Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

An Algebraic ILU(k) Based Two-Level Domain Decomposition Preconditioner

Douglas A. Augusto
DMA - IM - UFRJ
daa@labma.ufrj.br

Luiz M. Carvalho
DMA - IME - UERJ
luizmc@ime.uerj.br

Paulo Goldfeld
DMA - IM - UFRJ
goldfeld@ufrj.br

Ítalo C.L. Nievinski
PPGEM - FEN - UERJ
italonievinski@gmail.com

José R.P. Rodgrigues
CENPES - Petrobras
jrprodrigues@petrobras.com.br

Michael Souza
DEMA - UFC
michael@ufc.br

Abstract: *With the ultimate goal of designing a scalable parallel preconditioner for reservoir simulation problems, we combine domain decomposition ideas (prove suitable for parallelization) with incomplete factorizations (which are standard in reservoir simulation) at subdomain level. We introduce an ILU(k)-based two-level domain decomposition preconditioner and compare its performance with a two-level ILU(k)-Block-Jacobi preconditioner.*

Keywords: *two-level preconditioners, domain decomposition, Krylov methods, linear systems.*

1 Definition of the Preconditioner

1.1 Preliminaries

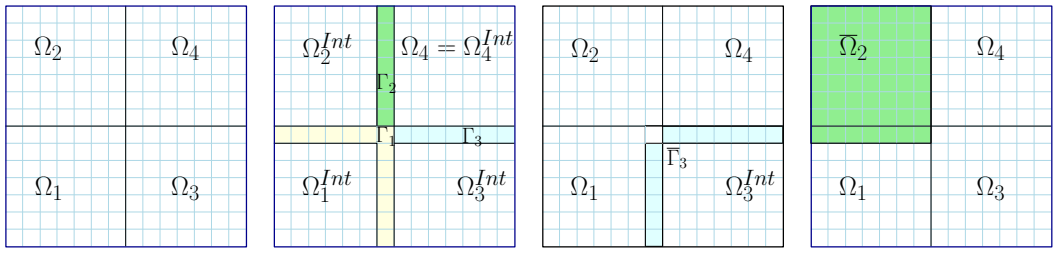
We consider the $(n \times n)$ linear system of algebraic equations

$$Ax = b \tag{1}$$

arising from the discretization of a PDE by a block-centered finite difference scheme with n blocks. We identify blocks and their respective indexes, in a way that Ω will denote either the domain of the PDE or the set of indexes $\{1, 2, \ldots, n\}$. We introduce a disjoint partition of Ω, i.e.,

$$\{\Omega_J\}_{1 \le J \le P} \text{ such that } \bigcup_{J=1}^{P} \Omega_J = \Omega \text{ and } \Omega_I \cap \Omega_J = \emptyset \ \ \forall I \neq J.$$

Figure 1a displays an example of a domain Ω decomposed into subdomains Ω_J.



(a) Partition into disjoint subdomains.

(b) Local interfaces and interiors.

(c) Extended interface $\overline{\Gamma}_3$.

(d) Extended subdomain $\overline{\Omega}_2$.

Figure 1: 2D domain partitioned into four subdomains; discretization based on a 5-point stencil.

DOI: 10.5540/03.2015.003.01.0093

Associated to each subdomain Ω_J, we define a local interface Γ_J,

$$\Gamma_J = \big\{ j \in \Omega_J \mid \big(\exists K > J, \text{ such that } \exists k \in \Omega_K \text{ with } (a_{jk} \neq 0 \text{ or } a_{kj} \neq 0) \big) \big\}, \tag{2}$$

where a_{ij} is the entry at the i-th row and j-th column of A. We also define the subdomain interior,

$$\Omega_J^{\text{Int}} = \Omega_J \setminus \Gamma_J.$$

The (global) interface is then defined as

$$\Gamma = \bigcup_{J=1}^{p} \Gamma_J.$$

We note that $\{\Gamma_J\}_{1 \leq J \leq P}$ form a disjoint partition of Γ. See Figure 1b.

We define the set $\overline{\Gamma}_J$ as

$$\overline{\Gamma}_J = \left\{ j \in \Gamma \;\middle|\; \big(j \in \Gamma_J\big) \text{ or } \big(\exists k \in \Omega_J \text{ such that } (a_{jk} \neq 0 \text{ or } a_{kj} \neq 0)\big) \right\}. \tag{3}$$

Note that $\Gamma_J \subset \overline{\Gamma}_J \subset \Gamma$. We point out that $\overline{\Gamma}_J$ is the result of augmenting Γ_J with the blocks of the interface Γ whose corresponding equations/variables are connected to Ω_J in the graph of A. We refer to $\overline{\Gamma}_J$ as an extended interface, see Figure 1c. We also define the extended subdomains $\overline{\Omega}_J = \Omega_J^{\text{Int}} \bigcup \overline{\Gamma}_J$, see Figure 1d.

Notice that $a_{jk} = 0$ for any $j \in \Omega_J^{\text{Int}}$ and $k \in \Omega_K^{\text{Int}}$, with $J \neq K$, so that, if the equations/variables corresponding to $\Omega_1^{\text{Int}}, \ldots, \Omega_P^{\text{Int}}$ are numbered consecutively followed by the ones corresponding to Γ, A has the following structure:

$$A = \left[\begin{array}{ccc|c} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{PP} & A_{P\Gamma} \\ \hline A_{\Gamma 1} & \cdots & A_{\Gamma P} & A_{\Gamma\Gamma} \end{array}\right]. \tag{4}$$

1.2 ILU(k) based Domain Decomposition

We now describe a two-level preconditioner M^{-1}, featuring a fine component M_F^{-1} described in Subsection 1.2.1 and a coarse component M_C^{-1} discussed in Subsection 1.2.2.

1.2.1 Preconditioner Construction

The fine part of the domain decomposition preconditioner we describe is based on the following block LU factorization of A:

$$A = LU = \left[\begin{array}{ccc|c} L_1 & & & \\ & \ddots & & \\ & & L_P & \\ \hline B_1 & \cdots & B_P & I \end{array}\right] \left[\begin{array}{ccc|c} U_1 & & & C_1 \\ & \ddots & & \vdots \\ & & U_P & C_P \\ \hline & & & S \end{array}\right], \tag{5}$$

where $A_{JJ} = L_J U_J$ is the LU factorization of A_{JJ}, $B_J = A_{\Gamma J} U_J^{-1}$, $C_J = L_J^{-1} A_{J\Gamma}$ and

$$S = A_{\Gamma\Gamma} - \sum_{J=1}^{P} A_{\Gamma J} A_{JJ}^{-1} A_{J\Gamma} = A_{\Gamma\Gamma} - \sum_{J=1}^{P} B_J C_J, \tag{6}$$

is the Schur complement of A with respect to the interior points. The inverse of A is

$$A^{-1} = \left[\begin{array}{ccc|c} U_1^{-1} & & & -U_1^{-1} C_1 S^{-1} \\ & \ddots & & \vdots \\ & & U_P^{-1} & -U_P^{-1} C_P S^{-1} \\ \hline & & & S^{-1} \end{array}\right] \left[\begin{array}{ccc|c} L_1^{-1} & & & \\ & \ddots & & \\ & & L_P^{-1} & \\ \hline -B_1 L_1^{-1} & \cdots & -B_P L_P^{-1} & I \end{array}\right]. \tag{7}$$

Our goal is to define a preconditioner M_F^{-1} that approximates the action of A^{-1} on a vector. For that sake, we need to define suitable approximations for the actions of L_J^{-1}, U_J^{-1}, B_J and C_J, $J = 1, \ldots, P$, and for that of S^{-1}. In the remaining of this subsection, we describe how these approximations are taken.

First we define $\widetilde{L}_J$ and $\widetilde{U}_J$ as the result of the incomplete LU factorization of A_{JJ} with level of fill k_{Int}, $[\widetilde{L}_J, \widetilde{U}_J] = \text{ILU}(\text{A}_{\text{JJ}}, \text{k}_{\text{Int}})$. Even though $\widetilde{L}_J$ and $\widetilde{U}_J$ are sparse, $\widetilde{L}_J^{-1} A_{J\Gamma}$ and $\widetilde{U}_J^{-T} A_{\Gamma J}^T$ (which would approximate C_J and B_J^T) are not. To ensure sparsity, we define $\widetilde{C}_J \approx \widetilde{L}_J^{-1} A_{J\Gamma}$ as the result of an incomplete triangular solve, by extending the definition of *level of fill* as follows. Let v_l and w_l be the sparse vectors corresponding to the l-th columns of $A_{J\Gamma}$ and $\widetilde{L}_J^{-1} A_{J\Gamma}$ respectively. Based on the solution of a triangular system by forward substitution, the components of v_l and w_l are related by

$$w_{l_k} = v_{l_k} - \sum_{i=1}^{k-1} \widetilde{L}_{J_{ki}} w_{l_i}. \tag{8}$$

We define the level of fill-in of component k of w_l recursively as

$$\text{Lev}(w_{l_k}) = \min \left\{ \text{Lev}(v_{l_k}), \min_{1 \le i \le (k-1)} \left\{ \text{Lev}(\widetilde{L}_{J_{ki}}) + \text{Lev}(w_{l_i}) + 1 \right\} \right\}, \tag{9}$$

where $\text{Lev}(v_{l_k}) = 0$ when $v_{l_k} \neq 0$ and $\text{Lev}(v_{l_k}) = \infty$ otherwise, and $\text{Lev}(\widetilde{L}_{J_{ki}})$ is the level of fill of entry ki in the $\text{ILU}(k_{\text{Int}})$ decomposition of A_{JJ} when $\widetilde{L}_{J_{ki}} \neq 0$ and $\text{Lev}(\widetilde{L}_{J_{ki}}) = \infty$ otherwise. The approximation $\widetilde{C}_J$ to $C_J = L_J^{-1} A_{J\Gamma}$ is then obtained by what we call *incomplete forward substitution with level of fill* k_{Bord}, in which we drop any terms with level of fill greater than k_{Bord} during the forward substitution process. We denote $\widetilde{C}_J = \text{IFS}(\widetilde{L}_J, A_{J_\Gamma}, k_{\text{Bord}})$. Notice that when $k_{\text{Bord}} = k_{\text{Int}}$, $\widetilde{C}_J$ is what would result from a standard partial incomplete factorization of A. The approximation $\widetilde{B}_J$ to $B_J = A_{\Gamma J} \widetilde{U}_J^{-1}$ is defined analogously.

Similarly, in order to define an approximation $\widetilde{S}$ to S, we start by defining $F_J = \widetilde{B}_J \widetilde{C}_J$ and defining a level of fill for the entries of F_J,

$$\text{Lev}(F_{J_{kl}}) = \min \left\{ \text{Lev}(A_{\Gamma\Gamma_{kl}}), \min_{1 \le i \le m} \left\{ \text{Lev}(\widetilde{B}_{J_{ki}}) + \text{Lev}(\widetilde{C}_{J_{il}}) + 1 \right\} \right\}, \tag{10}$$

where $m = \#\Omega_J^{\text{Int}}$ is the number of columns in $\widetilde{B}_J$ and rows in $\widetilde{C}_J$, $\text{Lev}(A_{\Gamma\Gamma_{kl}}) = 0$ when $A_{\Gamma\Gamma_{kl}} \neq 0$ and $\text{Lev}(A_{\Gamma\Gamma_{kl}}) = \infty$, otherwise, and $\text{Lev}(\widetilde{C}_{J_{ki}})$ is the level of fill according to definition (9) when $\widetilde{C}_{J_{ki}} \neq 0$ and $\text{Lev}(C_{J_{ki}}) = \infty$, otherwise $\big(\text{Lev}(\widetilde{B}_{J_{il}})$ is defined analogously$\big)$. Next, we define $\widetilde{F}_J$ as the matrix obtained retaining only the entries in F_J with level less than or equal to k_{Prod} according to (10). We refer to this *incomplete product* as $\widetilde{F}_J = \text{IP}(\widetilde{B}_J, \widetilde{C}_J, k_{\text{Prod}})$.

$\widetilde{S}$ is then defined as

$$\widetilde{S} = A_{\Gamma\Gamma} - \sum_{J=1}^{P} \widetilde{F}_J. \tag{11}$$

We remind the reader that while $\widetilde{S}$ approximates S, we need to define an approximation for S^{-1}. Since $\widetilde{S}$ is defined on the global interface Γ, it is not practical to perform ILU on it. Instead, we follow the approach employed in [3] and define for each subdomain a local version of $\widetilde{S}$,

$$\widetilde{S}_J = R_J \widetilde{S} R_J^T, \tag{12}$$

where $R_J : \Gamma \to \overline{\Gamma}_J$ is a restriction operator such that $\widetilde{S}_J$ is the result of pruning $\widetilde{S}$ so that only the rows and columns associated with $\overline{\Gamma}_J$ remain. More precisely, if $\{i_1, i_2, \ldots, i_{n_{\overline{\Gamma}_J}}\}$ is a list of the nodes in Γ that belong to $\overline{\Gamma}_J$, then the k-th row of R_J is $e_{i_k}^T$, the i_k-th row of the $n_\Gamma \times n_\Gamma$ identity matrix,

$$R_J = \begin{bmatrix} e_{i_1}^T \\ \vdots \\ e_{i_{n_\Gamma}}^T \end{bmatrix}.$$

Finally, our approximation S_F^{-1} to S^{-1} is defined as

$$S_F^{-1} = \sum_{J=1}^{P} T_J (L_{\widetilde{S}_J} U_{\widetilde{S}_J})^{-1} R_J \approx \sum_{J=1}^{P} T_J \widetilde{S}_J^{-1} R_J, \tag{13}$$

where $L_{\widetilde{S}_J}$ and $U_{\widetilde{S}_J}$ are by $ILU(k_\Gamma)$ of $\widetilde{S}_J$. Here $T_J : \overline{\Gamma}_J \to \Gamma$ is an extension operator that takes values from a vector that lies in $\overline{\Gamma}_J$, scales them by $w_1^J, \ldots, w_{n_{\overline{\Gamma}_J}}^J$ (which we call weights), and places them in the corresponding position of a vector that lies in Γ. Therefore, using the same notation as before, the k-th column of T_J is $w_k^J e_{i_k}$,

$$T_J = \begin{bmatrix} w_1^J e_{i_1} & \cdots & w_{n_\Gamma}^J e_{i_{n_\Gamma}} \end{bmatrix}.$$

We consider three different choices for the weights, giving rise to three options for T_J, which we denote T_J^{ones}, T_J^{was} and T_J^{ras}.

T_J^{ones} corresponds to the choice $w_i^J = 1$, $i = 1, \ldots, n_{\Gamma_J}$, so that $T_J^{\text{ones}} = R_J^T$. We define the so called *counting function* on the interface as

$$\mu = \sum_{J=1}^{P} T_J^{\text{ones}} 1_J,$$

where 1_J is a vector that lies in $\overline{\Gamma}_J$ with all entries equal to 1. The i-th entry of μ, denoted μ_i, counts how many extended interfaces the i-th interface node belongs to.

T_J^{was} corresponds to the choice $w_i^J = \mu_i^{-1}$. We point out that $T_J^{\text{was}}, J = 1, \ldots, P$, form a *partition of unity*, in the sense that $\sum_{J=1}^{P} T_J^{\text{was}} 1_J = 1_\Gamma$, where 1_Γ is an interface vector with all entries equal to 1.

At last, in order to define T_J^{ras}, we define

$$w_k^J = \begin{cases} 1, & \text{if the } i_k\text{-th node of } \Gamma \text{ belongs to } \Gamma_J \\ 0, & \text{if the } i_k\text{-th node of } \Gamma \text{ belongs to } \overline{\Gamma}_J \setminus \Gamma_J. \end{cases}$$

We note that $T_J^{\text{was}}, J = 1, \ldots, P$, also form a partition of unity. The notations "ras" and "was" are motivated by the Restricted and Weighted Additive Schwarz methods, see [2].

These three different choices for T_J yield three different versions of the preconditioner, which we refer to as ONES, RAS, and WAS.

1.2.2 Coarse Space Correction

We define a coarse space spanned by the columns of an $(n \times P)$ matrix that we call R_0^T. The J-th column of R_0^T is associated to the extended subdomain $\overline{\Omega}_J$ and its i-th entry is

$$(R_0^T)_{iJ} = \begin{cases} 0, \text{if node } i \text{ is not in } \overline{\Omega}_J \text{ and} \\ \mu^{-1}, \text{if node } i \text{ is in } \overline{\Gamma}_J, \end{cases}$$

where $\mu_i =$ number of $\overline{\Omega}_J$'s s.t. $i \in \overline{\Omega}_J$. Notice that $(R_0^T)_{iJ} = 1 \ \ \forall i \in \Omega_J^{\text{Int}}$ and that the columns of R_0^T form a partition of unity, in the sense that their sum is a vector with all entries equal to 1.

We define M_C by the formula

$$M_C^{-1} = R_0^T (R_0 A R_0^T)^{-1} R_0 \tag{14}$$

Notice that this definition ensures that $M_C^{-1} A$ is a projection onto range(R_0^T) and that for A symmetric positive definite this projection is A-orthogonal. Since $R_0 A R_0^T$ is small $(P \times P)$, we use exact LU (rather than ILU) when applying its inverse.

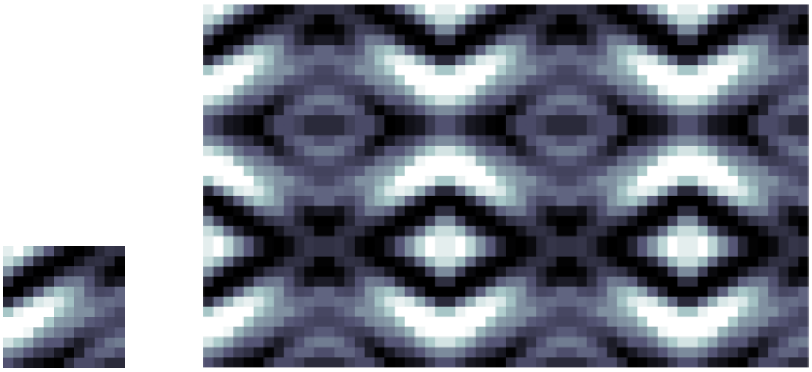


Figure 2: On the left, the basic 12×12 tile used in the experiments with heterogeneous field. On the right, a composition of 3×5 such tiles, suitably reflected.

Finally, the complete preconditioner has two components: one related to the complete grid, called M_F^{-1} in equation (7), and another related to the coarse space (14). The combined preconditioner is written down as

$$M^{-1} = M_F^{-1} + M_C^{-1}. \tag{15}$$

This formulation implies that the preconditioner will be applied additively and can be interpreted as having two levels, see [3]. This preconditioner is called algebraic ILU(k) based two-level domain decomposition, or simply Schur, in the following sections.

2 Numerical Experiments

2.1 Description of the Models

In order to evaluate the performance of the proposed preconditioners, we designed two sets of numerical experiments.

All the experiments that we present regard the solution of a linear system $Ax = b$. We use a matrix associated with the modeling of three-dimensional single-phase incompressible flow in porous medium with a seven-point block-centered finite-difference discretization of the differential operator $\nabla \cdot K\nabla$, imposing Dirichlet boundary conditions.

We considered two models, which we refer to as homogeneous, and heterogeneous. The first model assumes a homogeneous and isotropic field K. For our heterogeneous model, we generated a 12×12 "tile" of values for K, as shown in the left side of Figure 2. This tile is a realization of a stationary log-normal distribution with an anisotropic Gaussian covariance function (see [5], Section 5.4). Each layer of the field K was then created by the tiling of this 12×12 module (upon suitable reflections, in order to maintain smoothness). All layers were made identical (i.e., $K = K(x, y)$ and not $K = K(x, y, z)$.) The resulting fields were heterogeneous (with values varying two orders of magnitude), yet isotropic. The tile and the resulting 5×3 tiling are given in Figure 2.

2.2 Description of the Tests and Results

We use right-preconditioned GMRES with a restart of 30. The initial guess is zero, the stopping criterion is the reduction of the relative residual to less than 10^{-4}, and the maximum number of internal iterations is 500. The code is written in Matlab and C, using Matlab's mex feature, in order to improve the performance. The current version of the code is sequential, therefore we don't report CPU times, but only iteration counts. Using gcc as the C compiler, the Tests were run in a system featuring an 8-core 2.2GHz Intel Core i7-3632QM CPU (hyperthreading enabled) and 8GiB of main memory.

DOI: 10.5540/03.2015.003.01.0093

Table 1: Comparing three solvers for a homogeneous problem.

	Preconditioners		
Subdomains	None	Jacobi	Schur
$2 \times 2 \times 2$	56	16	**12**
$5 \times 5 \times 5$	191	25	**18**
$8 \times 8 \times 8$	353	25	**18**

Table 2: Comparing three solvers for a heterogeneous problem.

	Preconditioners		
Subdomains	None	Jacobi	Schur
$2 \times 2 \times 2$	329	16	**12**
$5 \times 5 \times 5$	NC	36	**23**
$8 \times 8 \times 8$	NC	33	**21**

Table 3: Schur alternatives without coarse for a heterogeneous problem.

	Without Coarse		
Subdomains	ONES	RAS	WAS
$2 \times 2 \times 2$	17	11	**11**
$4 \times 4 \times 4$	57	33	**32**
$6 \times 6 \times 6$	31	50	**50**
$8 \times 8 \times 8$	103	57	**56**

Table 4: Schur alternatives with coarse for a heterogeneous problem.

	Coarse		
Subdomains	ONES	RAS	WAS
$2 \times 2 \times 2$	16	12	**12**
$4 \times 4 \times 4$	29	21	**20**
$6 \times 6 \times 6$	30	21	**21**
$8 \times 8 \times 8$	33	22	**21**

In the first set of experiments, we compare the Schur preconditioner with inexact block Jacobi preconditioner [4], both using a coarse space correction operator as described in Section 1.2.2. In this batch of tests, we used version WAS of the Schur preconditioner, with all fill-in levels set to one. Table 1 and Table 2 compare the iterations of the homogeneous and the heterogeneous models, with 8, 125, and 512 subdomains, arranged in a 3D cube. Each subdomain has a fixed size of $12 \times 12 \times 12 = 1728$ blocks. These tables show the number of GMRES internal iterations, and for each problem (each line of the table), the smaller one is highlighted. NC indicates no convergence (more than 500 iterations). There is a column, None, which informs the number of GMRES iterations with no preconditioner.

Tables 3 and 4 show a comparison for the six versions of the Schur preconditioner described in Subsection 1.2 when solving the heterogenous problem. Table 3 presents the number of iterations without the coarse component, whereas Table 4 reports those with the coarse component. Here there are 8, 64, 216, and 512 subdomains, arranged in a 3D cube. All the next tables use this same set of values for the subdomains.

Tables 5 and 6 present the number of iterations for the Schur preconditioner when solving the heterogeneous problem, using the WAS version with a coarse space correction. In this case, we are interested in the variation of the different fills, k_{Int}, k_{Γ}, k_{Bord}, and k_{Prod} as described in Section 1.2.1. We show results when combining fills 0 and 1. In Table 5 we present the number of GMRES iterations for $k_{\text{Int}} = 0$ and in 6 for $k_{\text{Int}} = 1$. In those tables, the header 011, for instance, stands for $k_{\Gamma} = 0$, $k_{\text{Bord}} = 1$, and $k_{\text{Prod}} = 1$, and so forth.

Table 5: Fill variation for $k_{\text{Int}} = 0$

	XYZ where X=k_{Γ} Y=k_{Bord} Z=k_{Prod}							
Subdomains	000	001	010	100	011	101	110	111
$2 \times 2 \times 2$	15	15	14	15	14	15	14	14
$4 \times 4 \times 4$	23	22	22	22	21	22	21	21
$6 \times 6 \times 6$	23	23	23	23	22	23	22	22
$8 \times 8 \times 8$	23	23	23	23	23	23	23	22

DOI: 10.5540/03.2015.003.01.0093

Table 6: Fill variation for $k_{\text{Int}} = 1$

	XYZ where X=k_Γ Y=k_{Bord} Z=k_{Prod}							
Subdomains	000	001	010	100	011	101	110	111
$2 \times 2 \times 2$	14	14	13	13	13	13	12	12
$4 \times 4 \times 4$	23	23	23	22	22	22	21	20
$6 \times 6 \times 6$	24	23	23	23	22	22	22	21
$8 \times 8 \times 8$	24	24	23	23	23	23	22	21

2.3 Discussion

Although we address here only a few tests, in all the thousands of experiments we have performed, the Schur preconditioner took less iterations than Jacobi. In favor of the Jacobi alternative one can argue that there is less communication among the subdomains during its application, but we are able to cope with this advantage by overlapping communication and computation, as reported in the literature of the area, see [3]. The coarse space correction plays an essential role and we expect to see a similar behavior when dealing with systems arising from reservoir simulation problems. Even though the various levels of fill in the different parts of the preconditioner did not seem to play a role in the reported experiments, we believe that this flexibility of the method might be advantageous when tackling actual reservoir simulation matrices.

3 Future work

Our next step is extending our experiments to actual reservoir simulation matrices, provided by Petrobras. For this sake, we are currently implementing a parallel version of the preconditioner in PETSc [1].

Another challenge is to adapt the coarse space, tailoring it for this new kind of systems, which couples variables such as pressures and saturations.

References

[1] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2013. http://www.mcs.anl.gov/petsc.

[2] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2):792–797, 1999.

[3] L. M. Carvalho, L. Giraud, and P. Le Tallec. Algebraic two-level preconditioners for the Schur complement method. *SIAM J. Scientific Computing*, 22(6):1987–2005, 2001.

[4] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 4rd edition, 2013.

[5] D. Oliver, A. Reynolds, and N. Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, 2008.

