ELSEVIER

Applied Numerical Mathematics 19 (1995) 129–146

APPLIED NUMERICAL MATHEMATICS

Incomplete block LU preconditioners on slightly overlapping subdomains for a massively parallel computer

E. de Sturler *

Interdisciplinary Project Center for Supercomputing (IPS-ETH Zürich), ETH-Zentrum, CH-8092 Zürich, Switzerland

Abstract

The ILU (Meijerink and van der Vorst, 1977; 1981) and MILU (Gustafsson, 1978) preconditioners have become more or less the standard for preconditioning. On parallel computers, however, the inherent sequentiality of these preconditioners precludes efficient implementation. Replacing (M)ILU by blocked variants may seem a good way to improve the parallelism, but generally these blocked variants come with a penalty in the form of more iterations.

We will consider possibilities to improve the convergence of the block preconditioners by adapting ideas from Radicati and Robert (1989) and Tang (1992). We will use as preconditioner(s) the incomplete factorizations of the local systems of equations corresponding to slightly overlapping subdomains with certain parameterized algebraic boundary conditions. Although the selection of the optimal parameters is still an open problem, numerical experiments suggest that the iteration count can be almost constant when going from the sequential case to as much as 400 subdomains. We will also give details on the performance on a 400-processor parallel computer.

Keywords: Preconditioners; Parallel computing; Distributed-memory computers

1. Introduction

The efficiency of iterative solvers on massively parallel computers depends on two factors. The implementation must be efficient, and the increase in the number of iterations, compared with the (best) sequential implementation, must be low. In this paper we will focus on the GMRES(m) method [6] with ILU preconditioning. In [1] it has been shown how to implement this method efficiently using only ILU factorization per subdomain. The good efficiencies in [1] resulted from the reduction in the communication overhead for the inner products. However, the increase in the number of iterations due to less effective preconditioners has not been considered there. We will focus on this aspect here.

* E-mail:sturler@ips.id.ethz.ch. The author wishes to acknowledge Shell Research B.V. and STIPT for the financial support of this research.

SSDI 0168-9274(95)00077-1

We are concerned with preconditioners for massively parallel computers. These preconditioners should preferably have the following characteristics:

- They should have a convergence rate close to that for the ILU factorization over the entire domain.
- They should lead to only moderate communication cost at most.
- They should have a degree of parallelism that allows efficient implementation on a massively parallel computer.
- They should allow for a (relatively) simple implementation.

Replacing the ILU factorization over the entire domain by block versions, where a block corresponds to the equations for the unknowns in a subdomain, satisfies all of our requirements except the first one. Therefore, we will try to improve the convergence of such localized ILU preconditioners. We will refer to these preconditioners as Incomplete Block LU (IBLU) preconditioners.

In [5] Radicati and Robert suggest to use incomplete factorizations of slightly overlapping blocks of the matrix as a block preconditioner. For a small number of shared-memory processors (and hence a small number of matrix blocks) they report good results. Their matrix decomposition is based on the (possibly reordered) matrix itself and does not require any information on the underlying problem or physical domain, since they aim at black box solvers for sparse matrices with a general structure.

However, in a domain decomposition situation we can apply an analogous strategy using incomplete factorizations of slightly overlapping subdomains. In this case, the physical meaning of the overlap is known and can perhaps be exploited by using, for example, certain artificial boundary conditions. This leads to (parameterized) matrix splittings, so-called generalized Schwarz enhanced matrices (SEMs), of a form proposed by Tang in [9] for block Jacobi and block Gauss–Seidel iterations. We will use these parameterized Schwarz enhanced matrices as the basis for the IBLU preconditioners. In this case, there is no need for special regularity requirements on the generalized Schwarz splitting, and we do not need exact solvers on the subdomains. This gives us more freedom for the specification of the generalized Schwarz enhanced matrix than in the approach in [9].

In Section 2 we will discuss the construction of the generalized SEM and the preconditioner. We will show how the preconditioner can be used for a Krylov subspace method in Section 3. Finally, in Section 4, we use series of experiments with different overlaps and different parameter choices to gain some idea of the convergence behaviour when more subdomains (blocks) are used. We will report results from experiments on a 400-processor parallel computer (at the Koninklijke/Shell-Laboratorium in Amsterdam).

2. Construction of the preconditioner

In this section we discuss the construction of the generalized Schwarz enhanced matrix on which the IBLU factorization is based. The IBLU factorization of this matrix is then defined by the factorization of the diagonal blocks, which correspond to the subdomains. To simplify the explanation we will first describe the one-dimensional decomposition in detail; after that, we will describe the extension to higher-dimensional decompositions (as implemented on the 400-processor parallel computer) and to decompositions for more general meshes.

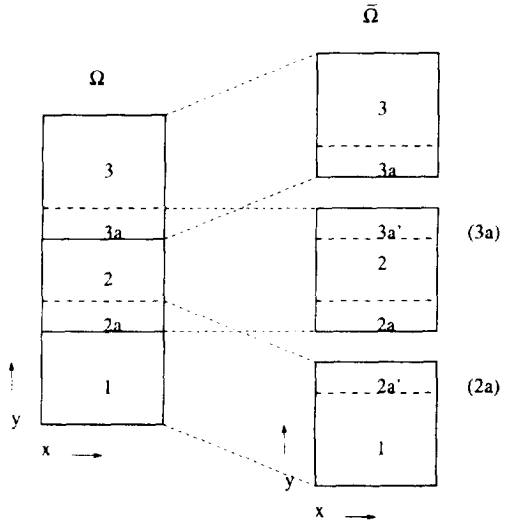

Fig. 1. A one-dimensional decomposition of the domain.

2.1. One-dimensional decomposition

We assume a lexicographical ordering of the unknowns, a rectangular domain and the block tridiagonal matrix which is derived by the well-known 5-point discretization star for two-dimensional Poisson-type problems.

We make a decomposition in the y-direction with slightly overlapping subdomains as in Fig. 1. The decomposition corresponds to matrices as shown in Figs. 2 and 3, which show the original matrix and the matrix for the decomposed domain. The latter matrix is referred to as the Schwarz enhanced matrix (SEM), see also [8], [9]. Fig. 5 gives a more detailed picture of the matrix structure for the duplicated grid lines corresponding to an overlap as indicated in Fig. 4.

For a block relaxation iteration, where a block corresponds to a subdomain, we need an approximation to the solution on the neighbouring grid lines of each subdomain (the exterior), before we can solve the local equations. Usually, we take the approximate solution of a previous iteration (Jacobi) or an intermediate approximation (Gauss–Seidel). This amounts to a Dirichlet boundary condition; the solution on the boundary of the subdomain is given.

However, we can also define the boundary conditions on the artificial boundaries for each subdomain in a more general way. Since the subdomains overlap, the artificial boundary of one

$$\left(\begin{array}{c|cc|cc} A_{1,1} & A_{1,2a} & & & \\ \hline A_{2a,1} & A_{2a,2a} & A_{2a,2} & & \\ & A_{2,2a} & A_{2,2} & A_{2,3a} & \\ & & A_{3a,2} & A_{3a,3a} & A_{3a,3} \\ & & & A_{3,3a} & A_{3,3} \end{array}\right)$$

Fig. 2. The matrix corresponding to the original domain in Fig. 1.

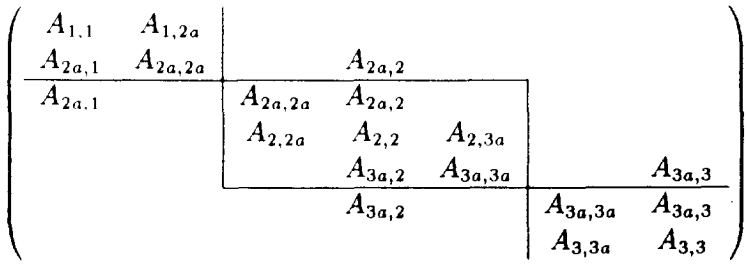

Fig. 3. The Schwarz enhanced matrix corresponding to the decomposition in Fig. 1.

subdomain is in the interior of another subdomain. The solution is obviously identical on each of the two duplicate parts, so that any function of the solution defined on these duplicate parts must give the same result on each part. We can select a function to define a relaxed consistency condition on the duplicate parts of the subdomain. Instead of requiring that the solution is equal on both parts, we require that this function of the solution is equal on both parts. Let the solution in subdomain P be u^P and in subdomain Q be u^Q. Then we have $g(u^P) = g(u^Q)$ for arbitrary g on the overlap of the subdomains (more specifically, on the artificial boundaries). For example, in [9] boundary conditions of the type

$$\omega u^P + (1-\omega)\frac{\partial u^P}{\partial n} = \omega u^Q + (1-\omega)\frac{\partial u^Q}{\partial n}$$

are used. For $\omega = 1$ this results in a Dirichlet boundary condition, for $\omega = 0$ in a Neumann boundary condition, and for $0 < \omega < 1$ in a mixed boundary condition. We can use such a consistency condition to define boundary conditions on the artificial boundaries. The substitution of the discretized boundary conditions into the SEM leads to a so-called generalized Schwarz splitting, which gives the generalized Schwarz enhanced matrix.

Since we will use the generalized SEM only for the construction of an IBLU preconditioner, we may use arbitrary algebraic equations for defining artificial boundary conditions. The objective is to find those algebraic conditions that lead to generalized Schwarz splittings which yield good preconditioners. For example, consider the subdomains P and Q, as shown in Fig. 4, where u_k corresponds to the approximation on the kth grid line in the y-direction and we have duplicated two grid lines, u_{k-1} and u_k; the small circles indicate that the grid line is not

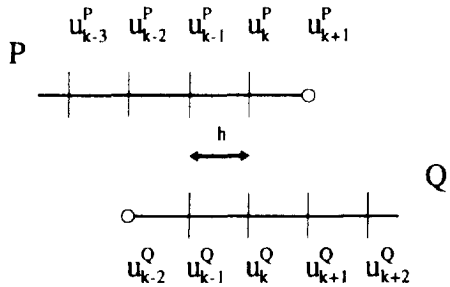

Fig. 4. Overlapping subdomains P and Q with an overlap of two grid lines. The small vertical lines u_k indicate the kth grid line in the original domain. The small circles indicate that the grid line is not included in the subdomain.

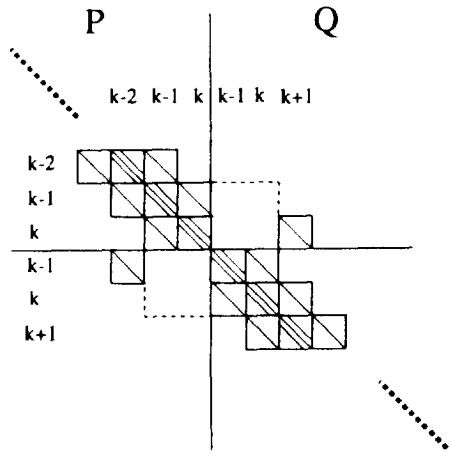

Fig. 5. The part of the SEM corresponding to the overlap of the subdomains P and Q. The indices $k-2,\ldots,k+1$, indicate the grid lines in the original domain.

included in the subdomain. The equations for the kth grid line (except at the domain boundaries) are

$$A_{k,k-1}u_{k-1} + A_{k,k}u_k + A_{k,k+1}u_{k+1} = b_k, \tag{1}$$

and the associated part of the SEM is given in Fig. 5. We define the following two boundary conditions for the two artificial boundaries:

$$\alpha_1 u_k^P - u_{k+1}^P = \alpha_1 u_k^Q - u_{k+1}^Q, \tag{2}$$

$$\alpha_2 u_{k-1}^P - u_{k-2}^P = \alpha_2 u_{k-1}^Q - u_{k-2}^Q. \tag{3}$$

These boundary conditions give equations for the exterior grid lines u_{k+1} (for subdomain P) and u_{k-2} (for subdomain Q), which can be substituted into the SEM,

$$u_{k+1}^P = \alpha_1 u_k^P - \alpha_1 u_k^Q + u_{k+1}^Q, \tag{4}$$

$$u_{k-2}^Q = u_{k-2}^P - \alpha_2 u_{k-1}^P + \alpha_2 u_{k-1}^Q. \tag{5}$$

The substitution of (4) and (5) into the equations for u_k^P and u_{k-1}^Q (1) gives

$$A_{k,k-1}u_{k-1}^P + (A_{k,k} + \alpha_1 A_{k,k+1})u_k^P - \alpha_1 A_{k,k+1}u_k^Q + A_{k,k+1}u_{k+1}^Q = b_k, \tag{6}$$

$$A_{k-1,k-2}u_{k-2}^P - \alpha_2 A_{k-1,k-2}u_{k-1}^P + (A_{k-1,k-1} + \alpha_2 A_{k-1,k-2})u_{k-1}^Q + A_{k-1,k}u_k^Q = b_{k-1}. \tag{7}$$

This substitution for u_{k+1}^P and u_{k-2}^Q leads to the generalized SEM of which the part corresponding to the overlapping grid lines is given in Fig. 6. If we duplicate more than one grid line, we can also define "higher-order" boundary conditions. The boundary conditions

$$\beta_1 u_{k-1}^P + \alpha_1 u_k^P - u_{k+1}^P = \beta_1 u_{k-1}^Q + \alpha_1 u_k^Q - u_{k+1}^Q, \tag{8}$$

$$\beta_2 u_k^P + \alpha_2 u_{k-1}^P - u_{k-2}^P = \beta_2 u_k^Q + \alpha_2 u_{k-1}^Q - u_{k-2}^Q, \tag{9}$$

$$
\begin{array}{cc|ccc}
\ddots & & & & \\
A_{k-2,k-2} & A_{k-2,k-1} & & & \\
A_{k-1,k-2} & A_{k-1,k-1} & A_{k-1,k} & & \\
 & A_{k,k-1} & A_{k,k}+\alpha_1 A_{k,k+1} & -\alpha_1 A_{k,k+1} & A_{k,k+1} \\
\hline
A_{k-1,k-2} & -\alpha_2 A_{k-1,k-2} & A_{k-1,k-1}+\alpha_2 A_{k-1,k-2} & A_{k-1,k} & \\
 & & A_{k,k-1} & A_{k,k} & A_{k,k+1} \\
 & & & A_{k+1,k} & A_{k+1,k+1} \\
 & & & & \ddots
\end{array}
$$

Fig. 6. The parameterized matrix splitting derived from the artificial boundary conditions given in (2) and (3).

$$
\begin{array}{cc|ccc}
\ddots & & & & \\
A_{k-2,k-2} & A_{k-2,k-1} & & & \\
A_{k-1,k-2} & A_{k-1,k-1} & A_{k-1,k} & & \\
 & A_{k,k-1}+\beta_1 A_{k,k+1} & A_{k,k}+\alpha_1 A_{k,k+1} & -\beta_1 A_{k,k+1} & -\alpha_1 A_{k,k+1} \quad A_{k,k+1} \\
\hline
A_{k-1,k-2} & -\alpha_2 A_{k-1,k-2} & -\beta_2 A_{k-1,k-2} & A_{k-1,k-1}+\alpha_2 A_{k-1,k-2} & A_{k-1,k}+\beta_2 A_{k-1,k-2} \\
 & & & A_{k,k-1} & A_{k,k} \quad A_{k,k+1} \\
 & & & & A_{k+1,k} \quad A_{k+1,k+1} \\
 & & & & \ddots
\end{array}
$$

Fig. 7. The parameterized matrix splitting derived from the artificial boundary conditions given in (8) and (9).

lead to the generalized SEM of which the part corresponding to the overlapping grid lines is given in Fig. 7. It is easily verified that the generalized SEMs are formed by a parameterized splitting of the matrix blocks corresponding to the overlap.

2.2. *Higher-dimensional decompositions and general meshes*

It is complicated to work out the generalized SEM in the case of a two-dimensional decomposition using the approach described above, not to mention further generalization to three dimensions or general meshes. Therefore, instead of adapting the equations in the matrix directly, we adapt the discretization star on the boundaries of the subdomain. This makes it easier to work out the equations at any given grid point analogous to the approach for the one-dimensional decomposition.

Consider a two-dimensional decomposition as given in Fig. 8, where we have duplicated a number of grid lines in the x-direction as well as in the y-direction. Note that for some grid points duplicates exist on four subdomains. To keep the explanation simple we only give an example on a regular grid; however, the approach can be followed in the same way for an irregular grid.

Let the point (i, j) be on the east boundary of subdomain P, and let Q be the neighbouring subdomain in the east. The equation for point (i, j) is given by

$$su_{i,j-1} + wu_{i-1,j} + cu_{i,j} + eu_{i+1,j} + nu_{i,j+1} = b_{i,j}, \tag{10}$$

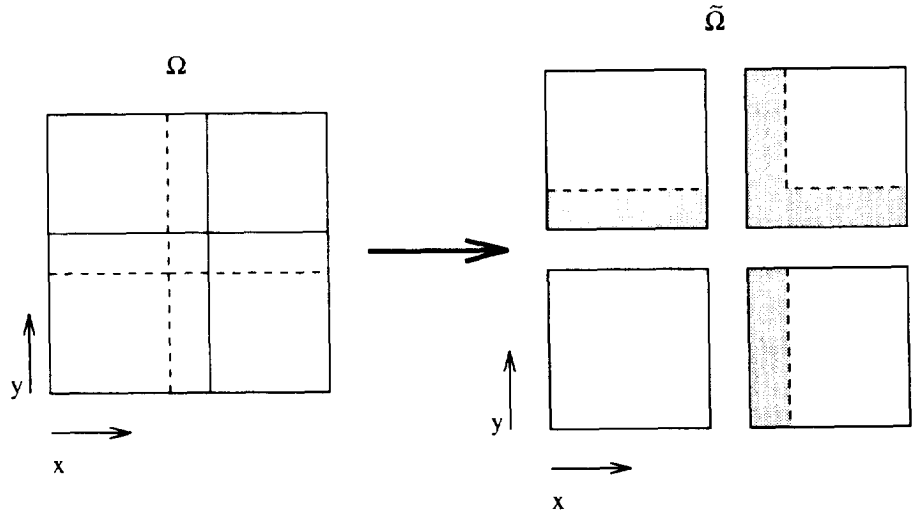

Fig. 8. A two-dimensional decomposition of the domain.

which is the standard 5-point discretization star, as shown in Fig. 9. On the artificial boundaries of the subdomain the discretization star is changed by the decomposition of the domain and the duplication of the grid lines as indicated in Fig. 9; the new discretization star includes the duplicates of the grid points to which it refers. We use a boundary condition like (8) for exterior points at the eastern boundary. This leads to

$$\beta u^P_{i-1,j} + \alpha u^P_{i,j} - u^P_{i+1,j} = \beta u^Q_{i-1,j} + \alpha u^Q_{i,j} - u^Q_{i+1,j}, \tag{11}$$

or

$$u^P_{i+1,j} = \beta u^P_{i-1,j} + \alpha u^P_{i,j} - \beta u^Q_{i-1,j} - \alpha u^Q_{i,j} + u^Q_{i+1,j}. \tag{12}$$

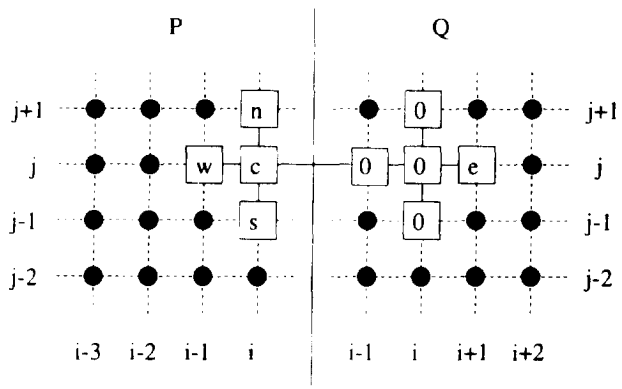

Fig. 9. The original discretization star on the artificial domain boundaries with two duplicated grid lines.

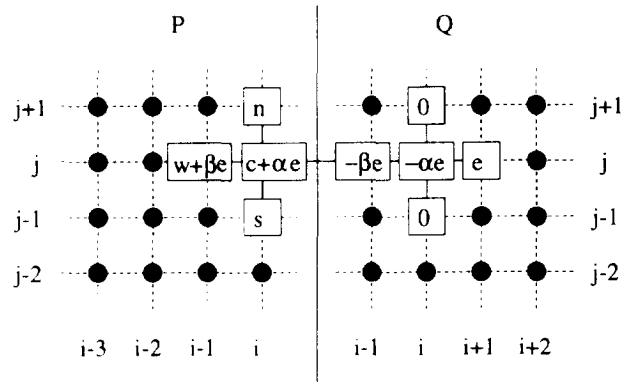

Fig. 10. The adapted discretization star on the artificial domain boundaries with two duplicated grid lines.

If we take $\beta = 0$, then we get a boundary condition similar to (2). We substitute $u^P_{i+1,j}$ (given by (12)) into (10), and this gives

$$su^P_{i,j-1} + (w + \beta e)u^P_{i-1,j} + (c + \alpha e)u^P_{i,j} - \beta e u^Q_{i-1,j} - \alpha e u^Q_{i,j} + e u^Q_{i+1,j} + n u^P_{i,j+1} = b_{i,j}. \qquad (13)$$

Eq. (13) is equivalent to the discretization star given in Fig. 10. Obviously, we can make more than one such adaptation to the discretization star at the same time (e.g. if it is also in the south boundary). The approach amounts to making corrections to the discretization star for duplicate grid points. The idea is that the "weights" in the discretization star are redistributed over entries that refer to the same grid point in the original domain (a set of duplicate grid points). Note that the redistribution of the weights in the discretization star follows directly from the choice of an artificial boundary condition. Furthermore, this permits artificial boundary conditions that change the entries also in directions non-orthogonal to the boundary (the entries for "n", point $(i, j + 1)$; "s", point $(i, j - 1)$; and their duplicates). Such a change of the discretization star reflects (physical) boundary conditions that are non-orthogonal to the boundary, e.g., tangential derivatives. Although we will not pursue this further, the approach might lead to better results; see [7].

After the construction of the generalized SEM, we compute ILUs for the local equations of each subdomain, that is for the diagonal blocks corresponding to the subdomains. Since this does not involve the off-diagonal blocks, we may avoid their computation in the generalized SEM.

3. Preconditioning with overlapping subdomains

Similar to our exposition in the previous section, we will first discuss the one-dimensional decomposition and then discuss the generalization to higher-dimensional decompositions and more general meshes.

3.1. One-dimensional decomposition

We will introduce some notation using the example in the previous section. The generalization to more strips is straightforward. Let the vector x be defined on the original domain Ω as shown in Fig. 1. Then x can be written as

$$x = (x_1, x_{2a}, x_2, x_{3a}, x_3)^T, \tag{14}$$

where the segments of x are vectors defined on the subdomains of Ω. Let $\tilde{x}$ be a vector defined on $\tilde{\Omega}$. Then $\tilde{x}$ can be written as

$$\tilde{x} = (x_1, x_{2a'}, x_{2a}, x_2, x_{3a'}, x_{3a}, x_3)^T, \tag{15}$$

where the segments of $\tilde{x}$ are vectors defined on the subdomains of $\tilde{\Omega}$. The operators $S: \Omega \mapsto \tilde{\Omega}$ and $J_\omega: \tilde{\Omega} \mapsto \Omega$ are defined by

$$S = \begin{pmatrix} I & O & O & O & O \\ O & I & O & O & O \\ O & I & O & O & O \\ O & O & I & O & O \\ O & O & O & I & O \\ O & O & O & I & O \\ O & O & O & O & I \end{pmatrix}, \tag{16}$$

$$J_\omega = \begin{pmatrix} I & O & O & O & O & O & O \\ O & \omega I & (1-\omega)I & O & O & O & O \\ O & O & O & I & O & O & O \\ O & O & O & O & \omega I & (1-\omega)I & O \\ O & O & O & O & O & O & I \end{pmatrix}, \tag{17}$$

where the matrix blocks correspond to the segments in x and $\tilde{x}$. Note that S and J_ω are determined completely by the decomposition and the size of the overlap(s) except for the choice of ω. Using S and J_ω we construct the following two operators.

- $J_\omega S: \Omega \mapsto \Omega$ is the identity operator: $J_\omega S = I$.
- $SJ_\omega: \tilde{\Omega} \mapsto \tilde{\Omega}$ is the identity operator on grid points that do not have a duplicate; it computes a weighted average over duplicate grid points.

Let $\tilde{A}$ be a generalized SEM derived from A. From the definition of S and the definition of the generalized SEM in Section 2 we can derive that

$$\tilde{A}S = SA. \tag{18}$$

We will now define the preconditioner K. In the previous section we have described the construction of the generalized SEM $\tilde{A}$ from A. From $\tilde{A}$ we can compute the (block) factors $\tilde{L}$, $\tilde{D}$ and $\tilde{U}$, which are all operators over $\tilde{\Omega}$. We will assume throughout this paper that these factors are nonsingular. Note that this can easily be verified by inspection of the diagonal elements of the matrices $\tilde{L}$, $\tilde{D}$ and $\tilde{U}$. We define operators over Ω using S and J_ω: $J_\omega \tilde{L}^{-1} S$, $J_\omega \tilde{D} S$, and $J_\omega \tilde{U}^{-1} S$. We now define the preconditioner as

$$K: \Omega \mapsto \Omega = J_\omega \tilde{U}^{-1} S J_\omega \tilde{D} S J_\omega \tilde{L}^{-1} S. \tag{19}$$

This preconditioner requires four boundary exchanges. A boundary exchange is the exchange of data that are allocated to one processor and necessary on other processors (in the matrix–vector product or preconditioner). In the case of a domain decomposition this generally amounts to exchanging data that are defined on the boundaries of the subdomains. Note that SJ_ω requires only one boundary exchange to compute the average over duplicate subdomains. We can remove one boundary exchange if we make $\tilde{D}$ such that $\tilde{D}S = SD$ for some $D: \Omega \mapsto \Omega$. This is not difficult because $\tilde{D}$ is a diagonal matrix. Hence, we only need to average the entries of $\tilde{D}$ corresponding to duplicate grid points. If $\tilde{D}S = SD$ we have

$$SJ_\omega \tilde{D}S = SJ_\omega SD = SD = \tilde{D}S,$$

so that K can be written as

$$K: \Omega \mapsto \Omega = J_\omega \tilde{U}^{-1} \tilde{D} S J_\omega \tilde{L}^{-1} S. \tag{20}$$

Although K is defined over Ω, we need intermediate vectors defined on $\tilde{\Omega}$ in the preconditioner. This means that either we have to change the data representation in order to expand the vectors, or we have to work with vectors that are not contiguous. This may be cumbersome, and it requires at least more complicated programs. Therefore, it may be attractive to work entirely on the larger subdomain $\tilde{\Omega}$ and to use simpler data structures and programs at the cost of some computational overhead. This overhead will not be that important for large problems, because we consider small overlaps only.

We can define a Krylov subspace iteration over $\tilde{\Omega}$ that is completely equivalent to the one over Ω. Let $\tilde{K}: \tilde{\Omega} \mapsto \tilde{\Omega}$ be defined as

$$\tilde{K} = SJ_\omega \tilde{U}^{-1} \tilde{D} S J_\omega \tilde{L}^{-1}. \tag{21}$$

Note that $\tilde{K}S = SK$. Let $\tilde{A}$ be the generalized SEM constructed from A, so that $\tilde{A}S = SA$. Then the operator $\tilde{K}\tilde{A}: \tilde{\Omega} \mapsto \tilde{\Omega}$ satisfies

$$(\tilde{K}\tilde{A})^n S = S(KA)^n, \quad n = 0, 1, 2, \ldots. \tag{22}$$

So there is a one-to-one correspondence between the vectors of the Krylov subspace $span\{Sr, \tilde{K}\tilde{A}Sr, (\tilde{K}\tilde{A})^2 Sr, \ldots\} = K(\tilde{K}\tilde{A}, Sr)$ and the vectors of $K(KA, r)$. We only need to specify an adapted inner product on $K(\tilde{K}\tilde{A}, Sr)$ in order to solve the same minimization problem on $K(\tilde{K}\tilde{A}, Sr)$ as on $K(KA, r)$. This inner product is defined by

$$\langle \tilde{x}, \tilde{y} \rangle_{J_\omega} = \langle J_\omega \tilde{x}, J_\omega \tilde{y} \rangle, \quad \tilde{x}, \tilde{y} \in K(\tilde{K}\tilde{A}, Sr). \tag{23}$$

Whether it is more suitable to iterate with KA or to iterate with $\tilde{K}\tilde{A}$ depends on the following consequences of using $\tilde{K}\tilde{A}$:

- The matrix–vector product and the vector update require more computational work.
- We have one boundary exchange less; this means less communication on distributed-memory computers.
- We work only in one subspace, so we have simpler data structures, and we do not have to switch between representations.
- In some cases, e.g., in the case of the 5-point discretization star, we do not need to store the preconditioner separately except for the diagonal matrix $\tilde{D}$, so we can save on storage.

Obviously, not all points are valid in all cases. If we run the program on a shared-memory computer, there is no communication. On the other hand, complex data structures may affect the performance of (multi) vector processors.

There is another point of concern. Since we project from $\tilde{\Omega}$ to Ω the preconditioner may be singular, even though the operators $\tilde{U}$, $\tilde{D}$, and $\tilde{L}$ are assumed to be nonsingular. However, for a one-dimensional decomposition where the equations for different overlaps are not coupled we can construct, with a few small modifications, a (provably) nonsingular preconditioner. For higher dimensional decompositions with overlap it will be more difficult to avoid the coupling of the equations of different overlaps.

For the example decomposition given in Fig. 1, we will construct a modified $J_\omega \tilde{L}^{-1} S$ that cannot be singular if $\tilde{L}^{-1}$ is not singular. Provably nonsingular versions of the other factors of the preconditioner can then be constructed in the same way. We will assume, without loss of generality, that $\omega = \frac{1}{2}$, and we use $J_{1/2} = J$.

The matrix $J\tilde{L}^{-1}S$ is singular if and only if

$$\exists x \neq 0,\ x \in \Omega \text{ such that } J\tilde{L}^{-1}Sx = 0. \tag{24}$$

This is equivalent to

$$\exists x \neq 0,\ x \in \Omega \text{ such that } \begin{cases} y = \tilde{L}^{-1}Sx, \\ y = (0, -y_{2a},\ y_{2a},\ 0, -y_{3a},\ y_{3a},\ 0)^{\mathrm{T}}. \end{cases} \tag{25}$$

Furthermore, $\tilde{L}^{-1}Sx = y \Leftrightarrow Sx = \tilde{L}y$. This gives for our example decomposition in Fig. 1

$$\begin{pmatrix} \tilde{L}_{1,1} & & & & & & \\ \tilde{L}_{1,2a'} & \tilde{L}_{2a',2a'} & & & & & \\ & & \tilde{L}_{2a,2a} & & & & \\ & & \tilde{L}_{2,2a} & \tilde{L}_{2,2} & & & \\ & & 0 & \tilde{L}_{3a',2} & \tilde{L}_{3a',3a'} & & \\ & & & & & \tilde{L}_{3a,3a} & \\ & & & & & \tilde{L}_{3,3a} & \tilde{L}_{3,3} \end{pmatrix} \begin{pmatrix} 0 \\ -y_{2a} \\ y_{2a} \\ 0 \\ -y_{3a} \\ y_{3a} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -\tilde{L}_{2a',2a'} y_{2a} \\ \tilde{L}_{2a,2a} y_{2a} \\ \tilde{L}_{2,2a} y_{2a} \\ -\tilde{L}_{3a',3a'} y_{3a} \\ \tilde{L}_{3a,3a} y_{3a} \\ \tilde{L}_{3,3a} y_{3a} \end{pmatrix}. \tag{26}$$

The equation $Sx = \tilde{L}y$ can hold only if

$$-\tilde{L}_{2a',2a'} y_{2a} = \tilde{L}_{2a,2a} y_{2a}, \tag{27}$$

$$-\tilde{L}_{3a',3a'} y_{3a} = \tilde{L}_{3a,3a} y_{3a}. \tag{28}$$

If this holds for some vector y such that not both $y_{2a} = 0$ and $y_{3a} = 0$, then an x satisfying (25) can be constructed and hence the operator $J\tilde{L}^{-1}S$ is singular. However, with

$$\tilde{L}_{2a',2a'} = \tilde{L}_{2a,2a}, \tag{29}$$

$$\tilde{L}_{3a',3a'} = \tilde{L}_{3a,3a}, \tag{30}$$

it follows from (27) and (28) that both $y_{2a} = 0$ and $y_{3a} = 0$, so that this condition insures nonsingularity.

In the case of an IBLU(0) preconditioner this does not pose any problems, because $\tilde{L}_{2a',2a'}$ and $\tilde{L}_{2a,2a}$, etc. have pairwise the same sparsity structure, so that we can satisfy (29) and (30) with few modifications. Moreover, since we consider only small overlaps, these matrix blocks ($\tilde{L}_{2a',2a'}$ and $\tilde{L}_{2a,2a}$, etc.) are also small, and we can give corresponding entries the same value without too much work. In the case of incomplete (block) factorizations of the form $(\tilde{L} + \tilde{D})\tilde{D}^{-1}(\tilde{D} + \tilde{U})$, where $\tilde{L}$ and $\tilde{U}$ are the strictly lower and upper triangular parts of the block diagonal of $\tilde{A}$, we only have to make the elements of the diagonal matrix $\tilde{D}$ equal for the overlap regions. Note that this is the same as we suggested for the elimination of one communication in Section 3.

3.2. Higher-dimensional decompositions and more general meshes

Also for higher dimensional decompositions and more general meshes the operators S and J_ω are determined by the decomposition into subdomains and the choice of the overlaps, except for the choice of the weight parameters ω. Note that ω can be varied for each overlap. Given S and J_ω, the algebraic definition of the preconditioner and the Krylov subspace iteration are generally applicable. Therefore, the extension to higher-dimensional decompositions and/or general meshes is straightforward.

4. Experimental results

With respect to the parallel performance, we will only consider the additional overhead introduced by the preconditioner. This has three components:

(1) The potential increase in the number of iterations.

(2) The computational overhead due to the duplication of grid points.

(3) The communication overhead due to the communication in the preconditioner.

The communication overhead in the IBLU preconditioners is relatively small because they only require communication of a processor with a few nearby processors. Also, if the number of local grid points is sufficiently large the computational overhead of one or two extra grid lines will be small. The increase in iterations is by far the most important factor, since any increase in the number of iterations gives a proportional decrease in efficiency (including the overhead in communication). We will therefore focus on the iteration count, and we will discuss the computational overhead and the resulting loss in efficiency only briefly at the end of this section.

We will begin with a discussion of the increase in the iteration count when we use only local preconditioning for an increasing number of subdomains. Then, we consider the improvement by preconditioners defined on (slightly) overlapping subdomains, and after that we consider the reduction of the iteration count when using overlapping subdomains and adapting the artificial boundary conditions, as described in Section 2.

We have done a number of experiments to see how the choice of parameters influences the convergence and to look at the potential of the described preconditioners. The experiments

with one-dimensional decompositions (all in the y-direction) were done on a single processor and we will only study the increase in the number of iterations. We give iteration counts as the number of complete cycles plus the number of iterations in the last cycle, because for GMRES(m) the cost of an iteration increases within a cycle. The experiments with two-dimensional decompositions (as described in Section 2) were done on a distributed-memory parallel computer, a 400-processor Parsytec Supercluster at the Koninklijke/Shell-Laboratorium in Amsterdam, and we will discuss the performance and parallel efficiency of this implementation in more detail at the end of this section.

Our model problem comes from the discretization of

$$-(u_{xx} + u_{yy}) + bu_x + cu_y = 0$$

on $[0, 1] \times [0, 4]$, with

$$b(x, y) = \begin{cases} 10, & \text{for } 0 \leqslant y \leqslant 1, \\ -10, & \text{for } 1 < y \leqslant 2, \\ 10, & \text{for } 2 < y \leqslant 3, \\ -10, & \text{for } 3 < y \leqslant 4, \end{cases}$$

and $c = 10$. The boundary conditions are $u = 1$ on $y = 0$, $u = 0$ on $y = 4$, and $\partial u/\partial n = 0$ for $x = 0$ and $x = 1$; see Fig. 11. We have discretized this problem over a 100×200 grid and over a 200×400 grid by the finite volume method.

We have chosen relatively small model problems since the parallelization on 400 processors then causes a rather extreme "decomposition" of the mesh, and it will be interesting to see how well the preconditioners can deal with this. Another reason is to show that the computational overhead has only a relatively small influence on the performance, even though for such large numbers of subdomains for a relatively small problem the computational overhead will be large.

We have solved the model problem for two grid sizes to see how much this influences the convergence behaviour for different decompositions. Table 1 gives the results for several decompositions without overlap. We see for the 100×200 grid that there is only little increase in the iteration count going from one subdomain to as many as 200. For the 20×20 decomposition we see a substantial increase in the iteration count. It is also interesting to see that the iteration count may be better for small decompositions (1×5) than for the sequential

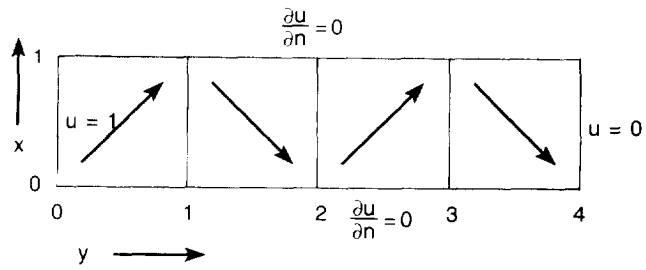

Fig. 11. The model problem.

Table 1
Iteration counts for several decompositions with strictly local preconditioning

Decomposition	100×200	200×400
1×1	$11 \times 50 + 7$	$18 \times 50 + 40$
1×5	$10 \times 50 + 35$	$21 \times 50 + 27$
1×10	$11 \times 50 + 33$	$21 \times 50 + 37$
1×20	$12 \times 50 + 10$	$20 \times 50 + 45$
10×20	$13 \times 50 + 29$	$21 \times 50 + 19$
20×20	$19 \times 50 + 40$	$24 \times 50 + 44$

implementation. For the 200×400 problem we see that the iteration count increases slowly with the number of subdomains, and even for 20×20 subdomains the iteration count is only about one-third higher than for the sequential case.

We will further restrict the discussion of the convergence for several one-dimensional decompositions with overlapping grid lines and adapted boundary conditions to the problem on the 100×200 grid. We will come back to the 200×400 problem in the discussion of the two-dimensional decompositions.

In Table 2 we have given the iteration counts for several decompositions with one or two overlapping grid lines. The number of iterations is more or less constant going from the sequential algorithm to the algorithm based on the 1×20 decomposition. Note that even the iteration count for the 1×20 decomposition is smaller than for the sequential case. Obviously we can use the same preconditioner also in a sequential algorithm.

In Table 3 we have shown the iteration counts for several decompositions with overlapping grid lines and adapted, artificial boundary conditions. We have taken the parameters α and β from the set $\{0.0, 0.1, 0.2, \ldots, 1.0\}$. There seems to be a trend that the optimal α increases if the number of subdomains increases, but this is not always so. In any case, the convergence does not seem to depend too critically on the choice of α. The results suggest that there is a rather large interval around the optimal value from which all values give comparable convergence.

It is clear that adapting the boundary conditions can improve the convergence even further. The convergence for the 1×20 decomposition with two overlapping grid lines when using both α and β is much better than for the sequential algorithm. Since we could also use an IBLU preconditioner based on some domain decomposition for the sequential algorithm, it is important that the difference between the best iteration count in Table 3, $(9 \times 50 + 15)$, and the best iteration count observed for the 1×20 decomposition, $(9 \times 50 + 47)$, is small.

Table 2
Iteration counts for several decompositions with overlapping grid lines without adapted boundary conditions

Decomposition	ovl = 0	ovl = 1	ovl = 2
1×1	$11 \times 50 + 7$	—	—
1×5	$10 \times 50 + 35$	$9 \times 50 + 15$	$10 \times 50 + 17$
1×10	$11 \times 50 + 33$	$11 \times 50 + 44$	$11 \times 50 + 11$
1×20	$12 \times 50 + 10$	$10 \times 50 + 22$	$10 \times 50 + 28$

Table 3
Iteration counts for several decompositions with overlapping grid lines and adapted boundary conditions. The given values for α identify a range of values giving good (comparable) iteration counts; the optimum is bold. For the 1×20 decomposition we also give the optimal parameters with $\beta \neq 0$

Decomposition	ovl = 0	ovl = 1		ovl = 2	
1×1	$11 \times 50 + 7$	—		—	
1×5	$10 \times 50 + 35$	$\boldsymbol{\alpha = 0.0}$	$\mathbf{9 \times 50 + 15}$	$\alpha = 0.3$	$10 \times 50 + 22$
		$\alpha = 0.1$	$9 \times 50 + 41$	$\boldsymbol{\alpha = 0.4}$	$\mathbf{9 \times 50 + 45}$
		$\alpha = 0.2$	$9 \times 50 + 45$	$\alpha = 0.5$	$9 \times 50 + 47$
		$\alpha = 0.3$	$10 \times 50 + 37$	$\alpha = 0.6$	$10 \times 50 + 7$
1×10	$11 \times 50 + 33$	$\alpha = 0.3$	$10 \times 50 + 10$	$\alpha = 0.1$	$10 \times 50 + 29$
		$\boldsymbol{\alpha = 0.4}$	$\mathbf{10 \times 50 + 1}$	$\alpha = 0.2$	$9 \times 50 + 47$
		$\alpha = 0.5$	$10 \times 50 + 5$	$\boldsymbol{\alpha = 0.3}$	$\mathbf{9 \times 50 + 41}$
		$\alpha = 0.6$	$10 \times 50 + 9$	$\alpha = 0.4$	$10 \times 50 + 10$
1×20	$12 \times 50 + 10$	$\alpha = 0.3$	$10 \times 50 + 23$	$\alpha = 0.4$	$10 \times 50 + 7$
		$\boldsymbol{\alpha = 0.4}$	$\mathbf{10 \times 50 + 8}$	$\alpha = 0.5$	$10 \times 50 + 25$
		$\boldsymbol{\alpha = 0.5}$	$\mathbf{10 \times 50 + 8}$	$\boldsymbol{\alpha = 0.6}$	$\mathbf{10 \times 50 + 1}$
		$\alpha = 0.6$	$10 \times 50 + 21$	$\alpha = 0.7$	$11 \times 50 + 40$
				$\alpha = 0.6$, $\beta = 0.1$	$9 \times 50 + 47$

We will now discuss the convergence for preconditioners derived from two-dimensional decompositions of our model problem. We will give results for the 20×20 decomposition of the 100×200 grid and the 200×400 grid using GMRES(50). The number of potential parameters is now very large. We can choose parameters ω_x and ω_y for each overlap to define the projector J_ω, and we can choose α and β different in each direction on each subdomain. Further, we can take the overlap size different in x- and y-direction. For simplicity we have taken $\omega_x = \omega_y = \frac{1}{2}$. For the experiments described here we use the same overlap size in each direction, and also for α and β (when appropriate) we use for all directions the same value. However, for completeness we mention that for some experiments not described here it was useful to have different parameters for different directions.

In Table 4 we give the iteration counts for the 20×20 decomposition. We used an overlap size of one grid line for both problems. For the 200×400 grid we also used an overlap size of three grid lines; for the smaller problem this is too expensive. For both problems an overlap size of two grid lines resulted in a very poor convergence (or none at all) irrespective of the chosen parameters. This might be due to (near) singularity; see the discussion in Section 3. For the 100×200 problem it seems that the parameter choice for two-dimensional decompositions is more sensitive than that for one-dimensional decompositions. For both grid sizes we are able to stay very close to the iteration count of the sequential algorithm, although for the 200×400 grid this requires an overlap size of three grid lines. These convergence results indicate that we can go from the sequential algorithm to an algorithm based on a decomposition into as much as 400 subdomains with only a small increase in the iteration count, provided that we can find the right parameters (which is an open problem yet).

Table 4
Iteration counts for the 100×200 and 200×400 problems with a 20×20 decomposition and adapted, artificial boundary conditions.

100×200			200×400			
Overlap	α	Iteration count	Overlap	α	β	Iteration count
1	0.1	$15\times50+10$	1	0.4	—	$22\times50+36$
1	0.2	$13\times50+41$	1	0.6	—	$22\times50+14$
1	0.4	$16\times50+25$	3	0.1	0.3	$19\times50+37$

Finally, we will discuss some performance and efficiency issues. Since we want to study the efficiency and performance of the preconditioners, we must try to separate these from the efficiency and performance of the Krylov subspace methods using these preconditioners. We will proceed as follows. We compare the measured performance of the preconditioned GMRES(m) iterations with the performance of a virtual, preconditioned GMRES(m) algorithm. The runtime of a single cycle of this virtual, preconditioned GMRES(m) algorithm is defined as the runtime of a GMRES(m) cycle with a strictly local preconditioner, as shown in Table 5, and the iteration count of this virtual, preconditioned GMRES(m) algorithm is defined as the iteration count of the sequential, preconditioned GMRES(m). This models a (virtual) preconditioned GMRES(m) with a "perfect" speed-up for the preconditioner: there is no computational overhead, no communication overhead, and the iteration count remains the same as for the sequential algorithm. The overhead of the preconditioners is measured by the relative increase in the runtime of a cycle and the relative increase (or decrease) in the number of iterations. The product of these two gives approximately the relative increase of the total solution time with respect to the solution time of the virtual, preconditioned GMRES(m) iteration. This relative increase in the solution time is the inverse of the efficiency, because the virtual GMRES(m) has an assumed efficiency of one (considering only the preconditioner). The true efficiency of the preconditioned GMRES(m) iterations can be estimated by the product of the derived efficiency and the efficiency of a single cycle of a GMRES(m) with a strictly local preconditioner (see for example [1]).

Table 5
Performance results on a 20×20 processor grid: measured runtimes for a single cycle, the number of iterations, and the measured total solution time for 100×200 and 200×400 discretizations. In version (a) we use the duplicated grid lines only for the preconditioner. In version (b) we use the duplicated grid lines in the preconditioner, the matrix vector product and the vector update.

Overlap (x/y)	100×200			200×400		
	Cycle time (s)	Iteration count	Solution time (s)	Cycle time (s)	Iteration count	Solution time (s)
0	2.21	$19\times50+40$	43.3	4.81	$24\times50+44$	119.
1(a)	2.31	$13\times50+41$	31.8	5.00	$22\times50+14$	111.
3(a)	not avail.	not avail.	not avail.	5.14	$19\times50+37$	102.
1(b)	2.46	$13\times50+41$	33.7	5.17	$22\times50+14$	114.
3(b)	not avail.	not avail.	not avail.	6.25	$19\times50+37$	123.

Table 6
Relative performance on a 20×20 processor grid: the overhead for the runtime of a cycle, for the iteration count, and for the entire runtime to compute the solution, and the efficiency of the parallel preconditioner.

Overlap (version)	100×200				200×400			
	Runtime cycle	Iteration count	Runtime solution	Efficiency (%)	Runtime cycle	Iteration count	Runtime solution	Efficiency (%)
0	1.00	1.78	1.78	56.3	1.00	1.32	1.32	75.6
1(a)	1.05	1.24	1.30	77.1	1.04	1.19	1.23	81.2
3(a)	not avail.	not avail.	not avail.	not avail.	1.07	1.05	1.12	89.1
1(b)	1.11	1.24	1.38	72.3	1.07	1.19	1.27	78.5
3(b)	not avail.	not avail.	not avail.	not avail.	1.30	1.05	1.36	73.3

For both versions of our problem we have selected the tests with the best iteration counts, because we want to look at the potential of the preconditioners. For these tests we have measured the runtime of a single GMRES(m) cycle and the runtime to compute the solution. The results are presented in Table 5. For comparison we have also included the data for the 20×20 decomposition with (strictly) local preconditioning. From Table 5 we can compute the relative increases in cycle time, iteration count, and solution time. These results are given in Table 6. We have implemented two versions of the preconditioned GMRES(m) iterations. Version (a) uses the extra unknowns from duplicated grid lines only in the preconditioner. Version (b) works entirely on the domain with duplicated grid lines.

We see that, except for one case where we used the enlarged problem for the whole iteration, the communication and computation overhead is always outweighed by the reduction in iterations. For the 200×400 case (version a) it is even worth while to have an overlap size of three grid lines instead of one due to the lower number of iterations. We can see from the results in Table 6 that very high efficiencies can be attained: 77% and 89%.

5. Conclusions

We have proposed a type of preconditioners that is based on a domain decomposition with subdomains that have only a small overlap. The construction of these preconditioners is straightforward and efficient; it does not introduce any significant parallel overhead.

Our results indicate that we may keep the increase in the iteration count quite small going from the sequential algorithm to a parallel algorithm on as much as 400 subdomains. For moderate numbers of subdomains we may keep the iteration count even constant and sometimes we may improve it compared with the sequential algorithm. Moreover, the overhead introduced by the duplication of grid lines results in only a small increase in the runtime of a single cycle. Therefore, as far as the preconditioner is concerned efficient implementations on (massively) parallel computers are possible.

Our examples indicate that the reduction in the iteration count is very important even at the cost of additional computation. This is due to the following two facts. On the one hand, on large parallel computers for Krylov subspace methods the communication cost of the inner

products dominates the performance, and this makes extra iterations very expensive. On the other hand, the runtime increase by additional computation is relatively small because of the communication cost of the inner products.

The a priori choice of good parameters is still an open problem, and future research in this direction is necessary. Some analyses have already been carried out in the case of block relaxation methods with exact solvers on the subdomains [7,9], and this has to be extended to incomplete solvers and Krylov subspace methods.

References

[1] E. de Sturler and H.A. van der Vorst, Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers, Technical Report 832, Mathematical Institute, University of Utrecht, Netherlands (1993).
[2] I. Gustafsson, A class of first order factorization methods, *BIT* 18 (1978) 142–156.
[3] J.A. Meijerink and H.A. van der Vorst, An iterative solution method for linear equations systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comp.* 31 (1977) 148–162.
[4] J.A. Meijerink and H.A. van der Vorst, Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems, *J. Comput. Phys.* (1981) 134–155.
[5] G. Radicati di Brozolo and Y. Robert, Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor, *Parallel Comput.* 11 (1989) 223–239.
[6] Y. Saad and M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* 7 (1986) 856–869.
[7] K.H. Tan and M.J.A. Borsboom, Problem-dependent optimization of flexible couplings in domain decomposition methods, with an application to advection-dominated problems, Technical Report 830, Mathematical Institute, University of Utrecht, Netherlands (1993).
[8] W.P. Tang, Schwarz splitting and template operators, Ph.D. Thesis, Stanford University, CA (1987).
[9] W.P. Tang, Generalized Schwarz splittings, *SIAM J. Sci. Statist. Comput.* 13 (1992) 573–595.