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

L. M. Carvalho[*] L. Giraud[†] G. Meurant[‡]

CERFACS Technical Report TR/PA/99/38 - November 1999

Abstract

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 behavior 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 behavior. We compare their numerical performance with well-known robust preconditioners such as BPS [6] and the balanced Neumann-Neumann method [15].

Keywords : 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 by 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) [6] and Vertex Space [12, 19] for additive two-level preconditioners, and to Balancing Neumann-Neumann [15, 16], as well as FETI [13] for examples of multiplicative ones. We refer to [10] and [20] for a more exhaustive overview of domain decomposition techniques.

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 [19] and Neumann-Neumann [11]. 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.

[*]COPPE-UFRJ, Brazil and IME-UERJ, Brazil. This work was partially supported by FAPERJ-Brazil under grant 150.177/98.

[†]CERFACS, 42 av. Gaspard Coriolis, 31057 Toulouse Cedex, France.

[‡]CEA/DIF, DCSA, BP12, 91680 Bruyeres le Chatel, France.

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 behaviors with well-known robust preconditioners such as the balanced Neumann-Neumann method [15]. Finally, 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.

2 Preconditioner description

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

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

where $\partial\Omega_{\text{Dirichlet}} \cap \partial\Omega_{\text{Neumann}} = \emptyset$ and $a(x,y)$, $b(x,y) \in I\!R^2$ 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:

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

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

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

where

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

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 $\Gamma_i = \partial\Omega_i - \partial\Omega$. 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 $R_{\Gamma_i}^T : \Gamma_i \to \Gamma$ be its transpose. For a stiffness matrix A arising from a finite element discretization, the Schur complement matrix (4) can also be written as:

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

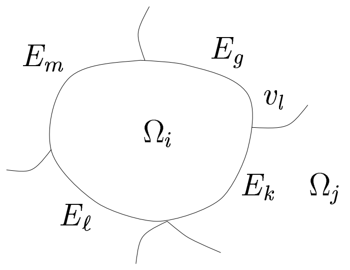

Figure 1: An internal subdomain

where

$$S^{(i)} = A^{(i)}_{\Gamma_i} - A^T_{i\Gamma_i} A^{-1}_{ii} A_{i\Gamma_i} \tag{5}$$

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

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

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:

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

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 is $S_{kk} = S^{(i)}_{kk} + S^{(j)}_{kk}$. 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 neighbor to neighbor communications enable each processor to get its $\bar{S}^{(i)}$ once $S^{(i)}$ has been compuuted 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:

$$U = U_1 + ... + U_p.$$

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

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

where

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

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) then that M_{loc} is SPD.
Let $< .,. >$ denotes the scalar product associated with the 2-norm.

- M_i^{-1} is SPD is equivalent to show that M_i is SPD.
  By definition M_i is symmetric.

  $$\begin{aligned} \forall x \neq 0 < x, M_i x > &= < x, R_i S R_i^T x > \\ &= < R_i^T x, S R_i^T x > \end{aligned}$$

  In addition

  $$\left.\begin{array}{l} R_i \text{ is full rank} \Rightarrow R_i^T x \neq 0 \\ S \text{ is SPD} \end{array}\right\} \Rightarrow < R_i^T x, S R_i^T x > \text{ is strictly positive.}$$

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

  $$< x, M_{loc} x > = < x, \sum_{i=1}^{p} R_i^T M_i^{-1} R_i x > = \sum_{i=1}^{p} < R_i x, M_i^{-1} R_i x >, \tag{9}$$

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

■

Remark 1 *If $U = U_1 \oplus .... \oplus U_n$, 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 requested to be efficient on parallel distributed memory platforms. Therefore, we do 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 neighborhs. This approach introduces only cheap neighbor to neighbor 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 neighbors of its ended 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, $S_{ii} = R_{E_i} S R_{E_i}^T = M_i$. Similarly, we consider R_{v_l} the restriction operator 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

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

This preconditioner aims at capturing the interaction between neighboring 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 neighbor 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 [19], for which we merge into a single subspace the edge and vertex subspaces that appear in an additive way in [12, 19].

In Figure 2, we depict U_k, the image of the restriction operator $R_k \equiv R_{VE_k}$ associated with the vertex-edge E_k. With this notation the vertex-edge preconditioner is defined by

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

In that case, two neighbor 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 neighborhood of the vertex v_l defines the amount of overlap between the blocks M_i of the preconditioner.

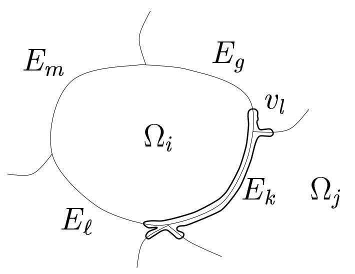

Figure 2: U_k associated to the vertex-edge preconditioner.

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 [5] and [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 [3] 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 an Stieltjes matrix.*

Proof
It is enough to show that

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

since Theorem 7.1 in [2] will then insure that the resulting approximate Schur is a M-matrix. By construction $\tilde{S}$ is symmetric then is a Stieltjes matrix consequently SPD.
A_{II} is a symmetric M-Matrix, so by Theorem 2.4 [17], $A_{II} = (LL^T) - R$ is a regular splitting (ie. $(LL^T)^{-1} \geq 0$ and $R \geq 0$).

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

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

$$0 \leq (LL^T)^{-1} \leq A_{II}^{-1}.$$

■

We note that the same property holds for approximate Schur complement computed with AINV. In [4] it is shown that the approximate inverse G of a M-matrix A computed by AINV also satisfies the inequality $0 \leq G \leq A^{-1}$.

Notice that Lemma 1 and 2 insure that for M-matrices the local preconditioner 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:

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

Lemma 3 *If the matrix* $A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{I\Gamma}^T & A_{\Gamma\Gamma} \end{pmatrix}$ *is a Stieltjes matrix then the sparse approximation* $\hat{S}$ *computed by* (12) *applied to* $S = A_{\Gamma\Gamma} - A_{I\Gamma}^T A_{II}^{-1} A_{I\Gamma}$ *is also an Stieltjes matrix.*

Proof
It is well-known that S is a Stieltjes matrix (see [2] for instance), then it is easy to see that removing off diagonal entries while preserving symmetry preserves this property.

■

The two alternatives can be combined, that is dropping the smallest entries of approximate M_i, to produce preconditioner cheap to compute and to store. We note that, for M-matrices, this combination gives rise to preconditioner 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 [10]) that the local preconditioners alone are not numerically scalable for elliptic problems in the sense that

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

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 from 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 [6] that, when applying the original BPS technique to a uniformly elliptic operator, the preconditioned system has a condition number

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

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

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

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

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 $A_0 = R_0 S R_0^T$, 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

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

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. The coarse space matrix A_0 can be computed using four matrix-vector products in a distributed memory environment. We refer to [8] 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_{BPS-E} for $M_{loc} = M_E$. Notice that this local preconditioner is the one used in the genuine BPS; in this respect M_{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_{BPS-VE} for $M_{loc} = M_{VE}$.
- M_{BPS-S} for $M_{loc} = M_S$.

3 Numerical experiments

In this section, we report through a set of model problems the numerical behavior of the preconditioners introduced in Section 2. We consider not only the new BPS variants but, additionally, the well established balanced Neumann-Neumann preconditioner [15]. Before reporting the comparison results, we briefly recall the balanced Neumann-Neumann preconditioner.

The balanced Neumann-Neumann preconditioner has proven to be an efficient domain decomposition preconditioner for some fairly difficult problems [14], 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 [15] 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 [7]. 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).

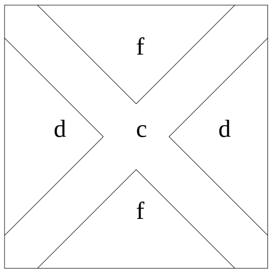

Figure 3: Example 1 - Flag

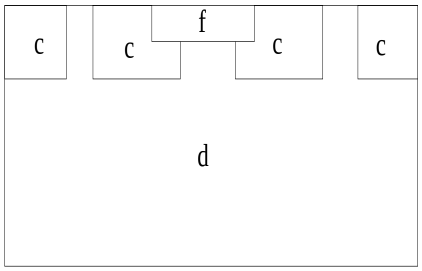

Figure 4: Example 2 - Region

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 $b = c, d$ or f which depend on x and y.
  - AD-F1: c=1, d= 10^2 and f= 10^{-2}.
  - AD-F2: c=1, d= 10^3 and f= 10^{-3}.
- discontinuous problems with $a = b = c, d, f$.
  - D-F1: c=1, d= 10^2 and f= 10^{-2}.
  - D-F2: c=1, d= 10^3 and f= 10^{-3}.

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 $b = c, d$ or f.
  - AD-R: c= 10^1 , d= 10^{-2} and f= 10^{-1}.
- discontinuous problems: $a = b = c, d, f$.
  - D-R: c= 10^1 , d= 10^{-2} and f= 10^{-1}.

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. For $\theta = 0$, this corresponds to the classical model anisotropic equation:

$$\varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \text{ with } \varepsilon \ll 1. \tag{15}$$

3.1.2 Elliptic problems involved in the solution of parabolic linear equations

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

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

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 centered 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

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

so we have to solve

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

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

$$A_t \delta = h^2(f^{m+1} + f^m) - 2Au^m, \tag{16}$$

then advance the solution in time

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

The linear system (16) 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 neighborhood of the vertices v_l in each direction. For a more detailed study about the influence of the size of the overlap in the neighborhood of the vertices v_l on the convergence rate, we refer to [7]. We just state here that a very small overlap is usually enough to improve the behavior of M_{VE} with respect to M_E. Both preconditioners have a comparable computational complexity and consequently a similar parallel performance [7].

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 the size of each subdomain constant (i.e. 16×16 mesh for each subdomain, that is $\frac{H}{h} = 16$); the initial guess x_0 for the conjugate gradient iterations was the null vector.

3.2.1 Anisotropic and discontinuous elliptic problems

In Table 1, 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. Accordingly 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_{BPS-E} and M_{BNN}, the results are similar to those of other authors [9], [15].

	4×4	8×8	16×16
M_E	13	28	51
M_{VE}	12	22	40
M_S	11	19	32
M_{BPS-E}	9	11	11
M_{BPS-VE}	10	12	12
M_{BPS-S}	10	10	11
M_{BNN}	11	12	12

Table 1: # iterations on the Poisson problem.

In Table 2, we depict the numerical behavior of the preconditioners on the model problems 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 to have an optimal convergence behavior independent of the number of subdomains, even though the number of iterations is quite decreased by the coarse space component. Furthermore, for some problems M_{BPS-VE} becomes less efficient than the simpler M_{BPS-E} while M_{BPS-S} always 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.

	4×4			8×8			16×16		
	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_{BPS-E}	27	24	20	58	34	28	81	43	35
M_{BPS-VE}	25	21	21	48	33	35	85	43	49
M_{BPS-S}	20	19	17	33	26	21	47	33	26

Table 2: # iterations - Anisotropy ($\varepsilon = 10^{-3}$) with several angles.

In Tables 3 and 4, we study the numerical behavior of the two-level preconditioners on model problems arising from the discretization of (1) that exhibit either discontinuity (Table 3) or both discontinuity and anisotropy (Table 4). For the problems with only discontinuity, all the variants M_{BPS-*} have comparable convergence behaviors.

As it can be seen in Table 4, problems with anisotropy and discontinuity are more difficult to solve. Again M_{BPS-VE} does not outperform the basic M_{BPS-E}. For those examples, similarly to the pure anisotropic situation reported in Table 2, M_{BPS-S} exhibits once again the best convergence behavior.

[1]'* 'means no convergence after 1000 iterations

# subdomains	4 × 4			8 × 8			16 × 16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
M_{BPS-E}	12	11	10	11	11	11	14	15	11
M_{BPS-VE}	13	12	11	13	12	12	16	16	12
M_{BPS-S}	12	10	10	11	11	11	14	14	11
M_{BNN}	25	27	21	29	28	38	48	65	52

Table 3: # iterations for problems with discontinuity.

# subdomains	4 × 4			8 × 8			16 × 16		
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
M_{BPS-E}	18	24	25	29	65	35	42	103	39
M_{BPS-VE}	18	23	24	33	80	40	56	141	55
M_{BPS-S}	16	20	18	22	43	22	33	79	26
M_{BNN}	37	52	147	60	158	644	97	311	*[1]

Table 4: # iterations for problems with discontinuity and anisotropy.

The relative poor performance of M_{BNN}, reported in Table 3 and 4, could be improved. An alternative way, as suggested in [11], should be a better choice of the weight matrices D_i, involved in Equation (11), when the diagonal entries of S are available. With this appropriated choice of the weights, it can be expected a reduction of the gap between M_{BNN} and M_{BPS-*} for discontinuous problems, as suggested by the results reported in [16]. 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 5 we report the numerical behavior of M_{BPS-S} and M_{BNN} for

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

Table 5: # iterations varying the anisotropy with a 8 × 8 subdomain decomposition.

the anisotropic problems defined by Equation (15) for different values of the anisotropic coefficient ε is varied. For those problems, the choice of the weighted matrices used in [16] for M_{BNN} would reduce to the simple ones we have considered; that is, $\frac{1}{2}$ for the nodes on the edges and $\frac{1}{4}$ for the vertex points v_l. For anisotropic problems, we cannot expect M_{BNN} to become competitive with M_{BPS-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. $ILL^T(0)$, or with some fill-in controlled through a threshold, i.e. $ILL^T(t)$. 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 $ILL^T(0)$.

In Table 6 and 7, we denote by $\tilde{M}_{BPS-E}$, $\tilde{M}_{BPS-VE}$ and $\tilde{M}_{BPS-S}$ the preconditioners computed using those inexact local solves. More precisely, we report in Table 6 the number of iterations when $ILL^T(0)$ is used and in Table 7 those observed when some fill-in is enabled with a fill-in ratio lower than 3.5.

# subdomains	4 × 4			8 × 8			16 × 16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{BPS-E}$	14	13	14	13	13	14	17	17	14
$\tilde{M}_{BPS-VE}$	20	18	20	19	19	19	24	26	20
$\tilde{M}_{BPS-S}$	14	13	15	13	13	12	17	18	13
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{BPS-E}$	24	30	28	36	67	37	50	112	45
$\tilde{M}_{BPS-VE}$	27	34	31	40	84	48	64	143	64
$\tilde{M}_{BPS-S}$	24	26	23	30	53	31	47	80	41

Table 6: # iterations using inexact local solvers $ILL^T(0)$ to build the preconditioners.

# subdomains	4 × 4			8 × 8			16 × 16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\tilde{M}_{BPS-E}$	13	12	12	13	14	12	16	19	11
$\tilde{M}_{BPS-VE}$	15	17	12	17	19	13	22	27	12
$\tilde{M}_{BPS-S}$	12	12	10	11	12	11	16	18	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\tilde{M}_{BPS-E}$	23	31	28	35	70	37	48	114	40
$\tilde{M}_{BPS-VE}$	21	33	26	36	85	44	59	147	56
$\tilde{M}_{BPS-S}$	19	27	20	27	54	24	39	81	29

Table 7: # iterations using inexact local solvers $ILL^T(t)$ to build the preconditioners.

The comparison of the results depicted in Table 6 and 7 and those in Tables 3 and 4 shows that the approximation of the local Schur complement used to build the preconditioners generally deteriorates the numerical behaviors of the preconditioner. This approximation does not affect significantly the numerical behavior of $\tilde{M}_{BPS-E}$ and $\tilde{M}_{BPS-S}$ but deteriorates noticeably the one of $\tilde{M}_{BPS-VE}$. 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 $\tilde{M}_{BPS-VE}$ and $\tilde{M}_{BPS-S}$.

Sparse approximation of the Schur complement In Table 8 we report the number of iterations using an approximate Schur complement $\hat{S}$ with η in (12) such that we only retains around 5 % of the entries in S. The resulting preconditioners are denoted respectively by $\hat{M}_{BPS-E}$, $\hat{M}_{BPS-VE}$ and $\hat{M}_{BPS-S}$.

# subdomains	4 × 4			8 × 8			16 × 16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\hat{M}_{BPS-E}$	13	12	12	11	11	13	14	15	12
$\hat{M}_{BPS-VE}$	16	16	18	16	16	18	20	20	18
$\hat{M}_{BPS-S}$	12	11	12	12	12	11	15	16	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\hat{M}_{BPS-E}$	18	25	27	29	65	35	43	111	40
$\hat{M}_{BPS-VE}$	24	30	26	36	81	42	57	142	57
$\hat{M}_{BPS-S}$	18	21	18	23	44	23	36	79	27

Table 8: # iterations using sparse Schur to build the preconditioners.

The comparison of these results with those displayed in Table 3 and 4 indicates that, except for $\hat{M}_{BPS-VE}$ 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 are roughly the same in both cases (except for $\hat{M}_{BPS-VE}$ on discontinuous problems).

In addition, as mentioned in Section 2.2.2, the inexact local solvers and dropping strategies can be combined to build variants of the preconditioners. The resulting preconditioners are respectively denoted by $\underline{M}_{BPS-E}$, $\underline{M}_{BPS-VE}$ and $\underline{M}_{BPS-S}$. Numerical experiments where we dropped the smallest elements of the local preconditioners built using $ILL^T(t)$ are reported in Table 9. Comparing these results with those of Tables 8 and 7 indicates that the numerical quality of the resulting preconditioners are mainly governed by the use of ILL^T.

# subdomains	4 × 4			8 × 8			16 × 16		
	D-F1	D-F2	D-R	D-F1	D-F2	D-R	D-F1	D-F2	D-R
$\underline{M}_{BPS-E}$	14	13	12	13	13	13	16	18	13
$\underline{M}_{BPS-VE}$	19	18	18	20	22	18	26	29	18
$\underline{M}_{BPS-S}$	12	12	12	12	13	11	17	19	11
	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R	AD-F1	AD-F2	AD-R
$\underline{M}_{BPS-E}$	22	32	29	35	70	36	47	113	41
$\underline{M}_{BPS-VE}$	26	38	27	39	86	45	61	150	58
$\underline{M}_{BPS-S}$	19	29	21	28	55	28	42	81	31

Table 9: # iterations using preconditioner based on sparse Schur built using inexact local solvers $ILL^T(t)$.

3.2.2 Elliptic problems in the solution of parabolic equations

In Table 10, 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 an anisotropic equation with the anisotropy not necessarily aligned with the x or y axis, but making an angle θ. The time step and the mesh size are such that $\mu = 0.02$, which gives raise to a well conditioned linear system (16) (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 [18]. On those examples M_{VE} is generally between 20 % up-to 40 % faster than M_E, while both, as already noticed, have a similar computational complexity [7]. 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 of M_S. Furthermore, for M_{VE} this factorization is performed only once for each edge E_k compared to two factorizations for M_S as each E_k is shared by two subdomains. As before, approximate local Schur complements based on $ILL^T(t)$ factorization can be used to compute the preconditioners without significantly deteriorating their numerical performances.

	4×4			8×8			16×16		
	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
$\tilde{M}_{VE}$	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

Table 10: Diffusion with several angles - $\varepsilon = 10^{-3}$.

4 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 affect significantly the numeriacl behavior of the resulting preconditioner $\tilde{M}_{BPS-VE}$. 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 complexity and parallel performance [7]. In addition, the use of approximate local Schur complement does not penalize significantly the numerical behavior of M_{VE}.

The second one, closely related to the Neumann-Neumann preconditioner, demonstrates a very attractive numerical behavior on heterogeneous and anisotropic problems. These problems appear, for instance, in the solution of the drift-diffusion equations involved in semi-conductor device modeling. An efficient implementation of the local Schur complement construction may directly benefit, in the future, from the ongoing development of advanced sparse direct solvers like MUMPS [1]. However, we propose an alternative based on approximated local Schur complements built thanks to incomplete Cholesky factorizations. Based on an extensive benchmarking, we show that the resulting preconditioner, with a cheap construction, retains the main numerical features of M_{BPS-S}.

Acknowlegments

Part of this work was performed while first author was a PhD student at CERFACS or was visiting CERFACS after he was graduated: the hospitality and support of CERFACS are greatly appreciated. The second author would like to thank Michele Benzi for fruitful discussions on the AINV and two-level AINV techniques as well as on some aspects related to the work presented in this paper.

References

[1] P.R. Amestoy, I.S. Duff, and J.Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *to appear in special issue of Comput. Methods in Appl. Mech. Eng. on domain decomposition and parallel computing*, 1998.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[3] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.

[4] M. Benzi and M. Tůma. Approximate inverse preconditioning of iterative methods for non-symmetric linear systems. *SIAM*, 19(3):968–994, 1998.

[5] J.F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu. Variational formulation and algorithm for trace operator in domain decomposition calculations. In T. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Second International Conference on Domain Decomposition Methods.* SIAM, Philadelphia, PA, 1989.

[6] J.H. Bramble, J.E. Pasciak, and A.H. Schatz. The construction of preconditioners for elliptic problems by substructuring I. *Math. Comp.*, 47(175):103–134, 1986.

[7] L.M. Carvalho. *Preconditioned Schur complement methods in distributed memory environments.* PhD thesis, INPT/CERFACS, Toulouse, France, october 1997. TH/PA/97/41, CERFACS.

[8] L.M. Carvalho, L. Giraud, and P. Le Tallec. Algebraic two-level preconditioners for the Schur complement method. Tech. Rep. TR/PA/98/18, CERFACS, France, 1998.

[9] T. Chan, T. Mathew, and J-P. Shao. Fourier and probe variants of the vertex space domain decomposition algorithm. In D. Keyes, T. Chan, G. Meurant, J. Scroggs, and R. Voigt, editors, *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 236–249, Phil., 1992. SIAM.

[10] T.F. Chan and T.P. Mathew. *Domain Decomposition Algorithms*, volume 3 of *Acta Numerica*, pages 61–143. Cambridge University Press, Cambridge, 1994.

[11] Y.-H. De Roeck and P. Le Tallec. Analysis and test of a local domain decomposition preconditioner. In R. Glowinski, Y. Kuznetsov, G. Meurant, J. Périaux, and O. Widlund, editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 112–128. SIAM, Philadelphia, PA, 1991.

[12] M. Dryja, B.F. Smith, and O.B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. *SIAM J. Numer. Anal.*, 31(6):1662–1694, 1993.

[13] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engng.*, 32:1205–1227, 1991.

[14] P. Le Tallec. *Domain decomposition methods in computational mechanics*, volume 1 of *Computational Mechanics Advances*, pages 121–220. North-Holland, 1994.

[15] J. Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9:233–241, 1993.

[16] J. Mandel and M. Brezina. 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.

[17] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems for which the coefficient matrix is a symmetric M-matrix. *Mathematics od Computations*, 31(137):148–162, 1977.

[18] G. A. Meurant. Numerical experiments with a domain decomposition method for parabolic problems. In R. Glowinski, Yu. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. B. Widlund, editors, *Proc. Fourth Int. Conf. on Domain Decomposition Meths.*, pages 394–408, Philadelphia, 1991. SIAM.

[19] B.F. Smith. *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity.* PhD thesis, Courant Institute of Mathematical Sciences, September 1990. Tech. Rep. 517, Department of Computer Science, Courant Institute.

[20] B.F. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1st edition, 1996.