NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2001; **8**:207–227

Local preconditioners for two-level non-overlapping domain decomposition methods

L. M. Carvalho[1,†], L. Giraud[2,*,‡] and G. Meurant[3,§]

[1]*IME-UERJ, Rio de Janeiro, Brazil*
[2]*CERFACS, 42 Av. Gaspard Coriolis, 31057 Toulouse Cedex, France*
[3]*CEA/DIF, DCSA, BP12, 91680 Bruyeres le Chatel, France*

SUMMARY

We consider additive two-level preconditioners, with a local and a global component, for the Schur complement system arising in non-overlapping domain decomposition methods. We propose two new parallelizable local preconditioners. The first one is a computationally cheap but numerically relevant alternative to the classical block Jacobi preconditioner. The second one exploits all the information from the local Schur complement matrices and demonstrates an attractive numerical behaviour on heterogeneous and anisotropic problems. We also propose two implementations based on approximate Schur complement matrices that are cheaper alternatives to construct the given preconditioners but that preserve their good numerical behaviour. Through extensive computational experiments we study the numerical scalability and the robustness of the proposed preconditioners and compare their numerical performance with well-known robust preconditioners such as BPS and the balancing Neumann–Neumann method. Finally, we describe a parallel implementation on distributed memory computers of some of the proposed techniques and report parallel performances. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: domain decomposition; two-level preconditioning; Schur complement; parallel distributed computing; elliptic partial differential equations; parabolic partial differential equations

1. INTRODUCTION

In recent years, there has been an important development of domain decomposition algorithms for numerically solving partial differential equations. Nowadays some preconditioners for Krylov methods possess optimal convergence rates for given classes of elliptic problems. These optimality or quasi-optimality properties are often achieved thanks to the use of two-level preconditioners that are composed of local and global terms acting either in an additive or in a multiplicative way. In the framework of non-overlapping domain decomposition techniques, we refer for instance to BPS (Bramble, Pasciak and Schatz) [1] and Vertex Space

*Correspondence to: L. Giraud, CERFACS, 42 Av. G. Coriolis, 31057 Toulouse Cedex, France
†E-mail: luizmc@magnum.ime.uerj.br
‡E-mail: giraud@cerfacs.fr
§E-mail: meurant@bruyeres.cea.fr

Contract/grant sponsor: FAPERJ-Brazil; contract/grant number: 150.177/98

Received 25 November 1999
Accepted 25 September 2000

[2, 3] for additive two-level preconditioners, and to balancing Neumann–Neumann [4, 5], as well as FETI [6] for examples of multiplicative ones. We refer to References [7–9] for a more exhaustive overview of domain decomposition techniques.

In this paper we consider additive two-level preconditioners similar to BPS that can be written as the sum of a local and a global component. In Section 2, we describe a set of parallelizable local preconditioners that are the main focus of this paper and discuss the connections with well-known preconditioners like vertex space [3] and Neumann–Neumann [10]. We also briefly describe the global/coarse space component we have used for the numerical experiments reported in Section 3. These numerical experiments are conducted for two types of partial differential equations on two-dimensional domains. For elliptic equations, we show experiments for heterogeneous and/or anisotropic problems. We also solve systems arising from the time-implicit discretization of linear parabolic equations. To assess the relevance of the new preconditioners, we compare their numerical behaviours with well-known robust preconditioners such as the balancing Neumann–Neumann method [4]. In order to alleviate the computational cost for constructing these new local preconditioners, that require the explicit computation of the local Schur complement, we propose cheaper alternatives and show experimental results that demonstrate their efficiency. Finally, for the solution of heterogeneous anisotropic problems we report some parallel performance observed on a distributed memory platform for the most promising approaches.

2. PRECONDITIONER DESCRIPTION

We consider the following second-order self-adjoint elliptic problem on an open polygonal domain $\Omega \subset \mathbb{R}^2$:

$$\begin{cases} -\dfrac{\partial}{\partial x}\left(a(x,y)\dfrac{\partial v}{\partial x}\right) - \dfrac{\partial}{\partial y}\left(b(x,y)\dfrac{\partial v}{\partial y}\right) = F(x,y) & \text{in } \Omega, \\ v = 0 & \text{on } \partial\Omega_{\text{Dirichlet}} \neq \emptyset, \\ \dfrac{\partial v}{\partial n} = 0 & \text{on } \partial\Omega_{\text{Neumann}}, \end{cases} \tag{1}$$

where $\partial\Omega_{\text{Dirichlet}} \cap \partial\Omega_{\text{Neumann}} = \emptyset$ and $a(x,y),\ b(x,y) \in \mathbb{R}^2$ are strictly positive and bounded functions on Ω. We assume that the domain Ω is partitioned into N non-overlapping subdomains $\Omega_1, \ldots, \Omega_N$ with boundaries $\partial\Omega_1, \ldots, \partial\Omega_N$; this defines a coarse mesh, τ^H, with mesh size H being the largest diameter of the subdomains. We assume that a mesh is given which is a refinement of the subdomain partitioning. We discretize (1) by linear finite elements resulting in a symmetric and positive definite linear system

$$Au = f.$$

Let Γ be the set of all the indices of the mesh points which belong to the interfaces between the subdomains. Grouping the unknowns for the mesh points corresponding to Γ in the vector u_Γ and the ones corresponding to the unknowns in the interior I of the subdomains in u_I, we get the reordered problem

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^{\mathrm{T}} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I \\ f_\Gamma \end{pmatrix}. \tag{2}$$

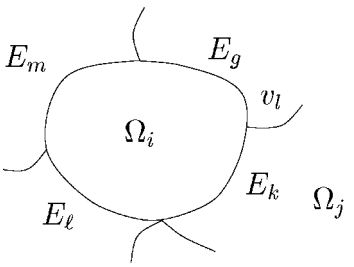

Figure 1. An internal subdomain.

Eliminating u_I from the second block row of (2) leads to the following reduced equation for u_Γ:

$$Su_\Gamma = f_\Gamma - A_{I\Gamma}^{\mathrm{T}} A_{II}^{-1} f_I, \tag{3}$$

where

$$S = A_{\Gamma\Gamma} - A_{I\Gamma}^{\mathrm{T}} A_{II}^{-1} A_{I\Gamma} \tag{4}$$

is the Schur complement of the matrix A_{II} in A. The matrix S inherits from A the symmetric positive definiteness property. Therefore we use preconditioned conjugate gradient iterations for solving (3).

In Figure 1, we depict an internal subdomain Ω_i with its edge interfaces E_m, E_g, E_k, E_ℓ and vertex points as v_l that define $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$. Let $R_{\Gamma_i} : \Gamma \to \Gamma_i$ be the canonical pointwise restriction which maps full vectors defined on Γ into vectors defined on Γ_i, and let $R_{\Gamma_i}^{\mathrm{T}} : \Gamma_i \to \Gamma$ be its transpose. For a stiffness matrix A arising from a finite element discretization, the Schur complement matrix (4) can also be written as

$$S = \sum_{i=1}^{N} R_{\Gamma_i}^{\mathrm{T}} S^{(i)} R_{\Gamma_i},$$

where

$$S^{(i)} = A_{\Gamma_i}^{(i)} - A_{i\Gamma_i}^{\mathrm{T}} A_{ii}^{-1} A_{i\Gamma_i} \tag{5}$$

is referred to as the local Schur complement associated with the subdomain Ω_i. $S^{(i)}$ involves submatrices from the local stiffness matrix $A^{(i)}$, defined by

$$A^{(i)} = \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i}^{\mathrm{T}} & A_{\Gamma_i}^{(i)} \end{pmatrix}. \tag{6}$$

The matrix $A^{(i)}$ corresponds to the discretization of Equation (1) on the subdomain Ω_i with Neumann boundary condition on Γ_i and A_{ii} corresponds to the discretization of Equation (1) on the subdomain Ω_i with homogeneous Dirichlet boundary conditions on Γ_i. In a parallel distributed memory environment, where each subdomain is assigned to one processor, all the local Schur complement matrices can be computed simultaneously by all the processors and the complete Schur matrix S defined by (4) is never fully assembled.

The local Schur complement matrix, associated with the subdomain Ω_i depicted in Figure 1, is dense and has the following block structure:

$$S^{(i)} = \begin{pmatrix} S_{mm}^{(i)} & S_{mg} & S_{mk} & S_{m\ell} & S_{mV}^{(i)} \\ S_{gm} & S_{gg}^{(i)} & S_{gk} & S_{g\ell} & S_{gV}^{(i)} \\ S_{km} & S_{kg} & S_{kk}^{(i)} & S_{k\ell} & S_{kV}^{(i)} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell}^{(i)} & S_{\ell V}^{(i)} \\ S_{Vm}^{(i)} & S_{Vg}^{(i)} & S_{Vk}^{(i)} & S_{V\ell}^{(i)} & S_{VV}^{(i)} \end{pmatrix},$$

where V is the set of vertices v_l of Ω_i. The first four diagonal blocks represent the local coupling between nodes on an edge interface introduced by the subdomain Ω_i and are only contributions to the diagonal blocks of the complete Schur complement matrix S. For instance, the diagonal block of the complete matrix S associated with the edge interface E_k, depicted in Figure 1, is $S_{kk} = S_{kk}^{(i)} + S_{kk}^{(j)}$. Assembling each diagonal block of the local Schur complement matrices and the blocks associated with the vertices, we obtain the local assembled Schur complement, that is

$$\bar{S}^{(i)} = \begin{pmatrix} S_{mm} & S_{mg} & S_{mk} & S_{m\ell} & S_{mV} \\ S_{gm} & S_{gg} & S_{gk} & S_{g\ell} & S_{gV} \\ S_{km} & S_{kg} & S_{kk} & S_{k\ell} & S_{kV} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell} & S_{\ell V} \\ S_{Vm} & S_{Vg} & S_{Vk} & S_{V\ell} & S_{VV} \end{pmatrix},$$

which corresponds to the restriction of S to the unknowns associated with the interface Γ_i of Ω_i.

In a parallel distributed memory framework, few neighbour-to-neighbour communications enable each processor to get its $\bar{S}^{(i)}$ once $S^{(i)}$ has been computed locally.

2.1. Local preconditioners

The new local preconditioners can be described using a set of canonical restriction operators. Let U be the space on which S operates and $(U_i)_{i=1,p}$ a set of subspaces of U such that:

$$U = U_1 + \cdots + U_p.$$

Let R_i be the canonical pointwise restriction of nodal values defined on U_i. Its transpose extends grid functions in U_i by zero to the rest of U. Using the above notation, we can define a wide class of block preconditioners by

$$M_{\text{loc}} = \sum_{i=1}^{p} R_i^{\mathrm{T}} M_i^{-1} R_i, \tag{7}$$

where

$$M_i = R_i S R_i^{\mathrm{T}}. \tag{8}$$

The properties of the operators (7) and (8) are given by the following lemma:

Lemma 1. *If the operator R_i^{T} is of full rank and if S is symmetric and positive definite, then the matrix M_i, defined in Equation (8), and the matrix M_{loc} defined in Equation (7) are symmetric and positive definite.*

Proof
The proof can be done in two steps. We first show that M_i^{-1} is symmetric positive definite (SPD) and then that M_{loc} is SPD.

Let $\langle .,. \rangle$ denote the scalar product associated with the 2-norm.

- If M_i^{-1} is SPD it is equivalent to showing that M_i is SPD.
- By definition, M_i is symmetric.

$$\begin{aligned} \forall x \neq 0 \quad \langle x, M_i x \rangle &= \langle x, R_i S R_i^{\mathrm{T}} x \rangle \\ &= \langle R_i^{\mathrm{T}} x, S R_i^{\mathrm{T}} x \rangle \end{aligned}$$

In addition

$$\left.\begin{array}{ll} R_i & \text{is full rank } \Rightarrow\ R_i^{\mathrm{T}} x \neq 0 \\ S & \text{is SPD} \end{array}\right\} \Rightarrow\ \langle R_i^{\mathrm{T}} x, S R_i^{\mathrm{T}} x \rangle \quad \text{is strictly positive.}$$

- M_{loc} is SPD.
  Let $x \in U$.

$$\langle x, M_{\text{loc}} x \rangle = \left\langle x, \sum_{i=1}^{p} R_i^{\mathrm{T}} M_i^{-1} R_i x \right\rangle = \sum_{i=1}^{p} \langle R_i x, M_i^{-1} R_i x \rangle, \tag{9}$$

  where $\forall i, \langle R_i x, M_i^{-1} R_i x \rangle \geqslant 0$ since M_i^{-1} is SPD. So the expression (9) can be zero, if and only if, $\forall i\ \langle R_i x, M_i^{-1} R_i x \rangle = 0$ which implies that $x = 0$ since R_i are canonical restrictions such that $U_i = \mathrm{Im}(R_i)$ and $U = U_1 + \cdots + U_p$. ■

Remark 1
If $U = U_1 \oplus \cdots \oplus U_n$, then M_{loc} is a block Jacobi preconditioner. Otherwise, M_{loc} is a block diagonal preconditioner with an overlap between the blocks as $U_i \cap U_j \neq \emptyset$. In this case, the preconditioner can be viewed as an algebraic additive Schwarz preconditioner for the Schur complement.

The preconditioners are required to be efficient on parallel distributed memory platforms. Therefore, we only consider subspaces U_i that involve information mainly stored in the local memory of the processors; that is information associated with only one subdomain and its closest neighbours. This approach introduces only cheap neighbour-to-neighbour communications between processors. In this respect, we present three different decompositions of U by associating each subspace, respectively, with:

1. each edge E_k and each vertex v_l of the decomposition giving rise to the edge preconditioner described in Section 2.1.1,
2. each edge E_k enlarged with neighbours of its end points v_l resulting in the vertex-edge preconditioner presented in Section 2.1.2,
3. each interface Γ_i of the subdomains giving the subdomain preconditioner presented in Section 2.1.3.

2.1.1. Edge preconditioners. For each edge E_i we define $R_i \equiv R_{E_i}$ as the standard pointwise restriction of nodal values on E_i. Its transpose extends grid functions in E_i by zero to the rest of the interface. Thus, $S_{ii} = R_{E_i} S R_{E_i}^{\mathrm{T}} = M_i$. Similarly, we consider R_{v_l} the restriction operator

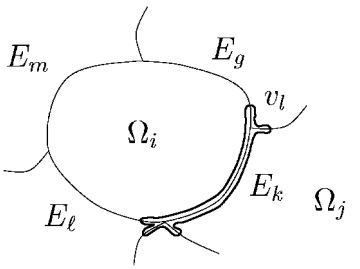

Figure 2. U_k associated to the vertex-edge preconditioner.

for each vertex of the coarse mesh τ^H defined by the decomposition. Using the above notation we define the edge-based local preconditioner by

$$M_{\text{loc}} = M_E = \sum_{E_i} R_{E_i}^{\text{T}} S_{ii}^{-1} R_{E_i} + \sum_{v_l} R_{v_l}^{\text{T}} S_{v_l v_l}^{-1} R_{v_l}. \tag{10}$$

This preconditioner aims at capturing the interaction between neighbouring nodes within the same edge interface. Notice that $S_{v_l v_l}$ in (10) is just a scalar which is the diagonal coefficient of the equation associated with the vertex v_l; this only corresponds to a diagonal scaling at the vertices of τ^H. This preconditioner is the straightforward block Jacobi that is well-known to be efficiently parallelizable. The main criticism against M_E is that it does not manage consistently neighbour nodes that are close to a vertex but belong to different edges, see Figure 1. We describe in the next section a preconditioner that intends to address this deficiency.

2.1.2. Vertex-edge preconditioners. The vertex-edge preconditioner is similar to the Vertex-Space preconditioner introduced in Reference [3], for which we merge into a single subspace the edge and vertex subspaces that appear in an additive way in References [2, 3].

In Figure 2, we depict U_k, the image of the restriction operator $R_k \equiv R_{\text{VE}_k}$ associated with the vertex-edge E_k enabling to define $S_{\text{VE}_i} = R_{\text{VE}_i} S R_{\text{VE}_i}^{\text{T}}$. With this notation the vertex-edge preconditioner is defined by

$$M_{\text{loc}} = M_{\text{VE}} = \sum_{E_i} R_{\text{VE}_i}^{\text{T}} S_{\text{VE}_i}^{-1} R_{\text{VE}_i}.$$

In that case, two neighbour vertex-edges (for instance, E_k and E_g in Figure 2) intercept each other, then the associated space splitting $(U_i)_i$ does not define a direct sum of the space U and the number of nodes in the neighbourhood of the vertex v_l defines the amount of overlap between the blocks M_i of the preconditioner.

2.1.3. Subdomain preconditioner. In this alternative, we try to exploit all the information available on each subdomain and we associate each subspace U_i with the entire boundary Γ_i of subdomain Ω_i. Here, we have $R_i \equiv R_{\Gamma_i}$. Consequently, $M_i = \bar{S}^{(i)}$ is the assembled local Schur complement. This splitting $(U_i)_i$ is not a direct sum of the space U and we have introduced some overlap between the blocks defining the subdomain preconditioner M_{S}.

We should notice the similitude between M_{S} and the Neumann–Neumann preconditioner, M_{NN}, originally proposed in References [10, 11].

M_S can be written as

$$M_S = \sum_{i=1}^{N} R_{\Gamma_i}^T (\bar{S}^{(i)})^{-1} R_{\Gamma_i},$$

while the Neumann–Neumann preconditioner is

$$M_{NN} = \sum_{i=1}^{N} R_{\Gamma_i}^T (D_i (S^{(i)})^+ D_i) R_{\Gamma_i}. \tag{11}$$

In equation (11) the matrices D_i are weighted matrices such that $\sum_{i=1}^{N} R_{\Gamma_i}^T D_i R_{\Gamma_i} = I$. I denotes the identity matrix and $(S^{(i)})^+$ is the Moore–Penrose pseudo-inverse since the local Schur complement matrices $S^{(i)}$ are singular for internal subdomains. Notice that assembling the local Schur complement $\bar{S}_i$ removes these singularities.

2.2. Computing alternatives

The construction of the proposed local preconditioners can be computationally expensive because the exact local Schur complement $S^{(i)}$ needs to be formed explicitly and then dense matrices M_i should be factorized. To alleviate these costs we propose two alternatives that can be combined. The first intends to reduce the construction cost of $S^{(i)}$ by using approximated solution of the local Dirichlet problems A_{ii}; the second intends to reduce the storage and the computational cost to apply the preconditioner by using sparse approximation of the M_i obtained by dropping the smallest entries.

2.2.1. Local Schur with inexact local solvers. Using the up-to-date sparse direct technology of efficient sparse direct solver, A_{ii} is factorized and $S^{(i)}$ can be computed via many forward/backward substitutions. Nonetheless, this procedure remains computationally expensive. To alleviate this cost, the exact solution of the local Dirichlet problems A_{ii}^{-1} (see equation (5)) can be replaced by some cheap approximations. For symmetric positive definite problems, approximations can be efficiently computed either by approximate inverses like AINV [12] or by an Incomplete Cholesky factorization, ILL^T resulting in an approximate Schur complement $\tilde{S}$.

Lemma 2. *If the matrix*

$$A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^T & A_{\Gamma\Gamma} \end{pmatrix}$$

is a Stieltjes matrix and (LL^T) *is an incomplete Cholesky factorization of* A_{II} *then* $\tilde{S} = A_{\Gamma\Gamma} - A_{I\Gamma}^T (LL^T)^{-1} A_{I\Gamma}$ *is also a Stieltjes matrix.*

Proof
It is enough to show that

$$0 \leqslant (LL^T)^{-1} \leqslant A_{II}^{-1},$$

since Theorem 7.1 in Reference [13] will then ensure that the resulting approximate Schur is an M-matrix. By construction $\tilde{S}$ is symmetric then a Stieltjes matrix is consequently SPD.

A_{II} is a symmetric M-Matrix, so by Theorem 2.4 in Reference [14], $A_{II} = (LL^T) - R$ is a regular splitting (i.e. $(LL^T)^{-1} \geqslant 0$ and $R \geqslant 0$).

$$A_{II} = (LL^T) - R \;\Rightarrow\; (LL^T)^{-1} A_{II} = I - (LL^T)^{-1} R \leqslant I.$$

Since A_{II} is an M-matrix, $A_{II}^{-1} \geqslant 0$ then

$$0 \leqslant (LL^{\mathrm{T}})^{-1} \leqslant A_{II}^{-1}. \quad \blacksquare$$

We note that the same property holds for the approximate Schur complement computed with AINV. In Reference [15] it is shown that the approximate inverse G of an M-matrix A computed by AINV also satisfies the inequality $0 \leqslant G \leqslant A^{-1}$.

Notice that Lemmas 1 and 2 ensure that for M-matrices the local preconditioners built using either ILL^{T} or AINV are SPD.

2.2.2. Sparse approximation of the local Schur complement. Another possible alternative to get a cheaper preconditioner is to consider a sparse approximation for S in (8) which may result in a saving of memory to store the preconditioner and saving of computation to factorize and apply the preconditioner. This approximation $\hat{S}$ can be constructed by dropping the elements of S that are smaller than a given threshold. More precisely, the following dropping strategy can be applied:

$$\hat{s}_{ij} = \begin{cases} 0 & \text{if } s_{ij} \leqslant \eta(|s_{ii}| + |s_{jj}|) \\ s_{ij} & \text{else} \end{cases} \tag{12}$$

Lemma 3. *If the matrix*

$$A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^{\mathrm{T}} & A_{\Gamma\Gamma} \end{pmatrix}$$

is a Stieltjes matrix then the sparse approximation $\hat{S}$ computed by (12) *applied to $S = A_{\Gamma\Gamma} - A_{I\Gamma}^{\mathrm{T}} A_{II}^{-1} A_{I\Gamma}$ is also a Stieltjes matrix.*

Proof
It is well known that S is a Stieltjes matrix (see Reference [13] for instance); then it is easy to see that removing off-diagonal entries while preserving symmetry preserves this property. $\blacksquare$

The two alternatives can be combined, that is dropping the smallest entries of approximate M_i, to produce a preconditioner that is cheap to compute and to store. We note that, for M-matrices, this combination gives rise to preconditioners that are still SPD.

In Section 3.2, we report some experiments using ILL^{T} as well as experiments with $\hat{S}$ and combining the two strategies.

2.3. Coarse space preconditioner

It can be shown (see for instance Reference [7]) that the local preconditioners alone are not numerically scalable for elliptic problems in the sense that

$$\kappa(M_{\mathrm{loc}} S) = \mathcal{O}(H^{-2}), \tag{13}$$

where H denotes the diameter of the subdomains and $\kappa(A)$ is the condition number of the matrix A. This means that when the number of subdomains increases the number of conjugate gradient iterations increases as well. To ensure a quasi-optimality property, that is, the condition number of the associated preconditioned systems is independent of the number of

subdomains and only logarithmically dependent on the size of the subdomains, a coarse problem defined on the whole physical domain should be incorporated into the preconditioner. This global coupling is critical for scalability. In particular, it has been shown in Reference [1] that, when applying the original BPS technique to a uniformly elliptic operator, the preconditioned system has a condition number

$$\kappa(M_{\text{BPS}}S) = \mathcal{O}(1 + \log^2(H/h)), \tag{14}$$

where h is the mesh size. This implies that the condition number depends only weakly on the number of points per subdomain but no longer depends on the number of subdomains. Therefore, such a preconditioner is numerically appropriate for large systems of equations on large processor systems.

Similar to BPS, we consider a class of additive two-level preconditioners that can be written in a generic way as

$$M_{\text{BPS-}*} = M_{\text{loc}} + M_{\text{glob}},$$

where M_{glob} is computed using a Galerkin formula involving S and not the original matrix A, as it is done in the regular BPS.

Let U_0 be a q-dimensional subspace of U. This subspace will be called the coarse space. Let $R_0 : U \to U_0$ be a restriction operator which maps full vectors of U into vectors in U_0, and let $R_0^T : U_0 \to U$ be the transpose of R_0, an interpolation operator which extends vectors from the coarse space U_0 to full vectors in the fine space U.

The Galerkin coarse space operator $A_0 = R_0 S R_0^{\text{T}}$, in some way, represents the Schur complement on the coarse space U_0. The global coupling mechanism is introduced by the coarse component of the preconditioner which can thus be defined as

$$M_{\text{glob}} = R_0^{\text{T}} A_0^{-1} R_0.$$

For the experiments reported in this paper, we consider the space U_0 obtained by associating one degree of freedom with each vertex v_l of τ^H (corner points) resulting from the partition $(\Omega_i)_{i=1,N}$ and a restriction operator R_0 specially designed to deal with the possible discontinuities in the PDE coefficients. We refer to Reference [16] and the references therein for a more detailed description of this coarse component and its numerical and parallel scalability.

Combining this coarse-space preconditioner with the three local preconditioners gives rise to variants of the BPS preconditioner that will be denoted:

— $M_{\text{BPS-}E}$ for $M_{\text{loc}} = M_E$. Notice that this local preconditioner is the one used in the genuine BPS; in this respect $M_{\text{BPS-}E}$ is the closest variant to regular BPS. It is a slight improvement of regular BPS as the coarse component does not rely on the spectral equivalence property between A and S for uniformly elliptic operators.
— $M_{\text{BPS-VE}}$ for $M_{\text{loc}} = M_{\text{VE}}$.
— $M_{\text{BPS-}S}$ for $M_{\text{loc}} = M_S$.

3. NUMERICAL EXPERIMENTS

In this section, we report through a set of model problems the numerical behaviour of the preconditioners introduced in Section 2. We consider not only the new BPS variants

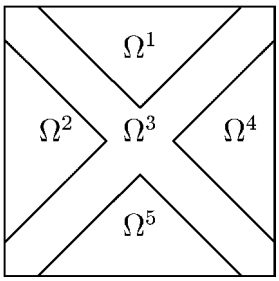

Figure 3. Example 1 — Flag.

but, additionally, the well-established balanced Neumann–Neumann preconditioner [4]. Before reporting the comparison results, we briefly recall the balanced Neumann–Neumann pre-conditioner.

The balanced Neumann–Neumann preconditioner has proven to be an efficient domain decomposition preconditioner for some fairly difficult problems [17], such as linear systems arising from structural analysis. At each iteration, two linear systems per subdomain must be solved, one corresponding to the PDE with Dirichlet boundary conditions (i.e. A_{ii} in (6)) and the other with Neumann boundary conditions (i.e. $A^{(i)}$ in (6)). It is through the solution of this latter Neumann problem that the action of $(S^{(i)})^+$ on a vector, defining the Neumann boundary conditions, is effectively computed. A global/coarse space problem is solved at each iteration to remove the possible singularity associated with the Neumann problem. We omit the details and refer to Reference [4] for a complete description. It is important to note that the balanced Neumann–Neumann preconditioner is scalable and in fact has the same condition number bound as BPS (see equation (14)). Henceforth, the balanced Neumann–Neumann preconditioner will be denoted by M_{BNN}.

3.1. Model problems

We consider the solution of two classes of elliptic problems. First, we compute the solution of equation (1) discretized by linear finite elements on a uniform mesh. A second set of experiments is related to a series of elliptic problems that arises in the solution of parabolic equations when using a time implicit scheme and a finite-element scheme in space.

3.1.1. Anisotropic and discontinuous elliptic model problems. For the solution of equation (1), the background of our study is the numerical solution of the 2D drift-diffusion equations for the simulation of semi-conductor devices [18, 19]. In this respect, we intend to evaluate the sensitivity of the preconditioners to anisotropy and to discontinuity. With this in mind, we consider the following 2D model problems.

In Figure 3, we represent the unit square divided into five regions where piecewise constant functions are used to define a first set of test problems. In addition, we have performed experiments with the problem defined by piecewise constant functions as depicted in Figure 4. Let $a()$ and $b()$ be the diffusion coefficients of the elliptic problem as described in equation (1).

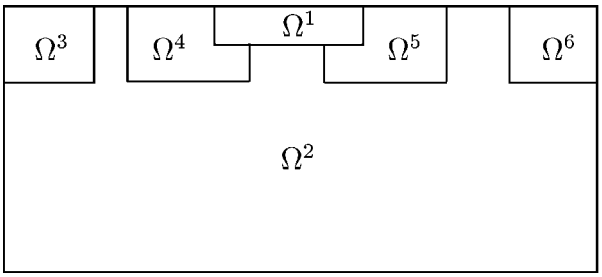

Figure 4. Example 2 — Region.

Using this notation and Figure 3, we define the first set of model problems with different degrees of difficulty:

- *Poisson problem*: $a() = 1$ and $b() = 1$,
- anisotropic and discontinuous problems with $a() = 1$ and $b()$ which depends on x and y.

*Problem AD-F*1:

$$b() = \begin{cases} 1 & \text{in } \Omega^3, \\ 10^2 & \text{in } \Omega^2 \cup \Omega^4, \\ 10^{-2} & \text{in } \Omega^1 \cup \Omega^5. \end{cases}$$

*Problem AD-F*2:

$$b() = \begin{cases} 1 & \text{in } \Omega^3, \\ 10^3 & \text{in } \Omega^2 \cup \Omega^4, \\ 10^{-3} & \text{in } \Omega^1 \cup \Omega^5. \end{cases}$$

- discontinuous problems with

*Problem D-F*1 :

$$a() = b() = \begin{cases} 1 & \text{in } \Omega^3, \\ 10^2 & \text{in } \Omega^2 \cup \Omega^4, \\ 10^{-2} & \text{in } \Omega^1 \cup \Omega^5. \end{cases}$$

*Problem D-F*2:

$$a() = b() = \begin{cases} 1 & \text{in } \Omega^3, \\ 10^3 & \text{in } \Omega^2 \cup \Omega^4, \\ 10^{-3} & \text{in } \Omega^1 \cup \Omega^5. \end{cases}$$

Using piecewise constant functions on the regions depicted in Figure 4, we define a second set of test problems:

- anisotropic and discontinuous problems: $a() = 1$ and
  Problem AD-R:

$$b() = \begin{cases} 10^{-1} & \text{in } \Omega^1, \\ 10^{-2} & \text{in } \Omega^2, \\ 10^1 & \text{in } \Omega^3 \cup \Omega^4 \cup \Omega^5 \cup \Omega^6. \end{cases}$$

- discontinuous problems:
  Problem D-R:

$$a() = b() = \begin{cases} 10^{-1} & \text{in } \Omega^1, \\ 10^{-2} & \text{in } \Omega^2, \\ 10^1 & \text{in } \Omega^3 \cup \Omega^4 \cup \Omega^5 \cup \Omega^6. \end{cases}$$

We have also considered a last set of problems associated with (1). We have introduced anisotropy not necessarily aligned with the axis but making an angle θ with the x-direction corresponding to the following PDE:

$$(\varepsilon c^2 + s^2)\frac{\partial^2 u}{\partial x^2} + 2cs(1-\varepsilon)\frac{\partial^2 u}{\partial x \partial y} + (c^2 + \varepsilon s^2)\frac{\partial^2 u}{\partial x^2} = f \tag{15}$$

where $\varepsilon \ll 1$, $c = \cos\theta$ and $s = \sin\theta$. For $\theta = 0$ equation (15) reduces to the classical model anisotropic equation:

$$\varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f. \tag{16}$$

3.1.2. Elliptic problems involved in the solution of parabolic linear equations. As in other model problems, let us consider the solution of the linear systems arising from the implicit discretization of parabolic linear partial differential equations like

$$\begin{cases} \dfrac{\partial v}{\partial t} - \dfrac{\partial}{\partial x}\left(a(x,y)\dfrac{\partial v}{\partial x}\right) - \dfrac{\partial}{\partial y}\left(b(x,y)\dfrac{\partial v}{\partial y}\right) = F(x,y) & \text{in } \Omega, \\ v = 0 & \text{on } \partial\Omega, \\ v(x,0) = v_0(x). \end{cases}$$

In an abstract form, this problem can be written as

$$\frac{\partial v}{\partial t} + Lu = f,$$

where L is a second-order self-adjoint linear elliptic problem. This problem is discretized in time using an implicit Crank–Nicholson centred scheme with time step Δt and in space by linear finite elements with mesh size h giving rise to the stiffness matrix $h^{-2}A$. The solution of the parabolic equation reduces to a sequence of elliptic problems. With $u^m = v(x, t^m)$ at each time step we have

$$\frac{u^{m+1} - u^m}{\Delta t} + \frac{1}{2h^2}(Au^{m+1} + Au^m) = \tfrac{1}{2}(f^{m+1} + f^m),$$

so we have to solve

$$\left(2\frac{h^2}{\Delta t}I + A\right) u^{m+1} = 2\frac{h^2}{\Delta t}u^m - Au^m + h^2(f^{m+1} + f^m).$$

Letting $\mu = 2h^2/\Delta t$, and $A_t = (\mu I + A)$, we have to solve the linear systems

$$A_t \delta = h^2(f^{m+1} + f^m) - 2Au^m, \tag{17}$$

then advance the solution in time

$$u^{m+1} = u^m + \delta.$$

The linear system (17) can be solved using a Schur complement approach, preconditioned with the techniques described in Section 2.

3.2. Experimental results

For the experimental results related to M_{VE}, we have considered two extra edge points in the neighbourhood of the vertices v_l in each direction. For a more detailed study about the influence of the size of the overlap in the neighbourhood of the vertices v_l on the convergence rate, we refer to Reference [18]. We just state here that a very small overlap is usually enough to improve the behaviour of M_{VE} with respect to M_E. Both preconditioners have comparable computational complexities and consequently, similar parallel performances [18].

Table I. Number of iterations on the Poisson problem.

No. of subdomains	4×4	8×8	16×16
M_{E}	13	28	51
M_{VE}	12	22	40
M_{S}	11	19	32
$M_{\mathrm{BPS}\text{-}E}$	9	11	11
$M_{\mathrm{BPS\text{-}VE}}$	10	12	12
$M_{\mathrm{BPS}\text{-}S}$	10	10	11
M_{BNN}	11	12	12

For all the experimental results reported in the next section, the convergence of the preconditioned conjugate gradient method is attained when the 2-norm of the residual of the current iteration normalized by the 2-norm of the right-hand side is less than 10^{-6}. For all the experiments reported in the following tables, the number of subdomains varies from 16 (4×4 decomposition) up to 256 (16×16 decomposition) keeping constant the number of grid points per subdomain (i.e. 16×16 mesh for each subdomain, that is $H/h = 16$); the initial guess x_0 for the conjugate gradient iterations was the null vector. Notice that for the first set of test examples the discontinuities in the coefficients $a(x, y)$ and $b(x, y)$ are not aligned with the interface of the square subdomains. The same observation is true for the pure anisotropic problem (15) with $\theta \neq 0$; the anisotropic behaviour is not aligned with the interface of the decomposition. All the numerical experiments, except the ones reported in Section 4 have been performed on a single-processor workstation using Matlab.

3.2.1. Anisotropic and discontinuous elliptic problems. In Table I, we report results observed on the Poisson equation using the preconditioners with and without the coarse-space component M_{glob}. When only local preconditioners are implemented, it can be seen that when the local information is more represented in the preconditioner, the convergence is better. These results also show that without a coarse space component the number of iterations required by the preconditioned conjugate gradient grows with the number of subdomains as predicted by the estimated condition number given by equation (13). Using the two-level preconditioners, these observations are no longer true. The coarse space component somehow smoothes the effect of the local component. According to the theoretical bound given by equation (14), the number of preconditioned conjugate gradient iterations becomes independent of the number of domains. Finally, we note that for the two-level preconditioners $M_{\mathrm{BPS}\text{-}E}$ and M_{BNN}, the results are similar to those of other authors [4, 20].

In Table II, we depict the numerical behaviour of the preconditioners on the model problem (15) that only exhibit anisotropy not aligned with the axes. When no coarse-space component is implemented M_{VE} still outperforms M_E, M_S is the most efficient and the number of iterations of all the preconditioners grows with the number of subdomains. For the two-level preconditioners, we first observe that the anisotropy prevents them from having an optimal convergence behaviour independent of the number of subdomains, even though the number of iterations is considerably decreased by the coarse-space component. Furthermore, for some problems $M_{\mathrm{BPS\text{-}VE}}$ becomes less efficient than the simpler $M_{\mathrm{BPS}\text{-}E}$ while $M_{\mathrm{BPS}\text{-}S}$ always

Table II. Number of iterations for solving (15) with several values of $\theta - \varepsilon = 10^{-3}$.

No. of subdomains	4×4			8×8			16×16		
θ	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$
M_{E}	21	34	30	47	67	77	88	132	164
M_{VE}	21	22	23	44	42	59	72	81	141
M_{S}	14	20	20	25	40	41	53	75	88
$M_{\mathrm{BPS}\text{-}E}$	27	24	20	58	34	28	81	43	35
$M_{\mathrm{BPS}\text{-VE}}$	25	21	21	48	33	35	85	43	49
$M_{\mathrm{BPS}\text{-}S}$	20	19	17	33	26	21	47	33	26

Table III. Number of iterations for problems with discontinuity.

No. of subdomains	4×4			8×8			16×16		
Model problem	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$M_{\mathrm{BPS}\text{-}E}$	12	11	10	11	11	11	14	15	11
$M_{\mathrm{BPS}\text{-VE}}$	13	12	11	13	12	12	16	16	12
$M_{\mathrm{BPS}\text{-}S}$	12	10	10	11	11	11	14	14	11
M_{BNN}	25	27	21	29	28	38	48	65	52

Table IV. Number of iterations for problems with discontinuity and anisotropy.

No. of subdomains	4×4			8×8			16×16		
Model problem	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$M_{\mathrm{BPS}\text{-}E}$	18	24	25	29	65	35	42	103	39
$M_{\mathrm{BPS}\text{-VE}}$	18	23	24	33	80	40	56	141	55
$M_{\mathrm{BPS}\text{-}S}$	16	20	18	22	43	22	33	79	26
M_{BNN}	37	52	147	60	158	644	97	311	*

*Means no convergence after 1000 iterations.

ensures the fastest convergence. So the conjecture, 'the richer the local preconditioner, the more efficient the preconditioner', is only true when the local preconditioners run alone.

In Tables III and IV, we study the numerical behaviour of the two-level preconditioners on model problems arising from the discretization of (1) that exhibit either discontinuity (Table III) or both discontinuity and anisotropy (Table IV). For the problems with only discontinuity, all the variants $M_{\mathrm{BPS}\text{-}*}$ have comparable convergence behaviours.

As it can be seen in Table IV, problems with anisotropy and discontinuity are more difficult to solve. Again, $M_{\mathrm{BPS}\text{-VE}}$ does not outperform the basic $M_{\mathrm{BPS}\text{-}E}$. For those examples, similar to the pure anisotropic situation reported in Table II, $M_{\mathrm{BPS}\text{-}S}$ exhibits once again the best convergence behaviour.

The relatively poor performance of M_{BNN}, reported in Tables III and IV, could be improved. An alternative way, as suggested in Reference [10], should be a better choice of the weight matrices D_i, involved in equation (11), when the diagonal entries of S are available.

Table V. Number of iterations varying the anisotropy with an 8×8 subdomain decomposition.

ε	1.0	10^{-1}	10^{-2}	10^{-3}
M_{BNN}	12	20	40	98
$M_{\mathrm{BPS}\text{-}S}$	12	15	22	33

Table VI. Number of iterations using inexact local solvers $ILL^{\mathrm{T}}(0)$ to build the preconditioners.

No. of subdomains	4×4			8×8			16×16		
Model problem	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{\mathrm{BPS}\text{-}E}$	14	13	14	13	13	14	17	17	14
$\tilde{M}_{\mathrm{BPS}\text{-}\mathrm{VE}}$	20	18	20	19	19	19	24	26	20
$\tilde{M}_{\mathrm{BPS}\text{-}S}$	14	13	15	13	13	12	17	18	13
Model problem	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{\mathrm{BPS}\text{-}E}$	24	30	28	36	67	37	50	112	45
$\tilde{M}_{\mathrm{BPS}\text{-}\mathrm{VE}}$	27	34	31	40	84	48	64	143	64
$\tilde{M}_{\mathrm{BPS}\text{-}S}$	24	26	23	30	53	31	47	80	41

With this appropriate choice of the weights, one can expect a reduction of the gap between M_{BNN} and $M_{\mathrm{BPS}\text{-}*}$ for discontinuous problems, as suggested by the results reported in Reference [5]. However, the key trick in the Neumann–Neumann preconditioner is to get the action of $(S^{(i)})^{-1}$ on a vector without explicitly forming $S^{(i)}$ and, thus, in the classical implementation of M_{BNN} those entries are usually not computed. Furthermore, in Table V we report the numerical behaviour of $M_{\mathrm{BPS}\text{-}S}$ and M_{BNN} for the anisotropic problems defined by equation (16) for different values of the anisotropic coefficient ε. For those problems, the choice of the weighted matrices used in Reference [5] for M_{BNN} would reduce to the simple ones we have considered; that is, $1/2$ for the nodes on the edges and $1/4$ for the vertex points v_l. For anisotropic problems, we cannot expect M_{BNN} to become competitive with $M_{\mathrm{BPS}\text{-}S}$ for ε lower than 10^{-1}.

Local Schur with inexact local solvers: To alleviate the cost of the preconditioners construction, the factorization of the local Dirichlet problem can be replaced by an incomplete Cholesky factorization without fill-in, i.e. $ILL^{\mathrm{T}}(0)$, or with some fill-in controlled through a threshold, i.e. $ILL^{\mathrm{T}}(t)$. In this later situation the amount of fill-in can be defined by the fill-in ratio that is the number of non-zeros in the incomplete factors divided by the number of non-zeros in the lower part of the original matrices; by definition this fill-in ratio is equal to one for $ILL^{\mathrm{T}}(0)$.

In Tables VI and VII, we denote by $\tilde{M}_{\mathrm{BPS}\text{-}E}$, $\tilde{M}_{\mathrm{BPS}\text{-}\mathrm{VE}}$ and $\tilde{M}_{\mathrm{BPS}\text{-}S}$ the preconditioners computed using those inexact local solves. More precisely, we report in Table VI the number of iterations when $ILL^{\mathrm{T}}(0)$ is used and in Table VII those observed when some fill-in is enabled with a fill-in ratio lower than 3.5.

Table VII. Number of iterations using inexact local solvers $ILL^{\mathrm{T}}(t)$ to build the preconditioners.

No. of subdomains	4×4			8×8			16×16		
Model problem	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{\text{BPS-}E}$	13	12	12	13	14	12	16	19	11
$\tilde{M}_{\text{BPS-VE}}$	15	17	12	17	19	13	22	27	12
$\tilde{M}_{\text{BPS-}S}$	12	12	10	11	12	11	16	18	11
Model problem	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{\text{BPS-}E}$	23	31	28	35	70	37	48	114	40
$\tilde{M}_{\text{BPS-VE}}$	21	33	26	36	85	44	59	147	56
$\tilde{M}_{\text{BPS-}S}$	19	27	20	27	54	24	39	81	29

Table VIII. Number of iterations using sparse Schur to build the preconditioners.

No. of subdomains	4×4			8×8			16×16		
Model problem	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\hat{M}_{\text{BPS-}E}$	13	12	12	11	11	13	14	15	12
$\hat{M}_{\text{BPS-VE}}$	16	16	18	16	16	18	20	20	18
$\hat{M}_{\text{BPS-}S}$	12	11	12	12	12	11	15	16	11
Model problem	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\hat{M}_{\text{BPS-}E}$	18	25	27	29	65	35	43	111	40
$\hat{M}_{\text{BPS-VE}}$	24	30	26	36	81	42	57	142	57
$\hat{M}_{\text{BPS-}S}$	18	21	18	23	44	23	36	79	27

The comparison of the results depicted in Tables VI and VII and those in Tables III and IV shows that the approximation of the local Schur complement used to build the preconditioners generally deteriorates the numerical behaviours of the preconditioner. This approximation does not affect significantly the numerical behaviour of $\tilde{M}_{\text{BPS-}E}$ and $\tilde{M}_{\text{BPS-}S}$ but deteriorates noticeably the one of $\tilde{M}_{\text{BPS-VE}}$. In addition, enabling some fill-in in the incomplete factorizations generally improves the convergence rate; the most significant improvements are observed on anisotropic and discontinuous problems with $\tilde{M}_{\text{BPS-VE}}$ and $\tilde{M}_{\text{BPS-}S}$.

Sparse approximation of the Schur complement: In Table VIII we report the number of iterations using an approximate Schur complement $\hat{S}$ with η in (12) such that we only retain around 5 per cent of the entries in S. The resulting preconditioners are denoted, respectively, by $\hat{M}_{\text{BPS-}E}$, $\hat{M}_{\text{BPS-VE}}$ and $\hat{M}_{\text{BPS-}S}$.

The comparison of these results with those displayed in Tables III and IV indicates that, except for $\hat{M}_{\text{BPS-VE}}$ on discontinuous problems, only retaining very few entries in the Schur complement is enough to ensure the numerical quality of these preconditioners since the number of iterations is roughly the same in both cases (except for $\hat{M}_{\text{BPS}-\text{VE}}$ on discontinuous problems).

In addition, as mentioned in Section 2.2.2, the inexact local solvers and the dropping strategy can be combined to build variants of the preconditioners. The resulting preconditioners

Table IX. Number of iterations using preconditioner based on sparse Schur built using inexact local solvers $ILL^{\mathrm{T}}(t)$.

No. of subdomains	4×4			8×8			16×16		
Model problem	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\underline{M}_{\text{BPS-}E}$	14	13	12	13	13	13	16	18	13
$\underline{M}_{\text{BPS-VE}}$	19	18	18	20	22	18	26	29	18
$\underline{M}_{\text{BPS-}S}$	12	12	12	12	13	11	17	19	11
Model problem	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\underline{M}_{\text{BPS-}E}$	22	32	29	35	70	36	47	113	41
$\underline{M}_{\text{BPS-VE}}$	26	38	27	39	86	45	61	150	58
$\underline{M}_{\text{BPS-}S}$	19	29	21	28	55	28	42	81	31

Table X. Number of iterations to solve the elliptic problems involved in the solution of a parabolic equation for one time step – $\varepsilon = 10^{-3}$.

No. of subdomains	4×4			8×8			16×16		
θ	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$	0	$\pi/8$	$\pi/4$
M_E	10	15	16	16	16	17	19	16	17
$\tilde{M}_E$	10	15	17	17	16	17	19	15	17
M_{VE}	11	9	12	13	10	13	14	10	14
$\tilde{M}_{\text{VE}}$	11	11	13	13	11	14	14	12	15
M_S	8	10	11	13	10	12	13	10	12
$\tilde{M}_S$	8	11	12	13	11	12	13	10	12

are, respectively, denoted by $\underline{M}_{\text{BPS-}E}$, $\underline{M}_{\text{BPS-VE}}$ and $\underline{M}_{\text{BPS-}S}$. Numerical experiments where we dropped the smallest elements of the local preconditioners built using $ILL^{\mathrm{T}}(t)$ are reported in Table IX. Comparing these results with those of Tables VIII and VII indicates that the numerical quality of the resulting preconditioners is mainly governed by the use of ILL^{T}.

3.2.2. Elliptic problems in the solution of parabolic equations. In Table X, we report experimental results for the solution of elliptic problems involved in the solution of a parabolic equation for one time step. Here the operator L corresponds to the anisotropic equation (15) where θ is varied. The time step and the mesh size are such that $\mu = 0.02$, which gives rise to a well-conditioned linear system (17) (independent of h for the classic heat equation) and consequently, a well-conditioned associated Schur complement. With this choice we note that the local preconditioners are numerically scalable with respect to the number of subdomains as it was already observed in an overlapping domain decomposition approach [21]. In Table X, M_* denotes the preconditioner built using the exact Schur complement and $\tilde{M}_*$ the ones computed using $\tilde{S}$. In those examples M_{VE} is generally from 20 up to 40 per cent faster than M_E, while both, as already noticed, have a similar computational complexity [18]. M_S is still the most efficient but for those problems, the gap between this preconditioner and the other two decreases. In that case, M_{VE} may be the most efficient alternative as the factorizations of its M_i require about 16 times less floating point operations than the factorization for M_S.

Table XI. Number of iterations for AD-F2.

No. of subdomains	4×4	8×8
$M_{\text{BPS-}E}$	35	82
$M_{\text{BPS-}S}$	25	51
$\tilde{M}_{\text{BPS-}E}$	46	85
$\tilde{M}_{\text{BPS-}S}$	39	61

Indeed the complexity of the Cholesky factorization is $\mathcal{O}(n^3)$ for a matrix of size n, the contributions computed on internal subdomains for $M_{\text{BPS-}E}$ requires four factorizations of matrices of size m while a factorization of one matrix of size $4m$ is required for $M_{\text{BPS-}S}$. As before, those results show that approximate local Schur complements based on $ILL^{\text{T}}(t)$ factorization can be used to compute the preconditioners without significantly deteriorating their numerical performances.

4. PARALLEL PERFORMANCE

The independent solution of local PDE problems expressed by the domain decomposition techniques is particularly suitable for parallel-distributed computation. In a parallel-distributed memory environment each subdomain is assigned to a different processor. Using the research software MUMPS [22] corresponding to the state of the art in sparse direct solvers we have implemented $M_{\text{BPS-}E}$, $M_{\text{BPS-}S}$, $\tilde{M}_{\text{BPS-}E}$ and $\tilde{M}_{\text{BPS-}S}$ on a parallel-distributed memory computer. This package is particularly suitable for our implementation since it enables to efficiently compute the local Schur complement using efficient sparse factorization techniques. For the dense computation involved in the factorization of S_{ii} and $\bar{S}^{(i)}$ we used the LAPACK [23] Cholesky factorization routines. The sparse computation involved in the factorization of the local Dirichlet problems, the construction of the local Schur complement matrices $S^{(i)}$, the factorizations required to build $\tilde{M}_{\text{BPS-}E}$ and $\tilde{M}_{\text{BPS-}S}$ were performed also using MUMPS. The exchanges between the processors were implemented using MPI. The target platform is a 57-node COMPAQ computer installed at CEA in Grenoble (France); each node is a 4-processor SMP.

To evaluate the numerical and parallel performance of the preconditioners we consider the heterogeneous anisotropic problem AD-F2 using 16 and 64 processors keeping constant the number of grid points per subdomain (i.e. each subdomain is a 256×256 mesh).

In Table XI we report the number of iterations for the four considered preconditioners, the threshold used to construct the preconditioners $\tilde{M}_{\text{BPS-}E}$ and $\tilde{M}_{\text{BPS-}S}$ enables to retain around 2.5 percent of the entries of the local Schur complement.

In Table XII we depict the elapsed time required to build the local preconditioner including the explicit computation of the local Schur complement. This time does not depend on the number of processors since the major part (i.e. 2.5 s) is spent in MUMPS for the factorization of the local Dirichlet problem and the computation of the local Schur complement. Notice that the assembling of the local Schur complement that requires some communication is negligible (i.e. about 0.2 s) and independent of the number of processors; the rest is spent in the Cholesky factorization dense or sparse depending on the preconditioner. Finally, it can be noticed that

Table XII. Elapsed time in seconds to build and factorize the preconditioners.

No. of subdomains	4×4	8×8
$M_{\text{BPS-}E}$	2.55	2.55
$M_{\text{BPS-}S}$	2.88	2.88
$\tilde{M}_{\text{BPS-}E}$	2.52	2.52
$\tilde{M}_{\text{BPS-}S}$	2.54	2.54

Table XIII. Elapsed time in seconds for a complete solution.

No. of subdomains	4×4	8×8
$M_{\text{BPS-}E}$	3.17	5.31
$M_{\text{BPS-}S}$	3.55	5.44
$\tilde{M}_{\text{BPS-}E}$	2.99	4.94
$\tilde{M}_{\text{BPS-}S}$	3.08	4.56

thanks to the performance of the dense linear algebra kernels LAPACK/BLAS-3, the cost of factorizing the assembled local Schur complement $\bar{S}^{(i)}$ involved in $M_{\text{BPS-}S}$ is only about seven times more expensive than that of factorizing the S_{ii} associated with the subdomains and required for $M_{\text{BPS-}E}$. We recall that the complexity of the former is 16 times larger than that of the latter. In our implementation we made the choice of redundantly factorizing on each processor that shares the edge E_i. The counterpart of this extra computation is to avoid one neighbour–neighbour communication when the local preconditioner is applied contributing to make the $M_{\text{BPS-}E}$ iteration faster than the $M_{\text{BPS-}S}$, the latter requires a communication to assemble the contribution computed by the processors that share an interface.

The overall elapsed time corresponding to the construction of the two components of the preconditioners plus the preconditioned conjugate gradient iterations to solve the Schur complement system are reported in Table XIII. It should be mentioned that since the local Schur complement is explicitly computed, the matrix vector product within the preconditioned conjugate gradient iteration is performed by a simple DGEMV BLAS-2 call involving a relatively small dense matrix, making the iteration extremely cheaper than in conventional Schur complement implementations where this step requires backward/forward substitution on the large sparse Cholesky factors associated with the local Dirichlet problem. The gain introduced by this dense matrix–vector product enables to overcome rapidly the extra cost due to the explicit computation of the local Schur complement; for our examples as soon as the convergence requires more than a dozen of iterations. Although it deteriorates the convergence speed of the iterative scheme, sparsifying the local Schur complement to build sparse preconditioners enables to noticeably reduce the overall elapsed time. Finally, even though constructing and applying the preconditioner is computationally more expensive for $\tilde{M}_{\text{BPS-}S}$ than for $\tilde{M}_{\text{BPS-}E}$, the reduction in terms of iterations often gives rise to the fastest method. The gain becomes larger as the problem becomes harder. In this situation, the robustness of $\tilde{M}_{\text{BPS-}S}$ enables much faster convergence than $\tilde{M}_{\text{BPS-}E}$; this is, in particular, the case in device modeling simulation [21].

5. CONCLUDING REMARKS

We have introduced two new local preconditioners. They are based on an explicit computation of the local Schur complement matrices and can be used in combination with a coarse-space component in an additive way.

The first one, M_{VE}, aims at recovering some information relative to the interface nodes close to the vertices of the coarse mesh τ^H defined by the decomposition. This preconditioner shows some advantages over the simple block Jacobi preconditioner M_E for the solution of linear systems arising in the solution of parabolic problems. These advantages vanish for the solution of elliptic problems when, to ensure the numerical scalability, the coarse space preconditioner component smoothes its effect compared to M_E. For those problems, the use of approximate local solvers affects significantly the numerical behaviour of the resulting preconditioner $\tilde{M}_{\mathrm{BPS\text{-}VE}}$. For the solution of the linear system arising in the solution of parabolic problems, M_{VE} is a cheap alternative to improve the simple block Jacobi preconditioner. Both have similar computational complexities and parallel performances [18]. In addition, the use of approximate local Schur complement does not penalize significantly the numerical behaviour of M_{VE}.

The second one, closely related to the Neumann–Neumann preconditioner, demonstrates a very attractive numerical behaviour on heterogeneous and anisotropic problems. These problems appear, for instance, in the solution of the drift-diffusion equations involved in semiconductor device modeling. We propose two alternatives based on approximated local Schur complements built either using incomplete Cholesky factorizations or using sparsified exact local Schur complement. Based on an extensive benchmarking, we show that the resulting preconditioner, with a cheap construction, retains the main numerical features of $M_{\mathrm{BPS}\text{-}S}$ and is well adapted for an efficient implementation on parallel-distributed memory computers.

ACKNOWLEDGEMENTS

Part of this work was performed while the first author was a Ph.D. student at CERFACS or was visiting CERFACS after he had graduated: the hospitality and support of CERFACS are greatly appreciated. The first author would also like to thank Pr. Nelson Maculan Filho (COPPE/UFRJ, Brazil) who welcomed him in his research group during part of this work. The second author would like to thank Michele Benzi for fruitful discussions on the AINV preconditioner and two-level AINV techniques as well as on some aspects related to the work presented in this paper. Finally, the authors would like to thank Fernando Guevara Vasquez who performed the parallel experiments whose results are reported in this paper.

REFERENCES

1. Bramble JH, Pasciak JE, Schatz AH. The construction of preconditioners for elliptic problems by substructuring I. *Mathematics of Computations* 1986; **47**(175):103–134.
2. Dryja M, Smith BF, Widlund OB. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. *SIAM Journal of Numerical Analysis* 1993; **31**(6):1662–1694.
3. Smith BF. *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*. PhD Thesis, Courant Institute of Mathematical Sciences, September 1990. Technical Report 517, Department of Computer Science, Courant Institute.
4. Mandel J. Balancing domain decomposition. *Communications in Numerical Methods in Engineering* 1993; **9**:233–241.
5. Mandel J, Brezina M. Balancing domain decomposition: Theory and computations in two and three dimensions. Technical Report UCD/CCM 2, Center for Computational Mathematics, University of Colorado at Denver, 1993.

6. Farhat C, Roux F-X. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering* 1991; **32**:1205–1227.
7. Chan TF, Mathew TP. *Domain Decomposition Algorithms*, volume 3 of *Acta Numerica*, Cambridge University Press: Cambridge, 1994; 61–143.
8. Quarteroni A, Valli A. *Domain decomposition methods for partial differential equations*. Numerical mathematics and scientific computation. Oxford science publications: Oxford, 1999.
9. Smith BF, Bjørstad P, Gropp W. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations* (1st edn). Cambridge University Press: New York, 1996.
10. De Roeck Y-H, Le Tallec P. Analysis and test of a local domain decomposition preconditioner. In Glowinski R, Kuznetsov Y, Meurant G, Périaux P, Widlund O. editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM: Philadelphia, PA, 1991; 112–128.
11. Bourgat JF, Glowinski R, Le Tallec P, Vidrascu M. Variational formulation and algorithm for trace operator in domain decomposition calculations. In Chan T, Glowinski R, Périaux J, Widlund O. editors, *Second International Conference on Domain Decomposition Methods*. SIAM, Philadelphia, PA, 1989.
12. Benzi M, Meyer CD, Tůma M. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal of Scientific Computing* 1996; **17**(5):1135–1149.
13. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1994.
14. Meijerink JA, van der Vorst HA. An iterative solution method for linear systems for which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computations* 1977; **31**(137):148–162.
15. Benzi M, Tůma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM Journal of Scientific Computing* 1998; **19**(3):968–994.
16. Carvalho LM, Giraud L, Le Tallec P, Algebraic two-level preconditioners for the Schur complement method. Tech. Rep. TR/PA/98/18, CERFACS, France, 1998. available on `http://www.cerfacs.fr/algor/algo_reports.html`; to appear in *SIAM Journal of Scientific Computing*.
17. Le Tallec P. *Domain decomposition methods in computational mechanics*, volume 1 of *Computational Mechanics Advances*, North-Holland: Amsterdam, 1994; 121–220.
18. Carvalho LM. *Preconditioned Schur complement methods in distributed memory environments*. PhD Thesis, INPT/CERFACS, Toulouse, France, October 1997. TH/PA/97/41, CERFACS.
19. Giraud L, Guevara Vasquez F, Rioual J-C. Non-overlapping domain decomposition in parallel semi-conductor device modeling. Technical Reports in preparation, CERFACS, France, 2000.
20. Chan TF, Mathew TP, Shao J-P. Fourier and probe variants of the vertex space domain decomposition algorithm. In Keyes D, Chan TF, Meurant G, Scroggs J, Voigt R. editors, *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*. SIAM: Philadelphia, 1992; 236–249.
21. Meurant G. Numerical experiments with a domain decomposition method for parabolic problems. In Glowinski R, Kuznetsov Y, Meurant GA, Périaux J, Widlund OB. editors, *Proceedings of Fourth International Conference on Domain Decomposition Methods* SIAM: Philadelphia, 1991; 394–408.
22. Amestoy PR, Duff IS, L'Excellent J-Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering* 2000; **184**:501–520.
23. Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Oustrouchov S, Sorensen D. *LAPACK – Users' Guide*. SIAM: Philadelphia, 2nd edition, 1995.

