Solving multidomain problems with PDELab and dune-multidomain

Steffen Müthing

October 6, 2010
Motivation

dune-multidomaingrid

dune-multidomain

Example
Motivation

Why software infrastructure for problem coupling and multidomain / multiphysics problems?

▶ Many interesting problems to investigate in multiphysics settings.
▶ Most real world problems involve more than a single equation / domain.
Motivation

Why software infrastructure for problem coupling and multidomain / multiphysics problems?

- Many interesting problems to investigate in multiphysics settings.
- Most real world problems involve more than a single equation / domain.
- Non-negligible amount of ”bookkeeping” required for tracking interfaces, degrees of freedom etc.
  ⇒ Simulations often restricted to simple geometries.
Typical Coupling Configurations

**Surface Couplings**

Direct coupling of two problems $P_1, P_2$ on their common interface:

\[
\begin{array}{c}
P_1 \\
\hline
P_2
\end{array}
\]

Indirect coupling using a mortar space $\Lambda$ with additional DOF on the interface:

\[
\begin{array}{c}
P_1 \\
\hline
\Lambda
\end{array}
\]

**Volume Couplings**

Distinct problems sharing (some) underlying function spaces:

\[
P_1, P_2
\]

Interface tracking using level set method with enrichment space $U_E$:

\[
P_1, P_2, U_E
\]
Typical Coupling Configurations

Surface Couplings

Direct coupling of two problems $P_1, P_2$ on their common interface:

\[ P_1 \quad P_2 \]

Indirect coupling using a mortar space $\Lambda$ with additional DOF on the interface:

\[ P_1 \quad P_2 \quad \Lambda \]

Volume Couplings

Distinct problems sharing (some) underlying function spaces:

\[ P_1 \quad P_2 \]

Interface tracking using level set method with enrichment space $U_E$:

\[ U_E \]
Challenges

Large number of mostly technical challenges:

▶ Labelling the spatial domains of function spaces / subproblems
▶ Manage the degrees of freedom of involved function spaces
▶ Efficient matrix / residual assembly:
  ▶ Minimize number of grid traversals
  ▶ Identify locally defined subproblems
  ▶ Load per-subproblem set of local degrees of freedom and invoke appropriate operators
▶ Output solution of function spaces defined on subdomains

Goal: Automate tasks and enable rapid prototyping of numerical methods with good performance and generality.
Approach

Split responsibilities:

▶ Spatial information about subdomains handled at the grid interface level
  ⇒ dune-multidomaingrid
Approach

Split responsibilities:

▶ Spatial information about subdomains handled at the grid interface level
  ⇒ dune-multidomaingrid

▶ PDELab extension for function space management and problem assembly
  ⇒ dune-multidomain

Currently limited to dune-multidomaingrid for subdomain information, extension to distinct per-subdomain grids possible (dune-grid-glue).
dune-multidomaingrid: Basics

- Provides meta grid `MultiDomainGrid`
- Basic assumption: Use a single underlying spatial discretisation – a single grid – for the complete domain.

1. Wrap existing grid in meta grid

2. Mark subdomains

3. Subdomains also exposed as separate meta grids
Design of MultiDomainGrid

- Many ideas from dune-subgrid by Oliver Sander and Carsten Gräser
- API for subdomain setup similar to grid adaptation API
- Subdomain layout not fixed, can be changed during the runtime of the program
- Subdomains always comprise the complete grid hierarchy
- Support for disabling certain features (indices for some codims, level index sets) for performance
- Pluggable storage backend
Short Introduction to PDELab

Main ideas:

- Support for rapid prototyping
- Good flexibility
- The user is only exposed to a local view of the problem (finite element on reference element and mapping to world space)
Short Introduction to PDELab

- Discrete function spaces
  - Bound to a grid view
  - Based on local finite elements from dune-localfunctions
  - General approach to constraints handling
  - Generic generation of product spaces for systems

- Operators based on weighted residual formulation
- Support for numerical schemes requiring at most face-neighbors
- Also responsible for local description of sparsity pattern
- Exchangeable linear algebra backend
- Integrated Newton solver and generic one step methods for instationary problems
Short Introduction to PDELab

- Discrete function spaces
  - Bound to a grid view
  - Based on local finite elements from dune-localfunctions
  - General approach to constraints handling
  - Generic generation of product spaces for systems

- Operators based on weighted residual formulation
  - Support for numerical schemes requiring at most face-neighbors
  - Also responsible for local description of sparsity pattern
Short Introduction to PDELab

- Discrete function spaces
  - Bound to a grid view
  - Based on local finite elements from dune-localfunctions
  - General approach to constraints handling
  - Generic generation of product spaces for systems
- Operators based on weighted residual formulation
  - Support for numerical schemes requiring at most face-neighbors
  - Also responsible for local description of sparsity pattern
- Exchangeable linear algebra backend
- Integrated Newton solver and generic one step methods for instationary problems
dune-multidomain: Features

Functionality provided by dune-multidomain for implementing multidomain problems with PDELab:

- Function spaces defined on parts of the whole domain.
- Support for defining subproblems, connecting operators and function spaces.
- Support for defining interface couplings between pairs of subproblems.
- Automatic assembly of resulting multidomain system.

Requires compiler support for variadic templates!
dune-multidomain: Features

Functionality provided by dune-multidomain for implementing multidomain problems with PDELab:

▶ Function spaces defined on parts of the whole domain.
▶ Support for defining subproblems, connecting operators and function spaces.
▶ Support for defining interface couplings between pairs of subproblems.
▶ Automatic assembly of resulting multidomain system.

Requires compiler support for variadic templates!
Specifying subproblem domains: Predicates

Problem: Subproblem domains not necessarily aligned with domain of any function space.

Example: Groundwater contamination

\[ U_{sn} \quad U_{pw} \quad \text{two phase flow} \quad \text{single phase flow} \]

Solution: Define predicate \( P : \mathcal{P}(S) \rightarrow \{0, 1\} \) based on set of function spaces present in a grid cell:

- Single phase flow: \( P_1(S) = \mathbb{1}_{\{U_{pw}\}}(S) \),
- Two phase flow: \( P_2(S) = \mathbb{1}_{\{U_{pw}, U_{sn}\}}(S) \).
Grid Function Space Handling

- Grid function spaces can be defined on MultiDomainGrid and any associated SubDomainGrid.
- Full support for function space trees (for modeling systems of PDEs).
- CouplingGridFunctionSpace for placing degrees of freedom on codim 1 manifolds in the grid.
- New MultiDomainGridFunctionSpace transparently glues together standard PDELab grid function spaces defined on different parts of the domain.
Example: Grid function spaces

```cpp
// define finite elements
typedef Dune::PDELab::Pk2DLocalFiniteElement<GV, double, 1> FEM1;
FEM1 fem1;
typedef Dune::PDELab::Pk2DLocalFiniteElement<GV, double, 2> FEM2;
FEM2 fem2;

typedef Dune::PDELab::ConformingDirichletConstraints CON;

// normal grid function spaces
typedef Dune::PDELab::GridFunctionSpace<MultiDomainGridView, FEM1, CON> GFS1;
GFS1 gfs1(multidomaingridview, fem1);
typedef Dune::PDELab::GridFunctionSpace<SubDomainGridView, FEM2, CON> GFS2;
GFS2 gfs2(subdomaingridview, fem2);

// composite grid function space
typedef Dune::PDELab::MultiDomain::MultiDomainGridFunctionSpace<
    Grid, GFS1, GFS2> MultiGFS;
MultiGFS multigfs(multidomaingridview, gfs1, gfs2);
```
Subproblem encapsulation

New class SubProblem bundles all information defining a subproblem:

- Local Operator,
- Required (ansatz and test) grid function spaces from MultiDomainGridFunctionSpace,
- Predicate for spatial domain of subproblem,
- Constraints assembler for subproblem boundaries.
Subproblem encapsulation

New class SubProblem bundles all information defining a subproblem:

- Local Operator,
- Required (ansatz and test) grid function spaces from MultiDomainGridFunctionSpace,
- Predicate for spatial domain of subproblem,
- Constraints assembler for subproblem boundaries.

Important: Subproblems (and associated operators etc.) are always defined directly on the MultiDomainGrid!
Example: Subproblems

```
// define predicates
typedef Dune::PDELab::MultiDomain::SubDomainEqualityCondition<Grid > EC;
EC c0();  // empty set
EC c1(0);  // exactly subdomain 0

// local operators
typedef SinglePhaseFlowOperator SPFO; SPFO spfo;
typedef TwoPhaseFlowOperator TPFO; TPFO tpfo;

// single phase flow problem
typedef Dune::PDELab::MultiDomain::SubProblem<MultiGFS,CON,
    MultiGFS,CON,SPFO,EC,GFS1> SPFOSubProblem;
SPFOSubProblem spfosubproblem(con,con,spfo,c0);

// two phase flow problem
typedef Dune::PDELab::MultiDomain::SubProblem<MultiGFS,CON,
    MultiGFS,CON,TPFO,EC,GFS1,GFS2> TPFOSubProblem;
TPFOSubProblem tpfosubproblem(con,con,tpfo,c1);
```
Surface couplings between subproblems

Couplings are completely defined by a tuple (SubProblemA, SubProblemB, CouplingOperator).

- Couplings are oriented, SubProblem A will always be the first argument to any operator methods and be located on the inside of the passed intersection.
- The CouplingOperator resembles a normal PDELab operator with different flags and methods:
  - Flags doPatternCoupling, doAlphaCoupling,
  - Methods pattern_coupling(), alpha_coupling(), jacobian_coupling(), jacobian_apply_coupling()
- Default Implementations for full pattern creation and numeric jacobian evaluation.
Example: Couplings

class CouplingOperator
{

    static const bool doAlphaCoupling = true;

    template<typename IG,
             typename LFSUA, typename LFSVA,
             typename LFSUB, typename LFSVB,
             typename X, typename R>
    void alpha_coupling(const IG& ig,
                         const LFSUA& lfsua, const X& xa,
                         const LFSVA& lfsva,
                         const LFSUB& lfsub, const X& xb,
                         const LFSVB& lfsvb,
                         R& ra, R& rb)
    {
        ...
    }
};

typedef Dune::PDELab::MultiDomain::Coupling<SPFOSubProblem,
                                          TPFOSubProblem, CouplingOperator> Coupling;
Coupling coupling(spfosubproblem, tpfosubproblem, couplingoperator);
MultiDomainGridOperatorSpace

- Replaces standard GridOperatorSpace.
- Variants for stationary and instationary problems.
- Synopsis:

  ```cpp
typedef MultiDomainGridOperatorSpace<MultiGFS, MultiGFS, CG, CG, MatrixBackend, SPFOSubProblem, TPFOSubProblem, Coupling> MultiGOS;
MultiGOS multigos(multigfs, multigfs, cg, cg, spfosubproblem, tpfosubproblem, coupling);
```

- Subproblems and couplings can be listed in arbitrary order.
- No limit on the number of subproblems or couplings (apart from compiler restrictions).
- Automatically assembles the residual and the mass matrix of the complete system.
Stokes-Darcy Coupling

Flow through a channel in a porous medium

Setting:

Mathematical model taken from:
Stokes-Darcy Coupling – Model (I)

Darcy equation with natural boundary conditions in the porous medium:

\[ \nabla \cdot (-K \nabla \phi_m) = f_2 \quad \text{in } \Omega_m, \]
\[ (\nabla \phi_m) \cdot n = 0 \quad \text{on } \Gamma_m, \]

where \( \phi_m \) the hydraulic head, \( K \) the permeability and \( n \) the outer unit vector. \( f_2 \) is a possible sink / source term.
Incompressible Navier-Stokes equations in the free-flow domain:

\[
\begin{align*}
\rho (\mathbf{v}_f \cdot \nabla) \mathbf{v}_f &= -\nabla p_f + \mu \nabla^2 \mathbf{v}_f + \mathbf{f}_1 \\
\nabla \cdot \mathbf{v}_f &= 0
\end{align*}
\]

in \( \Omega_f \),

where \( p_f \) pressure, \( \mathbf{v}_f \) velocity, \( \mu \) dynamic viscosity and \( \rho \) density. \( \mathbf{f}_1 \) contains exterior forces, in this case gravity.

We impose flux boundary conditions on the outer border of the free-flow domain:

\[
\mu \mathbf{v}_f \cdot \mathbf{n} = j \quad \text{on } \Gamma_f.
\]
Stokes-Darcy Coupling – Model (III)

Beavers – Joseph Conditions on the interface $\Gamma$:

\[
\begin{align*}
\mathbf{v}_f \cdot \mathbf{n} &= (\nabla \phi_m) \cdot \mathbf{n} \\
p_f - \frac{\mu}{\rho} \nabla^2 \mathbf{v}_f &= g(\phi_m - z) \\
\mathbf{v}_f \cdot \mathbf{n} &= \left(\frac{2 \mu g}{\rho \text{trace}(K)}\right) \frac{1}{\sqrt{\rho \text{trace}(K)}} P_T(\mathbf{v}_f - K \nabla \phi_m)
\end{align*}
\]

on $\Gamma$,

where $P_T(\cdot)$ denotes the projection onto the local tangent plane on $\Gamma$ and $z$ is the $z$ coordinate relative to the reference level of the hydraulic head.
Stokes-Darcy Coupling – Discretisation

- Taylor–Hood in the free-flow domain (reused implementation from Felix Heimann, included in PDELab).
- WIP-OBB degree 3 in the porous medium (reused implementation from Peter Bastian).
- Coupling operator implemented as described above \( \approx 150 \) LOC including parameter class.
Stokes-Darcy Coupling – Setting

- Underlying grid: UG.
- Mesh created in Gmsh (≈ 15 min.).
- 10267 elements, 88766 DOF.
- Parameters: \( \rho = 1000, \mu = 1, K = 10^{-4}, \phi = 0.5, \alpha = 1. \)
- Free-flow boundary conditions: \( j_{in} = 60, j_{out} = -60. \)
Stokes-Darcy Coupling – Results

Hydraulic pressure:

Velocity magnitude – different scales in the subdomains:
Summary

Fairly general extension of PDELab for handling multidomain and multiphysics problems.

- Can handle regular and mortar interface couplings, overlapping subdomains and local function space enrichment.
- Automates most of the management tasks related to the implementation of multidomain problems.
- (Currently) restricted to a single underlying master grid and assembly into one global matrix.

Current and future areas of work:

- Parallelisation
- Applications
- (Support for domain decomposition methods)
Thank you for your attention!