Infrastructure for the Coupling of Dune Grids

Peter Bastian, Gerrit Buse, and Oliver Sander

Abstract We describe an abstract interface for the geometric coupling of finite element grids. The scope of the interface encompasses a wide range of domain decomposition techniques in use today, including nonconforming grids and grids of different dimensions. The couplings are described as sets of remote intersections, which encapsulate the relationships between pairs of elements on the coupling interface.

The abstract interface is realized in a module `dune-grid-glue` for the software framework DUNE. Several implementations of this interface exist, including one for general nonconforming couplings and a special efficient implementation for conforming interfaces. We present two numerical examples to show the flexibility of the approach.

1 Introduction

Domain decomposition methods are a standard tool for a wide range of multiphysics problems. Whenever the application involves subdomains with different equations, discretizations, or grid types, coupling conditions and domain decomposition algorithms need to be employed. We refer to [7] for a general introduction.

Even though domain decomposition methods have found widespread use, the software support available is generally not satisfactory. Implementing domain decomposition methods can be tedious and error prone, especially when nonmatch-

Peter Bastian
Universität Heidelberg, Germany, e-mail: `peter.bastian@iwr.uni-heidelberg.de`

Gerrit Buse
Technische Universität München, Germany, e-mail: `buse@in.tum.de`

Oliver Sander
Freie Universität Berlin, Germany, e-mail: `sander@mi.fu-berlin.de`

ing grids are involved. A central problem is finding the geometric correspondences between the grids. Today, there still exist mainly ad hoc solutions geared towards specific purposes, with little chance of code reuse.

In this article, we propose a general implementation as part of the DUNE framework [1]. DUNE is a set of C++ libraries providing support for various aspects of grid-based PDE solution methods such as grids, linear algebra, or shape functions. DUNE's main goal is flexibility, achieved by defining abstract interfaces to such things as grids and shape functions, and allowing the user to select the appropriate implementation according to his or her needs. DUNE also promotes code reuse by a modular architecture and by allowing legacy implementations to be used with the interface.

For our domain decomposition infrastructure we have tried to follow the same philosophy:

- We propose abstract interfaces to general grid coupling mechanisms, allowing to implement most existing domain decomposition algorithms.
- We allow and encourage the use of existing coupling implementations as legacy backends.
- We strive to make the code efficient, using generic programming where appropriate.

Adhering to the modular structure of DUNE, our code is available as a DUNE module, termed `dune-grid-glue`.

2 General Grid Coupling

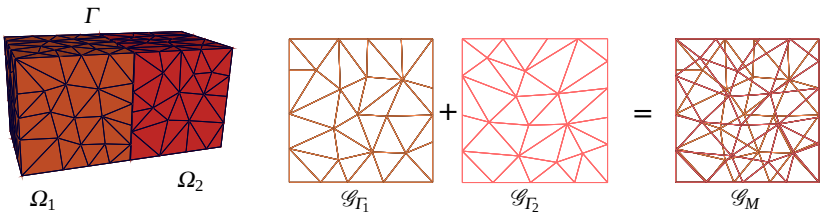

Fig. 1 Left: two domains Ω_1 and Ω_2 that meet at a common interface Γ. Center: the restrictions of the two grids on Γ. Right: together they form the set of remote intersections $\mathcal{G}_M$.

We begin by describing the concept of the abstract grid coupling interface. For simplicity we focus on the case of nonoverlapping coupling. Consider two domains Ω_1, Ω_2 that meet at a common interface Γ (Fig. 1). Both domains are assumed to be discretized by grids, not necessarily simplicial. The restrictions of the grids to the coupling boundary, denoted by $\mathcal{G}_{\Gamma_1}$ and $\mathcal{G}_{\Gamma_2}$, are not related to each other in any way.

Overlaying these two boundary grids results in a set of intersections of the elements of $\mathscr{G}_{\Gamma_1}$ and $\mathscr{G}_{\Gamma_2}$, which we call $\mathscr{G}_M$. Together with the embeddings into $\mathscr{G}_{\Gamma_1}$ and $\mathscr{G}_{\Gamma_2}$, the intersections constitute the information necessary to implement most nonoverlapping domain decomposition algorithms.

As an example, consider the mortar method. There, the coupling is effected through a mass matrix

$$M \in \mathbb{R}^{n\times m}, \qquad M_{ij} = \int_\Gamma \phi_i \psi_j \, ds, \tag{1}$$

where the ϕ_i, ψ_j, $0 \le i < n$, $0 \le j < m$, are finite element basis functions on $\mathscr{G}_{\Gamma_1}$ and $\mathscr{G}_{\Gamma_2}$, respectively. The matrix can be computed by splitting the integral in (1) into a sum of integrals over individual elements of $\mathscr{G}_M$. By construction, to each element $e \in \mathscr{G}_M$ correspond unique elements of $\mathscr{G}_{\Gamma_1}$ and $\mathscr{G}_{\Gamma_2}$, and associated shape functions there. If a quadrature rule is available for e, then $\int_e \phi_i \psi_j \, ds$ can be computed directly. Otherwise, e needs to be triangulated and $\int \phi_i \psi_j \, ds$ computed for each triangle.

The approach covers more than just mortar methods. If the two grids on Ω_1 and Ω_2 match, the set $\mathscr{G}_M$ degenerates and we have $\mathscr{G}_M = \mathscr{G}_{\Gamma_1} = \mathscr{G}_{\Gamma_2}$. In this case, the set of intersections e together with their embeddings into the elements of $\mathscr{G}_1$ and $\mathscr{G}_2$ allows to identify the grid vertices, or, more generally, edge and face degrees of freedom. Overlapping couplings can be handled by letting $\mathscr{G}_M$ have the same dimension as the computational grids $\mathscr{G}_1$ and $\mathscr{G}_2$. Finally, consider a d-dimensional grid attached in parallel to the boundary of a $d+1$-dimensional one (cf. Sec. 5.2). The grids may or may not be conforming on Γ. This time coupling is between the surface grid $\mathscr{G}_{\Gamma_2}$ and the grid $\mathscr{G}_1$ itself. As the dimensions are the same, a set of intersections just as in Fig. 1 is obtained.

3 Implementation: Remote Intersections

The intersections described in the previous section bear close resemblance to the intersections that are part of the DUNE grid interface [2, Sec. 4]. Within a single grid, DUNE intersections describe the coupling between neighboring elements. An intersection between two elements e_1 and e_2 is the (set-theoretic) intersection between θ_{e_1} and θ_{e_2}, where θ_{e_1} and θ_{e_2} are the subsets of the world space occupied by e_1 and e_2, respectively. The `Intersection` class of the DUNE grid interface provides information about these set intersections, e.g. their geometry in the world space, the geometry in coordinates of e_1 and e_2, normal vectors, and whether an intersection is conforming.

In the case of domain decomposition methods, the elements e_1 and e_2 are elements of different grids $\mathscr{G}_1$ and $\mathscr{G}_2$. However, the relevant information remains largely the same. We will call such intersections *remote intersections*, to distinguish them from the intersections of the DUNE grid interface. Remote intersections may be set-theoretic intersections if $\mathscr{G}_1$ and $\mathscr{G}_2$ meet at a common interface Γ. In case of

contact problems, where there may be a positive distance between $\mathscr{G}_{\Gamma_1}$ and $\mathscr{G}_{\Gamma_2}$, the remote intersections can be defined via a contact mapping $\Phi : \mathscr{G}_{\Gamma_1} \to \mathscr{G}_{\Gamma_2}$ (cf. [9]).

Due to the conceptual similarity between remote intersections and grid intersections it is natural to make the implementation of remote intersections resemble DUNE intersections as well. The `dune-grid-glue` module provides the class `RemoteIntersection`, which again has methods for the geometry of the intersection in world space, geometries in local coordinates of e_1 and e_2, normal vectors, etc. The main differences concern methods that deal with global coordinates. Since θ_{e_1} and θ_{e_2} may actually be disjoint (e.g., in a contact problem), there are two embeddings of the remote intersection in the world space. For the same reason, there are two methods for the normal vectors. Please see the class documentation provided with the module for details.

Access to the remote intersections is provided via three types of DUNE-style iterators. The `RemoteIntersectionIterator` iterates over the entire set of remote intersections and can be used to, e.g., assemble mortar mass matrices. The `DomainIntersectionIterators` and `TargetIntersectionIterators` iterate over all remote intersections of a given element of $\mathscr{G}_1$ or $\mathscr{G}_2$, respectively. This can be useful to assemble element-wise contributions in DG methods.

4 Constructing Couplings

The construction of sets of remote intersections proceeds in two steps. First, the grid interface boundaries or coupling parts are extracted and transformed to an intermediate representation. Then, two such extracted grids are combined to yield the set of remote intersections.

4.1 Extractors

`Extractor` classes select the subsets of grid entities that are involved in the coupling. They are classified according to the codimension (with respect to the grids) of the objects they extract. The most common one, `Codim1Extractor`, extracts boundary faces, and will be used for nonoverlapping couplings. The faces are marked using predicate classes provided by the user. The `Codim0Extractor` extracts actual elements. Such extractors will be needed for an overlapping coupling. A `Codim2Extractor` has not been implemented yet, but may be useful to couple, e.g., 1d partial differential equations to sequences of edges in a 3d mesh.

The extracted grid entities can be manipulated with a geometric transformation $\mu : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$, $n_1 \leq n_2$. This may be a deformation or an embedding into a higher-dimensional space. There are various uses for such a feature. For example, you may want to consider coupled problems on deformed meshes, such as the finite-strain contact problem described in [8]. Also, when coupling a 1d grid to the boundary of

a 2d grid, then most likely the 1d grid implementation will live in a 1d world. A transformation can then be used to place the 1d grid in the 2d world and deform it, if necessary (see Sec. 5.2 for an example).

4.2 Computing Remote Intersections

With the two interacting grid parts extracted, they can be combined to obtain the set of remote intersections. How this should be implemented differs considerably depending on the actual scenario. A general implementation computing remote intersections would have to handle nonmatching grids and geometries, grids of arbitrary dimensions and element types. Besides being very difficult to write and debug, such a program would be inefficient in more regular situations such as when the grids match.

To resolve this dilemma we follow the DUNE philosophy. We prescribe an abstract interface that algorithms computing remote intersections should conform to. We then provide different implementations of the interface for different cases such as contact problems, conforming meshes, or overlapping grids. Also in accordance with the DUNE philosophy, legacy implementations can be used through the interface.

The current default implementation uses the PSURFACE library. This library was originally written to manage boundary parametrizations [6], and extended to also handle mappings for contact problems [9]. It manages piecewise affine mappings between simplicial hypersurfaces in 2d and 3d. The surfaces are identified by a normal projection $\Phi : \Gamma_1 \to \Gamma_2$. PSURFACE is free software and can be downloaded from `http://numerik.mi.fu-berlin.de/dune/psurface`.

Also, a special efficient implementation `ConformingMerge` for conforming couplings is available.

5 Numerical Examples

In this last chapter we demonstrate some of the possibilities of `dune-grid-glue` with two example applications. The first one, a two-body contact problem, has already appeared in [1], where the coupling was implemented using PSURFACE directly.

5.1 Contact Between a Structured and an Unstructured Grid

In this first example we compute mechanical contact between a human femur bone and an elastic foundation. Consider two disjoint domains Ω_1, Ω_2 in $\mathbb{R}^3$. The

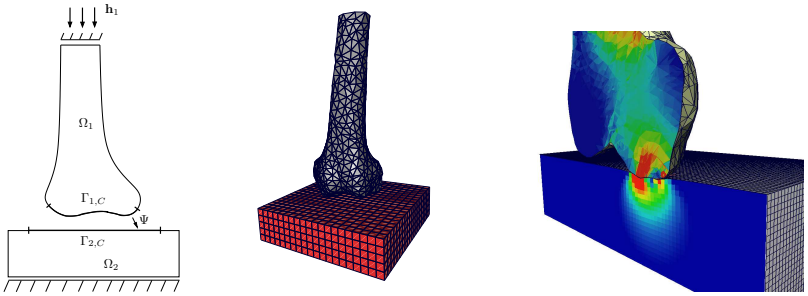

Fig. 2 Two-body contact problem. Left: schematic view. Center: coarse grids. Right: close-up view of the deformed solution.

boundary $\Gamma_i = \partial\Omega_i$, $i = 1,2$, of each domain is decomposed in three disjoint parts $\Gamma_i = \Gamma_{i,D} \cup \Gamma_{i,N} \cup \Gamma_{i,C}$. With $\mathbf{f}_i \in (L_2(\Omega_i))^3$ two body force density fields we look for functions $\mathbf{u}_i \in (H^1(\Omega_i))^3$ which fulfill

$$-\operatorname{div}\sigma(\mathbf{u}_i) = \mathbf{f}_i,$$

and suitable boundary conditions. The stress tensor σ is defined as $\sigma = \frac{E}{1+\nu}(\varepsilon + \frac{\nu}{1-2\nu}\operatorname{tr}\varepsilon I)$, and $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ is the linear strain tensor. For the contact condition, assume that the areas where contact occurs will be subsets of $\Gamma_{1,C}$ and $\Gamma_{2,C}$. These two contact boundaries are identified using a homeomorphism $\Phi : \Gamma_{1,C} \to \Gamma_{2,C}$, and this identification is used to define an initial distance function $g : \Gamma_{1,C} \to \mathbb{R}$, $g(x) = \|\Phi(x) - x\|$. The contact condition then states that the relative normal displacement of any two points x, $\Phi(x)$, $x \in \Gamma_{1,C}$, should not exceed this normal distance, in formulas

$$\mathbf{u}_1|_{\Gamma_{1,C}} \cdot \mathbf{n}_1 + (\mathbf{u}_2 \circ \Phi)|_{\Gamma_{2,C}} \cdot \mathbf{n}_2 \leq g, \tag{2}$$

where $\mathbf{n}_i$, $i = 1,2$, is the unit outward normal of $\Gamma_{i,C}$. Condition (2) can be derived as a linearization of the actual nonpenetration condition and is reasonable to use in the context of linear elasticity [4].

For the discretization of the problem we use first-order Lagrangian elements for the interior and dual mortar elements for the contact condition. That is, (2) is discretized in a weak form requiring

$$\int_{\Gamma_{1,C}} \left[\mathbf{u}_1|_{\Gamma_{1,C}} \cdot \mathbf{n}_1 + (\mathbf{u}_2 \circ \Phi)|_{\Gamma_{2,C}} \cdot \mathbf{n}_2\right] \theta \, ds \leq \int_{\Gamma_{1,C}} g\theta \, ds \tag{3}$$

for all θ from a cone of dual mortar test functions defined on $\Gamma_{1,C}$ [10]. The resulting discrete obstacle problem is solved with a truncated nonsmooth Newton multigrid method as described by Gräser et al. [5].

As the femur geometry we choose the distal part of the Visible Human femur data set. As grid implementations we use `UGGrid` for the femur and the structured hexahedral `SGrid` for the foundation. Material parameters are $E = 17\,\text{GPa}$,

$\nu = 0.3$ for the bone and softer $E = 250\,\mathrm{MPa}$, $\nu = 0.3$ for the obstacle. The latter is clamped at its base, whereas a uniform displacement of 3 mm downward is prescribed on the top section of the bone (see Fig. 2). The bone serves as the nonmortar domain. The computation of (3) involves a mortar mass matrix similar to (1). Two `Codim1Extractor`s are used to mark the contact boundaries and the remote intersections are computed using the PSURFACE backend. The result can be seen in Fig. 2, right.

5.2 Coupling a 2d Richards Equation and a 1d Shallow-Water Equation

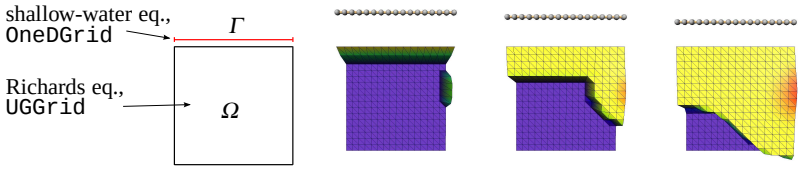

Fig. 3 Coupling the Richards equation to the shallow-water equation.

In the second example we show how `dune-grid-glue` can be used to couple two domains of differing dimensions.[1] Consider a domain Ω as in Fig. 3. It is supposed to represent a vertical section of ground. We assume unsaturated subsurface flow modeled by the Richards equation

$$\theta(p)_t + \operatorname{div}\mathbf{v}(p) = 0, \qquad \mathbf{v}(p) = -K\,\mathrm{kr}(\theta(p))\nabla(p - \rho g z),$$

for the water pressure p in Ω. We denote the upper horizontal boundary of Ω by Γ and assume surface water there modeled by the shallow water equations

$$\begin{aligned} h_t + \operatorname{div}\mathbf{q} &= F \\ \mathbf{q}_t + \operatorname{div}(\mathbf{q}^2/h + 0.5gh^2) &= -gh\nabla f, \end{aligned} \tag{4}$$

for the surface water height h and the horizontal water flux $\mathbf{q}$.

The two equations are coupled by assuming that the pressure p of the ground water on Γ equals the hydrostatic pressure induced by the surface water

$$p = \rho g h,$$

and that the flow $\mathbf{v}\cdot\mathbf{n}$ across Γ enters the surface water balance as an additive term in (4).

[1] The authors would like to thank C. Grümme and H. Berninger for their help with this example.

The coupled problem is solved with a Dirichlet–Neumann-type solver. At each iteration i, a Richards problem is solved on Ω with Dirichlet boundary conditions $p_i = \rho g h_i$ on Γ using a multigrid solver as described in [3]. Then 1 000 steps of the shallow-water equation are computed using a Lax–Friedrichs scheme. The flow $\mathbf{v}_i \cdot \mathbf{n}$ of subsurface water across Γ is interpolated in time and used as the source term in (4).

The Richards equation is discretized on a uniform triangle grid using the `UGGrid` grid manager. For the shallow water equation a `OneDGrid` is used. From the `UGGrid`, the interface Γ is extracted using a `Codim1Extractor` and the entire `OneDGrid` is extracted with a `Codim0Extractor`. A transformation $\tau : \mathbb{R} \to \mathbb{R}^2$ is given to the `Codim0Extractor` that places the 1d grid on the coupling boundary Γ such that the grids match. The `ConformingMerge` backend is used to generate the remote intersections. Fig. 3 shows several steps in the evolution of the problem.

References

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

[2] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander. A generic interface for parallel and adaptive scientific computing. Part I: Abstract framework. *Computing*, 82(2–3):103–119, 2008.

[3] H. Berninger, R. Kornhuber, and O. Sander. Fast and robust numerical solution of theRichards equation in homogeneous soil. *in preparation*, 2009.

[4] C. Eck. *Existenz und Regularität der Lösungen für Kontaktprobleme mit Reibung*. PhD thesis, Universität Stuttgart, 1996.

[5] C. Gräser, U. Sack, and O. Sander. Truncated nonsmooth Newton multigrid methods for convex minimization problems. In M. Bercovier, M. Gander, R. Kornhuber, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XVIII*, LNCSE. Springer, 2009.

[6] R. Krause and O. Sander. Automatic construction of boundary parametrizations for geometric multigrid solvers. *Comp. Vis. Sci.*, 9:11–22, 2006.

[7] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, 1999.

[8] O. Sander. A fast solver for finite deformation contact problems. Technical Report 319, DFG Research Center MATHEON, 2006.

[9] O. Sander. *Multidimensional Coupling in a Human Knee Model*. PhD thesis, Freie Universität Berlin, 2008.

[10] B. Wohlmuth and R. Krause. Monotone methods on nonmatching grids for nonlinear contact problems. *SIAM J. Sci. Comp.*, 25(1):324–347, 2003.