ACFem – Adaptive Convenient Finite Elements with Dune::Fem

C.-J. Heine

Dune User Meeting – Heidelberg – September 2015

Institute for Applied Analysis
and Numerical Simulation

NMH

mathematik
ians
fakultät8

University of Stuttgart
Germany

Contents

Motivation

$$\int_\Omega \nabla U \cdot \nabla V = \int_\Omega F \cdot V \text{ for all } V \in \mathbb{V}$$

- (not-so-) elliptic 2nd order PDEs occur as „building blocks“ in a variety of problems
- implementing „the“ example (Poisson’s equation) is straight forward
- implementing more complicated examples is often a time-consuming task
- many FEM-toolboxes provide interfaces for the definition of basic ingredients
- not so many provide the possibility to form linear combinations of basic models/equations

Concrete Example

- homogenized model for concrete carbonation (cf. Peter & Böhm (2009)) in $\Omega_T = \Omega \times (0,T)$

$$
\begin{aligned}
\partial_t \boldsymbol{j}^{\mathbf{a}} &= -(|Z^{\mathrm{w}}|\, C^{\mathrm{m}} C^{\mathrm{H}} R)\, \boldsymbol{u}\, \boldsymbol{v}^{\mathbf{w}}, && \text{in } \Omega_T,\\
|Z^{\mathrm{w}}|\, \partial_t \boldsymbol{v}^{\mathbf{w}} &= -(|Z^{\mathrm{w}}| C^{\mathrm{H}} m^{\mathrm{v}} R)\, \boldsymbol{u}\, \boldsymbol{v}^{\mathbf{w}}, && \text{in } \Omega_T,\\
\partial_t \left((\boldsymbol{j}^{\mathbf{a}}\, |Z^{\mathrm{a}}| + |Z^{\mathrm{w}}|)\boldsymbol{u}\right) - \nabla \cdot (\boldsymbol{j}^{\mathbf{a}} D^{\mathrm{a}} P^{\mathrm{a}} \nabla \boldsymbol{u}) &= -(|Z^{\mathrm{w}}|\, m^{\mathrm{u}} R)\, \boldsymbol{u}\, \boldsymbol{v}^{\mathbf{w}}, && \text{in } \Omega_T,\\
-j^{\mathrm{a}} D^{\mathrm{a}} P^{\mathrm{a}} \nabla \boldsymbol{u} \cdot \nu &= C^{\mathrm{ext}}(\boldsymbol{u} - u^{\mathrm{ext}}), && \text{on } \partial\Omega,\\
\boldsymbol{u} = u_0,\ \boldsymbol{v}^{\mathbf{w}} &= v_0^{\mathrm{w}},\ \boldsymbol{j}^{\mathbf{a}} = 1, && \text{for } t = 0.
\end{aligned}
$$

- $u = u(x,t)$ combined concentration of CO_2 in pore air and water
- $v^{\mathrm{w}} = v^{\mathrm{w}}(x,t)$ concentration of $\mathrm{Ca(OH)}_2$ in pore water
- $j^{\mathrm{a}} = j^{\mathrm{a}}(x,t)$ relative pore-volume reduction
- Robin b.c. for exchange with outer CO_2 concentration C^{ext}

. . . a few pages later . . .

- PDE-sub-problem

Compute $U_n \in \mathbb{V}(\mathcal{G})$ *by solving the linear problem*

$$\frac{1}{\tau_n}\langle (J_n(t_n)\,|Z^{\mathrm{a}}| + |Z^{\mathrm{w}}|)\,(U_n - U_{n-1}),\,\Phi\rangle_\Omega + \langle J_n(t_n) D^{\mathrm{a}} P^{\mathrm{a}} \nabla U_n,\,\nabla\Phi\rangle_\Omega$$
$$+ \langle (|Z^{\mathrm{w}}|\, m^{\mathrm{u}} R\, V_n(t_n) + |Z^{\mathrm{a}}|\,\partial_t J_n(t_n))\, U_n,\,\Phi\rangle_\Omega + C^{\mathrm{ext}}\langle U_n - u^{\mathrm{ext}},\,\Phi\rangle_{\partial\Omega} = 0$$

for all $\Phi \in \mathbb{V}(\mathcal{G})$.

. . . a few pages later . . .

- mass-part of previous equation:

$$\frac{1}{\tau_n}\langle (J_n(t_n)\,|Z^{\mathrm{a}}| + |Z^{\mathrm{w}}|)\,(U_n - U_{n-1}),\,\Phi\rangle_\Omega \ldots$$

- C++-snippet

```
39 auto simpleMass = massModel(discreteSpace);
40
41 auto massModel = 1./tau * (Z_a * J + Z_w) * simpleMass;
```

Contents

General Elliptic 2^{nd}-Order Equation

Currently implemented general model in `Dune::ACFem`

- „strong“ form:

$$\begin{aligned} -\nabla\cdot(A(x,\dots)\,\nabla u) + \nabla\cdot(b(x,\dots)\,u) + c(x,\dots)\,u &= f(x) && \text{in } \Omega,\\ u &= g_D && \text{on } \Gamma_D,\\ (A(x,\dots)\nabla u)\cdot\nu + \alpha(x,\dots)\,u &= g_N && \text{on } \Gamma_R,\\ (A(x,\dots)\nabla u)\cdot\nu &= g_N && \text{on } \Gamma_N, \end{aligned}$$

- $f \in L^2(\Omega,\mathbb{R}^m)$ is the usual „right hand side“
- $g_N : \Gamma_N \to \mathbb{R}^m$ data for Neumann resp. Robin b.c.
- potentially non-linear, ie. $A = A(x,u,\nabla u)$, $b = b(x,u,\nabla u)$, $c = c(x,u,\nabla u)$
- $u : \Omega \to \mathbb{R}^m$ potentially vector-valued

General Elliptic 2^{nd}-Order Equation

Currently implemented general model in `Dune::ACFem`

- weak form, determine $u \in H^1(\Omega)$ such that

$$\int_\Omega (A\,\nabla u)\cdot\nabla\phi\,dx + \int_\Omega (\nabla\cdot(b\,u) + c\,u)\,\phi\,dx - \int_\Omega f(x)\,\phi\,dx + \int_{\Gamma_R} \alpha\,u\,\phi\,do - \int_{\Gamma_N} g_N\,\phi\,do = 0,$$
$$\langle \Pi, u\rangle = \langle \Pi, g_D\rangle.$$

- usual test-space $\phi \in \{v \in H^1(\Omega) |\ v \equiv 0 \text{ on } \Gamma_D\}$
- Π are suitable test-functionals to enforce the Dirichlet-data

`Dune::Fem` context:

- the multiplication by the test-functions is provided by `Dune::Fem`
- the rest has to be provided by the application

Example for the “Flux”-Term

- „flux“ contribution

$$\int_\Omega (A(x, U, \nabla U)\, \nabla U) : \nabla V =: \sum_{E \in \mathcal{T}} \int_E \sigma_E(x, U, \nabla U) : \nabla V$$

- the application has to provide σ_E and its first variation $\delta\sigma_E$
- prototype for σ_E

```
1 template<class Entity, class Point>
2 void flux(const Entity& entity,
3           const Point &x,
4           const RangeType &value,
5           const JacobianRangeType &jacobian,
6           JacobianRangeType &flux) const
```

- prototype for $\delta\sigma_E$ accepts additional arguments for the point of linearization

The Model-Tuple

We consider vector-valued solutions $U : \mathbb{R}^d \to \mathbb{R}^m$.

- Model constituents:

flux	$\sigma_E : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^{m\times d}$
source	$\mu_E : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^m$
Robin-flux	$\rho_I : \mathbb{R}^d \times R^m \to \mathbb{R}^m$
Robin-indicator	$\chi_R : \partial\Omega \to \{0, 1\}$
Neumann-data	$g_N : \mathbb{R}^d \to \mathbb{R}^m$
Neumann-indicator	$\chi_N : \partial\Omega \to \{0, 1\}$
Dirichlet-data	$g_D : \mathbb{R}^d \to \mathbb{R}^m$
Dirichlet-weight	$w_D : \mathbb{R}^d \to \mathbb{R}^m$
Dirichlet-indicator	$\chi_D : \partial\Omega \to \{0, 1\}$
bulk-forces	$f : \mathbb{R}^d \to \mathbb{R}^m$

- an application has to provide a suitable „model-tuple“

$$\mathcal{M} := (\sigma_E,\, \mu_E,\, \rho_I,\, g_N,\, g_D,\, w_D,\, f, \chi_R,\, \chi_N,\, \chi_D)$$

Weak-formulation with Model-Tuple

- let a model-tuple be given

$$\mathcal{M} := (\sigma_E,\, \mu_E,\, \rho_I,\, g_N,\, g_D,\, w_D,\, f, \chi_R,\, \chi_N,\, \chi_D)$$

- the corresponding formal weak formulation reads

$$\sum_{E\in\mathcal{T}} \Big(\int_E \big(\sigma_E(x,U,\nabla U) : \nabla V + \mu_E(x,U,\nabla U)\cdot V - f\cdot V\big) + \int_{I\in E\cap\partial\Omega} \big(\big(\chi_R(x)\,\rho_I(x,U) - \chi_N(x)\,g_N(x)\big)\cdot V\big)\Big) = 0,$$

$$\langle \Pi, \chi_D\, w_D\, U\rangle = \langle \Pi, \chi_D\, g_D\rangle.$$

- normally $w_D \equiv 1$

Contents

Realization of Model-Tuples in `Dune::Fem`

- each application builds on top of a default *zero-model-tuple*
- only non-zero contributions need to be implemented
- although this is not overly complicated it is a cumbersome task

⇝ `Dune::ACFem` allows forming algebraic expression from existing models

⇝ the generated „model-expressions“ are „model-tuples“ just as „hand-coded“ ones

⇝ this allows forming of complicated models from a set of convenient pre-defined skeleton-models

Adding and Subtracting Model Tuples

- let $\mathcal{M}^\alpha$, $\alpha = 1, 2$ be two model tuples

$$\mathcal{M}^\alpha = (\sigma_E^\alpha,\, \mu_E^\alpha,\, \rho_I^\alpha,\, g_N^\alpha,\, g_D^\alpha,\, w_D^\alpha,\, f^\alpha, \chi_R^\alpha,\, \chi_N^\alpha,\, \chi_D^\alpha)$$

- all integral bulk-contributions can simply be added
- for boundary data:

$$g_B^\pm = \chi_B^1\, g_B^1 \pm \chi_B^2\, g_B^2, \quad B \in \{N, R, D\}.$$

- for addition as well as subtraction the indicator functions are always *added*:

$$\chi_D^\pm = (\chi_D^1 \oplus \chi_D^2) := \chi_D^1 + \chi_D^2 - \chi_D^1\, \chi_D^2,$$
$$\chi_B^\pm = (\chi_B^1 \oplus \chi_B^2)\,(1 - \chi_D^\pm), \quad B \in \{N, R\}.$$

Linear Combinations with Functions

- let
$$\mathcal{M} = (\sigma_E, \mu_E, \rho_I, g_N, g_D, w_D, f, \chi_R, \chi_N, \chi_D)$$
and $s \in L^\infty(E)$ some function
- scaling of model-tuples is defined by
$$s\,\mathcal{M} = (s\,\sigma_E, s\,\mu_E, s\,\rho_I, s\,g_N, s\,g_D, s\,w_D, \chi_R, \chi_N, \chi_D)$$
- scaling leaves the solution set invariant
- the indicator functions remain - of course - unscaled

- algebraically, model tuples form sort of an L^∞-module

Code-Example

```
1  auto Rbc0 = robinZeroModel(discreteSpace);
2  auto _Delta_U = laplacianModel(discreteSpace);
3  auto U = massModel(discreteSpace);
4
5  auto bulkModel = _Delta_U + 4.0*(C*C)*sqr(X)*U;
6  auto F = 2.0 * C * dimDomain * exactSolution;
7
8  auto pdeModel = bulkModel - F + C*(Rbc0 - gR));
9
10 typedef EllipticFemScheme<...> SchemeType;
11 SchemeType scheme(solution, pdeModel, exactSolution);
12
13 return adaptiveAlgorithm(scheme);
14 }
```

Code-Example

```
1  auto Rbc0 = robinZeroModel(discreteSpace);
2  auto _Delta_U = laplacianModel(discreteSpace);
3  auto U = massModel(discreteSpace);
4
5  auto bulkModel = _Delta_U + 4.0*(C*C)*sqr(X)*U;
6  auto F = righHandSide(bulkModel, exactSolution);
7
8  auto pdeModel = bulkModel - F + C*(Rbc0 - gR));
9
10 typedef EllipticFemScheme<...> SchemeType;
11 SchemeType scheme(solution, pdeModel, exactSolution);
12
13 return adaptiveAlgorithm(scheme);
14 }
```

Incomplete list of available „atomic“ Models

- $\int_\Omega U \cdot V$
- $\int_\Omega \nabla U : \nabla V$
- $\int_\Omega |U|^{p-2} U \cdot V$
- $\int_\Omega |\nabla U|^{p-2} \nabla U : \nabla V$
- $\int_\Omega (\nabla U + (\nabla U)^T) : \nabla V$
- $\int_\Omega [\nabla U]\, U \cdot V$
- $-\int_\Omega (U \otimes U) : \nabla V + \int_{\partial\Omega} (U \cdot \nu)\, (U \cdot V)$
- $\langle \Pi, U \rangle = \langle \Pi, g_D \rangle$
- $\int_\Omega f \cdot V$ (bulk forces)

Further models are easy to implement, as one can focus on a single model-component.

Dune::ACFem

- `Dune::ACFem` implements expression templates for model-tuples and grid-functions
- own „hand coded“ model can easily combined with predefined models
- in addition `Dune::ACFem` implements standard adaptive algorithms for conforming finite element methods

- integrates nicely on top of `Dune::Fem`
- parallel, adaptive, 2d, 3d, conforming finite elements

. . . planned

- DG support
- sort out possible auto differentiation support for „auto-linearization“

References I

Peter, M. A. & Böhm, M.: Multi-scale modelling of chemical degradation mechanisms in porous media with evolving microstructure. *SIAM Multisc. Mod. Sim.* **7** (2009), 1643–1668.