Numerical implementation

- **Numerical methods**
- **The θ-method applied to constitutive equations**
- **Description of the material interface**
- **Cycle jump technique**
- **Case studies: turbine blades, head engine**

–Do it yourself–

Numerical implementation

- *Numerical methods*
- The θ-method applied to constitutive equations
- Description of the material interface
- Cycle jump technique
- Case studies: turbine blades, head engine

Solution of non linear systems

$$f(x) = 0 \qquad \text{rewritten as} \qquad x = g(x)$$

Let us assume:

$$x_n = s + \varepsilon_n \qquad \text{resp. solution, error}$$

Then:

$$x_{n+1} = g(x_n) \quad = g(s) + (x_n - s)g'(s) + \frac{1}{2}(x_n - s)^2 g''(s)$$

- Convergence iff: $|g'| \leq K < 1$
- Order (linear, quadratic convergence ?)
  - ⋆ Order 1 : $\varepsilon_{n+1} \sim g'(s)\varepsilon_n$
  - ⋆ Order 2 : $\varepsilon_{n+1} \sim \frac{1}{2}g''(s)\varepsilon_n^2$, if $g' \equiv 0$

Fixed point method

$$\exists\, t \quad \text{such as} \quad g(x) - g(s) = g'(t)(x - s)$$

$$
\begin{aligned}
|x_n - s| = |g(x_{n-1}) - s| \quad &= g'(s)|x_{n-1} - s| \\
&\leq K|x_{n-1} - s| \\
&\leq ... \leq K^n|x_o - s|
\end{aligned}
$$

Newton method

Residual :

$$R(x) = 0$$

Taylor development :

$$R(x_{n+1}) = R(x_n) + \left(\frac{\partial R}{\partial x}\right)_n \Delta x + \ldots$$

To get $R(x_{n+1}) = 0$, try

$$\Delta x = -\left(\frac{\partial R}{\partial x}\right)_n^{-1} R(x_n)$$

Quasi-Newton, replace $\left(\frac{\partial R}{\partial x}\right)_n^{-1}$ by K (constant during iterations)

Order of Newton method

Writing :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Equiv to :

$$g(x) = x - \frac{f(x)}{f'(x)}$$

$$g'(x) = 1 - \frac{f'^2 - f f''}{f'^2} = \frac{f f''}{f'^2} = 0$$

$\varepsilon_{n+1} \sim \varepsilon^2$ ...order 2 $\rightarrow$ *quadratic* convergence

Order of Quasi-Newton method

Writing :

$$x_{n+1} = x_n - \frac{f(x_n)}{K}$$

Equiv to :

$$g(x) = x - \frac{f(x)}{K}$$

$$g'(x) = 1 - \frac{f'}{K}$$

Linear convergence if $|g'(x)| < 1$

Ordinary differential equations

- Differential systems of higher order can be reduced to 1, e.g.

$$\boxed{\frac{d^2y}{dt^2} + q(t)\frac{dy}{dt} = r(t)} \quad \equiv \quad \boxed{\begin{aligned} \frac{dy}{dt} &= z(t) \\ \frac{dz}{dt} &= r(t) - q(t)z(t) \end{aligned}}$$

- General form:

$$\{\dot{v}\} = \{f\}(t, \{v\}) \quad ; \quad \{v\}(t = t_0) = \{v\}_0$$

- Forward Euler:

$$\{v\}(t + \Delta t) = \{v\}(t) + \Delta t \{\dot{v}\}(t, \{v\})$$

Runge–Kutta (1)

Idea = Multiple evaluations on the same time increment

- Time increment $t \longrightarrow t + \Delta t$
- RK21 method, two evaluations, second order accurate
- RK34 method, four evaluations, fourth order accurate

Starts from Taylor:

$$\{v\}(t + \Delta t) = \{v\}(t) + \{\dot{v}\}(t)\Delta t + O(\Delta t^2)$$

Let us note $\{\delta v_1\} = \Delta t \{\dot{v}\}(t)$

Runge–Kutta (2)

RK21

Mid-point evaluation:

$$\begin{aligned}\{\delta v_2\} &= \Delta t\,\{\dot v\}\left(t+\frac{\Delta t}{2},\{v\}(t)+\frac{1}{2}\{\delta v_1\}\right)\\ &= \Delta t\left(\{\dot v\}(t)+\frac{\Delta t}{2}\{\ddot v\}(t)\right)\\ &= \{\delta v_1\}+\frac{\Delta t^2}{2}\{\ddot v\}(t)\end{aligned}$$

The $\{\ddot v\}(t)$ term can be eliminated ($\{\ddot v\}(t)=\{\delta v_2\}-\{\delta v_1\}$) the method is then second order accurate:

$$\begin{aligned}\{v\}(t+\Delta t) &= \{v\}(t)+\{\dot v\}(t)\Delta t+\{\ddot v\}(t)\frac{\Delta t^2}{2}+O(\Delta t^3)\\ \{v\}(t+\Delta t) &= \{v\}(t)+\{\delta v_2\}+O(\Delta t^3)\end{aligned}$$

Runge–Kutta (3)

RK34

$$
\begin{aligned}
\{\delta v_1\} &= \Delta t \, \{\dot{v}\} \, (t, \{v\}) \\
\{\delta v_2\} &= \Delta t \, \{\dot{v}\} \left(t + \frac{\Delta t}{2}, \{v\} + \frac{1}{2} \{\delta v_1\} \right) \\
\{\delta v_3\} &= \Delta t \, \{\dot{v}\} \left(t + \frac{\Delta t}{2}, \{v\} + \frac{1}{2} \{\delta v_2\} \right) \\
\{\delta v_4\} &= \Delta t \, \{\dot{v}\} \, (t + \Delta t, \{v\} + \{\delta v_3\}) \\
\{v\} \, (t + \Delta t) &= \{v\} \, (t) + \frac{1}{6} \{\delta v_1\} + \frac{1}{3} \{\delta v_2\} + \frac{1}{3} \{\delta v_3\} + \frac{1}{6} \{\delta v_4\} + O(\Delta t^5)
\end{aligned}
$$

θ–methods

Two solutions for the discretization:

$$\{\Delta v\} = \begin{cases} \Delta t \, \{\dot{v}\} \, (t + \theta \Delta t) & \text{type A} \\ \Delta t \, ((1-\theta) \, \{\dot{v}\} \, (t) + \theta \, \{\dot{v}\} \, (t + \Delta t)) & \text{type B} \end{cases} \tag{1}$$

Solution using Newton method:

Type	Residual $\{R\} = \{0\}$	Jacobian $\dfrac{\partial \{R\}}{\partial \{\Delta v\}}$
A	$\{\Delta v\} - \Delta t \, \{\dot{v}\} \, (t + \theta \Delta t)$	$[1] - \Delta t \left. \dfrac{\partial \{\dot{v}\}}{\partial \{\Delta v\}} \right\vert_{t+\theta\Delta t}$
B	$\{\Delta v\} - \Delta t \, ((1-\theta) \, \{\dot{v}\} \, (t) + \theta \, \{\dot{v}\} \, (t + \Delta t))$	$[1] - \Delta t \, \theta \left. \dfrac{\partial \{\dot{v}\}}{\partial \Delta v} \right\vert_{t+\Delta t}$

Gauss integration

r-point Gauss integration on a [-1:+1] segment:

$$\int_{-1}^{+1} f(t)dt = \sum_{1}^{r} w_i f(\xi_i)$$

gives exact result for a $(2r-1)$ order polynom

Evaluation at sampling points ξ_i, combined with weigthing coefficients w_i

Example, order 2:

$$\begin{array}{llll} f(t) = & 1 & 2 & = w_1 + w_2 \\ f(t) = & t & 0 & = w_1 x_1 + w_2 x - 2 \\ f(t) = & t^2 & 2/3 & = w_1 x_1^2 + w_2 x - 2^2 \\ f(t) = & t^3 & 0 & = w_1 x_1^3 + w_2 x - 2^3 \end{array}$$

then $w_1 = w_2 = 1$, and $\xi_1 = -\xi_2 = 1/\sqrt{3}$

Global algorithm

For each loading increment, do while $\|\{R\}_{iter}\| > EPSI$:
$iter = 0; iter < ITERMAX; iter++$

1. Update displacements: $\Delta\{u\}_{iter+1} = \Delta\{u\}_{iter} + \delta\{u\}_{iter}$
2. Compute $\Delta\{\varepsilon\} = [B].\Delta\{u\}_{iter+1}$ then $\Delta\underset{\sim}{\boldsymbol{\varepsilon}}$ for each Gauss point
3. Integrate the constitutive equation: $\Delta\underset{\sim}{\boldsymbol{\varepsilon}} \rightarrow \Delta\underset{\sim}{\boldsymbol{\sigma}},\ \Delta\alpha_I,\ \dfrac{\Delta\underset{\sim}{\boldsymbol{\sigma}}}{\Delta\underset{\sim}{\boldsymbol{\varepsilon}}}$
4. Compute int and ext forces: $\{F_{int}(\{u\}_t + \Delta\{u\}_{iter+1})\}, \{F_e\}$
5. Compute the residual force: $\{R\}_{iter+1} = \{F_{int}\} - \{F_e\}$
6. New displacement increment: $\delta\{u\}_{iter+1} = -[K]^{-1}.\{R\}_{iter+1}$

Numerical implementation

- **Numerical methods**
- ***The θ-method applied to constitutive equations***
- **Description of the material interface**
  - ***(Visco-)plastic models with isotropic hardening***
  - ***Multi-kinematic models***
  - ***Description of a material library***
- **Cycle jump technique**
- **Case studies: turbine blades, head engine**

Anatomy of constitutive equations

Definitions:

- *External parameters* (`ep`) imposed as input
- *Integrated variables* (`vint`)
- *Auxiliary variables* (`vaux`), just for output
- *Coefficients* (`coef`), material parameters
- *Primal and dual variables*, prescribed variables and associated fluxes

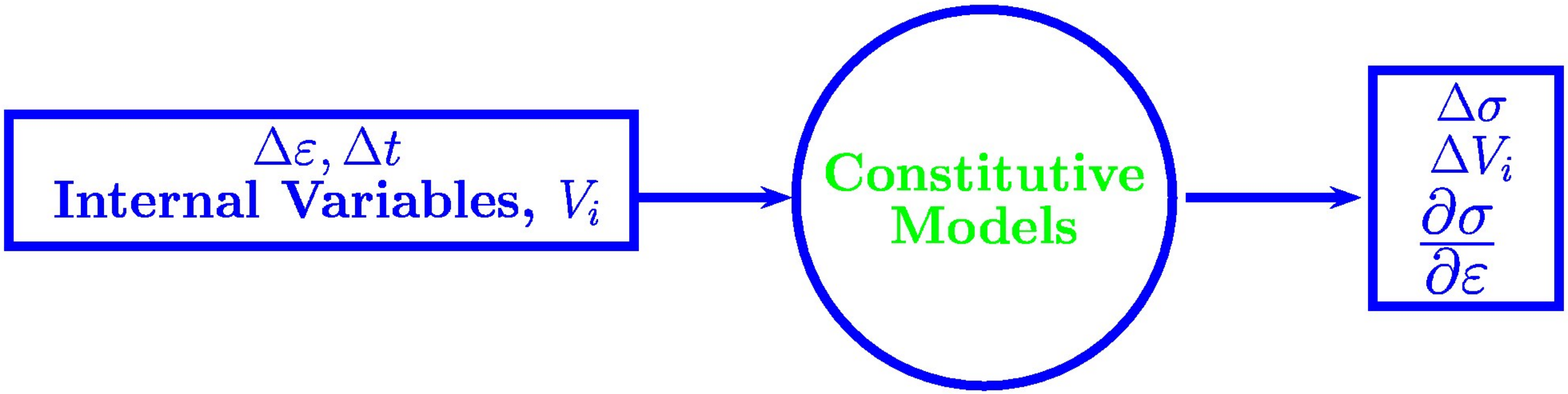
Primal and dual variables in various fields

problem	`primal`	`dual`
mechanics, small perturbation	$\underset{\sim}{\varepsilon}$	$\underset{\sim}{\boldsymbol{\sigma}}$
mechanics, large deformation	$\underset{\sim}{\boldsymbol{F}}$	$\underset{\sim}{\boldsymbol{\Pi}}$
thermal pb	$(T, \underline{\mathbf{grad}T})$	$(H, \underline{\boldsymbol{q}})$
diffusion	concentration	flux
electrostatics	$\underline{\mathbf{grad}V}$	$\underline{\boldsymbol{E}}$
magnetostatics	$rot\ \underline{\boldsymbol{A}}$	$\underline{\boldsymbol{H}}$

$\underset{\sim}{\varepsilon}$ strain tensor, $\underset{\sim}{\boldsymbol{F}}$ deformation gradient, T temperature, V electric potential, $\underline{\boldsymbol{A}}$ potential vector, $\underset{\sim}{\boldsymbol{\sigma}}$ Cauchy stress tensor, $\underset{\sim}{\boldsymbol{S}}$ second Piola–Kirchhoff stress tensor, H enthalpy, $\underline{\boldsymbol{q}}$ thermal flux , $\underline{\boldsymbol{E}}$ electric field $\underline{\boldsymbol{H}}$ magnetic field.

Generic interface for any constitutive equation

For each Gauss Point...



Time discretization

$$\Delta\alpha = \int_t^{t+\Delta t} \dot{\alpha} dt\tau$$

Explicit integration: Substepping, Runge-Kutta.

Implicit integration:

Generalized mid-point rule ($\theta - method\, A$):

$$\Delta\alpha = \dot{\alpha}(t + \theta\Delta t)\Delta t = \dot{\alpha}_\theta \Delta t$$

Trapezoidal integration ($\theta - method\, B$):

$$\Delta\alpha = ((1-\theta)\dot{\alpha}_t + \theta\dot{\alpha}_{t+\Delta t})\,\Delta t = ((1-\theta)\dot{\alpha}_0 + \theta\dot{\alpha}_1)\,\Delta t$$

- $0 < \theta < 1$, $\theta = 0$, explicit; $\theta = 1$, implicit
- $0.5 < \theta < 1$, stable
- $\theta = 0.5$, second order accurate

Discretization of the strain increment

$$\Delta \underset{\sim}{\varepsilon}^p = \int_t^{t+\Delta t} \dot{\underset{\sim}{\varepsilon}}^p d\tau = \int_t^{t+\Delta t} \dot{p}(\tau) \underset{\sim}{\boldsymbol{n}}(\tau) d\tau$$

With $\theta - method\, A$

$$\Delta \underset{\sim}{\varepsilon}^p = \dot{p}_\theta \underset{\sim}{\boldsymbol{n}}_\theta \Delta t = \underset{\sim}{\boldsymbol{n}}_\theta \Delta p$$

- full implicit case: $\Delta \underset{\sim}{\varepsilon}^p = \underset{\sim}{\boldsymbol{n}}_1 \Delta p$ direction given by the final normal, this gives the popular *radial return* algorithm

Radial return algorithm

Trial stress for a prescribed $\Delta\underset{\sim}{\varepsilon}$:

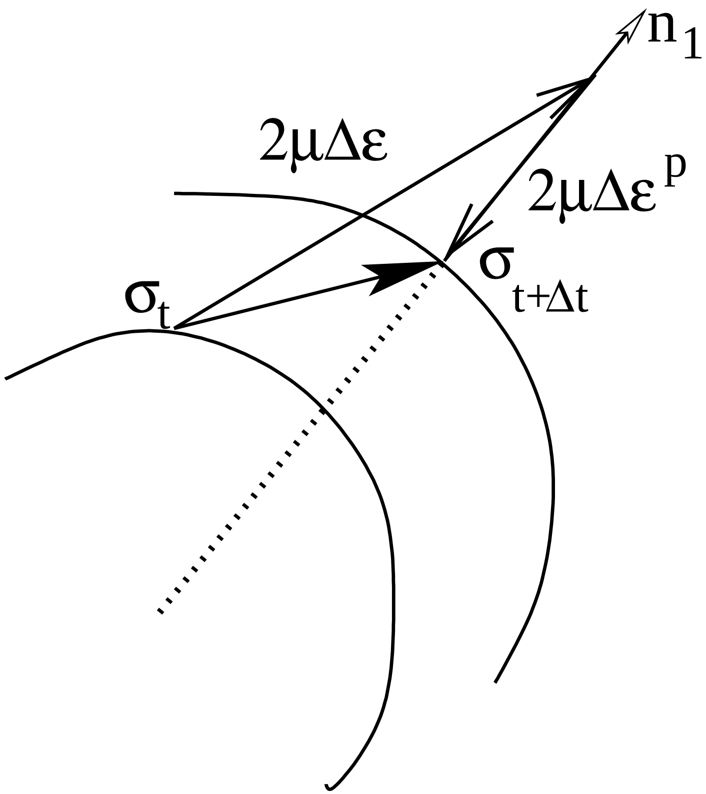
$$\underset{\sim}{\sigma}^* = \underset{\sim}{\sigma}_t + \underset{\approx}{\Lambda} : \Delta\underset{\sim}{\varepsilon}$$

Actual stress at $t + \Delta t$:

$$\underset{\sim}{\sigma}_{t+\Delta t} = \underset{\sim}{\sigma}_t + \underset{\approx}{\Lambda} : (\Delta\underset{\sim}{\varepsilon} - \Delta\underset{\sim}{\varepsilon}^p)$$

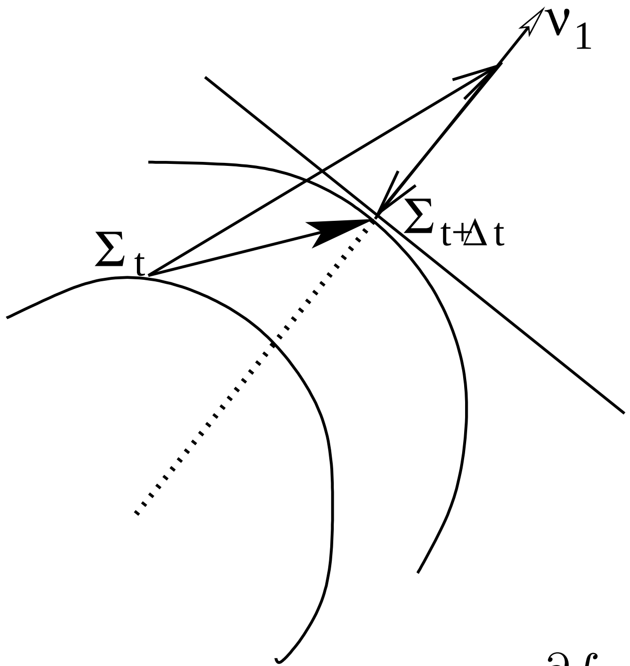
- The corrective term is oriented by the final normal

$$\underset{\sim}{\sigma}_{t+\Delta t} = \underset{\sim}{\sigma}^* - \Delta p \underset{\approx}{\Lambda} : \underset{\sim}{n}_1$$



Generalized radial return algorithm

Closest point projection algorithm



For generalized normality rule:

$$\Delta \underset{\sim}{\boldsymbol{\varepsilon}}^p = \frac{\partial f}{\partial \underset{\sim}{\boldsymbol{\sigma}}} \Delta p \quad ; \quad \Delta \alpha_I = \frac{\partial f}{\partial Y_I} \Delta p$$

Fluxes:

$$\Delta \underset{\sim}{\boldsymbol{\sigma}} = \underset{\approx}{\boldsymbol{\Lambda}} : (\Delta \underset{\sim}{\boldsymbol{\varepsilon}} - \Delta \underset{\sim}{\boldsymbol{\varepsilon}}^p)$$

$$\Delta Y_I = M_I . \Delta \alpha_I$$

$$\Delta \Sigma = \begin{pmatrix} \Delta \underset{\sim}{\boldsymbol{\sigma}} \\ \\ \Delta Y_I \end{pmatrix} = \begin{pmatrix} \underset{\approx}{\boldsymbol{\Lambda}} : \Delta \underset{\sim}{\boldsymbol{\varepsilon}} \\ \\ 0 \end{pmatrix} - \begin{pmatrix} \underset{\approx}{\boldsymbol{\Lambda}} & 0 \\ \\ 0 & M \end{pmatrix} \begin{pmatrix} \dfrac{\partial f}{\partial \underset{\sim}{\boldsymbol{\sigma}}} \\ \\ \dfrac{\partial f}{\partial Y_I} \end{pmatrix} \Delta p$$

Generic form of the implementation

Find state variables increment, $\Delta\underset{\sim}{\varepsilon}^e$ and $\Delta\alpha_I$, using strain partition rule and hardening rules.

$$\begin{aligned} \underset{\sim}{r}_e &= \Delta\underset{\sim}{\varepsilon}^e + \Delta p\, \underset{\sim}{n}_\theta = \Delta\underset{\sim}{\varepsilon} - \Delta\underset{\sim}{\varepsilon}^{th} - \Delta\underset{\sim}{\varepsilon}^{tr} \ldots \\ r_{p_I} &= r_{p_I}(\underset{\sim}{\varepsilon}^e, \alpha_I) \end{aligned}$$

Jacobian matrix [J] = $\begin{pmatrix} \dfrac{\partial \underset{\sim}{r}_e}{\partial \Delta\underset{\sim}{\varepsilon}^e} & \dfrac{\partial \underset{\sim}{r}_e}{\partial \Delta\alpha_I} \\ \dfrac{\partial \underset{\sim}{r}_{p_I}}{\partial \Delta\underset{\sim}{\varepsilon}^e} & \dfrac{\partial \underset{\sim}{r}_{p_I}}{\partial \Delta\alpha_I} \end{pmatrix}$

Note :

$$\frac{\partial \underset{\sim}{r}_e}{\partial \Delta\underset{\sim}{\varepsilon}^e} = \underset{\approx}{I} \quad ; \quad \underset{\approx}{N} = \frac{\partial \underset{\sim}{n}}{\partial \Delta\underset{\sim}{\sigma}} = \frac{1}{J}\left(\frac{3}{2}\underset{\approx}{J} - \underset{\sim}{n} \otimes \underset{\sim}{n}\right)$$

…accounts for normal rotation during the increment

Incremental consistent tangent matrix

After convergence,

$$\begin{pmatrix} d\Delta\underset{\sim}{\varepsilon} \\ 0 \end{pmatrix} = [\mathrm{J}] \begin{pmatrix} d\Delta\underset{\sim}{\varepsilon}^e \\ d\Delta\alpha_I \end{pmatrix} \ldots \text{then} \begin{pmatrix} d\Delta\underset{\sim}{\varepsilon}^e \\ d\Delta\alpha_I \end{pmatrix} = [\mathrm{J}]^{-1} \begin{pmatrix} d\Delta\underset{\sim}{\varepsilon} \\ 0 \end{pmatrix}$$

$$[\mathrm{J}]^{-1} = \left(\begin{array}{c|c} \mathbb{H} & x \\ \hline x & x \end{array}\right), \text{ with } [\mathbb{H}] = \frac{\partial\Delta\underset{\sim}{\varepsilon}^e}{\partial\Delta\underset{\sim}{\varepsilon}}$$

Consistent tangent matrix:

$$\underset{\approx}{\boldsymbol{L}}_c = \frac{\partial\Delta\underset{\sim}{\boldsymbol{\sigma}}}{\partial\Delta\underset{\sim}{\varepsilon}^e} : \frac{\partial\Delta\underset{\sim}{\varepsilon}^e}{\partial\Delta\underset{\sim}{\varepsilon}} = \underset{\approx}{\boldsymbol{\Lambda}} : \mathbb{H}$$

Numerical implementation

- Numerical methods
- The θ-method applied to constitutive equations
- Description of the material interface
  - ⋆ *(Visco-)plastic models with isotropic hardening*
  - ⋆ *Multi-kinematic models*
  - ⋆ *Description of a material library*
- Cycle jump technique
- Case studies: turbine blades, head engine

(Visco-)plastic models with isotropic hardening

$$f(\underset{\sim}{\boldsymbol{\sigma}}, R) = J(\underset{\sim}{\boldsymbol{\sigma}}) - R - \sigma_y$$

$$J(\underset{\sim}{\boldsymbol{\sigma}}) = \sqrt{(3/2)\underset{\sim}{\boldsymbol{s}} : \underset{\sim}{\boldsymbol{s}}} \quad ; \quad \underset{\sim}{\boldsymbol{s}} = \underset{\sim}{\boldsymbol{\sigma}} - (1/3)\,\text{trace}\underset{\sim}{\boldsymbol{\sigma}} \quad ; \quad \sigma_y = \text{init yield}$$

$$\dot{\underset{\sim}{\boldsymbol{\varepsilon}}}^p = \dot{p}\underset{\sim}{\boldsymbol{n}} \quad ; \quad \dot{p} = \sqrt{(2/3)\underset{\sim}{\boldsymbol{\varepsilon}}^p : \underset{\sim}{\boldsymbol{\varepsilon}}^p} \quad ; \quad R = (1 - \exp(-bp))$$

Time independent (TI) behavior: $f = 0$

Time dependent (TD) behavior: $\dot{p} = \left\langle \frac{f}{K} \right\rangle^{1/n}$;

	3D	1D tension
TI	$J(\underset{\sim}{\boldsymbol{\sigma}}) - R - \sigma_Y \quad = 0$	$\sigma = R + \sigma_y$
TD	$J(\underset{\sim}{\boldsymbol{\sigma}}) - R - \sigma_y \quad = K\dot{p}^{1/n}$	$\sigma = R + \sigma_y + K\left(\dot{\varepsilon}^p\right)^{1/n}$

Implementation of (visco-)plastic models with isotropic hardening (1)

Unknowns = $\Delta\underset{\sim}{\boldsymbol{\varepsilon}}^e$, Δp

Time–independent plasticity:

$$\begin{aligned} \underset{\sim}{\boldsymbol{r}}_e &= \Delta\underset{\sim}{\boldsymbol{\varepsilon}}^e + \Delta p\, \underset{\sim}{\boldsymbol{n}}_\theta = \Delta\underset{\sim}{\boldsymbol{\varepsilon}} - \Delta\underset{\sim}{\boldsymbol{\varepsilon}}^{th} - \Delta\underset{\sim}{\boldsymbol{\varepsilon}}^{tr} \ldots \\ r_p &= f(\underset{\sim}{\boldsymbol{\sigma}}_{t+\Delta t}) = 0 \end{aligned}$$

Δp is the increment of equiv (visco-)plastic strain

$\underset{\sim}{\boldsymbol{n}}_\theta$ is the normal to the yield surface at $t + \theta\Delta t$

Time–dependent plasticity, replace previous r_p by:

$$r_p = \Delta p - \Delta t\, \Phi_\theta(J - R, \ldots) = 0$$

Implementation of (visco-)plastic models with isotropic hardening (2)

Time–independent plasticity:

$$[\mathrm{J}] = \begin{pmatrix} \underset{\approx}{\boldsymbol{I}} + \theta \underset{\approx}{\boldsymbol{N}}_\theta : \underset{\approx}{\boldsymbol{\Lambda}}_\theta \Delta p & \underset{\sim}{\boldsymbol{n}}_\theta \\ \underset{\approx}{\boldsymbol{\Lambda}}_1 : \underset{\sim}{\boldsymbol{n}}_1 & -H = -dR/dp \end{pmatrix}$$

Incremental consistent operator $\underset{\approx}{\boldsymbol{L}}_c$ versus tangent continuous operator $\underset{\approx}{\boldsymbol{L}}_t$

$$\underset{\approx}{\boldsymbol{L}}_c = \underset{\approx}{\boldsymbol{L}}_t - 4\mu^2 \Delta p \underset{\approx}{\boldsymbol{N}}$$

Implementation of (visco-)plastic models with isotropic hardening (3)

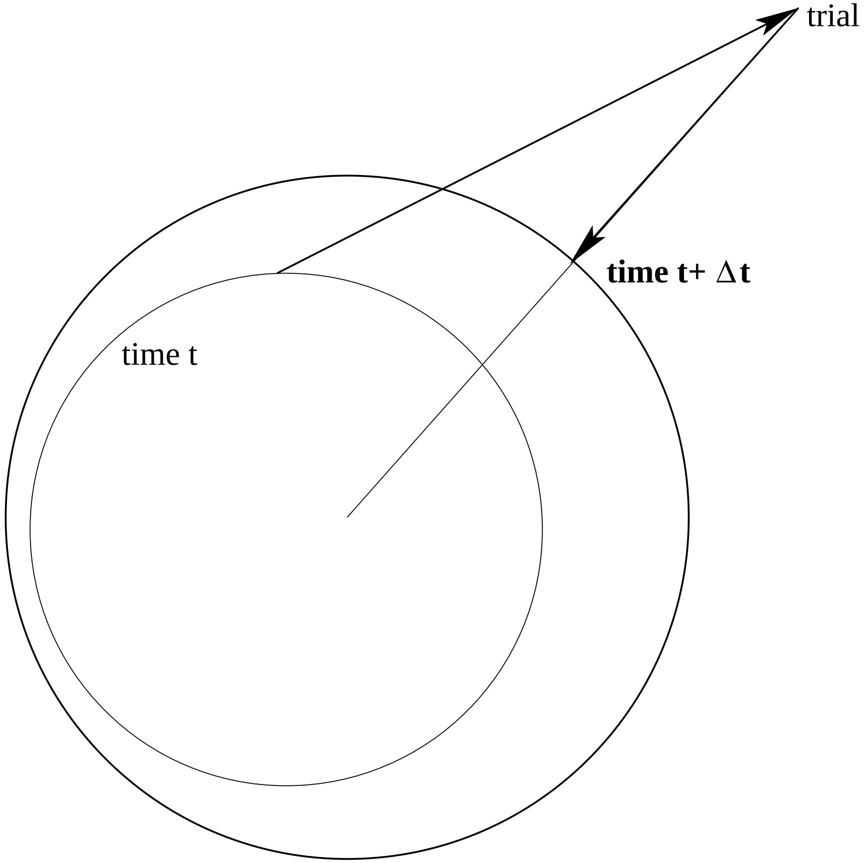
Time–dependent plasticity, now

$$r_p = \Delta p - \Delta t \, \Phi_\theta(J - R, \dots) = 0$$

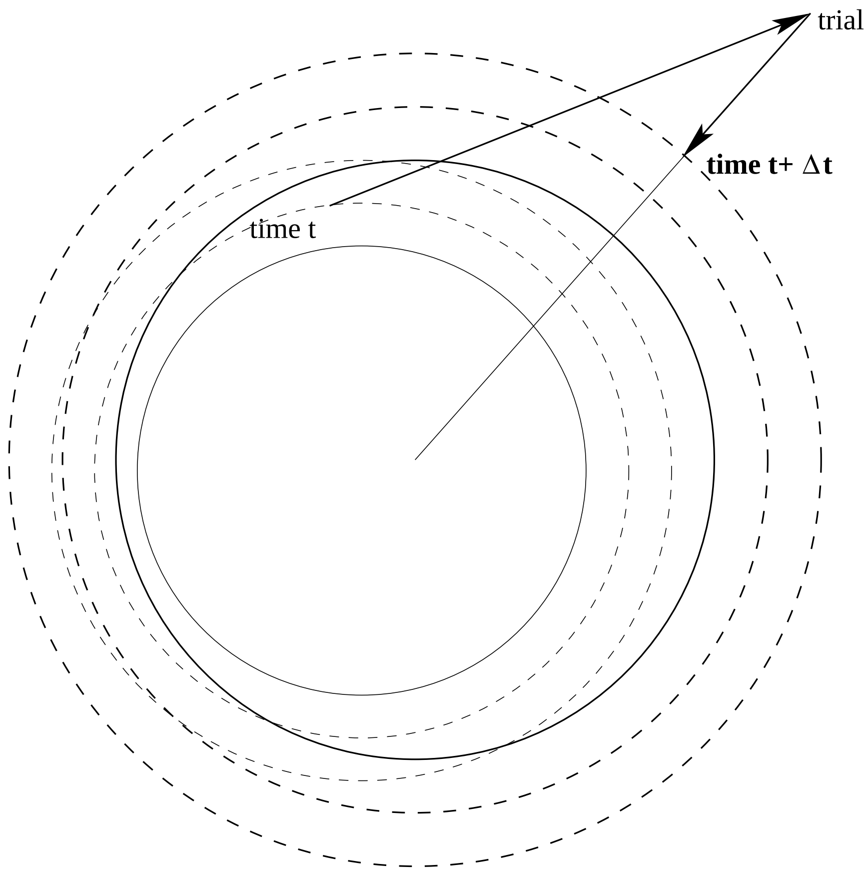
Assume $\Phi = K \left(\dfrac{\Delta p}{\Delta t} \right)^{1/n}$, then

$$\frac{\partial \Phi}{\partial \Delta \underset{\sim}{\varepsilon}^e} = 0 \quad ; \quad \frac{\partial \Phi}{\partial \Delta p} = \frac{K}{n \Delta t} \left(\frac{\Delta p}{\Delta t} \right)^{-1+1/n}$$

θ–method, TI and TD plasticity (1)

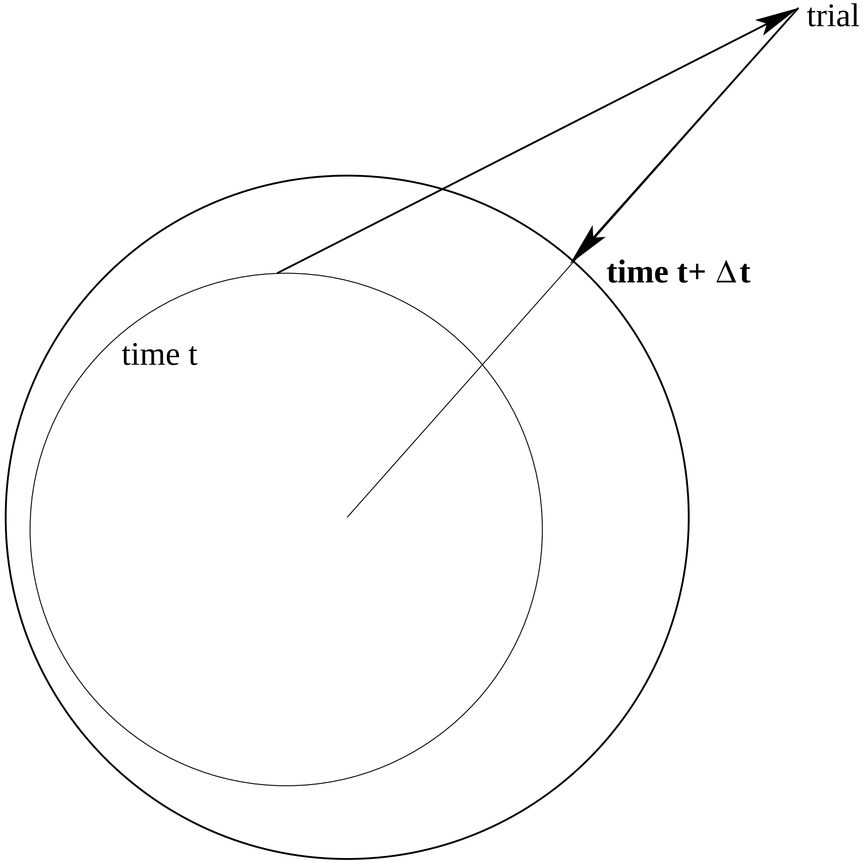


Time indep plasticity

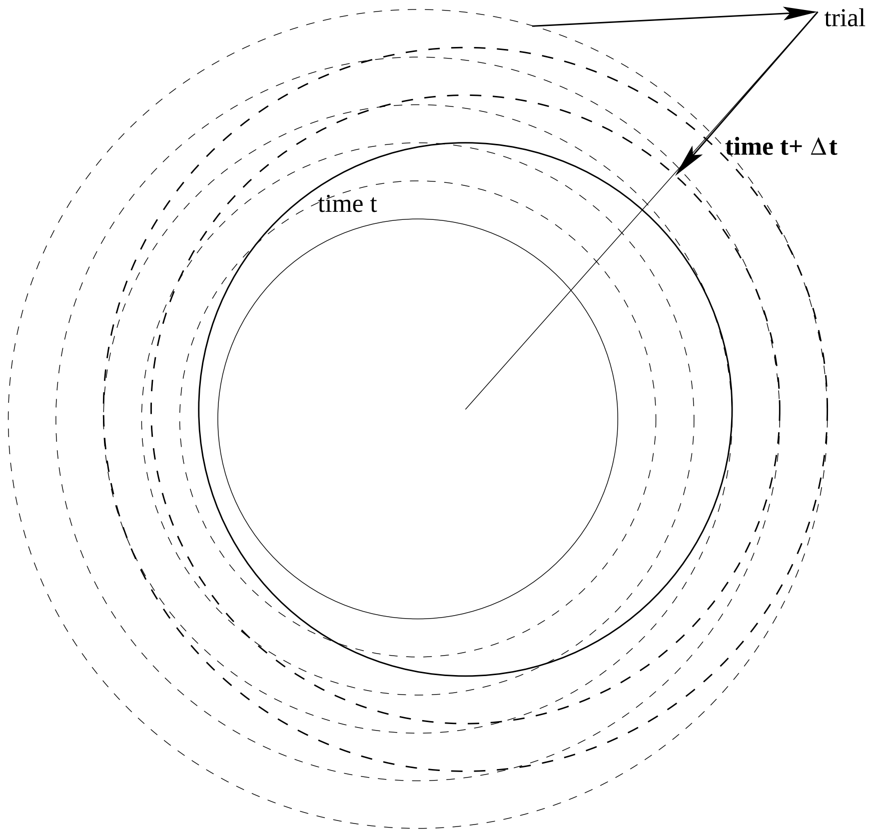


Time dep plasticity
increasing rate

θ–method, TI and TD plasticity (2)



Time indep plasticity



Time dep plasticity
decreasing rate

Implementation of models with multi-kinematic hardening (1)

Constitutive equations (i=1…N kinematic variables)

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

$$\dot{\underset{\sim}{\boldsymbol{\alpha}}}_i = \dot{\underset{\sim}{\boldsymbol{\varepsilon}}}^p - \dot{p}\underset{\approx}{\boldsymbol{D}}_i : \underset{\sim}{\boldsymbol{\alpha}}_i$$

$$\dot{r} = \dot{p} - \dot{p}br$$

Discrete counterpart

$$\begin{aligned}
\underset{\sim}{\boldsymbol{r}}_e &= \Delta\underset{\sim}{\boldsymbol{\varepsilon}}^e + \Delta p\underset{\sim}{\boldsymbol{n}} - \Delta\underset{\sim}{\boldsymbol{\varepsilon}} = \underset{\sim}{\boldsymbol{0}} \\
\underset{\sim}{\boldsymbol{r}}_{\alpha_i} &= \Delta\underset{\sim}{\boldsymbol{\alpha}}_i - \Delta p\,\underset{\sim}{\boldsymbol{n}} + \Delta p\underset{\approx}{\boldsymbol{D}}_i : \underset{\sim}{\boldsymbol{\alpha}}_i = \underset{\sim}{\boldsymbol{0}} \\
r_r &= \Delta r - \Delta p(1 - br) = 0 \\
r_p &= \Delta p - \phi(f, \dots)\Delta t = 0
\end{aligned}$$

Implementation of models with multi-kinematic hardening (2)

$$\frac{\partial \underset{\sim}{r}_e}{\partial \Delta \underset{\sim}{\varepsilon}^e} = \underset{\approx}{1} + \theta \Delta p \underset{\approx}{N} : \underset{\approx}{\Lambda} \quad (2)$$

$$\frac{\partial \underset{\sim}{r}_e}{\partial \Delta \underset{\sim}{\alpha}_i} = \Delta p \frac{\partial \underset{\sim}{n}}{\partial \underset{\sim}{X}_i} : \frac{\partial \underset{\sim}{X}_i}{\partial \underset{\sim}{\alpha}_i} : \frac{\partial \underset{\sim}{\alpha}_i}{\partial \Delta \underset{\sim}{\alpha}_i} = -\theta \Delta p \underset{\approx}{N} : \underset{\approx}{C}_i \quad (3)$$

$$\frac{\partial \underset{\sim}{r}_e}{\partial \Delta r} = 0 \quad (4)$$

$$\frac{\partial \underset{\sim}{r}_e}{\partial \Delta p} = \underset{\sim}{n} \quad (5)$$

$$\frac{\partial \underset{\sim}{r}_{\alpha_i}}{\partial \Delta \underset{\sim}{\varepsilon}^e} = -\Delta p \frac{\partial \underset{\sim}{n}}{\partial \underset{\sim}{\sigma}} : \frac{\partial \underset{\sim}{\sigma}}{\partial \underset{\sim}{\varepsilon}^e} : \frac{\partial \underset{\sim}{\varepsilon}^e}{\partial \Delta \underset{\sim}{\varepsilon}^e} = -\theta \Delta p \underset{\approx}{N} : \underset{\approx}{\Lambda} \quad (6)$$

$$\frac{\partial \underset{\sim}{r}_{\alpha_i}}{\partial \Delta \underset{\sim}{\alpha}_i} = \underset{\approx}{1} + \theta \Delta p \underset{\approx}{D}_i \quad (7)$$

$$\frac{\partial \underset{\sim}{r}_{\alpha_i}}{\partial \Delta r} = 0 \quad (8)$$

$$\frac{\partial \underset{\sim}{r}_{\alpha_i}}{\partial \Delta p} = -\underset{\sim}{n} + \underset{\approx}{D}_i : \underset{\sim}{\alpha}_i \quad (9)$$

(10)

Implementation of models with multi-kinematic hardening (3)

$$\frac{\partial r_r}{\partial \Delta \underset{\sim}{\varepsilon}^e} = \underset{\sim}{\mathbf{0}} \tag{11}$$

$$\frac{\partial r_r}{\partial \Delta \underset{\sim}{\boldsymbol{\alpha}}_i} = \underset{\sim}{\mathbf{0}} \tag{12}$$

$$\frac{\partial r_r}{\partial \Delta r} = 1 + \theta \Delta p \, d \tag{13}$$

$$\frac{\partial r_r}{\partial \Delta p} = b \, r \tag{14}$$

$$\frac{\partial r_p}{\partial \Delta \underset{\sim}{\varepsilon}^e} = -\frac{\partial \phi}{\partial f} \frac{\partial f}{\partial \underset{\sim}{\boldsymbol{\sigma}}} : \frac{\partial \underset{\sim}{\boldsymbol{\sigma}}}{\partial \underset{\sim}{\varepsilon}^e} : \frac{\partial \underset{\sim}{\varepsilon}^e}{\partial \Delta \underset{\sim}{\varepsilon}^e} = -\theta \Delta t \phi_{,f} \underset{\sim}{\boldsymbol{n}} : \underset{\approx}{\boldsymbol{\Lambda}} \tag{15}$$

$$\frac{\partial r_p}{\partial \Delta \underset{\sim}{\boldsymbol{\alpha}}_i} = \theta \Delta t \phi_{,f} \underset{\sim}{\boldsymbol{n}} : \underset{\approx}{\boldsymbol{C}}_i \tag{16}$$

$$\frac{\partial r_p}{\partial \Delta r} = -\frac{\partial \phi}{\partial R} \frac{\partial R}{\partial r} \frac{\partial r}{\partial \Delta r} \Delta t = \theta \Delta t c \phi_{,f} \tag{17}$$

$$\frac{\partial r_p}{\partial \Delta p} = 1 \tag{18}$$

***Presentation of the material library* Zmat**

- Numerous material models, plus `user` material
- Interface with the classical FE softwares
- Provide automatic time stepping and consistent tangent stiffness
- Coefficients presenting unlimited dependence on internal variables
- ZeBFRoNT, automatic code generation
- MuLTiMaT concept, for recursive multiscale modeling

ZeBFRoNT *concept*

- Preprocessor, using building bricks like `elasticity`, `flow`, etc...
- Use a macrolanguage, with a limited number of keywords like `Coefs`, `StrainPart`, `derivative`, `implicit`, etc...
- Generate C++ code

Explicit programming with **ZeBFRoNT**

`@Class NORTON_BEHAVIOR : BASIC_NL_BEHAVIOR`	
`{`	
`  @Name norton;`	Nom du comportement
`  @SubClass ELASTICITY elasticity;`	Objet matrice d'élasticité
`  @Coefs K, n;`	Coefficients de Norton
`  @tVarInt eel;`	Variable interne tensorielle : $\underset{\sim}{\varepsilon}_e$
`  @sVarInt evcum;`	Variable interne scalaire : p
`};`	
`@StrainPart {`	Calcul de la contrainte après intégration
`  sig = *elasticity*eel;`	$\underset{\sim}{\sigma} = \underset{\approx}{\mathbf{E}}\underset{\sim}{\varepsilon}_e$
`  m_tg_matrix=*elasticity;`	Matrice tangente approchée (RK !)
`}`	
`@Derivative {`	Calcul du vecteur dérivé $\underline{\dot{Y}}$
`  TENSOR2 sprime,norm;`	
`  double J;`	
`  sig=*elasticity*eel;`	
`  sprime=deviator(sig);`	Calcul du déviateur $\underset{\sim}{\sigma}'$
`  J=sqrt(1.5*(sprime|sprime));`	Calcul du deuxième invariant
`  devcum=pow(J/K,n);`	Fluage de Norton : $\dot{p} = (\frac{J}{K})^n$
`  norm=sprime*(1.5/J);`	Direction de l'écoulement
`  deel=deto-devcum*norm;`	Déformation élastique
`}`	

Implicit programming with ZeBFRoNT

```
@CalcGradF {
  ELASTICITY& E=*elasticity;
  sig = E*eel;
  f_vec_eel -= deto;
  TENSOR2 sigeff = deviator(sig);
  double J = sqrt(1.5*(sigeff|sigeff));
  if (J>(double)0.0) {
    TENSOR2 norm = sigeff*(1.5/J);
    f_vec_eel += norm*devcum;
    f_vec_evcum -= dt*pow(J/K,n);
    SMATRIX dn_ds = unit32;
    dn_ds -= norm ^ norm;
    dn_ds *= theta*devcum/J;
    deel_deel += dn_ds *E;

    deel_devcum += norm;
    double dv_df = tdt*n*pow(J/K,n-1)/K;
    TENSOR2 df_fs = dv_df*norm;
    devcum_deel -= df_fs*E;

  }
}
```

Intégration implicite

$\underset{\sim}{\sigma} = \underset{\approx}{\mathbf{E}}\,\underset{\sim}{\varepsilon}_e$

$\underset{\sim}{\mathbf{Re}} = \Delta\underset{\sim}{\varepsilon}_{\mathbf{e}} - \Delta\underset{\sim}{\varepsilon}$

Déviateur $\underset{\sim}{\sigma}'$

Deuxième invariant

Si on a plastifié

Direction de l'écoulement $\underset{\sim}{\mathbf{n}}$

$\underset{\sim}{\mathbf{Re}} = \Delta\underset{\sim}{\varepsilon}_{\mathbf{e}} - \Delta\underset{\sim}{\varepsilon} + \Delta p\,\underset{\sim}{\mathbf{n}}$

$\Delta p = \left(\frac{J}{K}\right)^n \Delta t$

$\frac{\partial \underset{\sim}{\mathbf{Re}}}{\partial \Delta\underset{\sim}{\varepsilon}_{\mathbf{e}}}$

$\frac{\partial \underset{\sim}{\mathbf{Re}}}{\partial \Delta p}$

$\frac{\partial Rp}{\partial \Delta\underset{\sim}{\varepsilon}_{\mathbf{e}}}$

$\frac{\partial Rp}{\partial \Delta p} = 1$

Multimat capabilities (1)

- Use homogenization rules
  - Localization rules
  - Local constitutive equations (possibly multimat)
- Macroscopic level (0)

```
***behavior mori_tanaka
**material 0.65 matrice
*file matrice.mat
**material  0.35 fibre
*file elas.mat
*rota-
tion  x1  0.2  0.3  0.4
          x2  0.7  0.1 -0.3
***return
```

- Material at level (1) to be defined

Multimat capabilities (2)

- Level 1

matrice.mat

```
***behavior berveiller_zaoui
  **mu  75000.  **nu  0.3
  **material  0.50 austenite
    *file austenite.mat
  **material  0.50 ferrite
    *file ferrite.mat
***return
```

fibre.mat

```
***behavior linear_elastic
 **elasticity orthotropic
   y1111 100000. y2222 120000.
       ...       y3131  90000.
***return
```

- Level 2

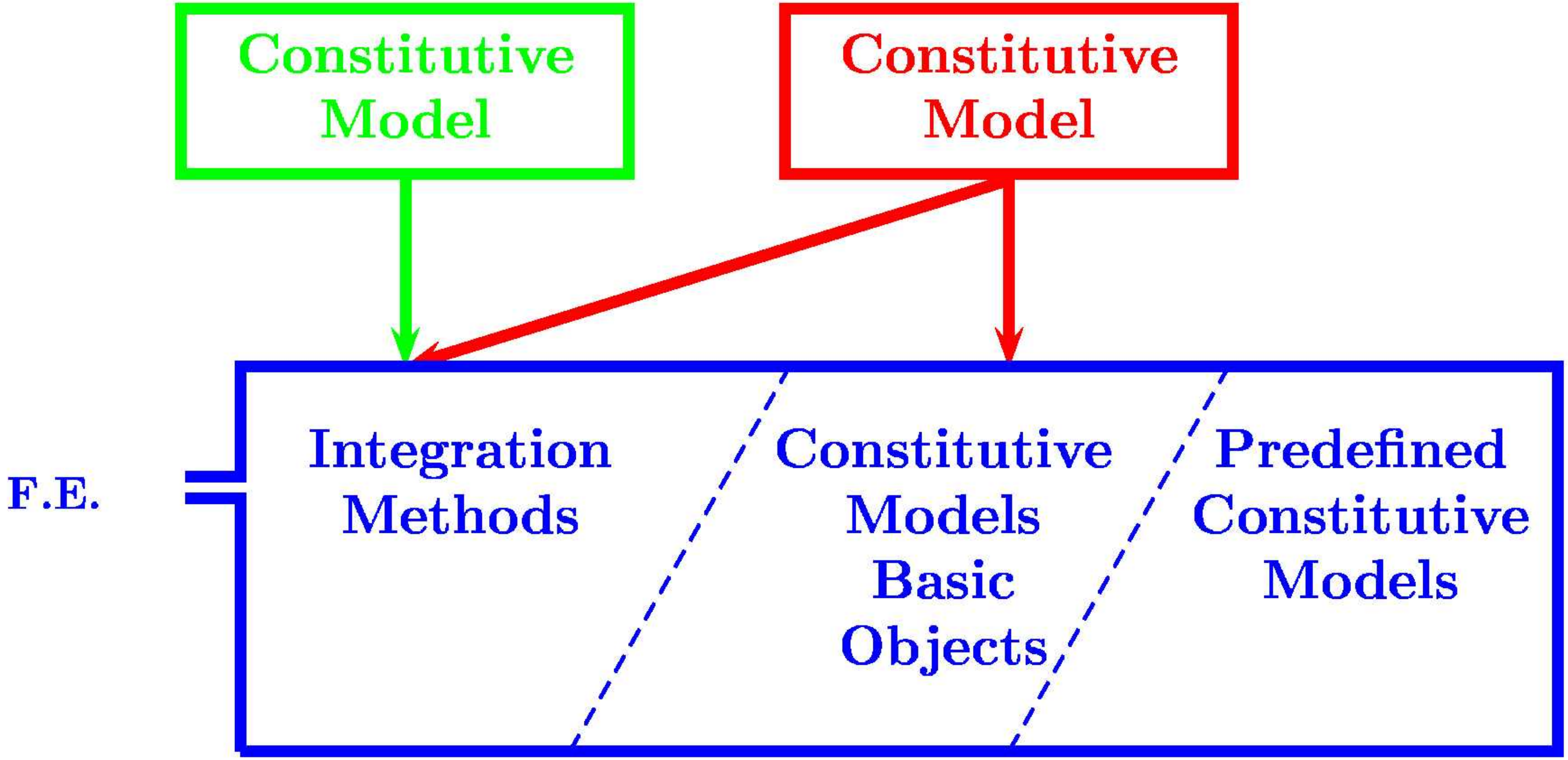
austenite.mat

```
***behavior gen_evp
 **elasticity isotropic
   young 260000. poisson 0.3
 **potential gen_evp ep
  *flow plasticity
  *isotropic constant
      R0     130.
***return
```

ferrite.mat

```
***behavior gen_evp
  ...
***return
```

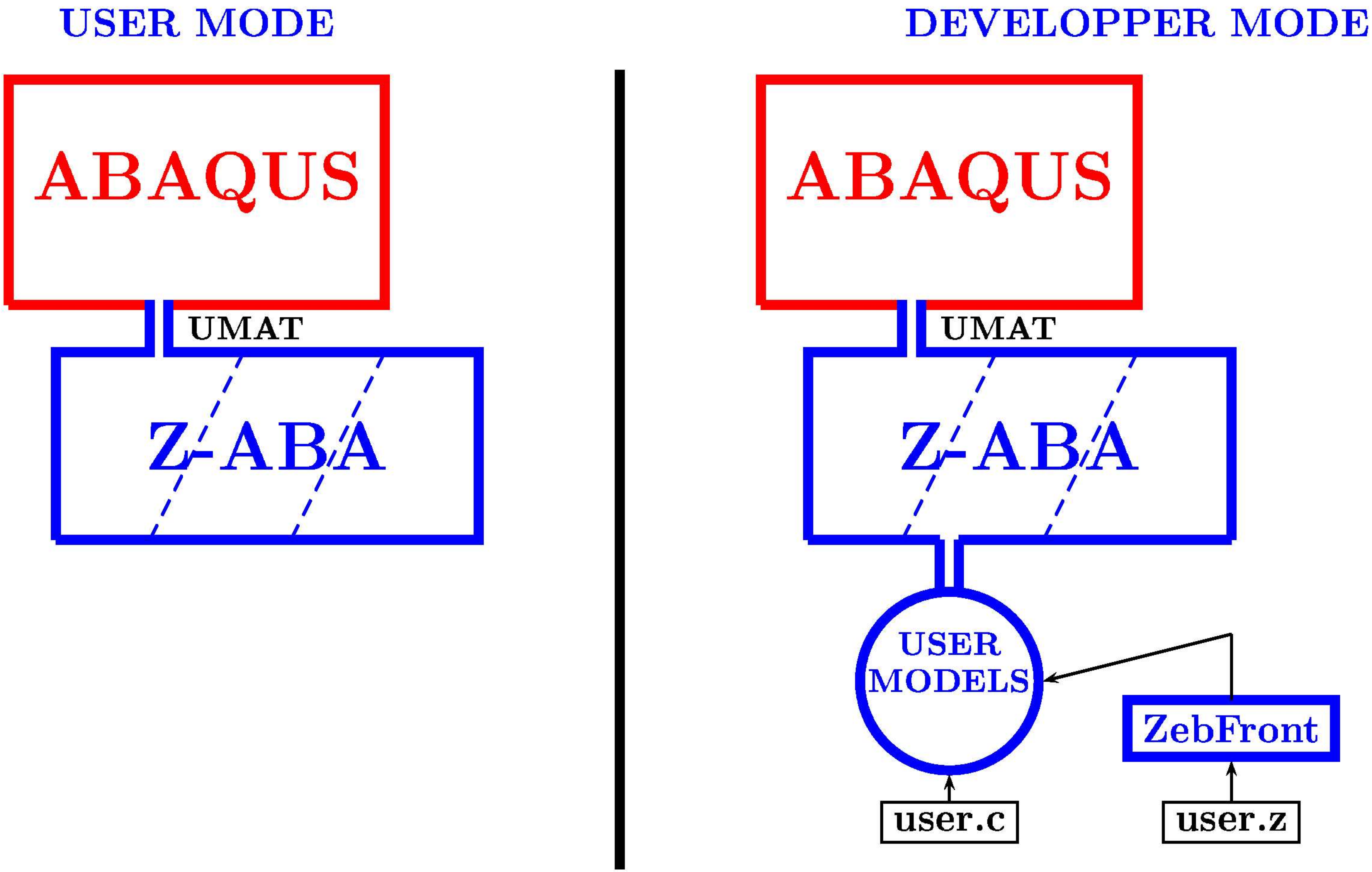
What is inside Zmat ?



- Runge-Kutta
- Theta method

- Elasticity
- Thermal Strain
- Isotropic Hardening
- Kinematic Hardening
- ...

Use of the material library Zmat



Use of the material library Zmat

ABAQUS input file:

```
****************************************
** ABAQUS INPUT FILE
****************************************
*NODE, NSET=all
1,0.,0.
...
**________________________________
*SOLID SECTION,ELSET=ALL,MATERIAL=steel
*MATERIAL,NAME=steel
*DEPVAR
13
*USER MATERIAL,CONSTANTS=1
0.0
*USER SUBROUTINES,INPUT=umat.f
**________________________________
...
```

Z-ABA material file: steel

```
***material
  *integration theta_method_a 1.0 1.e-12 1500
***behavior  gen_evp
  **elasticity isotropic
     young   210000.0
     poisson  0.33
  **potential gen_evp ep
    *criterion  mises
    *flow  plasticity
    *isotropic constant
      R0 150.0
    *kinematic nonlinear
      D  69.31
      C  8317.77
***return
```

Execution : > Z-aba struc

Object oriented modular design in Zmat

Material objects

- Flow
- Behavior
- Isotropic hardening
- Criterion
- Kinematic hardening
- Damage
- Elasticity
- Potential
- Thermal strain
- etc ...

Typical assembly for viscoplasticity

- Behavior
  - Elasticity
  - Thermal strain
  - Potential
    - Criterion
    - Flow
    - Isotropic hardening
    - Kinematic hardening
    - Kinematic hardening

Isotropic and nonlinear kinematic model in Zmat

behavior	
elasticity isotropic	
thermal_strain isotropic	$\underset{\sim}{\epsilon}^{th} = \alpha(T - T_{ref})$
potential ev	
criterion mises	$f = J\left(\underset{\sim}{\sigma}' - \sum_i \underset{\sim}{\mathbf{X}}_i\right) - R$
flow norton	$\dot{p} = \left\langle \frac{f}{K} \right\rangle^n \quad , \quad \dot{\underset{\sim}{\epsilon}}^{ev} = \dot{p}\underset{\sim}{\mathbf{n}}$
isotropic nonlinear	$R = R_0 + Q(1 - e^{-bp})$
kinematic nonlinear	$\underset{\sim}{\mathbf{X}} = \frac{2}{3}C\underset{\sim}{\alpha} \quad , \quad \dot{\underset{\sim}{\alpha}} = \dot{p}\left[\underset{\sim}{\mathbf{n}} - \frac{3D}{2C}\underset{\sim}{\mathbf{X}}\right]$

Example of data files in Zmat

Plasticity

```
***behavior gen_evp
 **elasticity isotropic
  young    100000.
  poisson  0.3
 **potential gen_evp ep
  *criterion mises
  *flow plasticity
  *isotropic nonlinear
    R0 210. Q 50. b 10.
  *kinematic nonlinear
    C 20000. D 500.
```

Viscoplasticity

```
***behavior gen_evp
 **elasticity isotropic
  young    100000.
  poisson  0.3
 **potential gen_evp ev
  *criterion mises
  *flow norton
    K 1000. n 4.5
  *isotropic nonlinear
    R0 210. Q 50. b 10.
  *kinematic nonlinear
    C 20000. D 500.
```

Crystal viscoplasticity

```
***behavior gen_evp
 **elasticity cubic
  y1111    100000.
  y1122    75000.
  y1212    112000.
 **potential octahedral
  *flow norton
    K 1000. n 4.5
  *isotropic nonlinear
    R0 210. Q 50. b 10.
  *kinematic nonlinear
    C 20000. D 500.
  *interaction slip
 h1 1. h2 1.2 h3 1.4 h5 1.3 h6 1.8
```

Use of the material library Zmat

- ZebFront model: model.z

```
@Class ZUSER : BASIC_NL_BEHAVIOR {
  @Name      Zuser;
  @SubClass ELASTICITY E;
  @Coefs     C1, C2;
  @VarInt    evi, eel, ...;
  @VarAux    X, Y;  }
@StrainPart {
  evi = eto - eel;
  sig = *E*eel;  }
@Derivative {
 @CalcCoeffs;
 devi    =  ...;
 deel    =  ...;  }
```

- Abaqus input file:

```
******************************************
...
*SOLID SECTION,ELSET=ALL,MATERIAL=steel
*MATERIAL,NAME=steel
*USER SUBROUTINES,INPUT=umat.f
...
```

- Compilation

> ZebFront model.z

- Link Z-ABA library:

> MAKE

- Z-ABA material file: steel

```
***material
 *integration theta_method_a 1.0 1.e-12 1500
***behavior  Zuser
 **elasticity isotropic
   young     210000.0
   poisson  0.33
 **model_coefficients
   C1        100.
   C2        100.
***return
```

Implementation of a Norton model in Zmat

Norton model with derivation of the material tangent matrix

```
@Class NORTON : BASIC_NL_BEHAVIOR {
  @name     norton;
  @SubClass ELASTICITY elasticity;
  @Coefs    K, n;
  @tVarInt  eel;
  @sVarInt  evcum;
  @Implicit
};
@StrainPart {
 sig = *elasticity*eel;  }
@Derivative {
 sig = *elasticity*eel;
 TENSOR2 sprime = deviator(sig);
 double  J      = sqrt(1.5*(sprime | sprime));
 TENSOR2      norm      = sprime*(1.5/J);
                 devcum    = pow(J/K,n);
                 deel      = deto - devcum*norm;
}
@CalcGradF {
 ELASTICITY& E=*elasticity;
 double  J      = sqrt(1.5*(sprime | sprime));
 if (J>0.0) {
   f_vec_eel      += -deto + norm*devcum;
   f_vec_evcum    -= dt*pow(J/K,n);
   TENSOR2 df_fs=n*pow(J/K,n-1.)/K*norm;
   SMATRIX dn_ds = (unit32-norm^norm)/J;
   deel_deel         += theta*devcum*dn_ds*E;
   deel_devcum       += norm;
   devcum_deel       -= theta*dt*df_fs*E;
 }
}
```

$\underset{\sim}{\sigma} = \underset{\approx}{E}\,\underset{\sim}{\epsilon}^{el}$

$f = J = \sqrt{\frac{3}{2}\underset{\sim}{\sigma}' : \underset{\sim}{\sigma}'}$

$\underset{\sim}{n} = \frac{\partial f}{\partial \underset{\sim}{\sigma}} = \frac{3}{2J}\,\underset{\sim}{\sigma}'$

$\dot{v} = \left\langle \frac{f}{K} \right\rangle^n$

$\dot{\underset{\sim}{\epsilon}}^{el} = \dot{\underset{\sim}{\epsilon}} - \dot{v}\,\underset{\sim}{n}$

Residual:

$\underset{\sim}{R}_{el} = \Delta\underset{\sim}{\epsilon}_{el} - \Delta\underset{\sim}{\epsilon} + \Delta v\,\underset{\sim}{n}$

$R_v = \Delta v - \left\langle \frac{f}{K} \right\rangle^n \Delta t$

Jacobian matrix:

$\frac{\partial \underset{\sim}{n}}{\partial \underset{\sim}{\sigma}} = \frac{3}{2\,J}\left(\underset{\approx}{1} - \underset{\sim}{n} \otimes \underset{\sim}{n}\right)$

$\frac{\partial \underset{\sim}{R}_{el}}{\partial \Delta\underset{\sim}{\epsilon}_{el}} = \underset{\approx}{1} + \theta\Delta v \frac{\partial \underset{\sim}{n}}{\partial \underset{\sim}{\sigma}} : \underset{\approx}{E}$

$\frac{\partial \underset{\sim}{R}_{el}}{\partial \Delta v} = \underset{\sim}{n}$

$\frac{\partial R_v}{\partial \Delta\underset{\sim}{\epsilon}_{el}} = -\theta\Delta t \frac{n}{K}\left\langle \frac{f}{K} \right\rangle^{n-1} \underset{\sim}{n}\ \underset{\approx}{E}$

4

- Nothing else is needed
- Valid for explicit and implicit integration mode
- All the coefficients are known by the code, they can depend on external parameters and internal variables
- The variables are automatically known by the code for postprocessing

Implementation of an aging model in Zmat

Modeled by a cyclic viscoplastic law, with a time and temperature dependent variable, a.

$$\dot{a} = \frac{a_\infty(T) - a}{\tau(T)}$$

Yield stress $R = R_{classical} + R^*$

$$R^* = R_0^*(T) + (1. - a)$$

a_∞, τ, R_0^* **are temperature dependent material coefficients**

Implementation of an aging model in Zmat

```
@Class AGEING_SIMUL : BASIC_NL_BEHAVIOR {
@SubClass PARAMETRIC_STRAIN  thermal_strain;
  @Name        ageing;
  @SubClass ELASTICITY E;
  @Coefs       R0, Q, b, R0_star, alfa;
  @Coefs       C, K, n, D, tau;
  @tObservable sig,eto;
  @tVarInt     evi, eth, alpha;
  @VarInt      evcum, age;
  @VarAux      R;
  @tVarAux     eel, X;
};
@StrainPart {
  eel = eto-evi-eth;
  sig = *E*eel;
}
@Derivative {
 @CalcCoeffs;
 X = (2./3.)*C*alpha;
 double      R_star  = R0_star*(1. - age);
             R       = R0 + Q*(1.-exp(-b*evcum));
 TENSOR2  sigeff  = deviator(sig) - X;
 double      J       = sqrt(1.5*(sigeff|sigeff))
 double      f       = J - R - R_star;
                dage  = (1. - age)/tau;
 TENSOR2 deth = thermal_strain->compute_dstrain();
 if (f>0.0) {
                   devcum   = pow(f/K, n);
     TENSOR2   norm     = sigeff*(1.5/J);
                   devi     = devcum*norm;
                   dalpha   = devi - D*alpha*devcum;
 }
 else              devi = dalpha = 0.0;
}
```

$$\underset{\sim}{\epsilon}^{el} = \underset{\sim}{\epsilon} - \underset{\sim}{\epsilon}^{p} - \underset{\sim}{\epsilon}^{th}$$

$$\underset{\sim}{\sigma} = \underset{\approx}{\mathbf{E}}\ \underset{\sim}{\epsilon}^{el}$$

$$\underset{\sim}{\mathbf{X}} = \frac{2}{3}\ \mathbf{C}\ \underset{\sim}{\alpha}$$

$$R^* = R_0^*(1-a)$$

$$R = R_0 + Q(1-e^{bv})$$

$$\underset{\sim}{\sigma}_{ef} = \underset{\sim}{\sigma}' - \underset{\sim}{\mathbf{X}}$$

$$J = (\mathbf{1.5}\ \underset{\sim}{\sigma}_{ef} : \underset{\sim}{\sigma}_{ef})^{0.5}$$

$$\mathbf{f} = \mathbf{J}(\underset{\sim}{\sigma}\text{-}\underset{\sim}{\mathbf{X}})\ \text{-}\mathbf{R}\ \text{-}\ \mathbf{R}^*$$

$$\dot{a} = \frac{(1-a)}{\tau}$$

$$\dot{\underset{\sim}{\epsilon}}^{th} = \dot{\alpha}\ \underset{\sim}{\mathbf{I}}\ T - \alpha\ \underset{\sim}{\mathbf{I}}\ \dot{T}$$

$$\dot{v} = \left(\frac{f}{K}\right)^n$$

$$\underset{\sim}{\mathbf{n}} = \frac{3}{2J}\ \underset{\sim}{\sigma}_{ef}$$

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

$$\dot{\underset{\sim}{\alpha}} = \dot{\underset{\sim}{\epsilon}}^{p} - D\ \underset{\sim}{\alpha}\ \dot{v}$$

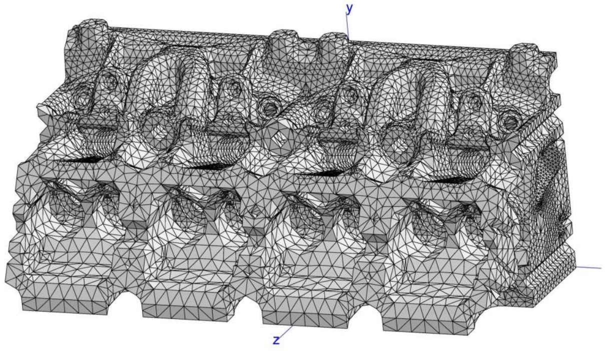
5

- Valid for explicit mode
- The thermal dilatation can be customized if needed
- Use of preprogrammed building bricks

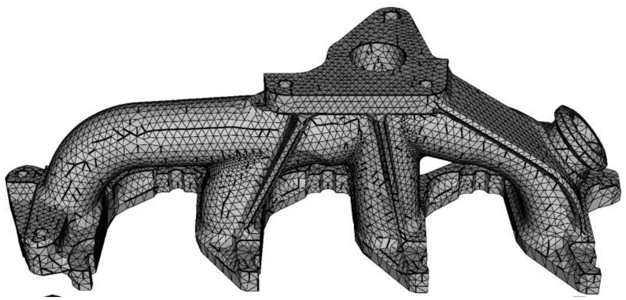
Parallel computations

- ZéBuLoN FE code
- Z-mat material library
- Computations on a linux PC cluster
- FETI method for parallel computation (Farhat-Roux, coll. Onera/Feyel)

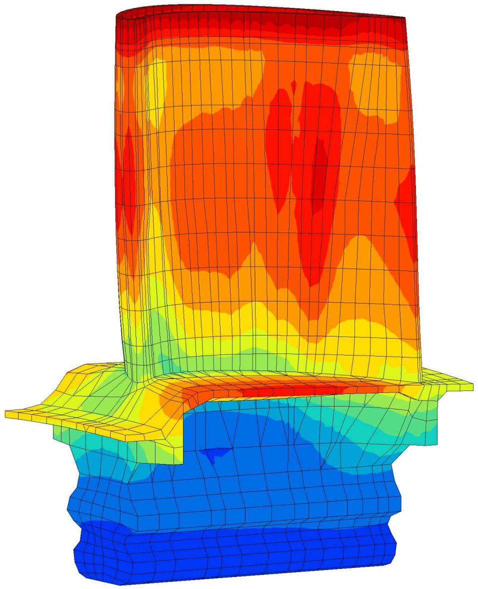
Example of recent computations



Head engine

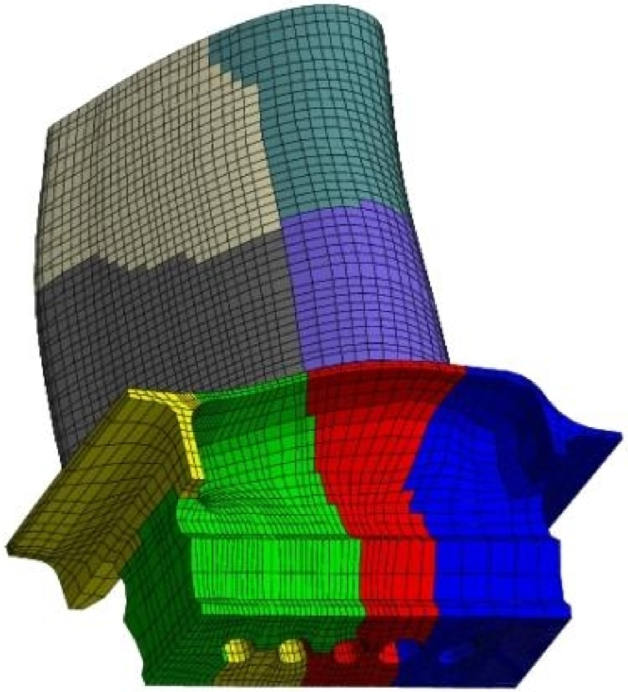


Exhaust manifold

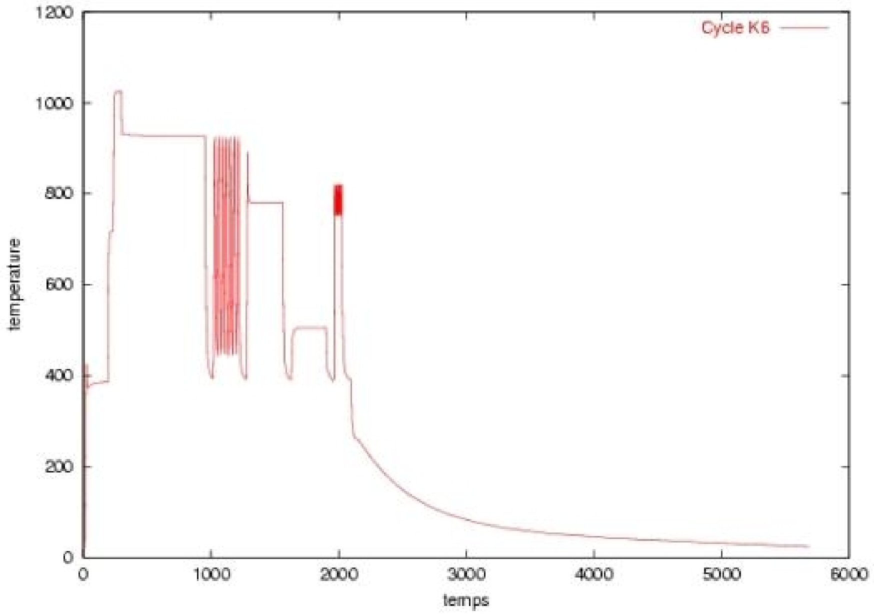


Turbine blades

Turbine blade: computation of several hundreds of cycles



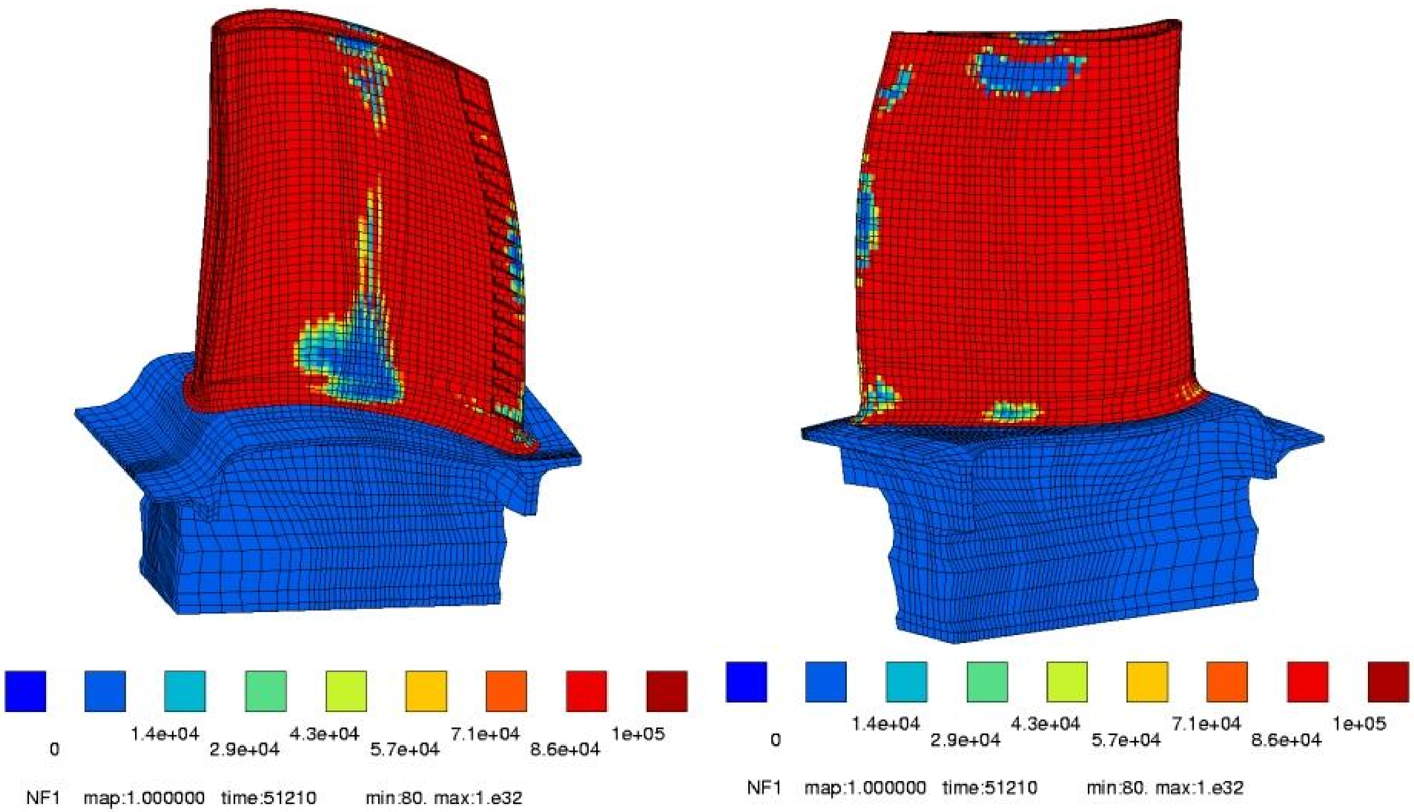
81000 dofs



68 incréments

Single crystal model

Post-processing for lifetime prediction



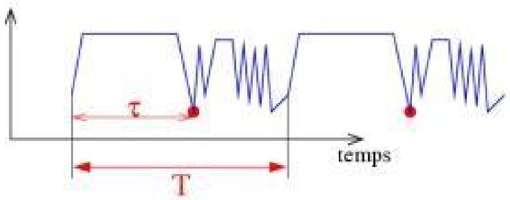
Performance of a PC cluster on the "small" turbine blade

	Temps total	Stockage matrice globale	Nombre d'itérations
Abaqus	1073 s		
Sparse dscpack	1203 s	515 Mo	
// 8 SDs + dscpack	91 s	41 Mo	120

The CPU time is reduced by a factor of 13 with 8 processors !

Use of a cycle skip technique

$Y(N)$ variables internes au cycle N



$$Y(N) = y((N-1)T + \tau)$$

$$Y(N) = \begin{bmatrix} \underset{\sim}{\varepsilon}^{e} \\ \bar{\varepsilon}^{p1} \\ \underset{\sim}{\alpha} \\ \bar{\varepsilon}^{p2} \end{bmatrix}$$

Extrapolation → Développement de Taylor à l'ordre 2

$$Y(N + \Delta N) = Y(N) + \Delta N \, Y'(N) + \frac{\Delta N^2}{2} Y''(N)$$

ΔN ? $Y'(N)$? $Y''(N)$?

Estimation of the cycle skip

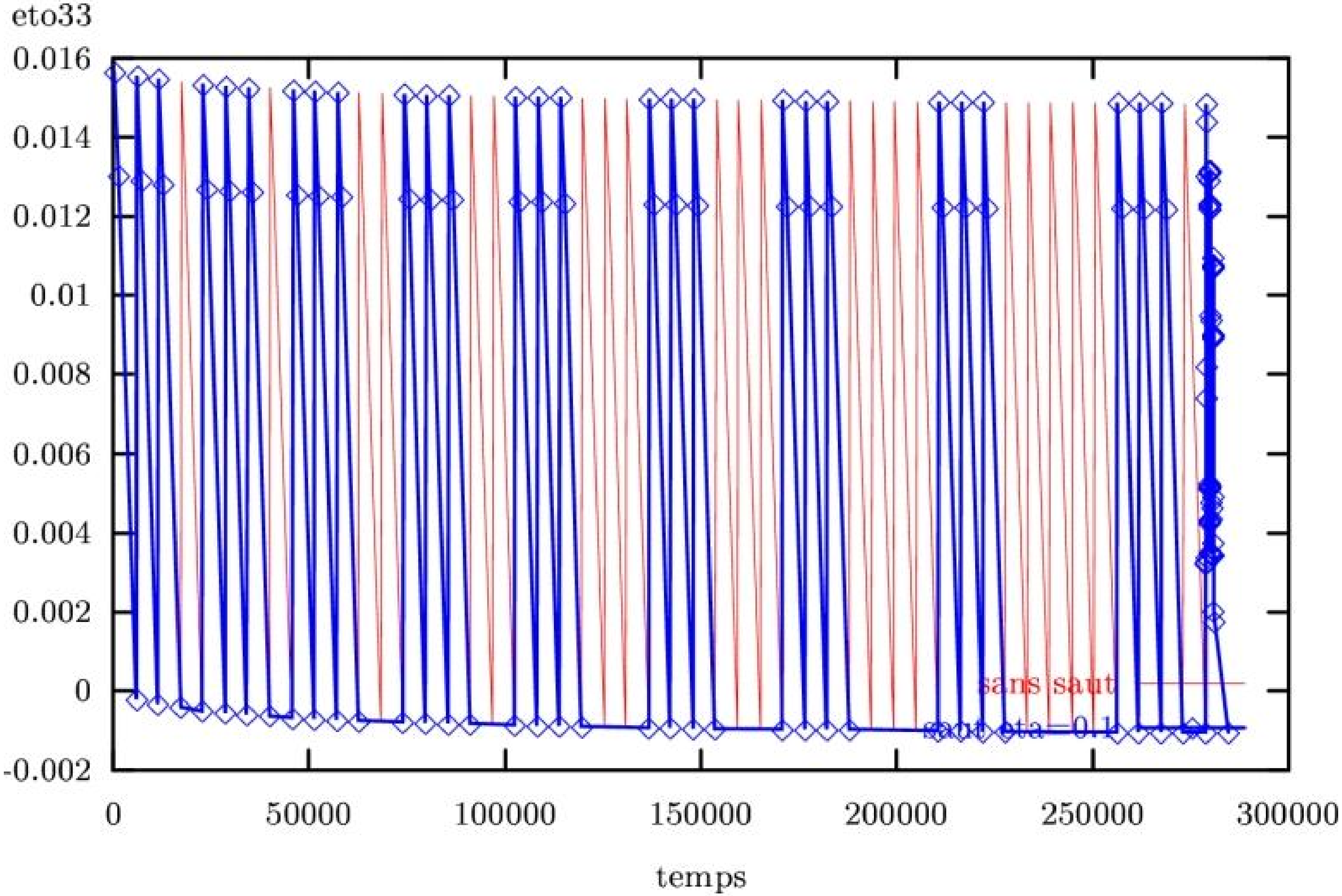
$$Y(N + \Delta N) = Y(N) + \Delta N\, Y'(N) + \frac{\Delta N^2}{2} Y''(N)$$

$$\begin{cases} Y(M) = Y(N + M - N) = Y(N) + (M - N)Y'(N) + \frac{(M-N)^2}{2} Y''(N) \\ Y(K) = Y(N + K - N) = Y(N) + (K - N)Y'(N) + \frac{(K-N)^2}{2} Y''(N) \end{cases}$$

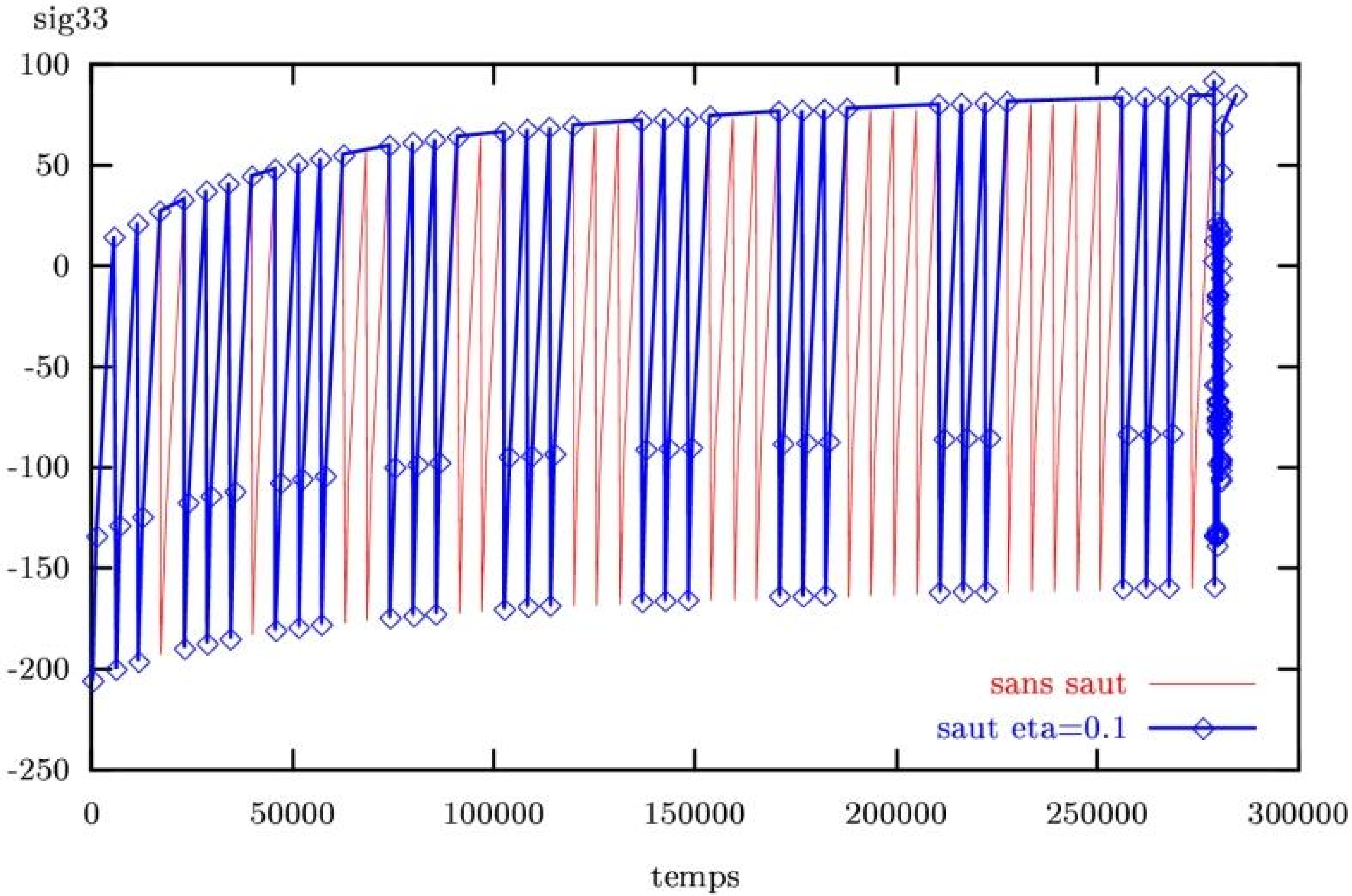
$$\frac{\Delta N^2}{2} Y''(N) = \eta \Delta N\, Y'(N) \implies \Delta N = 2\eta \frac{Y'(N)}{Y''(N)}$$

$$\eta \approx 0.05$$

Skip history in the time-strain diagram



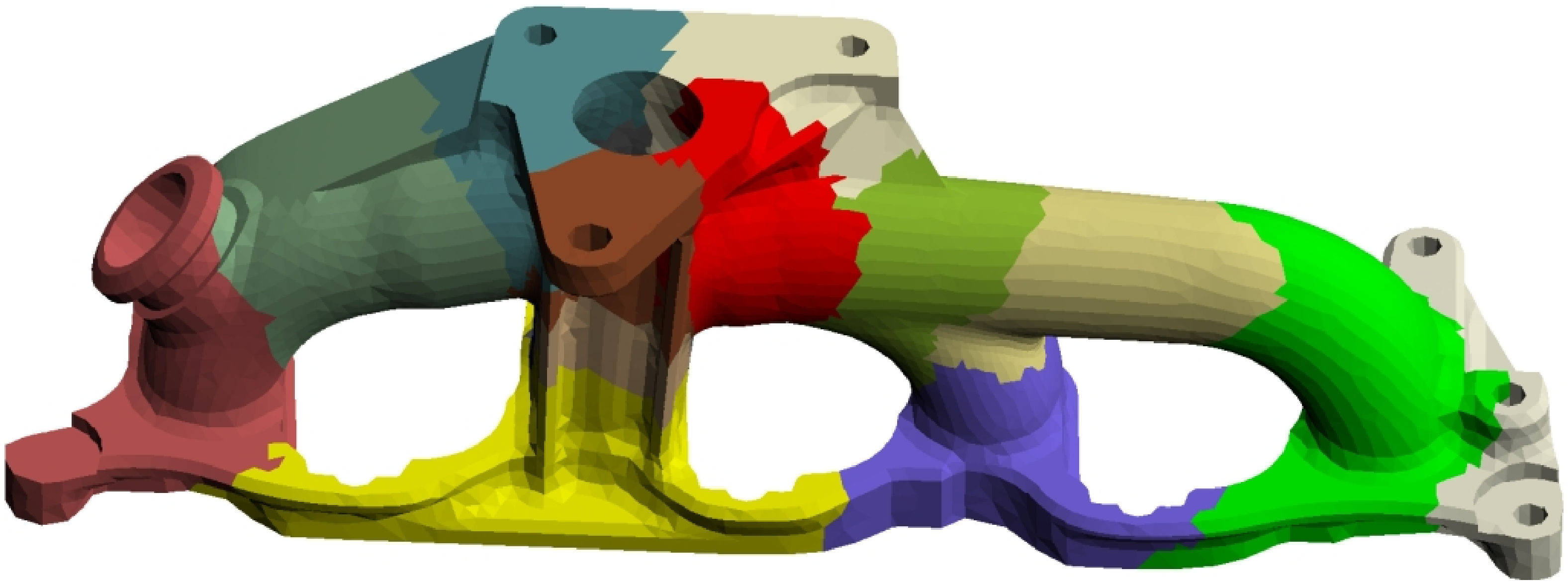
Skip history in the time-stress diagram



Comparison of the cycle skip technique with the direct calculation

	Référence	Saut	écart (%)
U1	0.5989	0.5989	0.00
S11	931.7	932.1	0.043
E11	7.4918E-03	7.493000E-03	0.016
eel11	3.5321E-03	3.5372E-03	0.14
evrcum	5.4729E-02	5.478E-02	0.093
evlcum	3.9179E-06	3.8936E-06	0.62

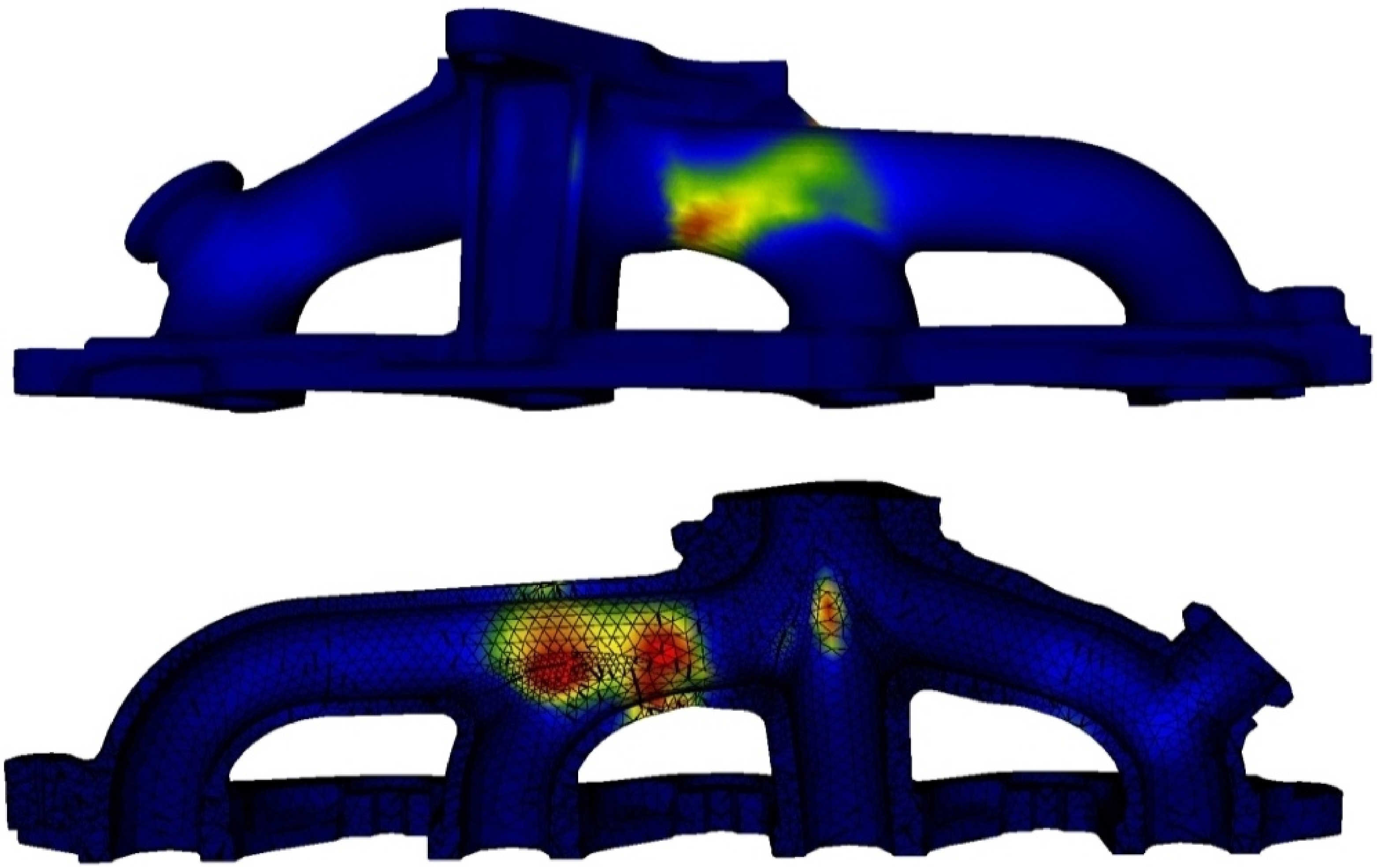
Computation of an exhaust manifold



345000 dof, Viscoplastic constitutive equations including aging

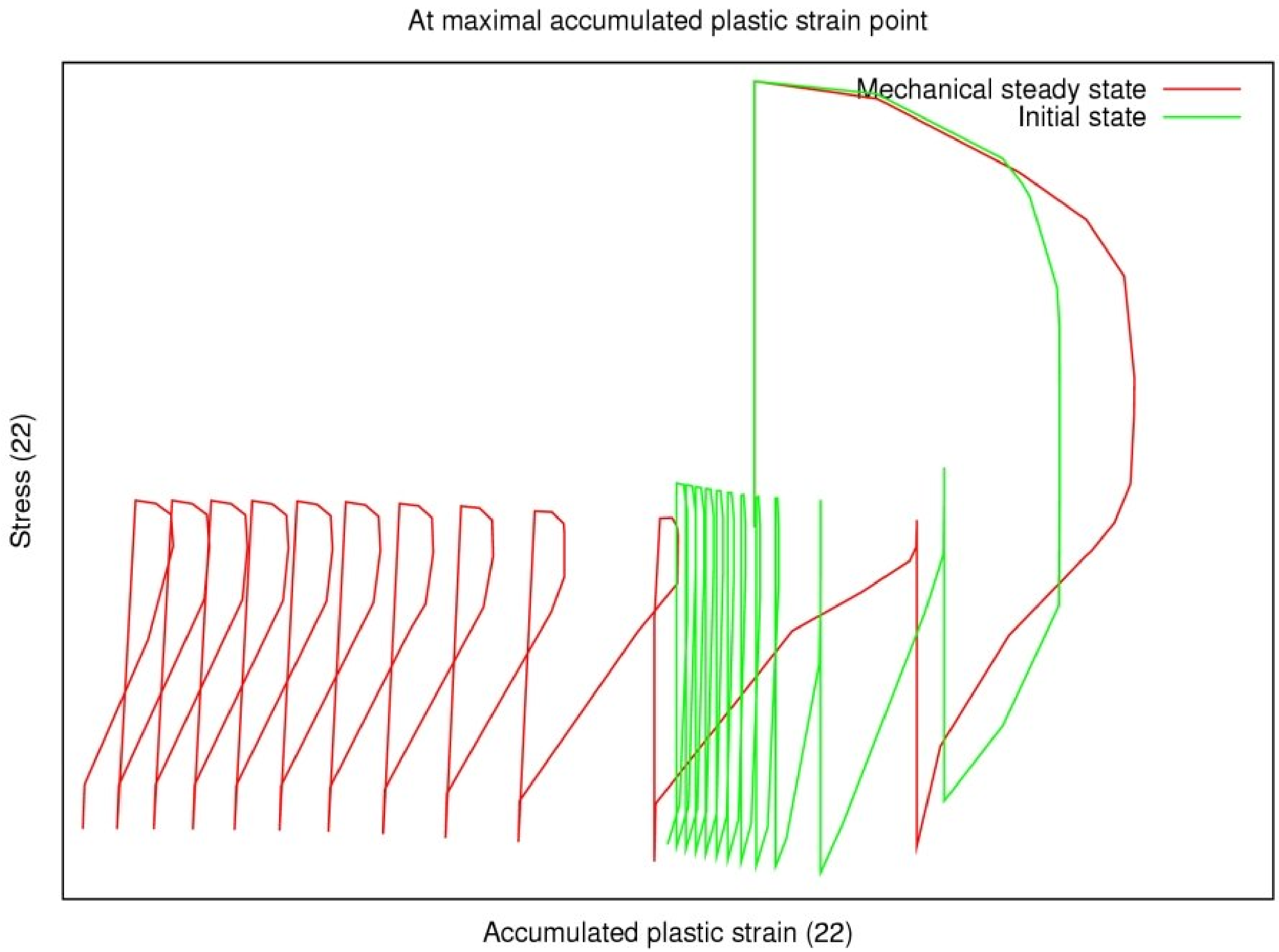
Mesh decomposition for parallel computing

Computation of an exhaust manifold (2)



Life computed after 12 cycles

Computation of an exhaust manifold (3)



Sensitivity of the constitutive equations

Numerical implementation

- **Open framework needed to implement new constitutive equations**
- **With parallel computing, problems in the range 10^5–10^6 can be solved**
- **Consistent lifeprediction rule**
- **MORE on aluminium alloys**
- **MORE on GS cast iron**
- **MORE on grey cast iron**

–Do it yourself–