UMAT4COMSOL: An Abaqus user material (UMAT) subroutine wrapper for COMSOL

Sergio Lucarini[a,*], Emilio Martínez-Pañeda[a]

[a]*Department of Civil and Environmental Engineering, Imperial College, London SW7 2AZ, UK*

Abstract

We present a wrapper that allows Abaqus user material subroutines (*UMAT*s) to be used as an *External Material* library in the software COMSOL Multiphysics. The wrapper, written in C language, transforms COMSOL's external material subroutine inputs and outputs into Fortran-coded Abaqus *UMAT* inputs and outputs, by means of a consistent variable transformation. This significantly facilitates conducting coupled, multi-physics studies employing the advanced material models that the solid mechanics community has developed over the past decades. We exemplify the potential of our new framework, UMAT4COMSOL, by conducting numerical experiments in the areas of elastoplasticity, hyperelasticity and crystal plasticity.

Keywords: Abaqus, COMSOL, Solid mechanics, Finite element method, User subroutine, External material

1. Introduction

Coupling multiple physical phenomena in continuum solids is a major research focus [1, 2]. The mechanical behaviour of structures and industrial components is relatively well understood in inert environments but challenges arise as a result of material-environment interactions. Multi-physics structural integrity problems such as hydrogen embrittlement [3, 4], corrosion [5, 6], and oxidation-assisted fatigue [7, 8] continue to challenge scientists and engineers. Predicting these structural integrity problems requires coupling state-of-the-art material models with equations describing chemical, electrical and thermal phenomena [9–13]. In addition, numerous modern devices and concepts involve multi-physics environments and mechanical loads, which are either applied externally or induced by other physical phenomena, such as thermal expansion, chemical strains or

*Corresponding author
Email address: s.lucarini@imperial.ac.uk (Sergio Lucarini)

magnetic forces. Examples of include Li-Ion batteries [14, 15], hydrogels [16], magnetorheological elastomers [17, 18], and piezo-electric and piezo-resistive materials [19, 20], to name a few. As a result, there is a pressing need to develop computational tools for characterising the coupled behaviour of materials in multiphysics environments [21–23].

Abaqus [24] and COMSOL [25] have arguably been the most popular finite element packages for materials modelling and multi-physics simulations, respectively. For decades, the solid mechanics community has developed Abaqus user material (*UMAT*) to numerically implement new and advanced material models; e.g., in the context of hyperelasticity [26], elastoplasticity [27], damage [28], and crystal plasticity [29, 30]. Abaqus *UMAT* subroutines are Fortran codes that describe constitutive material behaviour under either the small strains or finite strains convention. On the other hand, COMSOL stands out for its ability to handle coupled systems of partial differential equations, with multiple built-in physics modules that encompass most physical phenomena across the disciplines of chemistry, fluid flow, heat transfer, electromagnetism, structural mechanics and acoustics. Given the growing interest in material problems in multi-physics environments, there is great benefit in coupling these two tools. However, the COMSOL materials library does not currently support the automatic use of Abaqus *UMAT* subroutines as material models. Available materials in COMSOL are either built-in or can be programmed in C language as a user-defined *External Material* library. Therefore, our aim is to develop and share a wrapper that can take advantage of pre-programmed Abaqus *UMAT* material models and enable their use as *External Material* libraries in COMSOL. Both small and finite strains are considered, providing a general framework that can leverage the notable efforts made in materials model development with Abaqus *UMAT*s, while also taking advantage of the versatility of COMSOL for multi-physics simulations. The remainder of this paper is organised as follows. First, in Section 2, we proceed to describe the characteristics of the software that we have developed, UMAT4COMSOL. Then, in Section 3, usage instructions are provided. The potential of UMAT4COMSOL is demonstrated in Section 4 through examples encompassing small strains elastoplasticity, large strains hyperelasticity and crystal plasticity. Finally, concluding remarks end the manuscript in Section 5.

2. Software description: UMAT4COMSOL

UMAT4COMSOL is a C-coded subroutine that functions as a user-defined *External Material* in COMSOL, so as to describe the local mechanical behaviour of solid domains. The subroutine contains a call to an external Fortran-coded subroutine that corresponds to the Abaqus *UMAT*. During this process, the local state is inputted, and the constitutive equations describing the material behaviour are computed, with their outputs being returned to COMSOL for solving the multiphysics global problem.

UMAT4COMSOL's primary internal features are described below, starting with a concise explanation of the input requirements for both COMSOL *External Material*s and Abaqus *UMAT*s. Subsequently, the process of transferring input parameters from COMSOL *External Materials* to Abaqus *UMAT*s is elaborated upon. Finally, the output procedure, which involves a more detailed understanding of the various mathematical methods, is described.

2.1. Material models inputs and outputs

When modelling the mechanical behaviour of materials, the constitutive equations characterise the relation between the measure of the deformation and the stress state of the material. These equations can be linear, non-linear, time-dependent, and history-dependent.

The input to the material constitutive equations is typically the strain tensor ε_{ij}. In the small strain formalism, the strain tensor is defined in terms of the displacement vector u_i as

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) . \tag{1}$$

While the deformation gradient F_{ij}, given by

$$F_{ij} = \delta_{ij} + u_{i,j} , \tag{2}$$

is typically adopted when considering finite strains. Here, δ_{ij} denotes the Dirac delta function and the derivative definition in Eq. (2) is defined in the reference configuration. Another class of inputs are the material properties, which are typically constant values. Additionally, the constitutive equations can be posed as a function of other strain measures dependent on the aforementioned ones, such as the left Cauchy-Green or Green-Lagrange tensors. In the case of history-dependent

material behaviour, one should also consider a state variable vector α_i, that stores history information of the state of the material. Finally, in time-dependent problems, the time increment value is also stored, so as to calculate the strain rates measures and the evolution of state variables.

The outcome of the constitutive calculations is a calculated stress measure, which will be translated to forces at the element level. In the case of non-linear analyses, the mechanical equilibrium equation is usually linearised, requiring the definition of a consistent tangent modulus, which corresponds to the derivative of the adopted stress measure with respect to the adopted strain measure.

2.2. COMSOL External Material

COMSOL *External Material*s are C-coded subroutines that are calculated at the Gauss point level on each iteration using a subroutine called `eval`. The inputs include the strain/deformation gradient trials `E`/`Fl`, along with the properties vector `par` and deformation gradient at the previous time increment `FlOld`. The (pseudo)time steps/increments are defined in the `delta` variable, and a set of state variables are passed as a vector `state`. Once the constitutive equations are calculated in the core code, the results obtained for each (pseudo)time step/increment are passed using the stress vector `S`, the consistent tangent `D`/`Jac`, and the evolved state variables `state`.

2.3. Abaqus user material (UMAT) subroutine

Abaqus *UMAT*s are Fortran-coded subroutines that calculate the constitutive equations at the Gauss point level on every iteration. The inputs for the mechanical problem include the trial incremental quantities of strain/deformation trials at the current increment (`DSTRAN`/`DFGRD1`), the total strain/deformation measures at the end of the previous increment (`STRAN`/`DFGRD0`), the stresses at the previous time increment `STRESS`, and the material properties, which are contained in the `PROPS` vector. The (pseudo)time increments are defined in the `DTIME` variable, together with the current (pseudo)time value (`TIME`), and a set of state variables is passed as a vector in `STATEV`. Also, some constitutive equation in finite strains needs the incremental rotation tensor variable `DROT`. After calculating the constitutive equations, the key outputs at the end of the (pseudo)time increment are the updated stress vector `STRESS`, the consistent tangent matrix `DDSDDE`, and the updated state variables vector `STATEV`.

2.4. Transfer of inputs

The code initially transforms COMSOL's input variables to conform to the convention of Abaqus *UMAT* subroutines, which requires a change in the strain format. Specifically, two aspects must be taken care of in small strain problems. One is that Abaqus stores shear strains as engineering shear strains; $\gamma_{xy} = 2\varepsilon_{xy}$. A second one is that the ordering of the shear components is different. Accordingly, one must change the COMSOL input strain tensor (`E` variable) to fit the *UMAT* convention, such that

$$\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{yz}, \varepsilon_{xz}, \varepsilon_{xy} \rightarrow \varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \gamma_{xy}, \gamma_{xz}, \gamma_{yz} \tag{3}$$

where $\gamma_{\square} = 2\varepsilon_{\square}$ and the left-hand side corresponds to the COMSOL notation. Here, one should note that COMSOL's `E` vector contains the total trial strains. Consequently, the incremental trial strains vector, `DSTRAN`, is computed by subtracting from `E` the total strain at the end of the previous increment, which has been stored as state variables and corresponds to the variable `STRAN` in the *UMAT* convention. Moreover, the stress at the previous time increment is saved in the state variable vector and assigned to the *UMAT* variable `STRESS` before computing the constitutive equations. For the finite strains version, the *UMAT* requires the deformation gradient F_{ij} before and after the (pseudo)time increment. COMSOL provides the deformation gradients `Fl0ld` and `Fl` to the external material subroutine as a C-order matrix. These matrices are transformed into a Fortran-order matrix and passed to the *UMAT* as `DFGRD0` and `DFGRD1`. Additionally, the auxiliary variables of incremental strain `DSTRAN` ($\Delta\varepsilon$) and incremental rotation `DROT` (ΔR) are computed via $\Delta\varepsilon_{ij} = 1/2\left(\dot{F}_{ip}F^{-1}_{t+\Delta t\ pj} + F^{-1}_{t+\Delta t\ pi}\dot{F}_{jp}\right)$ (defining $\dot{F}_{ij} = F_{t+\Delta t\ ij} - F_{t\ ij}$ and forming a vector by using Eq. 3) and $\Delta F_{ij} = \Delta R_{ip}\Delta U_{pj}$ (standing for the polar decomposition of $\Delta F_{ij} = F_{t+\Delta t\ ip}F^{-1}_{t\ pj}$), respectively. Finally, the values of the current (pseudo)time and (pseudo)time increment are obtained by making use of COMSOL's `delta` variable and updating a state variable. The COMSOL variable `delta` contains the current time increment and thus corresponds to the *UMAT* variable `DTIME`. Since the *UMAT* subroutine also takes as input the total time at the beginning of the increment (`TIME` variable), this is stored as the first component of the `STATEV` vector, which is updated on every increment based on `delta`.

2.5. Transfer of outputs

The stress vector and consistent tangent matrix are the outputs of the *UMAT*s. In the small strain version, the stress `STRESS` can be directly transformed by rearranging stress vector shear

components, as in Eq. (3), to match the required output format for the COMSOL solver (`S` variable). In a similar fashion, the consistent tangent matrix `DDSDDE` is reordered to match the arrangement of the shear components (as per Eq. 3) and then transposed to convert from a Fortran-ordered matrix to a C-ordered matrix `D`.

For the finite strain case, the outputs of the *UMAT* subroutine (stresses and consistent tangent matrix) need to be transformed before being transferred. In regards to the stresses, Abaqus *UMAT* outputs the Cauchy stress tensor σ_{ij} in the variable `STRESS` (as a vector, using Voigt notation), while COMSOL works with the second Piola-Kirchhoff stress S_{ij}, using the variable `S`. The transformation, `STRESS` $\rightarrow$ `S`, is the defined using the following relation

$$S_{ij} = JF_{iq}^{-1}\sigma_{qp}F_{jp}^{-T} \ , \tag{4}$$

where $J = \det(F_{ij})$.

As described below, the transformation of the consistent tangent tensor (`DDSDDE` $\rightarrow$ `Jac`) in a finite strains context requires careful consideration. This is also a key step to guarantee the appropriate convergence rate. The material Jacobian obtained from the Abaqus *UMAT* user subroutine at finite strains is the tangent modulus tensor for the Jaumann rate of the Kirchhoff stress, denoted by C_{ijkl}^{Abaqus}, which is a fourth-order tensor defined as

$$\overset{\nabla}{\tau}_{ij} = \dot{\tau}_{ij} - w_{ip}\tau_{pj} - \tau_{ip}w_{jp} = J\ C_{ijkl}^{\text{Abaqus}}d_{kl} \tag{5}$$

where τ_{ij} denotes the Kirchhoff stress, ∇ is the Jaumann rate, and w_{ij} and d_{ij} are the spin tensor and stretch tensor, respectively. In contrast, the COMSOL framework requires the material tangent, K_{ijkl}^{COMSOL}, which is given by

$$\dot{S}_{ij} = K_{ijkl}^{\text{COMSOL}}\dot{F}_{kl} \ . \tag{6}$$

We shall then derive an explicit expression that relates the two fourth-order tensors, C_{ijkl}^{Abaqus} and K_{ijkl}^{COMSOL}. Combining Eqs. (4) and (6), one reaches

$$K_{ijkl}^{\text{COMSOL}} = \frac{\partial S_{ij}}{\partial F_{kl}} = \frac{\partial\left(JF_{iq}^{-1}\sigma_{qp}F_{jp}^{-1}\right)}{\partial F_{kl}} = \frac{\partial\left(F_{iq}^{-1}\tau_{qp}F_{jp}^{-1}\right)}{\partial F_{kl}}. \tag{7}$$

Expanding the derivative of the product in Eq. (7) and expressing it in index notation, we obtain:

$$K_{ijkl}^{\text{COMSOL}} = \frac{\partial S_{ij}}{\partial F_{kl}} = \frac{\partial F_{iq}^{-1}}{\partial F_{kl}}\tau_{qp}F_{jp}^{-1} + F_{iq}^{-1}\frac{\partial \tau_{qp}}{\partial F_{kl}}F_{jp}^{-1} + F_{iq}^{-1}\tau_{qp}\frac{\partial F_{jp}^{-1}}{\partial F_{kl}} \tag{8}$$

This equation expresses the components of the COMSOL material tangent, $K_{ijkl}^{\mathrm{COMSOL}}$, as a function of components that can be obtained using the definition of the Abaqus tangent modulus tensor $C_{ijkl}^{\mathrm{Abaqus}}$, see Eq. (5). The first and third terms in Eq. (8) are the derivatives of the inverse of a tensor with respect to itself: $\partial F_{jp}^{-1}/\partial F_{kl} = -F_{lp}^{-1}F_{jk}^{-1}$. To obtain the second term in Eq. (8), we linearise the rate quantities from Eq. (5) by multiplying by an infinitesimal time increment and reordering terms, yielding

$$\delta\tau_{ij} = J\ C_{ijkl}^{\mathrm{Abaqus}}\delta d_{kl} + \delta w_{ip}\tau_{pj} + \tau_{ip}\delta w_{jp}\ , \tag{9}$$

where δd_{ij} and δw_{ij} are obtained as functions of F_{ij} and δF_{ij} via

$$\delta d_{ij} = \frac{1}{2}\delta F_{ip}F_{pj}^{-1} + \frac{1}{2}\delta F_{jp}F_{pi}^{-1}\ ,\ \delta w_{ij} = \frac{1}{2}\delta F_{ip}F_{pj}^{-1} - \frac{1}{2}\delta F_{jp}F_{pi}^{-1}\ . \tag{10}$$

The Kirchhoff stress is also linearised with respect to the perturbation of the deformation gradient as

$$\delta\tau_{ip} = \frac{\partial\tau_{ip}}{\partial F_{kl}}\delta F_{kl}. \tag{11}$$

where $\delta\tau_{ij}$ and δF_{ij} are the Kirchhoff stress and deformation gradient perturbations, respectively. Inserting the expressions of Eq. (10) into Eq. (9), we obtain

$$\begin{aligned}\delta\tau_{ip} = &\frac{J}{2}C_{ipkm}^{\mathrm{Abaqus}}\delta F_{kl}F_{lm}^{-1} + \frac{J}{2}C_{ipmk}^{\mathrm{Abaqus}}F_{lm}^{-1}\delta F_{kl} + \\ &\frac{1}{2}I_{ipkq}\delta F_{kl}F_{lm}^{-1}\tau_{mq} - \frac{1}{2}I_{ipmq}F_{lm}^{-1}\delta F_{kl}\tau_{kq} + \\ &\frac{1}{2}I_{ipqk}\tau_{qm}F_{lm}^{-1}\delta F_{kl} - \frac{1}{2}I_{ipqm}\tau_{qk}F_{lm}^{-1}\delta F_{kl}\end{aligned} \tag{12}$$

which we can be used to compare the terms of Eqs. (11) and (12) and obtain $\partial\tau_{ip}/\partial F_{kl}$. By considering the minor symmetries of $C_{ijkl}^{\mathrm{Abaqus}}$ and the symmetry of τ_{ij}, we can simplify the expression and arrive to the following definition of $\partial\tau_{ip}/\partial F_{kl}$

$$\frac{\partial\tau_{ip}}{\partial F_{kl}} = J\ C_{ipkm}^{\mathrm{Abaqus}}F_{lm}^{-1} + \frac{1}{2}\delta_{ik}F_{lm}^{-1}\tau_{mp} - \frac{1}{2}F_{li}^{-1}\tau_{kp} + \frac{1}{2}\delta_{pk}F_{lm}^{-1}\tau_{im} - \frac{1}{2}F_{lp}^{-1}\tau_{ik}\ . \tag{13}$$

The relation between the COMSOL and Abaqus material tangents is then obtained by inserting Eq. (13) into Eq. (8), rendering

$$\begin{aligned}K_{ijkl}^{\mathrm{COMSOL}} = &-F_{lq}^{-1}F_{ik}^{-1}\tau_{qp}F_{jp}^{-1} + F_{iq}^{-1}\Big(J\ C_{qpkm}^{\mathrm{Abaqus}}F_{lm}^{-1} + \frac{1}{2}\delta_{qk}F_{lm}^{-1}\tau_{mp} \\ &- \frac{1}{2}F_{lq}^{-1}\tau_{kp} + \frac{1}{2}\delta_{pk}F_{lm}^{-1}\tau_{qm} - \frac{1}{2}F_{lp}^{-1}\tau_{qk}\Big)F_{jp}^{-1} - F_{iq}^{-1}\tau_{qp}F_{lp}^{-1}F_{jk}^{-1}\ .\end{aligned} \tag{14}$$

The resulting COMSOL consistent tangent can then be transformed to Voight notation and transposed from Fortran-order to C-order matrix, which is the required output to be used by the COMSOL solver.

3. Usage instructions

This section provides a brief description of the source code assembly, which aims to aid in understanding the software and enable personalised development. The relevant operations to use the wrapper are also described. UMAT4COMSOL is made freely available to download at `https://github.com/sergiolucarini` and `www.empaneda.com/codes`

3.1. Organization of the source code

UMAT4COMSOL is a C code that defines the `eval` function required in COMSOL for user-defined *External material*s. Two versions are available: a small strain version and a finite strain version. Both versions have a similar structure. One of the key elements of the code is an external *UMAT* function, which is linked to the Fortran *UMAT* subroutine. The first part of the code handles this *UMAT* function, which is followed by the standard header for a COMSOL user-defined external material subroutine, as per the COMSOL reference manual. Then, input variables are transformed using the operations described in Section 2.4. The core code calls the Abaqus *UMAT* subroutine to compute the constitutive equations, and the last part of the code converts the output data, as explained in Section 2.5, which is then passed to the COMSOL solver.

3.2. Main usage

The main file is a wrapper in C language called `UMAT4COMSOL.c`. Once compiled, it can be used as a user-defined *External Material* library in COMSOL. For the compilation process, the `UMAT4COMSOL.c` and `umat.f` files should be in the same folder. The following commands can be used to compile the code:

```
gfortran -c umat.f
gcc -c UMAT4COMSOL.c
gcc -shared -o extmat.dll umat.o UMAT4COMSOL.o -lgfortran -lquadmath
```

Here, the compilation process uses open-source compilers, but other Fortran and C compilers can also be used as long as the equivalent compiling flags are used. The above commands generate two objects and a dynamic library called `extmat.dll` which is required for running simulations in COMSOL. If Linux is used, the extension of the library should be ".so" (i.e., `extmat.so`).

To run non-linear simulations with external materials in COMSOL, one can use either the *Time-Dependent Solver* or *Stationary Solver*. For small strains, the material should be introduced as a *General stress-strain* relation, and for finite strains, as a *General stress-deformation* relation. Then, the properties vector needs to be included in the Material model parameters vector, and the number of state variables and initialisation values of the vector must be given. Finally, the path to the compiled dynamic library (`extmat.dll`) must be entered in the *External Material Library*'s field.

To use external materials in the *Solid Mechanics* module, an *External Stress-Strain* relation node must be created for each phase in the domain and linked to the corresponding *External Material* created. After setting the boundary conditions and solver settings, the simulation is ready to be run and post-processed.

3.3. Sofware aspects

As stated previously, this wrapper relies on the definition of a COMSOL *External Material* library. A comprehensive scheme of the functioning of the framework is shown in Figure 1. The procedure imports a compiled object, that encompasses Fortran and C code. The compiler choice is up to the user, taking into account the particularities of each compiler.

Once the compiled library is associated with the COMSOL input file, the wrapper will act as any material in COMSOL, enabling the use of the advanced functionalities that COMSOL offers. In the most simple case of non-linear analysis, COMSOL will access twice the *External Material* library, once for obtaining the stress and a second time for obtaining the consistent tangent. This makes the COMSOL analysis less efficient compared to Abaqus since the latter is optimised to extract both at once.

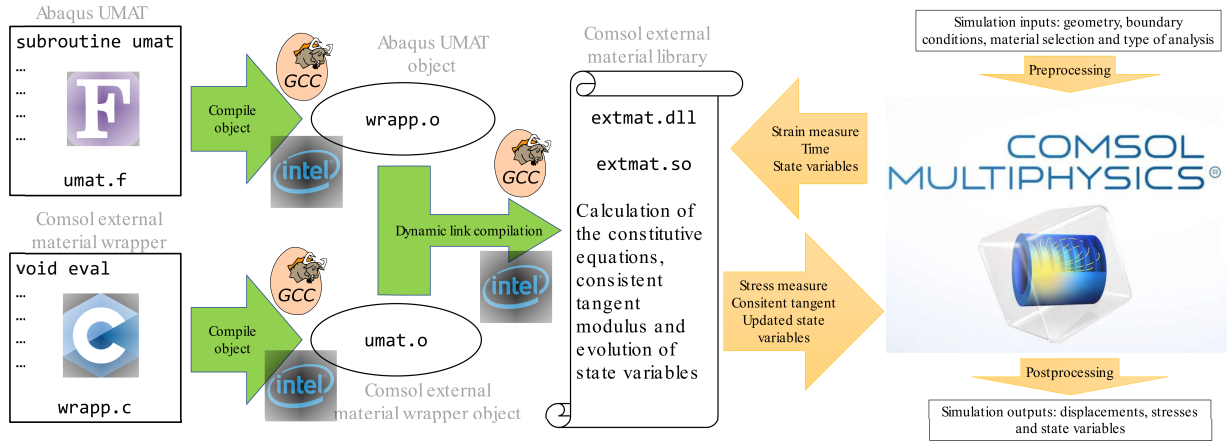

Figure 1: Flow chart of UMAT4COMSOL

For the postprocessing modules, COMSOL obtains the non-saved information from the calculation of the constitutive equations of the *External Material* by using the state variables saved on the last computed state, leading often to the failure for retrieving the stress fields. Nevertheless, the stress vector can be recovered from the saved state variables, needed for supplying the stress at the previous time increment to the Abaqus *UMAT*.

4. Representative results and applications

To demonstrate the capabilities of UMAT4COMSOL, we provide representative results in the context of three areas of particular interest: elastoplasticity, large strain hyperelasticity and crystal plasticity; validating the outputs against Abaqus calculations. Thus, we first predict the behaviour of an elastoplastic plane stress holed plate undergoing small strains (Section 4.1). Then, in Section 4.2, we verify the finite strains implementation by considering a neo-Hookean hyperelastic model. And finally, in Section 4.3, we employ an advanced crystal plasticity model to showcase UMAT4COMSOL potential in handling complex constitutive equations. These are just three examples of the capabilities of UMAT4COMSOL but its applicability is universal, as it provides a non-intrusive connection between COMSOL and any material models available as Abaqus *UMAT*s.

Of interest to all of these studies is the description of the tolerance criteria employed in Abaqus and COMSOL, as well as their comparison. Consider first the case of Abaqus. The tolerances for non-linear analysis in Abaqus are expressed in terms of maximum residual forces, such that

equilibrium is achieved when this value is below a tolerance $\text{tol}^{\text{Abaqus}}$. The inequality reads as

$$\text{tol}^{\text{Abaqus}} > \frac{r^{max}}{\widetilde{q}} \ , \tag{15}$$

where r^{max} is the maximum value of the residual force vector and $\widetilde{q}$ is the overall time-averaged value of the spatially averaged force over the entire model. On the other hand, COMSOL considers that convergence has been achieved when the relative tolerance $\text{tol}^{\text{COMSOL}}$ exceeds the relative error computed as the weighted Euclidean norm. That is,

$$\text{tol}^{\text{COMSOL}} > \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{|E_i|}{\widetilde{W}}\right)^2} \ , \tag{16}$$

where E_i is the estimated solution error vector, N is the number of degrees of freedom and $\widetilde{W}$ is the weight, which is determined by comparing the solution value at the specific degree of freedom with the average value of the solution and selecting the maximum of the two. For the numerical benchmarks a non-linear tolerance of $\text{tol}^{\text{Abaqus}} = 5 \times 10^{-3}$ and $\text{tol}^{\text{COMSOL}} = 1 \times 10^{-3}$ has been set for the Abaqus and COMOSOL solvers respectively. These are the default tolerances for both solvers.

4.1. Small strain elastoplastic model

In this example, we simulate the mechanical behaviour of a rectangular plate with a circular hole that is undergoing plastic deformation. The geometry and loading conditions are similar to the benchmark problem presented in Ref. [31]. The plate has a width of 36 mm and a height of 20 mm, and contains a hole of 5 mm radius at its centre. Due to symmetry, only one quarter of the model is simulated, with symmetry boundary conditions being applied on the left and bottom edges. The thickness of the plate is much smaller than the other dimensions, and the loads are confined to the plate plane, so plane stress conditions are assumed. The right edge of the plate is subjected to a traction that ramps linearly up to a maximum of 133.65 MPa. The plate is made of an isotropic elastoplastic material whose elastic response is characterised by a Young's modulus of $E = 70$ GPa and a Poisson's ratio of $\nu = 0.2$. The material response is assumed to be characterised von Mises plasticity, with a yield stress of $\sigma_y = 243$ MPa. Accordingly, the total strains are additively decomposed into elastic and plastic parts, $\varepsilon_{ij} = \varepsilon^e_{ij} + \varepsilon^p_{ij}$, and the Cauchy stresses are given by,

$$\sigma_{ij} = C^e_{ijkl}\varepsilon^e_{kl} \tag{17}$$

where C^e_{ijkl} is the elastic isotropic stiffness tensor. Linear isotropic hardening is assumed, with a tangent modulus of $h = 2171$ MPa. Accordingly, the yield condition is given by,

$$f\left(\sigma_{ij}\right) = \sqrt{\frac{3}{2} s_{ij} s_{ij}} - \left(\sigma_y + h\varepsilon^p\right) \tag{18}$$

where s_{ij} is the deviatoric part of the Cauchy stress, $s_{ij} = \mathrm{dev}\left(\sigma_{ij}\right)_{ij}$, and ε^p is the equivalent plastic strain, $\varepsilon^p = \int \sqrt{2/3\ \dot{\varepsilon}^p_{ij} \dot{\varepsilon}^p_{ij}}\ \mathrm{d}t$. The rate of the latter evolves as,

$$\dot{\varepsilon}^p_{ij} = \dot{\lambda} \frac{3}{2\sqrt{3\ s_{pq} s_{pq}}} s_{ij} \tag{19}$$

with $\dot{\lambda}$ being the plastic multiplier. The loading path is divided into 22 constant increments (using the *Stationary Solver* in COMSOL). The material constants are introduced as properties in the COMSOL *External Material* library and the plastic strain tensor is considered a state variable, initialised with zeros. Finite element calculations are run in both Abaqus and COMSOL using the same user material (*UMAT*) subroutine, with the COMSOL job exploiting the UMAT4COMSOL wrapper. The results obtained are given in Figs. 2 and 3.

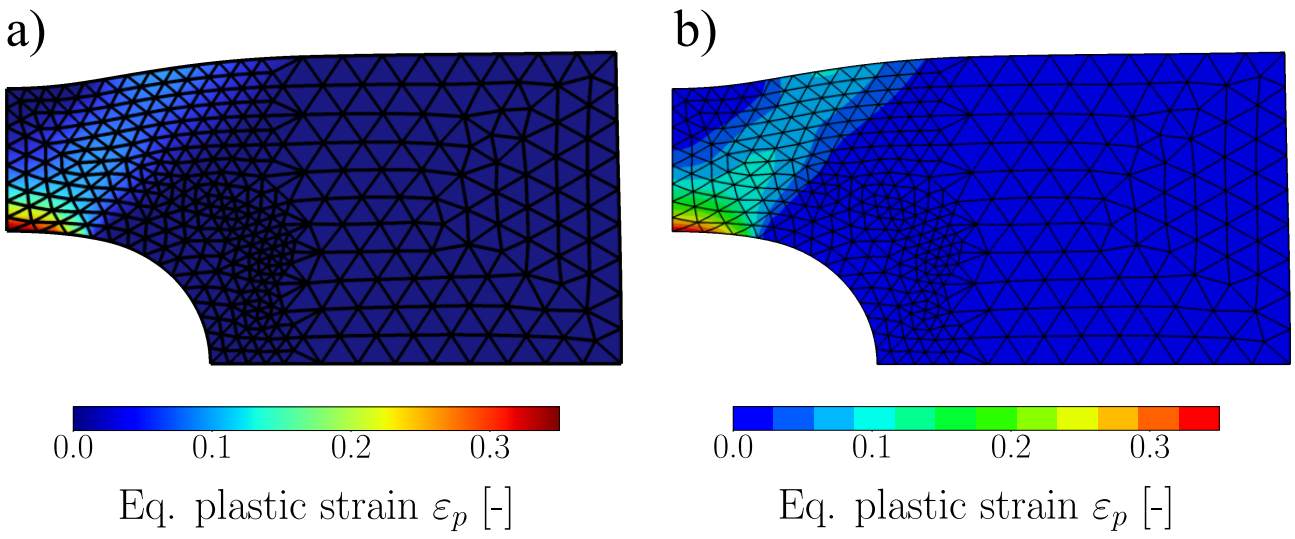

Figure 2: Mechanical response of an elastoplastic holed plate: contour plots of the equivalent plastic strain. Results obtained: (a) with Abaqus, using an elastoplastic *UMAT*, and (b) with COMSOL, using the same *UMAT* and UMAT4COMSOL. The deformation has been scaled by a factor of 20.

The equivalent plastic strain contours obtained with COMSOL and Abaqus appear to be indistinguishable, see Fig. 2. The displacement solution difference between the two approaches is less than 0.6% in terms of relative L2-norm, indicating that the results obtained with Abaqus and

COMSOL are identical. The perfect agreement obtained is also evident Fig. 3a, which shows the engineering stress-strain curves obtained with both packages.

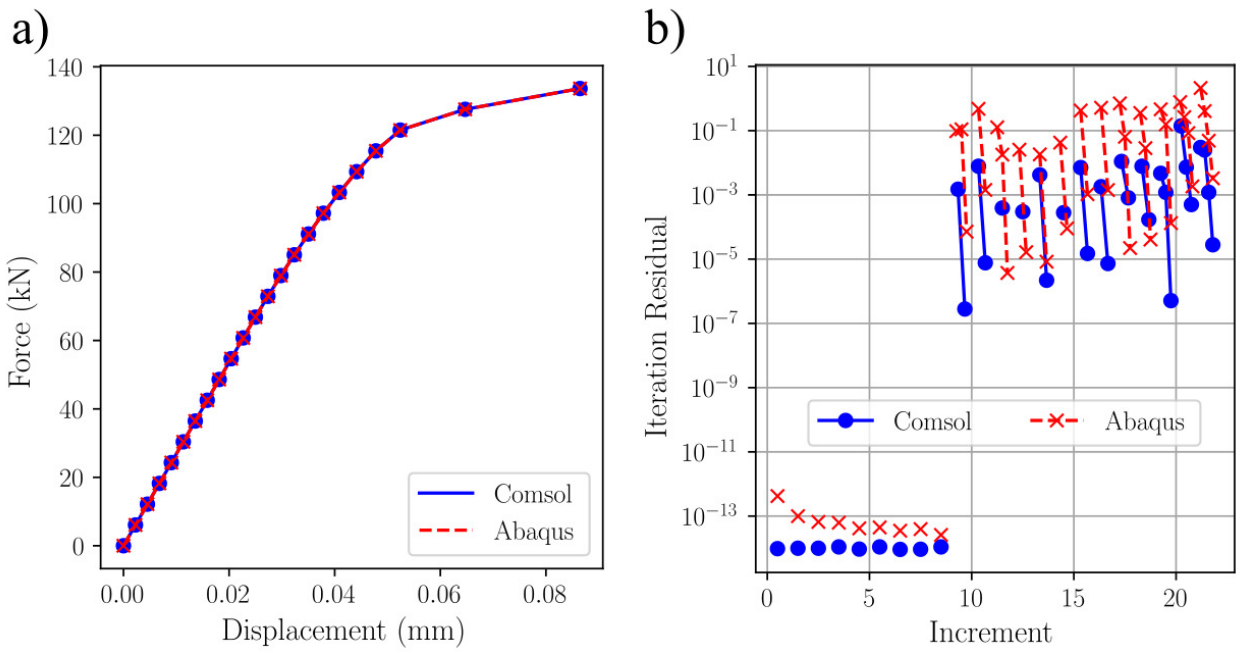

Figure 3: Mechanical response of an elastoplastic holed plate: (a) Force versus displacement response, predicted with Abaqus (+*UMAT*) and with COMSOL (+*UMAT* and UMAT4COMSOL); and (b) convergence plots for both COMSOL and Abaqus solvers, showing the magnitude of the residual for each iteration as a function of the load increment.

The convergence rates are given in Fig. 3b. Specifically, the magnitudes of the residuals are provided for each iteration as a function of the load/time increment. As shown in Fig. 3b, after approximately ten load increments, two iterations are needed to achieve convergence in every increment. This is the same for both Abaqus and COMSOL, and the number of iterations is identical, demonstrating that both provide a similar convergence rate. Some differences are seen in the magnitude of the residuals but this should be considered with care as COMSOL uses the L2-norm of the residual vector to evaluate the convergence while Abaqus uses the ∞-norm of the out-of-balance forces.

4.2. Finite strain neo-Hookean hyperelastic model

The case study aims at verifying the large strain implementation and showcasing the use of UMAT4COMSOL with another material model: non-linear hyperelasticity. To this end, the behaviour resulting from twisting a three-dimensional cube made of a neo-Hookean material is

studied. The characteristic length of the cube is 1 m, as in the hyperelasticity example presented in Ref. [32]. The constitutive equation of the neo-Hookean hyperelastic model is given by,

$$\sigma_{ij} = \frac{1}{J}\frac{E}{2\,(1+\nu)}\left(F_{iq}F_{jq} - \frac{1}{3}\delta_{ij}F_{kq}F_{kq}\right) + \frac{E}{3\,(1-2\nu)}\,(J-1)\,\delta_{ij} \tag{20}$$

and the material parameters used are $E = 10^6$ Pa and $\nu = 0.3$. These are introduced as properties to the COMSOL *External Material* library. The unitary cuboid is twisted by 60 degrees, and the boundary conditions are given, on one face, by Dirichlet boundary conditions of rotation with respect to the centre and, on the opposite face, by symmetry boundary conditions. Two loading cases are studied: one with a single load increment and another one splitting the load into 10 constant load increments. Calculations are conducted using stationary solvers in both Abaqus and COMSOL.

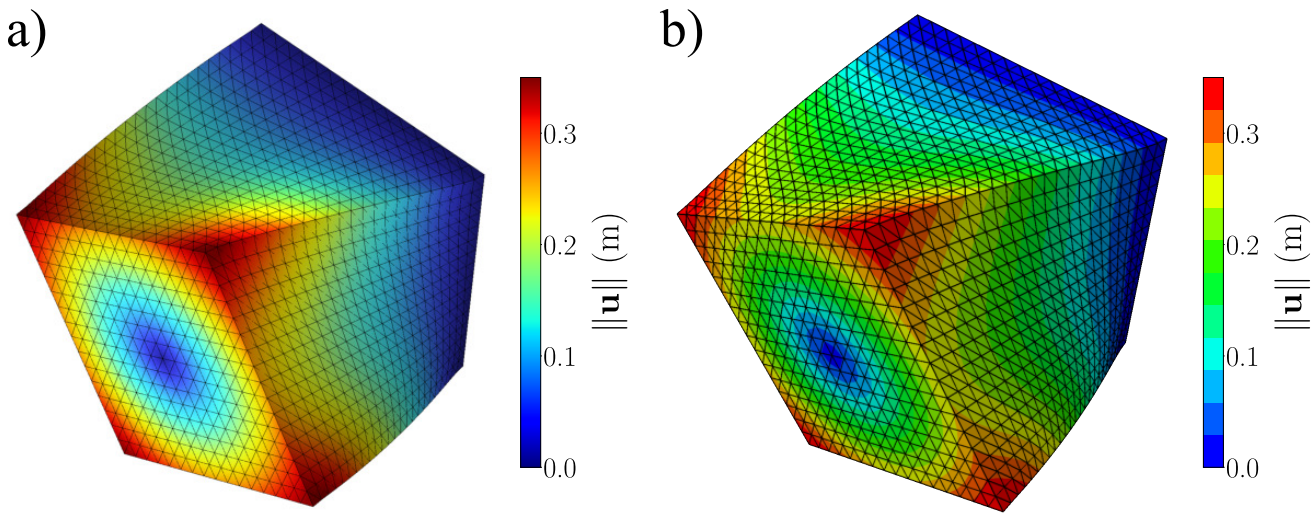

Figure 4: Mechanical response of a twisted neo-Hookean cube: contour plots of the displacement field magnitude (L2 norm). Results obtained: (a) with Abaqus, using a non-linear hyperelastic *UMAT*, and (b) with COMSOL, using the same *UMAT* and UMAT4COMSOL.

The predicted deformed shapes of the twisted cube are given in Fig. 4, showing the contours of the displacement field magnitude. Again, no differences are observed in the results. The contours are also insensitive to the number of load increments employed. Importantly, the relative error in the displacement solution between the Abaqus- and COMSOL-based calculations is below 0.3%, as measured by the L2-norm. Moreover, this difference does not increase when using only one pseudo-time increment.

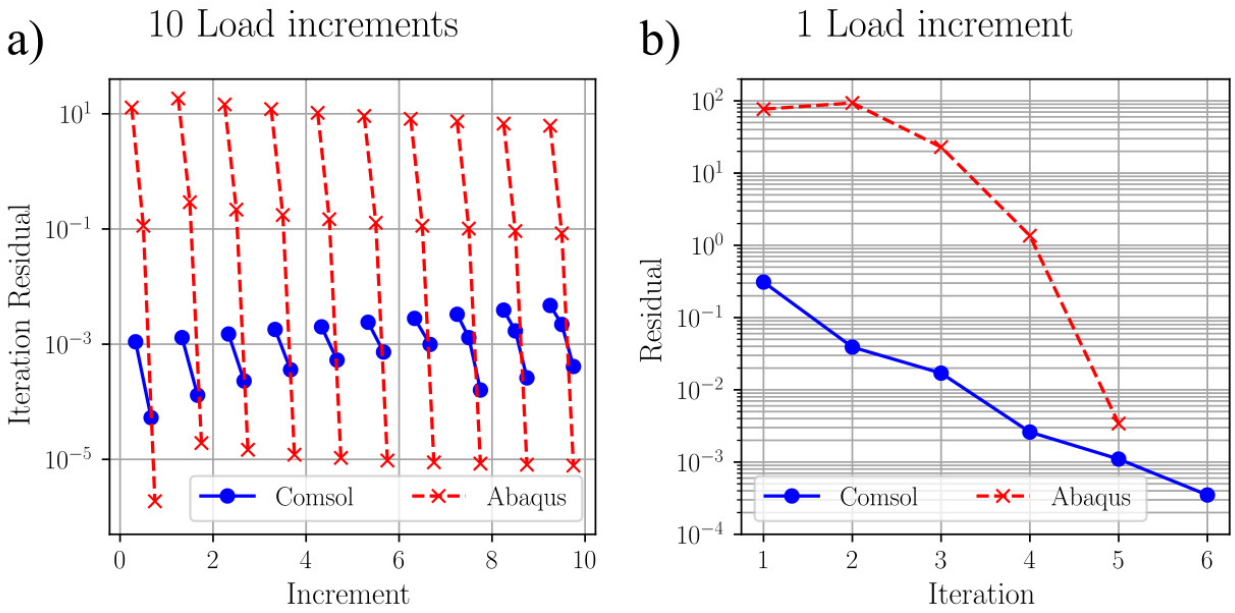

Figure 5: Convergence plots of twisting cuboid simulation using a neo-Hookean *UMAT* material in both solvers, Abaqus and COMSOL. Subfigure (a) shows the residual evolution in 10 pseudo-time increments, while subfigure (b) provides convergence plots for 1 pseudo-time increment.

The non-linear convergence of both solvers is comparable, as shown in Fig. 5, where the magnitude of the residual for each iteration is provided for both the 10-increment (Fig. 5a) and the 1-increment (Fig. 5b) studies. In both loading cases, COMSOL and Abaqus require a similar number of solver iterations to obtain a converged solution. As in the previous case study (Section 4.1), some quantitative differences can be observed in the magnitude of the residuals, which are intrinsically related to their definition.

4.3. Finite strain crystal plasticity

The last benchmark deals with the deformation of a metallic polycrystal where the material response is characterised using a crystal plasticity material model [33]. This benchmark has been chosen for mainly two reasons: (i) its complexity, as the crystal plasticity *UMAT* involves a large number of state variables and exhibits dependencies on loading path and strain rate, and (ii) its relevance, as it is not currently possible to run crystal plasticity studies in COMSOL despite its importance in many multi-physics problems. The chosen crystal plasticity material model is elastic-viscoplastic and assumes that plastic deformation only occurs by plastic shear along the slip systems. The deformation gradient is multiplicatively decomposed into elastic and plastic parts,

$$F_{ij} = F^e_{ip} F^p_{pj} \tag{21}$$

For a slip system α, the plastic slip rate ($\dot{\gamma}^\alpha$) is given in terms of the resolved shear stress (τ^α) through the following phenomenological power-law,

$$\dot{\gamma}^\alpha = \dot{a} f \frac{\tau^\alpha}{\tau_c^\alpha} \tag{22}$$

where $\dot{a}$ is the reference strain rate, and f is a general function describing the dependence of strain rate on the stress. In Eq. (22), the critical resolved shear stress on the α slip system is the denominator τ_c^α, which evolves following the Asaro-Needleman hardening rule [34]. Accordingly, the plastic part evolves following the sum of plastic slips as:

$$\dot{F}_{ij}^p = \sum_\alpha \dot{\gamma}^\alpha s_i^\alpha n_q^\alpha \; F_{qj}^p \tag{23}$$

where s^α and n^α are unit vectors, respectively parallel to the slip direction and normal to the slip plane. This crystal plasticity model represents a face-centred cubic metal lattice with 12 octahedral slip systems and the parameters and constitutive equations are taken from [29].

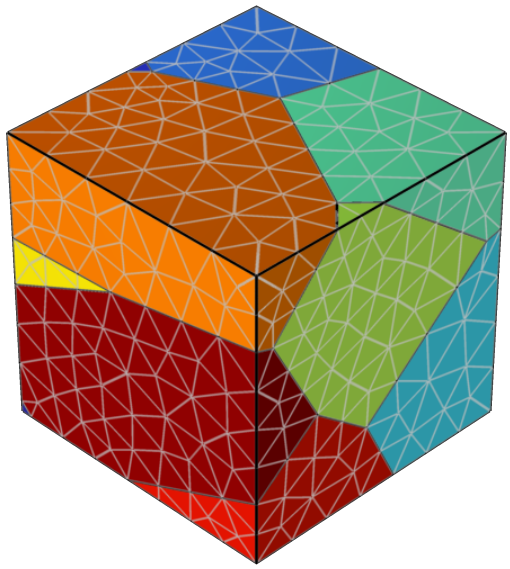

Figure 6: Crystal plasticity-based polycrystal containing 10 grains.

The boundary value problem is a unit cube containing 10 randomly oriented grains, each behaving as a single crystal with elastic-viscoplastic behaviour, see Fig. 6. This is a common geometry in the crystal plasticity community [33]. One *External Material* must be defined in COMSOL for each grain, as while they all use the same material model, they require different input properties due to the different orientations. The boundary conditions are chosen to mimic a

uniaxial strain-controlled tensile test, with the displacement imposed on one face of the cube and the opposite face fixed. A final elongation of 5% is imposed. Given the rate-dependent nature of the problem, the COMSOL's *Time Dependent* solver is used, with the time being discretised into 20 uniform increments of 1 s. The results obtained are shown in Figs. 7 and 8.

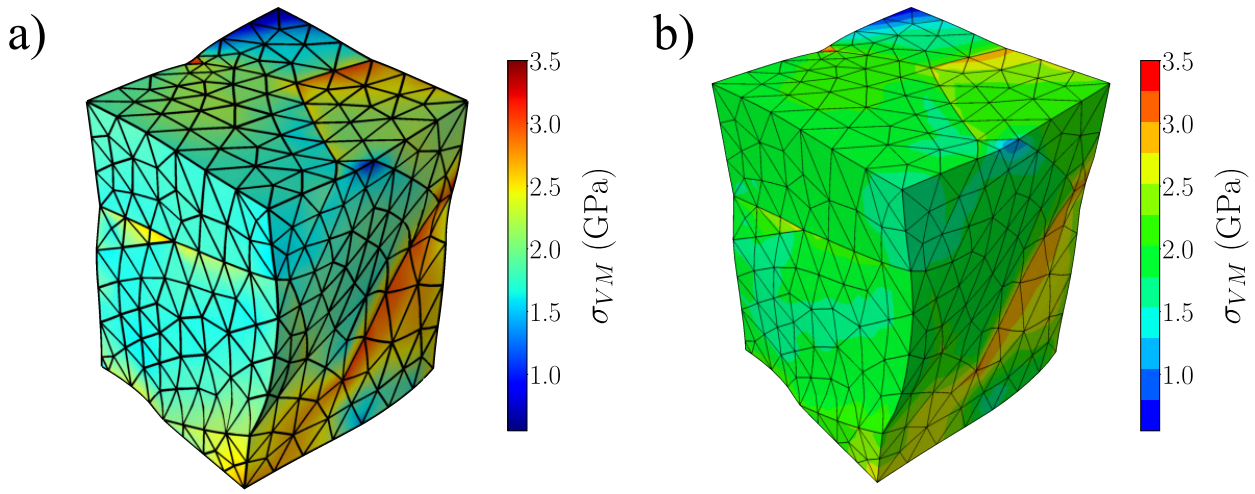

Figure 7: Crystal plasticity-based predictions of polycrystalline deformation: contour plots of the von Mises stress (in GPa). Results obtained: (a) with Abaqus, using crystal plasticity *UMAT*, and (b) with COMSOL, using the same *UMAT* and UMAT4COMSOL. The deformation has been scaled by a factor of 5.

As in the previous case studies, the results obtained appear to be identical, independently of the finite element package used. No visible differences are observed in the deformed shape or the stress contours, see Fig. 7. Due to the different grain orientations, various locations of stress concentration can be observed, and these are present in both COMSOL- and Abaqus-based predictions. From a quantitative viewpoint, the differences in the displacement solution fields observed are below 0.1%, validating our approach and implementation.

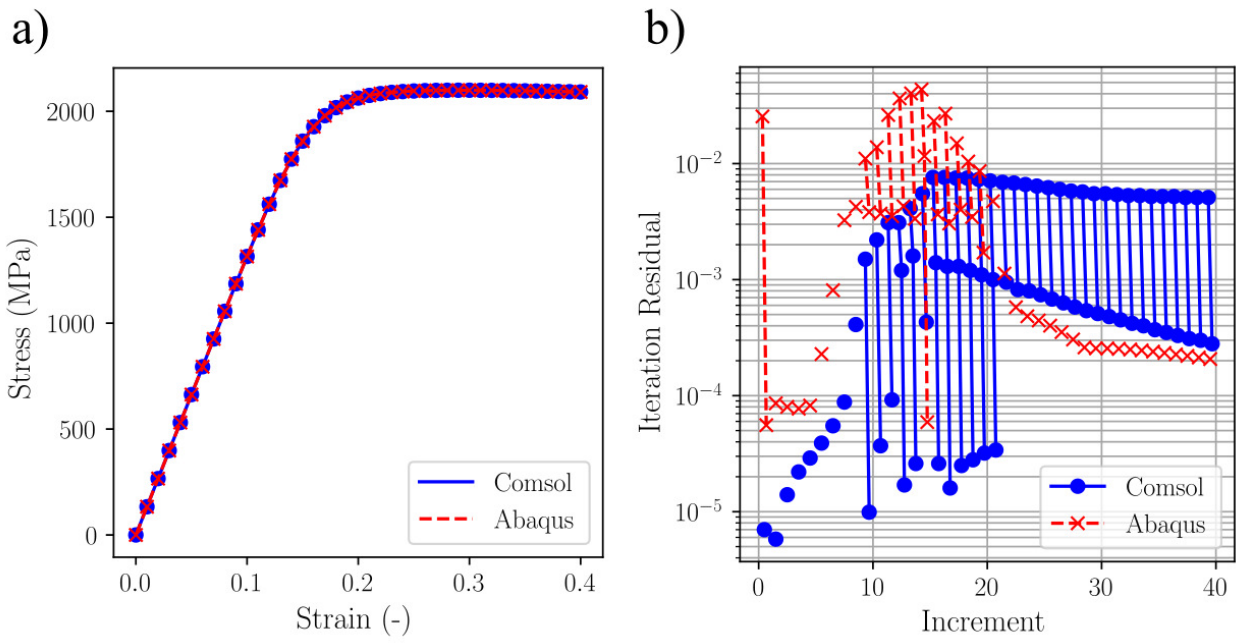

Figure 8: Crystal plasticity-based predictions of polycrystalline deformation: (a) resultant macroscopic stress-strain curve, as predicted with Abaqus (+*UMAT*) and with COMSOL (+*UMAT* and UMAT4COMSOL); and (b) convergence plots for both COMSOL and Abaqus solvers, showing the magnitude of the residual for each iteration as a function of the load increment.

The predicted macroscopic stress-strain curve is shown in Fig. 8a. The results from Abaqus and COMSOL perfectly overlap. The convergence of the simulations is evaluated in Fig. 8b, which displays the evolution of the equilibrium error estimator. Both COMSOL and Abaqus exhibit a similar performance although on this occasion Abaqus appears to provide a better performance in the last increments. This is likely to be related to the different methods employed for estimating the relative residual (L2-norm in COMSOL and ∞-norm in Abaqus), and, more importantly, to the use of extrapolation schemes of different orders for the solution predictor in Abaqus and COMSOL.

5. Conclusions

We have presented UMAT4COMSOL, a new wrapper that enables linking COMSOL C-coded *External Material* library with Abaqus Fortran-coded user material (*UMAT*) subroutines. This enables bringing to the multi-physics environment of COMSOL the advanced material models that the solid mechanics community has developed and implemented through Abaqus *UMAT* subroutines. Thus, UMAT4COMSOL enables the combination of advanced multi-physics and material modelling tools, as necessary to predict the degradation of Li-Ion batteries, the corrosion of metals

and the oxidation of superalloys, among others. We validate our framework and demonstrate its potential by addressing three case studies of particular relevance: (i) the elastoplastic behaviour of a holed plate, (ii) the finite strain non-linear hyperelastic deformation of a twisted cube, and (iii) the crystal plasticity-based prediction of microscopic and macroscopic deformation in a polycrystalline solid. The results show that predictions obtained using COMSOL and UMAT4COMSOL are identical to those obtained using Abaqus and the associated *UMAT* subroutine. Moreover, convergence rates also exhibit a remarkable agreement, despite the different characteristics of each solver.

Acknowledgements

The authors gratefully acknowledge financial support through grant EP/V038079/1 ("SINDRI") from the Engineering and Physical Sciences Research Council (EPSRC). Sergio Lucarini acknowledges financial support from the Marie Skłodowska-Curie Individual European Fellowship under the European Union's Horizon 2020 Framework Programme for Research and Innovation through the project SIMCOFAT (Grant agreement ID: 101031287). Emilio Martínez-Pañeda acknowledges financial support from UKRI's Future Leaders Fellowship programme [grant MR/V024124/1].

Data availability

The software developed, UMAT4COMSOL, is made freely available together with example case studies and documentation. UMAT4COMSOL can be downloaded from the website of the research group `https://www.imperial.ac.uk/mechanics-materials/codes/`, a website repository `www.empaneda.com/codes` and a GitHub repository `https://github.com/sergiolucarini`.

References

[1] E. Martínez-Pañeda, Progress and opportunities in modelling environmentally assisted cracking, RILEM Technical Letters 6 (2021) 70–77.

[2] J. R. Mianroodi, P. Shanthraj, C. Liu, S. Vakili, S. Roongta, N. H. Siboni, N. Perchikov, Y. Bai, B. Svendsen, F. Roters, D. Raabe, M. Diehl, Modeling and simulation of microstructure in metallic systems based on multi-physics approaches, npj Computational Materials 8 (1) (2022) 1–15.

[3] E. Martínez-Pañeda, C. F. Niordson, R. P. Gangloff, Strain gradient plasticity-based modeling of hydrogen environment assisted cracking, Acta Materialia 117 (2016) 321–332.

[4] G. Gobbi, C. Colombo, S. Miccoli, L. Vergani, A fully coupled implementation of hydrogen embrittlement in FE analysis, Advances in Engineering Software 135 (April) (2019) 102673.

[5] C. Cui, R. Ma, E. Martínez-Pañeda, A phase field formulation for dissolution-driven stress corrosion cracking, Journal of the Mechanics and Physics of Solids 147 (2021) 104254.

[6] T. Q. Ansari, H. Huang, S.-Q. Shi, Phase field modeling for the morphological and microstructural evolution of metallic materials under environmental attack, npj Computational Materials 7 (2021) 143.

[7] J. Reuchet, L. Remy, Fatigue Oxidation Interaction in a Superalloy - Application To Life Prediction in High Temperature Low Cycle Fatigue., Metallurgical transactions. A, Physical metallurgy and materials science 14 A (1) (1983) 141–149.

[8] D. G. Leo Prakash, M. J. Walsh, D. Maclachlan, A. M. Korsunsky, Crack growth micromechanisms in the IN718 alloy under the combined influence of fatigue, creep and oxidation, International Journal of Fatigue 31 (11-12) (2009) 1966–1977.

[9] E. Martínez-Pañeda, A. Golahmar, C. F. Niordson, A phase field formulation for hydrogen assisted cracking, Computer Methods in Applied Mechanics and Engineering 342 (2018) 742–761.

[10] P. K. Kristensen, C. F. Niordson, E. Martínez-Pañeda, A phase field model for elastic-gradient-plastic solids undergoing hydrogen embrittlement, Journal of the Mechanics and Physics of Solids 143 (2020) 104093.

[11] A. Valverde-González, E. Martínez-Pañeda, A. Quintanas-Corominas, J. Reinoso, M. Paggi, Computational modelling of hydrogen assisted fracture in polycrystalline materials, International Journal of Hydrogen Energy 47 (75) (2022) 32235–32251.

[12] T. Hageman, E. Martínez-Pañeda, An electro-chemo-mechanical framework for predicting hydrogen uptake in metals due to aqueous electrolytes, Corrosion Science 208 (2022) 110681.

[13] C. Cui, R. Ma, E. Martínez-Pañeda, A generalised, multi-phase-field theory for dissolution-driven stress corrosion cracking and hydrogen embrittlement, Journal of the Mechanics and Physics of Solids 166 (2022) 104951.

[14] A. M. Boyce, E. Martínez-Pañeda, A. Wade, Y. S. Zhang, J. J. Bailey, T. M. Heenan, P. R. Brett, Dan J. L., Shearing, Cracking predictions of lithium-ion battery electrodes by X-ray computed tomography and modelling, Journal of Power Sources 526 (2022) 231119.

[15] W. Ai, B. Wu, E. Martínez-Pañeda, A multi-physics phase field formulation for modelling fatigue cracking in lithium-ion battery electrode particles, Journal of Power Sources 544 (2022) 231805.

[16] Z. Pan, L. Brassart, Constitutive modelling of hydrolytic degradation in hydrogels, Journal of the Mechanics and Physics of Solids 167 (May) (2022) 105016.

[17] M. Rambausek, D. Mukherjee, K. Danas, A computational framework for magnetically hard and soft viscoelastic magnetorheological elastomers, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114500.

[18] M. A. Moreno-Mateos, M. Hossain, P. Steinmann, D. Garcia-Gonzalez, Hard magnetics in ultra-soft magnetorheological elastomers enhance fracture toughness and delay crack propagation, Journal of the Mechanics and Physics of Solids 173 (2023) 105232.

[19] J. Y. Wu, W. X. Chen, Phase-field modeling of electromechanical fracture in piezoelectric solids: Analytical results and numerical simulations, Computer Methods in Applied Mechanics and Engineering 387 (2021) 114125.

[20] L. Quinteros, E. García-Macías, E. Martínez-Pañeda, Electromechanical phase-field fracture modelling of piezoresistive CNT-based composites, Computer Methods in Applied Mechanics and Engineering 407 (2023) 115941.

[21] J. H. Nie, D. A. Hopkins, Y. T. Chen, H. T. Hsieh, Development of an object-oriented finite element program with adaptive mesh refinement for multi-physics applications, Advances in Engineering Software 41 (4) (2010) 569–579.

[22] B. Patzák, D. Rypl, J. Kruis, MuPIF-A distributed multi-physics integration tool, Advances in Engineering Software 60-61 (2013) 89–97.

[23] Y. Zhao, P. Stein, Y. Bai, M. Al-siraj, Y. Yang, B.-x. Xu, A review on modeling of electro-chemo-mechanics in lithium-ion batteries, Journal of Power Sources 413 (2019) 259–283.

[24] M. Smith, ABAQUS/Standard User's Manual, Version 2022, Dassault Systèmes Simulia Corp, United States, 2022.

[25] COMSOL Multiphysics® v. 6.1, COMSOL Multiphysics Reference Manual, COMSOL AB, Stockholm, Sweden, 2022.

[26] W. Sun, E. L. Chaikof, M. E. Levenston, Numerical approximation of tangent moduli for finite element implementations of nonlinear hyperelastic material models, J Biomech Eng 130 (6) (2008) 061003.

[27] M. Ramasubramanian, Y. Wang, A computational micromechanics constitutive model for the unloading behavior of paper, International Journal of Solids and Structures 44 (22) (2007) 7615–7632.

[28] Y. Navidtehrani, C. Betegón, E. Martínez-Pañeda, A unified Abaqus implementation of the phase field fracture method using only a user material subroutine, Materials 14 (8) (2021) 1913.

[29] Y. Huang, A user-material subroutine incorporating single crystal plasticity in the abaqus finite element program, in: Harvard University Report, 1991, pp. 1–24.

[30] J. W. Kysar, Addendum to "a user-material subroutine incorporating single crystal plasticity in the abaqus finite element program", in: Harvard University Report, 1997, pp. 1–3.

[31] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, 7th Edition, Butterworth-Heinemann, Oxford, 2013.

[32] A. Logg, K.-A. Mardal, G. Wells, Automated solution of differential equations by the finite element method: The FEniCS book, Vol. 84 of Lecture Notes in Computational Science and Engineering, Springer Berlin, Heidelberg, 2012.

[33] J. Segurado, R. A. Lebensohn, J. LLorca, Chapter one - computational homogenization of polycrystals, in: M. I. Hussein (Ed.), Advances in Crystals and Elastic Metamaterials, Part 1, Vol. 51 of Advances in Applied Mechanics, Elsevier, 2018, pp. 1–114.

[34] D. Peirce, R. Asaro, A. Needleman, An analysis of nonuniform and localized deformation in ductile single crystals, Acta Metallurgica 30 (6) (1982) 1087–1119.