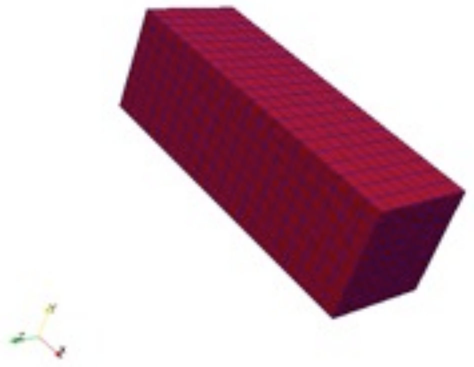
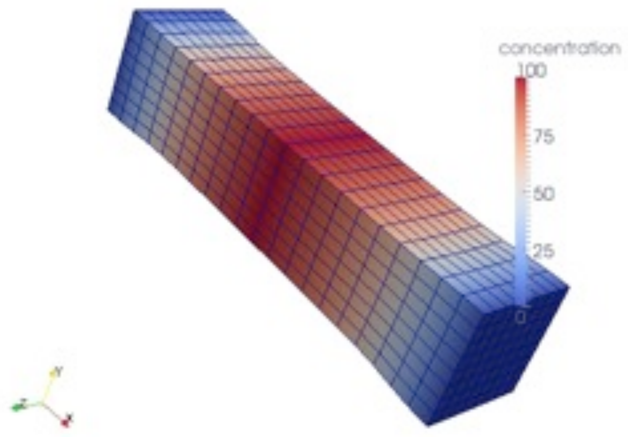
Coupled Problems in Nonlinear Solid Mechanics: Non-Fickian Diffusion

Deal.II Workshop

August 23 - 25 2010
Heidelberg

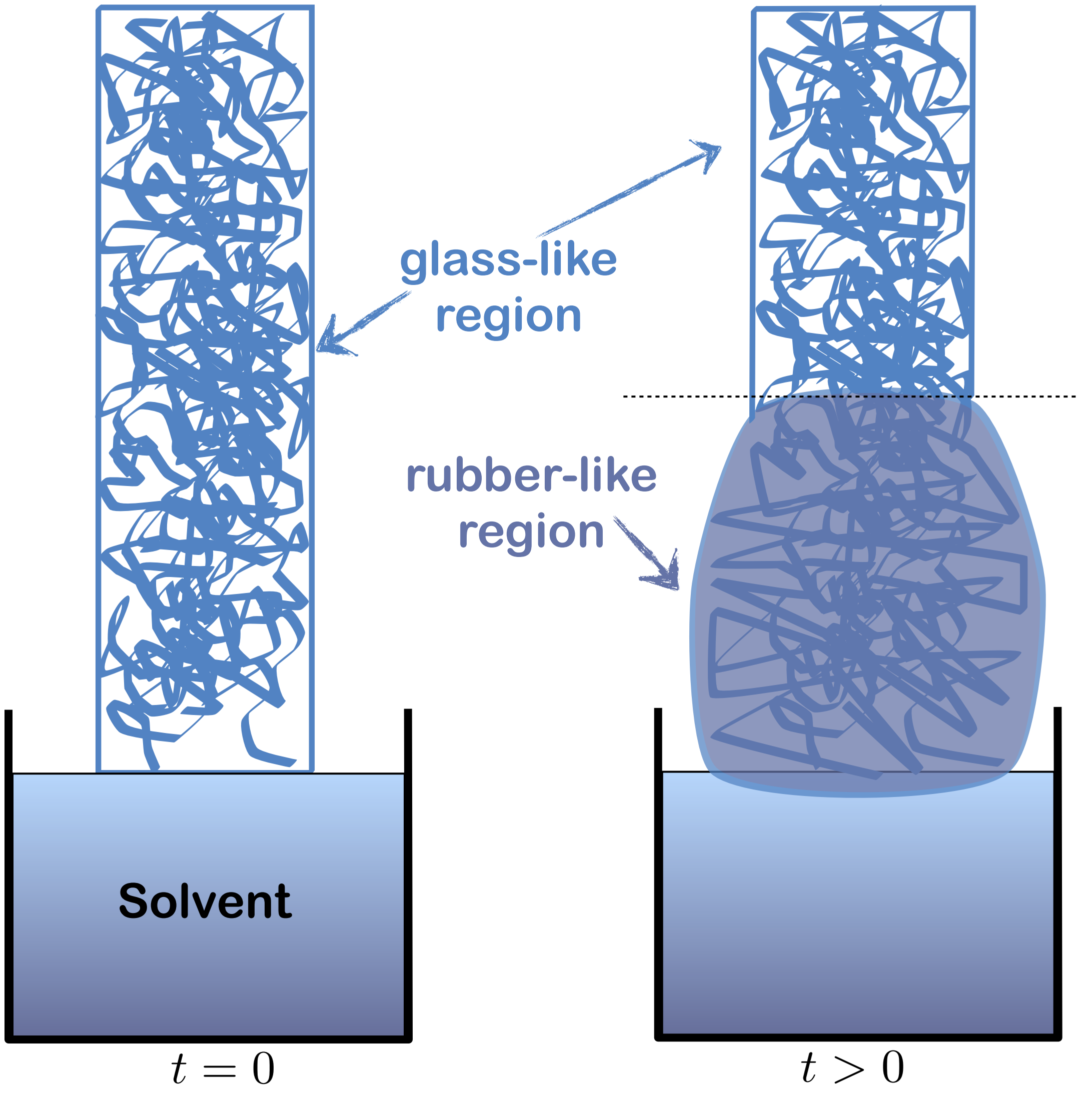




Andrew McBride and Paul Steinmann
Chair of Applied Mechanics
Friedrich-Alexander-University Erlangen-Nuremberg
Swantje Bargmann
Institute of Mechanics
Technische Universität Dortmund

Motivation: Case II diffusion

- Case II diffusion (CII) occurs during diffusion of low molecular weight solvent in polymeric solid
- Solid is originally in a glass-like state
- Solvent wave progresses through polymer at fixed rate (constant velocity)
- Solvent causes polymer change to a rubber-like material
- Highly-coupled non-linear phenomenon
  - Relaxation time of polymer depends on concentration and swelling
  - Solvent exerts a pressure on polymer and visa-vera (swelling and concentration dependent)
  - Diffusivity depends on swelling and concentration
  - Polymer needs finite amount time to rearrange to accommodate solvent limiting diffusion rate
  - Large swelling in rubber-like region
  - Polymer near-incompressible
- See DE KEE ET AL (2005) and VESELY (2008) for reviews



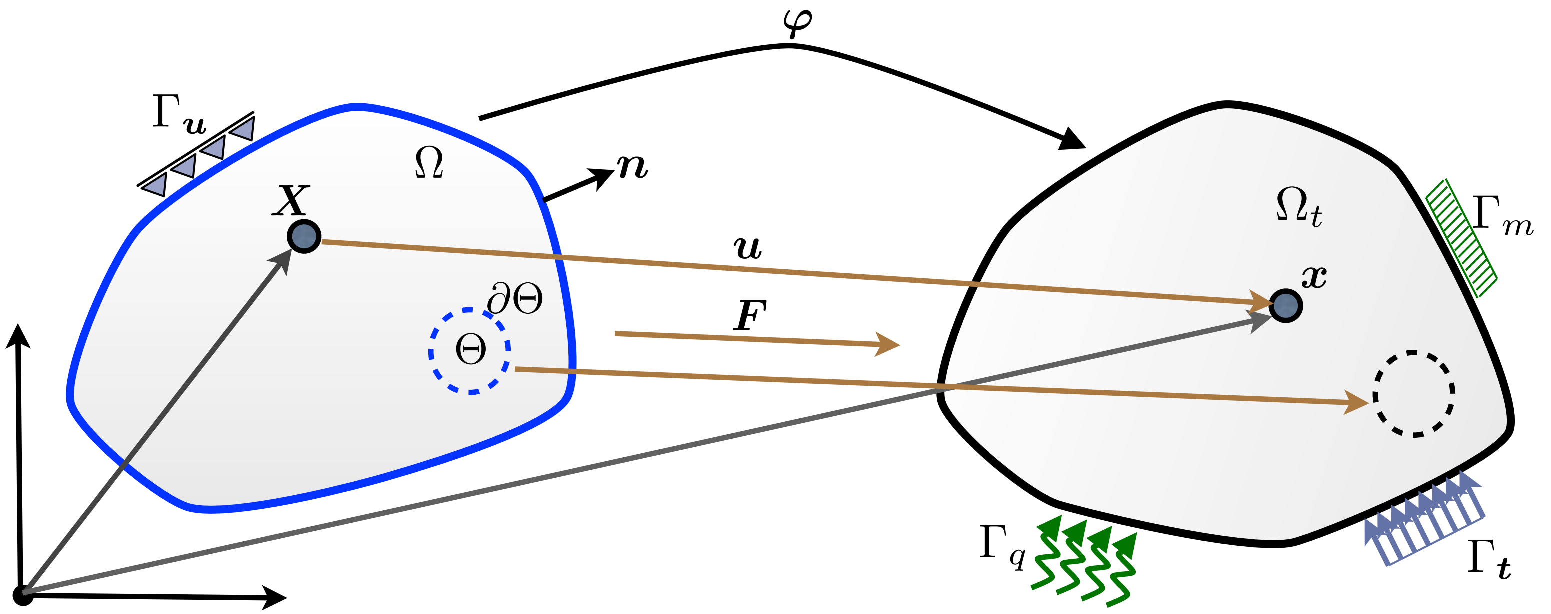
Wednesday 25 August 2010

Overview

- Applications based presentation
  - Case II a prototype for highly non-linear coupled problems
- Describe the governing equations
  - Focus on internal variable formulation of viscoelasticity
  - Integration algorithms for internal inelastic variables
- Review models for case II diffusion
  - Strongly coupled diffusion-deformation
- Solution strategies using finite elements
- Consider a reduced model
  - Focus on spatial adaptivity with internal variables in deal.II
- Example problem

Work in progress (does not actually all work yet): comments and suggestions greatly appreciated. Actually, a lot of the questions I ask in the talk have been answered and better strategies proposed. Thanks

Governing equations



$$\Gamma = \Gamma_{\boldsymbol{u}} \cup \Gamma_{\boldsymbol{t}}$$
$$\Gamma_{\boldsymbol{u}} \cap \Gamma_{\boldsymbol{t}} = \emptyset$$
$$\Gamma = \Gamma_m \cup \Gamma_q$$
$$\Gamma_m \cap \Gamma_q = \emptyset$$
$$\bar{\Omega} = \overline{\Omega \cup \Gamma}$$

Kinematics

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left[\nabla \boldsymbol{u} + [\nabla \boldsymbol{u}]^T\right]$$
$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$$
$$j = \det(\boldsymbol{F})$$

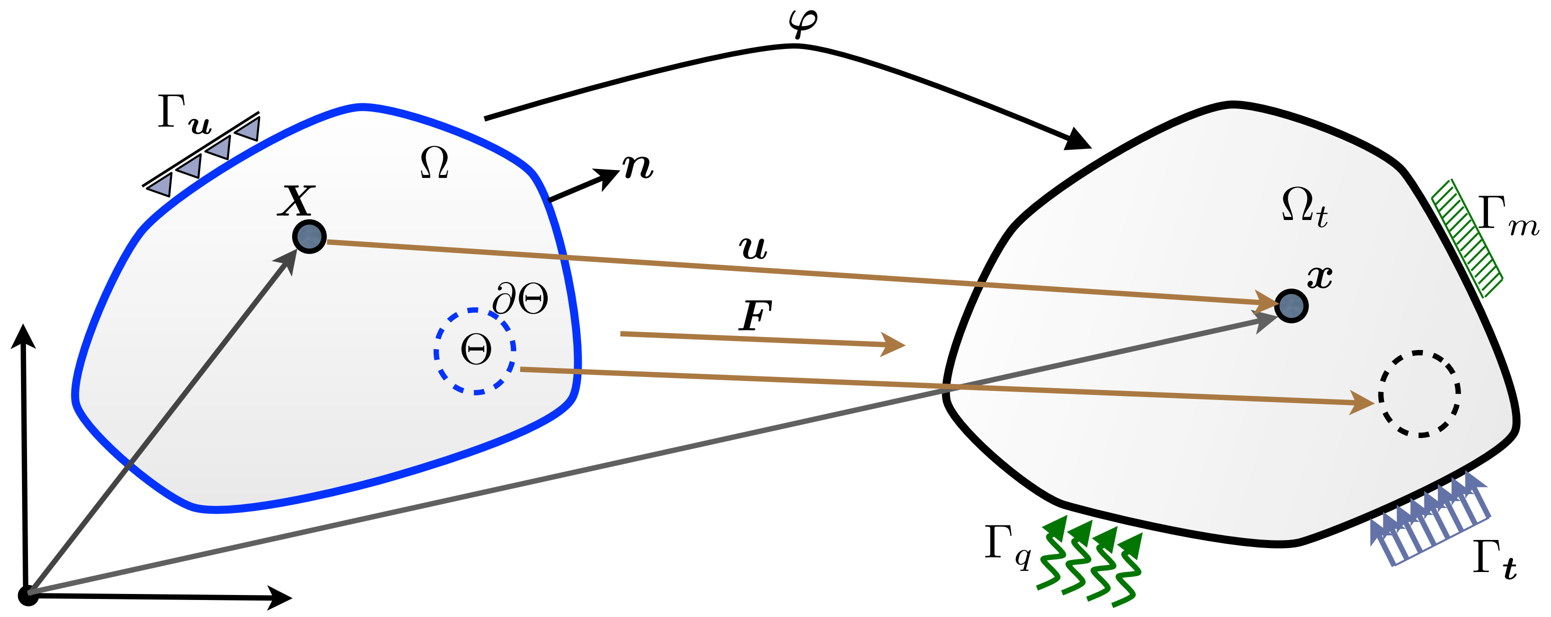
Equilibrium

$$\operatorname{div} \boldsymbol{\sigma} = \boldsymbol{0} \quad \text{in } \Omega$$
$$\boldsymbol{u} = \bar{\boldsymbol{u}} \quad \text{on } \Gamma_{\boldsymbol{u}}$$
$$\boldsymbol{t} := \boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_{\boldsymbol{t}}$$

Conservation of solvent mass

$$\dot{m} = -\operatorname{div} \boldsymbol{h} \quad \text{in } \Omega \times [0, T]$$
$$m = \bar{m} \quad \text{on } \Gamma_m$$
$$q := -\boldsymbol{h} \cdot \boldsymbol{n} = \bar{q} \quad \text{on } \Gamma_q$$

Constitutive relations



Free energy

$$\psi = \psi(\boldsymbol{\varepsilon}, \boldsymbol{\alpha}, m)$$

inelastic (viscous) strain

Stress

$$\boldsymbol{\sigma} = \frac{\partial\psi}{\partial\boldsymbol{\varepsilon}}$$

Solvent flux

ideal mixtures:

$$\boldsymbol{h} = -\boldsymbol{D}(m) \cdot \nabla m$$

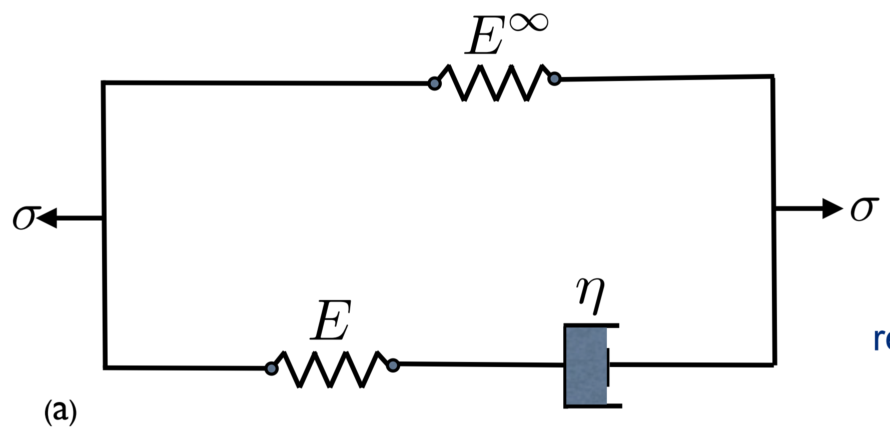
non-ideal mixtures:

$$\boldsymbol{h} = -\boldsymbol{D}(m)m \cdot \nabla\mu(m, j)$$

chemical potential

Constitutive relations: viscoelasticity

- Polymeric solid response is well described as a viscoelastic solid
- Adopt a model due to SIMO & HUGHES (1998)
  - Inelastic strain treated as an internal variable
- Consider the one-dimensional standard linear solid:



inelastic (viscous) strain

$$\sigma = E_{\infty}\varepsilon + \sigma^{v}$$

$$\sigma^{v} = \eta\dot{\alpha} = E\left[\varepsilon - \alpha\right]$$

$$\sigma = \underbrace{E_0}_{E_{\infty}+E}\ \varepsilon - E\alpha$$

viscosity

$$\tau = \frac{\eta}{E}$$

relaxation time

$$\text{Evolution of } \alpha \begin{cases} \dot{\alpha} + \dfrac{\alpha}{\tau} = \dfrac{\varepsilon}{\tau} \\ \lim\limits_{t\to-\infty} \alpha(t) = 0 \end{cases}$$

Viscoelasticity

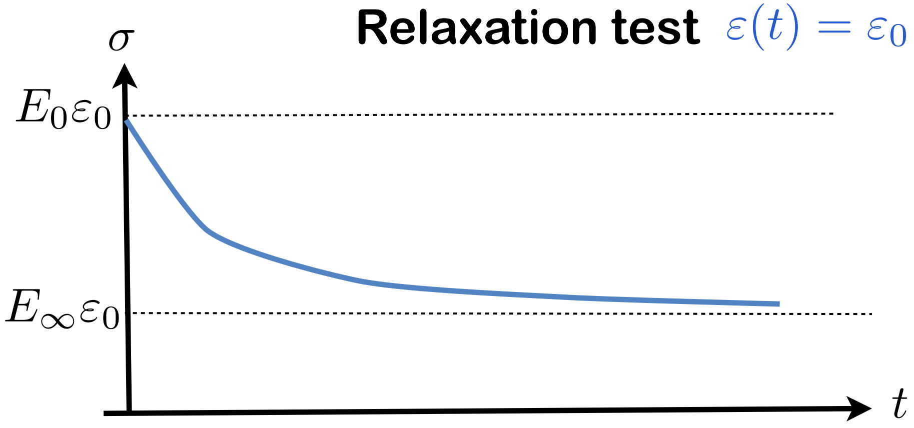
Convolution representation

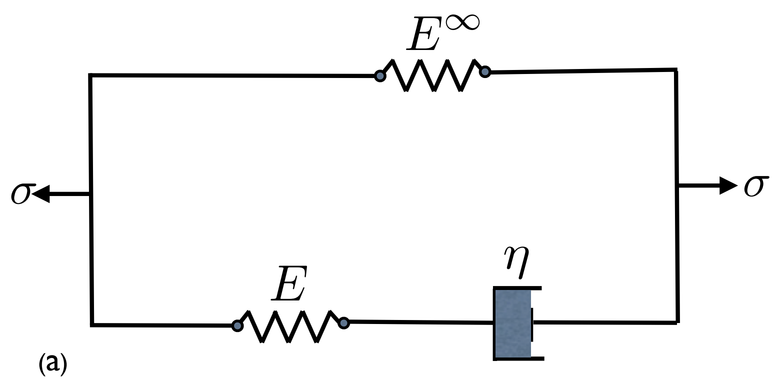
$$\sigma(t) = \int_{-\infty}^{t} G(t-s)\dot{\varepsilon}(s)\ \mathrm{d}s$$

$$G(t) = E_\infty + E\exp\left(\frac{-t}{\tau}\right)$$

relaxation function

Relaxation test $\varepsilon(t) = \varepsilon_0$





Thermodynamics

$$\psi(\varepsilon,\alpha) := \tfrac{1}{2}E_\infty\varepsilon^2 + \tfrac{1}{2}E\left[\varepsilon-\alpha\right]^2$$

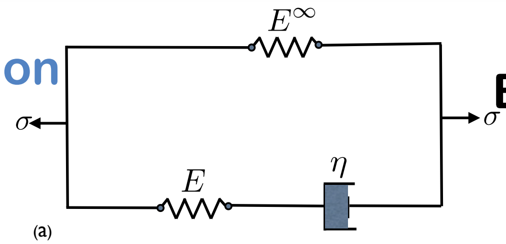
$$\mathcal{D} = \sigma^v\dot{\alpha} = \eta\dot{\alpha}^2 \geq 0$$ dissipation

$$\sigma^v = -\frac{\partial\psi(\varepsilon,\alpha)}{\partial\alpha}$$ viscous stress

$$\sigma = \frac{\partial\psi(\varepsilon,\alpha)}{\partial\varepsilon}$$ stress

Viscoelasticity

Alternative representation



viscous stress

$$q := E\alpha$$

$$\sigma = E_0 \varepsilon - q$$

$$\begin{cases} \gamma := \dfrac{E}{E_0} \\ \gamma_\infty := \dfrac{E_\infty}{E_0} \end{cases}$$

$$W^0(\varepsilon) := \tfrac{1}{2}\varepsilon E_0 \varepsilon$$

initial stored-energy function

elastic stress

$$\sigma = \frac{\partial W^0(\varepsilon)}{\partial \varepsilon} - q$$

viscous stress evolution

$$\dot{q} + \frac{q}{\tau} = \frac{\gamma}{\tau}\frac{\partial W^0(\varepsilon)}{\partial \varepsilon}$$

$$\lim_{t \to -\infty} q(t) = 0$$

Extension to $\mathbb{R}^3$

deviatoric / volumetric split

$$\boldsymbol{\varepsilon} = \boldsymbol{e} + \tfrac{1}{3}\Theta\mathbf{1}$$

$$\boldsymbol{e} := \operatorname{dev}\boldsymbol{\varepsilon} \quad \text{and} \quad \Theta := \operatorname{tr}\boldsymbol{\varepsilon}$$

$$W^0(\boldsymbol{\varepsilon}) = \bar{W}^0(\boldsymbol{e}) + U^0(\Theta)$$

$$\boldsymbol{\sigma}^0 := \frac{\partial W^0(\boldsymbol{\varepsilon})}{\partial \boldsymbol{\varepsilon}} = \operatorname{dev}\left(\frac{\partial \bar{W}^0}{\partial \boldsymbol{e}}\right) + U^{0\prime}\mathbf{1}$$

$$\boldsymbol{\sigma}(t) = \boldsymbol{\sigma}^0(t) - \boldsymbol{q}$$

$$\dot{\boldsymbol{q}} + \frac{\boldsymbol{q}}{\tau} = \frac{\gamma}{\tau}\operatorname{dev}\left(\frac{\partial W^0(\boldsymbol{e})}{\partial \boldsymbol{e}}\right)$$

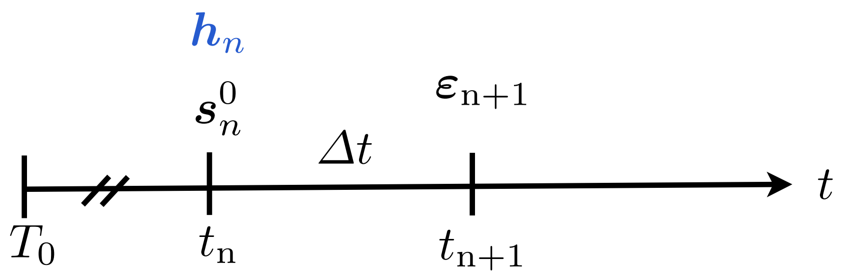
$$\lim_{t \to -\infty} \boldsymbol{q}(t) = 0$$

$$\boldsymbol{\sigma}(t) = U^{0\prime}\mathbf{1} + \int_{-\infty}^{t} g(t-s)\frac{\mathrm{d}}{\mathrm{d}s}\left[\operatorname{dev}\left(\frac{\partial \bar{W}^0(\boldsymbol{e}(s))}{\partial \boldsymbol{e}}\right)\right] \mathrm{d}s$$

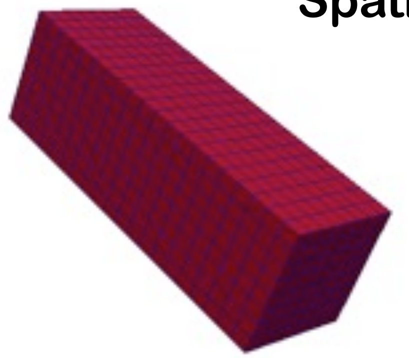
$$g(t) := \gamma_\infty + \gamma_i \exp\left(\frac{-t}{\tau}\right)$$

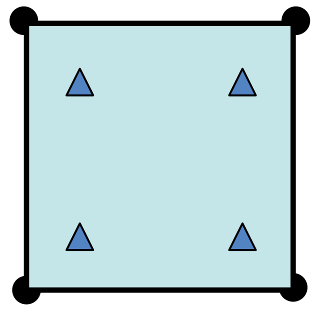
Integration algorithm for viscoelasticity

Temporal discretisation



Spatial discretisation





- Strain driven formulation
- Transform convolution representation for internal variables via two-step recurrence relationship
  - (approach restricted to relaxation functions consisting of linear combinations of functions in time that possess semi-group property)

$$\boldsymbol{e}_{n+1} = \operatorname{dev}(\boldsymbol{\varepsilon}_{n+1})$$

$$\boldsymbol{s}^0_{n+1} = \operatorname{dev}\left(\frac{\partial \bar{W}^0(\boldsymbol{e}_{n+1})}{\partial \boldsymbol{e}}\right)$$

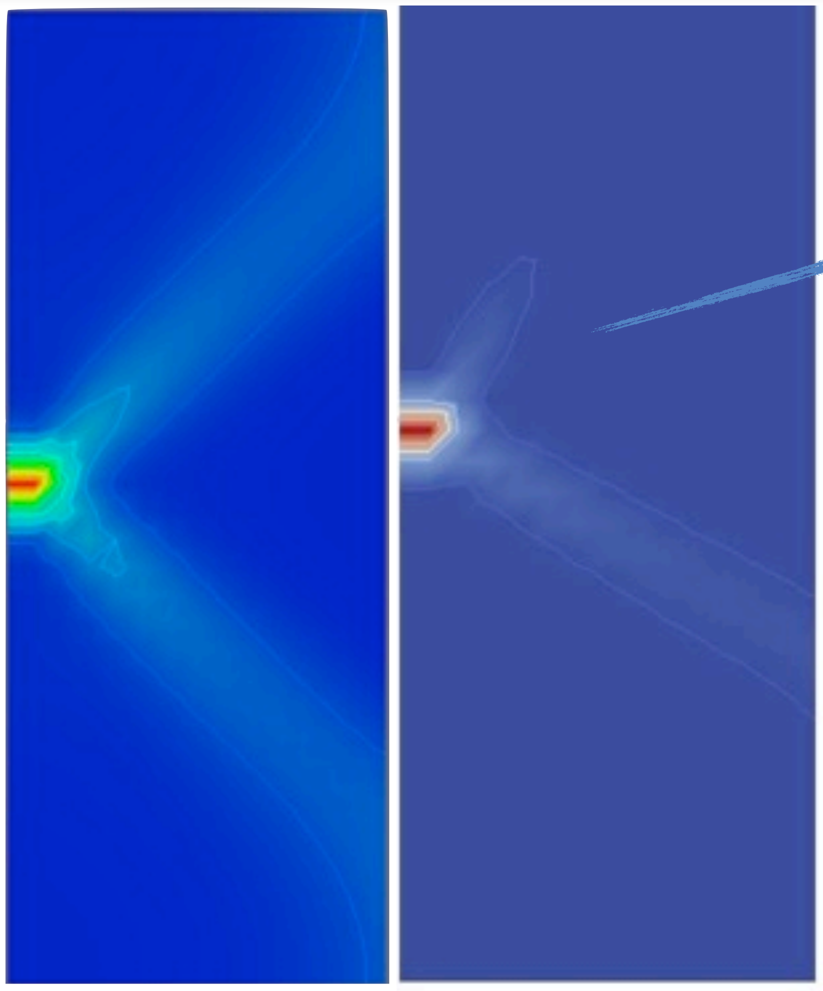
$$\boldsymbol{h}_{n+1} = \exp\left(\frac{-\Delta t_n}{\tau}\right)\boldsymbol{h}_n + \exp\left(\frac{-\Delta t_n}{2\tau}\right)\left[\boldsymbol{s}^0_{n+1} - \boldsymbol{s}^0_n\right]$$

$$\boldsymbol{\sigma}_{n+1} = U^{0'}(\Theta_{n+1})\mathbf{1} + \gamma_\infty \boldsymbol{s}^0_{n+1} + \gamma \boldsymbol{h}_{n+1}$$

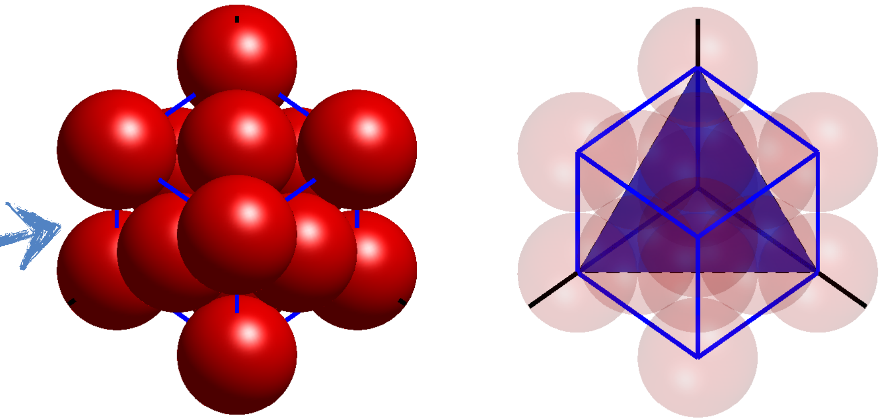
•data at level quadrature point
•no continuity relations
•State not determined from the nodal variables alone

Wednesday 25 August 2010

Internal variable formulations



$v_1 = 0$ $v_1 = 15$



Algorithms for crystal plasticity
McB, Reddy, Richardson

Internal variables:

- Additional state variables
- Evolve subject to an evolution equation that is possibly dependent on the primary nodal variables
- Possess no continuity requirements
- Generally treated directly in strong form at the level of the quadrature point

Another example: Plasticity

Rate independent continuum or crystal plasticity

- Plastic strain (multiplier) generally treated as an internal variable
- More complex as the evolution of the plastic strain is subject to KTT constraints
- Damage, etc.

Models for Case II diffusion

Hyperbolic diffusion

- Classical diffusion relations are parabolic
  - CII concentration propagates as a wave at a fixed velocity: hyperbolic behaviour
- Following CATTANEO (1948) and VERNOTTE (1958) for Fourier's law of heat conduction:

$$\dot{m} = -\operatorname{div} \boldsymbol{h}$$

$$\dot{\boldsymbol{h}} + \beta \boldsymbol{h} = -\beta D \nabla_{\boldsymbol{x}} m$$

assuming constant diffusivity

$$\underbrace{\operatorname{div} \dot{\boldsymbol{h}}}_{\ddot{m}} + \beta \underbrace{\operatorname{div} \boldsymbol{h}}_{\dot{m}} = \beta D \nabla^2 m$$

$$\dot{m} + \frac{1}{\beta} \ddot{m} = D \nabla^2 m \longleftarrow \text{hyperbolic diffusion}$$

- Propagation of heat as thermal wave (second sound) in fluids:

  PESHKOV (1944, 1946) and PELLAM (1948)

  and gases:

  ACKERMANN & OVERTON (1966), JACKSON ET AL (1970), NARAYANAMURTI & DYNES (1972)
- Also see framework of AIFANTIS (1980)
- Case II diffusion see KALOSPIROS ET AL (1991)

and the GENERIC formulation of EL AFIF & GRMELA (2002)

Models for Case II diffusion

Strongly coupled non-Fickian models

- Non-Fickian diffusion coupled to a viscoelastic solid
    - Wu & Peppas (1993), Govindjee & Simo (1993), Vijalapura & Govindjee (2003, 2005)
- Models of Govindjee et al are far more well developed and advanced

Equilibrium and Solvent mass balance

$$\mathrm{Div}\,(\boldsymbol{F}\cdot\boldsymbol{S}) = \boldsymbol{0}$$

$$\dot{M} = \mathrm{Div}\,(\boldsymbol{D}(M, J, \underset{\nwarrow}{R}))\, M\boldsymbol{C}^{-1}\cdot\nabla_{\boldsymbol{X}}\mu$$

concentration reacted sites

$$\left(\boldsymbol{c} := \boldsymbol{F}^T\boldsymbol{F}\right)$$

PKII stress $\boldsymbol{S} = Jp\boldsymbol{C}^{-1} + \boldsymbol{S}_{\mathrm{dev}}$

$$\boldsymbol{S}_{\mathrm{dev}} = \boldsymbol{S}^{\infty}_{\mathrm{dev}} + 2\rho_0 j^{\frac{2}{3}}\,\mathrm{dev}\,\boldsymbol{Q}$$

elastic stress $\boldsymbol{S}^{\infty}_{\mathrm{dev}} = 2\rho_0\dfrac{\partial\psi^E}{\partial\boldsymbol{C}}$

pressure $p = p^{\infty}(j) + p_s(j, M) + q(j, M)$

$$\dot{q} + \frac{q}{\tau(j, M)} = \gamma_1\dot{p}^{\infty}$$

$$\dot{\boldsymbol{Q}} + \frac{\boldsymbol{Q}}{\tau(j, M)} = \gamma_2\overline{\left[\frac{\partial\psi^E}{\partial\boldsymbol{C}}\right]}^{\,\boldsymbol{\cdot}}$$

- Finite deformation setting
- Solid modelled Neo-Hookean viscoelastic (incompressible)
- Highly coupled and non-linear
- Perfect mixing not assumed: extension of Flory-Huggins model to transient regime

Proposed models

Introduce the following features into two reduced models for case II diffusion:

Non-linear coupled diffusion and deformation

- Key features of the GOVINDJEE ET AL model but restricted to small strains
- Prototype for coupled mixed formulations
- Work in progress

1

Non-linear coupled hyperbolic diffusion deformation

- As 1 but adopt a hyperbolic diffusion model
- Treat reflection of concentration waves
- Future work

2

- Spatial adaptivity
  - Solvent propagates through medium as a wave
  - Exploit existing methodologies for hyperbolic problems
  - Need to consider the projection of inelastic internal variables
- Space-time finite elements
  - Extend spatial adaptivity features of deal.II to temporal adaptivity use finite element in space and time
- Optimal solvers and automatic differentiation routines (Sacado in Trilonos)
- Parallel implementation
- Finite strain formulation

Good problem for MeshWorker and NoX in Trilinos

Reduced model 1

Governing DAEs

$$\operatorname{div} \boldsymbol{\sigma}(\boldsymbol{u}, \boldsymbol{\alpha}, m) = \boldsymbol{0} \quad \text{in } \Omega \times [0, T]$$

$$\dot{m} = -\operatorname{div} \boldsymbol{h}(\boldsymbol{u}, m) \quad \text{in } \Omega \times [0, T]$$

Boundary and initial conditions

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \quad \text{on } \Gamma_{\boldsymbol{u}}$$

$$\boldsymbol{t} := \boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_{\boldsymbol{t}}$$

$$m = \bar{m} \quad \text{on } \Gamma_m$$

$$q = -\boldsymbol{h} \cdot \boldsymbol{n} = \bar{h} \quad \text{on } \Gamma_q$$

$$m(\boldsymbol{x}, t = 0) = m_0(\boldsymbol{x})$$

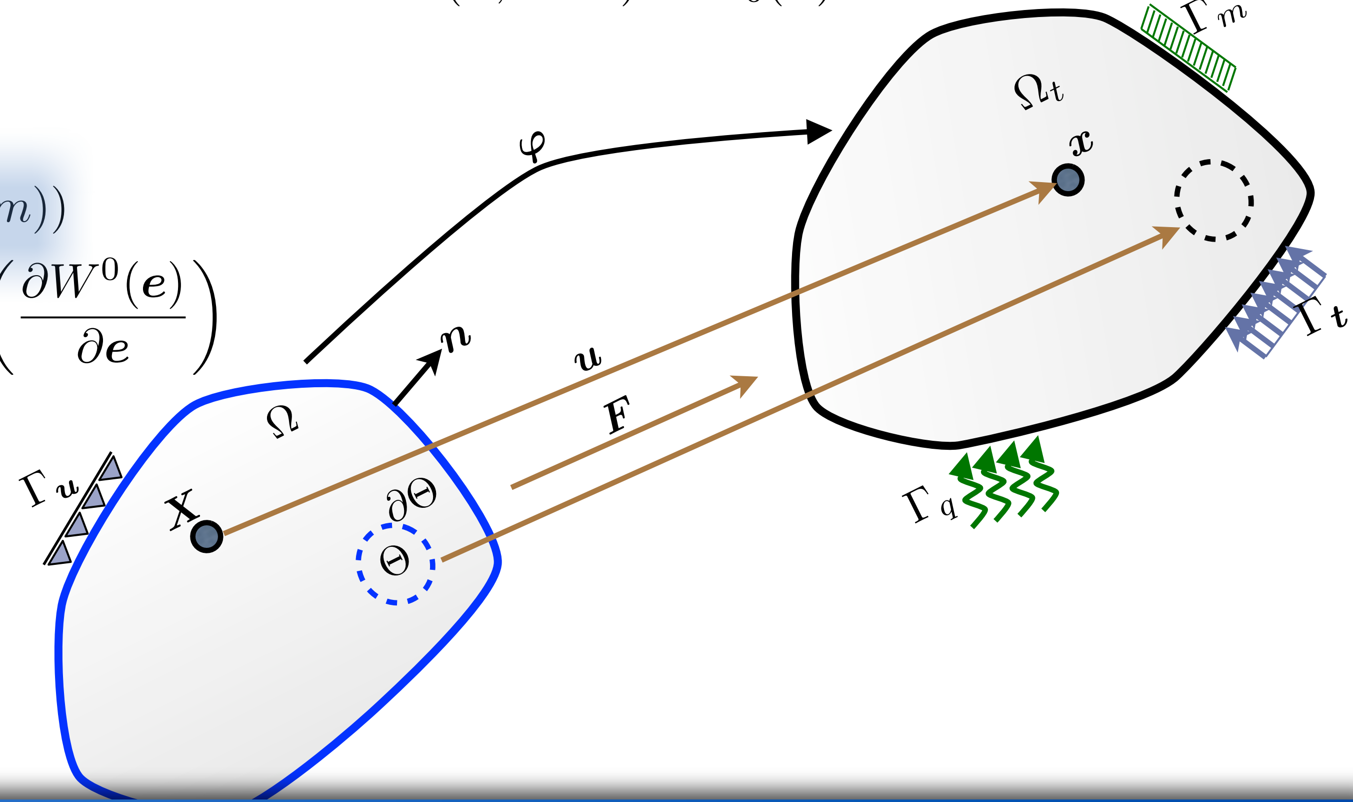
Constitutive relations

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}, \boldsymbol{q}, m)$$

$$\boldsymbol{h} = -D(m)\nabla m$$

$$(\boldsymbol{h} = -D(m)m\nabla\mu(J, m))$$

$$\dot{\boldsymbol{q}} + \frac{\boldsymbol{q}}{\tau(m)} = \frac{\gamma}{\tau(m)} \operatorname{dev}\left(\frac{\partial W^0(\boldsymbol{e})}{\partial \boldsymbol{e}}\right)$$



Reduced model 1

Weak form equilibrium equation

$$(\operatorname{div}\boldsymbol{\sigma}, \boldsymbol{v})_\Omega = 0 \qquad \forall \boldsymbol{v} \in H_0^1(\Omega)^{n_{\text{dim}}}$$
$$(\boldsymbol{\sigma}, \boldsymbol{\varepsilon}(\boldsymbol{v}))_\Omega = (\bar{\boldsymbol{t}}, \boldsymbol{v})_{\Gamma_t}$$

Temporal discretisation

$$[0,T] \approx [0, t^1, \dots, t^{\mathrm{n}}, t^{\mathrm{n}+1}, \dots, T]$$
$$\Delta t = t^{\mathrm{n}+1} - t^{\mathrm{n}}$$

Conservation of solvent mass

$$\frac{m^{\mathrm{n}+1} - m^{\mathrm{n}}}{\Delta t} = -\left[\theta \operatorname{div}\boldsymbol{h}^{\mathrm{n}+1} + [1-\theta]\operatorname{div}\boldsymbol{h}^{\mathrm{n}}\right] \qquad \theta \in [0,1]$$
$$m^{\mathrm{n}+1} + \theta \Delta t \operatorname{div}\boldsymbol{h}^{\mathrm{n}+1} = m^{\mathrm{n}} - [1-\theta]\,\Delta t \operatorname{div}\boldsymbol{h}^{\mathrm{n}}$$

Weak form

$$\left(m^{\mathrm{n}+1}, v^{\mathrm{n}+1}\right)_\Omega + \theta\Delta t\left(\operatorname{div}\boldsymbol{h}^{\mathrm{n}+1}, v^{\mathrm{n}+1}\right)_\Omega = \left(m^{\mathrm{n}}, v^{\mathrm{n}+1}\right)_\Omega - [1-\theta]\,\Delta t\left(\operatorname{div}\boldsymbol{h}^{\mathrm{n}}, v^{\mathrm{n}+1}\right)_\Omega \qquad \forall v \in H_0^1(\Omega)$$
$$\left(m^{\mathrm{n}+1}, v^{\mathrm{n}+1}\right)_\Omega - \theta\Delta t\left(\boldsymbol{h}^{\mathrm{n}+1}, \nabla v^{\mathrm{n}+1}\right)_\Omega = \left(m^{\mathrm{n}}, v^{\mathrm{n}+1}\right)_\Omega + [1-\theta]\,\Delta t\left(\boldsymbol{h}^{\mathrm{n}}, \nabla v^{\mathrm{n}+1}\right)_\Omega + \theta\Delta t\left(q^{\mathrm{n}+1}, v^{\mathrm{n}+1}\right)_{\Gamma_q} + [1-\theta]\,\Delta t\left(q^{\mathrm{n}}, v^{\mathrm{n}+1}\right)_{\Gamma_q}$$

- •Following Step-23 we discretise in time first and then space: Rothe's method
- •Allow for spatial adaptivity

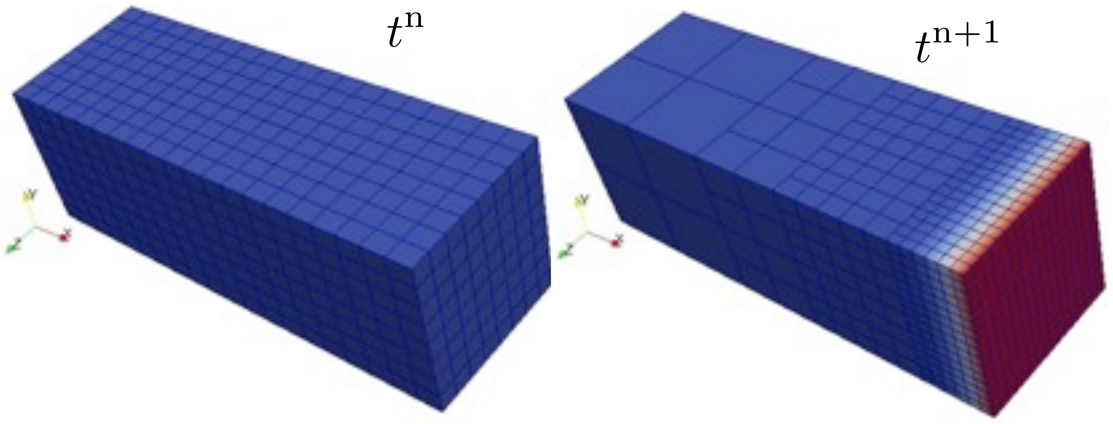
Reduced model 1

Displacement

$$\boldsymbol{u}(\boldsymbol{x}) \approx \boldsymbol{u}_h(\boldsymbol{x}) = \sum_{i=1}^{n_{\mathrm{dof}}} \boldsymbol{\Phi}_i^{\boldsymbol{u}}(\boldsymbol{x}) U_i$$

$$\boldsymbol{v}(\boldsymbol{x}) \approx \boldsymbol{v}_h(\boldsymbol{x}) = \sum_{i=1}^{n_{\mathrm{dof}}} \boldsymbol{\Phi}_i^{\boldsymbol{u}}(\boldsymbol{x}) \bar{U}_i$$

$$\boldsymbol{\varepsilon}(\boldsymbol{v}) \approx \boldsymbol{\varepsilon}(\boldsymbol{v}_h) = \sum_{i=1}^{n_{\mathrm{dof}}} \nabla^{\mathrm{sym}} \left(\boldsymbol{\Phi}_i^{\boldsymbol{u}}(\boldsymbol{x})\right) \bar{U}_i$$



$$\left(m^{\mathrm{n}}, v^{\mathrm{n+1}}\right)_\Omega \approx \sum_{i=1}^{n_{\mathrm{dof}}} M_i^{\mathrm{n}} \sum_{j=1}^{n_{\mathrm{dof}}} \left(\Phi_i^{m,(\mathrm{n})}, \Phi_j^{m,(\mathrm{n+1})}\right)_\Omega \bar{M}_j^{\mathrm{n+1}}$$

Concentration

$$m^{\mathrm{n+1}}(\boldsymbol{x}) \approx m_h^{\mathrm{n+1}}(\boldsymbol{x}) = \sum_{i=1}^{n_{\mathrm{dof}}} \Phi_i^{m,(\mathrm{n+1})}(\boldsymbol{x}) M_i^{\mathrm{n+1}}$$

$$m^{\mathrm{n}}(\boldsymbol{x}) \approx m_h^{\mathrm{n}}(\boldsymbol{x}) = \sum_{i=1}^{n_{\mathrm{dof}}} \Phi_i^{m,(\mathrm{n})}(\boldsymbol{x}) M_i^{\mathrm{n}}$$

$$\nabla v^{\mathrm{n+1}}(\boldsymbol{x}) \approx \nabla m_h^{\mathrm{n+1}}(\boldsymbol{x}) = \sum_{i=1}^{n_{\mathrm{dof}}} \nabla \left(\Phi_i^{m,(\mathrm{n+1})}(\boldsymbol{x})\right) \bar{M}_i^{\mathrm{n+1}}$$

- Shape functions are defined on different meshes.
  1. Perform the integration on the finest common mesh (Step 28)
  2. Project solution at *n* to *n+1* (Steps 31-33)

Reduced model 1

Fully-discrete residual equations

$$
\begin{aligned}
\boldsymbol{R}(\boldsymbol{U},\boldsymbol{M}) &:= [\boldsymbol{R}^{\boldsymbol{u}}\ \boldsymbol{R}^{m}]^T \\
\boldsymbol{R}^{\boldsymbol{u}} &:= (\boldsymbol{\sigma}_h, \boldsymbol{\varepsilon}(\boldsymbol{v}_h))_\Omega - (\bar{\boldsymbol{t}}_h, \boldsymbol{v}_h)_{\Gamma_{\boldsymbol{t}}} \\
\boldsymbol{R}^{m} &:= \left(m_h^{\mathrm{n}+1}, v_h^{\mathrm{n}+1}\right)_\Omega - \left(m_h^{\mathrm{n}}, v_h^{\mathrm{n}+1}\right)_\Omega \\
&\quad - \theta \Delta t \left(\boldsymbol{h}_h^{\mathrm{n}+1}, \nabla v_h^{\mathrm{n}+1}\right)_\Omega - [1-\theta]\,\Delta t \left(\boldsymbol{h}_h^{\mathrm{n}}, \nabla v_h^{\mathrm{n}+1}\right)_\Omega \\
&\quad - \theta \Delta t \left(q_h^{\mathrm{n}+1}, v_h^{\mathrm{n}+1}\right)_{\Gamma_q} - [1-\theta]\,\Delta t \left(q_h^{\mathrm{n}}, v_h^{\mathrm{n}+1}\right)_{\Gamma_q}
\end{aligned}
$$

$$\boldsymbol{R}_{\mathrm{i}+1} = \boldsymbol{0}$$

$$\boldsymbol{R}_{\mathrm{i}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\delta \boldsymbol{U} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{M}}\delta \boldsymbol{M} = \boldsymbol{0}$$

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{R}^{\boldsymbol{u}}}{\partial \boldsymbol{U}} & \dfrac{\partial \boldsymbol{R}^{\boldsymbol{u}}}{\partial \boldsymbol{M}} \\ \dfrac{\partial \boldsymbol{R}^{m}}{\partial \boldsymbol{U}} & \dfrac{\partial \boldsymbol{R}^{m}}{\partial \boldsymbol{M}} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{U} \\ \delta \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{R}^{\boldsymbol{u}}_i \\ -\boldsymbol{R}^{m}_i \end{bmatrix}$$

•Non-symmetric, potentially highly non-linear even for the reduced problem. Full finite deformation problem is horrendously non-linear! (see Govindjee et al.)

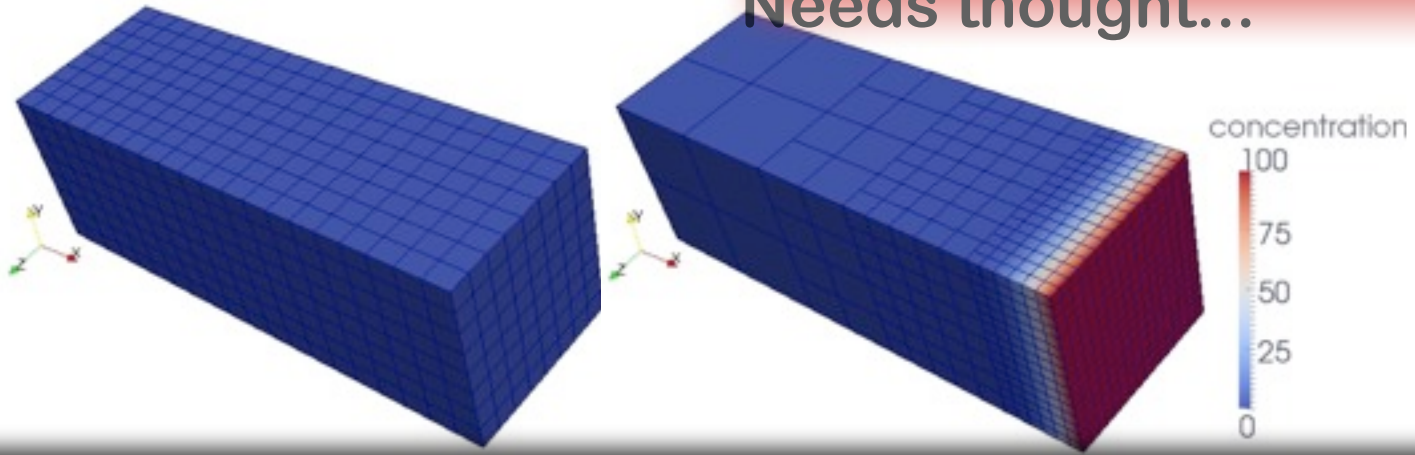
- Currently solve using a monolithic Newton scheme to solve
  - ToDo: Investigate the use of split schemes
- Currently using an approximation to tangent
  - ToDo: Investigate the use of automatic differentiation tools in Sacado (Trilinos)

(Or several other suggestions made so far)

Mesh adaptivity with internal variables

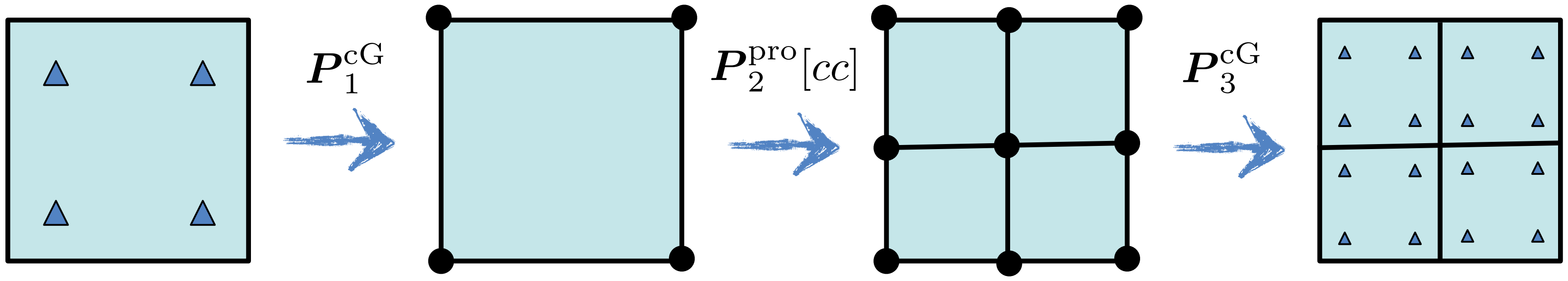
- Spatial adaptivity with nodal unknowns is mature within DEAL.II
  - Hanging nodes with continuity imposed via linear constraints
  - Fully parallelised implementation
- Spatial adaptivity with inelastic internal variables less well developed
  - Projection of quadrature point data between refinement levels
  - Storage of cell related data cumbersome and restricts parallel implementation

Discussed during workshop. Needs thought...



Mesh adaptivity with internal variables

1 Refinement



$\boldsymbol{y}' \in \mathbb{R}^{n_{\mathrm{qp}}}$

$\boldsymbol{y}[cc] \in \mathbb{R}^{n_{\mathrm{qp}}}$

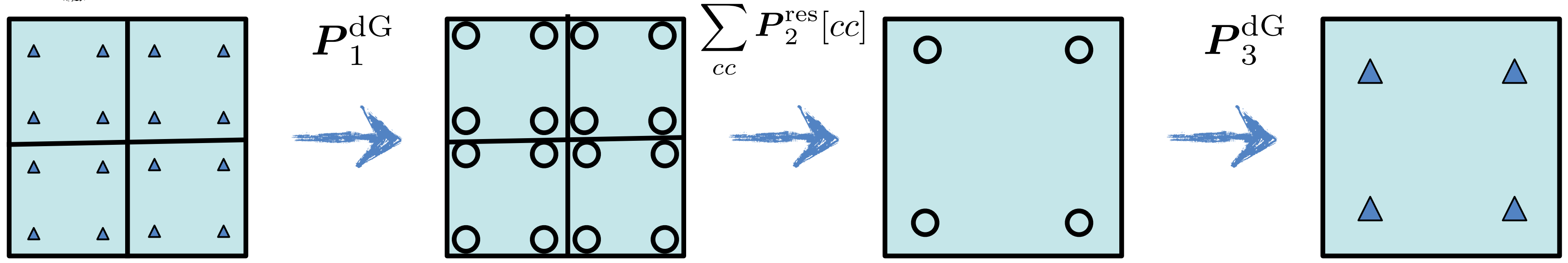
$$\boldsymbol{P}^{\mathrm{ref}}[cc] = \boldsymbol{P}_3^{\mathrm{cG}} * \boldsymbol{P}_2^{\mathrm{pro}}[cc] * \boldsymbol{P}_1^{\mathrm{cG}}$$
$$\boldsymbol{y}[cc] = \boldsymbol{P}^{\mathrm{ref}}[cc] * \boldsymbol{y}'$$

```
FE_Q<deal_II_dimension> fe_projection_cg(u_degree);

FETools::compute_projection_from_quadrature_points_matrix
(fe_projection_cg,...,P_1_cg);
FETools::compute_interpolation_to_quadrature_points_matrix
(fe_projection_cg,..., P_3_cg);
for (unsigned int cc = 0 ; cc < n_children ; cc++) {
     FullMatrix<double> P_2_i_prolongation =
     fe_projection_cg.get_prolongation_matrix(cc);
     P_refine[cc] = P3_P2_pro_P1;
}
```

•All projection matrices are computed once and for all on the reference cell
•Method valid for all components of internal variables

Mesh adaptivity with internal variables

2 Coarsening



$\boldsymbol{y}'[cc] \in \mathbb{R}^{n_{\text{qp}}}$

$$\boldsymbol{P}^{\text{coarse}}[cc] = \boldsymbol{P}_3^{\text{dG}} * \boldsymbol{P}_2^{\text{res}}[cc] * \boldsymbol{P}_1^{\text{dG}}$$

$$\boldsymbol{y} = \sum_{cc} \boldsymbol{P}^{\text{coarse}}[cc] * \boldsymbol{y}'[cc]$$

$\boldsymbol{y} \in \mathbb{R}^{n_{\text{qp}}}$

```
FE_DGQ<deal_II_dimension> fe_projection_dg(u_degree);

FETools::compute_projection_from_quadrature_points_matrix
(fe_projection_dg,...,P_1_dg);
FETools::compute_interpolation_to_quadrature_points_matrix
(fe_projection_dg,..., P_3_dg);
for (unsigned int cc = 0 ; cc < n_children ; cc++) {
     FullMatrix<double> P_2_i_restriction =
     fe_projection_dg.get_restriction_matrix(cc);
     P_coarsen[cc] = P3_P2_res_P1;
}
```

- All projection matrices are computed once and for all on the reference cell
- Note: the summation
- Method valid for all components of internal variables: i.e. works for scalars, vectors and symmetric second order tensors

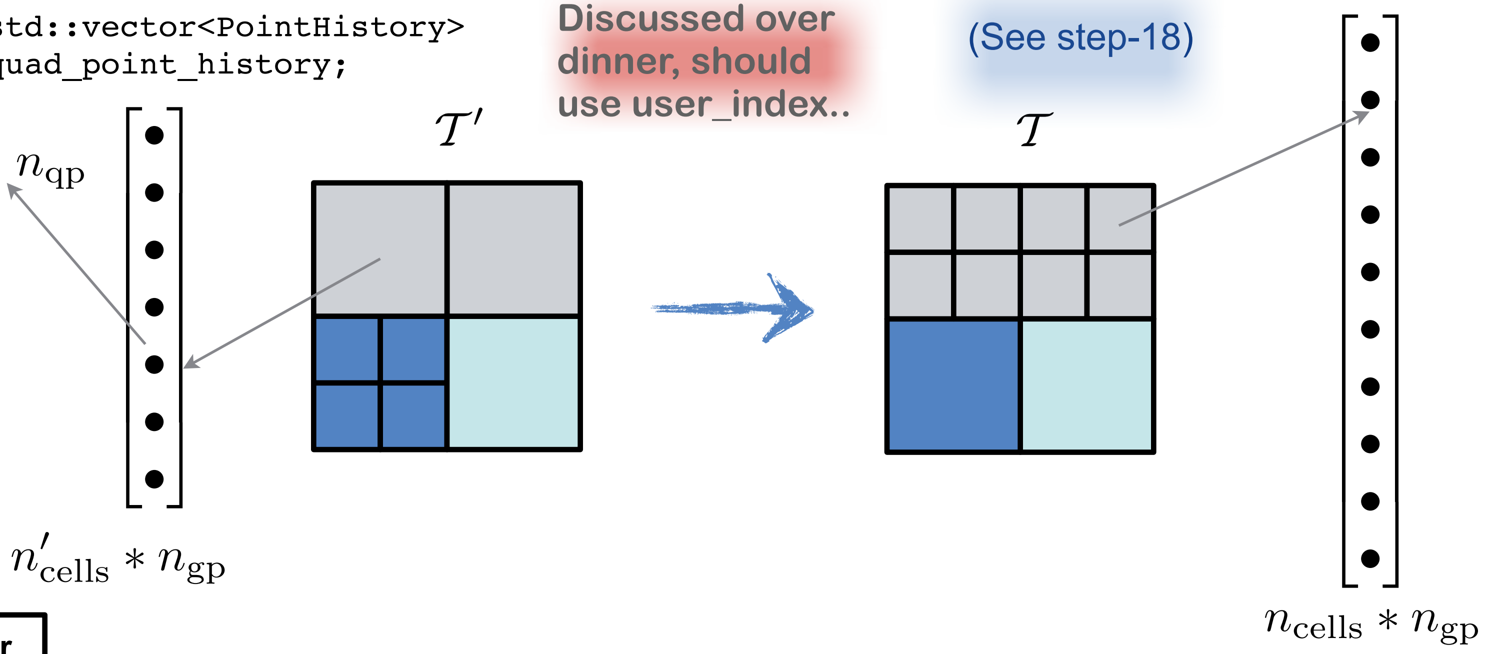
Mesh adaptivity with internal variables

```
std::vector<PointHistory>
quad_point_history;
```

Discussed over dinner, should use user_index..

(See step-18)

PointHistory
InternalVariables
+PointHistory (internal_variables) +getInternalVariables



n_{qp}
$\mathcal{T}'$
$\mathcal{T}$
$n'_{\text{cells}} * n_{\text{gp}}$
$n_{\text{cells}} * n_{\text{gp}}$

RefinementListener

↑

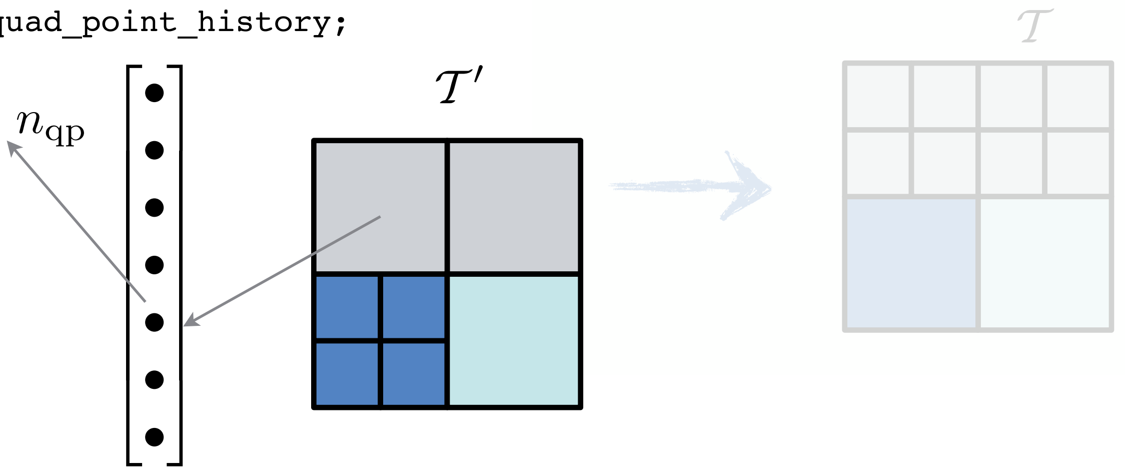
RefinementManager
quad_point_history quad_point_history_tmp refine / coarsen projections
+pre_refinement_notification (Triangulation&) +post_refinement_notification (Triangulation&)

```
GridRefinement::refine_and_coarsen_fixed_number (triangulation,...)
refinement_manager.limit_refinement_levels(triangulation);

BlockVector<double>  x_X = X;

SolutionTransfer<deal_II_dimension, BlockVector<double> >
        solution_transfer(dof_handler);
triangulation.prepare_coarsening_and_refinement();
solution_transfer.prepare_for_coarsening_and_refinement(x_X);
triangulation.execute_coarsening_and_refinement ();
setup_system();
solution_transfer.interpolate (x_X, X);
update_state(true);
```

Mesh adaptivity with internal variables

```
std::vector<PointHistory>
quad_point_history;
```

PointHistory
InternalVariables
+PointHistory (internal_variables)   +getInternalVariables



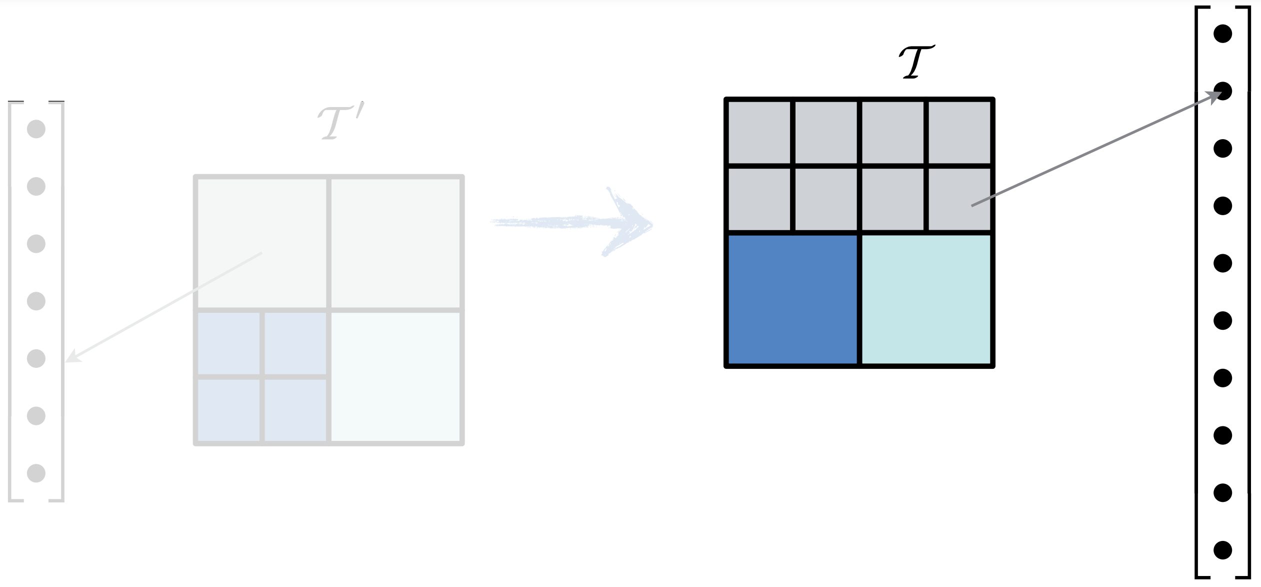
RefinementListener

RefinementManager
quad_point_history   quad_point_history_tmp   refine / coarsen projections
+pre_refinement_notification (Triangulation&)   +post_refinement_notification (Triangulation&)

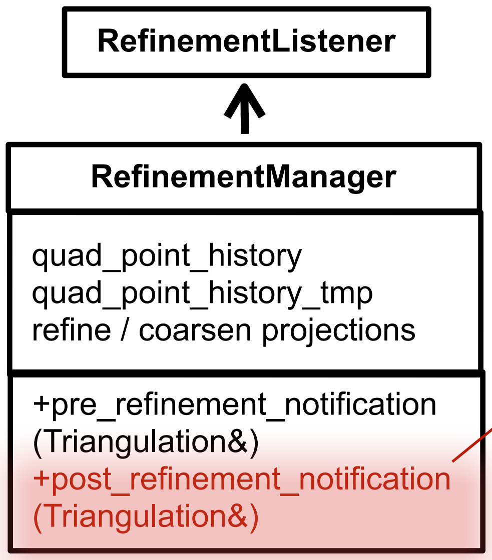
- Copy `quad_point_history` to `quad_point_history_tmp`
- `quad_point_history.clear()`
- Redirect the `cell->user_pointer()` to `quad_point_history_tmp`
- Record the cells to be refined, coarsened and to remain

Wednesday 25 August 2010

Mesh adaptivity with internal variables



Discussed over dinner, should use user_index..



quad_point_history
quad_point_history_tmp
refine / coarsen projections

+pre_refinement_notification (Triangulation&)
+post_refinement_notification (Triangulation&)

- Cells to remain:
  - Copy cell data from `quad_point_history_tmp` to `quad_point_history`
  - Redirect `cell->user_pointer`
- Cells to coarsen:
  - project data from children to parents
  - Add parent data to `quad_point_history`
  - Redirect `parent->user_pointer`
- Cells to refine:
  - project data from parents to children
  - Add children's data to `quad_point_history`
  - Redirect `children->user_pointer`

Mesh adaptivity with internal variables

Issues

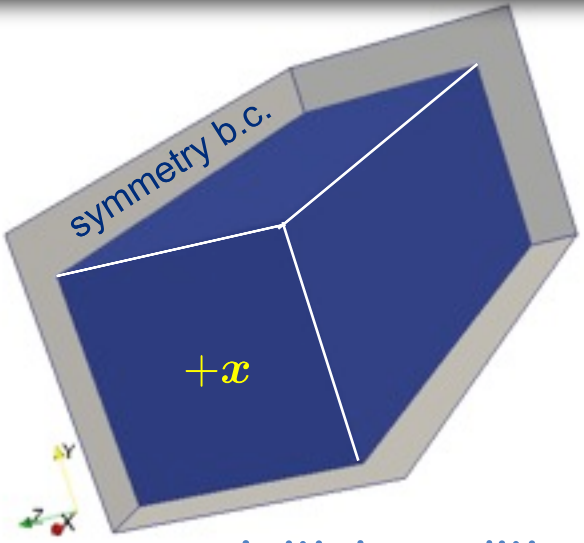
- Children do not know their parents :(
  - Complicates the coarsening routine as one can't simply ask those children flagged for coarsening for their parent
- The need to store the cell data in a std::vector causes headaches
  - The data should be associated more tightly with a cell and management abstracted from the user
  - Abstract class CellData that the user can overload
  - Member data of this class include the projections to perform coarsening and refinement
  - Simple to handle memory as the CellData is coupled to cell?
  - Parallelisation friendly?
- The cell user_pointer is used by deal.II code itself. Ideally a user should be "me" or "you" and not deal.II

Already discussed during workshop

Example problem

Constitutive laws

$$W^0(\boldsymbol{\varepsilon}) = \bar{W}^0(\boldsymbol{e}) + U^0(\Theta)$$
$$= \mu \boldsymbol{e} : \boldsymbol{e} + \tfrac{1}{2}\left[\lambda + \tfrac{2}{3}\mu\right]\Theta^2$$
$$\nu = \tfrac{1}{3} \quad \text{and} \quad \mu = 1000$$
$$D(c) = \begin{cases} \frac{1}{10} & \text{if } m > 50 \\ \frac{1}{1000} & \text{otherwise} \end{cases}$$
$$\tau(m) = \begin{cases} \frac{1}{10} & \text{if } m > 50 \\ 10000 & \text{otherwise} \end{cases}$$



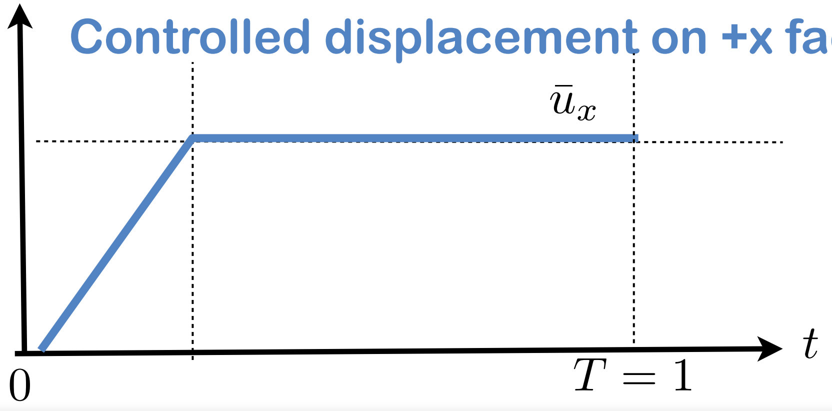
- 400 equal time-steps
  - ToDo temporal adaptivity
- Refine and coarsen a set fraction 0.1 at every time step
  - Need better refinement measure
- Kelly error indicator on concentration KELLY ET AL (1983)
- $Q_1 - Q_1$ elements
- SparseDirectUMFPACK!

Initial conditions and concentration shock boundary condition

$$m(\boldsymbol{x}, t=0) = 0$$
$$m(+\boldsymbol{x}, t=0) = 100$$

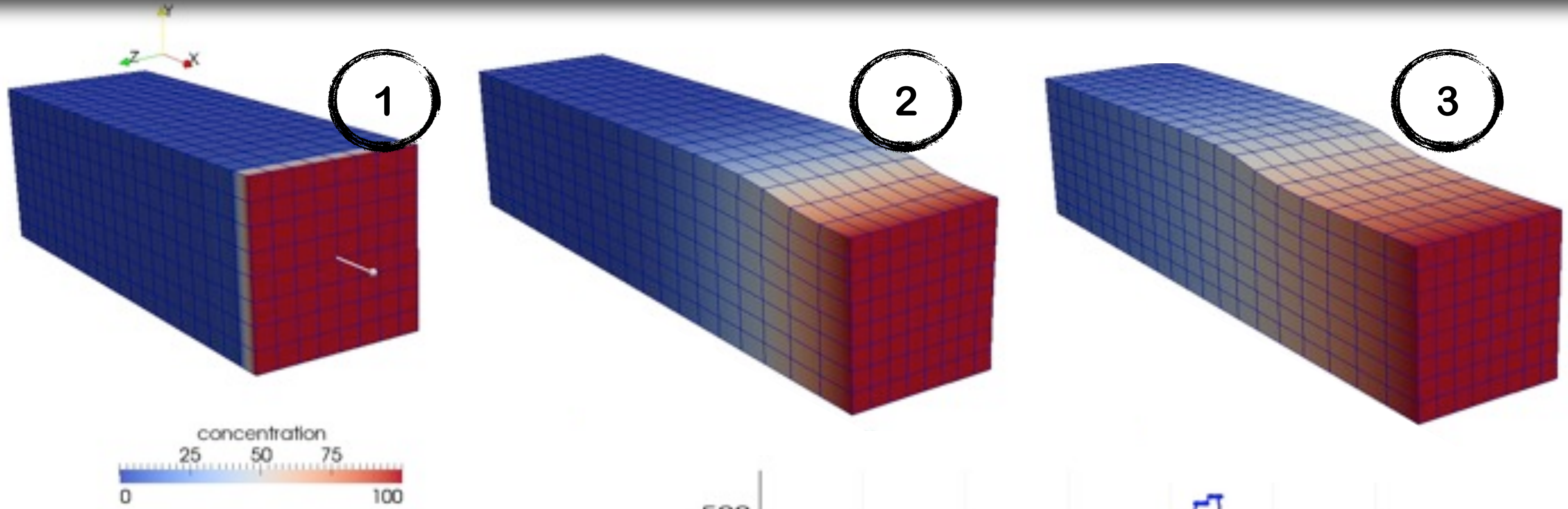
- Coupling is still only one-way: need to account for influence of the solvent on the swelling of the polymer
- Diffusivity and relaxation times are not functions of the deformation
- Polymeric solid still compressible

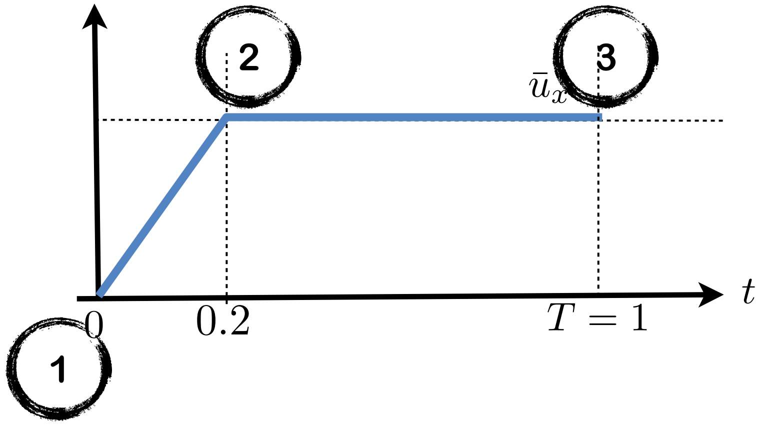
Controlled displacement on +x face

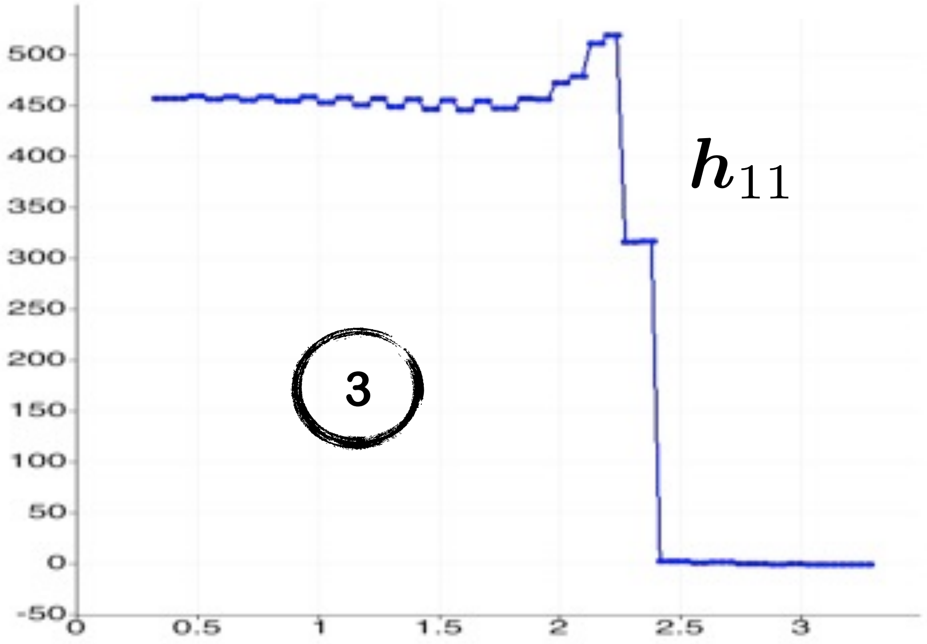


Wednesday 25 August 2010

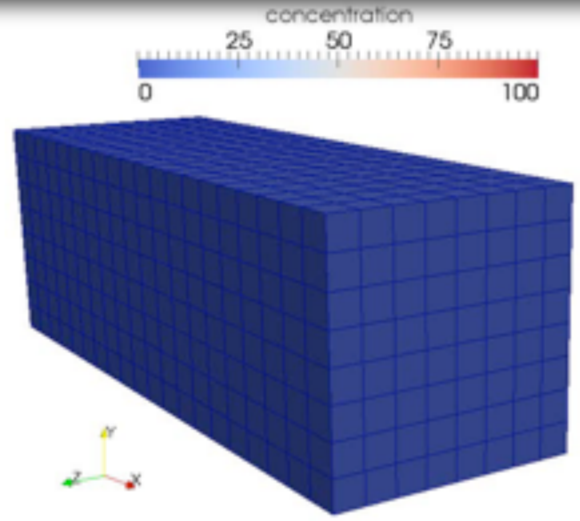
Example problem

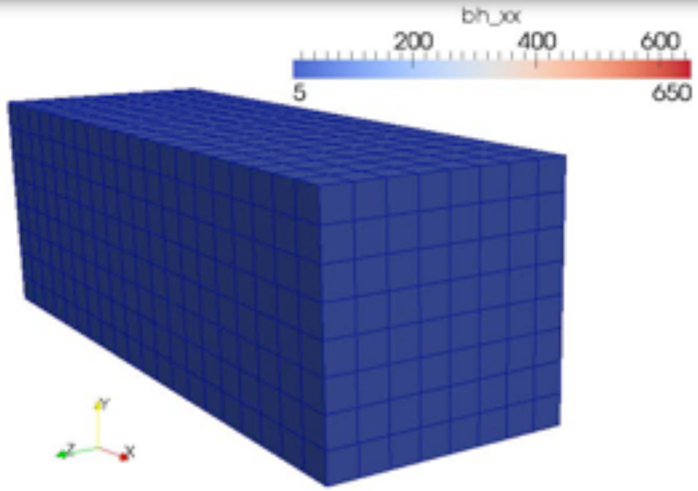


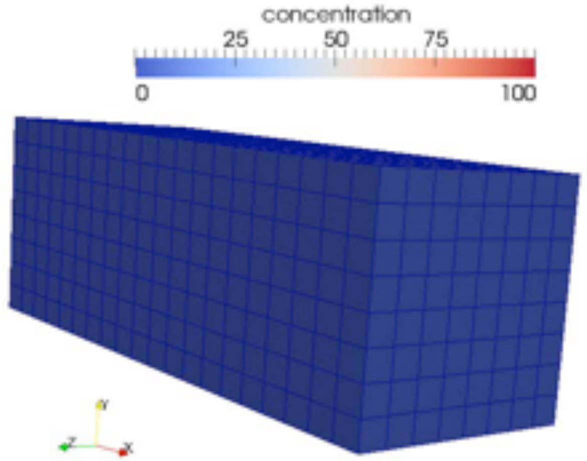


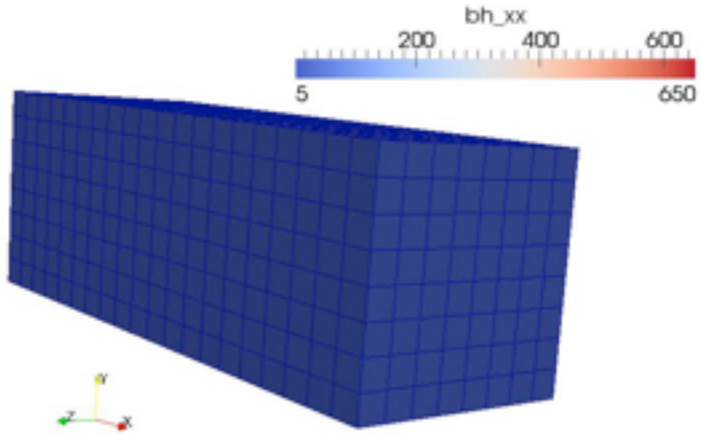


Example problem









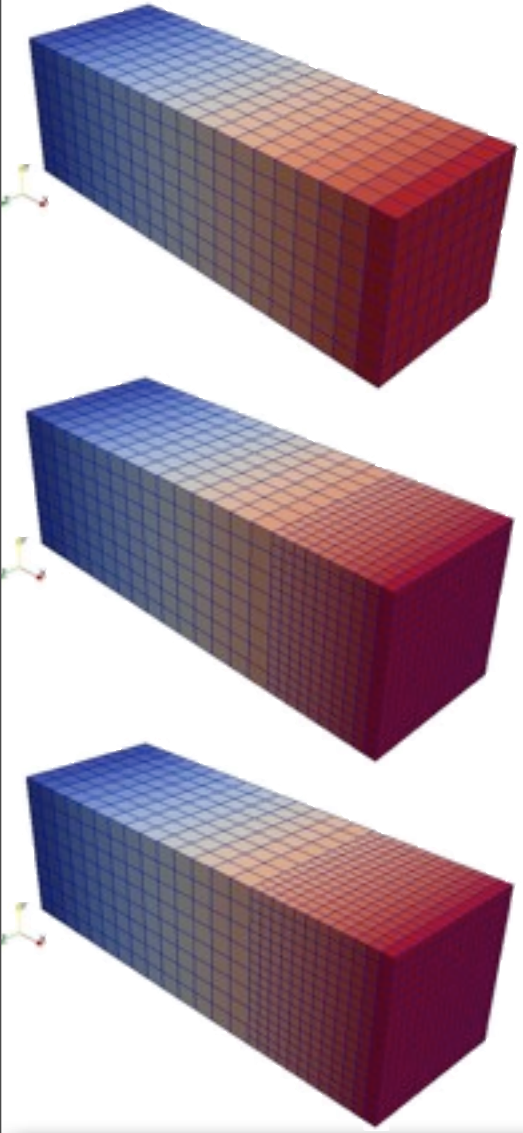
Refinement only...

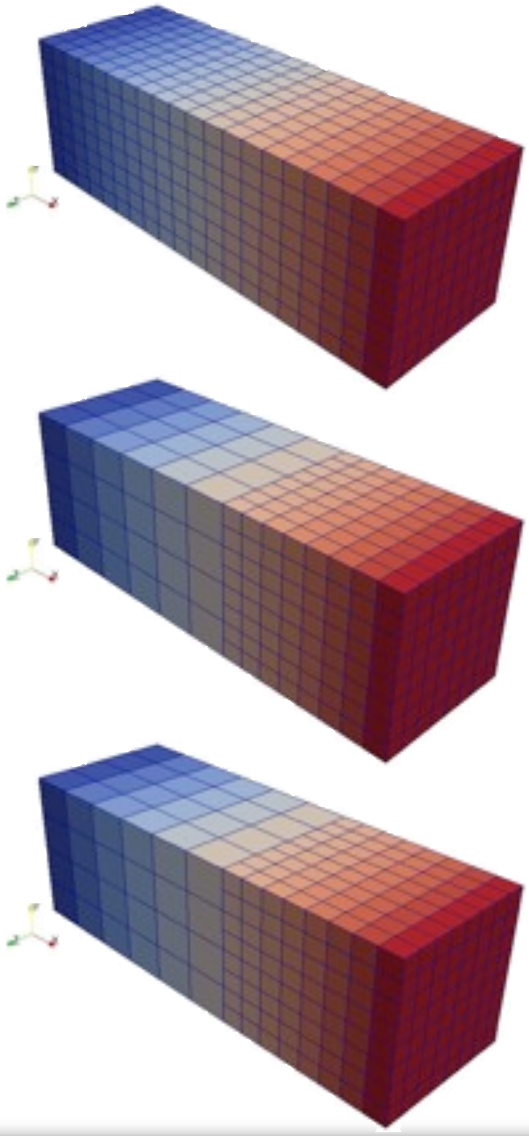
Example problem: the value of test cases. Add the x-coordinate as an internal variable...

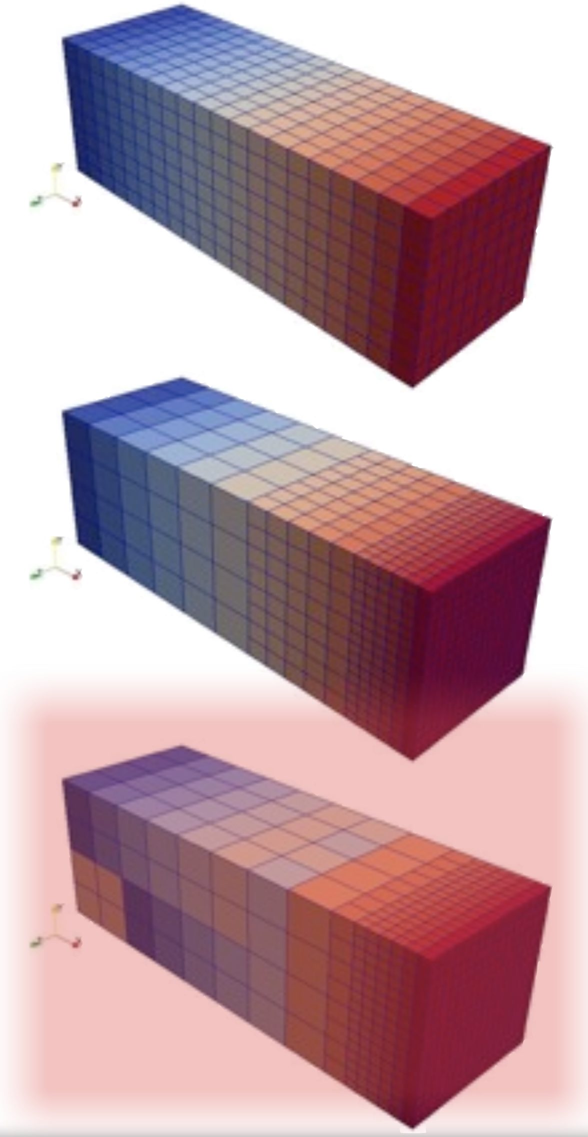
Refine

Coarsen

Refine and coarsen







Wednesday 25 August 2010

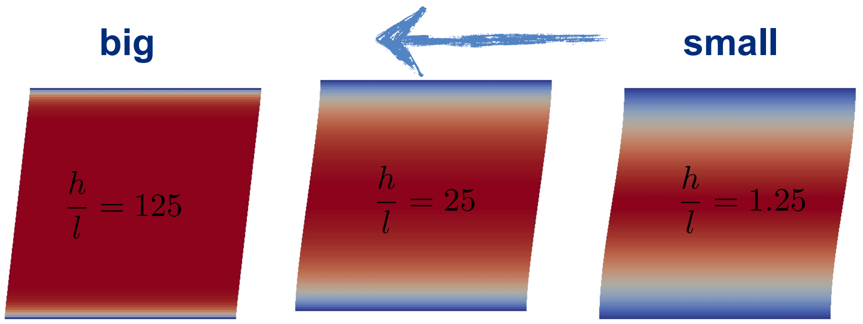
Conclusions

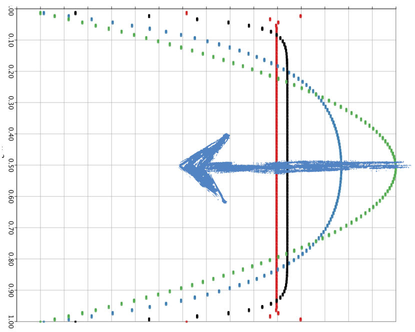
- Spatial adaptivity with internal variables is viable within current structure of deal.II
- Internal variable formulations are widely used in solid mechanics but few (no) solid mechanics codes offer the flexibility and capability of deal.II
- Extend the user base of deal.II if one could facilitate such formulations
  - Strategy to give internal variable data a similar status to nodal variables
  - refinement and projection of internal variables
  - parallel implementation

Discussed in deal.II future ideas

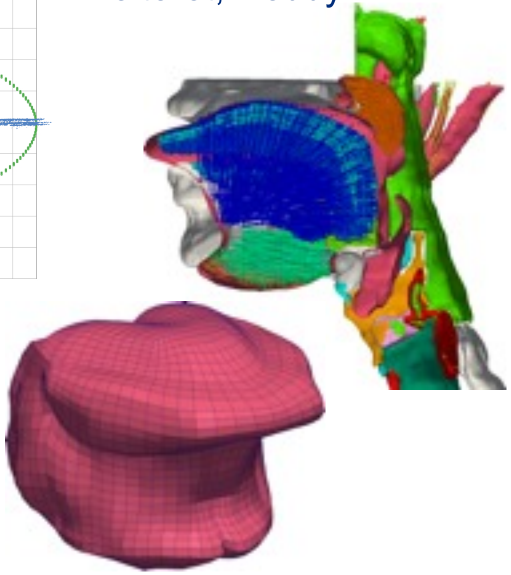
Work in progress: comments and suggestions greatly appreciated

A word of thanks





Sleep Apnoea
Pelteret, Reddy

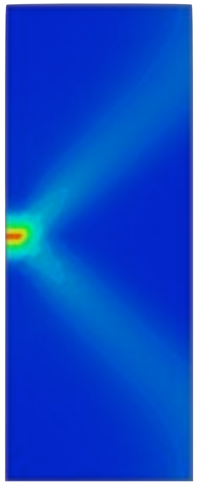


Non-local crystal plasticity
McB, Reddy, Richardson, Gurtin

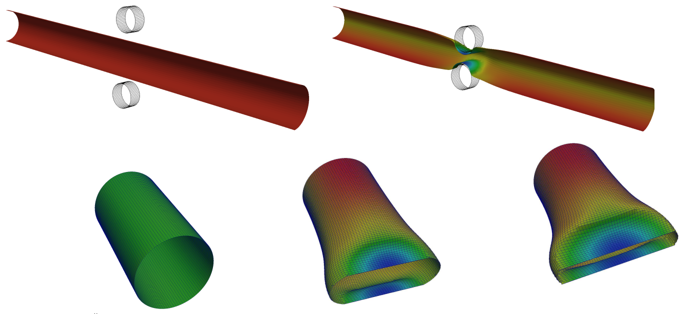
Deal.II at the University of Cape Town,

- Geographically distant and (very) small local support community
- Adopted deal.II as in-house code in 2008
- Small group of ~20 students in DEM, FEM, CFD, particle methods
- Completed or in-progress using deal.II:
  - 4 MSc and 4 PhD
- User group that meets regularly
- **Support and code greatly appreciated! Made projects possible and is a fantastic learning tool**

Algorithms for crystal plasticity
McB, Reddy, Richardson



Shell formulations with applications in biomechanics
Bartle, Reddy, McB



Wednesday 25 August 2010

Thanks. Comments, suggestions and questions please

The financial support of the German Science Foundation (Deutsche Forschungsgemeinschaft, DFG), grant STE 544/39-1, and the National Research Foundation of South Africa is gratefully acknowledged.

The support of all those involved in the deal.II project is most appreciated! thanks.

Thanks to Michael Rapson and Jean-Paul Pelteret