An investigation of Schwarz domain decomposition using accurate and inaccurate solution of subdomains

Report 95-18

Erik Brakkee
Kees Vuik
Piet Wesseling

TU Delft
Faculteit der Technische Wiskunde en Informatica
Faculty of Technical Mathematics and Informatics

Technische Universiteit Delft
Delft University of Technology

ISSN 0922-5641

An investigation of Schwarz Domain Decomposition using accurate and inaccurate solution of subdomains

Erik Brakkee
Kees Vuik, Piet Wesseling

March 22, 1995

Abstract

For the solution of practical complex problems in arbitrarily shaped domains, simple Schwarz domain decomposition methods with minimal overlap are used. Krylov subspace methods, such as the GMRES method, can be used to obtain significant acceleration of convergence. When accurate solution of subdomains is presupposed, this acceleration procedure can be quite efficient but the amount of time spent in solving subdomain problems may be prohibiting. To reduce computing time, inaccurate solution of subdomains is considered. This requires a different, GCR based, acceleration technique. Experiments show that computing time for a multi-domain problem can be reduced to that of single domain solution with the same total number of unknowns. For this purpose, the multiplicative domain decomposition algorithm should be used. This is an important practical observation, since this makes efficient domain decomposition available for complex problems, for which parallel implementation is not readily available, possible or feasible. The prospects for parallel implementation are also investigated.

1 Introduction

For the solution of the incompressible Navier-Stokes equations in domains of arbitrary shape, we use a block-structured finite volume method is used. References [20, 9, 27] describe the discretization in detail and reference [21] discusses the capability of the method to accurately solve a number of benchmark problems. A Schwarz type domain decomposition iteration in combination with a GMRES [23] acceleration is used to solve the resulting domain decomposition problem. Significant reductions in computing time can be obtained using the GMRES acceleration procedure, see [5] and [4].

However, since this method requires an accurate solution of subdomains, the computing time can be much larger than with single block solution for the same number of unknowns. Another problem with this method is that it is not known beforehand how accurate the subdomains must be solved. A possible solution to both problems is to solve the linear equations on the subdomains inaccurately. The effect of this is that the GMRES acceleration procedure can no longer be used because the preconditioner may vary in each iteration. Instead, a method based on GCR [14] is used. For the special case of a single domain, this method simplifies to GMRESR [24] if GMRES is used to solve the subdomains (inaccurately).

Theoretical analysis of the effect of inaccurate solution of subdomains seems intractable. Therefore we take recourse to numerical experiments. We compare some results on solving the subdomains accurately and inaccurately for a two-dimensional model problem, namely the advection-diffusion equation:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + a^1 \frac{\partial u}{\partial x} + a^2 \frac{\partial u}{\partial y} + cu = f \text{ on } \Omega = [-1,1] \times [-1,1]. \tag{1}$$

This equations is a good model of what can be expected when the method is applied to the momentum equations in Navier-Stokes solution methods. With $a^1 = a^2 = 0$ we obtain a Poisson equation which is a good model for the pressure equation occurring in the pressure-correction method. A cell-centered discretization is used, see [5] for details on domain decomposition and discretization for this equation. The results are reported in the entire range between very accurate subdomain solution with tolerance $\epsilon = 10^{-8}$ and very inaccurate solution using a blocked version of the subdomain ILU postconditioner.

In the literature, much focus is on parallel algorithms. However, parallel implementations are not immediately available and one can imagine situations where parallel execution is not (efficiently) possible. Therefore, to obtain an efficient sequential domain decomposition algorithm we pay much attention to multiplicative domain decomposition. The prospects for parallel implementation are also discussed.

2 Krylov subspace acceleration

The basic Schwarz domain decomposition iteration converges slowly because, for practical reasons, in computation fluid dynamics minimal overlap is usually used. Therefore, some acceleration procedure for this iterative procedure is needed. Krylov subspace methods are frequently used to accelerate domain decomposition methods, see for example [1] and many of the papers on *iterative substructuring* methods in [15, 7, 8, 16, 19]. This section presents both the acceleration procedure used with accurate solution of subdomains and the procedure used with inaccurate solution of subdomains.

Basically, the domain decomposition iteration is of the following form

$$u^{m+1} = (I - N^{-1}A)u^m + N^{-1}f. \tag{2}$$

with A the global discretization matrix over all domains and N^{-1} an approximation to the inverse of the block diagonal or block lower-diagonal matrix of A. The matrix N is called the block Jacobi and Gauss-Seidel matrix of A respectively. The matrix A is divided into blocks where each block corresponds with all unknowns in a single subdomain. For instance, for a decomposition into 3 blocks we have

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}. \tag{3}$$

In [5] we assumed that the subdomains were solved accurately so that N^{-1} *is the exact inverse* of the block diagonal or block lower-diagonal matrix of A, so

$$N = N_{gs} = \begin{bmatrix} A_{11} & \emptyset & \emptyset \\ A_{21} & A_{22} & \emptyset \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \text{ or } N = N_{jac} = \begin{bmatrix} A_{11} & \emptyset & \emptyset \\ \emptyset & A_{22} & \emptyset \\ \emptyset & \emptyset & A_{33} \end{bmatrix} \qquad (4)$$

with N_{gs} the Gauss-Seidel version and N_{jac} the Jacobi version of N. The Gauss-Seidel version is suitable for implementation on a single processor and leads to the sequential or multiplicative algorithm. The Jacobi version is suitable for parallelization and is called the parallel or additive version.

It can be seen that the left-hand side of (2) only depends on the values of u^m near an interface. This means that if we split the vector u into vectors w en v with w the non-interface unknowns and v the interface unknowns, see Figure 1, we have

$$(I - N^{-1}A)u = (I - N^{-1}A)Qv \qquad (5)$$

with $Q = \begin{bmatrix} 0 \\ I \end{bmatrix}$ an injection operator from v into $u = \begin{bmatrix} 0 \\ v \end{bmatrix}$. From (5), it follows that $(I - N^{-1}A)\begin{bmatrix} w \\ 0 \end{bmatrix} = 0$ so that the non-interface unknowns do not contribute to the computation of u^{m+1} in (2).

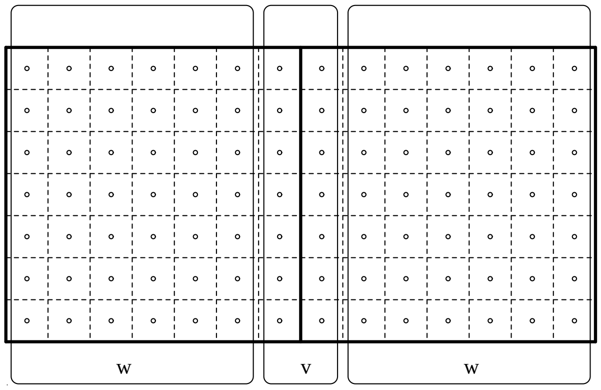

Figure 1: Interface variables v and non-interface variables w for a 5-point discretization stencil and a decomposition into two blocks

By substituting (5) into (2) and by premultiplying with Q^T we get

$$v^{m+1} = Q^T u^{m+1} = Q^T(I - N^{-1}A)Qv^m + Q^T N^{-1} f. \qquad (6)$$

Since we are interested in the stationary solution v of this iteration process we get

$$v = Q^T(I - N^{-1}A)Qv + Q^T N^{-1} f. \qquad (7)$$

which is equivalent to

$$Q^T N^{-1} A Q v = Q^T N^{-1} f. \tag{8}$$

Therefore, accurate solution of subdomains finally leads to a system concerning only the interface equations. Accelerated domain decomposition in [5] amount to solving the interface equations (8) using GMRESR [24]. The required matrix-vector product can be computed by doing one domain decomposition iteration, see [5] for details.

Inaccurate solution of subdomains implies that we no longer take the exact inverse of N but use some approximation $\tilde{N}$ to N, so

$$\tilde{N} = \tilde{N}_{gs} = \begin{bmatrix} \tilde{A}_{11} & \emptyset & \emptyset \\ A_{21} & \tilde{A}_{22} & \emptyset \\ A_{31} & A_{32} & \tilde{A}_{33} \end{bmatrix} \text{ or } \tilde{N} = \tilde{N}_{jac} = \begin{bmatrix} \tilde{A}_{11} & \emptyset & \emptyset \\ \emptyset & \tilde{A}_{22} & \emptyset \\ \emptyset & \emptyset & \tilde{A}_{33} \end{bmatrix}. \tag{9}$$

with $\tilde{N}_{gs}$ the Gauss-Seidel (sequential) version and $\tilde{N}_{jac}$ the Jacobi (parallel) version of $\tilde{N}$. In the literature, much focus is on parallel algorithms. However, parallel implementations are not immediately available and one can imagine situations where parallel execution is not possible. Suppose for instance that there is only one workstation to do computations on. Therefore, to obtain an efficient sequential domain decomposition algorithm we pay much attention the the Gauss-Seidel version of N.

The matrix vector product of $p = \tilde{N}_{gs}^{-1} v$ is computed like

$$\begin{aligned} p_1 &= \tilde{A}_{11}^{-1} v_1 \\ p_2 &= \tilde{A}_{22}^{-1}(v_2 - A_{21} v_1) \\ p_3 &= \tilde{A}_{33}^{-1}(v_3 - A_{31} v_1 - A_{32} v_2). \end{aligned} \tag{10}$$

Here $\tilde{A}_{ii}^{-1}$ represents the approximate solution in subdomain i up to a low accuracy using GMRES. A special case occurs when we take $\tilde{A}_{ii} = L_i U_i$ to be some incomplete LU factorization of A_{ii}. This paper also investigates this case for ILUD factorization, see further on.

The GMRES subdomain solution implicitly constructs a polynomial $p(A_{ii})$ of the subdomain matrix A_{ii} such that the final residual $p(A_{ii}) r_0$ is minimal in the $\| \star \|_2$ norm. Specifically, with initial guess $p_{i0} = 0$ and right-hand side v_i, we get for the final subdomain solution $p_i = p(A_{ii}) v_i$. Since the polynomial, $p(A_{ii})$ depends on both the required accuracy and the right-hand side (initial residual), the matrix $\tilde{A}_{ii}^{-1} = p(A_{ii})$ can be different for each v. Therefore, the GMRES acceleration procedure cannot be used since the preconditioner $\tilde{N}$ varies in each step.

The GCR [14] method can be easily adapted to cope with variable preconditioners. Because of its simplicity and for completeness, we derive the GCR method here. The GCR method is based on maintaining two subspaces, a subspace $S_k = < s_1, s_2, \ldots, s_k >$ for storing the search directions s_i and a subspace $V_k = < v_1, v_2, \ldots, v_k >$ with $A s_i = v_i$. In every operation of GCR the property $A s_i = v_i$ is preserved. The GCR method minimizes the residual $\|b - A x_k\|_2$ over $x_k \in S_k$. Clearly, if the $\{v_i\}_{i=1,\ldots,k}$ form an orthonormal basis, we can obtain the solution by projecting onto the space V_k. So we must find $x_k \in S_k$ such that $b - A x_k \perp v_i$ for $i = 1, \ldots k$, therefore,

$$(b - A x_k, v_i) = 0. \tag{11}$$

Since $Ax_k \in V_k$ we have

$$Ax_k = \sum_{j=1\ldots k} \lambda_j v_j \tag{12}$$

and by substituting (12) into (11) we get $\lambda_i = (b, v_i)$ so that

$$Ax_k = \sum_{i=1\ldots k} (b, v_i) v_i. \tag{13}$$

Since $As_i = v_i$, we have

$$Ax_k = \sum_{i=1\ldots k} (b, v_i) As_i = A \sum_{i=1\ldots k} (b, v_i) s_i \tag{14}$$

so that $x_k = \sum_{i=1\ldots k} (b, v_i) s_i$. This gives $x_{k+1} = x_k + (b, v_{k+1}) s_{k+1}$ and with $r_k = b - Ax_k$ we get $r_{k+1} = r_k - (b, v_{k+1}) v_{k+1}$. The GCR algorithm proceeds by choosing a new search direction s_{k+1} (preferably such that As_{k+1} approximates the residual r_k) and computes the vector $v_{k+1} = As_{k+1}$. A modified Gram-Schmidt procedure is used to make v_{k+1} orthogonal to v_i $(1 \leq i \leq k)$. The same linear combinations of vectors are applied to the space of search directions S_k to ensure that $As_i = v_i$ still holds for all i. Figure 2 shows the resulting GCR algorithm.

For the special case of the search direction $s_{k+1} = r_k$, we obtain the classical GCR algorithm, which is equivalent to GMRES [23]. For this choice of search direction, the space S_k is called the Krylov space. The difference between GCR and GMRES is that, at the benefit of allowing more general search directions, GCR requires twice the storage of GMRES and 3/2 times the number of floating point operations for orthogonalization. However GCR can be combined with truncation strategies, for instance the Jackson & Robinson [17] strategy, whereas GMRES can only be restarted. Because of this, GCR may converge faster than GMRES.

Recent developments have led to a more flexible GMRES algorithm which allows more general search directions, so called FGMRES [22]. Also, the amount of work in GCR can be reduced to approximately that of GMRES if restarted GCR is used, see [25]. The resulting algorithm is comparable to FGMRES both in memory requirements and work. The GCR method achieves the same goal as FGMRES in a more understandable way.

The present paper considers only restarted GCR to compare with subdomain solution (which uses restarted GMRES). The optimizations of [25] are not used in this paper but will certainly be considered for domain decomposition for the incompressible Navier-Stokes equations.

If we compute the search direction s_{k+1} using some GMRES iterations for solving $As_{k+1} = r_k$ we obtain the GMRESR [24] algorithm. In the present paper, we use $s_{k+1} = \tilde{N}^{-1} r_k$. If the subdomains are solved (inaccurately) using GMRES, this method reduces to GMRESR for the single domain case. If the subdomains are approximately solved using some ILU factorization we obtain a blocked version of the subdomain ILU postconditioning called IBLU, which was investigated for parallel implementation in (Incomplete Block LU) [13, 18, 10, 11]. In this paper we also investigate the sequential version of IBLU. For a single domain, this method is equivalent to GMRES with an ILU postconditioner.

```
$r_0 = b - Ax_0$; $k = 0$
**while** $\|r_k\| \geq \epsilon\|r_0\|$
    **choose a search direction** $s_{k+1}$
    **compute** $v_{k+1} = As_{k+1}$
    # modified Gram-Schmidt
    **for** $i = 1, \ldots, k$
        $\alpha = (v_{k+1}, v_i)$
        $v_{k+1} = v_{k+1} - \alpha \cdot v_i$
        # ensure $As_{k+1} = v_{k+1}$
        $s_{k+1} = s_{k+1} - \alpha \cdot s_i$
    **end for**
    $\beta = \|v_{k+1}\|_2$
    $v_{k+1} = v_{k+1}/\beta$
    $s_{k+1} = s_{k+1}/\beta$
    # end Gram-Schmidt
    # update $x$ and $r$
    $\gamma = (b, v_{k+1})$
    $x_{k+1} = x_{k+1} + \gamma s_{k+1}$
    $r_{k+1} = r_{k+1} - \gamma s_{k+1}$
    $k = k + 1$
**end while**
```

Figure 2: The GCR algorithm with general search directions without restart and with a relative stopping criterion [24]

For the special case where $\tilde{A}_{ii}$ corresponds to incomplete LU factorization, the preconditioner is constant and the GMRES acceleration procedure may also be applied. The only difference between the general method, based on GCR, and the GMRES acceleration is that GMRES requires less vector updates and less memory. Both methods of acceleration will be considered for IBLU postconditioning.

An important remark is that the stopping criterion for accurate solution differs from that for inaccurate solution. With accurate solution, the stopping criterion is based on the preconditioned residual $r = Q^T N^{-1} f - Q^T N^{-1} AQv$ of only the interface unknowns. On the other hand, with inaccurate solution, it is based on the unpreconditioned residual $r = f - Au$ of all unknowns. Therefore, a comparison between the two methods is difficult since differences in computing time can either be caused by a difference in convergence behavior or by the difference in the definition of the residual. In this paper, we assume that the difference in definition of the residual does not give different accuracies of the final solution when using the relative stopping criterion $\|r_k\| \leq \epsilon \|r_0\|$.

3 Results

This section compares accurate with inaccurate solution of subdomains. Three problems are considered. The first is a Poisson equation

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = -4 \text{ on } \Omega = [-1,1] \times [-1,1]. \tag{15}$$

with $u(x,y) = x^2 + y^2$ on the boundary. The second is a recirculating flow problem with oblique flow across the interface:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + a^1 \frac{\partial u}{\partial x} + a^2 \frac{\partial u}{\partial y} + 50u = 1 \text{ on } \Omega = [-1,1] \times [-1,1]. \tag{16}$$

with

$$\begin{cases} a^1(x,y) = 100 \cdot y \cdot (1 - x^2) \\ a^2(x,y) = -100 \cdot x \cdot (1 - y^2) + 10 \cdot (y+1). \end{cases} \tag{17}$$

The term $10 \cdot (y+1)$ makes the flow oblique across vertical and horizontal interfaces. On the left and lower sides $u(x,y) = 1$ is given and on the other sides $\partial u / \partial n = 0$ holds. This problem is known to be a difficult domain decomposition problem, see [5]. A third problem is one with simple uniform flow:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + 50\frac{\partial u}{\partial x} + 50\frac{\partial u}{\partial y} + 50u = 2 \text{ on } \Omega = [-1,1] \times [-1,1], \tag{18}$$

with the same boundary conditions as problem two. Central discretization in space is applied.

The subdomains are solved using GMRES with ILUD preconditioning and a relative stopping criterion. The subdomain solution accuracy is varied. As a special case the subdomain solution is approximated by means of the inverse of the ILUD preconditioner, see [12, 26]. This preconditioner is of the form $P = LD^{-1}U$ with L and U lower and upper triangular matrices and D a diagonal matrix with

$$\begin{cases} l_{i,j} = a_{i,j} & \text{for } j < i \\ u_{i,j} = a_{i,j} & \text{for } j > i \\ d_i = u_{i,i} = l_{i,i} & . \end{cases} \tag{19}$$

For a matrix with non-zero elements only on the positions $(i, i - n_x), (i, i-1), (i,i), (i, i+1), (i, i + n_x)$ this leads to the recursion

$$d_i = a_{i,i} - a_{i,i-w} d_{i-w}^{-1} a_{i-w,i} - a_{i,i-1} d_{i-1}^{-1} a_{i-1,i} \text{ for } i = 1,2,3,\ldots \tag{20}$$

with $a_{i,j} = 0$ if j is out of range. The ILUD preconditioner is cheap in memory because only the d_i for $1 \leq i \leq n$ need be stored and it is also cheap in work.

The multi-block problem (the outer loop) is solved up to a relative accuracy of 10^{-4}. In all experiments a Krylov space of dimension 20 is used for both GMRES and GCR multi-block acceleration and for GMRES subdomain solution. A restart after 20 iterations is used with both GMRES and GCR.

Iteration counts and computing times is given in the tables in the form *itercount/time*. The experiments were conducted on a HP9000/735 workstation.

In most of the experiments, the Gauss-Seidel (sequential) version of N is used. Only section 3.3 examines the possibilities for parallelism. Section 3.1 examines the effect of lowering the accuracy of the subdomain solution on the number of iterations and total computing time. Section 3.2 compares single block solution with multi-block solution with emphasis on the amount of additional time needed with multi-block (the multi-block penalty).

3.1 Lowering the subdomain accuracy of solution

Table 1 lists the iteration counts and computation times (itercount/time) for the three problems. The global grid consists of 80×80 grid cells and it was divided into 4×4 subdomains. The first two rows concern the algorithm for accurate solution of subdomains. The iteration counts for extremely accurate subdomain solution with tolerance $\epsilon = 10^{-8}$ and $\epsilon = 10^{-4}$ are the same. From this we conclude that the subdomain solution accuracy with tolerance 10^{-4} is accurate enough for this algorithm and test problem.

	ϵ	*Poisson*	*Recirculating flow*	*Uniform flow*
I	10^{-8}	19/38.74	12/20.74	7/7.98
	10^{-4}	19/22.98	12/13.52	7/5.60
II	10^{-4}	14/16.91	9/9.91	5/3.74
	10^{-3}	14/14.00	9/8.68	5/3.30
	10^{-2}	15/12.27	9/7.38	5/2.82
	10^{-1}	17/9.90	10/6.47	6/2.93
III	IBLUD post + GMRES	33/4.43	46/6.24	16/2.13
	IBLUD post + GCR	33/5.19	46/7.49	16/2.50

Table 1: Lowering the accuracy of subdomain solution. I is the algorithm for accurate solution of subdomains, II for inaccurate solution using GMRES, III is for IBLUD

The other rows are for the GCR based algorithm. Note that in the special case of IBLUD postconditioning we have also listed the more efficient GMRES acceleration. As the subdomain solution accuracy is lowered the number of iterations increases only slightly. Because of this the computing time drops significantly for lower subdomain accuracies. Only for the special case of IBLUD postconditioning, the number of iterations is significantly higher. This rise in number of iterations does however not outweigh the reduction in work by computing only $U^{-1}L^{-1} \times$ vector instead of doing GMRES. The computing time with IBLUD postconditioning is by far the lowest. Note that the amount of additional work in GCR acceleration compared to GMRES acceleration can be significant. The only difference with GMRES is that GCR needs some additional vector updates and requires some more memory. A more efficient implementation, see [25], will certainly be considered for the incompressible Navier-Stokes equations. Mathematically, the algorithms are the same for IBLUD postconditioning.

An important observation is that the GCR algorithm for inaccurate solution of subdomains requires fewer iterations than the algorithm for accurate subdomain solution using the same subdomain solution accuracy. To show that this difference is not caused by the different definition of the residual, we compare the computed solutions u^h of the Poisson equation with the exact solution $u(x, y) = x^2 + y^2$.

Table 2 shows the maximum norms and 2-norms of the difference with the exact solution. The 2-norm is defined as $\|x\|_2 = \sqrt{\sum_{i=1,\dots,n} x_i^2/n}$. Clearly, the solutions obtained with both algorithms have approximately the same accuracy. Only for large subdomain accuracy (giving very large computing time) the algorithm for accurate solution of subdomains gives a more accurate solution. This verifies our earlier claim that the solution obtained with GMRES acceleration used with accurate solution of subdomains is sensitive to the subdomain solution accuracy. This sensitivity is not present with the GCR based acceleration procedure used with inaccurate solution of subdomains.

	ϵ	$\|u^h - u\|_\infty$	$\|u^h - u\|_2$
I	10^{-8}	0.00001	0.0002
	10^{-4}	0.0020	0.0030
II	10^{-4}	0.0032	0.0016
	10^{-3}	0.0033	0.0016
	10^{-2}	0.0029	0.0014
	10^{-1}	0.0038	0.0020
III	IBLUD	0.0010	0.0022

Table 2: Accuracy of the solution to the Poisson problem using (I) the algorithm for accurate solution of subdomains, (II) the algorithm for inaccurate solution of subdomains using GMRES and (III) for IBLUD

We see that with inaccurate solution of subdomains by ILUD (the IBLUD postconditioning), we can reduce computing time is reduced by a factor $2 - 6$ compared to accurate solution of subdomains. In [6], a comparison between accurate and inaccurate solution was made based on the number of iterations only. This led to the conclusion that the simple accelerated Schwarz algorithm using accurate subdomain solution was a competitive method compared to single-domain ILU preconditioned GMRES. The basis of analysis in the present paper is that we do less work per iteration and therefore we allow some more iterations. Therefore, although the number of iterations is machine and implementation independent it should not be used as a basis for comparison. Computing time is more suitable to compare algorithms but may give different results depending on the implementation and machine architecture.

The most impressive reductions of computing time are obtained for the Poisson equation, which is also the most expensive part of the multi-block Navier-Stokes problem, see [4]. For the difficult recirculating flow problem we obtain a reduction of a factor of 2. The simpler uniform flow problem shows a reduction of a factor of 4.

3.2 Single domain versus multi domain

One of the main reasons for investigating inaccurate solution of subdomains was to reduce the excessive computing times observed in the multi-block incompressible Navier-Stokes solver [4], and to bring them closer to single block block solution. This also gives better prospects for parallel computing.

It is therefore interesting to compare single block solution times with multi-block solution times.

Table 3 lists the number of iterations and computing times for single block solution of a Poisson equation on an 80×80 grid. The results are given for GMRES subdomain solution using both ILUD preconditioning and postconditioning. The postconditioning is a special case of the IBLUD postconditioner for a single domain. The preconditioner is implemented efficiently on the level of the subdomain solver. The first row of the table thus represents an efficient ILUD preconditioner and the second row an ILUD postconditioner with some multi-block overhead (copying of vectors etc.)

	Poisson	*Recirculating flow*	*Uniform flow*
ILUD preconditioner	39/3.74	52/5.04	17/1.57
ILUD postconditioner	33/3.84	39/4.69	16/1.85

Table 3: Single block solution using GMRES with ILUD pre- and postconditioning

Table 4 shows a comparison of single block solution and multi-block solution for the Poisson equation for different decompositions of the domain. The decomposition of the domain is indicated as $b_x \times b_y - n_x \times n_y$ where $b_x \times b_y$ indicates the decomposition into blocks and $n_x \times n_y$ the size of each subdomain in grid cells.

We see that the number of iterations with accurate solution of subdomains approximately doubles as the same grid is divided up from 2×2 subdomains into $8 \times 8 = 64$ subdomains. As the subdomains are solved less accurate, this increase in the number of iterations is only slightly less. The IBLU postconditioner performs well. Although the number of iterations for the same decomposition of the domain is approximately twice as much as with GMRES solution of subdomains for the same decomposition of the domain, the computing time is still significantly smaller.

Note that, despite an increase in the number of iterations, the computing time is almost constant if subdomains are solved accurately ($\epsilon = 10^{-4}$, $\epsilon = 10^{-8}$). The reason is that subdomain solution becomes more efficient for smaller problems. This can be seen as follows. The amount of work required to solve a subdomain problem depends on the number of unknowns m:

$$W(m) = c \cdot m^{\alpha} \tag{21}$$

with $c > 0$ and $\alpha > 1$. If the global domain consists of N unknowns and p subdomains are used, we have $m = N/p$ and the amount of work for one domain decomposition iteration becomes

$$W = p \cdot W(N/p) = cN^{\alpha} \cdot \frac{1}{p^{\alpha} - 1}. \qquad (22)$$

Clearly W decreases as the number of subdomains p rises. Since (21) is only valid for m large enough, the result (22) is not valid for large p and computing time will start to increase again. p.

We see that the number of iterations for IBLUD does not increase significantly as the number of subdomains is increased. This property is probably caused by the A_{ji} $(j > i)$ terms in the preconditioner $\tilde{N}$, see Eq. (4), which were used in the preconditioner. The number of iterations is almost the same as with single-block solution. Also, note that for IBLUD postconditioning, the more general GCR acceleration gives an overhead of about $10 - 20\%$ with respect to the more efficient GMRES acceleration.

		decomposition		
	ϵ	$2 \times 2 - 40 \times 40$	$4 \times 4 - 20 \times 20$	$8 \times 8 - 10 \times 10$
I	10^{-8}	14/64.04	19/38.74	27/42.69
	10^{-4}	14/30.01	19/22.98	27/30.31
II	10^{-4}	11/22.73	14/16.91	20/22.24
	10^{-3}	11/18.73	14/14.00	21/21.36
	10^{-2}	11/12.56	15/12.27	21/19.18
	10^{-1}	13/8.64	17/9.90	24/19.44
III	IBLUD post + GMRES	33/3.85	33/4.43	35/5.82
	IBLUD post + GCR	33/4.73	33/5.19	35/7.00

Table 4: Subdivision of the same grid into subdomains for the Poisson equation. I is the algorithm for accurate solution of subdomains, II is inaccurate solution of subdomains using GMRES and III for IBLUD

Table 5 lists the results for the recirculating and uniform flow problems. Again there is an increase in the number of iterations as more subdomains are used. With inaccurate solution of subdomains using GMRES there is an increase of a factor 2 to 3 in the number of iterations for the recirculating flow problem as the grid is divided into more subdomains. For the uniform flow problem, this increase is only moderate. With IBLUD postconditioning, the number of iterations increases only moderately with the number of subdomains. The number of iterations is approximately equal to that of single domain solution with ILUD postconditioning, see Table 3, especially for the uniform flow problem. This means that, excluding overhead by the implementation, the computing time should be almost constant as the number of blocks is increased.

In all three problems, we see that with IBLUD postconditioning, the GCR based algorithm requires more computing time than IBLUD postconditioning combined with GMRES. This is what can be expected because GCR requires more vector updates than GMRES. Also, as the

Recirculating flow				
		decomposition		
	ϵ	$2 \times 2 - 40 \times 40$	$4 \times 4 - 20 \times 20$	$8 \times 8 - 10 \times 10$
I	10^{-8}	8/24.80	12/20.74	20/30.07
	10^{-4}	8/14.38	12/13.52	20/23.61
II	10^{-4}	6/9.99	9/9.91	16/18.16
	10^{-3}	6/8.45	9/8.68	16/17.22
	10^{-2}	7/7.64	9/7.38	16/15.95
	10^{-1}	9/6.21	10/6.47	18/16.42
III	IBLUD post + GMRES	43/5.20	46/6.24	49/8.31
	IBLUD post + GCR	43/6.40	46/7.49	49/9.67

Uniform flow				
		decomposition		
	ϵ	$2 \times 2 - 40 \times 40$	$4 \times 4 - 20 \times 20$	$8 \times 8 - 10 \times 10$
I	10^{-8}	6/8.64	7/7.98	8/10.16
	10^{-4}	6/5.37	7/5.60	8/8.21
II	10^{-4}	4/3.37	5/3.74	7/6.63
	10^{-3}	4/2.97	5/3.30	7/6.17
	10^{-2}	4/2.40	5/2.82	7/5.84
	10^{-1}	5/2.02	6/2.93	8/5.99
III	IBLUD post + GMRES	16/1.89	16/2.13	16/2.74
	IBLUD post + GCR	16/2.29	16/2.50	16/3.05

Table 5: Subdivision of the same grid into subdomains for the recirculating and uniform flow problems. I is the algorithm for accurate solution of subdomains, II is inaccurate solution of subdomains using GMRES and III is for IBLUD

number of subdomains is enlarged, the number of iterations with IBLUD does not increase significantly. However, the computing time still does. This increase in computing time is caused by overhead of the multi-block algorithm. Table 6 lists the overhead for solving the Poisson equation using IBLUD postconditioning with GMRES acceleration. The computing time is divided into several categories

- **copy:** for the copying of vector
- **bc:** for the computation of the *internal boundary conditions*, that is the terms $A_{ji}v_i$ for $j > i$, see formula (10)
- **prec:** for evaluating the subdomain ILUD preconditioner (computation $v = U^{-1}L^{-1}w$) (also present with single-block solution)
- **matvec:** for computation of the subdomain matrix vector product $A_{ii}v$.

Categories **copy** and **bc** are typical multi-block overhead and **prec** and **matvec** are also present in the single-block case. The amount of time spent in latter two categories should be approximately the same for both multi-block and single-block solution.

	Decomposition			
category	1×1	2×2	4×4	8×8
total time	3.81	3.85	4.43	5.82
copy	0.33	0.39	0.71	1.14
bc	0.16	0.23	0.38	0.88
prec	0.81	0.75	0.68	0.66
matvec	0.58	0.57	0.61	0.64

Table 6: Overhead in the multi-block algorithm for inaccurate solution of subdomains

The overhead in all categories increases significantly as the grid is subdivided into more domains. The overhead of copying vectors cannot be easily reduced if we want to retain a black box implementation of the subdomain solution algorithm. The overhead involved in the computation of internal boundary conditions can also not be avoided. As Table 6 shows, the amount of time spent in evaluating the preconditioner and computing the matrix-vector product is almost constant as more blocks are used. This is correct, since the number of iterations stays approximately the same and the global discretization matrix A is of the same size independent of the number of blocks.

3.3 Prospects for parallel implementation

In this section, we take a brief look at the possibilities for parallel implementation. Table 7 shows a comparison between the block Gauss-Seidel and block Jacobi versions of the postconditioner $\tilde{N}$. We see that the penalty of going from the sequential to the parallel algorithm is approximately a factor 2 if subdomains are solved inaccurately using GMRES. With IBLUD postconditioning on the other hand, this factor is much less than 2. Also, computing time on a single machine is minimized if IBLUD postconditioning is used. This means that we expect good results from parallelization of the IBLUD postconditioning.

In [2, 3] parallelization of domain decomposition for the incompressible Navier-Stokes equations using accurate solution of subdomains was investigated. The method performed well on a cluster of workstations. The reason was that with accurate solution of subdomains the parallelization is rather coarse grained. Furthermore, the reduction to a system of interface equations (8) made a very simple implementation possible.

The results of the present study show, however, that with the algorithms discussed in this paper, the domain decomposition method on a single machine will probably beat the current parallel implementation [2, 3] in the near future. Parallelization of the algorithms of this report is also possible but involves a parallelization of the GCR method itself. As Table 7 shows, the number of iterations increases only slightly as the subdomain solution accuracy is lowered to $\epsilon = 10^{-1}$. Therefore, the communication overhead remains almost constant while

	Poisson		*Recirculating flow*		*Uniform flow*	
ϵ	*seq*	*par*	*seq*	*par*	*seq*	*par*
10^{-4}	14/16.91	31/37.39	9/9.91	16/18.00	5/3.74	12/8.96
10^{-3}	14/14.00	32/31.95	9/8.68	16/15.99	5/3.30	12/8.00
10^{-2}	15/12.27	33/27.17	9/7.38	16/13.47	5/2.82	12/7.04
10^{-1}	17/9.90	34/19.94	10/6.47	17/11.34	6/2.93	12/5.97
IBLUD+GMRES	33/4.43	44/5.85	46/6.24	53/6.91	16/2.13	21/2.81

Table 7: Comparison between the sequential (Gauss-Seidel) and parallel (Jacobi) version of the postconditioner $\tilde{N}$ for a decomposition into 4×4 blocks.

the amount of work decreases, which gives a lower computing time (about a factor 2). The most efficient algorithm on a single machine will probably not perform well on the cluster because then the number of iterations is much larger which increases the communication overhead significantly.

4 Conclusions

The main reason for this model study was to reduce the computing times observed with the domain decomposition algorithm for the Navier-Stokes equations, see [4]. This report shows, that it is possible to reduce computing time of the domain decomposition method by a factor 2 to 6 depending on the problem. Specifically, we can reduce computing time to almost that of the single block solution.

As the subdomain solution accuracy is lowered, the number of iterations required to solve the problem remains constant or shows only a small increase, which leads to a reduction in total computing time in our experiments. The most impressive reduction is obtained if the subdomain solution is approximated by an incomplete LU factorization. In this case, we can reduce computing time to almost that of the single block solution. This is an important practical observation, since this makes efficient domain decomposition available for complex problems, for which parallel implementation is not readily available, possible or feasible.

The experiments show that with the Gauss-Seidel version of the IBLUD preconditioner, the number of iterations required for multi-block problems is approximately if not exactly the same as that for single-block solution with ILUD postconditioning. The only reason why there is an increase in computing time when more subdomains are used is overhead by the implementation. This overhead is only noticeable for a large number of relatively small subdomains.

The experiments show that with the algorithm for accurate solution of of subdomains, the solution is sensitive to the subdomain solution accuracy, see Table 2. The GCR based algorithm described in this paper is completely insensitive to the subdomain solution accuracy.

Inaccurate solution of subdomains is also interesting for parallel implementation. With parallel implementation, however, the IBLUD postconditioning is preferable. This is because the IBLUD postconditioned algorithm is more efficient than the algorithm with GMRES solution of subdomains. Also, the IBLUD postconditioner shows the smallest increase in iteration count when going from the sequential algorithm to the parallel algorithm. When communication is a real bottleneck, the algorithm using a small subdomain solution accuracy of 10^{-1} can be used instead of IBLUD.

The new methods investigated in this report will be implemented in the near future for the incompressible Navier-Stokes equations. The efficient implementation of restarted GCR discussed in [25] will be used for that purpose. The current parallel implementation for the Navier-Stokes equations, which uses accurate solution of subdomains, will probably be surpassed by these new methods on a single machine.

References

[1] P.E. Bjørstad and O.B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM Journal of Numerical Analysis*, 23:1097–1120, 1986.

[2] E. Brakkee and A. Segal. A parallel domain decomposition algorithm for the incompressible Navier-Stokes equations. In L. Dekker, W. Smit, and J.C. Zuidervaart, editors, *Massively Parallel Processing Applications and Development*, pages 743–752, Elsevier, Amsterdam, 1994.

[3] E. Brakkee, A. Segal, and C.G.M. Kassels. A parallel domain decomposition algorithm for the incompressible Navier-Stokes equations. Submitted to Journal of Simulation Practice and Theory.

[4] Erik Brakkee and Piet Wesseling. Schwarz domain decomposition for the incompresssible Navier-Stokes equations in general coordinates. Report 94-84, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1994. Available from anonymous ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1994/DUT-TWI-94-84.ps.gz.

[5] Erik Brakkee and Peter Wilders. A domain decomposition method for the advection-diffusion equation. Report 94-08, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1994. Available from anonymous ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1994/DUT-TWI-94-08.ps.gz.

[6] Xiao-Chuan Cai, William D. Gropp, and David E. Keyes. A comparison of some domain decomposition and ilu preconditioned iterative methods for nonsymmetric elliptic problems. *Numerical Linear Algebra with Applications*, 1, 1994.

[7] Tony F. Chan, Roland Glowinski, Jacques Périaux, and Olof B. Widlund, editors. *Proc. of the Second International Symposium on Domain Decomposition methods*, SIAM, Philadelphia, 1989.

[8] Tony F. Chan, Roland Glowinski, Jacques Periaux, and Olof B. Widlund, editors. *Proc. of the Third International Symposium on Domain Decomposition methods for Partial Differential Equations*, SIAM, Philadelphia, 1990.

[9] C.W.Oosterlee and P.Wesseling. A multigrid method for an invariant formulation of the incompressible navier-stokes equations in general coordinates. *Comm. Applied Num. Methods*, 8:721–725, 1992.

[10] E. de Sturler and D.R. Fokkema. Nested krylov methods and preserving the orthogonality. In N. Duane Melson, T.A. Manteuffel, and S.F. McCormick, editors, *Sixth Copper Mountain Conference on Multigrid Methods*, Nasa Conference Publication 3224, Part I, pages 111–125, Nasa Langley Research Center, Hampton, VA, USA, 1993.

[11] Eric de Sturler. IBLU preconditioners for massively parallel computers. In D. E. Keyes and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering (Proceedings of the Seventh International Conference on Domain Decomposition, October 27–30, 1993, The Pennsylvania State University).* American Mathematical Society. Providence, USA, 1995.

[12] H.A. Van der Vorst. Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from pde-problems. *J. Comput. Phys.*, 44:1–19, 1981.

[13] Radicati di Brozolo and Y. Robert. Parallel conjugate gradient like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor. *Parallel Computing*, 11:223–239, 1989.

[14] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal of Numerical Analysis*, 20:345–357, 1983.

[15] R. Glowinski, G.H. Golub, G.A. Meurant, and J. Périaux, editors. *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988.

[16] Roland Glowinski, Yuri A. Kuznetsov, Gérard Meurant, Jacques Périaux, and Olof B. Widlund, editors. *Proc. of the Fourth International Symposium on Domain Decomposition methods for Partial Differential Equations*, SIAM, Philadelphia, 1991.

[17] C.P. Jackson and P.C. Robinson. A numerical study of various algorithms related to the preconditioned conjugate gradient method. *Internal Journal for Numerical Methods in Engineering*, 21:1315–1338, 1985.

[18] Wang Jin-xiau. The parallel block preconditioned conjugate gradient algorithms. In David E. Keyes, Tony F. Chan, Gérard Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors, *Proc. of the Fifth International Symposium on Domain Decomposition methods for Partial Differential Equations*, pages 339–345, SIAM, Philadelphia, 1992.

[19] David E. Keyes, Tony F. Chan, Gérard Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors. *Proc. of the Fifth International Symposium on Domain Decomposition methods for Partial Differential Equations*, SIAM, Philadelphia, 1992.

[20] A.E. Mynett, P. Wesseling, A. Segal, and C.G.M. Kassels. The ISNaS incompressible Navier-Stokes solver: invariant discretization. *Applied Scientific Research*, 48:175–191, 1991.

[21] C.W. Oosterlee, P. Wesseling, A. Segal, and E. Brakkee. Benchmark solutions for the incompressible Navier-Stokes equations in general coordinates on staggered grids. *International Journal for Numerical Methods in Fluids*, 17:301–321, 1993.

[22] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comp.*, 14:461–469, 1993.

[23] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–869, 1986.

[24] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4), 1994.

[25] C. Vuik. New insights in GMRES-like methods with variable preconditioners. Reports of the Faculty of Technical Mathematics and Informatics 93–10, Delft University of Technology, Delft, 1993.

[26] C. Vuik. Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *International Journal for Numerical Methods in Fluids*, 16:507–523, 1993.

[27] P. Wesseling, A. Segal, J. van Kan, C.W. Oosterlee, and C.G.M. Kassels. Invariant discretization of the incompressible Navier-Stokes equations in general coordinates on staggered grids. *Comput. FLuids Dyn. J.*, 1:27–33, 1992.