  ?xr
        k  ?xk
        X0 ?xX0
***return
```

(b)

Figure 6. *Files for the identification of an elastoviscoplastic behavior with damage, using SQP optimizer and a reduced volume element*

4.3. *Crystal Plasticity*

The crystallographic approach provides an improved framework with respect to the classical macroscopic models to predict the stress–strain behavior of polycrystalline material. The model first represent the plastic behavior of a single crystal; single crystals can then be assembled using proper stress/strain localization rules.

4.3.1. *Constitutive equations for one grain*

It is assumed that slip is the predominant deformation mechanism, and that Schmid's law is valid. The resolved shear stress can then be used as a critical variable to evaluate the inelastic flow. A viscoplastic framework is chosen, in order to avoid the problems related to the determination of the active slip systems in plastic models. A threshold is introduced both in positive and negative direction on each slip system: twelve octahedral slip systems will be used for FCC materials. For each slip system $(1 \ldots N)$, scalar isotropic r^s and kinematic x^s variables are defined. A system will be active provided its resolved

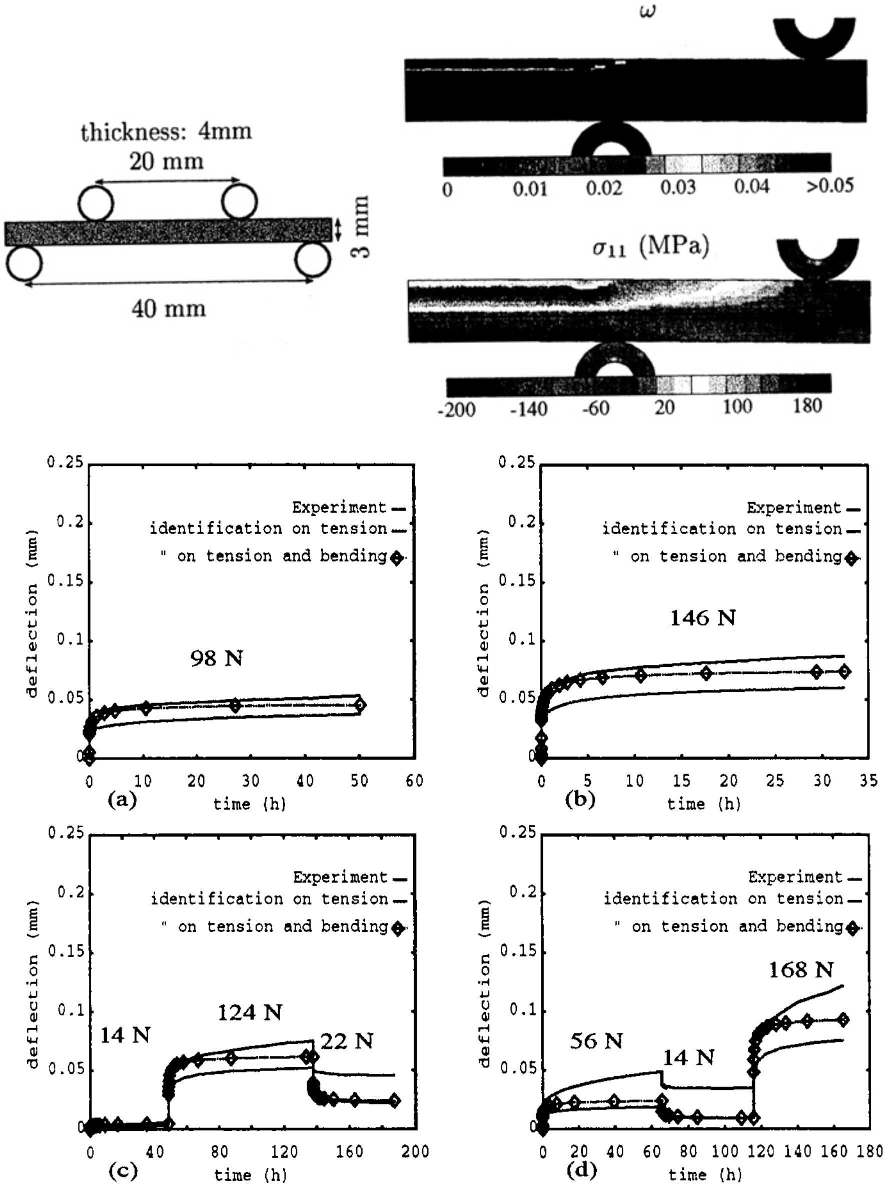


Figure 7. *Creep tests with RBSC at 1300°C, parameters identified using tension and bending: simulation and experiments*

shear stress τ^s is greater than $x^s + r^s$ or less than $x^s - r^s$. The state variables (IV) used to define the evolution of r^s and x^s are the cumulated slip v^s for isotropic hardening and the variable α^s for kinematic hardening.

Knowing the stress tensor applied to the grain g, the resolved shear stress for system s can be classically written as:

$$\tau^s = \underset{\sim}{\sigma}^g : \underset{\sim}{m}^s = \frac{1}{2}\underset{\sim}{\sigma}^g : (\underline{\mathbf{n}}^s \otimes \underline{\mathbf{l}}^s + \underline{\mathbf{l}}^s \otimes \underline{\mathbf{n}}^s)$$

where $\underline{\mathbf{n}}^s$ and $\underline{\mathbf{l}}^s$ are respectively, for system s, the normal to the slip plane and the slip direction in this plane. x^s and r^s are expressed as:

$$\begin{aligned} x^s &= c\alpha^s \\ r^s &= R_0 + Q_1\left(\sum_{r=1}^{N} h_{rs}\left(1 - e^{-b_1 v^r}\right)\right) + Q_2\left(1 - e^{-b_2 v^s}\right) \end{aligned}$$

The present formulation results in saturation of the hardening in both monotonic and cyclic loading, and takes into account the interactions between the slip systems, through the h_{rs} matrix, as in [KOC 66, FRA 85].

The evolution of the cumulated slip $\dot{v}^s$ is expressed as a function of the overstress $|\tau^s - x^s| - r^s$; a Norton flow rule is generally used:

$$\dot{v}^s = \left\langle \frac{|\tau^s - x^s| - r^s}{K} \right\rangle^n$$

The slip rate $\dot{\gamma}^s$ is then given by:

$$\dot{\gamma}^s = \dot{v}^s \mathrm{sign}(\tau^s - x^s)$$

The evolution of the kinematic hardening variable is given by:

$$\dot{\alpha}^s = \dot{\gamma}^s - d\alpha^s \dot{v}^s$$

The viscoplastic strain rate tensor $\dot{\underset{\sim}{\varepsilon}}^g$ is expressed as:

$$\dot{\underset{\sim}{\varepsilon}}^g = \sum_{s=1}^{N} \dot{\gamma}^s \underset{\sim}{m}^s$$

Such a formulation is an extension of the classical crystallographic approach for single crystal modeling in plasticity or in viscoplasticity (see for instance, [TAY 38, MAN 65, ASA 83]). This type of model has been used for single crystal modeling using FE simulations for copper [MÉR 94] and nickel-based superalloys [MÉR 91, NOU 95]

4.3.2. *Constitutive Equations for a Polycrystal*

In a polycrystalline aggregate, one grain may be characterized by its shape, size, crystallographic orientation, location with respect to the surface of the material, etc. In the present modeling, we simply retained the crystallographic orientation, and put in the same *crystallographic phase* all the grains having the same orientation. The alloy is then considered as a n–phase material where each phase is defined by a set of Euler angles and a volume fraction f^g. The model will then be used to describe the mean behavior of all of them. Such an approach is valid provided the grain size remains small with respect to the volume element considered in the future F.E. calculations, that is each elementary volume around a Gauss point must contain a sufficient number of grains. In these conditions, the concentration rule will define the way for computing the local stress tensor $\underset{\sim}{\sigma}^g$, which will be uniform in the phase, starting from the macroscopic stress tensor $\underset{\sim}{\sigma}$.

The self–consistent scheme is a good solution to schematically represent the phase interaction. The first papers on the subject were devoted to the definition of the elastic accommodation [KRÖ 61] and the plastic accommodation [HIL 65]. A pure viscoplastic formulation was also used in the past (see for instance [HUT 76, MOL 87]), but the elastoviscoplastic approximation is still not available. In the simple case where the phases have the same elastic properties, the results given by several models can be written as:

$$\underset{\sim}{\sigma}^g = \underset{\sim}{\sigma} + \alpha\mu\left(\underset{\sim}{\epsilon}^p - \underset{\sim}{\epsilon}^g\right)$$

where μ is the shear modulus. $\underset{\sim}{\epsilon}^p$ is the average plastic deformation $\underset{\sim}{\epsilon}^p = \sum_g f^g \underset{\sim}{\epsilon}^g$. α is a parameter which differs from a model to another. In the case of elastic accommodation, $\alpha = 1$. In the case of the simplified approach proposed in [BER 79], α is a decreasing function of the macroscopic elastoplastic secant modulus starting from 1 at the onset of plastic flow. This last formulation is equivalent to Hill's model for radial loading. It demonstrates that the corrective term α in the accommodation rule must be non–linear.

The model presented in this study is based on a phenomenological and pragmatic modification of the models presented above. The non–liner accommodation is reproduced using a new variable for each grain: $\underset{\sim}{\beta}^g$ [CAI 92, PIL 94]. The evolution law of $\underset{\sim}{\beta}^g$ can be calibrated by means of a specific identification procedure to numerically check self–consistency. The localization rule can be written as:

$$\underset{\sim}{\sigma}^g = \underset{\sim}{\sigma} + \mu\left(\underset{\sim}{\beta} - \underset{\sim}{\beta}^g\right) \quad \text{with} \quad \underset{\sim}{\beta} = \sum_g \underset{\sim}{\beta}^g$$

The evolution law of $\underset{\sim}{\beta}$ is written as (note that other forms could be chosen):

$$\dot{\underset{\sim}{\beta}}^g = \dot{\underset{\sim}{\epsilon}}^g - D\left(\underset{\sim}{\beta}^g - \delta\underset{\sim}{\epsilon}^g\right)\sum_g f_g \sum_s \dot{v}^s$$

This equation superimposes a non–linear kinematic hardening term and a linear kinematic hardening term (provided $\delta \neq 0$) at the intergranular level.

4.3.3. *Application*

The model is used to compute the yield surface of a polycrystal after deformation (Figure 8). Calculations were made using 40 grains whose orientations were chosen in order to obtain an isotropic material. The following coefficients were used (units correspond to MPa and s):

grain	$n = 25$	$K = 50$	$R_0 = 100$	$Q_1 = 50$	$b_1 = 50$
	$C = 2500$	$D = 10$	$h_{rs} = 1$		
polycrystal	$C = 50000$	$D = 100$	$\delta = 0$		

5. ZebFront: A Material Pre–Processor

As a result of the interface standardization of material objects, it becomes possible to implement a simple pre–processor that will translate a simplified behavior definition file into a C++ program file compatible with Zébulon. The use of this pre–processor (called ZebFront) is demonstrated in the case of the RBSC ceramic whose behavior has been introduced in Section **4.2.**. ZebFront files contains specific directives preceded by "`@`" which are interpreted by the pre–processor. Otherwise, standard C++ and Zébulon objects can be used in the program.

Figure 9 shows the program written to implement the behavior for RBSC. The ZebFront directives have the following meaning:

`@Class`. This directive defines a new object representative of a material behavior. The class name (e.g., `RBSC`) is given after the directive. Using this class declaration, the pre–processor automatically generates code needing to read and initialize variables, coefficients and member classes.

`@Name`. This directive defines the name to be used in input files to load the new behavior (see 6–b).

`@SubClass`. This directive allows the definition as a data member an already existing Zébulon object. It is followed by the object type (e.g., `ELASTICITY`) and the name of the data member. This name will be used both in the input file and in the program. Note that polymorphic objects can be used; in the present example any type of elastic behavior can be used (isotropic, cubic, orthotropic, etc.). Such a possibility could have been used here in the case of viscoplastic flow (`FLOW`) instead of using coefficients K and n which implies a Norton law.

`@Coefs`. This directive allows to define coefficients. It is also possible to define arrays of coefficients (e.g., `@Coefs K[10];`).

`@tVarInt`,... This directive declares tensorial integrated variables (e.g., $\underset{\sim}{\alpha}$). It is also possible to define scalar integrated variables using `@sVarInt`. Auxiliary variables are declared using `@tVarAux` or `@sVarAux`. The

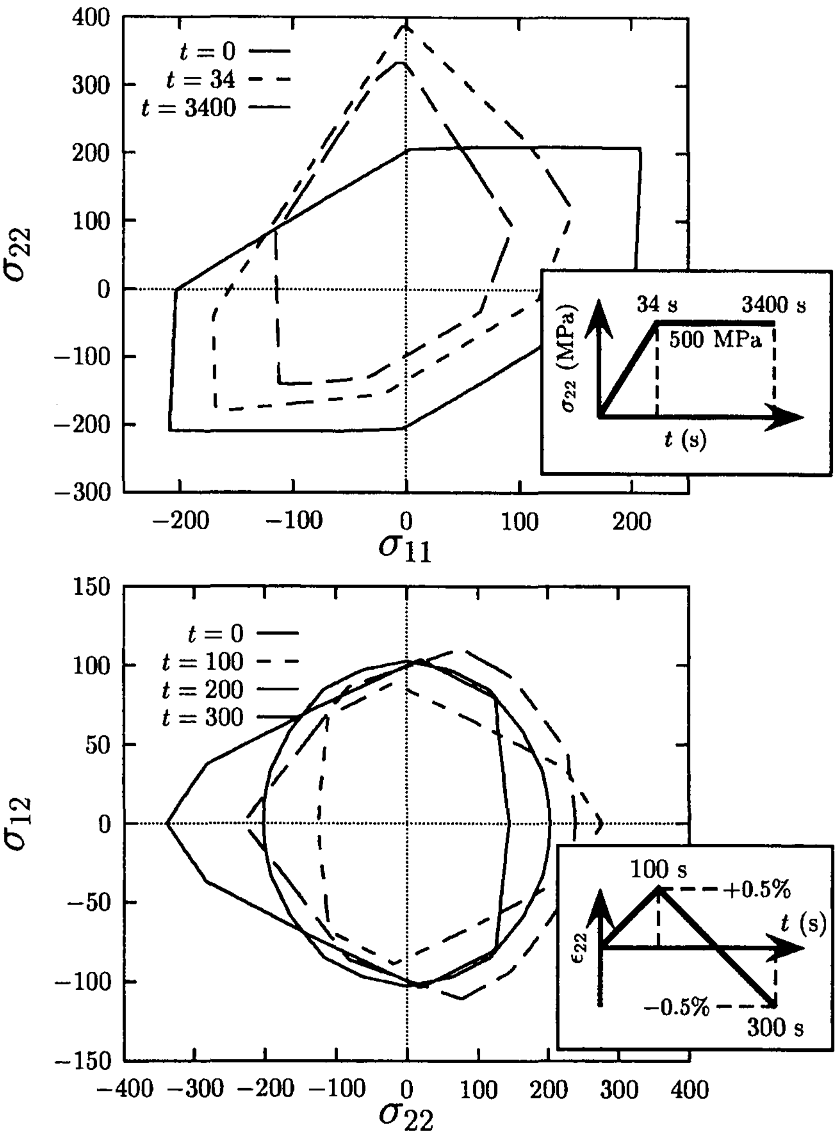


Figure 8. *Yield surfaces after deformation: (1) creep test under 500 MPa, (2) cyclic deformation $\epsilon = \pm 0.5\%$. The initial yield surface corresponds to the Tresca criterion*

variables names are used both in the program and for output purposes. As in the case of coefficients, arrays can be used (`@tVarInt alpha[0-N]`) It is also possible to define `primal` and `dual` variables. The pre–processor provides default names and consider that the behavior corresponds a small strain mechanical behavior: $\underset{\sim}{\sigma} \rightarrow$ `sig`, $\underset{\sim}{\varepsilon}_t \rightarrow$ `eto`.

@StrainPart. This directive allows the calculation, after integration, auxiliary variables and the material stiffness matrix $\underset{\sim}{D}$. In the present case, the stiffness matrix is set equal to the elasticity matrix. At the global level, it will be necessary to use a BFGS iterative method to solve the problem [BAT 82].

@Derivative. This directive is used to compute the time derivative of the different integrated variables. An explicit Runge–Kutta scheme, is then used to integrate the different equations. In the present case, variables to be integrated are: $\underset{\sim}{\epsilon}_e$, $\underset{\sim}{\alpha}$, ω and v. The pre–processor automatically generates variable names used to compute the time derivatives. For instance, the derivative of `alpha` is called `dalpha`. These variables also have the proper type: scalar or tensor. The imposed deformation rate $\dot{\underset{\sim}{\epsilon}}_t$ is computed as $\Delta\underset{\sim}{\epsilon}_t/\Delta t$ and is stored in a variable called `deto`. The constitutive equations can then be written using the different overloaded operators defined to handle tensors. As shown in Figure 9, 2D and 3D cases can be treated simultaneously. All coefficients can also be used as scalars. Using these functionalities, the time derivative of $\underset{\sim}{\alpha}$ can be written as:

```
dalpha = dep - (1.5*D/C/(1.-omega)*dv)*X;
```

which closely resembles

$$\dot{\underset{\sim}{\alpha}} = \dot{\underset{\sim}{\epsilon}}_p - \frac{3D}{2C(1-\omega)}\dot{v}\underset{\sim}{X}$$

@CalcCoeffs. This directive allows to compute the values of the material coefficients (they can depend on EP $\mathcal{V}_{\text{aux}}$ and $\mathcal{V}_{\text{int}}$). It should be used if the coefficients are planned not to be constant.

@CalcGradF. This directive is used to implement the implicit integration of the constitutive equation using a mid–point rule. In the case of the RBSC ceramic, the constitutive equations can be written over a time increment Δt as:

$$\begin{aligned}
\underset{\sim}{R}_e &= \Delta\underset{\sim}{\epsilon}_e - \Delta\underset{\sim}{\epsilon}_t - \Delta v\underset{\sim}{n} = \underset{\sim}{0} \\
R_v &= \Delta v - \left\langle \frac{J(\underset{\sim}{\alpha} - \underset{\sim}{X})}{K(1-\omega)} \right\rangle^n \Delta t = 0 \\
\underset{\sim}{R}_\alpha &= \Delta\underset{\sim}{\alpha} - (\underset{\sim}{n} - D\underset{\sim}{\alpha})\Delta v = \underset{\sim}{0} \\
R_\omega &= \Delta\omega - \left(\frac{X_I - X_0(1-\omega)}{A}\right)^r (1-\omega)^{-k}\Delta t = 0
\end{aligned}$$

This set of equations has to be solved with respect to the increment of the integrated variables: $\Delta\underset{\sim}{\varepsilon}_e$, Δv, $\Delta\underset{\sim}{\alpha}$, $\Delta\omega$. All variables (e.g. $\underset{\sim}{n}$, $\underset{\sim}{X}$, coefficients, ...) are computed at the mid-point $t + \theta\Delta t$ with $0 \leq \theta \leq 1$. A Newton–Raphson method is used to solve the equations: it is therefore necessary to compute the residual and the associated Jacobian matrix. The pre–processor defines variables corresponding to sub–pieces of the residual vector and associated to each integrated variable. This variable is called `R_<variable_name>` (e.g. `R_alpha`). The pre–processor also defines sub–matrices corresponding to parts of the Jacobian matrix associated with the partial derivative of one part of the residual with respect to one integrated variable: for instance the $\partial\underset{\sim}{R}_\alpha/\partial\Delta\underset{\sim}{\varepsilon}_e$ (called `dalpha_deel`).

$$[J] = \begin{array}{c|c|c|c|c}
\underset{\sim}{\varepsilon}_e & \underset{\sim}{\alpha} & v & \omega & \\
\vdots & & & & \underset{\sim}{\varepsilon}_e \\
\hline
\dfrac{\partial R_\alpha}{\partial\Delta\underset{\sim}{\varepsilon}_e} & \cdots & \cdots & \cdots & \cdots\underset{\sim}{\alpha} \\
\hline
 & & & & v \\
\hline
 & & & & \omega
\end{array}$$

Using the same notations as on Figure 9, the calculation of R_α is computed as:

```
R_alpha = dalpha - (n-D*alpha)*dv;
```

and $\partial\underset{\sim}{R}_\alpha/\partial\Delta\underset{\sim}{\varepsilon}_e$ as:

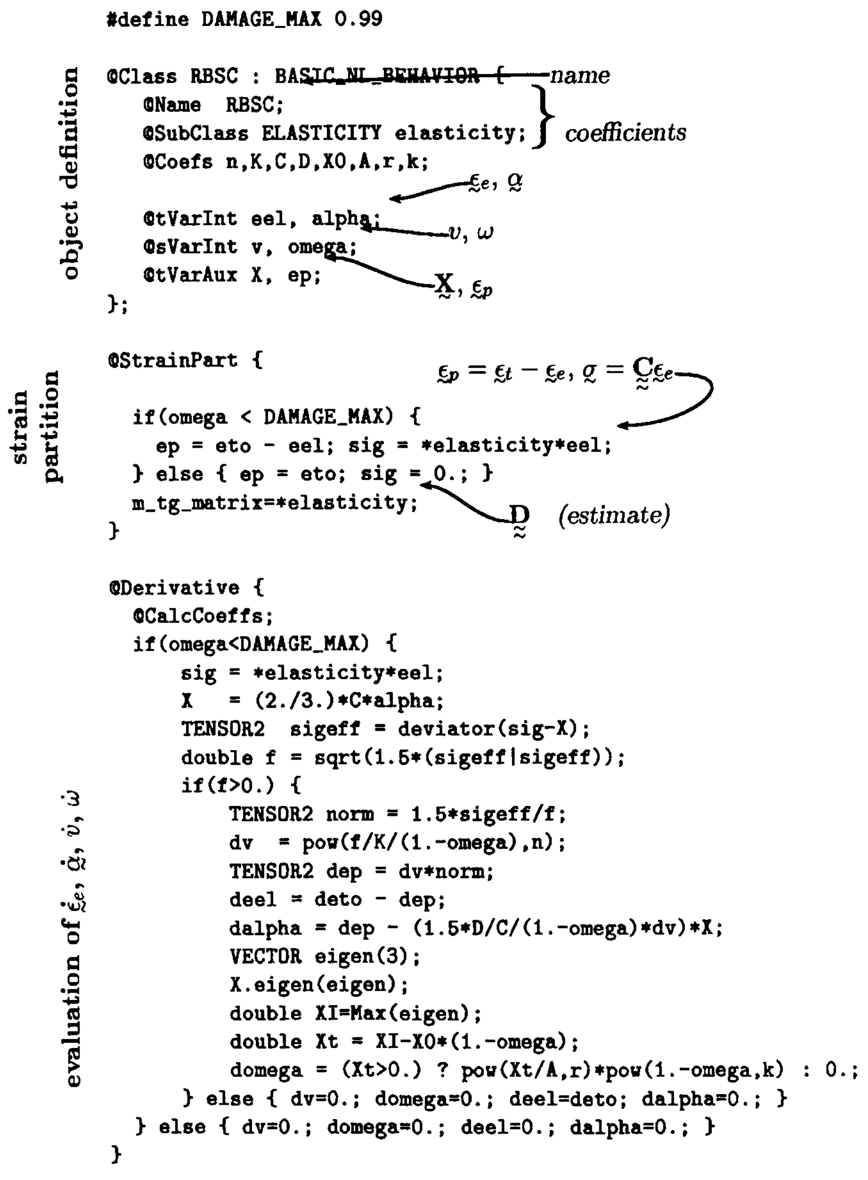
```
dalpha_deel = -(theta*dv)*N*(*elasticity);
```

where `N` is the fourth order tensor corresponding to $\underset{\approx}{N} = \partial\underset{\sim}{n}/\partial\underset{\sim}{\sigma}$.

6. Conclusion

The framework proposed by approaches using internal variables allows the user to build models able to represent a large class of material behaviors. It is especially shown in this paper that crystallographic (visco)plasticity can be treated exactly like classical (visco)plasticity using for instance von Mises criterion. The only difference between both models is that the number of internal variables is usually larger for the case of crystallographic approaches, leading to a longer vector of integrated variables, but the general structure of the models remains the same.

The objects needed to manage such a class of models have been developed. They characterize the elastic behavior (`ELASTICITY`), the type of hardening (`ISOTROPIC`, `KINEMATIC`, etc.), but also the criterion, the flow, etc., and can be combined to compose new models according to user's wishes. This can be made using the pre–processor code Zebfront, which offers the opportunity to



```
#define DAMAGE_MAX 0.99

@Class RBSC : BASIC_NL_BEHAVIOR {
   @Name  RBSC;
   @SubClass ELASTICITY elasticity;
   @Coefs n,K,C,D,X0,A,r,k;

   @tVarInt eel, alpha;
   @sVarInt v, omega;
   @tVarAux X, ep;
};

@StrainPart {

  if(omega < DAMAGE_MAX) {
    ep = eto - eel; sig = *elasticity*eel;
  } else { ep = eto; sig = 0.; }
  m_tg_matrix=*elasticity;
}

@Derivative {
  @CalcCoeffs;
  if(omega<DAMAGE_MAX) {
      sig = *elasticity*eel;
      X   = (2./3.)*C*alpha;
      TENSOR2  sigeff = deviator(sig-X);
      double f = sqrt(1.5*(sigeff|sigeff));
      if(f>0.) {
          TENSOR2 norm = 1.5*sigeff/f;
          dv   = pow(f/K/(1.-omega),n);
          TENSOR2 dep = dv*norm;
          deel = deto - dep;
          dalpha = dep - (1.5*D/C/(1.-omega)*dv)*X;
          VECTOR eigen(3);
          X.eigen(eigen);
          double XI=Max(eigen);
          double Xt = XI-X0*(1.-omega);
          domega = (Xt>0.) ? pow(Xt/A,r)*pow(1.-omega,k)  : 0.;
      } else { dv=0.; domega=0.; deel=deto; dalpha=0.; }
  } else { dv=0.; domega=0.; deel=0.; dalpha=0.; }
}
```

object definition
name
coefficients
$\underset{\sim}{\varepsilon}_e, \underset{\sim}{\alpha}$
v, ω
$\underset{\sim}{X}, \underset{\sim}{\varepsilon}_p$
strain partition
$\underset{\sim}{\varepsilon}_p = \underset{\sim}{\varepsilon}_t - \underset{\sim}{\varepsilon}_e, \underset{\sim}{\sigma} = \underset{\approx}{C}\underset{\sim}{\varepsilon}_e$
$\underset{\approx}{D}$ *(estimate)*
evaluation of $\dot{\underset{\sim}{\varepsilon}}_e, \dot{\underset{\sim}{\alpha}}, \dot{v}, \dot{\omega}$

Figure 9. *Code needed to implement the behavior for RBSC (Section* **4.2.***)*

implement new models with a minimum knowledge of C++ language. One should not also forget that, a `Multimat` option, is devoted to micro–macro approaches, and will produce new models directly from the material file, introducing elementary models for each phase, and combining them through predefined localisation–homogenization rules. As the mechanism is recursive, one can imagine to consider a `Multimat` material defined from several `Multimat` objects. In this last case, it can easily be understood that the user has to pay attention to the type of model he is going to produce. The "problem" connected with the robustness of the numerical implementation is that it will work, even if the model has no physical meaning. This is why a series of tools are offered to calibrate the model and compare simulations to experimental results.

These tools make a continuous link between experiment and structural calculations. The crucial point is that there is only one material file which is valid for the identification step and for the subsequent Finite Element calculation. It is then certain that the model used for describing the experimental work is the same as the model taken in the structure calculations, the integration procedures being exactly the same. Another important feature of the identification strategy is the use of a general optimizer, which can be disconnected from the remaining of the code. The optimization procedure can then be applied to numerical simulations of a volume element, but can also be treated as an inverse problem, involving Finite Element computations. Following the same idea, one can imagine optimization processes involving simultaneously the identification of the material parameters, the geometry, or friction coefficient, etc., all together.

New developments can still be made to produce better results. One is the implementation of a section including a symbolic computation engine, to avoid the calculations of the partial derivatives needed for the Newton integration scheme. On the other hand, the connection between the optimizer and the simulation processes can be improved, to increase the speed of the whole process. A graphical user interface might also help the user by offering the opportunity of a immediate verification of the effect of each of the coefficients.

7. References

[ASA 83] R.J. ASARO. "Crystal Plasticity". *J. Appl. Mech.*, 50:921–934, 1983.

[BÄC 96] T. BÄCK. "*Evolutionary Algorithms in Theory and Practice*". Oxford University Press, New York, 1996.

[BAT 82] K.J. BATHE. "*Finite element procedures in engineering analysis*". Prentice Hall, Inc., 1982.

[BER 79] M. BERVEILER A. ZAOUI. "An extension of the self–consistent scheme to plastically flowing polycrystals". *J. Mech. Phys. Solids*, 26:325–344, 1979.

[BES 98] J. BESSON R. FOERCH. "Application of object–oriented programming techniques to the finite element method. Part I— General concepts". *Revue europénne des éléments finis*, to be published, 1998.

[BRO 70] C.G. BROYDEN. "The convergence of a class of double-rank minimization algorithms 2: the new algorithm". *Journal Inst. of Math. and its Appl.*, 6:222–231, 1970.

[CAI 92] G. CAILLETAUD. "A Micromechanical Approach to Inelastic Behaviour of Metals". *"Int. J. of Plasticity"*, 8:55–73, 1992.

[CHA 89] J.-L. CHABOCHE. "Constitutive Equations for Cyclic Plasticity and Cyclic Viscoplasticity". *Int. J. of Plasticity*, 5:247–302, 1989.

[FRA 85] P. FRANCIOSI. "The concepts of latent hardening and strain hardening in metallic single crystals". *Acta. met.*, 33:1601–1612, 1985.

[GUT 97] M. GUTMANN-AMBROISE. *"Étude Expérimentale et Numérique du Comportement et de la Rupture à Chaud de Céramiques Techniques"*. PhD thesis, Ecole des Mines de Paris, 1997.

[HIL 65] R. HILL. "Continuum Micro-Mechanisms of Elastoplastic Polycrystals". *J. Mech. Phys. Sol.*, 13:89–101, 1965.

[HUT 76] J.W. HUTCHINSON. "Bounds and self-consistent estimates for creep of polycrystalline materials". *Proceedings Royal Society London A*, 348:101–127, 1976.

[KOC 66] U.F. KOCKS T.J. BROWN. "Latent hardening in aluminium". *Acta. met.*, 14:87–98, 1966.

[KRÖ 61] E. KRÖNER. "Zur plastischen Verformung des Vielkristalls". *Acta met.*, 9:155–161, 1961.

[MAN 65] J. MANDEL. "Une généralisation de la théorie de la plasticité de W.T. Koiter". *Int. J. Solids Struct.*, 1:273–295, 1965.

[MÉR 91] L. MÉRIC G. CAILLETAUD. "Single crystal modeling for structural calculations. Part 2: Finite element implementation". *Journal of Enginering Mat. Technol.*, 113:171–182, 1991.

[MÉR 94] L. MÉRIC, G. CAILLETAUD, M. GASPÉRINI. "F.E. calculations of coppe bicrystal specimens submitted to tension–compression tests". *Acta. met.*, 42(3):921–935, 1994.

[MOL 87] A. MOLINARI, R. CANOVA, S. AHZI. "A Self-Consistent Approach to the Large Deformation Polycrystal Viscoplasticity". *Acta met.*, 35:2983–2994, 1987.

[NEL 65] J.A. NELDER R. MEAD. "A simplex method for function minimization". *Computer Journal*, 7:308–313, 1965.

[NOU 95] D. NOUAILHAS, J.-P. PIERRE CULIÉ, G CAILLETAUD, L. MÉRIC. "F.E. Analysis of the Stress–Strain Behaviour of Single–Crystal Tubes". *Eur. J. Mech., A/Solids*, 14(1):137–154, 1995.

[PIL 94] P. PILVIN. "The Contribution of Micromechanical Approaches to the Modelling of Inelastic Behaviour". In A. PINEAU, G. CAILLETAUD, T. LINDLEY, editors, *Fourth International Conference on Biaxial/multiaxial Fatigue*, pages 31–46. ESIS, May 31–June 3, Saint–Germain, France 1994.

[SCH 86] K. SCHITTKOWSKI. *"QLD: A FORTRAN Code for Quadratic Programming, User's Guide"*. Mathematisches Institut, Universität Bayreuth, Germany, 1986.

[TAY 38] G.I. TAYLOR. "Plastic Strain in Metals". *J. Inst. Metals*, 62:307–324, 1938.

[ZHO 97] J.L. ZHOU A.L. LAWRENCE, C. Tits. "*User's Guide for CFSQP Version 2.5*". Institut for System Research TR-94-16r1, University of Maryland, College Park, MD 20742, 1997. USA, 1997.