The Distributed and Unified Numerics Environment (DUNE)

P. Bastian, M. Blatt, C. Engwer, O. Ippisch

Universität Heidelberg
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Im Neuenheimer Feld 368, D-69120 Heidelberg
email: Peter.Bastian@iwr.uni-heidelberg.de

Durham, July 10, 2010

Outline

Contents

Introduction

Software for the numerical solution of PDEs with grid based methods.

Goals:

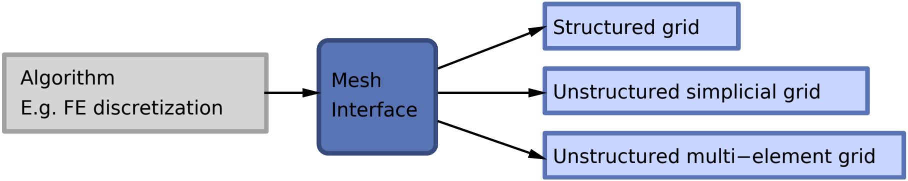
- Flexibility: Meshes, discretizations, adaptivity, solvers.
- Efficiency: Pay only for functionality you need.
- Parallelization.
- Reuse of existing code.
- Enable team work through standardized interfaces.

DUNE

- Developed since 2003 by groups at
  - Free University of Berlin: O. Sander and R. Kornhuber.
  - Freiburg University: A. Dedner, R. Klöfkorn, M. Nolte and D. Kröner.
  - Münster University: Mario Ohlberger.
  - Heidelberg University: C. Engwer, M. Blatt, S. Marnach and P. Bastian.
- Available under GNU LGPL license with linking exception.
- Platform for "Open Reservoir Simulator" (U Stuttgart, U Bergen, SINTEF, StatOil, . . .)
- DUNE courses given every spring (at least).

Programming With Concepts

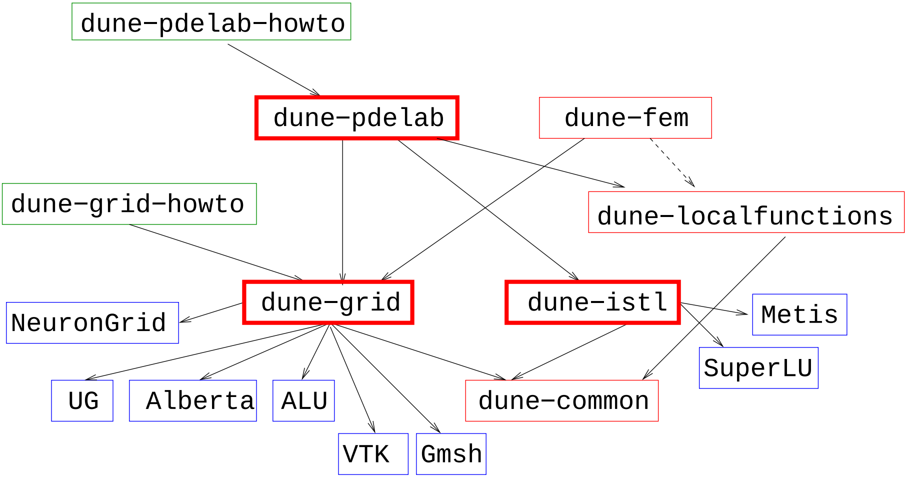
- Separation of data structures and algorithms



- Realization with generic programming (templates) in C++.
- Static polymorphism:
  - Inlining of "small" methods.
  - Allows global optimizations.
  - Interface code is removed at compile-time.
- Template Meta Programs: compile-time algorithms.
- Standard Template Library (STL) is a prominent example.

DUNE Module Architecture

Major DUNE modules are:



Contents

DUNE Grid Interface[1] Features

- Provide abstract interface to grids with:
  - Arbitrary dimension embedded in a world dimension,
  - multiple element type,
  - conforming or nonconforming,
  - hierarchical, local refinement,
  - arbitrary refinement rules (conforming or nonconforming),
  - parallel data distribution, dynamic load balancing.
- User data (e.g. DOF) is external to the grid.
- Reuse existing implementations (ALU, UG, Alberta) + special implementations (Yasp, NeuronGrid).
- More grids on the way: CornerPoint (Sintef), Peano (TU Mü.).

[1] B., P., M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger O. Sander: *A generic grid interface for parallel and adaptive scientific computing. Part I: Implementation and tests in DUNE*. Computing, 82(2-3):121–138, 2008.

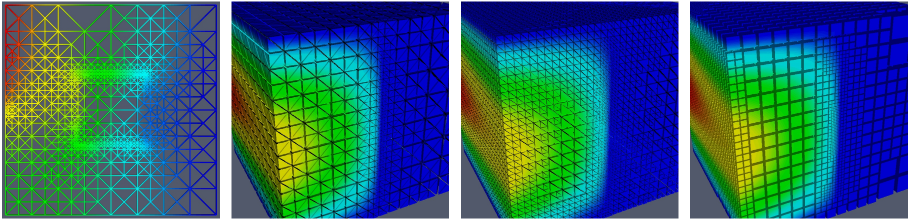
Generic Grid Traversal

```
template<typename GV>
void traversal (const GV& gv)
{
  // Get the iterator type
  typedef typename GV::template Codim<0>::Iterator ElementIterator;

  // iterate through all entities of codim 0
  int count = 0;
  for (ElementIterator it = gv.template begin<0>();
       it!=gv.template end<0>(); ++it)
    {
      Dune::GeometryType gt = it->type();
      std::cout << "visiting␣" << gt
                << "␣with␣first␣vertex␣at␣"
                << it->geometry().corner(0)
                << std::endl;
      count++;
    }

  std::cout << count << "␣element(s)" << std::endl;
}
```

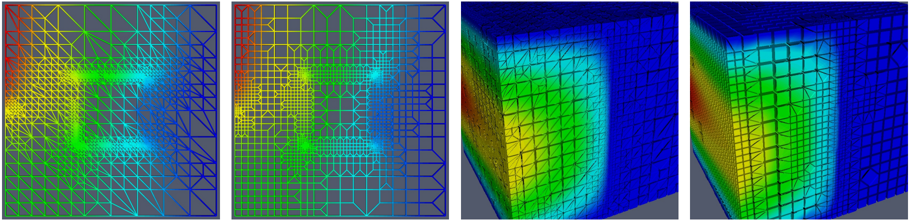
Runs on any mesh in any dimension.

Adaptive Finite Element Example

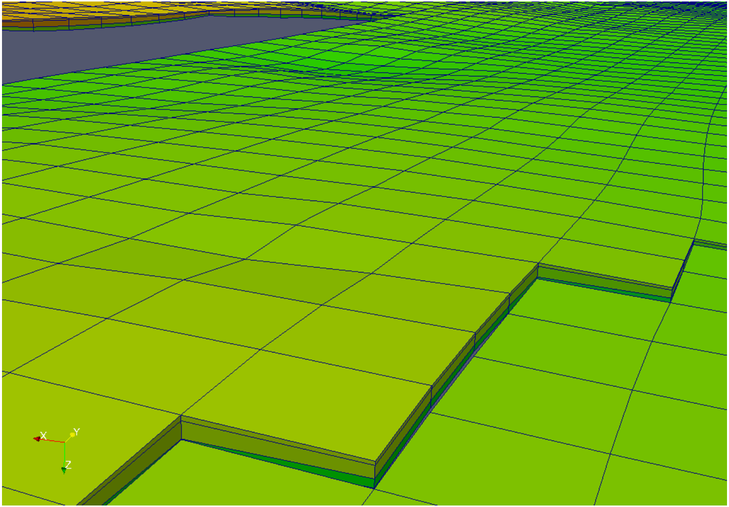
Alberta 2d, 3d, ALU3dGrid, simplices, cubes



UG 2d, simplices, cubes, 3d, simplices, cubes

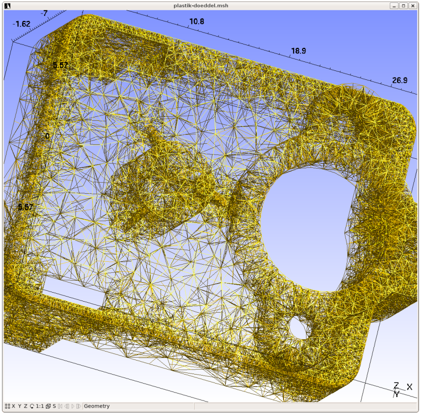


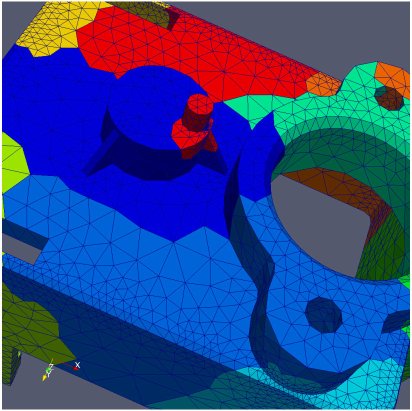
Nonstandard Grids



Corner point grid used in oil reservoir simulation (SINTEF, Oslo).

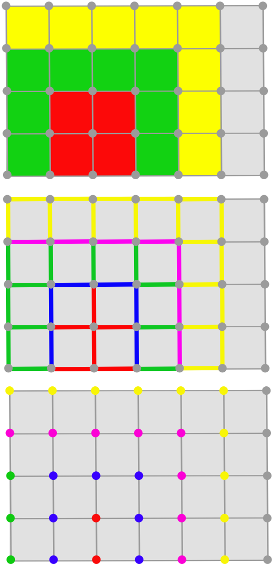
Complete Open Source Workflow





Salome+OpenCascade (CAD), Gmsh (mesh generation), (Par)Metis (load balancing), ParaView+VTK (Visualization)

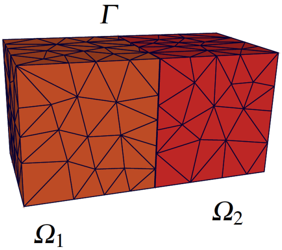
Parallel Data Decomposition

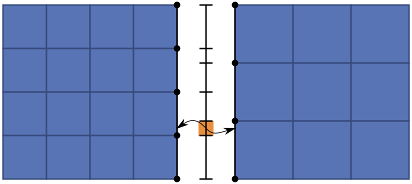


- Grid is mapped to $\mathcal{P} = \{0, \ldots, P-1\}$.
- $E = \bigcup_{p \in \mathcal{P}} E|_p$ possibly overlapping.
- $\pi_p : E|_p \to$ "partition type".
- For codimension 0 there are three partition types:
  - *interior*: Nonoverlapping decomposition.
  - *overlap*: Arbitrary size.
  - *ghost*: Rest.
- For codimension > 0 there are two additional types:
  - *border*: Boundary of interior.
  - *front*: Boundary of interior+overlap.
- Allows implementation of overlapping and nonoverlapping methods.

Meta Grids

- Grid Glue (O. Sander, C. Engwer, G. Buse): Intersects two arbitrary 2d/3d grids

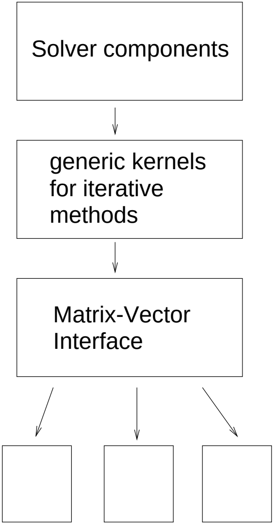




- Works also for two distributed grids.
- Subgrid: Select a subset of elements as a grid.
- MultiDomainGrid (S. Müthing): Partition a given grid into a number of overlapping subgrids.
- GeoGrid (M. Nolte, A. Dedner): Apply a global coordinate transformation.

Contents

Iterative Solver Template Library[2]



- Matrix-Vector Interface: Support recursively block structured matrices.
- Various implementations possible for dense, banded, sparse (similar to MTL).
- **Generic** kernels: E.g. Triangular solves, Gauß-Seidel step, ILU decomposition.
- Solver components: Based on operator concept, Krylov methods, (A)MG preconditioners.
- Flexible parallel infrastructure put on top of sequential components.

[2] M. Blatt, P. B. *The Iterative Solver Template Library.* Volume 4699, Lecture Notes in Scientific Computing, 666-675. Springer, 2007.

AMG Weak scaling, 3D Elliptic Problem

- BlueGene at Jülich Supercomputing Center
- Aggregation-based AMG preconditioner
- 10^{-8} reduction
- $P \cdot 64^3$ degrees of freedom (1024^3 finest mesh), Q_1 FE
- Clipped random permeability field ($\lambda = 1/64, \sigma^2 = 8$)

P	TBuild	IT	TIt	TTotal
1	47.1	14	2.28	78.9
8	56.8	20	2.53	107.0
64	89.8	26	2.73	161.0
512	89.6	35	2.79	187.2
4096	120.2	37	2.90	227.4

Results by Markus Blatt

Strong Scaling on Multi-core Machines

- 4×4 AMD Opteron 8380, 2.5 GHz, 4×0.5MB L2, 6MB L3.
- P_2/P_1 DG discretization of Stokes Problem
- 34×34 blocks, 3760128 degrees of freedom
- BiCGStab + Inexact overlapping Schwarz prec., single time step.

P		1	2	4	8	12	16
Ass	Time	3213	1749	906	481	330.3	280.4
	Speedup	1.0	1.8	3.6	6.7	9.7	11.5
Slv	Time/It	54.4	28.0	14.5	7.7	5.5	4.2
	Speedup	1.0	1.9	3.8	7.0	10.0	13.1
Opt.	Speedup	1.00	1.9	3.7	7.0	9.8	12.6

Results by Christian Engwer

Contents

DUNE PDELab Features

- Rapid prototyping: Substantially reduce time to implement discretizations and solvers for systems of PDEs based on DUNE.
- Simple things should be simple — suitable for teaching.
- Discrete function spaces:
  - Conforming and non-conforming,
  - *hp*-refinement,
  - general approach to constraints,
  - simple construction of product spaces for systems.
- Operators based on weighted residual formulation:
  - Linear and nonlinear,
  - stationary and transient,
  - FE and FV schemes requiring at most face-neighbors.
- Exchangeable linear algebra backend.
- User only involved with "local" view on (reference) element.

Weighted Residual Formulation (Stationary Case)

A large class of problems can be written in the abstract form

$$\text{Find } u_h \in w_h + \tilde{U}_h : \qquad r_h(u_h, v) = 0 \qquad \forall v \in \tilde{V}_h,$$

where:

- $\tilde{U}_h \subseteq U_h$, $\tilde{V}_h \subseteq V_h$ are finite-dimensional function spaces and corresponding subspaces.
- Affine shift: $w_h + \tilde{U}_h = \{u \,:\, u = w_h + \tilde{u}_h, \tilde{u}_h \in \tilde{U}_h\}$.
- $r_h : U_h \times V_h \to \mathbb{K}$ is the *residual form*.
  - r_h may be *nonlinear* in its first argument.
  - r_h *is always linear* in its second argument.
  - r_h may depend on the grid in non-conforming methods.
- We assume that this problem has a unique solution.

Algebraic Problem

Inserting a basis representation yields a nonlinear algebraic problem:

$$\begin{aligned} & u_h \in U_h & : \quad & r_h(u_h, v) = 0 & \forall v \in V_h, \\ \Leftrightarrow \quad & \mathbf{u} \in \mathbf{U} & : \quad & r_h\left(\mathrm{FE}_{\Phi_{U_h^k}}(\mathbf{u}), \phi_i\right) = 0 & i \in \mathcal{I}_{V_h}, \\ \Leftrightarrow \quad & \mathbf{u} \in \mathbf{U} & : \quad & \mathcal{R}(\mathbf{u}) = \mathbf{0}. & \end{aligned}$$

where

$$\mathcal{R} : \mathbf{U} \to \mathbf{V}, \qquad (\mathcal{R}(\mathbf{u}))_i := r_h\left(\mathrm{FE}_{\Phi_{U_h}}(\mathbf{u}), \phi_i\right).$$

For linear PDEs $\mathcal{R}$ is affine linear: $\mathcal{R}(\mathbf{u}) = \mathbf{Au} - \mathbf{b}$.

(Note: Can be extended to the constrained case).

Discontinuous Galerkin Finite Element Method

OBB method for Poisson equation $-\Delta u = f$ + BC reads

$$u_h \in W_h^k \quad : \quad r_h^{\text{OBB}}(u_h, v) = 0 \qquad \forall v \in W_h^k,$$

where W_h^k is piecewise polynomial of order k and

$$\begin{aligned} r_h^{\text{OBB}}(u,v) &= \sum_{e \in E_h^0} \int_{\Omega_e} \nabla u \cdot \nabla v \, dx - \sum_{e \in E_h^0} \int_{\Omega_e} fv \, dx \\ &+ \sum_{f \in E_h^1} \int_{\Omega_f} \langle \nabla v \cdot \nu_f \rangle [u]_f - [v]_f \langle \nabla u \cdot \nu_f \rangle \, ds \\ &+ \sum_{\substack{b \in B_h^1 \\ \Omega_b \subseteq \Gamma_D}} \int_{\Omega_b} (\nabla v \cdot \nu_f)(u - g) - v(\nabla u \cdot \nu_f) \, ds + \sum_{\substack{b \in B_h^1 \\ \Omega_b \subseteq \Gamma_N}} \int_{\Omega_b} jv \, ds. \end{aligned}$$

Separation into volume, skeleton and boundary terms.

No constraints are required.

Evaluation of Residual Map

Using splitting and localization properties we obtain

$$\begin{aligned}
\mathcal{R}(\mathbf{u}) = &\sum_{e\in E_h^0} \mathbf{R}_e^T \boldsymbol{\alpha}_{h,e}^{\text{vol}}(\mathbf{R}_e\mathbf{u}) &&+ \sum_{e\in E_h^0} \mathbf{R}_e^T \boldsymbol{\lambda}_{h,e}^{\text{vol}} \\
&+ \sum_{f\in E_h^1} \mathbf{R}_{l(e),r(e)}^T \boldsymbol{\alpha}_{h,f}^{\text{skel}}(\mathbf{R}_{l(e),r(e)}\mathbf{u}) &&+ \sum_{f\in E_h^1} \mathbf{R}_{l(e),r(e)}^T \boldsymbol{\lambda}_{h,f}^{\text{skel}} \\
&+ \sum_{b\in B_h^1} \mathbf{R}_{l(e)}^T \boldsymbol{\alpha}_{h,b}^{\text{bnd}}(\mathbf{R}_{l(e)}\mathbf{u}) &&+ \sum_{b\in B_h^1} \mathbf{R}_{l(b)}^T \boldsymbol{\lambda}_{h,b}^{\text{bnd}}.
\end{aligned}$$

At most six element-local methods for $\boldsymbol{\alpha}_{h,e}^{\text{vol}}$, $\boldsymbol{\alpha}_{h,f}^{\text{skel}}$, $\boldsymbol{\alpha}_{h,b}^{\text{bnd}}$, $\boldsymbol{\lambda}_{h,e}^{\text{vol}}$, $\boldsymbol{\lambda}_{h,f}^{\text{skel}}$ and $\boldsymbol{\lambda}_{h,b}^{\text{bnd}}$ need to be implemented by the user.

Restriction and prolongation operators are generic.

$\nabla\mathcal{R}$ can be generic through numerical differentiation.

Instationary Case

General one step method in method of lines approach:

1. $u_h^{(0)} = u_h^n$.
2. For $i = 1, \dots, s \in \mathbb{N}$, find $u_h^{(i)} \in w_h(t^n + d_i k^n) + \tilde{U}_h^k(t^{n+1})$:

$$\sum_{j=0}^{s} \Big[a_{ij} m_h \left(u_h^{(j)}, v; t^n + d_j k^n \right) + b_{ij} k^n r_h \left(u_h^{(j)}, v; t^n + d_j k^n \right) \Big] = 0 \qquad \forall v \in \tilde{U}_h^k(t^{n+1}).$$

3. $u_h^{n+1} = u_h^{(s)}$.

Requires a second residual form $m_h(u, v; t)$ for storage term.

General Discrete Function Spaces

$\Omega \subset \mathbb{R}^n$, $n \geq 1$, is a domain, $\mathbb{T}_h$ a grid partitioning the domain Ω.

$$U_h(\mathbb{T}_h) = \left\{ u_h(x) : \bigcup_{e \in E_h^0} \Omega_e \to \mathbb{K}^m \;\middle|\; u_h(x) = \sum_{e \in E_h^0} \sum_{i=0}^{k(e)-1} (\mathbf{u})_{g(e,i)}\, \pi_e(\hat{x})\, \hat{\phi}_{e,i}(\hat{x})\, \chi_e(x);\ \hat{x} = \mu_e^{-1}(x) \right\}$$

defines a general discrete function space of element-wise continuous functions.

$\chi_\omega(x) : \Omega \to \{0,1\}$ is the characteristic function of $\omega \subseteq \Omega$.

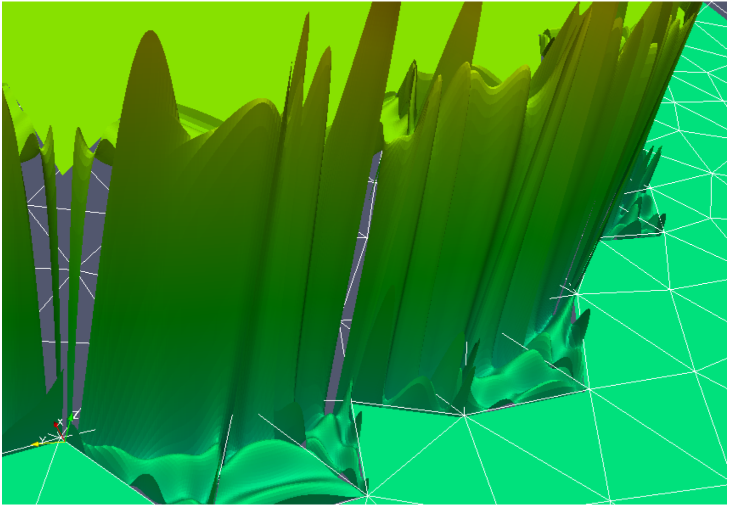
FE-Isomorphism: $\mathrm{FE}_{\Phi_{U_h}} : \mathbf{U} = \mathbb{K}^{\mathcal{I}_{U_h}} \leftrightarrow U_h(\mathbb{T}_h)$.

Some Code Metrics

Coding effort for different finite element spaces:

Local finite element	Source lines of code
Lagrange, order 1, simplex, $d = 1, 2, 3$	708
Lagrange, order 1, cube, $d = 1, 2, 3$	495
Lagrange, order 2, cube, $d = 2$	262
Lagrange, order k, simplex, $d = 2, 3$	1075
Monomial, order k, any d	520
Rannacher-Turek, quadrilateral	209
Raviart-Thomas, lowest order, simplex	323
L_2-orthogonal pol., any k, simplex, cube	800

Function Space Example



Order 7 discontinuous polynomials on triangles

Function Space Composition

- Systems of PDEs require composite function spaces.
- Given $U^{(0)}, \dots, U^{(k-1)}$, $k > 1$ define composite function space

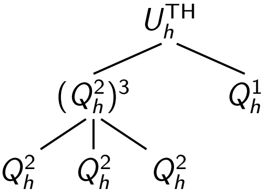
$$U = U^{(0)} \times U^{(1)} \times \dots \times U^{(k-1)}.$$

- For k identical components set

$$U = V^k.$$

- A component can be composite itself.
- Example: Taylor-Hood elements on cubes in 3D.

$U_h^{\text{TH}} = (Q_h^2)^3 \times Q_h^1$ function space tree:



Taylor-Hood Code Example

```
template<class GV> void taylorhood (const GV& gv) {
  // types
  typedef typename GV::Grid::ctype D;
  typedef double R;
  const int d = GV::dimension;

  // make finite element maps
  typedef Dune::PDELab::Q1LocalFiniteElementMap<D,R,d> Q1FEM;
  Q1FEM q1fem;                              // Q1 finite elements
  typedef Dune::PDELab::Q2LocalFiniteElementMap<D,R,d> Q2FEM;
  Q2FEM q2fem;                              // Q2 finite elements

  // make grid function spaces
  typedef Dune::PDELab::GridFunctionSpace<GV,Q1FEM> Q1GFS;
  Q1GFS q1gfs(gv,q1fem);                    // Q1 space
  typedef Dune::PDELab::GridFunctionSpace<GV,Q2FEM> Q2GFS;
  Q2GFS q2gfs(gv,q2fem);                    // Q2 space
  typedef Dune::PDELab::PowerGridFunctionSpace<Q2GFS,d> VGFS;
  VGFS vgfs(q2gfs);                         // velocity space
  typedef Dune::PDELab::CompositeGridFunctionSpace<
    Dune::PDELab::GridFunctionSpaceLexicographicMapper,
  VGFS,Q1GFS> THGFS;
  THGFS thgfs(vgfs,q1gfs);                  // Taylor-Hood space
  ...
```

General Constrained Spaces

- Constrained spaces turn up in a number of other cases:
  - Dirichlet boundary conditions.
  - Hanging nodes.
  - Functions with zero average, rigid body modes.
  - Conforming p-method.
  - Periodic boundary conditions.
  - Artificial boundary conditions in parallelization.
- PDELab has a general concept to handle all types of constraints.
- Given U_h with index set $\mathcal{I}_{U_h}$, construct a basis of the subspace:
  - Partition index set: $\mathcal{I}_{U_h^k} = \tilde{\mathcal{I}} \cup \bar{\mathcal{I}}$.
  - Construct new basis: $\tilde{\phi}_i = \phi_i + \sum_{j \in \bar{\mathcal{I}}} \omega_{i,j} \phi_j$, $i \in \tilde{\mathcal{I}}$.
  - $\tilde{U}_h$ is spanned by the new basis.

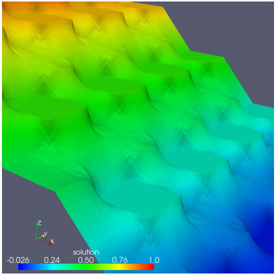
Six Easy Pieces

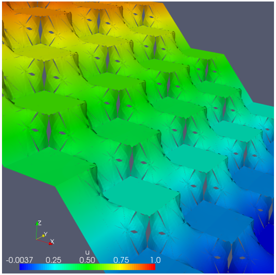
Solve

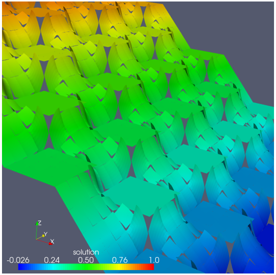
$$
\begin{aligned}
\nabla \cdot \sigma = f, \quad \sigma = -K(x)\nabla u & \qquad \text{in } \Omega, \\
u = g & \qquad \text{on } \Gamma_D \subseteq \partial\Omega, \\
\sigma \cdot \nu = j & \qquad \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D.
\end{aligned}
$$

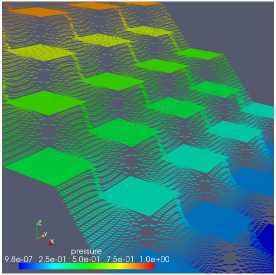
- P_k, Q_1, Q_2, P_1/Q_1 with hanging nodes, 298 LOC.
- Non-conforming finite elements (Rannacher-Turek)
- DG: OBB, SIPG, NIPG, 914 LOC.
- CCFV, two-point flux, harmonic averaging, 222 LOC.
- H(div)-conforming mixed method, no hybridization, 288 LOC.
- Mimetic finite difference method, 395 LOC.

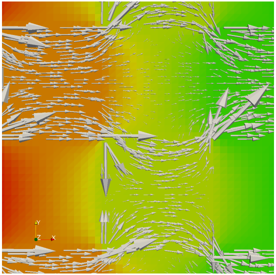
Six Easy Pieces (Pictures)

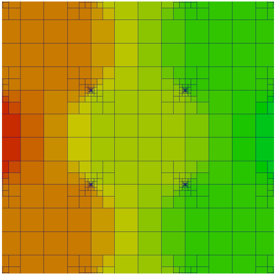






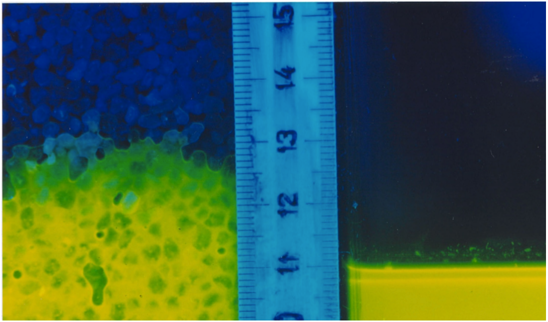




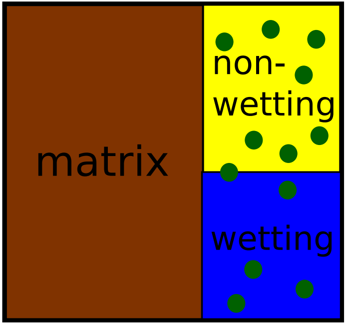


Contents

Model Concept



(Peter Gratwohl, Tübingen)



- Phases are immiscible on the micro scale
- Rock matrix assumed rigid
- Each fluid phase may be a mixture; phase exchange

Multiphase Reactive Flow Model

Mole (mass) balance of component $\kappa \in \mathcal{K}_\alpha$ in phase $\alpha \in \mathcal{P} = \{l, g\}$:

$$\partial_t(\phi s_\alpha \nu_\alpha x_{\alpha,\kappa}) + \nabla \cdot \{\nu_\alpha x_{\alpha,\kappa} u_\alpha + j_{\alpha,\kappa}\} = q_{\alpha,\kappa} + e_{\alpha,\kappa} + r_{\alpha,\kappa}$$

Extended Darcy's law, dispersion:

$$u_\alpha = -\frac{k_{r\alpha}(s_\alpha)}{\mu_\alpha(p_\alpha)} K \left(\nabla p_\alpha - \rho_\alpha g\right), \qquad j_{\alpha,\kappa} = -D_{\alpha,\kappa}(\ldots)\nabla x_{\alpha,\kappa}$$

Capillary pressure, ideal gas law, constraints ($0 \le s_\alpha, x_{\alpha,\kappa} \le 1$):

$$p_g - p_l = p_c(s_l), \quad \nu_g = p_g/RT, \quad s_l + s_g = 1, \quad \forall \alpha : \sum_{\kappa \in \mathcal{K}_\alpha} x_{\alpha,\kappa} = 1$$

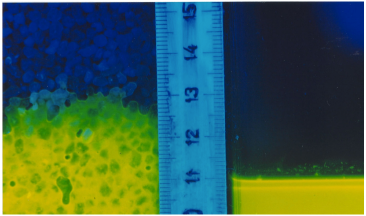
Phase exchange and reaction (equilibrium or kinetic):

$$x_{l,\kappa} = p_g x_{g,\kappa}/H \quad \text{(Henry's law)}, \qquad r_{\alpha,\kappa} = f(x_{\alpha,\kappa_1}, x_{\alpha,\kappa_2}, \ldots)$$

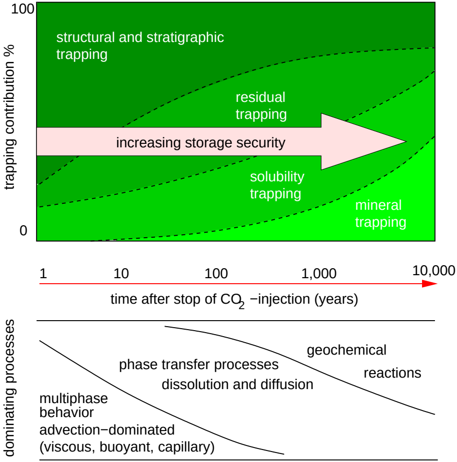
Energy balance, boundary conditions, initial conditions, ...

Applications

- Microbial activity in the capillary fringe
- Safety analysis of nuclear waste repositories
- CO_2 sequestration
- Enhanced oil and gas recovery



(Peter Gratwohl, Tübingen)



(IPCC, 2006)

Splitting Algorithm (with Olaf Ippisch)

1. **Phase transport**
   - Pressure/pressure formulation, solve two coupled PDEs for p_α.
   - Fixed phase composition (ρ_α in Darcy)
   - Cell-centered finite volume discretization, mobility upwinding
2. **Phase composition**
   - Mole fractions for $|\mathcal{K}_\alpha| - 1$ components per phase
   - Explicit schemes if advection dominated (substep if necessary)
   - Implicit schemes if diffusion dominated
3. **Phase exchange and reaction**
   - System of ODEs per cell or solve directly for equilibrium state
   - Yields new phase composition, p_g and s_α
4. Iterate if necessary

Only steps 1 and 2 are implemented so far :-(

Phase Balance Equations

- For $\alpha \in \mathcal{P} = \{l, g\}$:

$$\partial_t(\phi s_\alpha \nu_\alpha) + \nabla \cdot \{\nu_\alpha u_\alpha\} = q_\alpha,$$
$$u_\alpha = -\frac{k_{r\alpha}(s_\alpha)}{\mu_\alpha(p_\alpha)} K \left(\nabla p_\alpha - \rho_\alpha g\right).$$

with boundary and initial conditions

$$p_\alpha = \psi_\alpha \text{ on } \Gamma_\alpha^D, \quad \nu_\alpha u_\alpha \cdot n = \eta_\alpha \text{ on } \Gamma_\alpha^N \quad p_\alpha(0, x) = p_{\alpha,0}(x).$$

- Primary unknowns: p_l, p_g, upwinding of $p_c = p_g - p_l$.
- CCFV, two-point flux approximation, RT0 velocity reconstruction, 1100 LOC.
- Implicit Euler, Alexander2, fractional step θ time discretization.

Component Transport

- Generic balance equation:

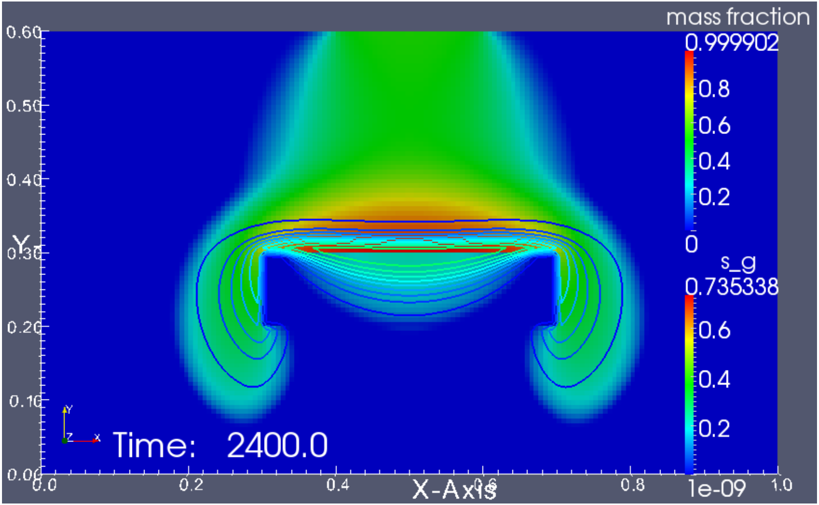
$$\partial_t(\phi s_\alpha c_{\alpha,\kappa}) + \nabla \cdot \{u_\alpha c_{\alpha,\kappa} - D_{\alpha,\kappa} \nabla c_{\alpha,\kappa}\} = q_{\alpha,\kappa}$$

- s_α, u_α vary in space and time
- $s_\alpha = 0$ if phase α is not present: “wetting and drying”
- if $s_\alpha^{n+1} < \epsilon$ then $s_\alpha^n < \epsilon$ is true and cell is marked “inactive”
- Phase disappearance is only allowed in reaction step
- Explicit CCFV, full upwinding, 600 LOC, stability constraint:

$$\Delta t^n \leq \min_{T \in \mathcal{T}_h} \frac{\phi_T s_T^n |T|}{\sum_{F \in \mathcal{F}_h^i(T) \cup \mathcal{F}_h^D(T)} \max(0, u_F^n \cdot n)|F|}$$

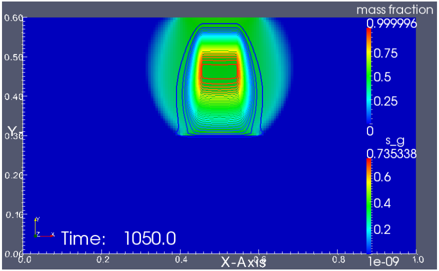
- Implicit scheme handles inactive cells as constrained DOFs

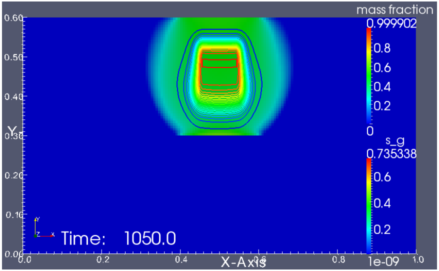
DNAPL Infiltration example + Transport

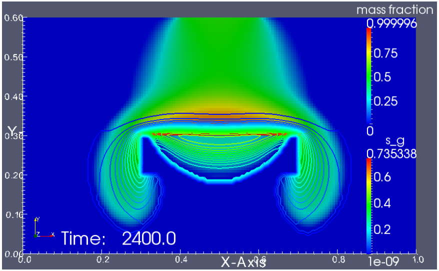


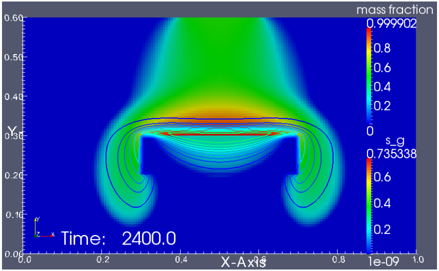
160 × 96, 160 time steps, implicit Euler / explicit Euler

Implicit / Explicit Comparison









Weak scaling, 3D, $120 \times 120 \times 72$ / Processor

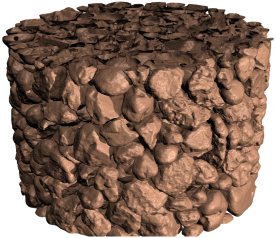
- Helics II: 160 nodes, AMD dual processor dual core, Myrinet 10G
- $T = 540[s]$, Newton reduction 10^{-8}
- Aggregation-based AMG preconditioner
- $P \cdot 2073600$ degrees of freedom per time step

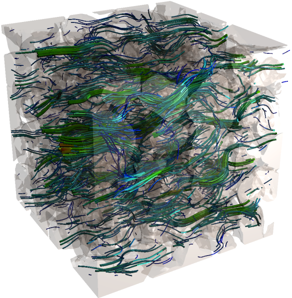
P	TS	ANlIt	ALinIt	TBuild	TIt	TAss	TTotal	S
1	9	5.6	2.8	6.5	5.3	20.8	2623	-
8	18	5.4	3.9	16.7	7.0	30.3	7611	5.5
64	36	5.6	5.4	18.8	8.5	30.3	19718	34
512	72	5.5	6.9	22.6	15.1	30.8	63994	168

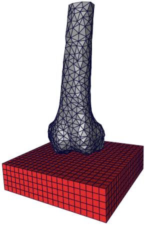
- Our scalability starts at $P = 1$!
- $P \leq 64$ uses only one core per node
- Room for improvement: reuse AMG hierarchy

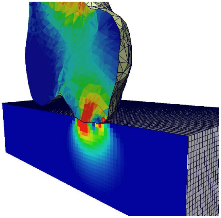
Other DUNE Applications

- Cut-cell DG method (C. Engwer, U Heidelberg)
- Compact and spectral DG methods (A. Dedner, R. Klöfkorn, U Freiburg)
- Two-body contact problem (O. Sander, FU Berlin)
- Maxwell equations (B. Oswald, PSI)









Summary

- DUNE, a powerful framework for PDE numerics:
  - Interface for general, parallel grids.
  - Platform for implementing high-end finite element methods.
  - Combine flexibility and efficiency through generic programming.
- Future work:
  - Adaptivity in PDELab.
  - Support for multi-domain multi-physics applications.
  - Integration of more grid implementations.
  - Adapt to many-core and hybrid architectures.