HAL open science

ComPASS: a tool for distributed parallel finite volume discretizations on general unstructured polyhedral meshes

Eric Dalissier, Cindy Guichard, Pascal Have, Roland Masson, Chang Yang

To cite this version:

Eric Dalissier, Cindy Guichard, Pascal Have, Roland Masson, Chang Yang. ComPASS: a tool for distributed parallel finite volume discretizations on general unstructured polyhedral meshes. ESAIM: Proceedings, 2013, CEMRACS 2012, 43, pp.147-163. 10.1051/proc/201343010 . hal-00781235v2

HAL Id: hal-00781235

https://hal.science/hal-00781235v2

Submitted on 30 Apr 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ComPASS : a tool for distributed parallel finite volume discretizations on general unstructured polyhedral meshes

E. Dalissier[*], C. Guichard[†], P. Havé[‡], R. Masson[§], C. Yang[¶]

April 30, 2013

Abstract

The objective of the ComPASS project is to develop a parallel multiphase Darcy flow simulator adapted to general unstructured polyhedral meshes (in a general sense with possibly non planar faces) and to the parallelization of advanced finite volume discretizations with various choices of the degrees of freedom such as cell centres, vertices, or face centres. The main targeted applications are the simulation of CO_2 geological storage, nuclear waste repository and reservoir simulations.

The CEMRACS 2012 summer school devoted to high performance computing has been an ideal framework to start this collaborative project. This paper describes what has been achieved during the four weeks of the CEMRACS project which has been focusing on the implementation of basic features of the code such as the distributed unstructured polyhedral mesh, the synchronization of the degrees of freedom, and the connection to scientific libraries including the partitioner METIS, the visualization tool PARAVIEW, and the parallel linear solver library PETSc. The parallel efficiency of this first version of the ComPASS code has been validated on a toy parabolic problem using the Vertex Approximate Gradient finite volume spatial discretization with both cell and vertex degrees of freedom, combined with an Euler implicit time integration.

Introduction

ComPASS, standing for Computing Parallel Architecture to Speed up Simulations, is a project focusing on the solution of time-evolution problems on general polyhedral meshes using Finite Volume (denoted FV in the following) discretizations which has been the subject of active researches over the last decades. The long-term ambition of this project is to develop a parallel prototype code in order to implement and compare on relevant problems a large class of promising FV schemes with degrees of freedom (d.o.f.) typically located at cells, faces or vertices as exhibited in the recent 3D benchmark for diffusion problems [16]. The targeted applications are those of the INRIA project team COFFEE (COmplex Flows For Energy and Environment), such as the modelization of multiphase flows in porous media applied to oil and gas recovery in petroleum reservoirs, oil exploration in sedimentary basins, the geological storage of CO_2 in saline aquifers or the disposal of high level radioactive waste.

These applications imply the simulation of multiphase Darcy flows in heterogeneous anistropic porous media, strongly coupling the mass conservation laws with the local hydrodynamical and thermodynamical closure laws and involving a large range of time and space scales. The time integration schemes are usually fully implicit in

[*]LJK UMR 5224 Université de Grenoble

[†]LJAD UMR 7351 Université de Nice Sophia Antipolis & team COFFEE INRIA Sophia Antipolis Mediterranée

[‡]IFP Energies nouvelles, Rueil-Malmaison

[§]LJAD UMR 7351 Université de Nice Sophia Antipolis & team COFFEE INRIA Sophia Antipolis Mediterranée

[¶]ICJ UMR 5208 Université de Lyon 1, this author is partially supported by European Research Council ERC starting Grant 2009, project 239983-NuSiKiMo.

order to allow for large time steps. The spatial discretization is typically a FV scheme with an approximation of the Darcy fluxes splitting their elliptic part (see $-\Lambda(x)\ \nabla P^\alpha$ in (1) below) and their hyperbolic part (mobility terms $m_i^\alpha(x,U)$ in (1) below). The discretization of the elliptic part represents the main difficulty on which we will be focusing since it must be adapted to complex meshes, and to the anisotropy and heterogeneity of the porous media. Most recent FV schemes use various d.o.f. such as cell, face, or vertex unknowns for the approximation of these fluxes. They are compact in the sense that the stencil of the scheme is limited to unknowns located at the neighbouring cells of a given d.o.f (in the sense that they share a vertex). The hyperbolic part of the fluxes is usually only dealt with a first order upwind approximation to allow for fully implicit time integration, and hence it does not break the compacity of the scheme in the above sense. These FV implicit discretizations lead to solve at each time step and at each Newton type iteration, ill conditioned large sparse linear systems using preconditioned Krylov methods.
The authors have already developed several non parallel codes implementing multiphase Darcy flow models with various FV schemes. Given the high memory requirements of these type of applications, these codes were limited basically to 2D or coarse 3D meshes. In order to overcome these limitations, both in terms of memory and CPU time, we have decided to start the implementation of a new code using a parallel distributed approach.
A first crucial step toward this objective is to develop a tool adapted to general polyhedral meshes, which should be able to partition the mesh and to distribute it on a given number of processes, to compute FV schemes using various d.o.f. such as cell, face and vertex unknowns, to assemble and solve in parallel non linear and linear systems within a time loop, and to visualize the solutions at given time steps. In order to make this project tractable in a small amount of time, we decided to focus on the following specifications:

- general meshes including polyhedral cells with an arbitrary number of faces and possibly non planar faces,
- various and open request connectivities for cells, faces, and vertices, and one full layer of ghost cells (including all its cell, face and vertex d.o.f.) in order to easily compute and assemble locally on each processor a large class of compact FV discretizations,
- in order to simplify the first implementation, limit in a first step the size of the problem up to say 10^7 cells on a maximum number of one thousand processes.

One possibility would have been to fully embed the code on an existing open source software such as the library PETSc [4] which offers a tool-kit for scientific computing with emphasis on linear systems solutions, or the Trilinos Project [5] which is an object-oriented software framework for the solution of large-scale complex multi-physics problems, or also the object oriented distributed framework DUNE [1], which is a modular C++ library for the solution of partial differential equations using grid-based methods.
However, it was not clear that these libraries were easy to adapt in order to deal with general polyhedral meshes, and our team did not include advanced users of these libraries. Alternatively, in order to have an easy full control on the core of the code (mesh format, connectivity and distribution, FV schemes, physics, assembly of the non-linear and linear systems) we have decided to develop it from scratch and to rely on existing up to date scientific computing libraries only for specific tasks such as the partitioner, the linear solver and the visualization tools. Among well known programming languages for High Performance Computing (HPC), Fortran 90 combined with MPI has been chosen for our implementation rather than an object oriented language like C++ mainly because our team is already experienced with Fortran 90 and not with C++. The research session of the CEMRACS 2012 summer school, devoted to HPC, and joining at the CIRM around 100 researchers during 5 weeks, has been clearly a perfect framework to start the development of the ComPASS project.

1 Models and discretization

As stated in the introduction, the final goal of the ComPASS project is to simulate complex flows in geosciences such as compositional multiphase flows in porous media used in many applications. For example, in oil reservoir modelling, the compositional triphase Darcy flow simulator is a key tool to predict and optimize the production of a reservoir. In sedimentary basin modelling, such models are used to simulate the migration of the oil and gas phases in a basin saturated with water at geological space and time scales. The objectives are to predict the location of the potential reservoirs as well as the quality and quantity of oil trapped therein. In CO_2 geological storage, compositional multiphase Darcy flow models are used to optimize the injection of CO_2 and to assess the long term integrity of the storage. The numerical simulation of such models is a challenging task, which has been the subject of a lot of works over a long period of time, see the reference books [8] and [21]. Let us briefly sketch the framework of these models.

We recall here the basis of the Coats' formulation [10] for compositional multiphase Darcy flow models. Let $\mathfrak{P}$ denote the set of possible phases $\alpha \in \mathfrak{P}$, and $\mathfrak{C}$ the set of components $i \in \mathfrak{C}$. Each phase $\alpha \in \mathfrak{P}$ is described by its non empty subset of components $\mathfrak{C}^\alpha \subset \mathfrak{C}$ and is characterized by its thermodynamical and hydrodynamical properties depending on its pressure P^α, its molar composition $C^\alpha = (C_i^\alpha)_{i \in \mathfrak{C}^\alpha}$, and on the volume fractions or saturations denoted by $S = (S^\alpha)_{\alpha \in \mathfrak{P}}$. The phases can appear or disappear due to thermodynamical equilibrium, which leads to the introduction of the new variable $\mathfrak{Q} \subset \mathfrak{P}$ representing the number of present phases at each point of the space time domain. The set of unknowns of the system in the Coat's formulation is defined by $U = (P^\alpha, S^\alpha, \alpha \in \mathfrak{P},\ C^\alpha, \alpha \in \mathfrak{Q})$ and the flow is governed by the following conservation equations,

$$\phi(x)\ \partial_t n_i(U)\ +\ \sum_{\alpha \in \mathfrak{Q}} \operatorname{div}\left(-m_i^\alpha(x,U)\ \Lambda(x)\ \nabla P^\alpha\right)\ =\ 0, \qquad \text{for all } i \in \mathfrak{C}, \tag{1}$$

where n_i denote the number of moles of the component $i \in \mathfrak{C}$ per unit volume, m_i^α is roughly speaking the mobility of the component i in the phase α, $\phi(x)$ and $\Lambda(x)$ are respectively the porosity and the permeability tensor and both are functions of the position x in space. The system is closed by the hydrodynamical and thermodynamical laws as well as the volume balance equation $\sum_{\alpha \in \mathfrak{P}} S^\alpha = 1$. We refer to [15] and the references therein for a more detailed discussion about this formulation.

There are two main difficulties to solve such models coupling parabolic/elliptic and hyperbolic/degenerate parabolic unknowns. First, degeneracies appear in the set of equations (1) typically at the end points of the hydrodynamical laws, and at phase transitions. Second, the heterogeneous and anisotropic nature of the media leads to possibly strong jumps of the permeability tensor $\Lambda(x)$, of the porosity $\phi(x)$, and also of the mobilities $m_i^\alpha(x,U)$ and of the capillary pressures $P_{c,\alpha,\beta}(x,S) = P^\alpha - P^\beta$.

FV discretizations are particularly adapted to the discretization of such models due to their conservative properties and their ability to capture discontinuous solutions. They are usually combined with a forward Euler integration in time coupling implicitly the closure laws and the conservation equations in order to obtain the stability of the scheme even for large time steps. The extension of FV discretizations to general polyhedral meshes and to anisotropic heterogeneous media has been the object of intensive research in the last two decades and has lead to numerous schemes using various d.o.f. as exhibited in the recent 3D benchmark for diffusion problem [16]. Nevertheless, finding the 'best' FV scheme is still a problem dependent issue and the main objective of the ComPASS project is to build an appropriate tool to test promising schemes on problems with realistic physics, mesh size, and time scales, using parallel distributed computations.

1.1 A 3D mesh format for unstructured general grids

In geological applications, in order to capture the geometry and the main heterogeneities of the media, the grid generation is constrained to match the main structural features of the reservoir or basin geometry defined by the geological sedimentary layers possibly cut by a network of faults.

For that purpose, the most commonly used mesh format in reservoir simulation is still the so called Corner Point Geometry (CPG) which is a globally structured hexahedral grid, locally unstructured due to erosion

leading to degenerate hexahedra (collapse of edges, or of entire faces), and locally non-matching due to faults as illustrated in Figure 1. In addition, non-conforming Local Grid Refinement (LGR) can be used in nearwell regions.

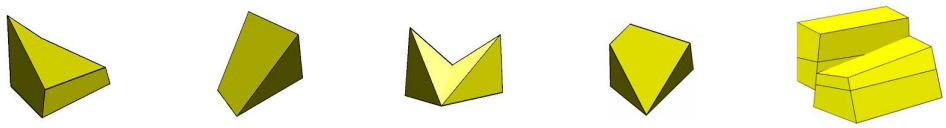
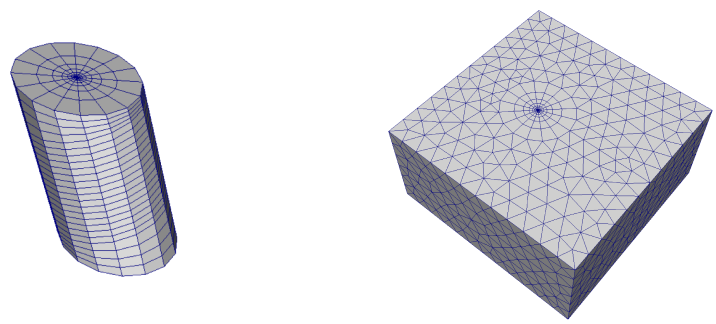


Figure 1: Examples of general cells due to geological erosion (left part) or faults (the last at the right).

Although CPG grids are widely used for full field reservoir simulations, they are not adapted to modelize complex fault networks such as Y-shaped faults at basin scales, nor to discretize complex geometries at finer scales such as in the neighbourhood of complex wells or fracture networks. This justifies the growing interest to use unstructured meshes for such applications. For instance, let us consider a 3D nearwell mesh exhibited in Figure 2. The first step of the discretization is to create a radial mesh that is exponentially refined down to the well boundary to take into account the singular behaviour of the pressure close to the well. Then, this radial grid is matched to the reservoir grid using tetrahedra as well as pyramids at the interface, as can be seen in Figure 2(b). This transition zone enables to couple the nearwell simulation to a global reservoir simulation.



(a) exponentially refined radial mesh

(b) hybrid mesh with hexahedra, tetrahedra and pyramids

Figure 2: Space discretization of a deviated well.

To conclude, geological applications require to be able to use general polyhedral meshes with possibly non planar faces. Our 3D mesh format implements the mathematical definition given below. It is a conforming mesh format with a description of the cells by their set of faces and of the faces by their set of vertices given in cyclic order. It is assumed that the possible non-conformities have been solved by computing the co-refinement of the surfaces on both sides of the non-conformities generating a conforming mesh with general polyhedral cells. Our format is very close to the one used for the 3D benchmark [16] without the description of the set of edges which is expensive in terms of memory requirement and usually not used in practice for most FV schemes. For the sake of simplicity, it will be assumed in the following that the edge data structure is not used. It could always be computed and added if required by the discretization scheme. Let us also specify that the rock properties such as the permeability tensor and the porosity will be defined as usual as cellwise constant.

Definition 1.1 (Admissible space discretization) *Let Ω be an open bounded subset of $\mathbb{R}^3$ and $\partial\Omega = \overline{\Omega} \setminus \Omega$ its boundary. An admissible space discretization of Ω is defined by assuming that a primary polyhedral mesh $\mathcal{M}$ is given where each element ("commonly named cell") $K \in \mathcal{M}$ is strictly star-shaped with respect to some point x_K, called the centre of K. We then denote by $\mathcal{F}_K$ the set of interfaces of non zero two dimensional measure*

among the inner interfaces $\overline{K} \cap \overline{L}$, $L \in \mathcal{M}$, and the boundary interface $\overline{K} \cap \partial\Omega$ possibly split in several boundary faces. Each $\sigma \in \mathcal{F}_K$ ("face of K") is assumed to be the union of triangles $\tau \in \mathcal{S}_\sigma$. We denote by $\mathcal{V}_\sigma$ the set of all the vertices of σ, located at the boundary of σ ("vertices of σ"), and by $\mathcal{V}_\sigma^0$ the set of all the internal vertices of σ. Therefore, the 3 vertices of any $\tau \in \mathcal{S}_\sigma$ are elements of $\mathcal{V}_\sigma^0 \cup \mathcal{V}_\sigma$. Let us denote by

$$\mathcal{F} = \bigcup_{K \in \mathcal{M}} \mathcal{F}_K$$

the set of faces, and by

$$\mathcal{V} = \bigcup_{\sigma \in \mathcal{F}} \mathcal{V}_\sigma,$$

the set of vertices of the mesh $\mathcal{M}$.

In practice, it will be assumed that the set $\mathcal{V}_\sigma^0$ reduces to the isobarycentre of the set of vertices $\mathcal{V}_\sigma$. Hence, the mesh format reduces to a numbering of the cells $\mathcal{M}$, of the faces $\mathcal{F}$ and of the vertices $\mathcal{V}$, to the coordinates of the vertices, to the definition of the sets $\mathcal{F}_K$ for all cells $K \in \mathcal{M}$, and of the sets $\mathcal{V}_\sigma$ in cyclic order for each face $\sigma \in \mathcal{F}$. In addition, the set of neighbouring cells $\mathcal{M}_\sigma$ is given for each face $\sigma \in \mathcal{F}$. It is well known that, for a conforming mesh, the set $\mathcal{M}_\sigma$ contains either two cells for an inner face and one cell for a boundary face.

1.2 Example of the VAG scheme on a parabolic problem

In the short term of the CEMRACS, it has been decided to implement the following simple linear parabolic equation,

$$\partial_t u(x,t) - \triangle u(x,t) = f(x,t), \tag{2}$$

in the space-time domain $\Omega \times (0,T)$, with the initial condition $u(x,0) = u_{\text{ini}}(x)$, for a.e. $x \in \Omega$, and an homogeneous Dirichlet boundary condition, under the following assumptions:

- Ω is an open bounded connected polyhedral subset of $\mathbb{R}^3$, and $T > 0$,
- $f \in L^2(\Omega \times (0,T))$ and $u_{\text{ini}} \in L^2(\Omega)$.

The implementation of this simple model already includes most of the features required for more complex models such as the compositional Darcy flow model (1): a time loop for time integration, the discretization of a second order elliptic operator that will be implemented using FV fluxes, and the solution of a linear system at each time step.
The ComPASS framework aims to implement a large class of FV schemes for the discretization of such fluxes on general meshes. It includes for example the cell centred Multi-Points Flux Approximation (MPFA) schemes with in particular the well-known MPFA O-scheme introduced in [6, 11]. These cell centred schemes are consistent on general meshes but they are non-symmetric, and their coercivity and convergence are mesh dependent. In order to recover an unconditional symmetry and coercivity while keeping a compact flux stencil, one needs to use additional d.o.f. For instance, Mimetic schemes [9] or similarly the Sushi scheme [13] use face unknowns in addition to the cell unknowns. The DDFV (Discrete Duality Finite Volume) scheme [7] uses both cell and vertex unknowns. For our test case, we have chosen to implement the VAG (Vertex Approximate Gradient) scheme, also using both cell and vertex d.o.f. It has been introduced in [14] in the framework of the gradient schemes. We will briefly describe the main characteristics of this scheme and we refer to [15] for its application to multiphase Darcy flows.

1.2.1 Gradient schemes for a parabolic problem

The gradient schemes were recently introduced in order to obtain a uniform framework for the definition and the convergence analysis of a large family of schemes including conforming discretizations (such as finite element

methods) and non conforming discretizations (such as mixed finite element and finite volume methods). The gradient schemes framework has already been applied to several problems, we mention especially here the case of the Stefan problem [12] since it includes our simple model (2) as a particular case.
Gradient schemes are based on the weak primal formulation of the model which in the case of the parabolic equation (2) leads to the following variational formulation: find $u \in L^2(0,T;H_0^1(\Omega))$ such that

$$\int_0^T \int_\Omega \left(- u(x,t)\partial_t\varphi(x,t) + \nabla u(x,t)\cdot\nabla\varphi(x,t)\right) \mathrm{d}x\mathrm{d}t - \int_\Omega u_{\mathrm{ini}}(x)\varphi(x,0)\mathrm{d}x = \int_0^T \int_\Omega f(x,t)\varphi(x,t)\mathrm{d}x\mathrm{d}t, \quad (3)$$

for all $\varphi \in C_c^\infty(\Omega \times [0,T[)$. The spirit of the gradient schemes is to replace the continuous operators by their discrete versions leading to the following definition of a space gradient discretization.

Definition 1.2 (Space gradient discretization) *A space gradient discretization of the problem* (3) *is defined by* $\mathcal{D} = (X_{\mathcal{D},0}, \Pi_\mathcal{D}, \nabla_\mathcal{D})$ *where,*

1. *the set of d.o.f.* $X_{\mathcal{D},0}$ *is a finite dimensional vector space on* $\mathbb{R}$ *suited to the Dirichlet homogeneous boundary condition,*
2. *the linear mapping* $\Pi_\mathcal{D} \ : \ X_{\mathcal{D},0} \to L^2(\Omega)$ *is the reconstruction of the approximate function,*
3. *the linear mapping* $\nabla_\mathcal{D} \ : \ X_{\mathcal{D},0} \to L^2(\Omega)^3$ *is the discrete gradient operator. It must be chosen such that* $\|\cdot\|_\mathcal{D} := \|\nabla_\mathcal{D}\cdot\|_{L^2(\Omega)^3}$ *is a norm on* $X_{\mathcal{D},0}$.

The time interval $(0,T)$ is discretized using, for the sake of simplicity, a constant time step denoted by δt and defined by $\delta t = t^{(n+1)} - t^{(n)}$ with $n = 0,\dots,N$ such that $t^{(0)} = 0 < t^{(1)} \dots < t^{(N)} = T$.
The problem (3) is then approximated by a space-time gradient discretization using the following Euler implicit time integration scheme. Given $u^{(0)} \in X_{\mathcal{D},0}$ with $\Pi_\mathcal{D} u^{(0)}$ an approximation of u_{ini}, the discrete solutions $u^{(n)} \in X_{\mathcal{D},0}$ satisfy

$$\left\{ \begin{array}{l} \displaystyle\int_\Omega \left(\Pi_\mathcal{D} \frac{u^{(n+1)} - u^{(n)}}{\delta t}(x)\Pi_\mathcal{D} w(x) + \nabla_\mathcal{D} u^{(n+1)}(x)\cdot\nabla_\mathcal{D} w(x)\right)\mathrm{d}x = \frac{1}{\delta t}\int_{t^{(n)}}^{t^{(n+1)}}\int_\Omega f(x,t)\Pi_\mathcal{D} w(x)\mathrm{d}x\mathrm{d}t, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall w \in X_{\mathcal{D},0},\ \forall n = 0,\dots,N-1. \end{array} \right. \quad (4)$$

1.2.2 Definition the VAG scheme

Following the Definition 1.1 of the space discretization of Ω, we also define $\mathcal{V}_K$ as the set of all elements of $\mathcal{V}$ which are vertices of K. For any $K \in \mathcal{M}$, $\sigma \in \mathcal{F}_K$, $\tau \in \mathcal{S}_\sigma$, we denote by $S_{K,\tau}$ the tetrahedron with vertex x_K and basis τ as illustrated in the Figure 3.

Let us then define the gradient scheme space and operators of Definition 1.2 for the VAG scheme.

- We define $X_{\mathcal{D},0}$ as the space of all vectors $u = (\ (u_K)_{K\in\mathcal{M}}\ ,\ (u_v)_{v\in\mathcal{V}}\ )$, $u_K \in \mathbb{R},\ u_v \in \mathbb{R}$ such that $u_v = 0$ for all $v \in \mathcal{V} \cap \partial\Omega$.

- Then, the mapping $\Pi_\mathcal{D}$ is simply defined, for any $u \in X_{\mathcal{D},0}$, by $\Pi_\mathcal{D} u(x) = u_K$, for a.e. $x \in K$.

- The mapping $\nabla_\mathcal{D}$ is defined, for any $u \in X_{\mathcal{D},0}$, by $\nabla_\mathcal{D} u = \nabla\widehat{\Pi}_\mathcal{D} u$, where $\widehat{\Pi}_\mathcal{D} u$ is the continuous reconstruction which is affine in all $S_{K,\tau}$, for all $K \in \mathcal{M}$, $\sigma \in \mathcal{F}_K$ and $\tau \in \mathcal{S}_\sigma$, with the values u_K at x_K, u_v at any vertex v of τ which belongs to $\mathcal{V}_\sigma$, and $\sum_{x\in\mathcal{V}_\sigma} \alpha_v^x u_x$ at any vertex v of τ which belongs to $\mathcal{V}_\sigma^0$, where the coefficients $\alpha_v^x, x \in \mathcal{V}_\sigma$ are the barycentric coordinates of v.

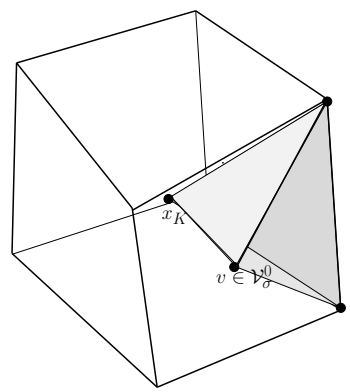


Figure 3: Tetrahedral submesh.

Using the above definition of the VAG scheme, and precisely thanks to the result that for all $x \in K$, for $K \in \mathcal{M}$, the discrete gradient $\nabla_{\mathcal{D}} u(x)$ is expressed as a linear combination of $(u_K - u_v)_{v \in \mathcal{V}_K}$, the Problem (4) can be rewritten as follows,

$$\sum_{K\in\mathcal{M}} \int_K \frac{u_K^{(n+1)} - u_K^{(n)}}{\delta t} w_K + \sum_{K\in\mathcal{M}} \sum_{v\in\mathcal{V}_K} F_{K,v}(u^{n+1})(w_K - w_v) = \sum_{K\in\mathcal{M}} \frac{1}{\delta t} \int_{t^{(n)}}^{t^{(n+1)}} \int_K f(x,t) w_K \mathrm{d}x\mathrm{d}t,$$

$$\forall w \in X_{\mathcal{D},0},\ \forall n = 0, \ldots, N-1, \tag{5}$$

where $F_{K,v}(u^{n+1})$ is interpreted as the flux from a cell $K \in \mathcal{M}$ to one of its vertices $v \in \mathcal{V}_K$ and is defined by,

$$F_{K,v}(u^{n+1}) = \sum_{v'\in\mathcal{V}_K} T_K^{v,v'} (u_K^{(n+1)} - u_{v'}^{(n+1)}). \tag{6}$$

Remark 1 Thanks to the definition of the above fluxes, the VAG scheme can be seen as a FV method. For this, we need to define some disjoint arbitrary control volumes $V_{K,v} \subset \bigcup_{v\in\overline{\tau}} S_{K,\tau}$ for all $K \in \mathcal{M}$ and $v \in \mathcal{V}_K$. It is important to notice that it is not in general necessary to provide a more precise geometric description of $V_{K,v}$ than its volume. The mapping $\Pi_{\mathcal{D}}$ is then defined for any $u \in X_{\mathcal{D},0}$ by $\Pi_{\mathcal{D}} u(x) = u_K$, for a.e. $x \in K \setminus \bigcup_{v\in\mathcal{V}_K} V_{K,v}$, and by $\Pi_{\mathcal{D}} u(x) = u_v$ for a.e. $x \in V_{K,v}$. Finally, related to the Definition (6) of the fluxes, the VAG scheme can be seen as a MPFA FV method for multiphase Darcy flow models in the set of control volumes defined as the union of the cells and of the vertices. We refer to [15] for a more detailed discussion.

1.2.3 Implementation and discussion

Finally, setting in (5), on the one hand $w_v = 1$ with all other values equal to zero, and, on the other hand $w_K = 1$ with all other values equal to zero, one obtains the following discretization of Problem (2): given $u^{(0)} \in X_{\mathcal{D},0}$, find $u^{(n+1)} \in X_{\mathcal{D},0}$ at each time step $n \geq 0$ such that

$$\sum_{K\in\mathcal{M}_v} \sum_{v'\in\mathcal{V}_K} -T_K^{v,v'} (u_K^{(n+1)} - u_{v'}^{(n+1)}) = 0 \quad \forall v \in \mathcal{V}\backslash\partial\Omega, . \tag{7a}$$

$$\frac{|K|}{\delta t}(u_K^{(n+1)} - u_K^{(n)}) + \sum_{v\in\mathcal{V}_K} \sum_{v'\in\mathcal{V}_K} T_K^{v,v'} (u_K^{(n+1)} - u_{v'}^{(n+1)}) = |K| f(x_K) \quad \forall K \in \mathcal{M}, \tag{7b}$$

where $|K|$ is the volume of the cell K and $\mathcal{M}_v$ is the set of cells sharing the vertex v, defined by

$$\mathcal{M}_v = \{K \in \mathcal{M} \mid v \in \mathcal{V}_K\}.$$

The advantage of this scheme is that it allows to eliminate without any fill-in all the cell unknowns $(u_K)_{K\in\mathcal{M}}$ by using the equation (7b). The system reduces to the Schur complement on the vertex unknowns $(u_v)_{v\in\mathcal{V}}$ leading to a compact vertex centred scheme. Its stencil involves only the neighbouring vertices of a given vertex, with a typical 27 points stencil for 3D topologically Cartesian meshes. Let us also point out that the VAG scheme applies to general meshes with possibly non planar faces. In addition, the VAG scheme is consistent and unconditionally coercive and has exhibited a good compromise between accuracy, robustness and CPU time in the recent FVCA6 3D benchmark [16].
In particular, the VAG fluxes can be used for the discretization of multiphase Darcy flows leading to a very efficient scheme on tetrahedral meshes or on hybrid meshes combining typically tetrahedra, hexahedra and pyramids. For such problems on tetrahedral meshes, the VAG scheme is say 15 times more efficient in comparison with cell centred discretizations such as the MPFA O-scheme. This result can be explained by the ratio of around 5 between the numbers of cells and the number of vertices, and by the very large stencil of the MPFA O-scheme on such meshes (about 3 times larger than the stencil of the VAG scheme). In addition, the usual difficulties of vertex centred approaches to deal with multiphase Darcy flows in heterogeneous media can be solved thanks to the fact that the cell unknown is kept in the discretization and thanks to the flexibility provided by the VAG scheme in the choices of the volumes $V_{K,v}$ (see [15] for more details).

2 ComPASS parallel distributed processing

In this section, assuming that a mesh satisfying Definition 1.1 is given, the main steps of the algorithm for the parallel distributed solution of equation (7) is detailed (see also Figure 4). It will be assumed that the topology of the mesh does not change during the time evolution. Consequently, the partitioning and the distribution of the mesh on the set of processes $\mathcal{P}$ will be done in a preprocessing step once and for all. Also in order to define the overlaps between processes (ghost cells, faces or vertices), the FV schemes are assumed to be compact in the sense that their stencil involves only d.o.f. located at the neighbouring cells of a given item. This remark leads to our definition of the overlap as one layer of cells.
ComPASS fits into the SPMD (Single Program, Multiple Data) paradigm. To simplify the implementation, it is assumed that a master process reads and stores the mesh, computes the partitioning, and distributes the local meshes to each process. After distribution of the mesh by the master process, all the computation are done in parallel, and synchronisations of the d.o.f. at the ghost items can be performed in order to match the overlapping d.o.f. between the processes. Following our example of equation (7), we will describe the parallel computation of the FV scheme as well as the assembling and the solution of the linear system at each time step using the PETSc [4] library. Thanks to our general definition of the overlap, the assembling step will be done locally on each process without communication.

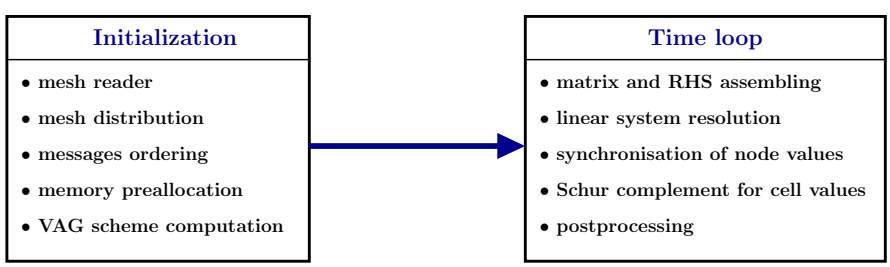


Figure 4: Main steps of the algorithm for the parallel distributed solution of equation (7).

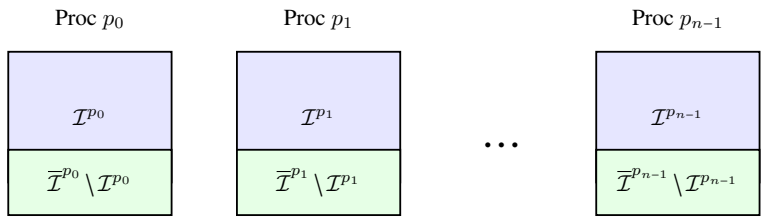


Figure 5: The overlapping decompositions of the sets of cells, vertices and faces. $\mathcal{I} = \mathcal{M}$, $\mathcal{V}$, or $\mathcal{F}$; $n = \text{card}(\mathcal{P})$.

2.1 Distributed mesh

The aim of this subsection is to distribute the mesh $\mathcal{M}$ on each process $p \in \mathcal{P}$. Since our objective is to deal with a large class of FV schemes with d.o.f. possibly located at various mesh items such as cells, faces or vertices, we have chosen to partition the set of cells

$$\mathcal{M} = \bigcup_{p \in \mathcal{P}} \mathcal{M}^p. \tag{8}$$

The overlapping decomposition of the set of cells

$$\overline{\mathcal{M}}^p, \quad p \in \mathcal{P},$$

is chosen in such a way that all compact finite volume schemes can be assembled locally on each processor. Hence $\overline{\mathcal{M}}^p$ is defined as the set of all cells sharing a vertex with $\mathcal{M}^p$.
The overlapping decompositions of the sets of vertices and faces follow from this definition

$$\overline{\mathcal{V}}^p = \bigcup_{K \in \overline{\mathcal{M}}^p} \mathcal{V}_K, \quad p \in \mathcal{P},$$

for the set of vertices $\mathcal{V}$, and

$$\overline{\mathcal{F}}^p = \bigcup_{K \in \overline{\mathcal{M}}^p} \mathcal{F}_K, \quad p \in \mathcal{P},$$

for the set of faces $\mathcal{F}$. These overlapping decompositions are sketched in Figure 5.
All these decompositions are computed on the master process from the initial mesh. The partitioning of the set of cells $\mathcal{M}$ is computed by the subroutine `METIS_PartGraphRecursive` of the Metis library [3] which uses a multi-constraint multilevel recursive bisection algorithm to minimize the number of edge cuts. It is provided with the number of processes $\text{card}(\mathcal{P})$, and with the sets $\mathcal{M}_K$, for $K \in \mathcal{M}$ of cells connected to K by a vertex defined by

$$\mathcal{M}_K = \bigcup_{v \in \mathcal{V}_K} \mathcal{M}_v.$$

The Metis subroutine output is the partitioning of the set of cells (8).
The next step is to choose the partitioning of the remaining sets of items, i.e. the set of vertices $\mathcal{V}$ and the set of faces $\mathcal{F}$. Let us start first with the vertices, the methodology is similar for the faces. The partitioning of the set of vertices is obtained by a rule of belonging of a given vertex to a process $p \in \mathcal{P}$, chosen to ensure a well balanced partitioning. To be more specific, let us denote by $\#_{\mathcal{M}}\, K$ the unique global index of a cell $K \in \mathcal{M}$. We have the following definition.

Definition 2.1 (Partitioning of the set of vertices) *A vertex $v \in \mathcal{V}$ belongs to the same process $p \in \mathcal{P}$ as the process of the cell with the smallest global index $\#_{\mathcal{M}}\ K$ among the set $\mathcal{M}_v$ of cells sharing v. Following this rule, we obtain the partitioning $\mathcal{V} = \bigcup_{p\in\mathcal{P}} \mathcal{V}^p$ with*

$$\mathcal{V}^p = \{v \in \mathcal{V} \mid K(v) \in \mathcal{M}^p\}, \quad p \in \mathcal{P},$$

where, for all $v \in \mathcal{V}$, $K(v) = \operatorname{argmin}_{L\in\mathcal{M}_v}\{\#_{\mathcal{M}}\ L\}$.

Similarly the partitioning of the set of faces $\mathcal{F}$ is given by the following definition.

Definition 2.2 (Partitioning of the set of faces) *A face $\sigma \in \mathcal{F}$ belongs to the same process $p \in \mathcal{P}$ as the process of the cell with the smallest global index $\#_{\mathcal{M}}\ K$ among the set $\mathcal{M}_\sigma$ of cells sharing σ. Following this rule, we obtain the partitioning $\mathcal{F} = \bigcup_{p\in\mathcal{P}} \mathcal{F}^p$ with*

$$\mathcal{F}^p = \{\sigma \in \mathcal{F} \mid K(\sigma) \in \mathcal{M}^p\}, \quad p \in \mathcal{P},$$

where, for all $\sigma \in \mathcal{F}$, $K(\sigma) = \operatorname{argmin}_{L\in\mathcal{M}_\sigma}\{\#_{\mathcal{M}}\ L\}$.

Let us emphasize that one clearly has from the above definitions that

$$\mathcal{V}^p \subset \bigcup_{K\in\mathcal{M}^p} \mathcal{V}_K \text{ and } \mathcal{F}^p \subset \bigcup_{K\in\mathcal{M}^p} \mathcal{F}_K.$$

It results, thanks to our choice of the overlaps, that the computation of the FV schemes will always be done locally on each process without communication, for all compact schemes requiring only one layer of neighbouring cells.

2.2 Communication and synchronization between processes

A basic ingredient for the parallelization of the solution algorithms is the synchronization of the d.o.f defined at the ghost items $\overline{\mathcal{I}}^p \setminus \mathcal{I}^p$, for all $p \in \mathcal{P}$, and for $\mathcal{I} = \mathcal{M}, \mathcal{V}$ or $\mathcal{F}$. This synchronization is achieved using MPI (Message Passing Interface, see [17, 18] for basic references) Send and Receive communications between processes. To that end, the following subsets of $\mathcal{P}$ are computed in a preprocessing step once and for all: first the subsets of processes from which p must receive ghost items

$$\mathcal{P}^{p,\leftarrow}_{\mathcal{I}} = \{q \in \mathcal{P}\backslash\{p\} \mid \overline{\mathcal{I}}^p \cap \mathcal{I}^q \neq \varnothing\}, \quad \mathcal{I} = \mathcal{M}, \mathcal{F}, \mathcal{V}, \quad p \in \mathcal{P}, \tag{9}$$

and second, the subsets of processes to which p needs to send ghost items

$$\mathcal{P}^{p,\rightarrow}_{\mathcal{I}} = \{q \in \mathcal{P}\backslash\{p\} \mid \mathcal{I}^p \cap \overline{\mathcal{I}}^q \neq \varnothing\}, \quad \mathcal{I} = \mathcal{M}, \mathcal{F}, \mathcal{V}, \quad p \in \mathcal{P}. \tag{10}$$

For each mesh item $\mathcal{I} = \mathcal{M}, \mathcal{V}, \mathcal{F}$, we also store once and for all, for each process $p \in \mathcal{P}$, the partitioning

$$\overline{\mathcal{I}}^p \setminus \mathcal{I}^p = \bigcup_{q\in\mathcal{P}^{p,\leftarrow}_{\mathcal{I}}} \mathcal{I}^q \cap \overline{\mathcal{I}}^p, \tag{11}$$

of the ghost items of p (see Figure 6 (b)), and the partitioning

$$\mathcal{I}^p \cap \left(\bigcup_{q\in\mathcal{P}, q\neq p} \overline{\mathcal{I}}^q \right) = \bigcup_{q\in\mathcal{P}^{p,\rightarrow}_{\mathcal{I}}} \mathcal{I}^p \cap \overline{\mathcal{I}}^q, \tag{12}$$

of the own items of p belonging to the ghost items of $q \neq p$, $q \in \mathcal{P}$ (see Figure 6 (a)).

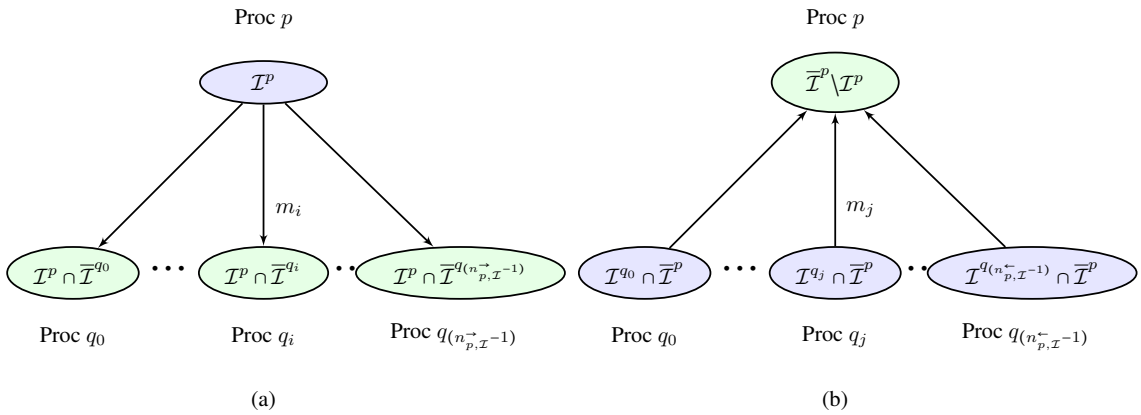


Figure 6: Communication of processor $p \in \mathcal{P}$ with other processors. (a) the processor p sends message $m_i = (\text{Send}, p, q_i)$ to the processor $q_i \in \mathcal{P}^{p,\rightarrow}_{\mathcal{I}}$, $i = 0, \ldots, n^{\rightarrow}_{p,\mathcal{I}} - 1$, with $n^{\rightarrow}_{p,\mathcal{I}} = \text{card}(\mathcal{P}^{p,\rightarrow}_{\mathcal{I}})$; (b) the processor p receives message $m_j = (\text{Receive}, p, q_j)$ from the processor $q_j \in \mathcal{P}^{p,\leftarrow}_{\mathcal{I}}$, $j = 0, \ldots, n^{\leftarrow}_{p,\mathcal{I}} - 1$, with $n^{\leftarrow}_{p,\mathcal{I}} = \text{card}(\mathcal{P}^{p,\leftarrow}_{\mathcal{I}})$.

These partitionings define the set of Send and Receive MPI messages that need to be sent and received to synchronize a given vector $w^p, p \in \mathcal{P}$ with $w^p \in V^{\mathcal{I}^p}$, where V is a given vector space of d.o.f (typically $\mathbb{R}^n$ for n a given number of unknowns per item). The synchronization provides the values of the vector at the ghost items on each process $p \in \mathcal{P}$ defining the overlapping vector $\overline{w}^p, p \in \mathcal{P}$ with $\overline{w}^p \in V^{\overline{\mathcal{I}}^p}$.
For each mesh item $\mathcal{I} = \mathcal{M}, \mathcal{V}, \mathcal{F}$, the set of all Send and Receive messages for a given process $p \in \mathcal{P}$ is denoted by

$$\mathbf{M}^p_{\mathcal{I}} = \{(\text{Receive}, p, q),\ q \in \mathcal{P}^{p,\leftarrow}_{\mathcal{I}}\} \cup \{(\text{Send}, p, q),\ q \in \mathcal{P}^{p,\rightarrow}_{\mathcal{I}}\}, \tag{13}$$

where, for the message $m = (\text{Send}, p, q)$, the process p sends to $q \in \mathcal{P}^{p,\rightarrow}_{\mathcal{I}}$ the values of w^p at the set of items $\mathcal{I}^p \cap \overline{\mathcal{I}}^q$, and for the message $m = (\text{Receive}, p, q)$, the process p receives in $\overline{w}^p$ from $q \in \mathcal{P}^{p,\leftarrow}_{\mathcal{I}}$ the values of w^q at the set of items $\mathcal{I}^q \cap \overline{\mathcal{I}}^p$.

Definition 2.3 (Complementary message) *For a given mesh item $\mathcal{I} = \mathcal{M}, \mathcal{V}, \mathcal{F}$, and a given message $m = (\text{mtype}, p, q)$, we denote by m^{-1} its complementary message which is defined by $m^{-1} = (\text{mtype}^{-1}, q, p)$ where* $\text{Send}^{-1} = \text{Receive}$*, and symmetrically* $\text{Receive}^{-1} = \text{Send}$.

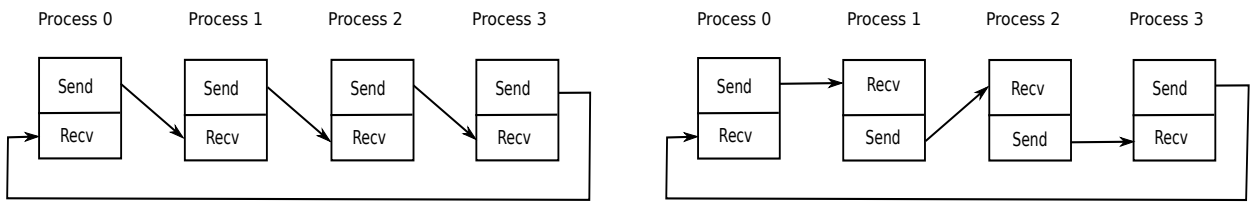
We have chosen to write the synchronization algorithm in a way that the execution will be deterministic and reproducible. This choice requires to use blocking point-to-point communications and an ordering of the messages to avoid possible deadlocks. This strategy has already proven its efficiency in the Arcane Framework [19] achieving a good scalability of the synchronization algorithm on more than 10000 processes.
Following [20, Subsection 2.1], there are two situations of deadlock that can occur when using blocking point-to-point MPI communications:

(d1) for a given message, its complementary message does not exist,

(d2) a Send-Receive cycle occurs, due to a wrong coordination of the Send and Receive messages.

The first case (d1) is avoided thanks our definition of the sets (9)-(10)-(13). Indeed, considering a mesh item $\mathcal{I}$ and a process $p \in \mathcal{P}$, then, for a given message $m \in \mathbf{M}^p_{\mathcal{I}}$, there clearly exists a unique process $q \in \mathcal{P}^{p,\leftarrow}_{\mathcal{I}} \bigcup \mathcal{P}^{p,\rightarrow}_{\mathcal{I}}$ such that $m^{-1} \in \mathbf{M}^q_{\mathcal{I}}$.
A Send-Receive cycle (d2) arises when two (or more) processes are all waiting for their complementary messages. Let us illustrate this case in the example exhibited on Figure 7 taken from [20, Subsection 2.1].

In this example $\mathcal{P} = \{p_i, i = 0,3\}$ and the set of messages (13) of each process is defined by $\mathbf{M}_{\mathcal{I}}^{p_i} = \{(\text{Send}, p_i, p_{i+1})\ ,\ (\text{Receive}, p_i, p_{i+1})\}$ for $i \in [\![0,3]\!]$, $p_4 = p_0$, with $\mathcal{I}$ a fixed mesh item. On the subfigure 7(a), deadlocks occur since all processes are sending their send message first. The reordering of the sets $(\mathbf{M}_{\mathcal{I}}^{p_i})_{i=0,3}$ proposed on the subfigure 7(b) allows to avoid such occurrence.



(a) Deadlocks (b) No deadlock

Figure 7: Example of communications between four processes.

We have to define a proper ordering of the set of messages $\mathbf{M}_{\mathcal{I}}^{p}$ in order to avoid deadlocks of type (d2). Roughly speaking our ordering rule will prescribe that:

(a) each process p performs its sendings and receptions in ascending order of the process ranks $q \in \mathcal{P}_{\mathcal{I}}^{p,\leftarrow} \bigcup \mathcal{P}_{\mathcal{I}}^{p,\rightarrow}$,

(b) when two processes p and q communicate, the one with lowest rank must sends its message first.

To be more specific, let us denote by $\#p$ the rank of a process $p \in \mathcal{P}$. Given two messages $m_i = (\text{mtype}_i, p, q_i)$, $i = 1,2$, $m_1 \neq m_2$ belonging to $\mathbf{M}_{\mathcal{I}}^{p}$, they are ordered according to the rule in the following Table 1.

```
if q1 = q2 = q then
   if mtype1 = Send then
      if #p < #q then
         m1 ≺ m2
      else
         m2 ≺ m1
      end if
   else if mtype1 = Receive then
      if #q < #p then
         m1 ≺ m2
      else
         m2 ≺ m1
      end if
   end if
else if #q1 < #q2 then
   m1 ≺ m2
else
   m2 ≺ m1
end if
```

Table 1: Ordering of two messages $m_1 \neq m_2 \in \mathbf{M}_{\mathcal{I}}^{p}$ with $m_i = (\text{mtype}_i, p, q_i)$, $i = 1,2$ and where $m_i \prec m_j$ means that the message m_i has to be executed before m_j for $i \neq j$.

Theorem 2.4 (Synchronisation without a deadlock situation (d2)) *For a given mesh item $\mathcal{I} = \mathcal{M}, \mathcal{V}, \mathcal{F}$, the execution of the messages in the ordering defined in the Table 1 applied to each set $\mathbf{M}_{\mathcal{I}}^{p}$, for all $p \in \mathcal{P}$, leads to a synchronisation without a deadlock occurrence in the sense of definition (d2).*

Proof. Let $\mathcal{P} = \{p_i\}_{i=1,n}$ the set of processes, with $\text{card}(\mathcal{P}) = n > 2$, and $\# \ p_i = i$ for all $i = 1, \cdots, n$. We assume, for the sake of simplicity, that all processes communicate, possibly with empty messages. This assumption simplifies the writing of the proof and corresponds in practice to a worst case assumption. Indeed, an empty message implies that its complementary message is also empty, which means in practice that no communication arises between both processes. The proof will be shown to hold even if a communication is considered in such a case.
The proof is done by induction over the number of processes n, and for a given mesh item $\mathcal{I}$.
Thus, let us first prove the result for $n = 2$. The ordering of the two sets of messages, $\mathbf{M}_{\mathcal{I}}^{p_1}$ and $\mathbf{M}_{\mathcal{I}}^{p_2}$ is done according to the point (b) of our ordering rule (corresponding to the lines $(1-14)$ of the algorithm described in the Table 1). It results that the ordered sets of messages are $\mathbf{M}_{\mathcal{I}}^{p_1} = [\ (\text{Send}, p_1, p_2) \ \prec \ (\text{Receive}, p_1, p_2)\ ]$ and $\mathbf{M}_{\mathcal{I}}^{p_2} = [\ (\text{Receive}, p_2, p_1) \ \prec \ (\text{Send}, p_2, p_1)\ ]$. It is clear that at the first execution step, the process p_1 executes the message (Send, p_1, p_2) and the process p_2 executes the complementary message $(\text{Receive}, p_2, p_1)$. Then, at the second step the process p_2 executes the message (Send, p_2, p_1) and the process p_1 executes the complementary message $(\text{Receive}, p_1, p_2)$ which shows that the communications are achieved without deadlock.
Let us now assume that the result is satisfied for n processes, and check that it is still true for $n+1$ processes. We denote by $\widetilde{\mathbf{M}}_{\mathcal{I}}^{p_i}$, for $i = 1, \cdots, n$, the ordered sets of messages between the first n processes which are, by assumption, well ordered to avoid a deadlock. The sets $\mathbf{M}_{\mathcal{I}}^{p_i}$, for a given $i \in [\![1, n]\!]$, are the union of $\widetilde{\mathbf{M}}_{\mathcal{I}}^{p_i}$ and of the messages (Send and Receive) between the processes p_i, $i \in [\![1, n]\!]$ and p_{n+1}. Since the process p_{n+1} has the highest rank, applying point (a) of our ordering rule (corresponding to the lines $(15-19)$ of the algorithm presented in the Table 1), each set of messages $\mathbf{M}_{\mathcal{I}}^{p_i}$, for $i = 1, \cdots, n$, is ordered starting with the ordered set $\widetilde{\mathbf{M}}_{\mathcal{I}}^{p_i}$. Then, using the point (b) of our ordering rule, we obtain the ordering,

$$\mathbf{M}_{\mathcal{I}}^{p_i} = [\ \widetilde{\mathbf{M}}_{\mathcal{I}}^{p_i} \ \prec \ (\text{Send}, p_i, p_{n+1}) \ \prec \ (\text{Receive}, p_i, p_{n+1})\ ], \ \ \forall i \in [\![1, n]\!].$$

Similar arguments lead to the following ordering of the set of messages of the process p_{n+1},

$$\mathbf{M}_{\mathcal{I}}^{p_{n+1}} = [\ (\text{Receive}, p_{n+1}, p_1) \ \prec \ (\text{Send}, p_{n+1}, p_1) \ \prec \ \cdots \ \prec \ (\text{Receive}, p_{n+1}, p_n) \ \prec \ (\text{Send}, p_{n+1}, p_n)\ ].$$

It results from our induction assumption that the messages $(\widetilde{\mathbf{M}}_{\mathcal{I}}^{p_i})_{i=1,n}$ will be executed first without deadlock. Next, from the above ordering of the messages, for $i = 1, \cdots, n$ in this order:

- the process p_i executes the message $(\text{Send}, p_i, p_{n+1})$,
- the process p_{n+1} executes the complementary message $(\text{Receive}, p_{n+1}, p_i)$,
- the process p_i executes the message $(\text{Receive}, p_i, p_{n+1})$,
- the process p_{n+1} executes the complementary message $(\text{Send}, p_{n+1}, p_i)$.

We conclude that all the communications are achieved without deadlock. □

2.3 Computation of the Finite Volume scheme

From our choice of the overlaps, the computation of the FV scheme can always be done locally on each process without communications, for all schemes requiring only one layer of neighbouring cells.
Let us consider our example of the computation of the VAG fluxes (6). For all process $p \in \mathcal{P}$, we compute the transmissivities $T_K^{v,v'}$, $v, v' \in \mathcal{V}_K$ for all $K \in \overline{\mathcal{M}^p}$. In our case, the diffusion coefficient is fixed to 1 and the transmissivities can be computed only once in the simulation, before the time-loop. Similarly, the cell volumes $|K|$ and the cell centres x_K are also computed locally for all $K \in \overline{\mathcal{M}^p}$ and for each process $p \in \mathcal{P}$.

2.4 Assembly of the linear system and linear solver

Thanks to our choice of the overlap, and to the above computation of the tranmissivities, cell volumes and cell centres for all cells $K \in \overline{\mathcal{M}}^p$, the following subset of the set of equations (7) is assembled on each process $p \in \mathcal{P}$ without any communications between processes:

$$-\sum_{K\in\mathcal{M}_v}\sum_{v'\in\mathcal{V}_K} T_K^{v,v'}(u_K^{(n)} - u_{v'}^{(n)}) = 0 \qquad \text{for all } v \in \mathcal{V}^p\backslash\partial\Omega, \tag{14a}$$

$$u_v^{(n)} = 0 \qquad \text{for all } v \in \mathcal{V}^p \cap \partial\Omega, \tag{14b}$$

$$\frac{|K|}{\delta t}(u_K^{(n)} - u_K^{(n-1)}) + \sum_{v\in\mathcal{V}_K}\sum_{v'\in\mathcal{V}_K} T_K^{v,v'}(u_K^{(n)} - u_{v'}^{(n)}) = |K| f(x_K) \qquad \text{for all } K \in \overline{\mathcal{M}}^p. \tag{14c}$$

Note that it includes the equations of all own vertices $v \in \mathcal{V}^p$, and the equations of all own and ghost cells $K \in \overline{\mathcal{M}}^p$. This set of equations can be rewritten as the following linear system (the time superscript (n) is removed for the sake of clarity),

$$\begin{pmatrix} A^p & B^p \\ C^p & D^p \end{pmatrix} \begin{pmatrix} U_{\overline{\mathcal{V}}^p} \\ U_{\overline{\mathcal{M}}^p} \end{pmatrix} = \begin{pmatrix} 0 \\ b^p \end{pmatrix},$$

where $U_{\overline{\mathcal{V}}^p} \in \mathbb{R}^{\overline{\mathcal{V}^p}}$, $U_{\overline{\mathcal{M}}^p} \in \mathbb{R}^{\overline{\mathcal{M}}^p}$ denote the vectors of discrete unknowns at vertices and cells, $b^p \in \mathbb{R}^{\overline{\mathcal{M}}^p}$ the right hand side, and where the matrices have the following sizes:

$$\begin{aligned} &A^p \in \mathbb{R}^{\mathcal{V}^p} \otimes \mathbb{R}^{\overline{\mathcal{V}^p}}, \\ &B^p \in \mathbb{R}^{\mathcal{V}^p} \otimes \mathbb{R}^{\overline{\mathcal{M}}^p}, \\ &C^p \in \mathbb{R}^{\overline{\mathcal{M}}^p} \otimes \mathbb{R}^{\overline{\mathcal{V}^p}}, \\ &D^p \in \mathbb{R}^{\overline{\mathcal{M}}^p} \otimes \mathbb{R}^{\overline{\mathcal{M}}^p}. \end{aligned} \tag{15}$$

Moreover, since the block D^p is a non singular diagonal matrix, the Schur complement can be easily computed without fill-in to reduce the linear system to the vertices unknowns only as mentioned in the presentation of the VAG scheme in the subsection 1.2. We obtain

$$U_{\overline{\mathcal{M}}^p} = (D^p)^{-1}(b^p - C^p U_{\overline{\mathcal{V}}^p}), \tag{16}$$

and the Schur complement system

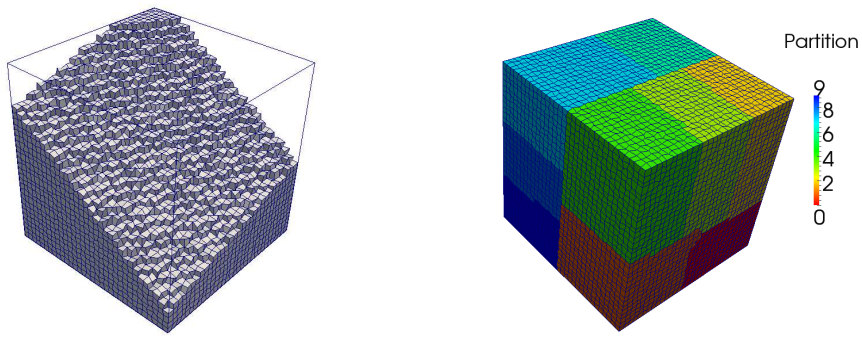
$$(A^p - B^p(D^p)^{-1}C^p)U_{\overline{\mathcal{V}}^p} = -B^p(D^p)^{-1}b^p. \tag{17}$$

Finally on each process $p \in \mathcal{P}$, we construct the matrix $(A^p - B^p(D^p)^{-1}C^p) \in \mathbb{R}^{\mathcal{V}^p} \otimes \mathbb{R}^{\overline{\mathcal{V}^p}}$. This computation requires no MPI communications thanks to our choice of the overlap. It is transferred line by line to PETSc. Next, the parallel linear system obtained from (17) for all $p \in \mathcal{P}$, is solved using the GMRES PETSc Krylov linear solver with a block Jacobi ILU2 preconditioner. PETSc provides the solution vector $U_{\mathcal{V}^p} \in \mathbb{R}^{\mathcal{V}^p}$ at own vertices for each process $p \in \mathcal{P}$. The solution at own and ghost vertices $U_{\overline{\mathcal{V}}^p}$, $p \in \mathcal{P}$ is obtained by a synchronization of the vertices, and the solution at own and ghost cells $U_{\overline{\mathcal{M}}^p}$ is computed from (16) locally on each process $p \in \mathcal{P}$.

3 Numerical validation on a parabolic model

In this section, the strong scalability of ComPASS is studied on the discrete problem described in subsection 1.2 using the parallelization of the algorithm described in section 2. The mesh is a topologically Cartesian grid of size fixed to 128^3 cells, and the code is run on a number of processes ranging from 1 to 500. To be more specific, two types of grid have been tested. The first one is a Cartesian mesh and the second one is obtained

from the previous one by a random perturbation of its nodes leading to hexahedral cells with non planar faces. For the sake of clarity, a coarser randomly perturbed grid of size 32^3 is exhibited on the Figure 8(a), and its partitioning is presented on the Figure 8(a).



(a) Randomly pertubated topologically Cartesian grid

(b) Cell partitioning with $\mathrm{card}(\mathcal{P}) = 10$

Figure 8: Randomly perturbed topologically Cartesian grid type used for the numerical validation.

In this test case, the parabolic equation is solved on the unit cube $\Omega = [0, 1]^3$, with Dirichlet boundary condition on $\partial\Omega$, and an initial condition both defined by the function

$$u(x, y, z) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z) + 4.$$

The time interval is given by $[0, 0.05]$ and is discretized using a constant time step of 0.001.

This numerical example has been run using the OpenMPI library version 1.4.5 on the *meso-centre* of Aix-Marseille University which provides a cluster architecture of 96 nodes each equipped with 12 cores and connected by an InfiniBand QDR communications link (see [2] for more details).
The Figure 9 exhibits the speed-up obtained with both types of grids with the following splitting of the run times:

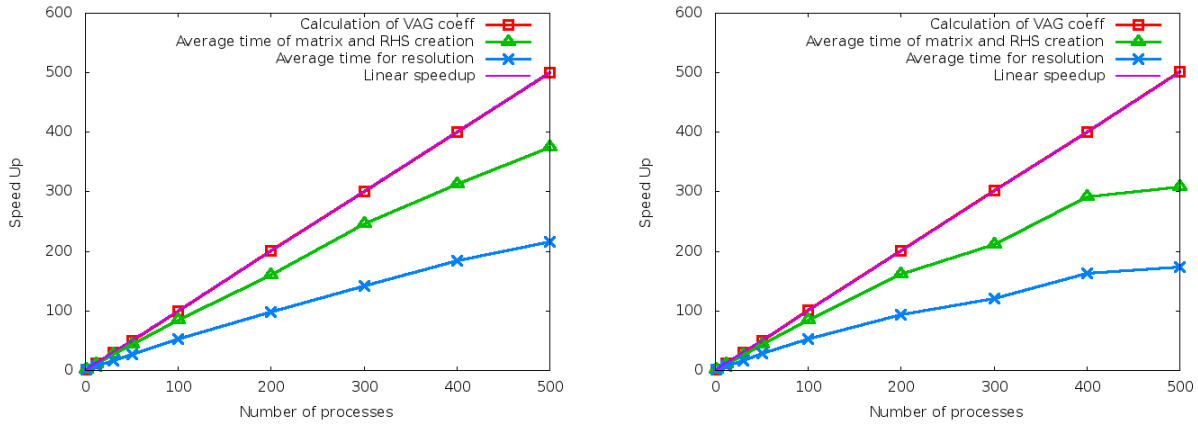
(i) computation of the VAG transmissivities as discussed in the subsection 2.3,

(ii) creation of the PETSc objects, assembly of the matrix and of the right-hand side (denoted RHS), and computation of the Schur complement as described in the subsection 2.4,

(iii) computation of the discrete solution, including both the call to the linear solver of PETSc, and the calculation of the cell unknowns using (16).

We observe on the Figure 9 that the computation of the VAG transmissivities (i) satisfies as expected the optimal linear speed-up since all the computations are done without communication between the processes. For the two other steps (ii) and (iii), the parallel efficiency decreases clearly with the number of processes. This can be explained by the cost of the communications needed for each step. Indeed, for step (ii), one synchronization of the global index of the local vertices is required for the PETSc objects creation, and the synchronization of the discrete solution at ghost vertices is required at the end of each time step. For step (iii), the linear solver involves communications for the matrix vector product and for the scalar products at each solver iteration. It results that for that type of implicit algorithm, the speed up can only be optimal for roughly

speaking a minimum of 50000 cells for each process. This explains the decrease of the parallel efficiency in our case over a few tens of processes. All together, the results of the two test cases exhibit a rather good strong scalability of the code ComPASS which validates our implementation.



(a) Cartesian grid

(b) Randomly perturbed grid

Figure 9: Speed-up study using two types of topologically Cartesian grids with 128^3 cells and up to 500 processes. Speed Up : $Sp_{(n\,proc)} = \text{Time}_{(1\,proc)}/\text{Time}_{(n\,proc)}$; linear Speed up $Sp_{(n\,proc)} = n$.

4 Conclusion and perspectives

The CEMRACS has offered an ideal framework to start in a collaborative project the development of the new code ComPASS adapted to the distributed parallelization of Finite Volume schemes on general unstructured meshes using various d.o.f. including typically cell, face or vertex unknowns. The parallelization is based on a classical domain decomposition approach with one layer of cells as overlap allowing to assemble the systems locally on each process. The code is adapted to fully implicit time integrations of evolution problems thanks to the connection with the PETSc library in order to solve the linear systems.
To simplify the implementation, at least in a first step, it has been decided to read and partition the mesh on a master process which distributes it to all processors in a preprocessing step. All the remaining computations are done in parallel. The code has been tested on a parabolic toy problem to evaluate its parallel efficiency on up to 500 processes. The results exhibit a rather good strong scalability of the CPU time with a typical behaviour of the parallel efficiency for implicit discretizations on a fixed mesh size.
This is clearly only the first step of development of a more advanced version of the code. In the near future, we intent first to improve the parallelization of the post-processing step of the code, and of course to include more physics starting with multiphase porous media flow applications.
Last but not least, this CEMRACS project has also offered to the youngest authors, the opportunity to learn MPI parallel programming and to work in a collaborative project using appropriate software engineering tools.

Acknowledgement

The authors would like to thank the organisers of CEMRACS 2012 and especially Pierre Navaro for its precious help in using the package PETSc and running our code on the cluster of Marseille.

References

[1] DUNE – Distributed and Unified Numerics Environment. http://www.dune-project.org/.

[2] Meso-centre of Aix-Marseille university. http://cbrl.up.univ-mrs.fr/~mesocentre/.

[3] Metis. http://glaros.dtc.umn.edu/gkhome/views/metis.

[4] PETSc. http://www.mcs.anl.gov/petsc.

[5] The Trilinos Project. http://www.http://trilinos.sandia.gov/.

[6] I. Aavatsmark, T. Barkve, O. Bøe, and T. Mannseth. Discretization on non-orthogonal, curvilinear grids for multi-phase flow. in prov. of the 4th european conf. on the mathematics of oil recovery, 1994.

[7] B. Andreianov, F. Hubert, and S. Krell. Benchmark 3d: a version of the ddfv scheme with cell/vertex unknowns on general meshes. 2011.

[8] K. Aziz and A Settari. *Petroleum reservoir simulation*. Applied Science Publishers, 1979.

[9] F. Brezzi, K. Lipnikov, and V. Simoncini. A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 15(10):1533–1552, 2005.

[10] K.H. Coats. Implicit compositional simulation of single-porosity and dual-porosity reservoirs. In *SPE Symposium on Reservoir Simulation*, 1989.

[11] M.G. Edwards and C.F. Rogers. A flux continuous scheme for the full tensor pressure equation. in prov. of the 4th european conf. on the mathematics of oil recovery, 1994.

[12] R. Eymard, P. Féron, T. Gallouët, C. Guichard, and R. Herbin. Gradient schemes for the stefan problem. *submitted*, 2012. preprint http://hal.archives-ouvertes.fr/hal-00751555.

[13] R. Eymard, T. Gallouët, and R. Herbin. Discretisation of heterogeneous and anisotropic diffusion problems on general non-conforming meshes, sushi: a scheme using stabilisation and hybrid interfaces. *IMA J. Numer. Anal.*, 30(4):1009–1043, 2010.

[14] R. Eymard, C. Guichard, and R. Herbin. Small-stencil 3d schemes for diffusive flows in porous media. *M2AN Math. Model. Numer. Anal.*, 46:265–290, 2012.

[15] R. Eymard, C. Guichard, R. Herbin, and R. Masson. Vertex-centred discretization of multiphase compositional darcy flows on general meshes. *Computational Geosciences*, pages 1–19, 2012.

[16] R. Eymard, G. Henry, R. Herbin, F. Hubert, R. Klöfkorn, and G. Manzini. 3D benchmark on discretization schemes for anisotropic diffusion problems on general grids. *Finite Volumes for Complex Applications VI Problems & Perspectives*, pages 895–930, 2011.

[17] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

[18] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message passing interface*, volume 1. MIT press, 1999.

[19] Gilles Grospellier and Benoit Lelandais. The arcane development framework. In *Proceedings of the 8th workshop on Parallel/High-Performance Object-Oriented Scientific Computing*, POOSC '09, pages 4:1–4:11, New York, NY, USA, 2009. ACM.

[20] G.R. Luecke, Y. Zou, J. Coyle, J. Hoekstra, and M. Kraeva. Deadlock detection in mpi programs. *Concurrency and Computation: Practice and Experience*, 14(11):911–932, 2002.

[21] D.W. Peaceman. *Fundamentals of numerical reservoir simulation*, volume 6. Elsevier, 1977.