Numerical implementation of a creep damage model

Classical creep damage rules depend on the von Mises equivalent stress, on the trace of the stress tensor, and on the maximum eigenstress [1]. They can be either used in the framework or post-processors, to compute the time to failure of components, or they can be coupled with elastic and plastic constitutive equations, in order to model also in service degradation of the material. Coupled problems usually generate softening in the mechanical behaviour, and a series of numerical problems, like non unicity of the solution, etc... nevertheless these problems are less important with viscoplastic constitutive equations, since there is a *regularization* of the solutions.

The purpose of the present mini-project is first to compare the performance of explicit and implicit local integrations for constitutive equations, and also to show that a quite naive explicit integration of a damage model in a Norton law may produce efficient results for some simple cases, provided a time step control is implemented, even if an implicit implementation is preferable. The second goal is to demonstrate that, depending on the type of predominant critical variable in the damage model (namely von Mises equivalent stress or the trace), the crack initiation in a notched specimen submitted to a creep loading can be observed at the notch root, or in the center of the specimen.

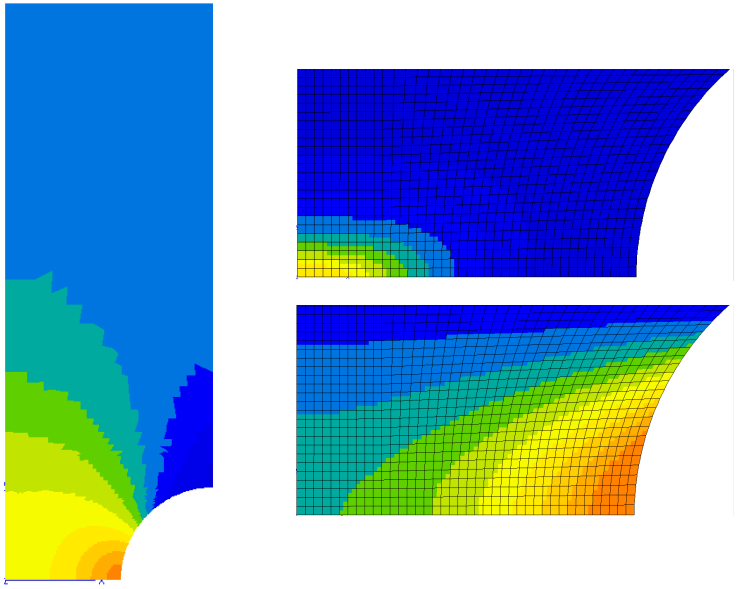
The following directories are provided:

– `Code`, a directory containing source files for the Norton model, where the new coupled damage model will be developed.

– `data`, a directory containing mesh and input files needed to run explicit and implicit FE computations with the Norton model. No development is needed here, since the purpose is just to check the way each algorithm operates for the existing implementations;

– `data2`, a directory containing input files needed to perform the computations with the newly implemented model. This one will be used for the model developed in the framework of the mini-project.

The model will be applied to an axisymmetric notched specimen. The pictures below show typical results which will be obtained in the mini-project (axial stress on the half specimen on the left; two damage fields, with predominant von Mises and predominant trace of the stress tensor on the right).



Keywords: *Notched specimens; coupled damage models; integration of constitutive equations*

1 Comparison of explicit and implicit local integrations

Go to directory `data` The directory contains the mesh file `Qua8_axi.geof`, that defines the nodes, the elements and the various groups used for applying the boundary conditions, and the material file `norton.mat`, that defines elastic and viscoplastic material properties, using the *native implementation* of the ZeBuLoN code. In this last file, the initial yield is taken equal to zero, so that the model is a pure Norton model. Theses two files are used by each of the input files present in the directory. The file `quae.inp` correspond to an integration by means of an explicit Runge–Kutta algorithm, with automatic time-stepping. The file `quai.inp` introduces a θ-method which leads to an implicit system solved by a Newton technique. The global algorithm is implicit, for both cases, and the choice of the tangent matrix will be discussed.

Edit the various files. The applied load is a hardening test followed by a relaxation period. Check the strategy used for each case: the implicit integration provides *for free* a good tangent matrix, that allows to make rather large time increments, meanwhile the explicit integration does not give any direct solution to estimate a tangent matrix, so that an initial stiffness method is applied here.

Run the implicit case by means of the command

```
Zrun quai
```

and then the explicit case by means of the command

```
Zrun quae
```

By looking at files quai.msg and quae.msg, compare the number of time increments in each computation, the number of iterations for each increment, and the convergence rate (quadratic or not ?). Compare the global results obtained in each case by means of the command

```
./Dforce
```

that plot the evolution of the force applied to the specimen as a function of time. Try to decrease the CPU time by decreasing the number of increments, or by relaxing a little bit the precision on a global level (the number after `*ratio absolu` instruction). For each new attempt, check that the computation accuracy is still acceptable. You can also check the effect of the instruction `init_d_dof` (suppress it and see what happens), or of the suggested increment of `evcum` in the `**automatic_time` option. For the implicit integration method, the number after the `*integration theta_method` instruction are respectively θ, then the value of the variation of the variables in the residual which stops the Newton iterations, and finally the maximum number of Newton iterations. For the explicit integration method, the unique value provided to the algorithm regulates the difference between the value at $\Delta t/2$ and the second order estimate in the Runge–Kutta method.

Perform a short analysis of the results by using the command `Zmaster` for `quai` and `quae`. Observe the location of the maximum viscoplastic strain, and the transformation of the contour maps of the equivalent von Mises stress and of the axial stress, σ_{22}. This can be made either on contour maps or on the ligament `bottom`.

2 Development of the creep–damage model

The model is defined through the following equations:

- Elastic–plastic partition:

$$\underset{\sim}{\varepsilon} = \underset{\sim}{\varepsilon}^e + \underset{\sim}{\varepsilon}^p$$

- Elastic constitutive equation with isotropic damage D:

$$\underset{\sim}{\sigma} = (1-D)\,\underset{\approx}{\Lambda} : \underset{\sim}{\varepsilon}^e$$

- Equivalent stress, computed with the deviator $\underset{\sim}{s}$ of $\underset{\sim}{\sigma}$:

$$J(\underset{\sim}{\sigma}) = \left(\frac{3}{2}\underset{\sim}{s} : \underset{\sim}{s}\right)^{1/2}$$

- Equivalent viscoplastic strain rate:

$$\dot{v} = \left(\frac{J}{K(1-D)} \right)^n$$

- Direction of the viscoplastic rate:

$$\underset{\sim}{\mathbf{n}} = \frac{3}{2} \frac{\underset{\sim}{\mathbf{s}}}{J}$$

- Viscoplastic strain rate tensor:

$$\dot{\underset{\sim}{\varepsilon}}^p = \dot{v}\, \underset{\sim}{\mathbf{n}}$$

- Damage evolution (where $< . >$ takes the positive part)

$$\dot{D} = \left\langle \frac{(1-\alpha)J + \alpha\, \mathrm{trace}(\underset{\sim}{\sigma})}{K_d} \right\rangle^{n_d}$$

Go to directory `Code/source`

There are two files in this directory:

– `Norton.z` is a user defined version of the Norton model. The first section of the code (keyword `@Class`) defines the name of the model, the names of the material parameters (`K` and `n`) and the names of the variables to integrate (the tensor `eel` and the scalar `evcum`). The section `@PostStep` is called after convergence, in order to evaluate the tangent matrix. The five lines in section `@Derivative` are the only ones that are needed to fully define the explicit integration, for any kind of conditions (1D, 2D, 3D, even plane stress). The section `@CalcGradF` corresponds to the definition of the implicit integration. Depending on the type of computation he wishes to run, the user can define `@Derivative` only, or `@CalcGradF` only;

– `Creepdom.z` is a first draft of the new model, including damage evolution. Four parameters are added, `Kd`, `nd` and `alpha` for defining the damage rule, and `dammax`, which is D_{max}, the maximum acceptable value for damage, for which the element is assumed to be broken. One scalar variable is added, `damage`. The section `@PostStep` is already updated. Instructions are given in the section `@Derivative`. Nothing is done up to now in section `@CalcGradF`.

After inspecting these two files, go one level up in directory `Code`, and build the dynamic libray by means of the command

Zmake

This will translate the two files with extensions `.z` into c++ files, compile them and produces the file `libLocal.so`. At that stage, the `Norton` model can be used, but *not* the model `creepdom`.

Go to directory `Code/data2`

Three problems are ready in this directory:

– `quai.inp` is an other version of the previous implicit integration problem. Instead of using the native implementation of the code, the Norton law is now addressed through the user defined material in `Code/source/Norton.z` file. This model will be loaded when the code start running, provided the symbolic link to the dynamic library is set: `libLocal.so` in `data2` should point toward `../Code/libLocal.so`. You can try to remove/set this link to see what happens: enter `rm libLocal.so`, then run the code, which should fail; then redo the link by means of the instruction `ln -s ../Code/libLocal.so .` and rerun the code, that should work again, hopefully. Check that the result obtained with the user defined material is the same as the result obtained in the previous section.

– `qua1.inp` and `qua2.inp`, which are files ready for use with the new model. All the conditions of the computations are equivalent to the previous computations, the only difference is that they will use new material files, respectively `creepdom1.mat` (where $\alpha = 1$) and `creepdom2.mat` (where $\alpha = 0$). This will give the user the opportunity to test a model sensitive to von Mises stress only, or to the trace of the stress tensor only.

Go to directory `Code/source`

Edit the `Creepdom.z` source file, introduce the new damage model, then go one level up and recompile with `Zmake` as previously. When you think you have the good implementation ready, you go back to `Code/data2` and you run `creepdom1` and `creepdom2` models. Several conditions can be

considered for α, and also for the other parameters, to increase or reduce damage rate. Compare the convergence rates with the convergence rate obtained with the Norton model.

An experienced user can also try to implement the same model with an implicit integration. In the Norton model, two residuals are introduced to compute the *increments* (and no longer the rates) of the elastic strain $\Delta\underset{\sim}{\varepsilon}^e$ and of the equivalent viscoplastic strain rate Δv:

- the tensorial elastic-plastic partition:

$$\mathcal{F}_e = \Delta\underset{\sim}{\varepsilon}^e + \Delta v\,\underset{\sim}{\mathbf{n}} - \Delta\underset{\sim}{\varepsilon}$$

- the scalar flow increment (with Δt the time increment):

$$\mathcal{F}_p = \Delta v - \left(\frac{J}{K}\right)^n \Delta t$$

The user will read the file to see the expressions of the partial derivatives of $\mathcal{F}_e$ and $\mathcal{F}_p$ with respect to $\Delta\underset{\sim}{\varepsilon}^e$ and Δv. The damage model will just need:

- an additional scalar residual to define the damage increment ΔD

$$\mathcal{F}_d = \Delta D - \left\langle \frac{(1-\alpha)J + \alpha\,\mathrm{trace}(\underset{\sim}{\sigma})}{K_d} \right\rangle^{n_d} \Delta t$$

- the modification of $\mathcal{F}_e$ and $\mathcal{F}_p$ to account for damage in them,

- the modification of the existing partial derivatives, and the computation of new ones (nine partial derivatives instead of just four).

References

[1] F.A. Leckie and D.R. Hayhurst. Creep rupture of structures. *Proc. Royal Soc. London*, 340:323–347, 1974.