Algebraic Domain Decomposition Methods for Highly Heterogeneous Problems

Pascal Have, Roland Masson, Frédéric Nataf, Mikolaj Szydlarski, Hua Xiang, Tao Zhao

Algebraic Domain Decomposition Methods for Highly Heterogeneous Problems

Pascal Havé * Roland Masson † Frédéric Nataf ‡
Mikolaj Szydlarski § Hua Xiang ¶ Tao Zhao ‖

February 19, 2013

Abstract

We consider the solving of linear systems arising from porous media flow simulations with high heterogeneities. Using a Newton algorithm to handle the non-linearity leads to the solving of a sequence of linear systems with different but similar matrices and right hand sides. The parallel solver is a Schwarz domain decomposition method. The unknowns are partitioned with a criterion based on the entries of the input matrix. This leads to substantial gains compared to a partition based only on the adjacency graph of the matrix. From the information generated during the solving of the first linear system, it is possible to build a coarse space for a two-level domain decomposition algorithm that leads to an acceleration of the convergence of the subsequent linear systems. We compare two coarse spaces: a classical approach and a new one adapted to parallel implementation.

Contents

*IFPEN, pascal.have@ifpenergiesnouvelles.fr
†IFPEN, roland.masson@ifpenergiesnouvelles.fr
‡Laboratoire J.L. Lions (LJLL), University of Paris VI, nataf@ann.jussieu.fr
§IFPEN and LJLL, University of Paris VI, mikolaj.szydlarski@gmail.com
¶School of Mathematics and Statistics, Wuhan University. Partly supported by the National Natural Science Foundation of China under grant 10901125, hxiang@whu.edu.cn
‖LJLL, University of Paris VI, zhao@ann.jussieu.fr, supported by ANR Petalh

1 Introduction

Multiphase, compositional porous media flow models, used for example in reservoir simulations or basin modeling, lead to the solution of complex nonlinear systems of Partial Differential Equations (PDEs) accounting for the mass conservation of each component and the multiphase Darcy law, coupled with thermodynamical equilibrium and volume balance closure laws. These PDEs are typically discretized using a cell-centered finite volume scheme and a fully implicit Euler integration in time in order to allow for large time steps. After Newton type linearization, one ends up with the solution of a linear system at each Newton iteration which represents up to 90 percent of the total simulation elapsed time.

Such linear systems couple an elliptic (or parabolic) unknown, the pressure, and hyperbolic (or degenerate parabolic) unknowns, the volume or mass fractions. They are nonsymmetric, and ill-conditioned in particular due to the elliptic part of the system, and the strong heterogeneities and anisotropy of the media. Their solution using an iterative Krylov method, such as GMRES or BiCGStab, requires the construction of an efficient preconditioner which should be scalable with respect to the heterogeneities and anisotropy of the media, the mesh size, and the number of processors, and should cope with the coupling of the elliptic and hyperbolic unknowns.

Nested factorization [AC83] and CPR-AMG [LVW01], [SMW03] are the main state of the art preconditioners currently used in industrial reservoir simulators. Nevertheless they still suffer from major drawbacks that should be addressed in response to the evolution of the computing architectures, and the demand for more complex physics and geology in reservoir and basin simulations. The nested factorization preconditioner is not adapted to distributed architectures and has a limited scalability with respect to the heterogeneities and anisotropy of the media. CPR-AMG is a more recent preconditioner which combines multiplicatively a parallel Algebraic MultiGrid preconditioner for a pressure block of the linear system, with a parallel incomplete factorization preconditioner (typically ILU0) for the full system. This preconditioner exhibits a very good robustness with respect to the heterogeneities and anisotropy of the media. However, its parallel scalability requires a very large number of cells per processor, say 100000, due to the coarsening step of AMG which is not strongly scalable. The robustness of the CPR-AMG also requires the definition of a pressure block which should be close to an M-matrix. This induces a sensitivity to the physics such as complex wells couplings, capillary driven flows or strong nonlinearities in pressure of the thermodynamical closure laws, and also a sensitivity to the distorsion of the mesh when multipoint flux approximations are used for the Darcy fluxes discretization.

Solving these drawbacks motivates our research for an alternative pressure block preconditioner which should remain scalable with respect to the heterogeneities and anisotropy of the media and should exhibit improved strong scalability on massively distributed architectures and improved robustness with respect to the physics and to the discretization.

The pressure block matrix is related to the discretization of a Darcy equation with high contrasts and anisotropy in the coefficients. We work in the context of overlapping Schwarz type methods on parallel computers. In order to deal with anisotropy, we force the domain decomposition to respect the strong connections between the nodes as much as possible. This is inspired by coarsening techniques in algebraic multigrid (AMG) methods. It is also well known that efficient domain decomposition methods demand a well suited coarse grid correction. For matrices arising from scalar problems with smooth coefficients, it is possible to build *a priori* (i.e. before any linear solve) efficient coarse spaces based on domain wise constant vectors, see[TW05] and references therein. For problems with high heterogeneities, the numerical computation of these coarse spaces is often based on solving generalized eigenvalue problems in subdomains, see [EGW11] and [NXD10]. The corresponding local matrices are not submatrices of the global matrix A and cannot be built at the algebraic level. In this paper, we introduce a new algebraic coarse space construction based on an analysis of the Krylov space generated by the linear solve at the first iteration of a Newton-Ralphson algorithm. We take advantage of a parallel implementation to build a richer coarse space than the one proposed in[RR00, RR98].

The paper is organized as follows. In section 2, we recall the basis of the overlapping Schwarz method. In the next section, we show the benefit of an AMG style partitioning. In section 4 we consider coarse grid corrections. In § 4.1 we recall the principles of coarse grid correction in the context of Newton-Ralphson method, and in § 4.2 we introduce a new domain wise split coarse space and give numerical results on problems arising from reservoir simulations.

2 Parallel Schwarz method

The discretization of a linear partial differential equation on a domain Ω yields a linear system of the form

$$A\,u = f \in \mathbb{R}^n \tag{1}$$

that we solve using a domain decomposition method. Without loss of generality, we consider here a decomposition of a domain Ω into two overlapping subdomains Ω_1 and Ω_2. The overlap is denoted by $O := \Omega_1 \cap \Omega_2$. This yields a partition of the domain: $\bar{\Omega} = \bar{\Omega}_I^{(1)} \cup \bar{O} \cup \bar{\Omega}_I^{(2)}$ where $\Omega_I^{(i)} := \Omega_i \backslash \bar{O}$, $i = 1, 2$. At the algebraic level this corresponds to a partition of the set of indices $\mathcal{N}$ into three sets: $\mathcal{N}_I^{(1)}$, O and $\mathcal{N}_I^{(2)}$.

After the discretization of the boundary value problem defined in domain Ω, we obtain a linear system of the following form

$$A\ u := \begin{bmatrix} A_{II}^{(1)} & A_{IO}^{(1)} & \\ A_{OI}^{(1)} & A_{OO} & A_{OI}^{(2)} \\ & A_{IO}^{(2)} & A_{II}^{(2)} \end{bmatrix} \begin{bmatrix} u_I^{(1)} \\ u_O \\ u_I^{(2)} \end{bmatrix} = \begin{bmatrix} f_I^{(1)} \\ f_O \\ f_I^{(2)} \end{bmatrix}. \tag{2}$$

We can also define the extended linear system by duplicating the variables

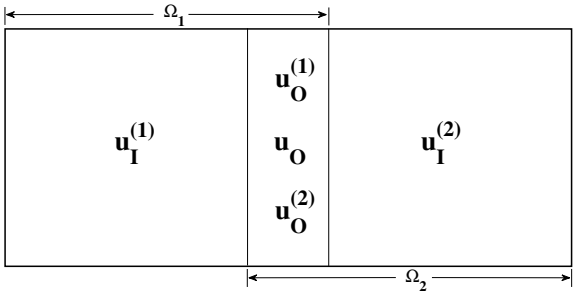

Figure 1: Decomposition into two overlapping subdomains.

located in the overlapping region

$$\tilde{A}\ \tilde{u} := \begin{bmatrix} A_{II}^{(1)} & A_{IO}^{(1)} & & \\ A_{OI}^{(1)} & A_{OO} & & A_{OI}^{(2)} \\ A_{OI}^{(1)} & & A_{OO} & A_{OI}^{(2)} \\ & & A_{IO}^{(2)} & A_{II}^{(2)} \end{bmatrix} \begin{bmatrix} u_I^{(1)} \\ u_O^{(1)} \\ u_O^{(2)} \\ u_I^{(2)} \end{bmatrix} = \begin{bmatrix} f_I^{(1)} \\ f_O \\ f_O \\ f_I^{(2)} \end{bmatrix}, \tag{3}$$

where the subscript 'O' stands for 'overlap', $u_O^{(i)}$ are the duplicated variables in the overlapping domain O, $u_I^{(i)}$ are variables in the subdomain Ω_I^i. It is easy to check that if A_{OO} is invertible, there is an equivalence between problems (2) and (3).

We introduce now three preconditioners, two for the linear system (2) and one for the extended linear system (3). A classical preconditioner to problem (2) is the additive Schwarz method. Let R_j be the rectangular restriction matrix to subdomain Ω_j, $j = 1, 2$. The additive Schwarz preconditioner is:

$$M_{ASM}^{-1} := R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2$$

where $A_i := R_i A R_i^T$, $i = 1, 2$. If A is SPD, then M_{ASM}^{-1} is SPD as well and the theory of this preconditioner is very well developped, see[TW05] and references therein. But in the overlap, corrections are added twice. This somehow delays the convergence. In order to fix this problem, another classical preconditioner was designed: the restricted additive Schwarz (RAS) method, see [CS99]. Define $\tilde{R}_j$ by setting some ones in R_j to zeros, such that the operators $\tilde{R}_j$ correspond to a non-overlapping decomposition,

$$\tilde{R}_1^T R_1 + \tilde{R}_2^T R_2 = I. \tag{4}$$

Then the restricted additive Schwarz preconditioner reads

$$M_{RAS}^{-1} := \tilde{R}_1^T A_1^{-1} R_1 + \tilde{R}_2^T A_2^{-1}\, R_2\,. \tag{5}$$

Note that the RAS avoids extra correction in the overlapping zone but at the expense of the loss of the symmetry of the preconditioner. As a result, there

are less theoretical results for RAS than for ASM. In practice, it was noticed that the RAS preconditioner outperforms the ASM preconditioner. The RAS preconditioner leads to an iterative method that has equivalences with the discretization of the original Schwarz method [Sch70], see [EG03].

The third Schwarz type method is called the Jacobi-Schwarz method (JSM). It corresponds to the discretization of the parallel variant of the original Schwarz algorithm [Sch70]. It consists of solving the extended problem (3) preconditioned by a block Jacobi method:

$$M_{JSM}(\widetilde{A}) := \begin{bmatrix} A_{II}^{(1)} & A_{IO}^{(1)} & & \\ A_{OI}^{(1)} & A_{OO} & & \\ & & A_{OO} & A_{OI}^{(2)} \\ & & A_{IO}^{(2)} & A_{II}^{(2)} \end{bmatrix}. \tag{6}$$

and one can easily notice that $M_{JSM}^{-1}(\widetilde{A})$ can be computed in parallel. Actually, it is sufficient to factorize the diagonal blocks in parallel. When used in a Richardson algorithm, it was proved in [EG03] that $M_{JSM}(\widetilde{A})$ applied to (3) and M_{RAS} applied to (2) lead to equivalent algorithms. Since RAS is easier to implement than JSM, it gives a clear advantage to RAS. But when considering two level methods, JSM has the advantage that no partition of unity is needed. It brings some benefit both in terms of implementation and efficiency as it was noticed in [NXDS11]. Thus, in the sequel, we will only consider the JSM preconditioner $M_{JSM}^{-1}(\widetilde{A})$ applied to (3).

In the general case with many subdomains, the set of indices $\mathcal{N}$ is decomposed into N overlapping subsets

$$\mathcal{N} = \cup_{i=1}^{N} \mathcal{N}_i \,.$$

From this decomposition, we define the extended system whose right handside belongs to $\mathbb{R}^{N_E}$ where $N_E := \sum_{i=1}^{N} \#\mathcal{N}_i > n$. For $i = 1, \ldots, N$ we denote by $R_{E,i}$ the boolean matrix corresponding to the restriction operator from $\mathbb{R}^{N_E} \mapsto \mathbb{R}^{N_i}$:

$$R_{E,i} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{7}$$

where there is 1 on a column if the corresponding node belongs to i-th subdomain. The transpose operator $R_{E,i}^T$ is the extension by zero from the i-th subdomain to global set of unknowns $\mathbb{R}^{N_E}$. In practice, we first perform a partition of the unknowns using a graph partitioner working on the adjacency graph of the matrix. Then each partition is extended with the adjacent nodes.

3 Partitioning

Graph partitioning softwares such as METIS [KK99] or SCOTCH [CP08] are commonly used with the goal to compute a balanced partitioning that minimizes a cost function which for sake of efficiency has to take integer values only. The default cost function is the number of edge cuts. In order to accommodate the heterogeneous cases, it is possible to weight the costs of the edgecuts. Here,

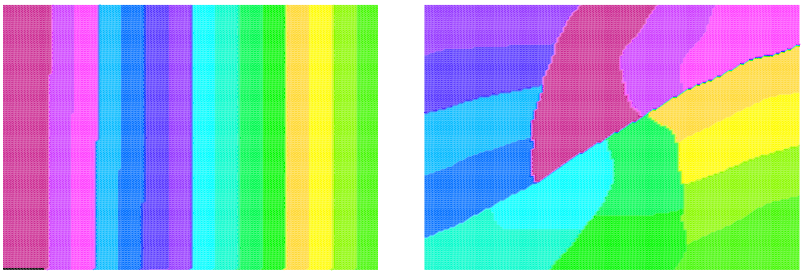

Figure 2: Weighted vs. un-weighted partitions for a problem with a strong anisotropy (see equation (9))

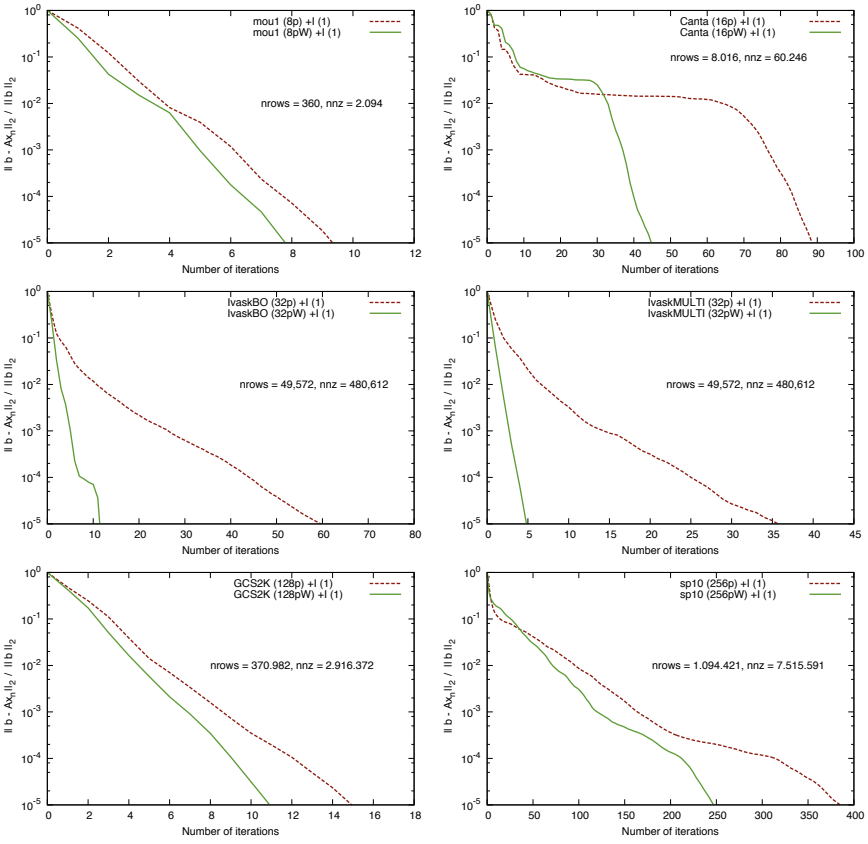

Figure 3: Convergence curves for weighted vs. un-weighted partitions for reservoir oil simulations

we want to get a partition that follows the anisotropy of the underlying partial differential equation (PDE). It is known to have a strong effect on the efficiency of preconditioners. Since we work at the algebraic level, this partitioning of the unknowns cannot be made using the underlying mesh and values of the parameters in the PDE. The only information on the problem comes from the entries of the matrix of the linear system. Inspired by algebraic multigrid methods, the weight of the edge (i, j) which has to be an integer is defined as follows:

$$\text{weight}(i,j) := \left\lfloor \gamma \frac{|a_{i,j}|}{|a_{i,i}| + |a_{j,j}|} \right\rfloor \tag{8}$$

where γ is a large number. When minimizing the edge-cuts with weights, nodes that are strongly connected are kept in the same subdomain as much as possible. In order to choose the value for γ, we have to balance two requirements. On one hand, γ must be small enough so that the value of the cost function which is the sum of the weighted edge cuts is smaller than the maximum value of an integer allowed by the partitioning software. On the other hand, γ must be large enough so that the weights take as many different values as possible in order to accurately distinguish between the different values of $|a_{i,j}|/(|a_{i,i}| + |a_{j,j}|)$. Typically in our tests $\gamma = 80000$.

As an example, we consider the following anisotropic problem:

$$-div(\kappa \nabla \mathbf{u}) = f\,,$$

discretized by a finite element method (FreeFem++, see [Hec10]) on 2D unit square of size $N_x \times N_y$, where $N_x = N_y = 128$ so that the number of non zero entries is 98000. The diffusion tensor κ has a strong anisotropy:

$$\kappa = \begin{bmatrix} \kappa_{xx} & 0 \\ 0 & \kappa_{yy} \end{bmatrix} = \begin{bmatrix} 1 \times 10^{-6} & 0 \\ 0 & 1 \end{bmatrix}. \tag{9}$$

In Figure 2, we show on the left picture the partition obtained using the weight function of formula (8). We see that strongly connected nodes are kept as much as possible in the same domain. On the right picture, we show the unweighted partition. The problem with this partition is that strongly connected nodes belong to more than one subdomain. As a result, the Jacobi-Schwarz method converges in only 2 iterations as compared to 106 iterations for the unweighted partition. The effect of the weighted partition is now tested against an un-weighted partition on several test cases coming from "real life" reservoir simulations with problems of various sizes, see Figure 3. The first figure corresponds to 50.000 nodes whereas the last figure corresponds to the SPE10 test case with one million unknowns for the pressure block. On Figure 3, we plot the convergence histories for a weighted partition and an un-weighted one. As we see, the weighted partitioning yields a better convergence of the Schwarz method. When the iteration counts are already small (smaller than 20) for the un-weighted partition, there is no much improvement. But, when iterations counts are large, the weighted partition brings a substantial benefit. We also noticed that the cost of one iteration and the time required for partitioning are about the same in both cases.

4 Two-level Schwarz method

4.1 Background

In the previous section, we have seen the benefit of using weighted partitions that take into account the anisotropy of the problem. When the number of subdomains becomes large, this is not sufficient to prevent stagnation in the convergence of Schwarz type algorithms. Indeed, any of the three preconditioners M_{ASM}, M_{RAS} or M_{JSM} removes the very large eigenvalues of the coefficient matrix, which correspond to high frequency modes. But the small eigenvalues still exist and hamper the convergence. These small eigenvalues correspond to low frequency modes and represent certain global information. We need a suitable coarse space to efficiently deal with them. This problem is closely related to deflation techniques, see for instance [TNVE09], [PdSM$^+$06] or [SYEG00] and references therein. These methods are based on a knowledge of an approximation to the eigenvectors corresponding to the "bad" (small in our case) eigenvalues. Let Z denote the rectangular matrix that stores these vectors columnwise. The number of columns of Z is the size of the coarse space. It is then classical (see for instance [TNVE09]) to define the following matrices:

$$P := I - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ\,.$$

Notice that if A is symmetric, we have $QAZ = Z$, $P^T Z = 0$ and $P^T Q = 0$. From these matrices, it is possible to introduce new preconditioners which will not suffer from plateaus in the convergence. Let M denote a Schwarz method, the balancing Neumann-Neumann preconditioner

$$\mathcal{P}_{BNN} := P^T M^{-1} P + Q$$

was introduced in[Man93]. In [TNVE09], a related form is introduced:

$$\mathcal{P}_{A-DEF2} := P^T M^{-1} + Q\,. \tag{10}$$

These preconditioners used in any Krylov method have the interesting property that at any step n the residual r_n remains orthogonal to the vector space spanned by the columns of Z:

$$Z^T r_n = 0\,.$$

Compared to the original preconditioner M, the extra cost of the new preconditioner lies in the solving of a linear system with the small matrix E. In domain decomposition methods, the resulting method is called a two-level method.

The ideal coarse space is the invariant subspace corresponding to the low part of the spectrum of the preconditioned operator. But, computing the small eigenvalues of the preconditioned operator $M^{-1}A$ is of course too expensive. It is thus necessary to find a way to "guess" some good approximations to them: *a priori* i.e. before any solve or *a posteriori* i.e. after a first linear solve with the matrix A or with a matrix close to A.
The *a priori* construction demands some analytic knowledge on the problem to be solved. For instance for Poisson or elasticity type problems, coarse spaces must contain the kernel of the operators: constant functions or rigid body motions, see [TW05] and references therein. For domain decomposition methods for

problems with high heterogeneities, the numerical computation of these coarse spaces is often based on solving generalized eigenvalue problems in subdomains, see [EGW11] and [NXD10]. The corresponding local matrices are not submatrices of the global matrix A. They cannot be built at the algebraic level.
Thus, we focus on *a posteriori* constructions of the coarse space that can be done at the algebraic level. In order to build suitable coarse spaces, we will reuse information coming from previous solves in the context of nonlinear problems. Let F be a nonlinear mapping and suppose we solve

$$F(U) = G$$

by a Newton algorithm

$$\begin{array}{ll} \text{For } k & = 1, \ldots, K \\ & F'(u^k) \cdot \delta u^{k+1} = G - F(u^k) \\ & u^{k+1} = u^k + \delta u^{k+1} \end{array} \tag{11}$$

where K is the number of iterations to reach convergence. Let us denote by A_k the matrix $F'(u^k)$. The sequence of linear systems (11) are solved by the parallel Schwarz method. The idea is to use spectral information from this first solve with the matrix A_1 to build a coarse space Z and thus a coarse correction that will accelerate the solve of the subsequent linear systems with matrices A_k, $k \geq 2$. Although there are many variants (see for instance [RR98, RR00, GR03], [PdSM$^+$06] or [SYEG00] and references therein), we can say that the principle is to proceed in the following way. From the solve of the first linear system $A_1\, u_1 = f_1$, a spectral analysis of the Krylov subspace enables the selection of a suitable coarse space denoted Z. More precisely, Ritz eigenvalues associated with the Krylov subspace generated by the solving of the first linear system $A_1\, \delta u^1 = \ldots$ by the Krylov method preconditioned by M_{JSM} are computed. If it exists, the long plateau in the convergence is due to the presence of a few small positive eigenvalues close to zero. By computing the Ritz eigenvectors, we get an approximation to the corresponding eigenvectors. We select the $nsmeig$ small Ritz eigenvectors, denoted $(z_1, \ldots, z_{nsmeig})$, whose real parts of the eigenvalues are smaller than a given threshold ϵ_{eig}. A rectangular matrix $Z := |z_1|z_2|\ldots|z_{nsmeig}|$ is formed out of them. Then, it is used to build a two-level preconditioner for solving the subsequent linear systems $A_k\, u_k = f_k$, $k \geq 2$.

4.1.1 Retrieving approximate eigenvector from GMRES solver

We detail the Ritz eigencomputations after the solve of the first linear system with matrix $\widetilde{A}_1$ and preconditioner $M_1 := M_{JSM}^{-1}(\widetilde{A}_1)$ into m iterations with a GMRES algorithm. The final Krylov subspace associated with the preconditioned operators reads

$$\mathcal{K}_m(B_1, r_0) = \text{SPAN}\left\{r_0, B_1 r_0, B_1^2 r_0, \ldots, B_1^{m-1} r_0\right\} \tag{12}$$

with $B_1 := M_1^{-1}\, \widetilde{A}_1$. The computational kernel of GMRES [SS86] is the Arnoldi process which computes the orthonormal basis $W_m = |w_1\,|w_2\,|\ldots|\,w_m|$ for the Krylov space $\mathcal{K}_m(B_1, r_0)$. In the orthogonalisation process the scalars h_{ij} are

computed so that the square upper Hessenberg matrix $H_m \in \mathbb{R}^{m\times m}$ satisfies the fundamental relation

$$B_1 W_m = W_m H_m + h_{m+1,m} w_{m+1} e_m^H = W_{m+1} \overline{H}_m. \tag{13}$$

The rectangular upper Hessenberg matrix $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$ is the square upper Hessenberg matrix H_m supplemented with an extra row $(0, \ldots, 0, h_{m+1,m})$. From (13) we can derive the following expression for H_m:

$$H_m = W_m^H B_1 W_m. \tag{14}$$

The eigenvalues of H_m are called Ritz values and they approximate the eigenvalues of B_1. We approximate eigenvectors of B_1 by first computing eigenpairs (t_i, λ_i) of matrix H_m and then compute

$$z_i := W_m t_i \tag{15}$$

which is close to an eigenvector of B_1, for the same eigenvalue λ_i. Indeed, we left multiply

$$W_m^H B_1 W_m t_i = \lambda_i t_i$$

by W_m and get

$$W_m W_m^H B_1 W_m t_i \simeq B_1 W_m t_i = \lambda_i W_m t_i \,.$$

4.2 A sparse coarse space and the two-level preconditioner

4.2.1 Two coarse spaces

In this paper, we shall focus on the JSM method since there is no need to build a discrete partition of unity as in (4). We use spectral information from the first solve in order to build a coarse space for the subsequent solves. More precisely, from the first solve we select *nsmeig* Ritz eigenvectors corresponding to the smallest Ritz eigenvalues. The rectangular matrix $Z := |z_1|z_2|\ldots|z_{nsmeig}|$ stores these Ritz eigenvectors columnwise. Next, we introduce a larger coarse space Z^s whose columns are defined by:

$$Z^s_{i+N(j-1)} := R_{E,i}^T \, R_{E,i} \, z_j, \text{ for } 1 \le i \le N, \ 1 \le j \le nsmeig \,. \tag{16}$$

where $R_{E,i}$ is the restriction operator to the i-th subdomain, see (7) and z_j is the j-th column of Z. The coarse space Z^s is N times as large as Z, $dim(Z^s) = N \times nsmeig$. It has a very sparse structure. For instance, for a three subdomain decomposition and $nsmeig = 2$, the transformation of Z into Z^s has the following form:

$$Z = \left[\begin{array}{c|c} z_{11} & z_{12} \\ z_{21} & z_{22} \\ z_{31} & z_{32} \end{array}\right] \mapsto Z^s = \left[\begin{array}{ccc|ccc} z_{11} & 0 & 0 & z_{12} & 0 & 0 \\ 0 & z_{21} & 0 & 0 & z_{22} & 0 \\ 0 & 0 & z_{31} & 0 & 0 & z_{32} \end{array}\right]. \tag{17}$$

where $z_{ij} := R_{E,i} z_j$. If the Ritz eigenvectors are orthogonal, this is not necessarily the case for the columns of Z^s. In order to improve stability of the method, we orthogonalize Z^s and denote $Z^s_\perp$ the orthogonalized basis of the coarse space. Due to the sparse structure of $Z^s_\perp$, this process can be performed

in parallel, see § 4.2.2 for details.

We now define the coarse corrections that will be used to solve the subsequent linear systems with matrices $A_k,\ k \geq 2$. They are inspired by $\mathcal{P}_{A-DEF2}$ (10) but different. In order to ease notations in the definition of the two-level method, we note

$$M_k^{-1} := M_{JSM}^{-1}(\widetilde{A}_k)\,.$$

First of all, we build a coarse correction for the preconditioned system

$$B_k := M_k^{-1}\,\widetilde{A}_k\,. \tag{18}$$

Recall that if A_k is symmetric, M_k^{-1} is symmetric as well but the extended system $\widetilde{A}_k$ is not symmetric. As a result B_k is not symmetric and we first modify formula (10):

$$\mathcal{P}_k := P_k + Q_k\,, \tag{19}$$

where

$$P_k := I - Q_k B_k, \quad Q_k := Z_\perp^s E_k^{-1} {Z_\perp^s}^T, \quad E_k := {Z_\perp^s}^T B_k Z_\perp^s\,. \tag{20}$$

It can easily be checked that we have:

(a) $P_k = P_k^2$;

(b) $P_k Z_\perp^s = 0$, $P_k Q_k = 0$;

(c) $Q_k B_k = I - P_k, \quad Q_k B_k Z_\perp^s = Z_\perp^s, \quad Q_k B_k Q_k = Q_k$.

We have

Lemma 4.1 *The coarse correction $\mathcal{P}_k$ defined by* (19) *is invertible and has the following left-filtering property:*

$${Z_\perp^s}^T B_k = {Z_\perp^s}^T \mathcal{P}_k^{-1}$$

Proof. We first prove that $\mathcal{P}_k$ is one to one and thus invertible. Let u such that $\mathcal{P}_k\,u = P_k\,u + Q_k\,u = 0$. Then, we left multiply by P_k and use $P_k Q_k = 0$ and $P_k^2 = P_k$ to obtain $P_k\,u = 0$ and thus $Q_k u = 0$ as well. From $P_k\,u = 0$, we have $u = Z_\perp^s E_k^{-1} {Z_\perp^s}^T B_k u$. Let $w = E_k^{-1} {Z_\perp^s}^T B_k u$, we have $u = Z_\perp^s w$. Then, from $Q_k u = 0$, we get $Z_\perp^s E_k^{-1} {Z_\perp^s}^T Z_\perp^s w = 0$. We left multiply by ${Z_\perp^s}^T B_k$ to get ${Z_\perp^s}^T Z_\perp^s w = 0$. We take the scalar product with w and get $Z_\perp^s w = 0$ and so $u = 0$.
Let us prove now the filtering property. It is equivalent to $(P_k^T + Q_k^T) B_k^T Z_\perp^s = Z_\perp^s$. Consider first the term $P_k^T B_k^T Z_\perp^s$ and let's prove it is null:

$$\begin{aligned} P_k^T B_k^T Z_\perp^s &= B_k^T Z_\perp^s - B_k^T Q_k^T B_k^T Z_\perp^s \\ &= B_k^T Z_\perp^s - B_k^T Z_\perp^s ({Z_\perp^s}^T B_k^T Z_\perp^s)^{-1} {Z_\perp^s}^T B_k^T Z_\perp^s = 0 \end{aligned}$$

It remains to prove that $Q_k^T B_k^T Z_\perp^s = Z_\perp^s$:

$$Q_k^T B_k^T Z_\perp^s = Z_\perp^s ({Z_\perp^s}^T B_k^T Z_\perp^s)^{-1} {Z_\perp^s}^T B_k^T Z_\perp^s = Z_\perp^s$$

■

A first method would consist in using $\mathcal{P}_k$ as a preconditioner to matrix B_k or equivalently $\mathcal{P}_k M_k^{-1}$ as a preconditioner to the extended system $\widetilde{A}_k$ in a Krylov method. Remark that from the left filtering property, it is easy to check that for any Krylov method with right preconditioning and at any step m the residual r_m remains orthogonal to the vector space spanned by the columns of $Z_\perp^s$. More precisely, we have

Lemma 4.2 *Let x_0 be an initial guess and $r_0 := b - B_k x_0$ be the initial residual such that ${Z_\perp^s}^T r_0 = 0$. Let $\mathcal{K}_m(B_k\mathcal{P}_k, r_0)$ denote the Krylov space of dimension m, $\mathcal{K}_m(B_k\mathcal{P}_k, r_0) := Span\{r_0, \ldots, (B_k\mathcal{P}_k)^{m-1} r_0\}$.*
Then, for any $x_m \in \{x_0\} \oplus \mathcal{P}_k\mathcal{K}_m(B_k\mathcal{P}_k, r_0)$ we have

$$ {Z_\perp^s}^T (b - B_k x_m) = 0 \,. $$

Proof. First remark that solving $B_k x = b$ with $\mathcal{P}_k$ as a right preconditioner in a Krylov method amounts to solve $B_k\mathcal{P}_k y = b$ with a Krylov method and then apply $\mathcal{P}_k$ to the resulting approximation to y. Thus, at step m of the algorithm, the approximate solution $x_m \in \{x_0\} \oplus \mathcal{P}_k\mathcal{K}_m(B_k\mathcal{P}_k, r_0)$. That is,

$$ x_m = x_0 + \mathcal{P}_k \sum_{i=0}^{m-1} \alpha_i^m (B_k\mathcal{P}_k)^i r_0 $$

for some $(\alpha_i^m)_{0\le i\le m-1} \in \mathbb{R}^m$. The corresponding residual reads

$$ r_m = b - B_k x_0 - \sum_{i=0}^{m-1} \alpha_i^m (B_k\mathcal{P}_k)^{i+1} r_0 = r_0 - \sum_{i=0}^{m-1} \alpha_i^m (B_k\mathcal{P}_k)^{i+1} r_0 \,. $$

It suffices to prove by induction that ${Z_\perp^s}^T (B_k\mathcal{P}_k)^i r_0$ for any $0 \le i \le m$. By assumption, it is true for $i = 0$. Assume it holds for some i then by using Lemma 4.1, we have

$$ {Z_\perp^s}^T (B_k\mathcal{P}_k)^{i+1} r_0 = {Z_\perp^s}^T B_k\mathcal{P}_k (B_k\mathcal{P}_k)^i r_0 = {Z_\perp^s}^T (B_k\mathcal{P}_k)^i r_0 = 0 \,. $$

This ends the proof of the lemma. Notice as well that the assumption ${Z_\perp^s}^T r_0$ is not difficult to satisfy. Indeed, let $\widetilde{x}_0$ be any initial guess, then $x_0 := \mathcal{P}_k(b + P_k^T\widetilde{x}_0)$ satisfies the assumption of the lemma:

$$ {Z_\perp^s}^T r_0 = {Z_\perp^s}^T (b - B_k\mathcal{P}_k(b + P_k^T\widetilde{x}_0)) = {Z_\perp^s}^T b - {Z_\perp^s}^T (b + P_k^T\widetilde{x}_0) = 0 \,, $$

where we have used once again Lemma 4.1 and also that $P_k Z_\perp^s = 0$. ■

Formula (19) demands the computation of matrix E_k via formula (20) before solving each linear system with matrix $\widetilde{A}_k$, $k \ge 2$. With a parallel implementation as explained in § 4.2.2 the cost is almost the same as *nsmeig* iterations of the one-level Schwarz method. When the size of the coarse space is large, the factorization of the coarse operator E_k can be costly. In order to avoid this factorization cost, we reuse matrix E_2 by simplifying formula (20) :

$$ P_{2,k} := I - Q_2 B_k, \quad Q_2 := Z_\perp^s E_2^{-1} {Z_\perp^s}^T, \quad E_2 := {Z_\perp^s}^T B_2 Z_\perp^s \,. \tag{21} $$

The new coarse correction that replaces (19) reads

$$ \mathcal{P}_{2,k} := P_{2,k} + Q_2 \,. \tag{22} $$

Operator $\mathcal{P}_{2,k}$ is a preconditioner to matrix B_k. In practice, we will use

$$C_k := \mathcal{P}_{2,k} M_k^{-1} = [(I_d + Z_\perp^s E_2^{-1} {Z_\perp^s}^T) - Z_\perp^s E_2^{-1} {Z_\perp^s}^T M_k^{-1} \widetilde{A}_k] \, M_k^{-1}$$

as a preconditioner to the extended system $\widetilde{A}_k$. A naive implementation would, at each iteration, demand that one-level Schwarz preconditioner M_k^{-1} be applied twice so that the cost of the two-level preconditioner would be at least double that of the one-level method. In order to avoid this, an equivalent definition is the following:

$$C_k = [(I_d + Z_\perp^s E_2^{-1} {Z_\perp^s}^T) - Z_\perp^s E_2^{-1} S_k^T \widetilde{A}_k] \, M_k^{-1} \tag{23}$$

where we precompute

$$S_k^T := {Z_\perp^s}^T M_k^{-1} \, . \tag{24}$$

As explained in § 4.2.2, the cost of precomputing S_k^T is about $nsmeig$ applications of the one-level Schwarz method.

The corresponding algorithm for solving a series of related linear systems is summed up as follows:

Algorithm 1 *For solving a series of linear systems $A_k u_k = f_k$, $1 \leq k \leq K$:*

1. *Solve system $\widetilde{A}_1 \tilde{u}_1 = \tilde{f}_1$ with GMRES method preconditioned by M_1.*
   Compute the Ritz eigenpairs (z_i, λ_i) for $|Real(\lambda_i)| < \epsilon_{eig}$, see § 4.1.1.
2. *Orthogonalize Z^s given by formula (16) into $Z_\perp^s$.*
   Compute E_2 via formula (21)
   Gauss factorization of E_2
3. *For $2 \leq k \leq K$*
   Compute S_k^T given by formula (24)
   Solve system $\widetilde{A}_k \tilde{u}_k = \tilde{f}_k$ with GMRES method preconditioned by C_k, formula (23).

Numerical results in § 4.2.3 will assert the efficiency of this approach.

4.2.2 Parallel data structure

In this section, we explain that due to a parallel implementation, provided the number of cores is equal to the number of MPI processes, the construction of the coarse space has the same cost as in [GR03]. The only difference is in the size of the matrix E, see § 4.2.1.

The method makes use of three distributed data structures:

- vector
- sparse matrix
- coarse space

Without loss of generality, we take examples for three MPI processes and a three subdomain decomposition. A vector $x \in \mathbb{R}^{N_E}$ is distributed among the three processes, process i ($1 \leq i \leq 3$) will own $R_{E,i}\, x$.
According to this vector distribution, the sparse matrix $\widetilde{A}$ is stored blockwise, $\widetilde{A} = (\widetilde{A}_{ij})_{1 \leq i,j \leq N}$, see [BFF+09] for a related data structure. For all $1 \leq i \leq N$ submatrices $(\widetilde{A}_{ij})_{1 \leq j \leq N}$ are owned by the i-th process. Note that since $\widetilde{A}$ is sparse, the diagonal blocks $\widetilde{A}_{ii}$, $1 \leq i \leq N$ are sparse as well and the off-diagonal blocks are hypersparse since they correspond to interactions between neighboring subdomains. Thus, the diagonal blocks are stored in a classical CSR format. The off-diagonal blocks are stored in a compressed sparse row and columns format which means only non zero rows and columns are stored in a CSR format, see [BG08] for a related data structure. The matrix-vector product is then naturally parallel.
The implementation of the coarse correction demands a distributed storage for sparse rectangular matrices such as $Z_\perp^s$ and S_k^T (see equation (24)). Without entering into details, our storage is similar to that of our sparse matrix. We give some details on the parallel implementation of the coarse space operations. The first one is to split the Ritz eigenvectors following equation (16). Actually, since the vectors are already distributed, it is just a matter of pointers management. The next operation is to orthogonalize the split coarse space Z^s into $Z_\perp^s$. Recall that even if the columns of Z are orthogonal that does not necessarily holds for Z^s. It is easy to confirm (see formula (17)) that it is sufficient to orthogonalize in parallel for $1 \leq i \leq N$ the sub-vectors z_{ij}, $1 \leq j \leq nsmeig$ and that there is no need for communication. Thus the cost of the parallel orthogonalizing of the split coarse space Z^s equals the sequential cost of orthogonalizing $nsmeig$ subvectors.

The next step in Algorithm 1 is to compute the matrix

$$E_2 := {Z_\perp^s}^T\, M_2^{-1}\, \widetilde{A}_2 Z_\perp^s = (M_2^{-1}\, Z_\perp^s)^T\, \widetilde{A}_2 Z_\perp^s\,.$$

We shall see that the cost to compute $F_2 := (M_2^{-1}\, Z)^T\, \widetilde{A}_2 Z$ for the classical coarse space is comparable to that of computing E_2, see § 4.2.3 for wall-clock measurements as well. For simplicity, we consider the example given by formula (17): two Ritz eigenvectors and three subdomains. Remember that M is a block diagonal preconditioner so that we have:

$$M_2^{-1}\, Z = \left[\begin{array}{cc} M_{11}^{-1}\, z_{11} & M_{11}^{-1}\, z_{12} \\ \hline M_{22}^{-1}\, z_{21} & M_{22}^{-1}\, z_{22} \\ \hline M_{33}^{-1}\, z_{31} & M_{33}^{-1}\, z_{32} \end{array}\right] \begin{array}{l} \leftarrow \texttt{Proc 1} \\ \leftarrow \texttt{Proc 2} \\ \leftarrow \texttt{Proc 3} \end{array}$$

and $M_2^{-1}\, Z_\perp^s$ is given by:

$$\left[\begin{array}{cc} M_{11}^{-1}\, z_{11_\perp} & M_{11}^{-1}\, z_{12_\perp} \\ \hline M_{22}^{-1}\, z_{21_\perp} & M_{22}^{-1}\, z_{22_\perp} \\ \hline M_{33}^{-1}\, z_{31_\perp} & M_{33}^{-1}\, z_{32_\perp} \end{array}\right] \begin{array}{l} \leftarrow \texttt{Proc 1} \\ \leftarrow \texttt{Proc 2} \\ \leftarrow \texttt{Proc 3} \end{array}$$

Clearly computational costs per process are the same, they consist of factorizing once the local contributions M_{ii} to subdomain $1 \leq i \leq N$ and then performing several backward-forward substitions. For this we use multithreaded direct

solvers inside an MPI process,. As for the computation of $\widetilde{A}Z$ and $\widetilde{A}Z_{\perp}^{s}$, we take the following example

$$\begin{aligned}\widetilde{A}\,Z &= \left[\begin{array}{cc|c} \widetilde{A}_{11} & & \widetilde{A}_{13} \\ \hline \widetilde{A}_{21} & \widetilde{A}_{22} & \\ \hline \widetilde{A}_{31} & & \widetilde{A}_{33} \end{array}\right] \left[\begin{array}{c|c} z_{11} & z_{12} \\ \hline z_{21} & z_{22} \\ \hline z_{31} & z_{32} \end{array}\right] \\ &= \left[\begin{array}{c|c} \widetilde{A}_{11}z_{11} + \widetilde{A}_{13}z_{31} & \widetilde{A}_{11}z_{12} + \widetilde{A}_{13}z_{32} \\ \hline \widetilde{A}_{21}z_{11} + \widetilde{A}_{22}z_{21} & \widetilde{A}_{21}z_{12} + \widetilde{A}_{22}z_{22} \\ \hline \widetilde{A}_{31}z_{11} + \widetilde{A}_{33}z_{31} & \widetilde{A}_{31}z_{12} + \widetilde{A}_{33}z_{32} \end{array}\right]\end{aligned}$$

and

$$\begin{aligned}\widetilde{A}\,Z_{\perp}^{s} &= \left[\begin{array}{cc|c} \widetilde{A}_{11} & & \widetilde{A}_{13} \\ \hline \widetilde{A}_{21} & \widetilde{A}_{22} & \\ \hline \widetilde{A}_{31} & & \widetilde{A}_{33} \end{array}\right] \left[\begin{array}{cc|c} z_{11_\perp} & & z_{12_\perp} \\ \hline & z_{21_\perp} & z_{22_\perp} \\ \hline & z_{31_\perp} & z_{32_\perp} \end{array}\right] \\ &= \left[\begin{array}{cc|cc} \widetilde{A}_{11}z_{11_\perp} & & \widetilde{A}_{13}z_{31_\perp} & \widetilde{A}_{11}z_{12_\perp} & & \widetilde{A}_{13}z_{32_\perp} \\ \hline \widetilde{A}_{21}z_{11_\perp} & \widetilde{A}_{22}z_{21_\perp} & & \widetilde{A}_{21}z_{12_\perp} & \widetilde{A}_{22}z_{22_\perp} & \\ \hline \widetilde{A}_{31}z_{11_\perp} & & \widetilde{A}_{33}z_{31_\perp} & \widetilde{A}_{31}z_{12_\perp} & & \widetilde{A}_{33}z_{32_\perp} \end{array}\right].\end{aligned}$$

We see that the local matrix vector products computations are the same. The difference is that for $\widetilde{A}Z$ we sum local contributions whereas for $\widetilde{A}Z_{\perp}^{S}$ we store them. The extra memory requirement comes from the contribution of neighbor subdomains through the non zero off-diagonal blocks $\widetilde{A}_{ij}$, $i \neq j$. This extra storage cost is proportional to the number of neighbors of a subdomain. It is bounded as the number of subdomains increases. Similar considerations hold for the other operations like for instance computing S_k^T, for $k \leq 2$ from formula (24). Indeed, M_k is a block diagonal matrix per subdomain so that computing S_k^T is an embarrassingly parallel task that can be performed in $nsmeig$ steps. Thus, the cost is smaller than $nsmeig$ iterations of the Schwarz preconditioner.
The only major difference between the classical and split coarse spaces lies in the size of matrices E_2 and F_2. Due to the sparsity of $Z_{\perp}^{S}$, matrix E_2 is a square block sparse matrix of size $(N \times n_{nsmeig})^2$ with typically $D \times N \times n_{nsmeig}^2$ non zero elements where D is the number of neighbours of a subdomain. Whereas matrix F_2 is full and of size n_{nsmeig}^2. The application of a direct solver to E_2 will be more costly than for matrix F_2, see § 4.2.3 for quantitative data. In practice, each process stores one copy of matrix E_2 or F_2 even when we have for more than one subdomain per process. Usually, each process consists of several cores (four in our experiments) so that we can use multithreaded direct solvers.

4.2.3 Numerical performance

All runs were made on a cluster of nodes interconnected by an Infiniband network. Each node is composed of two AMD "MagnyCours" twelve-core (2.1GHz) processors. We use the optimal mapping (so called 'sweat spot') for hybrid (MPI/OpenMP) codes i.e., 4 MPI processes per node. Each MPI process has access to 6 cores, therefore we have used 6 OpenMP threads per MPI process. In practice we map either one or two subdomains per MPI process.

We consider matrices coming from oil reservoir simulations. We compare the one level JSM preconditioner (6) with two two-level methods: the classical one and the new one where the coarse space is split subdomain wise. For all experiments, the overlapping regions are made of two layers of nodes. The first matrix is the pressure block extracted from the well known SPE10 benchmark [CB01], see the convergence curves in Figure 4. This test case is characterised by large permeability variations, 8–12 orders of magnitude. The discretization is made on a regular Cartesian grid with $60 \times 220 \times 85$ cells. The two other series of matrices come from a nonlinear black oil (BO) simulation, see the convergence curves in Figures 5 and 6. In all convergence curves, *No Coarse Space* refers to the one-level JSM preconditioner, (Z) to the classical coarse space and (ZS) to the subdomain wise split coarse space. Notice that in the black oil simulations, even with the two level preconditioners we have some plateaus at the first iterations. This is due to the fact that the two level preconditioners are built using spectral information coming from the first linear system which is different from the subsequent ones. This is not the case for the SPE10 experiment where only the right hand side is changed from one solve to the next ones. As for numerical efficiency they are detailed in table 1 that we comment now giving more information on the test cases.

The first experiment is on the pressure block of the SPE10 (Society of Petroleum Engineers) benchmark problem [CB01]. The matrix comes from three-dimensional reservoir simulation in a highly heterogeneous and anisotropic medium with one million unknowns. In contrast with the next experiments, we have only one matrix and two solves with different right hand sides. The first solve is performed with the one level Jacobi Schwarz preconditioner, see (6) with a domain decomposition into 256 subdomains. The first solve takes 29.6s using 128 MPI processes, two subdomains per MPI process. From this first solve, we build coarse spaces for the second solve. In Table 1, we compare the classical and split coarse spaces. In terms of iterations, the coarse spaces bring huge improvements, especially for the split coarse space. This can be seen as well on the convergence curves, see Figure 4. The matrix is very ill-conditioned so that the coarse space built according to the tolerance $\epsilon_{eig} = 0.1$ is quite large for the split coarse space: $52 \times 256 = 13,312$ degrees of freedom. The first row of Table 1 shows that it affects the cost of building the coarse space as well the overall speed-up which is close to one in this case. It motivates us to reduce the number of smallest Ritz eigenvalues ($nsmeig$) to ten. The results are reported in the second row of Table 1. Although the iteration counts are not as good as in the first row, we now have a substantial gain ($\times 5.5$) in terms of wall-clock time.

The two other series of matrices, denoted BO, come from a black oil simulation. This computation implies the solving of large-scale nonlinear problems arising from the finite-volume discretization in which the non-linearity is handled by a Newton-Raphson algorithm. Solving these problems leads to a succession of linear problems the solution to which converges towards the solution to the considered problem. In our numerical experiment we consider a series of linear systems extracted at some time step in the simulation. Following the strategy described in § 4.2, we solve the first linear system with a one-level Schwarz algorithm. Then, we reuse the coarse operator built from eigenvectors approximated

during the first resolution in the solutions of the linear systems for the remaining Newton-Raphson iterations. To be more precise, we consider two series of five linear systems. In the first serie we have $60 \times 60 \times 32$ grid points and in the second one $120 \times 120 \times 64$. At each linear solve (except for the first one), we use the solution of the previous linear system as an initial guess.

In Table 2, we report results for the black oil test case with $60 \times 60 \times 32 = 115,200$ grid points. Due to the nonlinearity of the problem, the number of non zero entries is not the same for the various matrices in the series. The average nnz entries is $791,550$. We compare the classical coarse space made of $nsmeig$ Ritz eigenvectors corresponding to small eigenvalues of the preconditioned system with the coarse space introduced in § 4.2 based on a domain wise splitting of the previous coarse space. We have 64 subdomains and we use either 32 or 64 MPI processes. The average iteration counts for the one-level Schwarz method is 54 iterations. The solving time of one system is 2.07s when using 32 MPI processes (two subdomains per MPI process) and 1.23s with 64 MPI processes (one subdomain per MPI process). When we solve the next four linear systems in the Newton-Raphson algorithm by the one-level Schwarz method, the iteration counts (see Figure 5) and solving time slightly decrease due to the use of the previous solution as an initial guess. Therefore we do not report them in the table. In the third and fourth columns, we give the average iteration counts for the next four solves using the classical (denoted by Z) or split (denoted by Z_S) coarse spaces. When using the split coarse space, we gain a factor 2.7 in iteration counts. We automatically selected $nsmeig = 7$ Ritz eigenvectors whose eigenvalues are smaller than a given threshold $\epsilon_{eig} = 0.1$. In column 6, we report the time needed to build and factorize the square matrix E_2 of size 448×448 for the split coarse space. Recall that this operation is performed only once for the series of matrices, see Algorithm 1. In the last two columns, we report the average speedup for the next subsequent four solves taking into account all extra costs due to the coarse space: the construction of the coarse space matrix E_2 and at each iteration the cost of precomputing matrices S_k^T, see (24). Since the classical coarse space is much smaller than the split coarse space if the same threshold ϵ_{eig} is used, we also tried to use larger $nsmeig$'s. This yields a classical coarse space which is slightly larger but richer. Results are reported in Table 3. We see that enlarging $nsmeig$ has a small impact on both iteration counts and speedup.

Table 4 is organized as Table 2 except that we consider the large black oil test case with $120 \times 120 \times 64$ grid points. The average nnz entries is $6,391,950$. We have 128 subdomains and we use either 64 or 128 MPI processes. The solve performed by the one-level Schwarz method require in average 102 iterations. The solving time is 21.05s when using 64 MPI processes and 11.97s with 128 MPI processes. When using the split coarse space, we gain a factor of almost four in iteration counts. We automatically selected $nsmeig = 14$ small Ritz eigenvectors. The overall speed-up is better than two for the split coarse space Z_S. This speed-up computed from wall-clock times is in agreement with the iteration counts. Indeed, the one-level solves for the five matrices take $5 \times 102 = 510$ iterations. Whereas, the split coarse space solves take 104 iterations for the first (one-level) solve plus 14 iterations for building the coarse operator plus 4×25 iterations for the four subsequent two-level solves which means 218 iterations for the whole process. This is about half the total number of iterations for the one-level method. As in the previous test case, we also tried to use larger

SRA		Iterations			C.S.	Speedup	
Matrix	nsmeig	1-lvl sol.	Z	Z_S	Z_S	Z	Z_S
SPE10	52_{TOL}	399	165	28	11.19s	2.03	1.37
	10_{FV}		192	81	0.97s	2.63	5.54

Table 1: Comparison of the coarse spaces for SPE10 benchmark with 256 subdomains

BO	Iterations			C.S.		Speedup	
# MPI	1-lvl sol.	Z	Z_S	nsmeig	Z_S	Z	Z_S
32	54	32	20	7	0.46s	1.09	1.40
64					0.25s	1.10	1.44

Table 2: Comparison of the coarse spaces, $60 \times 60 \times 32$ unknowns for a black oil simulation and 64 subdomains. Speedup refers to wall-clock timings of the overall process.

nsmeig's for the classical coarse space. Results are reported in Table 5 which is organized as Table 3. We see that enlarging *nsmeig* has a marginal impact on both the iteration counts and wall-clock times.

5 Conclusion

We studied the solving of linear systems arising from porous media flows simulations with high heterogeneities by Schwarz type methods. We first investigated the influence of the partition into subdomains. Using an AMG type partitioning in Metis or Scotch improves the convergence with almost no extra cost. Then, we introduced two two-level preconditioners based on coarse space corrections. They are algebraic in the sense that they only make use of information generated during the solving of the first linear system. Taking advantage of a parallel implementation, it is possible, at nearly no extra cost, to use larger but sparse coarse space (Z_S) obtained by splitting subdomain wise a classical coarse space. The iteration counts and convergence curves are then always substantially better than with the classical coarse space. As long as the coarse space is not too large, we also have a gain in wall-clock solving times. But when the coarse space is too large, the direct solver used to invert E_2 in (23) takes too much time, see Table 1. Our current solution is to reduce the size of the coarse space by limiting the number of Ritz eigenvectors that span the classical coarse space. Another way to bypass this limitation would be to design iterative methods for this specific type of problem. This requires further investigation.

BO	Iterations		Speedup	
# MPI	$Z(12)$	$Z(24)$	$Z(12)$	$Z(24)$
32	31	30	1.10	1.10
64			1.10	1.09

Table 3: Effect of the size of the classical coarse space, $60 \times 60 \times 32$ unknowns for a black oil simulation and 64 subdomains.

large BO	Iterations			C.S.		Speedup	
# MPI	1-lvl sol.	Z	Z_S	nsmeig	Z_S	Z	Z_S
64	103	80	25	14	2.52s	1.04	2.22
128					1.27s	1.05	2.25

Table 4: Comparison of the coarse spaces, $120 \times 120 \times 64$ unknowns for a black oil simulation and 128 subdomains. Speedup refers to wall-clock timings of the overall process.

BO	Iterations		Speedup	
# MPI	$Z(28)$	$Z(56)$	$Z(28)$	$Z(56)$
64	80	79	1.05	1.05
128			1.05	1.06

Table 5: Effect of the size of the classical coarse space and 128 subdomains, $120 \times 120 \times 64$ unknowns for a black oil simulation

References

[AC83] J. Appleyard and I. Cheshire. Nested factorization. *SPE Reservoir Simulation Symposium*, 12264, 1983.

[BFF+09] Aydin Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *SPAA '09: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 233–244, New York, NY, USA, 2009. ACM.

[BG08] A. Buluc and J. R. Gilbert. On the representation and multiplication of hypersparse matrices. In *IPDPS*, pages 1–11, 2008.

[CB01] M. A. Christie and M. J. Blunt. Tenth spe comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Engineering and Evaluation*, 4(4):308–317, 2001.

[CP08] C. Chevalier and F. Pellegrini. PT-SCOTCH: a tool for efficient parallel graph ordering. *Parallel Computing*, 6-8(34):318–331, 2008.

[CS99] Xiao-Chuan Cai and Marcus Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21:239–247, 1999.

[EG03] E. Efstathiou and M. J. Gander. Why Restricted Additive Schwarz converges faster than Additive Schwarz. *BIT Numerical Mathematics*, 43:945–959, 2003.

[EGW11] Y. Efendiev, J. Galvis, and X.-H. Wu. Multiscale finite element methods for high-contrast problems using local spectral basis functions. *Journal of Computational Physics*, 230:937–955, 2011.

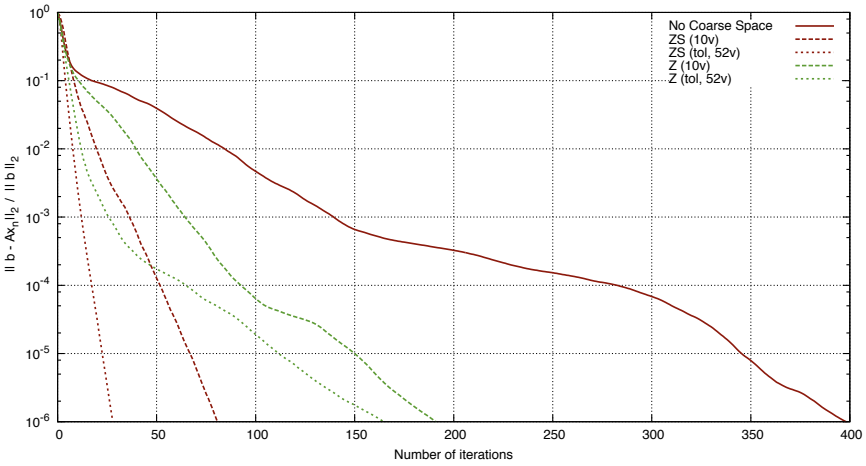

Figure 4: Convergence curves for SPE10 benchmark – domain decomposition into 256 subdomains

[GR03] P. Gosselet and C. Rey. On a selective reuse of Krylov subspaces in Newton-Krylov approaches for nonlinear elasticity. In *Domain decomposition methods in science and engineering*, pages 419–426 (electronic). Natl. Auton. Univ. Mex., México, 2003.

[Hec10] Frédéric Hecht. *FreeFem++*. Numerical Mathematics and Scientific Computation. Laboratoire J.L. Lions, Université Pierre et Marie Curie, http://www.freefem.org/ff++/, 3.7 edition, 2010.

[KK99] George Karypis and Vipin Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[LVW01] S. Lacroix, Y. Vassilevski, and M.F. Wheeler. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (ipars). *Numer. Linear Algebra with Applications*, 8:537–549, 2001.

[Man93] Jan Mandel. Balancing domain decomposition. *Communications in Applied and Numerical Methods*, 9:233–241, 1993.

[NXD10] F. Nataf, H. Xiang, and V. Dolean. A two level domain decomposition preconditioner based on local Dirichlet-to-Neumann maps. *C. R. Mathématique*, 348(21-22):1163–1167, 2010.

[NXDS11] F. Nataf, H. Xiang, V. Dolean, and N. Spillane. A coarse space construction based on local Dirichlet to Neumann maps. *SISC*, to appear, http://hal.archives-ouvertes.fr/hal-00491919/fr/:308–317, 2011.

[PdSM+06] Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674 (electronic), 2006.

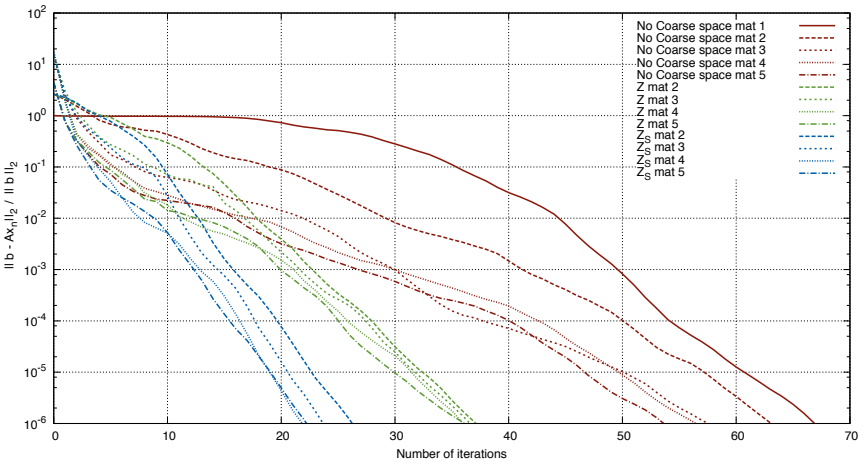

Figure 5: Convergence curves for the small BO matrices. $60 \times 60 \times 32$ unknowns decomposed into 64 subdomains.

[RR98] Christian Rey and Franck Risler. A Rayleigh-Ritz preconditioner for the iterative solution to large scale nonlinear problems. *Numer. Algorithms*, 17(3-4):279–311, 1998.

[RR00] Franck Risler and Christian Rey. Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems. *Numer. Algorithms*, 23(1):1–30, 2000.

[Sch70] H. A. Schwarz. Über einen Grenzübergang durch alternierendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, 15:272–286, May 1870.

[SMW03] R. Scheichl, R. Masson, and J. Wendebourg. Decoupling and block preconditioning for sedimentary basin simulations. *Computational Geosciences*, 7:295–318, 2003.

[SS86] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3), 1986.

[SYEG00] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.*, 21(5):1909–1926 (electronic), 2000. Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).

[TNVE09] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *J. Sci. Comput.*, 39:340–370, 2009.

[TW05] A. Toselli and Olof Widlund. *Domain Decomposition Methods: algorithms and theory*. Springer, 2005.

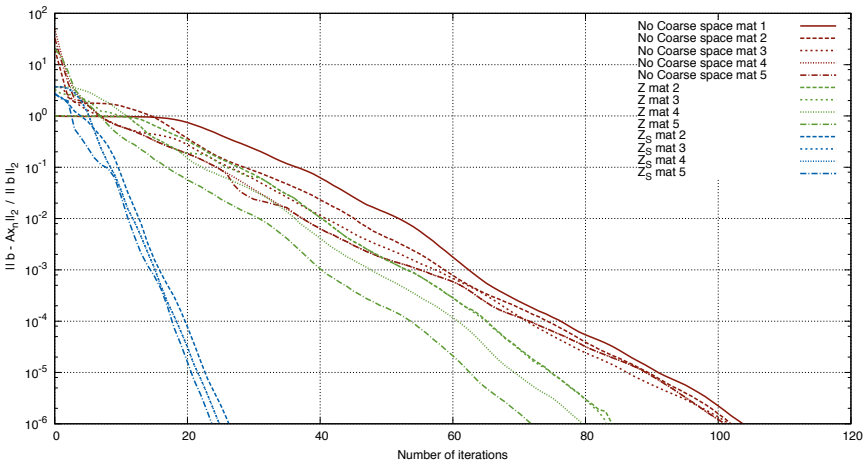

Figure 6: Convergence curves for the large BO matrices. $120\times120\times64$ unknowns decomposed into 128 subdomains.