Domain Decomposition with local refinement for flow simulation around a nuclear waste disposal : direct computation and simulation using code coupling with OCamlP3L.

François Clément* Vincent Martin† Arnaud Vodicka‡ Roberto Di Cosmo§ Pierre Weis¶

Abstract

We consider the application of a non-overlapping domain decomposition method with non-matching grids based on Robin interface conditions to the problem of flow surrounding an underground nuclear waste disposal. We show on a simple example how one can refine the mesh locally around the storage with this technique. A second aspect is studied in this paper. The coupling between the subdomains can be achieved informatically in two ways: either directly (i.e. the domain decomposition algorithm is included in the code that solves the problems on the subdomains) or using code coupling. In the latter case, each subdomain problem is solved separately and the coupling is performed by another program. We wrote a coupling program in the functionnal language OCaml, using the OCamlP3l environment devoted to ease the parallelism. Thus in the same time we test the code coupling and we use the natural parallel property of domain decomposition methods. Some simple 2D numerical tests show promising results, and further studies are under way.

Keywords: domain decomposition, local refinement, mixed finite elements, mortar finite element, coupling, parallelism, high-level language, Caml.

*Inria Rocquencourt, projet Estime, BP 105, 78153 Le Chesnay Cedex, France. `Francois.Clement@inria.fr`.

†Inria Rocquencourt and Andra, Parc de la Croix Blanche, 1-7 rue Jean Monnet, 92298 Chatenay-Malabry. `Vincent.Martin@inria.fr`.

‡Inria Rocquencourt, projet Estime. `Arnaud.Vodicka@inria.fr`.

§Inria Rocquencourt, projet Cristal et PPS, Université de Paris 7. `roberto@dicosmo.org`.

¶Inria Rocquencourt, projet Cristal. `Pierre.Weis@inria.fr`.

1 Introduction

We are concerned with the simulation of flow and transport in porous media. These problems have become more and more popular in the recent past because underground environmental issues such as water supplies management, pollution... have shown to have important consequences. In particular, in many industrial countries, the feasibility of an underground nuclear waste disposal is being studied [1]. The reliable simulation of flow and transport in porous media is crucial to try to forecast the behavior of contaminants in the geological layers. Most of these simulations should be computed in 3 dimensions on very long time scales (transport simulations over 10 million years), with very differents length scales (going from the meter to 20 or 30 kilometers), involving possibly different physics. They obviously require very efficient numerical methods and computational capabilities. A good way of treating this kind of large problems is to divide them into smaller subproblems, and then to couple them. We use a domain decomposition method to divide the problem in a proper manner and the coupling is obtained through two different ways: directly or via a specific program that performs the code coupling. Our ultimate goal is to compare the two couplings in terms of efficiency in solving the global problem, and in terms of time required to develop the programs.

We will restrict ourselves in this paper to the efficient simulation of stationary flow. Two different but linked aspects are studied. First, in the Section 2, we present in full details a non-overlapping domain decomposition method that allows the treatment of non-matching grids. We used a mixed finite element method to solve the subdomain problems. This method is based on Robin interface conditions and was studied in [3] in the mixed finite element case and in [1] in the finite volume case. It is somewhat different from the mortar method first introduced in [4]. This kind of domain decomposition method is particularly practical as it allows a separate meshing of the subdomains. This is useful to the underground storage simulations as it allows a local refinement around the storage: we create a subdomain that consists of a rectangular box around the storage, with a regular and fine mesh. The rest of the domain can be meshed with a coarser mesh, following the geological layers. We illustrate this on a very simple 2D example. Some more complex 3D examples are being studied.

The second aspect that we focus on is the coupling between the subdomains. The first natural way to link the subdomains is to program directly the algorithm in the code. This is what we did in the first experiment in Section 3, where the programs are written exclusively in `C++`. The other way that shows promising developments is to couple the subdomain solver by using a specific coupling code. The final goal is to couple different codes that are already available and that can solve efficiently their own problem: for example, one code solves the transport equations and another one treats the chemistry... The first step of this long term project that we present in Section 4 is the coupling of a code with itself in a domain decomposition method. The coupling program is written in the functional language OCaml with some OCamlP3L skeletons (see `http://www.ocaml.org/` and `http://www.ocamlp3l.org/` or [9]). OCamlP3L is a parallel functional programming environment with skeletons. A few number of these skeletons (we used 4 of them in our program) are used to describe all the (parallel) structure of the algorithm. The code can be compiled and tested sequentially and then a simple recompilation without any change in the coupling program provides parallel run. Our first tests were performed on a simple 2D problem, with a simple Schwarz domain

[1] In France, ANDRA (the French National Agency for the Management of Nuclear Wastes) is in charge of supervizing these studies. See `http://www.andra.fr`.

decomposition algorithm (see [10]) and give good results with a particularly short coupling program (about 170 lines) (see Section 4.4). Some more realistic 3D experiments using the domain decomposition method with non-matching grids described in Section 2 are under way.

2 Description of the multiblock problem

Let Ω be a convex domain in $\mathbb{R}^d$, $d = 2$ or 3, and we denote by $\Gamma = \partial\Omega$ the boundary of Ω. We suppose that the flow in Ω is governed by a conservation equation together with Darcy's law relating the gradient of the pressure p to the Darcy velocity $\mathbf{u}$:

$$\begin{array}{rcll} \operatorname{div}\mathbf{u} & = & q & \text{in } \Omega \\ \mathbf{u} & = & -\mathbf{K}\boldsymbol{\nabla} p & \text{in } \Omega \\ p & = & \overline{p} & \text{on } \Gamma, \end{array} \tag{1}$$

where p is the pressure, $\mathbf{u}$ the Darcy velocity, $\mathbf{K}$ the hydraulic conductivity (or permeability) tensor, q a source term and $\overline{p}$ the given pressure on the boundary Γ. We suppose that $\mathbf{K}$ is a symmetric, positive definite and uniformly bounded tensor.

Let Ω be decomposed into n non-overlapping subdomains, also called blocks, $\Omega_i, i \in I = \{1, 2, \ldots, n\}$, and we denote by Γ_i the part of the boundary of Ω_i, $i \in I$ in common with the boundary of Ω. Hence,

$$\bar{\Omega} = \bigcup_{i\in I} \bar{\Omega}_i, \quad \text{and} \quad \Gamma_i = \partial\Omega_i \cap \Gamma, \quad i \in I = \{1, 2, \ldots, n\},$$

where $\bar{\Omega}_i$ is the closure of the open set Ω_i. Let Σ_i be the geometric *internal boundary* of Ω_i, i.e. the interior of the internal boundary of Ω_i, and Σ the union of all internal boundaries.

$$\Sigma_i = \overset{\circ}{\overline{(\partial\Omega_i \backslash \partial\Omega)}}, \quad i \in I, \qquad \Sigma = \bigcup_{i\in I} \Sigma_i.$$

Then we can define the geometric *interface between subdomains* Ω_i and Ω_j as

$$\Sigma_{ij} = \Sigma_{ji} = \Sigma_i \cap \Sigma_j, \qquad \Sigma_{ii} = \emptyset, \quad i, j \in I.$$

If $\Sigma_{ij} \neq \emptyset$, the subdomains Ω_i and Ω_j, $i \neq j \in I$, are called neighbors.

Let ν_i denote the external unit normal to $\Omega_i, i \in I$. We denote the $L^2(\Omega_i)$ or $(L^2(\Omega_i))^d$ the inner product by $(\cdot,\cdot)_i$. In these notations, we omit the subscript i to refer to the entire domain Ω. Let the interface inner product on $L^2(\Sigma_i)$ be denoted by $\langle\cdot,\cdot\rangle_i$ and on $L^2(\Sigma_{ij})$ by $\langle\cdot,\cdot\rangle_{ij}$. Note that if $\Sigma_{ij} = \emptyset$, this inner product is zero.

We denote by $p_i, \mathbf{u}_i, \mathbf{K}_i$, and q_i the restrictions of $p, \mathbf{u}, \mathbf{K}$, and q respectively to Ω_i, $i \in I$, and by $\overline{p}_i$ the restriction of $\overline{p}$ to Γ_i, $i \in I$. We can rewrite the above problem (1) as a transmission or multiblock problem:

$$\begin{array}{rcll} \operatorname{div}\mathbf{u}_i & = & q_i & \text{in } \Omega_i, \quad i \in I, \\ \mathbf{u}_i & = & -\mathbf{K}_i\boldsymbol{\nabla} p_i & \text{in } \Omega_i, \quad i \in I, \\ p_i & = & \overline{p}_i & \text{on } \Gamma_i, \quad i \in I, \end{array} \tag{2}$$

together with transmission conditions

$$\begin{array}{rcll} p_i & = & p_j & \text{on } \Sigma_{ij}, \quad i, j \in I, \\ \mathbf{u_i} \cdot \nu_\mathbf{i} & = & -\mathbf{u_j} \cdot \nu_\mathbf{j} & \text{on } \Sigma_{ij}, \quad i, j \in I. \end{array} \tag{3}$$

The two equations in (3) yield the continuity of pressure and of normal velocity (as $\nu_i = -\nu_j$) across each interface $\Sigma_{ij} (\neq \emptyset)$ between two neighboring subdomains Ω_i and Ω_j, $i, j \in I$. These interface equations can be rewritten with Robin conditions.

Let $\alpha_{ij} > 0$ and $\alpha_{ji} > 0$ be the Robin coefficients on the interface $\Sigma_{ij} = \Sigma_{ji}$, $i, j \in I$. First, multiply the first equation by α_{ij} and substract the second equation. Second, multiply the first equation by α_{ji} and add the second equation. Thus the two equations in (3) are equivalent to the Robin interface conditions

$$\begin{array}{rcll} -\mathbf{u_i} \cdot \nu_\mathbf{i} + \alpha_{ij} p_i & = & \mathbf{u_j} \cdot \nu_\mathbf{j} + \alpha_{ij} p_j & \text{on } \Sigma_{ij}, \quad i, j \in I, \\ -\mathbf{u_j} \cdot \nu_\mathbf{j} + \alpha_{ji} p_j & = & \mathbf{u_i} \cdot \nu_\mathbf{i} + \alpha_{ji} p_i & \text{on } \Sigma_{ij}, \quad i, j \in I. \end{array} \tag{4}$$

Clearly the solution to (1) satisfies (2) and (3) or equivalently (2) and (4). Conversely, by elliptic regularity, the solution to (2) and (4) satisfies (1), so the two formulations are equivalent.

2.1 Weak multiblock problem

Throughout the rest of this section, for simplicity, we will assume that homogeneous Dirichlet boundary conditions are imposed on $\partial\Omega$: $\overline{p} = 0$ on Γ.

Let us define the approximation spaces used in this paper.

$$\begin{array}{lll} \mathbf{Z_i} = \mathcal{H}(\mathrm{div}, \Omega_i), & i \in I. & \mathbf{Z} = \bigoplus_{i \in I} \mathbf{Z_i} \not\subset \mathcal{H}(\mathrm{div}, \Omega). \\ N_i = L^2(\Omega_i), & i \in I. & N = \bigoplus_{i \in I} N_i = L^2(\Omega). \\ \Lambda_i = L^2(\Sigma_i), & i \in I. & \Lambda = \bigoplus_{i \in I} \Lambda_i. \end{array} \tag{5}$$

With these notations, we can write the mixed formulation. A weak solution of (2)-(4) is a couple $(\mathbf{u}, p) \in \mathbf{Z} \times N$ such that for each $i \in I$, $p_i|_{\Sigma_i} \in \Lambda_i$, and $\mathbf{u_i} \cdot \nu_\mathbf{i}|_{\Sigma_i} \in \Lambda_i$, and

$$\begin{array}{rcll} \left(K^{-1}\mathbf{u}, \mathbf{v}\right)_i - (\mathrm{div}\, \mathbf{v}, p)_i & = & -\left\langle \mathbf{v} \cdot \nu_\mathbf{i}, p_i \right\rangle_i, & \mathbf{v} \in \mathbf{Z_i}, \\ (\mathrm{div}\, \mathbf{u}, r)_i & = & (q_i, r)_i, & r \in N_i, \\ \sum_{j \in I} \left\langle -\mathbf{u_i} \cdot \nu_\mathbf{i} + \alpha_{ij} p_i, \mu_i \right\rangle_{ij} & = & \sum_{j \in I} \left\langle \mathbf{u_j} \cdot \nu_\mathbf{j} + \alpha_{ij} p_j, \mu_i \right\rangle_{ij}, & \mu_i \in \Lambda_i. \end{array} \tag{6}$$

2.2 Meshes on the subdomains

Let $\mathcal{T}_i^h$ be a conforming finite element partition of the subdomain Ω_i, $i \in I$, where the maximal element diameter is bounded by the real $h > 0$, and the interior of each element face (or edge in 2D) are entirely contained in Σ_i or in $\Gamma_i = \partial\Omega_i \cap \partial\Omega$. The meshes of two neighboring subdomains need not match at their interface Σ_{ij}.

Let us now define more precisely the interface meshes. We have seen that two subdomains are called *neighbors* if their interface $\Sigma_{ij} \neq \emptyset$. For $i \in I$, we denote the number of neighbors of a subdomain Ω_i by n_i, and we denote the set of indices of these neighbors by $\mathcal{N}_i = \{j_1, j_2, \ldots, j_{n_i}\}$. The sets I and $\mathcal{N}_i$ inherit the natural order on $\mathbb{N}$. Thus $k \in 1, 2, \ldots, n_i \mapsto j_k \in \mathcal{N}_i$ is increasing. For an $i \in I$ and a $j \in \mathcal{N}_i$, the trace on Σ_{ij} of the mesh $\mathcal{T}_i^h$ of Ω_i is denoted by $\mathcal{S}_{ij}^h$. In the same way, for a $j \in I$ and an $i \in \mathcal{N}_j$, the trace on $\Sigma_{ji} = \Sigma_{ij}$ of the

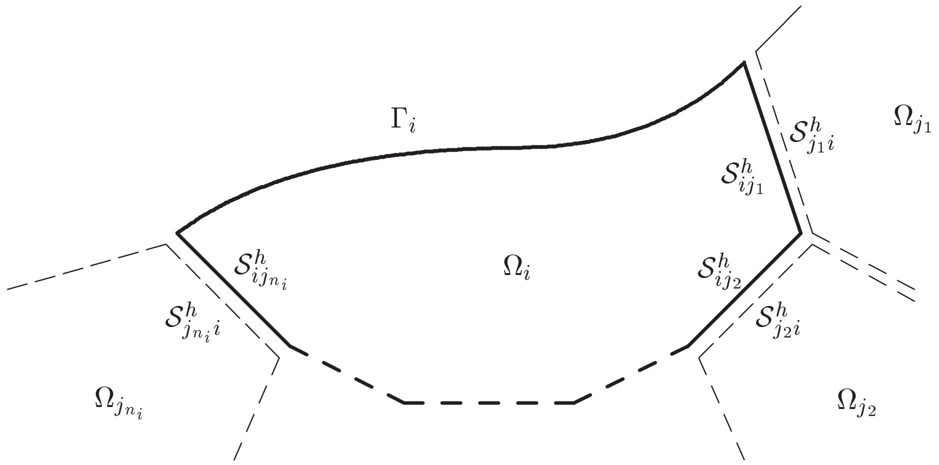

Figure 1: Subdomain Ω_i, its neighbors and the interface meshes.

mesh $\mathcal{T}_j^h$ is denoted by $\mathcal{S}^h_{ji}$. In general $\mathcal{S}^h_{ji} \neq \mathcal{S}^h_{ij}, i \neq j$, as the meshes $\mathcal{T}_i^h$ and $\mathcal{T}_j^h$ do not match. (See Figure 1)

We denote by $\mathcal{S}_i^h$ the trace on Σ_i of the mesh $\mathcal{T}_i^h$ of Ω_i. It is also the union of all the interface meshes coming from the subdomain Ω_i. We call the mesh $\mathcal{S}_i^h$ the *inner interface mesh* of Ω_i.

$$\forall i \in I,\ \forall j \in \mathcal{N}_i \quad \mathcal{S}^h_{ij} = \mathcal{T}_i^h|_{\Sigma_{ij}}. \qquad \mathcal{S}_i^h = \biguplus_{j \in \mathcal{N}_i} \mathcal{S}^h_{ij}.$$

The set of the ordered couples of neighbors is denoted by $\mathcal{N} = \{(i,j), i \in I, j \in \mathcal{N}_i\}$. It is ordered according to lexicographic order on $\mathbb{N}^2$. We denote by s the permutation

$$\forall (i,j) \in \mathcal{N}, \quad s(i,j) = (j,i) \in \mathcal{N}.$$

We call *inner interface structure* $\mathcal{S}^h$ of the domain Ω the union of all inner interface meshes (each geometric interface is thus counted twice). It is ordered by $\mathcal{N}$.

$$\mathcal{S}^h = \biguplus_{i \in I} \mathcal{S}_i^h = \biguplus_{i \in I} \biguplus_{j \in \mathcal{N}_i} \mathcal{S}^h_{ij} = \biguplus_{(i,j) \in \mathcal{N}} \mathcal{S}^h_{ij}.$$

Up to now, we have defined the interface meshes that were seen from a subdomain Ω_i. Now, we define the interface meshes of the neighbors of Ω_i. For $(i,j) \in \mathcal{N}$, we denote $\tilde{\mathcal{S}}^h_{ij} = \mathcal{S}^h_{ji}$. We denote by $\tilde{\mathcal{S}}_i^h$ the *outer interface mesh* of the subdomain Ω_i that is the union of the interface meshes that are the traces of the neighbors of Ω_i. It is also ordered by $\mathcal{N}_i$.

$$\tilde{\mathcal{S}}_i^h = \biguplus_{j \in \mathcal{N}_i} \tilde{\mathcal{S}}^h_{ij} = \biguplus_{j \in \mathcal{N}_i} \mathcal{S}^h_{ji}.$$

In the same way, we denote by $\tilde{\mathcal{S}}^h$ the *outer interface structure* of the domain Ω that is the union of all outer interface meshes. It is also the union of all interface meshes, seen from a subdomain or its neighbor, but this time it is ordered by $s(\mathcal{N})$,

$$\tilde{\mathcal{S}}^h = \biguplus_{i\in I} \tilde{\mathcal{S}}_i^h = \biguplus_{i\in I}\biguplus_{j\in\mathcal{N}_i} \tilde{\mathcal{S}}_{ij}^h = \biguplus_{i\in I}\biguplus_{j\in\mathcal{N}_i} \mathcal{S}_{ji}^h = \biguplus_{(i,j)\in s(\mathcal{N})} \mathcal{S}_{ij}^h.$$

2.3 Discrete weak multiblock problem

Let

$$\mathbf{Z_{h,i}} \times N_{h,i} \times \Lambda_{h,i} \subset \mathbf{Z_i} \times N_i \times \Lambda_i, \quad i \in I,$$

be the usual mixed finite element approximation made of Raviart-Thomas (and Nedelec in 3D) spaces of lowest order ([13], [14], [11]), with the hybrid Lagrange multiplier spaces $\Lambda_{h,i}$ on Σ_i, [5]. We recall that $Z_{h,i}$ consists of some linear vector functions and that $N_{h,i}$ (respectively $\Lambda_{h,i}$) are made up of cellwise (resp. facewise in 3D or edgewise in 2D) constant scalar functions. Let

$$\mathbf{Z_h} = \bigoplus_{i\in I} \mathbf{Z_{h,i}}, \qquad N_h = \bigoplus_{i\in I} N_{h,i}, \qquad \Lambda_h = \bigoplus_{i\in I} \Lambda_{h,i}.$$

We also recall that

$$\mathrm{div}\mathbf{Z_{h,i}} = N_{h,i} \qquad \Lambda_{h,i} = \mathbf{Z_{h,i}} \cdot \nu_{\mathbf{i}}|_{\Sigma_i} = P_0(\mathcal{S}_i^h), \quad i \in I,$$

where $P_0(\mathcal{S}_i^h)$ denotes the space of facewise (or edgewise in 2D) constant functions on $\mathcal{S}_{ij}^h$.

One can notice that the space Λ_h contains two interface unknows on each interface $\Sigma_{ij}, i \in I, j \in \mathcal{N}_i$: the one that lives in $P_0(\mathcal{S}_{ij}^h)$, and the other one that lives in $P_0(\mathcal{S}_{ji}^h) = P_0(\tilde{\mathcal{S}}_{ij}^h)$. As we did in the previous section for the interface meshes, we introduce on the geometric interface between two subdomains Σ_{ij} two different approximation spaces $\Lambda_{h,ij}$ and $\tilde{\Lambda}_{h,ij} = \Lambda_{h,ji}$.

$$\begin{array}{rcl} \Lambda_{h,ij} &=& \mathbf{Z_{h,i}} \cdot \nu_{\mathbf{i}}|_{\Sigma_{ij}} = P_0(\mathcal{S}_{ij}^h), \quad i \in I,\ j \in \mathcal{N}_i \quad (\text{i.e. } (i,j) \in \mathcal{N}), \\ \tilde{\Lambda}_{h,ij} = \Lambda_{h,ji} &=& \mathbf{Z_{h,j}} \cdot \nu_{\mathbf{j}}|_{\Sigma_{ij}} = P_0(\mathcal{S}_{ji}^h), \quad j \in I,\ i \in \mathcal{N}_j. \end{array} \tag{7}$$

Next we define the space $\tilde{\Lambda}_{h,i}$ that contains the functions on the interface meshes $\tilde{\mathcal{S}}_i^h$ of the neighbors of $\Omega_i,\ i \in I$. This yields

$$\begin{array}{rcll} \Lambda_{h,i} &=& \bigoplus_{j\in\mathcal{N}_i} \Lambda_{h,ij} & = \mathbf{Z_{h,i}} \cdot \nu_{\mathbf{i}}|_{\Sigma_i} = P_0(\mathcal{S}_i^h), \qquad i \in I, \\ \tilde{\Lambda}_{h,i} &=& \bigoplus_{j\in\mathcal{N}_i} \tilde{\Lambda}_{h,ij} = \bigoplus_{j\in\mathcal{N}_i} \Lambda_{h,ji} & = \mathbf{Z_{h,j}} \cdot \nu_{\mathbf{j}}|_{\Sigma_i} = P_0(\tilde{\mathcal{S}}_i^h), \qquad i \in I, \end{array} \tag{8}$$

and

$$\begin{array}{rcl} \Lambda_h &=& \bigoplus_{i\in I} \Lambda_{h,i} = \bigoplus_{i\in I}\bigoplus_{j\in\mathcal{N}_i} \Lambda_{h,ij} = \bigoplus_{(i,j)\in\mathcal{N}} \Lambda_{h,ji}, \\ \tilde{\Lambda}_h &=& \bigoplus_{i\in I} \tilde{\Lambda}_{h,i} = \bigoplus_{i\in I}\bigoplus_{j\in\mathcal{N}_i} \tilde{\Lambda}_{h,ij} = \bigoplus_{i\in I}\bigoplus_{j\in\mathcal{N}_i} \Lambda_{h,ji} = \bigoplus_{(i,j)\in s(\mathcal{N})} \Lambda_{h,ji}. \end{array} \tag{9}$$

We write the discrete mixed finite element approximation for the original problem (1) or, equivalently, for the transmission problem (6) introducing interelement multipliers $\lambda_{h,i}$ (see [8], [2] and references therein).

We seek $(\mathbf{u_h}, p_h, \lambda_h) \in \mathbf{Z_h} \times N_h \times \Lambda_h$ such that

$$\begin{array}{rcll} \left(K^{-1}\mathbf{u_h},\, \mathbf{v}\right)_i - (\operatorname{div} \mathbf{v},\, p_h)_i &=& -\left\langle \mathbf{v}\cdot\nu_\mathbf{i},\, \lambda_{h,i}\right\rangle_i, & \mathbf{v} \in \mathbf{Z_{h,i}}, \\ (\operatorname{div} \mathbf{u_h},\, r)_i &=& (q_i,\, r)_i, & r \in N_{h,i}, \\ \sum\limits_{j\in\mathcal{N}_i} \left\langle -\mathbf{u_{h,i}}\cdot\nu_\mathbf{i} + \alpha_{ij}\lambda_{h,i},\, \mu_i\right\rangle_{ij} &=& \sum\limits_{j\in\mathcal{N}_i} \left\langle \mathbf{u_{h,j}}\cdot\nu_\mathbf{j} + \alpha_{ij}\lambda_{h,j},\, \mu_i\right\rangle_{ij}, & \mu_i \in \Lambda_{h,i}. \end{array} \tag{10}$$

It is proved that this problem (10) is well-posed in [3] and in [1] for the finite volume method. It is pointed out in [3] that local mass conservation is insured everywhere except on the interfaces. There is continuity of the normal velocity $\mathbf{u}\cdot\nu$ across every element faces (or edges in 2D) that does *not lie on* Σ. This is the price to pay the non-matching meshes. The flux is not conserved across any portion of Σ, but since $\mu_i = 1 \in \Lambda_{h,i}$, we continue to conserve the mass globally.

A simple method to solve this problem (10) is to solve iteratively the fixed point problem (11). Given a $x_h^0 \in \Lambda_h$, where the superscript is used for the iteration $k \geq 0$ in the method, solve iteratively

Compute $x_{h,i}^{k+1}|_{\Sigma_{ij}} = \mathbf{u_{h,i}^{k+1}}\cdot\nu_\mathbf{i} + \alpha_{ij}\lambda_{h,i}^{k+1} \quad j \in \mathcal{N}_i$, such that
$(\mathbf{u_{h,i}^{k+1}}, p_{h,i}^{k+1}, \lambda_{h,i}^{k+1}) \in \mathbf{Z_{h,i}} \times N_{h,i} \times \Lambda_{h,i}$ is the solution of

$$\begin{array}{rcll} \left(K^{-1}\mathbf{u_h^{k+1}},\, \mathbf{v}\right)_i - \left(\operatorname{div} \mathbf{v},\, p_h^{k+1}\right)_i &=& -\left\langle \mathbf{v}\cdot\nu_\mathbf{i},\, \lambda_{h,i}^{k+1}\right\rangle_i, & \mathbf{v} \in \mathbf{Z_{h,i}}, \\ \left(\operatorname{div} \mathbf{u_h^{k+1}},\, r\right)_i &=& (q_i,\, r)_i, & r \in N_{h,i}, \\ \sum\limits_{j\in\mathcal{N}_i} \left\langle -\mathbf{u_{h,i}^{k+1}}\cdot\nu_\mathbf{i} + \alpha_{ij}\lambda_{h,i}^{k+1},\, \mu_i\right\rangle_{ij} &=& \sum\limits_{j\in\mathcal{N}_i} \left\langle x_{h,i}^k,\, \mu_i\right\rangle_{ij}, & \mu_i \in \Lambda_{h,i}. \end{array} \tag{11}$$

When the method has converged, the Robin interface conditions (4) are imposed weakly, and thus the solution of (10) is reached. It is proved in [1] that this fixed point method converges. One can notice that it is non-symmetric eventhough the Laplace operator is symmetric. It is advisable to accelerate the convergence with a non-symmetric Krylov method, such as a BiCGStab or a GMRes method.

2.4 Interface operators

In order to clearly define the fixed point problem, we introduce discrete Robin to Robin operators S_i in each subdomain Ω_i defined by (12).

$$\begin{array}{rcll} S_i : \Lambda_{h,i} \times N_{h,i} &\longrightarrow& \Lambda_{h,i}, & i \in I \\ S_i(x_{h,i}, q_i)|_{\Sigma_{ij}} &=& \mathbf{u_{h,i}}\cdot\nu_\mathbf{i} + \alpha_{ij}p_{h,i}, & i \in I, \quad \forall j \in \mathcal{N}_i, \end{array} \tag{12}$$

such that $\mathbf{u_{h,i}}(x_h), p_{h,i}(x_h)$ is the solution of the following problem (13) set on $\Omega_i,\ i \in I$.

We seek $(\mathbf{u_{h,i}}, p_{h,i}, \lambda_{h,i}) \in \mathbf{Z_{h,i}} \times N_{h,i} \times \Lambda_{h,i}$ such that

$$\begin{array}{rcll} \left(K^{-1}\mathbf{u_{h,i}},\, \mathbf{v}\right)_i - (\operatorname{div} \mathbf{v},\, p_{h,i})_i &=& -\left\langle \mathbf{v}\cdot\nu_\mathbf{i},\, \lambda_{h,i}\right\rangle_i, & \mathbf{v} \in \mathbf{Z_{h,i}}, \\ (\operatorname{div} \mathbf{u_{h,i}},\, r)_i &=& (q_i,\, r)_i, & r \in N_{h,i}, \\ \sum\limits_{j\in\mathcal{N}_i} \left\langle -\mathbf{u_{h,i}}\cdot\nu_\mathbf{i} + \alpha_{ij}\lambda_{h,i},\, \mu_i\right\rangle_{ij} &=& \sum\limits_{j\in\mathcal{N}_i} \left\langle x_{h,ij},\, \mu_i\right\rangle_{ij}, & \mu_i \in \Lambda_{h,i}. \end{array} \tag{13}$$

For a pair of neighbors $(\Omega_i, \Omega_j),\ (i,j) \in \mathcal{N}$, we define a projection operator and a restriction operator in (14) and in (15) that are defined on the interface between the subdomains.

For a subdomain Ω_i $i \in I$, we define a restriction operator and a projection operator in (16) and in (18) that are defined on the internal boundary of the subdomain.

Let $P_{i\to j}$ be the L^2-projection operator onto $\tilde{\Lambda}_{h,ij} = \Lambda_{h,ji}$. With this operator one can project the $\lambda_{ij} \in \Lambda_{h,ij}$ onto $\tilde{\Lambda}_{h,ij}$.

$$\begin{array}{rcll} P_{i\to j} : L^2(\Sigma_{ij}) & \longrightarrow & \tilde{\Lambda}_{h,ij}, & \\ x_{ij} & \longmapsto & P_{i\to j} x_{ij} & \text{such that} \\ \langle x_{ij} - P_{i\to j} x_{ij}, \tilde{\mu}_{ij} \rangle_{ij} & = & 0, & \tilde{\mu}_{ij} \in \tilde{\Lambda}_{h,ij}. \end{array} \tag{14}$$

Define also the restriction operator from $\Lambda_{h,i}$ onto $\Lambda_{h,ij}$

$$\begin{array}{rcl} R_{ij} : \Lambda_{h,i} & \longrightarrow & \Lambda_{h,ij}, \\ \lambda_i = \left(\lambda_{ij_1}, \lambda_{ij_2}, \dots, \lambda_{ij_{n_i}}\right)^\top \text{ on } \mathcal{S}_i^h & \longmapsto & \lambda_{ij} \text{ on } \mathcal{S}_{ij}^h. \end{array} \tag{15}$$

For a $i \in I$ we also define two restriction operators: one from Λ_h onto $\Lambda_{h,i}$, the other one from $\tilde{\Lambda}_h$ onto $\tilde{\Lambda}_{h,i}$.

$$\begin{array}{rcl} R_i : \Lambda_h & \longrightarrow & \Lambda_{h,i}, \\ \lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)^\top \text{ on } \mathcal{S}^h & \longmapsto & \lambda_i \text{ on } \mathcal{S}_i^h. \\ \tilde{R}_i : \tilde{\Lambda}_h & \longrightarrow & \tilde{\Lambda}_{h,i}, \\ \tilde{\lambda} = \left(\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_n\right)^\top \text{ on } \tilde{\mathcal{S}}^h & \longmapsto & \tilde{\lambda}_i \text{ on } \tilde{\mathcal{S}}_i^h. \end{array} \tag{16}$$

By an abuse of notation, we use s as a permutation from Λ_h onto $\tilde{\Lambda}_h$:

$$\begin{array}{rcl} s : \Lambda_h & \longrightarrow & \tilde{\Lambda}_h, \\ s(\lambda_h) & = & \tilde{\lambda}_h. \end{array} \tag{17}$$

Define also, for a $i \in I$, the projection operator from $L^2(\Sigma_i)$ onto $\tilde{\Lambda}_{h,i}$. With this operator, one can project $\Lambda_{h,i}$ that lives on the *inner* boundary interface onto the *outer* boundary interface $\tilde{\Lambda}_{h,i}$.

$$\begin{array}{rcl} P_i : L^2(\Sigma_i) & \longrightarrow & \tilde{\Lambda}_{h,i}, \\ x_i = \left(x_{ij_1}, x_{ij_2}, \dots, x_{ij_{n_i}}\right)^\top & \longmapsto & \left(P_{i\to j_1} x_{ij_1}, P_{i\to j_2} x_{ij_2}, \dots, P_{i\to j_n} x_{ij_n}\right)^\top. \end{array} \tag{18}$$

In all these notations, to solve the problem (10), it suffices to find $\lambda_h \in \Lambda_h$ solution of the fixed point problem (19). The matrix-free matrix vector product that is necessary in a Krylov method (such as BiCGStab) can be easily implemented in this form (19).

$$\mathrm{Id}_{\Lambda_h} \lambda_h - s\left(\sum_{i\in I} \tilde{R}_i^\top P_i S_i(R_i \lambda_h, 0)\right) = s\left(\sum_{i\in I} \tilde{R}_i^\top P_i S_i(0, q|_{\Omega_i})\right) \quad \text{in } \Lambda_h. \tag{19}$$

3 Numerical tests of the domain decomposition

The first numerical results are performed to test the non conforming domain decomposition method defined in equation (19). Thus they are particularly simple. There are only 2 subdomains in 2D. A quadrature rule is used in the mixed finite element method in each subdomain to eliminate the velocity unknows, thus yielding the usual finite volume method (see [6]). The Krylov method used was BiCGStab. The stopping criterion to check the convergence is $\sup_{(i,j)\in\mathcal{N}} \|(\mathbf{u_i} \cdot \nu_\mathbf{i} + \alpha_{ij} p_i)|_{\mathcal{S}_{ij}^h}\|_\infty \leq 10^{-8}$.

3.1 An academic test case

The domain consists of the unit square $\Omega = [0,1]^2$. The first subdomain $\Omega_1 = [0.6, 0.8]^2$ is embedded in the second $\Omega_2 = \Omega \backslash \Omega_1$. The source term q and the Dirichlet boundary conditions $\overline{p}$ are chosen such that the exact solution is $p(x,y) = x^3y^2 + sin(xy)$.

The discrete L^∞ and H^1 norms of the error are shown for a decomposition into 2 subdomains with refined grids.

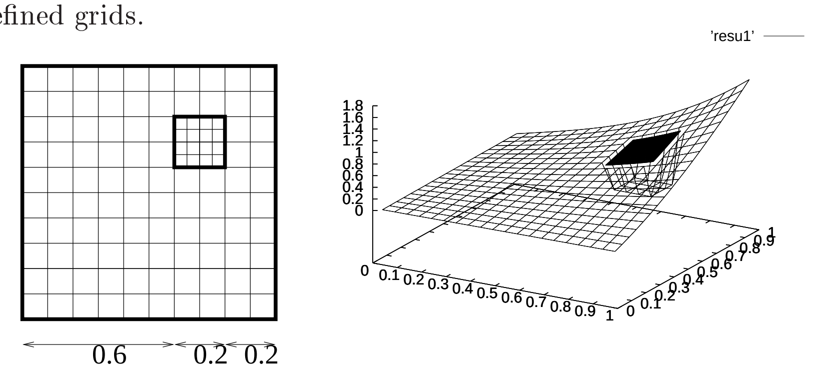

Figure 2: Right: example of the **nested** meshes of the 2 subdomains in the academic test case. Grids (10×10) in Ω_2 and (4×4) in Ω_1. Left: cell centered pressure in the case 2.

The test cases 1 and 2 are performed with regular meshes, with a local refinement in the internal subdomain. The fine grid is a sub-mesh of the coarse grid. The meshes are thus *nested*. It is not the case in the test case 3. In the case 1, the mesh is two times finer in the internal subdomain than in the rest of the domain. In the second test case, the mesh is 10 times finer in the refined zone. Three cases are considered. In each case, we give the number of cells along the Ox and Oy axes. An initial grid is chosen, then it is refined successively by dividing a cell into 4 cells, i.e. by dividing the mesh size h by a factor 2.

Case 1 Grids: $(5 \times 5) * 2^n$ in $\Omega_2; (2 \times 2) * 2^n$ in $\Omega_1, n = 0, 1, 2, 3, 4, 5$. See an example for $n = 1$ in Figure 2. The grids are nested and the meshes are regular.

Case 2 Grids: $(5 \times 5) * 2^n$ in $\Omega_2; (10 \times 10) * 2^n$ in $\Omega_1, n = 0, 1, 2, 3, 4$. The grids are nested and the meshes are regular.

Case 3 Grids: $(5 \times 5) * 2^n$ in $\Omega_2; (2 \times 2) * (2^n + 1)$ in $\Omega_1, n = 0, 1, 2, 3, 4, 5$. The grids are *not* nested but the meshes remain regular.

3.1.1 Results

The solution of the test case 2 is depicted in the Figure 2. It is the superposition of the pressure given by the two subdomains. The number of cells in Ω_2 is 20×20 and in Ω_1 : 40×40. The solution in the internal subdomain Ω_1 appears black as it is too fine for the picture. Also for picture purposes, we added some zeros to the solution of subdomain Ω_2 in the zone of Ω_1 in order to plot the solution on a regular square mesh. This is why one can notice some unphysical lines below the black square in the Figure 2.

In the Figure 3 is plotted the L^∞ error norm of each subdomain.

The H^1 error reduction factor from a grid of maximum mesh size h_1 to the next h_2 is denoted by η and is computed such that $\frac{||\text{error}_{h_1}||_{H^1}}{||\text{error}_{h_2}||_{H^1}} = \left(\frac{h_1}{h_2}\right)^\eta$.

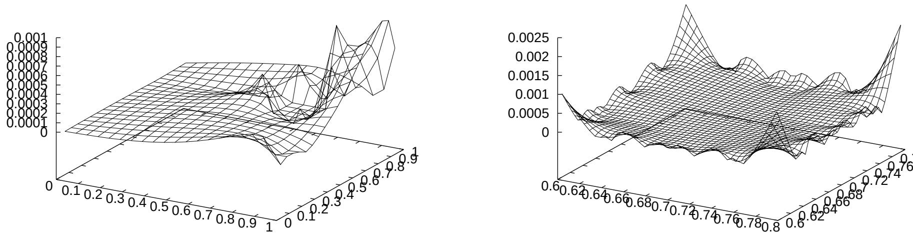

Figure 3: Infinite error : case 2. Left: in the external subdomain Ω_2. Scale: from 0 to $1E-3$. Right: in the internal subdomain Ω_1. Scale: from 0 to $2.5E-3$.

2^n	$1/h$	$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_{H^1}$	H^1 error reduction	η
1	5	$2.784E-2$	$4.744E-2$		
2	10	$7.379E-3$	$1.464E-2$	3.24	1.70
4	20	$2.054E-3$	$5.054E-3$	2.90	1.53
8	40	$5.617E-4$	$1.777E-3$	2.84	1.51
16	80	$1.520E-4$	$6.281E-4$	2.83	1.50

Table 1: Case 2: error as a function of h.

For the cases 1 and 2, the asymptotic **order of convergence in the H^1 norm is** $\mathcal{O}(h^{3/2})$ (see for example the Table 1 for the case 2). The super-convergence that occurs in the finite volume method for structured grids seems partly kept when the meshes are *regular and non-matching but nested*. The order $\mathcal{O}(h^{1/2})$ that is missing to attain the second order that is expected for the matching regular grids is probably the cost to pay the non-conforming meshes.

In the case 3, where the meshes are not nested, we cannot see a clear asymptotic order of convergence. But we can notice that the H^1 error is much larger in the unnested case than in the nested case.

3.2 Simplified ANDRA test case

It is a very simplified 2D static model of a geological layer containing 50 alveolas in which a source releases a contaminant. We normalized to one all the physical parameters, as the problem is linear. Thus the results **cannot be interpreted physically**. The purpose of this exercise is to show the capabilities of the non-conforming domain decomposition presented in the paper to treat a problem that requires a local refinement.

The geometry is depicted in Figure 4. The domain $\Omega = [0, 1100] \times [0, 100]$ is decomposed into 2 subdomains $\Omega_1 = [40, 1060] \times [40, 60]$, that contains all the sources, and $\Omega_2 = \Omega \backslash \Omega_1$. The interface boundary is depicted with dashed lines. The source term is defined by $q = 1$ in the alveolas that are the grey squares in the Figure 4, and 0 elsewhere.

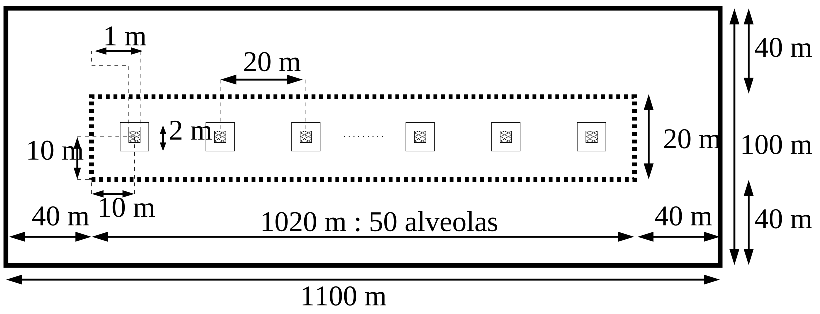

Figure 4: Geometry of the very simplified Andra test case.

3.2.1 Results

One can see an example of a result in the Figure 5.

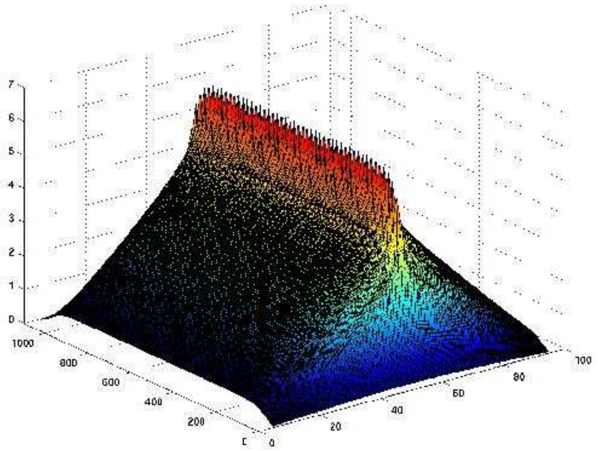

Figure 5: Interpolated solution of the simplified Andra test case. Grids: (110 x 10) and (2040 x 40) cells. There are 50 peaks corresponding to the 50 sources.

To perform this computation, the grid of the external subdomain Ω_2 is made up of 110×10 square cells of length $10m \times 10m$. The grid of the internal subdomain Ω_1 is made up of 4080×80 square cells of length $0.25m \times 0.25m$. BiCGStab converges within 34 iterations (less than 10 minutes a standard PC) for 8528 unknowns on the interface and more than 163 000 cells.

Note that the external mesh is much coarser than the internal mesh. *The mesh in the geological layer is 40 times coarser than the mesh surrounding the disposal site.*

4 The programming environment OCamlP3L

We studied so far a domain decomposition method for non-matching grids. To perform the numerical tests, we programmed in `C++` the subdomain solver and added another layer to perform the coupling between the subdomains. We call this a *direct* simulation. In this section we present a way of coupling the subdomain solver via an external program written in OCamlP3L.

4.1 Presentation

We used the modules (Seqp3l, Parp3l, Grap3l) written in OCaml and denoted by OCamlP3l, cf. [9] and `http://www.ocamlp3l.org/`. OCamlP3l is a library that allows to write parallel programs using OCaml according to skeleton models. This provides a very brief description of the parallel structure of a program following the ideas of [7]. At first our main goal was just to *couple* some applications (written in `C, C++` or `Fortran`) and the ability of OCamlP3L to couple *and parallelize* the applications appeared as an interesting extra feature to us. It was particularly convenient that the parallel execution of our applications required no extra development, once the sequential program is validated.

4.2 Basic skeletons

For the present study, we have used 4 of the basic skeletons provided by the OCamlP3l environment. The `mapvector` skeleton encapsulates the parallel execution of a skeleton over the components of a vector on a given number of processes, typically the computation on each subdomain. The `loop` skeleton is recursive, it encapsulates the repeated execution of a skeleton until some condition is false, typically an iterative algorithm as BiCGStab to solve a linear system. The `pipe` skeleton encapsulates the sequential composition of two skeletons. And the `seq` skeleton encapsulates the execution of a function in a sequential process.

The external encapsulation is performed by the function `startstop`. It encapsulates a skeleton by generating the input data and executing post-processing, this defines an OCamlP3l network.

The `pardo` function is used to call the (parallel) execution of a valid OCamlP3l network defined by a `startstop` call.

4.3 Compilation

To compile a network, one can use the tool `ocampl3lcc` with one of the 3 options `-seq` (for sequential), `-par` (for parallel) or `-gra` (for graphics). The purpose of the sequential compilation is to run the program sequentially on one machine, or to test the logic of the implemented algorithm with all the usual sequential debugging tools. The graphics mode allows to get a picture of the processor network described by the `P3l` skeleton expression, and it is useful to grasp the parallel structure of the program. The parallel option allows to run in parallel the program, after debugging in sequential and a simple recompilation.

4.4 Numerical tests of the coupling

OCaml coupling program

We briefly describe the main program of the coupling code defined with the function `startstop` taking three input arguments. We want to point out that we wrote a program based on a very simple algorithm and the resulting coupling code is very short (about 170 lines), thanks to the conciseness of the OCaml language and of the OCamlP3L environment.

The first argument is used to initialize the coupling. It generates the boundary conditions for each subdomain. The second argument finalizes the computations and prints the results. The third argument describes the structure of the coupling. The algorithm that makes the coupling is defined here.

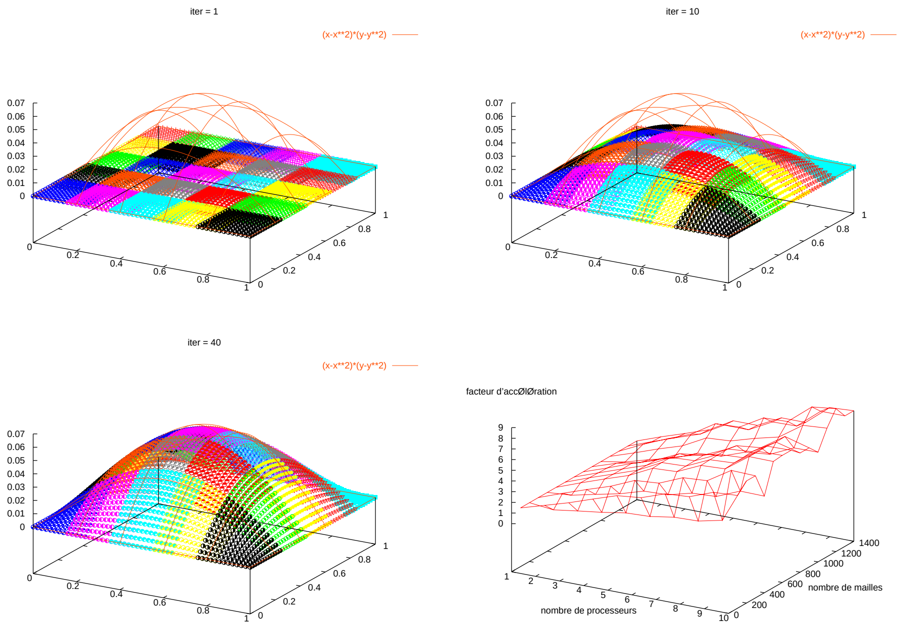

Figure 6: A computation with 25 subdomains (the subdomains are square and have their own color). Initial solution (up left), solution computed after 10 iterations (up right) and final solution after 40 iterations (down left). The analytic solution is depicted in red. Down right: speed up for 10 processors with refined grids: from 100×100 cells up to 1400×1400 cells.

For these first experiments we chose a very simple example of overlapping domain decomposition fixed point algorithm, using Dirichlet boundary conditions, with matching and structured grids (Schwarz algorithm, see for instance [12] or [10]). We do not discuss this algorithm in this paper, but we simply remark that it is particularly easy to program on a structured grid and thus was a good starting point to test the coupling through OCamlP3L. Obviously it is not the algorithm that we intend to use to make some more physical simulations.

The structure of the third argument is a loop. At each iteration, the program solves on `nbproc` processes a subdomain problem and returns as output a vector of interface conditions. These interface conditions are taken as input of a projection function `proj` from a subdomain to its neighbor and then the current result is plotted with `plot`.

```
let program () =
  startstop
    (generate_input_stream, ignore)
    (print_result, ignore, ignore)
    (loop (convergence, mapvector (seq (computation_on_a_subdomain), nbproc)
      ||| seq (proj) ||| seq (plot)))
in
pardo program;;
```

Subdomain solver program (in C)

The program that solves a subdomain problem has the following structure. It reads on the standard input the vector containing the input Dirichlet boundary conditions, solves the Poisson problem, and then returns on the standard output the vector containing the computed output vectors. Then it waits for another input vector. The standard input and output are redirected to the ones of the coupling program.

4.5 Results

The tests that are presented were performed on the unit square $\Omega = [0,1] \times [0,1]$, divided into 5×5 square subdomains with regular matching grids. The overlap is made up of 2 cells. The evolution of the solution for different iterations in the fixed point algorithm is presented in Figure 6.

In Figure 6 (down right), we can notice that we obtained a speed up of order 6 on 10 processors. This is a reasonable results considering that the implementation is not optimised at all. The speed up seem unsensitive to the number of cells. This means that we did not detected in this experiment the drawback of this simple algorithm: the projection phase is performed with only one processor and can produce delays due to communication overhead.

5 Conclusion and perspectives

We presented in detail a nonoverlapping domain decomposition method for non matching meshes based on Robin interface conditions. The matrix-free matrix vector product can be easily implemented with the details provided. Two 2D numerical experiments illustrate the good behavior of the method, particularly when the grids are nested, but not necessarily. They also show that it can be applied to simulate the flow around a nuclear waste disposal. Some more realistic 3D experiments are under way.

We also presented preliminary results of code coupling using the OCamlP3L environment written in the OCaml language. The final goal is to couple codes that solve different physics, or the same problem on different subdomains. The first tests we performed are based on the very simple overlapping Schwarz domain decomposition algorithm. The coupling consists in linking the subdomain solvers. The results are produced with a particularly short coupling program. The program can be executed in parallel, thus taking advantage of the natural parallel properties of the domain decomposition method, with no extra development cost: it only requires a simple recompilation. All this seems very promising.

What we are working on now is the implementation of the 3D domain decomposition method with non matching grids in the coupling code. Comparisons will be made between the direct simulation and the computation using code coupling with OCamlP3L, in terms of efficiency in solving the global problem, and in terms of time required to develop the programs.

References

[1] Y. Achdou, C. Japhet, Y. Maday, and F. Nataf. A new cement to glue non-conforming grids with Robin interface conditions: the finite volume case. *Numer. Math.*, 92(4):593–620, 2002.

[2] Todd Arbogast, Lawrence C. Cowsar, Mary F. Wheeler, and Ivan Yotov. Mixed finite element methods on nonmatching multiblock grids. *SIAM J. Numer. Anal.*, 37(4):1295–1315 (electronic), 2000.

[3] Todd Arbogast and Ivan Yotov. A non-mortar mixed finite element method for elliptic problems on non-matching multiblock grids. *Comput. Methods Appl. Mech. Engrg.*, 149(1-4):255–265, 1997. Symposium on Advances in Computational Mechanics, Vol. 1 (Austin, TX, 1997).

[4] C. Bernardi, Y. Maday, and A. T. Patera. A new nonconforming approach to domain decomposition: the mortar element method. In *Nonlinear partial differential equations and their applications. Collège de France Seminar, Vol. XI (Paris, 1989–1991)*, pages 13–51. Longman Sci. Tech., Harlow, 1994.

[5] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods.* Springer-Verlag, Berlin, 1991.

[6] G. Chavent and J. Roberts. A unified physical presentation of mixed, mixed-hybrid finite elements and standard finite difference approximations for the determination of velocities in waterflow problems. *Advances in Water Ressources*, 14(6):329–348, 1991.

[7] M. Cole. *Algorithmic Skeletons: Structured Management of Parallel Computations.* Research Monographs in Parallel and Distributed Computing. Pitman, 1989.

[8] Lawrence C. Cowsar, Jan Mandel, and Mary F. Wheeler. Balancing domain decomposition for mixed finite elements. *Math. Comp.*, 64(211):989–1015, 1995.

[9] Marco Danelutto, Roberto Di Cosmo, Xavier Leroy, and Susanna Pelagatti. Parallel functional programming with skeletons: the ocamlp3l experiment. *The ML Workshop*, 1998.

[10] Patrick Le Tallec. Domain decomposition methods in computational mechanics. *Comput. Mech. Adv.*, 1(2):121–220, 1994.

[11] J.-C. Nedelec. A new mixed finite elements in R^3. *Numer. Math.*, 50:57–81, 1986.

[12] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations.* The Clarendon Press Oxford University Press, New York, 1999. Oxford Science Publications.

[13] P.-A. Raviart and J.-M. Thomas. A mixed finite element method method for second order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical Aspects of Finite Element Methods; Lecture Notes in Mathematics 606*, pages 292–315. Springer-Verlag, Berlin, 1977.

[14] J.E. Roberts and J.-M. Thomas. Mixed and hybrid methods. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis Vol.II*, pages 523–639. North Holland, Amsterdam, 1991.

[15] Van der Vorst H. A. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.