Concepts for flexible parallel multi-domain simulations

Christian Engwer[1] and Steffen Müthing[2]

1 Introduction

Domain Decomposition methods provide a flexible tool for developing multi-physics simulations and coupling different discretization methods. In general, multi-physics simulations will require the handling of non-matching grids. Domain Decomposition methods like the Mortar method [3] enable us to simulate complex applications like contact problems, mechanics of moving parts, or heterogeneous coupling like surface-/groundwater flow.

As we will discuss, coupling unrelated parallel meshes poses significant practical problems. To our knowledge only very few implementations exist: both the well-known MpCCI library [7] and the SIERRA framework implement a parallel rendezvous algorithm [8] based on intersection algorithms, but neither of them is publicly available. An alternative approach can be based on radial basis functions, see [5].

The DUNE framework [1] offers different strategies for Domain Decomposition methods, which are available as DUNE extensions. One approach is to construct individual meshes for each sub-domain and relate them afterwards, the alternative is to create one mesh for the whole domain and define sub-domain meshes as appropriate sub-meshes. In this paper we only discuss the first approach. In [6] Gander and Japhet describe a new algorithm that improves the complexity of matching unrelated meshes from $O(n^2)$ to $O(n)$, where n is the number of coupling elements. This algorithm is implemented in the DUNE GRID-GLUE [2] library. We discuss extensions of this library for handling distributed meshes.

When using methods like Dirichlet-Neumann coupling in the parallel context the user is forced to manage distributed data, as the necessary coupling information is not available locally. We present an abstraction that hides this

Institute for Computational und Applied Mathematics, University of Münster
`christian.engwer@wwu.de` · `steffen.muething@iwr.uni-heidelberg.de`

non-locality and allows the user to implement his Domain Decomposition strategy in a clear mathematical setting. By introducing two auxiliary Finite Element spaces on the coupling interface we can reformulate the original domain decomposition algorithm and hide all parallel data handling from the user. In a proof of concept we implement these auxiliary spaces for the DUNE PDELAB library, where they are created in a completely automatic fashion.

2 Relating unrelated Meshes

In the following we only describe a non-overlapping scenario, although the presented techniques are applicable to more general settings.

We consider a domain $\Omega \subset \mathbb{R}^d$. Ω is partitioned into two sub-domains Ω_0 and Ω_1 which meet at an interface Γ. The domains are triangulated into meshes $\mathcal{T}_0$ and $\mathcal{T}_1$ which are independent and in general do not match at the interface. Each mesh describes a set of entities, e.g. cells, faces, etc. We select a subset of entities which covers the interface Γ, i.e. the patches $\mathcal{P}_0$, $\mathcal{P}_1$; on these we impose the coupling conditions.

In order to relate information on Ω_0 and Ω_1 one has to transfer data like approximate solutions and evaluations of local residuals. We follow a mesh intersection approach, requiring us to compute the intersections of all entities in $\mathcal{P}_0$ with those in $\mathcal{P}_1$ (see Figure 1). Based on the algorithm presented in [6] we identify pairs of overlapping entities from both sides, for which we then compute entity clippings, yielding a set of polyhedral intersections.

This algorithm is available as `Dune::GridGlue::Merger` within DUNE GRID-GLUE; it is provided as a native implementation and as an interface to legacy codes. Using a predicate m_i the coupling patches are defined as $\mathcal{P}_i = \{\gamma | \gamma \in \mathcal{T}_i \cap \partial\Omega_i \wedge m_i(\gamma)\}$. The computed intersections are modelled as the intersections in the DUNE grid interface and exposed as `Dune::GridGlue::Intersection`, which provides topological and geometrical information. In the sequential case it gives access to the adjacent cells in the two grids $\mathcal{T}_0$ and $\mathcal{T}_1$. To compute coupling conditions, intersections provide a mapping from local coordinates to global coordinates as well as mappings to the local coordinate systems of the adjacent cells (see Figure 2).

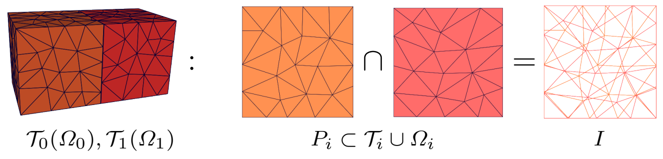
$\mathcal{T}_0(\Omega_0), \mathcal{T}_1(\Omega_1)$
:
$\cap$
$=$
$P_i \subset \mathcal{T}_i \cup \Omega_i$
I

Fig. 1 Intersecting the coupling patches $\mathcal{P}_0$ and $\mathcal{P}_1$ yields a set I of intersections, which can be used to evaluate the coupling conditions.

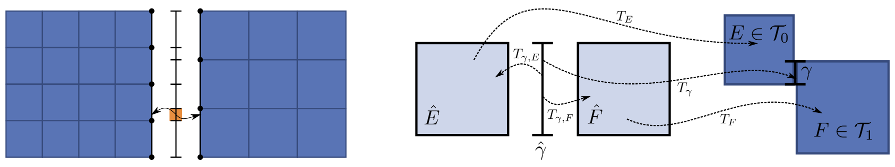

Fig. 2 Left: Intersections relate adjacent cells of unrelated grids. Right: Geometric mappings provided by an intersection.

2.1 Coupling via intersections

As a short example, let us consider a two-domain Poisson problem with Dirichlet-Neumann coupling condition: Find u_0 and u_1 such that

$$\begin{aligned} -\Delta u_i &= 0 && \text{on } \Omega_i,\ i \in 0,1 \\ u_i &= g && \text{at } \partial\Omega_i \setminus \Gamma,\ i \in 0,1 \\ u_0 &= u_1 && \text{at } \Gamma \\ \nabla u_1 \cdot \mathbf{n} &= \nabla u_0 \cdot \mathbf{n} && \text{at } \Gamma\ . \end{aligned} \tag{1}$$

We follow the usual approach and introduce discrete trial and test spaces V_0, V_1 on Ω_0 and Ω_1. In the simplest case this might be a conforming Lagrange discretization. Testing with functions $v_i \in V_i$ and integration by parts yields the problem in its weak formulation. On Ω_0 we impose Dirichlet boundary conditions along Γ, whereas Neumann boundary conditions are imposed along Γ on Ω_1. As the interface Γ is in general non-conforming, we can employ a Clément interpolation to interpolate the solution u_1 onto Ω_0. For given bases Φ_0, Φ_1, we obtain a system matrix of the following form, where C_0 and C_1 correspond to Dirchlet and Neumann coupling blocks:

$$\begin{pmatrix} A_0 & C_0 \\ C_1 & A_1 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \tag{2}$$

The matrix entries in the off-diagonal blocks are given by

$$C_0^{i,j} = -\langle \nabla\phi_0^i \mathbf{n}, \phi_1^j \rangle_\Gamma\ , \qquad C_1^{i,j} = -\omega_{\phi_0^j} \langle \phi_1^i, \phi_0^j \rangle_\Gamma\ ,$$

with $\omega_{\phi_0^j} = 1/\langle 1, \phi_0^j \rangle_\Gamma$ the weights of the Clément operator and $\phi_*^i \in \Phi_*$.

A straightforward approach to solving this problem iteratively is a fix point iteration on the split problem. In order to better illustrate the differences to the following parallel setting, we sketch this iteration in Algorithm 1.

Algorithm 1 Classic Dirichlet-Neumann iteration

```
$u^0, u^1$ = initial
**while** ! converged **do**
    $u_0 \leftarrow A_0^{-1}(b_0 - C_\sigma u_1)$
    $u_1 \leftarrow A_1^{-1}(b_1 - C_\sigma u_0)$
**end while**
```

Algorithm 2 Parallel grid matching algorithm

```
**parallel** GridGlue
    $P$                                                         ▷ $P$: # of parallel processes
    $\mathcal{T}(\Omega_0), \mathcal{T}(\Omega_1)$                                   ▷ Sub domain meshes
    $m_0, m_1$                                                  ▷ Predicates for $\Omega_1$ and $\Omega_2$
    **process** $\Pi$ $[p \in \{0, ..., P-1\}]$
        $\mathcal{P}_0 = \{\gamma | \gamma \in \mathcal{T}_0|_p \cap \partial\Omega_0 \wedge m_0(\gamma)\}$        ▷ Local coupling patches
        $\mathcal{P}_1 = \{\gamma | \gamma \in \mathcal{T}_1|_p \cap \partial\Omega_1 \wedge m_1(\gamma)\}$
        $I^p \leftarrow \mathrm{merge}(\mathcal{P}_0, \mathcal{P}_1)$                         ▷ Set of intersection
        $(\widehat{\mathcal{P}}_0, \widehat{\mathcal{P}}_1) \leftarrow (\mathcal{P}_0, \mathcal{P}_1)$
        **for** $i \in [0, P-2)$ **do**
            **asend**: $(\widehat{\mathcal{P}}_0, \widehat{\mathcal{P}}_1) \longrightarrow (p+1)\%P$               ▷ send to right neighbor
            **arecv**: $(\widehat{\mathcal{P}}_0, \widehat{\mathcal{P}}_1) \longrightarrow (p-1+P)\%P$           ▷ receive from left neighbor
            $I^p \leftarrow I^p \cup \mathrm{merge}(\widehat{\mathcal{P}}_0, \mathcal{P}_1)$                  ▷ merge remote patches
            $I^p \leftarrow I^p \cup \mathrm{merge}(\mathcal{P}_0, \widehat{\mathcal{P}}_1)$                  . . . with local patches
        **end for**
    **end process**
**end parallel**
```

2.2 Concepts of Parallel Mesh Coupling

Based on the previously introduced local grid matching algorithm we derive a parallel grid matching algorithm, see Algorithm 2. We extract the local part of the coupling patches $\mathcal{P}_0$, $\mathcal{P}_1$, merge these and communicate the data in a ring. We retrieve the neighboring patches and intersect them with our local patches. This yields the set of all intersections of local entities, either in Ω_0 or Ω_1, with any other entity, including remote entities. This provides all topological and geometric information required to evaluate the coupling conditions, but in general, as illustrated in Figure 3, we lack access to the data in the adjacent domain. We therefore assign a globally unique ID to each intersection to provide parallel communication on the interfaces. This communication is built upon the *parallel IndexSets* [4] of DUNE and allows a gather/scatter mechanism to send and receive data across domain intersection patches. In analogy to the parallel communication in the DUNE grid interface, the user has to provide a `DataHandle` object which implements the gather and scatter operations. The communicated data depends on the chosen Domain Decomposition method, thus the user is usually required to implement the data communication himself. For high level frameworks this a very unsatisfactory situation.

For methods like Mortar or FETI-DP the problems are less immanent as we have no direct coupling along the sub-domain faces. These methods introduce additional degrees of freedom on the interface, the sub-domains couple only to the interface and then the arising Schur-Complement system for the interface is solved.

Other methods like classic non-overlapping Schwarz methods or Dirichlet-Neumann coupling directly couple the sub-domains and require explicit communication of remote data. The main difference is that in the latter case we cannot fully represent the local part of the Poincaré-Steklov operator on a single processor, but only the local contributions.

3 Hiding Parallel Communication using Auxiliary Spaces

We now describe a mathematical abstraction which allows implementations to hide all communications from the user. We introduce additional function spaces V_λ and V_σ on the coupling interface Γ, see Figure 4. The definition of these function spaces is general; they can thus be constructed automatically as

$$V_\lambda = \left\{ v \in L^2(\Gamma) \,\middle|\, v|_\gamma \in P_k(\gamma), \gamma \in I, k = \mathrm{order}(V_0) \right\} \supseteq \mathrm{tr}(V_0)$$
$$V_\sigma = \left\{ v \in L^2(\Gamma) \,\middle|\, v|_\gamma \in P_k(\gamma), \gamma \in I, k = \mathrm{order}(V_1) \right\} \supseteq \mathrm{tr}(V_1) \ ,$$

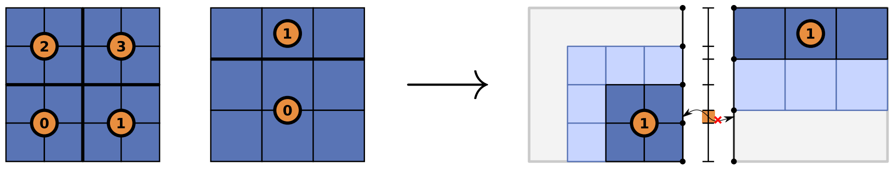

Fig. 3 When coupling distributed grids, neighboring cells of the remote mesh might not be accessible locally, making it impossible to evaluate coupling conditions. (Numbers in circles denote the process rank)

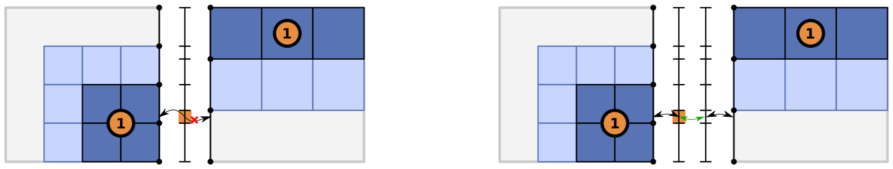

Fig. 4 Through the use of auxiliary spaces on the coupling interface Γ, direct access to non-local cells of the neighboring domain is avoided. (Numbers in circles denote the process rank)

Algorithm 3 Auxiliary space iterative algorithm

```
$u^0, u^1$ = initial
while ! converged do
    $\sigma \leftarrow D_\sigma u_1$                          ▷ implicit data communication
    $u_0 \leftarrow A_0^{-1}(b_0 - C_\sigma \sigma)$          ▷ parallel solver on $\Omega_0$
    $\lambda \leftarrow D_\lambda u_0$                        ▷ implicit data communication
    $u_1 \leftarrow A_1^{-1}(b_1 - C_\sigma \lambda)$         ▷ parallel solver on $\Omega_1$
end while
```

where P_k denotes the space of polynomial functions up to degree k. V_λ and V_σ are defined as discontinuous polynomial spaces on the interface, where V_λ is the minimal DG space containing the trace spaces of V_0 and V_σ for V_1, respectively. For efficiency we choose L^2 orthonormal bases. Note that for $\mathrm{order}(V_0) = \mathrm{order}(V_1)$ it follows that $V_\lambda = V_\sigma$. The arising structure of the global system is as follows, although it is never assembled as a whole:

$$\begin{pmatrix} A_0 & 0 & C_\sigma & 0 \\ -D_\lambda & M_\lambda & 0 & 0 \\ 0 & 0 & M_\sigma & -D_\sigma \\ 0 & C_\lambda & 0 & A_1 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ \lambda \\ \sigma \\ u_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ 0 \\ 0 \\ b_1 \end{pmatrix},$$

where M_λ, M_σ denote the mass matrices of V_λ, V_σ and C_λ, D_λ, C_σ, D_σ are coupling operators.

The auxiliary spaces V_λ and V_σ eliminate the direct coupling between A_0 and A_1. We split the original coupling operator C_1 to obtain the pair C_λ, D_λ and proceed analogously for C_0. As we have chosen L^2 orthonormal basis functions for V_λ and V_σ, the mass matrices reduce to the identity $\mathbb{1}$. Therefore the coupling operators can be evaluated on the fly in an efficient fashion. All computations are completely local and can be handled by a generic gather/scatter implementation. The relation between C_1 and C_λ, D_λ becomes obvious when eliminating λ or σ, respectively. We use $M_* = \mathbb{1}$ and obtain the classical coupled system as in (2)

$$\begin{pmatrix} A_0 & C_\sigma D_\sigma \\ C_\lambda D_\lambda & A_1 \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$$

In analogy to Algorithm 1, we can solve the coupled parallel system using Algorithm 3. As we recover the original DD method, it is also possible to use it as a preconditioner in existing Krylov methods.

3.1 Implementation in DUNE PDELab

When implementing the Poisson example from Section 2.1 with the auxiliary spaces approach, DUNE PDELAB transparently synthesizes the auxiliary spaces V_λ and V_σ and represents the overall solution space $V = V_0 \times V_1 \times V_\lambda \times V_\sigma$ as a tree of elementary function spaces (cf. Figure 5). Given a weak problem of the form $u \in U : a(u,v) = b(v)\ \forall\ v \in V$, Dune PDELab splits the (bi)linear forms into sums of entity-local contributions α_e, α_s and α_b for cells, interior facets and boundary facets, respectively, isolating the user from mesh and DOF handling. $a(u,v)$ thus reads

$$a(u,v) = \sum_{e\in E_h} R_e^E(\alpha_v, u, v) + \sum_{f\in F_h^{(i)}} R_f^F(\alpha_s, u, v) + \sum_{f\in F_h^{(b)}} R_b^B(\alpha_b, u, v). \quad (3)$$

R^E, R^F and R^B map the global spaces U and V to the element-local restrictions on the cells adjacent to the current entity, leaving the user with the task of implementing the local contributions α_e, α_s and α_b.

The coupling operators D_λ, C_λ, D_σ and C_σ resemble the interior facet terms in that they involve restricted function spaces with different supports, but differ in that the restrictions do not belong to the same global space. Those terms consequently require an extension of eq. (3) with additional coupling terms on the interface Γ and the two sub-domains.

D_λ and D_σ form projection operators onto V_λ and V_σ, whereas C_λ and C_σ mimic the operators C_0 and C_1. The first one behaves like a source on the interface, whereas the second one is a direct adoption of the Clément operator. Given local bases Φ_*^γ on γ (with $V_*|\gamma = \mathrm{span}(\Phi_*^\gamma)$) the user has to implement the following local contributions to the global stiffness matrix:

$$\alpha_\gamma^{0,\lambda}(\Phi_0^\gamma, \Phi_\lambda^\gamma) = \sum_{\substack{\phi_\lambda\in\Phi_\lambda^\gamma \\ \phi_0\in\Phi_0^\gamma}} -\langle\nabla\phi_0\mathbf{n}, \phi_\lambda\rangle_\gamma\ , \quad \alpha_\gamma^{\lambda,1}(\Phi_\lambda^\gamma, \Phi_1^\gamma) = \sum_{\substack{\phi_1\in\Phi_1^\gamma \\ \phi_\lambda\in\Phi_\lambda^\gamma}} \langle\phi_\lambda, \phi_1\rangle_\gamma\ ,$$

$$\alpha_\gamma^{1,\sigma}(\Phi_1^\gamma, \Phi_\sigma^\gamma) = \sum_{\substack{\phi_\sigma\in\Phi_\sigma^\gamma \\ \phi_1\in\Phi_1^\gamma}} -\langle\phi_1, \phi_\sigma\rangle_\gamma\ , \quad \alpha_\gamma^{\sigma,0}(\Phi_\sigma^\gamma, \Phi_0^\gamma) = \sum_{\substack{\phi_0\in\Phi_0^\gamma \\ \phi_\sigma\in\Phi_\sigma^\gamma}} -\omega_{\phi_0}\langle\phi_\sigma, \phi_0\rangle_\gamma\ ,$$

which correspond to D_λ, C_λ, D_σ and C_σ, respectively.

4 Conclusions

The Dune Grid-Glue library offers software infrastructure for the coupling of unrelated grids. We presented recent extensions to Dune Grid-Glue to

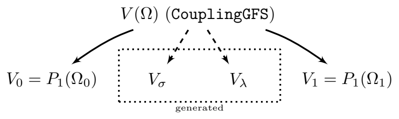

Fig. 5 Sketch of the hierarchic construction of the global function space for a coupled problem. Dune PDELab automatically generated the spaces V_λ and V_σ.

work in the context of distributed meshes. Reconstructed geometrical and topological relations between the grids are encapsulated as intersection objects. Although presented for non-overlapping intersections, the parallel implementation also handles overlapping and mixed-dimensional setups.

The coupling of distributed grids usually requires substantial changes to the user code and explicit use of parallel communication. We discussed a concept to reformulate the numerical scheme using auxiliary spaces on the coupling interface Γ, which allows the implementation of domain decomposition methods in a common framework that can hide the parallel communication from the user. This reformulated coupling problem integrates nicely with the hierarchic function space and operator concepts available in DUNE PDELAB.

The presented parallel mesh matching is available in the current version of the DUNE GRID-GLUE library. A prototype implementation for DUNE PDELAB is available, a more general implementation is under development. The code is available under an open source license from the DUNE website `http://dune-project.org/`.

Acknowledgements This work was supported by the German Research Foundation (DFG) through the Priority Programme 1648 'Software for Exascale Computing' (SPPEXA) and the DFG EXC 1003 Cells in Motion – Cluster of Excellence, Münster, Germany.

References

[1] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part II: Implementation and Tests in DUNE. *Computing*, 82(2–3):121–138, 2008.

[2] P. Bastian, G. Buse, and O. Sander. Infrastructure for the coupling of dune grids. In *Proceedings of ENUMATH 2009*, pages 107–114. Springer, 2010.

[3] C. Bernardi, Y. Maday, and A. T. Patera. Domain decomposition by the mortar element method. In *Asymptotic and numerical methods for partial differential equations with critical parameters*, pages 269–286. Springer, 1993.

[4] M. Blatt and P. Bastian. On the generic parallelisation of iterative solvers for the finite element method. *International Journal of Computational Science and Engineering*, 4(1):56–69, 2008.

[5] S. Deparis, D. Forti, and A. Quarteroni. A rescaled localized radial basis functions interpolation on non-cartesian and non-conforming grids. Technical Report 37.2013, EPFL, 2013.

[6] M. Gander and C. Japhet. Algorithm 932: PANG: Software for non-matching grid projections in 2d and 3d with linear complexity. *ACM Transactions on Mathematical Software*, 40(1):6:1–6:25, October 2013.

[7] W. Joppich and M. Kürschner. MpCCI – a tool for the simulation of coupled applications. *Concurrency and Computation: Practice and Experience*, 18(2):183–192, 2006.

[8] S. J. Plimpton, B. Hendrickson, and J. R. Stewart. A parallel rendezvous algorithm for interpolation between multiple grids. *Journal of Parallel and Distributed Computing*, 64(2):266–276, 2004.