INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **26**: 1217–1237 (1998)

DOMAIN DECOMPOSITION FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS: SOLVING SUBDOMAIN PROBLEMS ACCURATELY AND INACCURATELY

E. BRAKKEE[a,*], C. VUIK[b,1] AND P. WESSELING[b,2]

[a] *GMD/SCAI Forschungszentrum Informationstechnik, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany*
[b] *Applied Analysis Group, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, Netherlands*

SUMMARY

For the solution of practical flow problems in arbitrarily shaped domains, simple Schwarz domain decomposition methods with minimal overlap are quite efficient, provided Krylov subspace methods, e.g. the GMRES method, are used to accelerate convergence. With an accurate subdomain solution, the amount of time spent solving these problems may be quite large. To reduce computing time, an inaccurate solution of subdomain problems is considered, which requires a GCR-based acceleration technique. Much emphasis is put on the multiplicative domain decomposition algorithm since we also want an algorithm which is fast on a single processor. Nevertheless, the prospects for parallel implementation are also investigated. © 1998 John Wiley & Sons, Ltd.

KEY WORDS: domain decomposition; GCR; Krylov–Schwarz; incompressible Navier–Stokes; boundary-fitted coordinates; finite volume

1. INTRODUCTION

For the solution of the incompressible Navier–Stokes equations in domains of arbitrary shape, we use a finite volume method on structured boundary fitted grids. References [1–6] describe the discretization in detail and [7,5] discuss the capability of the method to accurately solve a number of laminar and turbulent flows. A Schwarz-type domain decomposition iteration [8] in combination with GMRES [9] acceleration is used. In [10,11], significant reductions in computing time can be obtained using the GMRES acceleration procedure.

However, since the method described in [12] requires accurate solutions of subdomain problems, it appears that the computing time can be much larger than that using single-block solutions for the same number of unknowns. Also, it is not known beforehand how accurately the subdomain problems must be solved. The required subdomain solution accuracy may be quite high, especially when grid cells are stretched near block interfaces, and a too low accuracy generally gives wrong results. A possible solution to both problems is to abandon the assumption of exact subdomain solutions and to allow (very) inaccurate subdomain solutions. Since the preconditioner may now vary in each iteration, GMRES acceleration may no longer be applied. Instead, a method based on GCR [13] is used.

* Correspondence to: GMD/SCAI Forschungszentrum Informationstechnik, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany. E-mail: erik.brakkee@gmd.de
[1] E-mail: c.vuik@math.tudelft.nl
[2] E-mail: p.wesseling@math.tudelft.nl

CCC 0271–2091/98/101217–21$17.50

Received 3 July 1995
Revised 15 June 1997

Considerable reductions in computing time can be obtained in this way for a two-dimensional advection–diffusion equation, see [14]. Approximate subdomain solutions using a single iteration with ILUD factorization reduced multiblock computing time to almost that of single-block computing time. This encouraged us to extend this approach to the Navier–Stokes equations. Theoretical results and numerical experiments are presented to illustrate the effect of inaccurate solutions of subdomain problems for the incompressible Navier–Stokes equations.

Parallel computing is of increasing importance. This makes it important to compare the parallel (additive) domain decomposition algorithms with the best multiplicative algorithms; which are known to be faster than additive algorithms. Thus, it is also important to give attention to multiplicative algorithms.

2. DISCRETIZATION

For the spatial discretization, a finite volume method employing a staggered grid and central discretization was used. The normal velocity components are located at the center of the faces of the cells and the pressure unknowns are located in the center of the cells, see Figure 1.

For the time discretization, the implicit Euler method is used. With V^n and P^n representing the algebraic vectors of velocity and pressure unknowns at time t^n, respectively, we get

$$\frac{V^{n+1} - V^n}{\Delta t} = M(V^n, P^n)V^{n+1} - GP^{n+1}, \tag{1}$$

$$DV^{n+1} = 0, \tag{2}$$

where (1) represents the momentum equations and (2) represents the incompressibility condition $\operatorname{div} u = 0$. The matrix M represents the linearized spatial discretization of the Navier–Stokes equations around time level n, G is the discretized gradient operator and D is the discretized divergence operator on a staggered grid. Figure 2 shows the discretization stencils.

To solve (1) and (2) with the pressure correction method [15–17], these equations are approximated by the following variant of the Peaceman–Rachford scheme

$$\frac{V^* - V^n}{\Delta t} = M(V^n, P^n)V^* - GP^n \tag{3}$$

and

$$\frac{V^{n+1} - V^n}{\Delta t} = M(V^n, P^n)V^* - GP^{n+1}, \tag{4a}$$

$$DV^{n+1} = 0. \tag{4b}$$

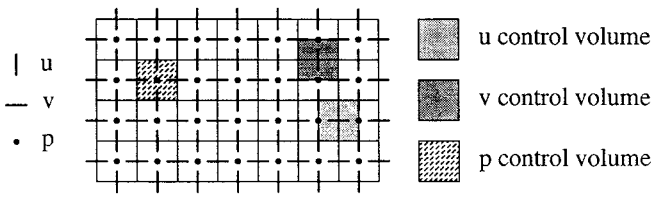

Figure 1. Arrangement of unknowns in a staggered grid.

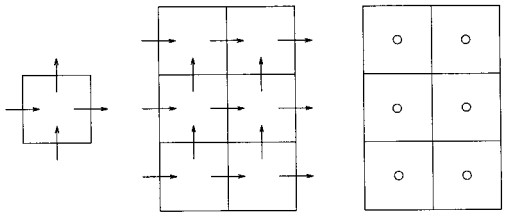

Figure 2. Discretization stencils: discretization of divergence operator D; x-component of momentum equations, M_1; and x-component of gradient matrix, G_1.

Subtraction of (3) from (4a) gives

$$\frac{V^{n+1} - V^*}{\Delta t} = -G(P^{n+1} - P^n). \tag{5}$$

Substitution of (4b) into (5) results in

$$DG\,\Delta P = \frac{DV^*}{\Delta t}, \tag{6}$$

where $\Delta P = P^{n+1} - P^n$. This is the pressure correction equation. Because the boundary conditions are already implied in Equations (1) and (2), with corresponding boundary modifications of the operators D and G, no boundary conditions are required for the pressure correction equation. This is fortunate, since such conditions are not available.

After the pressure correction ΔP has been computed from Equation (6), it is substituted into Equation (5), which leads to

$$V^{n+1} = V^* - \Delta t G\,\Delta P. \tag{7}$$

In summary, the pressure correction method consists of three steps: (i) computation of V^* from Equation (3), (ii) computation of ΔP (and P^{n+1}) from Equation (6), and (iii) computation of V^{n+1} from Equation (7).

3. DOMAIN DECOMPOSITION

A basic approach to handle flow problems in geometrically complicated domains while still maintaining structured grids is called domain decomposition. The domains can have large or small overlaps. For practical reasons, domain decomposition with minimal overlap prevails in industry, and is used here so that the analysis and numerical experiments shed light on and suggest easy-to-apply improvements to approaches common in engineering applications. Given this type of geometrically inspired domain decomposition, fast iterative convergence is desired. If fast convergence is the sole purpose, a large overlap could be used and a global coarse grid correction could be employed.

It is known from theory [18] and experiment [19] that both a constant overlap in physical space and a multilevel acceleration are required to keep the iteration count constant as the mesh is refined. Examples of constant overlap in *physical* space can be found in References [20–22]. However, as observed in [19,23,24], algorithms with small overlaps can be quite

effective, even for large and ill-conditioned problems. Although the number of GMRES iterations is typically higher with a small overlap, this is compensated for by the fact that there is less duplication of work in the overlap regions. Methods using a small overlap are also much easier to implement for practical complicated problems, and tend to dominate engineering applications.

A coarse grid correction [25–28] can be quite effective for improving the convergence of domain decomposition. However, in large codes used for engineering computations, coarse grid correction is difficult to implement and will not be considered here. The aim of this paper is to optimize the efficiency of domain decomposition with minimal overlap.

3.1. General description

The pressure correction algorithm, (3)–(7), is used for Navier–Stokes solution on the global domain. The Equations (3) and (6) are solved using domain decomposition. In this paper, we assume that the subdomains intersect regularly, i.e. the grid lines are continuous across block-interfaces. For the description of the domain decomposition algorithm, we start from a discretization of the momentum and pressure equations on the global grid.

The discretization matrix of the linearized momentum equations on the global domain is

$$S(V^n, P^n) = \frac{I}{\Delta t} - M(V^n, P^n), \tag{8}$$

and the discretization matrix of the pressure equations on the global domain is

$$T = DG, \tag{9}$$

with D being the global divergence and G the global gradient operator. Equations (8) and (9) are solved using domain decomposition. The correction of V^* is independently carried out in all blocks.

Both the pressure equations (6) and the momentum equations (3) can be written as

$$Av = f, \tag{10}$$

with either $A = S$ from (8) and $v = V$, for the momentum equations, or $A = T$ from (9) and $v = \Delta p$, for the pressure equations. If we decompose A into blocks such that each block corresponds to all unknowns in a single subdomain, with a small modification for the momentum equations (see further on), then for two subdomains

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{11}$$

where A_{11} and A_{22} represent the subdomain discretization matrices and A_{12} and A_{21} represent the coupling between subdomains. Unaccelerated domain decomposition iteration for Equation (10) is of the following form

$$v^{m+1} = (I - N^{-1}A)v^m + N^{-1}f, \tag{12}$$

with N^{-1} as an approximation to the inverse of the block diagonal or block lower-triangular matrix of A. The matrix N is called the block Jacobi or Gauss–Seidel matrix of A, depending on the method used.

Block Gauss–Seidel and Jacobi iterations are algebraic generalizations of the Schwarz domain decomposition algorithm [8,29]. Similar to Schwarz domain decomposition in each iteration, subdomain problems are solved using values from neighboring blocks. For instance, Equation (12) solved for domain 1 becomes

$$v_1^{m+1} = A_{11}^{-1}(f - A_{12}v_2^m), \tag{13}$$

where A_{11}^{-1} represents the subdomain solution and v_2^m represents the values from the neighboring block. The subdomain problems $A_{11}x_1 = \ldots$ and $A_{22}x_2 = \ldots$ are solved using GMRES [9] with appropriate preconditioners [30]. GMRES may be used to solve subdomain problems as well as to accelerate domain decomposition. We cannot apply the above described block Gauss–Seidel and Jacobi algorithms directly to the momentum matrix S because the normal velocity components at the block interfaces belong to two blocks. First we augment the matrix S in the following way. For the sake of argument, consider a decomposition into two blocks as in Figure 3.

Suppose that the velocity unknowns are divided into three sets as in Figure 3.

- The first set consists of velocities belonging to block 1, excluding the normal velocities at the block interfaces.
- The second set consists of the normal velocities at the interface.
- The third set consists of the velocities belonging to block 2, excluding the normal velocities at the block interfaces.

With respect to these three sets of unknowns, the matrix $S(V^n, P^n)$ has the block form

$$S(V^n, P^n) = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix}. \tag{14}$$

The system of equations $S(V^n, P^n)V^* = f$ can be transformed to the equivalent system

$$\bar{S}(V^n, P^n)\bar{V}^* = \begin{bmatrix} S_{11} & S_{12} & 0 & S_{13} \\ S_{21} & S_{22} & 0 & S_{23} \\ S_{21} & 0 & S_{22} & S_{23} \\ S_{31} & 0 & S_{32} & S_{33} \end{bmatrix} \cdot \begin{bmatrix} \bar{V}_1^* \\ \bar{V}_2^* \\ \bar{V}'^*_2 \\ \bar{V}_3^* \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_2 \\ f_3 \end{bmatrix}. \tag{15}$$

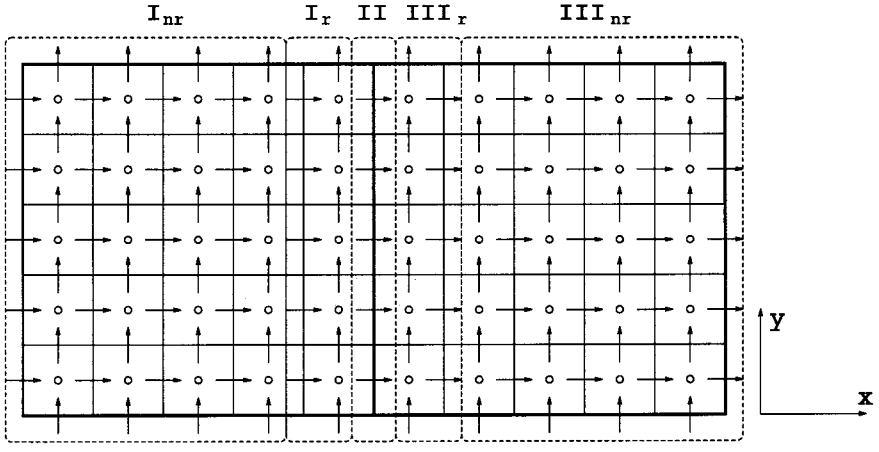

Figure 3. Definition of three sets of unknowns: I_{nr} and I_r constitute set I; III_{nr} and III_r constitute set III.

The solution of Equation (15) always satisfies $\bar{V}_2^* = \bar{V}'^*_2$ if S_{22} is invertible (see [31]) and thus, Equation (15) is equivalent to the original system of equations $S(V^n, P^n)V^* = f$. In view of Equation (11), we have

$$A_{11} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \quad \text{and} \quad A_{22} = \begin{bmatrix} S_{22} & S_{23} \\ S_{32} & S_{33} \end{bmatrix}, \tag{16}$$

so that the domain decomposition for the momentum equations has been described.

In Equation (13), v_1^{m+1} only depends on $A_{12}v_2^m$. Since A_{12} has only non-zero coefficients for unknowns in $\mathrm{III_r}$ (see Figures 2 and 3), the left-hand side v_1^{m+1} only depends on these components. Similarly, v_2^{m+1} only depends on the components of v_1^m in region $\mathrm{I_r}$. The components in $\mathrm{I_r}$ and $\mathrm{III_r}$ are assembled in a vector v_r, also called the *interface unknowns*, and the remaining ones in v_{nr}. The normal fluxes at the block interface in region II are a part of the inner regions of the subdomains, and are solved for in each iteration.

Finally, the last step (7) of the pressure-correction algorithm is independently carried out in all blocks. The above discussion can easily be extended to the general multidomain case. Also, extensions to irregular intersections are possible (see [20,22,32] for examples).

3.2. Accurate subdomain solution

In Reference [12], subdomain problems are assumed to be solved accurately so that N^{-1} is the *exact inverse* of the block diagonal or block lower-triangular matrix of A, so

$$N = N_\mathrm{gs} = \begin{bmatrix} A_{11} & \varnothing \\ A_{21} & A_{22} \end{bmatrix} \quad \text{or} \quad N = N_\mathrm{jac} = \begin{bmatrix} A_{11} & \varnothing \\ \varnothing & A_{22} \end{bmatrix}, \tag{17}$$

where N_gs is the Gauss–Seidel version and N_jac is 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 observed from Figures 2 and 3 that the left-hand side of Equation (12) only depends on the values of u^m in regions $\mathrm{I_r}$ and $\mathrm{III_r}$ in Figure 3. The unknowns u are ordered in such a way that

$$u = \begin{bmatrix} w \\ v \end{bmatrix},$$

where v is the interface unknowns (regions $\mathrm{I_r}$, $\mathrm{III_r}$), and w is the remaining unknowns. We have

$$(I - N^{-1}A)u = (I - N^{-1}A)Qv, \tag{18}$$

with

$$Q = \begin{bmatrix} 0 \\ I \end{bmatrix} \text{ an injection operator such that } Qv = \begin{bmatrix} 0 \\ v \end{bmatrix}.$$

By substituting Equation (18) into Equation (12) and by premultiplying with Q^T we get

$$v^{m+1} = Q^T u^{m+1} = Q^T(I - N^{-1}A)Qv^m + Q^T N^{-1} f. \tag{19}$$

Since we are interested in the stationary solution v of (19) we get

$$v = Q^T(I - N^{-1}A)Qv + Q^T N^{-1} f, \tag{20}$$

which is equivalent to

$$Q^T N^{-1} A Q v = Q^T N^{-1} f. \tag{21}$$

In this way, accurate solutions of subdomain problems finally leads to a system involving only the interface equations. Accelerated domain decomposition in Reference [12] amounts to solving the interface Equations (21) using GMRESR [33]. In this paper, we use GMRES; the required matrix–vector product can be computed by doing one domain decomposition iterations, see [12] for details.

3.3. Inaccurate subdomain solution

Domain decomposition iteration (12) is typically implemented as

$$\tilde{N} u^{m+1} = (N - A) u^m + f, \tag{22}$$

where the right-hand side term $(N-A)u^m$ represents the discretization of the internal boundary conditions, which is always exact, and the left-hand side term $\tilde{N}u^{m+1}$ indicates solutions of the subdomain problems using some type of solver, which was assumed accurate enough in the previous section.

In general, the stationary solution of Equation (22) satisfies the perturbed equations $(A + \tilde{N} - N)u = f$ instead of $Au = f$. With inaccurate subdomain solutions, the difference between $\tilde{N}$ and N may be quite large, thus, the computed solution u may have a very large error. Since the algorithm of the previous section relies on Equation (22), we may not use this procedure with inaccurate solutions of subdomain problems. Instead we must use

$$u^{m+1} = u^m + \tilde{N}^{-1}(f - A u^m), \tag{23}$$

for which the stationary solution u always satisfies $Au = f$.

With inaccurate subdomain solutions, we have

$$\tilde{N} = \tilde{N}_{\text{gs}} = \begin{bmatrix} \tilde{A}_{11} & \varnothing \\ A_{21} & \tilde{A}_{22} \end{bmatrix} \quad \text{or} \quad \tilde{N} = \tilde{N}_{\text{jac}} = \begin{bmatrix} \tilde{A}_{11} & \varnothing \\ \varnothing & \tilde{A}_{22} \end{bmatrix}, \tag{24}$$

where $\tilde{N}_{\text{gs}}$ is the Gauss–Seidel (sequential/multiplicative) version and $\tilde{N}_{\text{jac}}$ is the Jacobi (parallel/additive) version of $\tilde{N}$. The matrices $\tilde{A}_{ii}$ represent inaccurate subdomain solutions. The matrix–vector product $p = \tilde{N}_{\text{gs}}^{-1} t$ is computed as

$$\begin{aligned} p_1 &= \tilde{A}_{11}^{-1} t_1 \\ p_2 &= \tilde{A}_{22}^{-1}(t_2 - A_{21} t_1), \end{aligned} \tag{25}$$

where, for instance, $\tilde{A}_{11}^{-1} t_1$ represents an approximate solution in subdomain 1 with a low accuracy. Another possibility is to take $\tilde{A}_{ii} = L_i U_i$ to be some incomplete LU factorization of A_{ii} (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 Euclidean norm. Specifically, with an initial guess $p_{i_0} = 0$ 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, GMRES acceleration cannot be used since the preconditioner $\tilde{N}$ varies in each step. Only in the case $\tilde{A}_{ii} = L_i U_i$ may we apply GMRES acceleration, but we still apply GCR in this case.

3.4. Theoretical motivation

Inaccurate solutions of subproblems reduces the amount of work in each domain decomposition iteration at the cost of some additional work in the outer domain decomposition iteration. Therefore, this approach can only lead to a reduction in computing time if the increase in outer domain decomposition iterations is small.

A simple analysis of the condition number of the postconditioned matrix $A\tilde{N}^{-1}$ confirms this statement. For symmetric problems, the condition number is a good estimate for the rate of convergence $((\sqrt{\kappa}-1)/(\sqrt{\kappa}+1)$ for CG). For non-symmetric problems, the condition number is less-closely linked to convergence.

Each iteration involves solving

$$Nu = g, \tag{26}$$

where N is the matrix from Equation (12). With inaccurate solutions of subdomains, we solve a problem

$$\tilde{N}\tilde{u} = g, \tag{27}$$

where $\tilde{N}$ as in Equations (23) and (24). All subproblems are solved using a relative accuracy.

Condition 1. Each subproblem $A_{ii}u_i = g_i$ is solved using an initial guess of 0, and with a relative accuracy of ϵ so that $\|g_i - A_{ii}\tilde{u}_i\| \leq \epsilon \|g_i\|$ in the Euclidean norm.

Theorem 1 relates N and $\tilde{N}$.

Theorem 1. If condition 1 holds for all subdomains and all possible right-hand side g_i, then
(a) $\|I - N_{\text{gs}}\tilde{N}_{\text{gs}}^{-1}\| \leq C\epsilon$, for some constant $C > 0$.
(b) $\|I - N_{\text{jac}}\tilde{N}_{\text{jac}}^{-1}\| \leq \epsilon$.

Proof
(a) The combination of Condition 1 with $\tilde{A}_{ii}\tilde{u}_i = g_i$ (inaccurate subdomain solution) gives $\|g_i - A_{ii}\tilde{A}_{ii}^{-1}g_i\| = \|(I - A_{ii}\tilde{A}_{ii}^{-1})g_i\| \leq \epsilon\|g_i\|$ for all g_i. From the definition of a matrix norm, it follows that $\|I - A_{ii}\tilde{A}_{ii}^{-1}\| \leq \epsilon$. Without loss of generality, we take two subdomains, so that N and $\tilde{N}$ are described by Equations (17) and (24), respectively. We get

$$I - N\tilde{N}^{-1} = \begin{bmatrix} I - A_{11}\tilde{A}_{11}^{-1} & \varnothing \\ -(I - A_{22}\tilde{A}_{22}^{-1})A_{21}\tilde{A}_{11}^{-1} & I - A_{22}\tilde{A}_{22}^{-1} \end{bmatrix}. \tag{28}$$

The partition x,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

and note that for the Euclidean norm

$$\|x\| \leq \left\| \begin{bmatrix} x_1 \\ 0 \end{bmatrix} \right\| + \left\| \begin{bmatrix} 0 \\ x_2 \end{bmatrix} \right\| = \|x_1\| + \|x_2\|,$$

then we have

$$\begin{aligned} \|I - N\tilde{N}^{-1}\| &= \sup_{\|x\| \leq 1} \|(I - N\tilde{N}^{-1})x\| \\ &\leq \sup_{\|x\| \leq 1} [\|(I - A_{11}\tilde{A}_{11}^{-1})x_1\| + \|(I - A_{22}\tilde{A}_{22}^{-1})A_{21}\tilde{A}_{11}^{-1}x_1\| \\ &\quad + \|(I - A_{22}\tilde{A}_{22}^{-1})x_2\|]. \end{aligned}$$

Furthermore, for the Euclidean norm $\|x\| \le 1$ implies $\|x_1\| \le 1$ and $\|x_2\| \le 1$ so that finally, (a) follows with $C = 2 + \|A_{21}\tilde{A}_{11}^{-1}\|$.

(b) For any block diagonal matrix $B = \mathrm{diag}(D_1, D_2, \ldots, D_n)$, we have

$$\|B\| = \sqrt{\rho(B^T B)} = \sqrt{\rho[\mathrm{diag}(D_1^T D_1, \ldots, D_n^T D_n)]} = \max\{\sqrt{\rho(D_1^T D_1)}, \ldots, \sqrt{\rho(D_n^T D_n)}\}$$
$$= \max\{\|D_1\|, \ldots, \|D_n\|\}.$$

If we use the additive postconditioner, then $I - N\tilde{N}^{-1}$ is a block diagonal matrix with blocks $D_i = I - A_{ii}\tilde{A}_{ii}^{-1}$, so that

$$\|I - N\tilde{N}^{-1}\| = \max_i \|I - A_{ii}\tilde{A}_{ii}^{-1}\| \le \epsilon.$$

Therefore, (b) holds. □

Theorem 1 enables a relationship between the condition numbers of $A\tilde{N}^{-1}$ and AN^{-1} to be given.

Theorem 2. Under the conditions of Theorem 1 and $C\epsilon < 1$, the condition number of $A\tilde{N}^{-1}$ satisfies

$$\kappa(A\tilde{N}^{-1}) \le \frac{1 + C\epsilon}{1 - C\epsilon} \cdot \kappa(AN^{-1}). \tag{29}$$

Proof
The application of Theorem 1, and noting that $\|*\|$ is a least upper-bound norm, gives $\|N\tilde{N}^{-1}\| = \|N\tilde{N}^{-1} - I + I\| \le 1 + C\epsilon$ and $\|(N\tilde{N}^{-1})^{-1}\| = \|(N\tilde{N}^{-1})^{-1}(I - N\tilde{N}^{-1}) + I\| \le 1 + \|(N\tilde{N}^{-1})^{-1}\| C\epsilon$.
Since $C\epsilon < 1$, $\kappa(N\tilde{N}^{-1}) = \|NN^{-1}\| \cdot \|(N\tilde{N}^{-1})^{-1}\| \le (1 + C\epsilon)/(1 - C\epsilon)$.
The inequality in (29) follows from $\kappa(A\tilde{N}^{-1}) = \kappa(AN^{-1}N\tilde{N}^{-1}) \le \kappa(AN^{-1}) \cdot \kappa(N\tilde{N}^{-1})$. □

Theorem 2 shows that the subdomain solution accuracy has only a small effect on the condition number of the postconditioned matrix. This means that (at least for symmetric problems) the number of outer iterations will not increase (significantly) when the subdomain accuracy is lowered. The sensitivity of outer-loop convergence to ϵ is given by the constant C in Theorem 1, which can be chosen as 1 for the additive algorithm, independently of the number of subdomains. This means that for additive algorithms, the condition $C\epsilon$ is satisfied automatically ($\epsilon < 1$). For multiplicative algorithms, this sensitivity constant C will probably also be small and independent of the number of subdomains; however, sharper bounds may require a much more detailed analysis.

The theorems only hold for constant $\tilde{N}$. Also, the exact convergence behavior of the Krylov subspace methods depends on the distribution of the eigenvalues of $A\tilde{N}^{-1}$. For symmetric, positive definite matrices $A\tilde{N}^{-1}$, the condition number $\kappa(A\tilde{N}^{-1})$ only relates to the extreme eigenvalues of $A\tilde{N}^{-1}$, and as such provides only a rough convergence estimate. Nevertheless, these theorems provide a theoretical justification of inaccurate subdomain solutions in simple cases. We shall see in Section 6 that the conclusions also hold in the case if $\tilde{N}$ varies in each iteration.

4. KRYLOV SUBSPACE ACCELERATION

The basic Schwarz domain decomposition iteration converges slowly and is not always convergent for the Navier–Stokes equations. Therefore, we use Krylov subspace acceleration, which is frequently used to accelerate domain decomposition methods, see e.g. Reference [34] and many of the papers on *iterative substructuring* methods in [35–39]. The acceleration procedure used with accurate solutions of subdomain problems is GMRES applied to the interface Equations (21) and is described in detail in References [10–12]. This section describes the procedure used with inaccurate subdomain solutions.

The GCR [13] method for solving $Ax = f$ can easily be adapted to cope with variable preconditioners. Because of its simplicity and for completeness sake, we describe the GCR method here. GCR is based on maintaining two subspaces—a subspace $S_k = \langle s_1, s_2, \ldots, s_k \rangle$ for storing the search directions s_i, and a subspace $V_k = \langle v_1, v_2, \ldots, v_k \rangle$ with $As_i = v_i$. In every operation of GCR, the property $As_i = v_i$ is preserved. For simplicity, we take the initial guess $x_0 = 0$, in which case GCR minimizes the residual $\|f - Ax_k\|_2$ 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 $f - Ax_k \perp v_i$ for $i = 1, \ldots, k$, therefore

$$(f - Ax_k, v_i) = 0. \tag{30}$$

Since $Ax_k \in V_k$ we have

$$Ax_k = \sum_{j=1,\ldots,k} \lambda_j v_j, \tag{31}$$

and by substituting Equation (31) into Equation (30) we get $\lambda_i = (f, v_i)$ so that

$$Ax_k = \sum_{i=1,\ldots,k} (f, v_i) v_i. \tag{32}$$

Since $As_i = v_i$, we have

$$Ax_k = \sum_{i=1,\ldots,k} (f, v_i) As_i = A \sum_{i=1,\ldots,k} (f, v_i) s_i, \tag{33}$$

so that $x_k = \Sigma_{i=1,\ldots,k} (f, v_i) s_i$. This gives $x_{k+1} = x_k + (f, v_{k+1}) s_{k+1}$ and with $r_k = f - Ax_k$, we get $r_{k+1} = r_k - (f, v_{k+1}) v_{k+1}$. 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 4 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 [9]. For this choice of search direction, the space S_k is called the Krylov space. The difference between GCR and GMRES is that, with the benefit of allowing more general search directions, GCR requires twice the storage of GMRES and 1.5 times the number of floating point operations for orthogonalization. However, GCR can be combined with truncation strategies, e.g. the Jackson and Robinson [40] strategy, whereas GMRES can only be restarted. Because of this, truncated GCR may converge faster than GMRES (see Section 6.2). Furthermore, restarted GCR can be optimized [41], which makes GCR just as efficient as GMRES. Both optimized restarted GCR and truncated GCR will be considered in our numerical experiments.

$r_0 = f - Ax_0$; $k = 0$
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
    $\gamma = (f, v_{k+1})$
    $x_{k+1} = x_{k+1} + \gamma s_{k+1}$
    $r_{k+1} = r_{k+1} - \gamma v_{k+1}$
    $k = k + 1$
end while

Figure 4. The GCR algorithm with general search directions without restart and with a relative stopping criterion [33].

Recent developments have led to a more flexible GMRES algorithm which allows more general search directions, so-called FGMRES [42]. FGMRES is used in [43] to investigate the Neumann–Dirichlet method with inexact subdomain solutions. The emphasis in [43] is on restrictions on subdomain solution accuracy to retain the h-independent convergence of the Neumann–Dirichlet algorithm rather than on reduction of computing time. Optimized restarted GCR is just as efficient as FGMRES, both in memory requirements and work.

In this paper, we use $s_{k+1} = \tilde{N}^{-1} r_k$, which corresponds to a single iteration of Equation (23) with an initial guess $u^m = 0$. The case of multiple iterations of Equation (23) to determine s_{k+1} is not considered in this paper. If the subdomain problems are solved (inaccurately) using GMRES, this method reduces to GMRESR [33] for the single domain case. In case $\tilde{A}_{ii} = L_i U_i$ is the (relaxed) incomplete LU factorization of A_{ii}, we obtain a blocked version of the subdomain RILU(α) [44] postconditioner (with parameter α), here called RIBLU(α) (relaxed incomplete block LU). The parameter α may be varied to improve convergence. The RIBLU(α) *pre*conditioner is investigated for parallel implementation in e.g. [45–48]. The present paper also investigates the multiplicative version of the RIBLU(α) postconditioner. The GMRES acceleration procedure may be applied with RIBLU(α), which in this case is equivalent to GCR acceleration.

The stopping criterion for accurate solutions of subdomain problems differs from that for inaccurate solutions. With accurate solutions, 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 solutions, it is based on the unpreconditioned residual $r = f - Au$ of all unknowns. Therefore, a comparison between the two methods is difficult. Nevertheless, we assume that the final solution obtained with both methods is equally accurate if the relative stopping criterion $\|r_k\| \leq \epsilon \|r_0\|$ is used. This assumption was confirmed in [14]. With inaccurate

subdomain solutions, the results for different subdomain solution accuracies can be compared since the stopping criterion does not depend on the way subdomain problems are solved.

5. THE MODEL PROBLEM

We shall consider flow around a cylinder in a wall-bounded shear flow. This problem models the removal of particles from surfaces. Examples of where this type of flow occurs are: the cleaning of surfaces by water jets, vacuum cleaners and the contamination of surfaces. An example of the latter is the disposal route of irradiated fuel of nuclear reactors. Therefore, this problem is of considerable practical interest. From a numerical point of view, it is interesting because it requires a non-orthogonal grid and the results of the computation can be used to verify assumptions made by experimentalists [49,50]. The problem also requires large computing times, about 7 h on a single workstation, which makes it a challenge for algorithmic improvements and parallel computing.

Figure 5 shows the geometry and decomposition of the domain and a coarse version of the multiblock and single-block grids. Decompositions into more blocks are obtained by further decomposing the two blocks into subblocks.

The cylinder has a diameter $a = 2$. The Reynolds number is defined as

$$\mathrm{Re} = \frac{au^*}{v}, \tag{34}$$

with

$$u^* = \sqrt{\frac{\tau_0}{\rho}}, \tag{35}$$

where $\tau_0 = \mu\, \partial u/\partial y$ is the shear stress associated with the linear inlet velocity profile. Typical Reynolds numbers for this problem are $Re = 1$–5. Our results are given for $Re = 2$. In the computation we used $L = H = 10$. The boundary conditions are as follows

- **ABGFIBC:** $u = 0,\ v = 0$

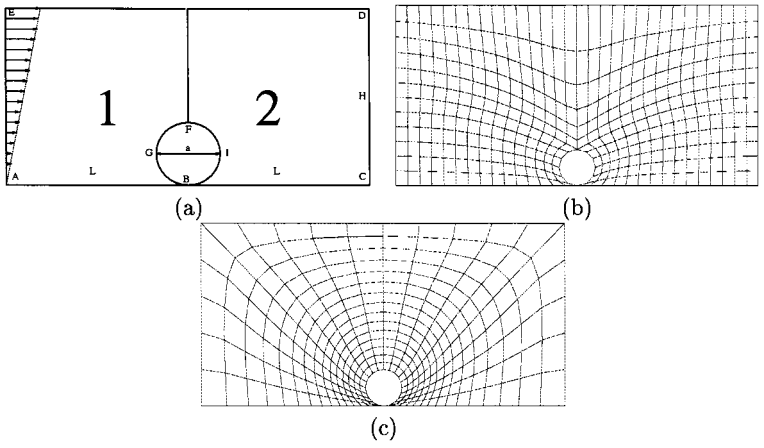

Figure 5. (a) Geometry and decomposition of the domain, (b) multiblock grid and (c) single-block grid.

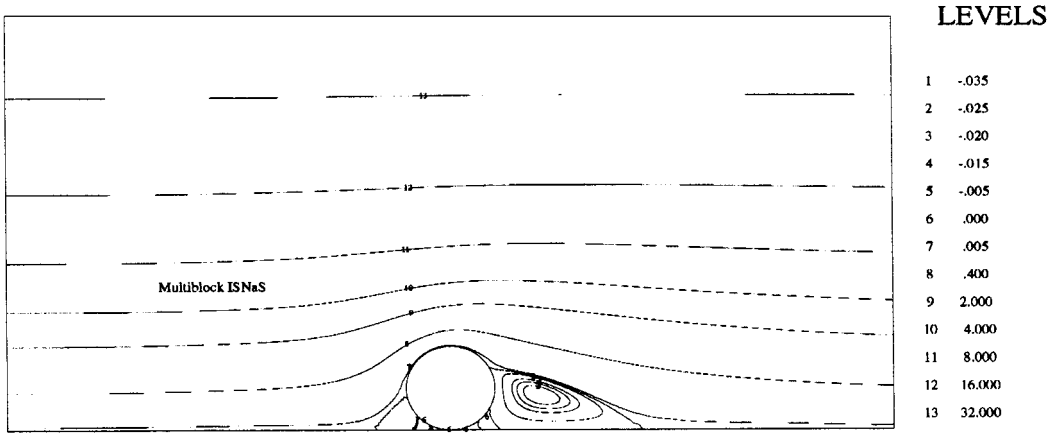

Figure 6. Streamlines of the stationary solution.

- **AE:** $\mathrm{u} = (\tau_0/\mu) \cdot y,\ u = 0$
- **DC:** $\sigma_{xx} = 0,\ v = 0$
- **ED:** $\sigma_{xy} = \tau_0,\ v = 0$

The stationary solution was computed using the implicit Euler time integration scheme, with a start time $t = 0$, an end time $t = 10$ and a time step of 0.02. The time measurements in the next section are only given for the first ten time steps to avoid excessive computing times. Figure 6 shows the streamlines for the stationary solution. For futher details on this computation refer to Reference [12].

6. RESULTS

This section compares accurate solutions with inaccurate solutions of subdomain problems. We consider both the cylinder problem from the previous section and a Poisseuille flow in a 0.1×0.1 rectangle. The global grid for the cylinder problem consists of 12 240 grid cells. The single-block cylinder grid in Figure 5 consists of 10 800 grid cells. For the Poisseuille problem, a Cartesian grid of 80×80 cells is used.

The subdomain problems are solved using GMRES with RILU(α) preconditioning [30,51] and a relative stopping criterion. For $\alpha = 0$, we get the standard ILU preconditioner [52] and for $\alpha = 1$ we get the modified ILU preconditioner [44]. RILU(α) [53] lies between these two. RILUD(α) represents RILU(α) restricted to the diagonal. The momentum equations are solved using a RILUD(0.95) preconditioner and the pressure equations using a RILU(0.975). RILU(α) will be used to mean RILUD(0.95) and RILU(0.975) whenever the momentum or the pressure equations are involved, respectively. The subdomain solution accuracy is varied. As a special case, the subdomain solution is approximated by means of the inverse of the RILU(α) [54,30,51] preconditioner, thereby omitting GMRES from subdomain solutions.

The multiblock problem (the outer-loop) is solved to a relative accuracy of 10^{-4}. In all experiments a Krylov space of dimension 20 is used for both GMRES and GCR multiblock acceleration and for GMRES subdomain solutions. GMRES always uses a restart after 20 iterations. With GCR, we investigate both optimized restart, denoted GCR (restart), and

truncation, denoted GCR (trunc). Iteration counts and computing times are given in the tables in the form *time*(*iteration count*). The iteration counts and times are totalled over all time steps taken (ten time steps are used in all examples). The experiments are run on a HP9000/735 workstation.

In most of the experiments, the multiplicative algorithm is used. Only Section 6.4 examines the additive algorithm. Section 6.1 examines the effect of lowering the accuracy of the subdomain solutions on the number of iterations and total computing time. Section 6.2 compares single-block solution time with multiblock solution time. Section 6.3 examines the effect of the α parameter in the subdomain RILU(α) preconditioner on convergence using the RIBLU(α) postconditioner.

6.1. Lowering the subdomain solution accuracy

Table I lists the computation times and iteration counts for the cylinder problem. A decomposition into two blocks is used as in Figure 5(a). The table shows the following quantities

- *Total:* the total computing time.
- *Momentum:* the total time and number of domain decomposition iterations needed to solve the multiblock problem for the momentum equations.
- *Pressure:* the total time and iterations needed to solve the pressure equations.
- *Other:* the total time involved in 'other' work, like building matrices, computing coefficients, correcting the subdomain velocity fields and writing the output file.

The time listed in the column *Other* is almost perfectly constant, as it should be since the amount of work in this category does not depend on the type of domain decomposition algorithm used. Method I uses GMRES for the outer-loop, based on (hypothetical) accurate solutions of subdomain problems, method II uses GCR for the outer-loop and uses subdomain solutions (with possible low accuracy) using GMRES, and method III approximates the subdomain solution using a single application of the subdomain RILU(α) preconditioner to the subdomain right-hand side.

As the subdomain solution accuracy is lowered from 10^{-4} to 10^{-1}, the number of outer GCR iterations shows only a small increase, which because of the reduced work in solving subproblems, results in a reduction of total computing time (in this case approximately by a factor of two). This result is in accordance with Theorem 2. This theorem proves that, for simple problems, only a small increase in condition number and therefore convergence rate is

Table I. Results with varying accuracy of the subdomain solution for the cylinder problem, multiplicative algorithm

	ϵ	Total	Momentum	Pressure	Other
I	10^{-8}	924.7	146.6(37)	722.8(157)	50.5
	10^{-4}	449.1	78.6(38)	315.2(154)	50.5
II	10^{-4}	465.2	58.4(37)	351.6(155)	50.5
	10^{-2}	292.1	43.5(38)	193.4(168)	50.5
	10^{-1}	230.3	47.5(48)	127.6(210)	50.4
III	RIBLU(α) post+GCR (trunc)	190.3	28.4(85)	106.8(566)	50.4
	RIBLU(α) post+GCR (restart)	179.0	26.7(85)	97.0(646)	50.6

Table II. Single-block solution using GMRES with RILU(α) postconditioning

Total	Momentum	Pressure	Other
Poisseuille flow			
94.0	46.5(407)	21.0(395)	24.4
Cylinder problem			
128.0	31.6(140)	48.9(497)	44.0

expected when the subdomain solution accuracy is lowered. See Section 3.4 for more discussion about the applicability of these theorems.

The use of the RIBLU(α) postconditioner (method III) results in smaller amounts of work per iteration at the cost of much larger iteration counts. The computing time is somewhat lower than for method II. This is contrary to our model study for the advection–diffusion equation [14], where the RIBLU(α) postconditioner resulted in a more significant drop in computing time. The reason is that RIBLU(α) preconditioner shows a larger increase in the number of iterations with respect to subdomain RILU(α) for α close to 1.0. This increase is not present when $\alpha = 0$, see Reference [14] and Section 6.3. The use of optimized restarted GCR instead of Jackson and Robinson truncation gives only a small reduction in computing time. For the momentum equations, the total number of iterations is the same; this is because the number of iterations per time step is below 20 (the dimension of the Krylov space).

6.2. Single-domain versus multidomain

One of the main reasons for investigating inaccurate solutions of subdomain problems is to reduce the excessive computing times observed in the multiblock incompressible Navier–Stokes solver [12], and to bring them closer to the single-block solution time. This also gives better prospects for parallel computing.

Table II lists the number of iterations and computing times for single-block solutions of the Poisseuille flow on an 80×80 grid and the single-block cylinder grid with 10 800 cells from Figure 5. The results are given for GMRES subdomain solutions using RlLU(α) postconditioning. Table III shows a comparison of single-block solutions and multiblock solutions for the momentum equations of different decompositions of the domain. Table IV shows a comparison of single-block solutions and multiblock solutions of the pressure equations for different decompositions of the domain. It is important to note that with the optimized restarted GCR method with RIBLU(α) postconditioning (the bottom row in Table IV), the maximum dimension of the Krylov space had to be increased to 40 to obtain convergence within 200 iterations per time step of the pressure equations.

Comparing Tables II and III, we see that for small numbers of subdomains, computing time for the momentum equations can be reduced to below that of single-block solutions in method II. However, for larger numbers of blocks this is not the case, which is possibly due to superlinear convergence of the subdomain solvers (see below). Method III is faster than method II for the cylinder problem, but not for the Poisseuille problem. An explanation is that the time step for the cylinder problem (0.02) is much smaller than that for the Poisseuille flow (0.1); this increases the diagonal of the momentum matrix considerably and improves convergence.

Comparing Tables II and IV, we see that for small numbers of subdomains, the computing time of method II, with inaccurate subdomain solutions, is still 2–3-fold larger than that with

single-block solutions. Also, for the Poisseuille flow in Table IV, inaccurate subdomain solutions do not always provide a speed-up. Method III leads to growing iteration counts and computing times for increasing numbers of blocks. The reason for the bad performance of the RIBLU(α) postconditioner is that for α close to 1.0, it performs badly with respect to single domain RILU(α) than for $\alpha = 0$ (see Section 6.3). We also see that the Jackson and Robinson truncation strategy is quite effective in reducing iteration counts compared with restarted GCR. The optimizations in GCR do not outweigh this increase in iteration count.

A possible reason for the modest reduction in computing time by method II for larger numbers of blocks is given below. For larger numbers of blocks, the subdomains are smaller and therefore, superlinear convergence of the subdomain GMRES solver can occur earlier. For example, in the case of superlinear convergence, lowering the subdomain solution accuracy from 10^{-2} to 10^{-1} might only save a single subdomain GMRES iteration (out of say six iterations). The subdomain solution accuracy therefore, only gives a small reduction of work needed to solve subdomains, but it may still cause a significant increase in the number of GCR iterations in the outer-loop.

The results in Tables III and IV confirm the remark in Section 3.4, that the constant C in Theorem 2 does not depend on the number of blocks for the multiplicative algorithm; the ratio of the number of iterations needed with $\epsilon = 10^{-2}$ and 10^{-1} does not increase as the domain is decomposed into subdomains.

Table III. Results with various decompositions into subdomains for the momentum equations, multiplicative algorithm

III		RIBLU(α) post+GCR (restart)		
Poisseuille flow				
	ϵ	Decomposition		
		2×2	4×4	5×5
I	10^{-8}	333.6(101)	190.8(120)	171.3(129)
	10^{-4}	147.3(102)	90.5(121)	87.5(129)
II	10^{-4}	120.1(89)	81.0(106)	82.8(116)
	10^{-2}	55.6(89)	50.6(108)	56.6(117)
	10^{-1}	45.3(111)	49.9(136)	57.1(146)
III	RIBLU(α) post+GCR (trunc)	51.5(241)	78.8(281)	92.5(295)
	RIBLU(α) post+GCR (restart)	42.8(241)	65.8(281)	86.4(295)
Cylinder problem				
	ϵ	Number of blocks		
		2	4	8
I	10^{-8}	146.6(37)	145.9(49)	113.6(49)
	10^{-4}	78.6(38)	80.7(49)	63.0(49)
II	10^{-4}	58.4(37)	65.7(47)	53.0(47)
	10^{-2}	43.5(38)	49.3(47)	42.4(47)
	10^{-1}	47.5(48)	52.5(56)	59.8(56)
III	RIBLU(α) post+GCR (trunc)	28.4(85)	30.8(86)	30.8(86)
	RIBLU(α) post+GCR (restart)	26.7(85)	29.1(86)	29.0(86)

Table IV. Results with various decompositions into subdomains for the pressure equations, multiplicative algorithm

Poisseuille flow				
	ϵ	Decomposition		
		2×2	4×4	5×5
I	10^{-8}	135.6(180)	104.2(297)	120.5(334)
	10^{-4}	63.8(188)	59.7(303)	77.1(344)
II	10^{-4}	77.8(201)	85.2(323)	108.2(361)
	10^{-2}	52.9(216)	69.1(333)	90.8(374)
	10^{-1}	56.0(303)	81.5(422)	103.8(456)
III	RIBLU(α) post+GCR (trunc)	53.7(483)	100.6(633)	120.3(660)
	RIBLU(α) post+GCR (restart)	43.3(599)	84.8(767)	124.4(867)
Cylinder problem				
	ϵ	Number of blocks		
		2	4	8
I	10^{-8}	722.8(157)	688.9(282)	496.2(343)
	10^{-4}	315.2(154)	328.4(285)	247.7(344)
II	10^{-4}	351.6(155)	352.1(279)	283.2(399)
	10^{-2}	193.4(168)	208.6(296)	201.6(374)
	10^{-1}	127.6(210)	155.1(371)	176.6(467)
III	RIBLU(α) post+GCR (trunc)	106.8(566)	149.5(742)	183.0(811)
	RIBLU(α) post+GCR (restart)	97.0(646)	173.2(1076)	225.2(1332)

6.3. *The influence of the parameter* α

The properties of the blocked RIBLU(α) preconditioner depend on the parameter α. Figure 7 shows the effect of the α parameter on convergence of domain decomposition for the momentum and pressure equations for the Poisseuille flow problem.

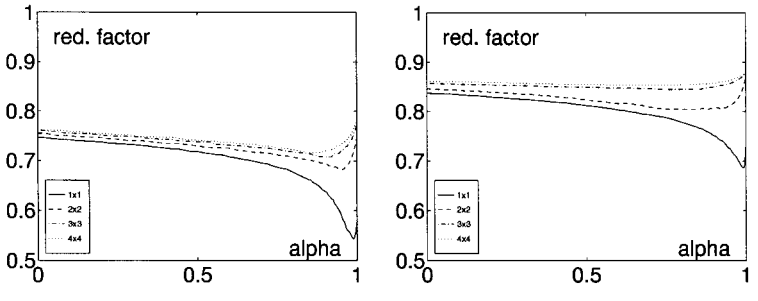

Figure 7. Effect of the α parameter using the RIBLU(α) preconditioner on the reduction factor: (a) momentum and (b) pressure equations.

Table V. Comparison between the multiplicative (Gauss–Seidel) and additive (Jacobi) algorithm for a decomposition into 8 blocks

ϵ	Multiplicative		Additive	
	Momentum	Pressure	Momentum	Pressure
10^{-4}	53.0(47)	283.2(399)	87.0(76)	574.8(674)
10^{-2}	42.4(47)	201.6(374)	68.2(76)	394.9(749)
10^{-1}	59.8(56)	176.6(467)	68.1(84)	276.0(736)
RIBLU(α) post+GCR (trunc)	30.8(86)	183.0(811)	37.0(101)	213.2(998)
RIBLU(α) post+GCR (restart)	29.0(86)	225.2(1332)	34.5(101)	332.3(1965)

Figure 7 confirms the observation in [14], that with standard ILU preconditioning (RILU(0)), the number of iterations shows a relatively small increase when the number of subdomains is increased. Larger values of $\alpha \leq 1$ can have a significant effect on convergence, but this effect diminishes rapidly as more subdomains are used. Clearly, varying the α parameter for multidomain problems has a much lower influence on convergence than for single-domain problems. Also, the optimal value of α is lower for multidomain problems.

6.4. Prospects for parallel implementation

In References [55,56], parallelization of domain decomposition for the incompressible Navier–Stokes equations using accurate solutions of subdomain problems is investigated. The method performs well on a cluster of workstations. The reason is that with accurate solutions of subdomain problems, the parallelization is rather coarse grained. Furthermore, the reduction to a system of interface Equations (21) makes a very simple parallel implementation possible.

In this section, we take a brief look at the possibilities for parallel implementation of the GCR accelerated method of this paper. Table V shows a comparison between the multiplicative and additive algorithms. We see that the penalty for going from the multiplicative to the additive algorithm is between 1.5 and 2 for method II, which is more than that for method III.

Table V shows that the number of iterations only increases slightly as the subdomain solution accuracy is lowered to $\epsilon = 10^{-1}$. This means that lowering the subdomain accuracy will almost certainly give a lower computing time. Method III requires much more iterations, especially for the pressure equations, and therefore communication. Therefore, method II is more suitable for parallel processing than method III. Again, GCR with Jackson and Robinson truncation is quite effective for method III compared with optimized restarted GCR.

The results in Table V show that for this problem, the multiplicative algorithm is more sensitive to the subdomain solution accuracy than the additive algorithm is; probably because of errors made in solving subproblems propagating to other subdomains within a single iteration.

7. CONCLUSIONS

It is possible to obtain significant reductions in computing time by using inaccurate solutions of subproblems. When the subdomain solution accuracy is lowered, the number of iterations

increases only slightly, which is confirmed by Theorem 2. Especially for small numbers of blocks (equivalently, large subdomains), the reduction in computing time can be quite large. The actual reductions obtained by inaccurate subdomain solutions are very problem-dependent.

For small subdomain problems, superlinear convergence for the subdomain GMRES solver can occur earlier so that a reduction in subdomain solution accuracy can lead to only a very small reduction of work needed to solve subdomain problems, but may still cause a more significant increase in the number of iterations needed by the outer GCR iteration. Also, for smaller subdomain problems, the amount of work needed to solve the subdomain problems is already small compared with the overhead of GCR acceleration, so that solving the subdomain problems more inaccurately can only lead to small reductions in computing time.

The sensitivity of convergence in the outer GCR loop stays approximately the same as the number of subdomains is enlarged. This was shown to be true for the additive algorithm in Section 3.4 (Theorems 1 and 2), but it probably also holds for the multiplicative algorithm. Convergence of the multiplicative algorithm seems to be more sensitive to the subdomain solution accuracy than the additive algorithm.

The RIBLU(α) postconditioned GCR method does not perform well for α close to 1. For such methods, the α parameter only improves convergence of single-block solutions; its effect on convergence of multiblock solutions is less. As shown in [14] and Figure 7, the number of iterations needed with multiplicative RIBLU(0) postconditioners is only slightly larger than that with single-block solutions. Generalizations of the RIBLU(α) postconditioner to more subdomains that preserve this property are therefore of interest. Furthermore, overheads in the implementation can be quite important and are to the disadvantage of the RIBLU(α) algorithms. These disadvantages of the current RIBLU(α) postconditioner prevent a reduction of computing time to that of single-block solutions. The optimized restarted GCR method does not give significant reductions in computing time because of an increased number of iterations compared with Jackson and Robinson truncation.

Parallel implementation of the GCR-based algorithm is attractive because convergence of the outer GCR loop does not depend sensitively on the subdomain solution accuracy. Therefore, the number of iterations will, in general, be approximately the same as with very accurate subdomain solutions, so that reduced computing time is almost certain. Only for very inaccurate subdomain solutions, e.g. when the RIBLU(α) postconditioner is used, we get a significant increase in the number of iterations and therefore communication.

Inaccurate solutions of subdomain problems combined with GCR acceleration removes the restriction, inherent in GMRES solutions of interface Equations (21), that subdomain problems should be solved accurately (enough). The GCR-based algorithm is therefore, more reliable than the GMRES algorithm for solving interface equations.

REFERENCES

1. A.E. Mynett, P. Wesseling, A. Segal and C.G.M. Kassels, 'The ISNaS incompressible Navier–Stokes solver: invariant discretization', *Appl. Sci. Res.*, **48**, 175–191 (1991).
2. C.W. Oosterlee and P. Wesseling, 'A multigrid method for an invariant formulation of the incompressible Navier–Stokes equations in general co-ordinates', *Comm. Appl. Numer. Methods*, **8**, 721–725 (1992).
3. 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 co-ordinates on staggered grids', *Comput. Fluids Dyn. J.*, **1**, 27–33 (1992).
4. A. Segal, P. Wesseling, J.J.I.M. van Kan, C.W. Oosterlee and C.G.M. Kassels, 'Invariant discretization of the incompressible Navier–Stokes equations in boundary fitted co-ordinates', *Int. j. numer. methods fluids*, **15**, 411–426 (1992).

5. M. Zijlema, A. Segal and P. Wesseling, 'Finite volume computation of incompressible turbulent flows in general co-ordinates on staggered grids', *Int. j. numer methods fluids*, **20**, 621–640 (1995).
6. M. Zijlema, A. Segal and P. Wesseling, 'Invariant discretization of the k-ε model in general co-ordinates for prediction of turbulent flows in complicated geometries', *Comput. Fluids*, **24**, 209–225 (1995).
7. C.W. Oosterlee, P. Wesseling, A. Segal and E. Brakkee, 'Benchmark solutions for the incompressible Navier–Stokes equations in general co-ordinates on staggered grids', *Int. j. numer. methods fluids*, **17**, 301–321 (1993).
8. H.A. Schwarz, *Gesammelte Mathematische Abhandlungen*, vol. 2, Springer, Berlin, 1890, pp. 133–143; first published in *Viertelsjahrschrift der Naturforschenden Gesellschaft in Zürich*, vol. 15, 1870, pp. 272–286.
9. Y. Saad and M.H. Schultz, 'GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).
10. E. Brakkee and P. Wilders, 'The influence of interface conditions on convergence of Krylov–Schwarz domain decomposition for the advection–diffusion equation', *J. Sci. Comput.* (1997), to appear.
11. E. Brakkee, 'Domain decomposition for the incompressible Navier–Stokes equations', *Ph.D. Dissertation*, Faculty of Applied Mathematics and Informatics, Delft University of Technology, Delft, Netherlands, 1996.
12. E. Brakkee and P. Wesseling, 'Schwarz domain decomposition for the incompressible Navier–Stokes equations in general co-ordinates', Rep. 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, submitted to Int. j. numer. methods fluids.
13. S.C. Eisenstat, H.C. Elman and M.H. Schultz, 'Variational iterative methods for non-symmetric systems of linear equations', *SIAM J. Numer. Anal.*, **20**, 345–357 (1983).
14. E. Brakkee, C. Vuik and P. Wesseling, 'An investigation of Schwarz domain decomposition using accurate and inaccurate solution of subdomains', Rep. 95-18, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1995. Available from anonymous ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1995/DUT-TWI-95-18.ps.gz.
15. F.H. Harlow and J.E. Welch, 'Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface', *Phys. Fluids*, **8**, 2182–2189 (1965).
16. A.J. Chorin, 'Numerical solution of the Navier–Stokes equations', *Math. Comput.*, **22**, 745–762 (1968).
17. J.J.I.M. van Kan, 'A second-order accurate pressure correction method for viscous incompressible flow', *SIAM J. Sci. Stat. Comput.*, **7**, 870–891 (1986).
18. O.B. Widlund, 'Some Schwarz methods for symmetric and non-symmetric elliptic problems', in D.E. Keyes, T.F. Chan, G. Meurant, J.S. Scroggs and R.G. Voigt (eds), *Proc. 5th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1992, pp. 19–36.
19. Xiao-Chuan Cai, W.D. Gropp and D.E. Keyes, 'A comparison of some domain decomposition and ILU preconditioned iterative methods for non-symmetric elliptic problems', *Numer. Linear Algebra Appl.*, **1**, 477–504 (1994).
20. W.D. Henshaw and G. Chesshire, 'Multigrid on composite meshes', *SIAM J. Sci. Stat. Comput.*, **8**, 914–923 (1987).
21. J.C. Strikwerda and C.D. Scarbnick, 'A domain decomposition method for incompressible flow', *SIAM J. Sci. Comput.*, **14**, 49–67 (1993).
22. J.A. Wright and W. Shyy, 'A pressure-based composite grid method for the Navier–Stokes equations', *J. Comput. Phys.*, **107**, 225–238 (1993).
23. W.D. Gropp and B.F. Smith, 'Experiences with domain decomposition in three dimensions: overlapping Schwarz methods', in A. Quarteroni, J. Périaux, Yu.A. Kuznetsov and O.B. Widlund (eds), *Proc. 6th Int. Symp. on Domain Decomposition Methods in Science and Engineering*, AMS, Providence, RI, 1992, pp. 323–333.
24. M. Dryja and O.B. Widlund, 'Some recent results on Schwarz-type domain decomposition algorithms', in A. Quarteroni, J. Périaux, Yu.A. Kuznetsov and O.B. Widlund (eds), *Proc. 6th Int. Symp. on Domain Decomposition Methods in Science and Engineering*, AMS, Providence, RI, 1992, pp. 53–61.
25. J. Mandel and S. McCormick, 'Iterative solution of elliptic equations with refinement: the two-level case', in T.F. Chan, R. Glowinski, J. Périaux and O.B. Widlund (eds), *Proc. 2nd Int. Symp. on Domain Decomposition Methods*, SIAM, Philadelphia, PA, 1989, pp. 81–92.
26. J.H. Bramble, J.E. Pasciak and A.H. Schatz, 'The construction of preconditioners for elliptic problems by substructuring I', *Math. Comput.*, **47**, 103–134 (1986).
27. P. Bjørstad and M.D. Skogen, 'Domain decomposition algorithms of Schwarz type, designed for massively parallel computers', *Report in Informatics 54*, Department of Informatics, University of Bergen, Bergen, 1991.
28. C.P. Jackson and P.C. Robinson, 'A numerical study of various algorithms related to the preconditioned conjugate gradient method', *Int. j. numer. methods eng.*, **21**, 1315–1338 (1985).
29. P.L. Lions, 'On the Schwarz alternating method, I', in R. Glowinski, G.H. Golub, G.A. Meurant and J. Périaux (eds), *1st Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1988, pp. 1–42.
30. C. Vuik, 'Solution of the discretized incompressible Navier–Stokes equations with the GMRES method', *Int. j. numer. methods fluids*, **16**, 507–523 (1993).
31. Wei Pai Tang, 'Generalized Schwarz splittings', *SIAM J. Sci. Stat. Comput.*, **13**, 573–595 (1992).
32. M.J. Berger, 'On conservation at grid interfaces', *SIAM J. Numer. Anal.*, **24**, 967–984 (1987).

33. H.A. van der Vorst and C. Vuik, 'GMRESR: a family of nested GMRES methods', *Numer. Linear Algebra Appl.*, **1**, 369–386 (1994).
34. P.E. Bjørstad and O.B. Widlund, 'Iterative methods for the solution of elliptic problems on regions partitioned into substructures', *SIAM J. Numer. Anal.*, **23**, 1097–1120 (1986).
35. R. Glowinski, G.H. Golub, G.A. Meurant and J. Périaux (eds), *1st Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1988.
36. T.F. Chan, R. Glowinski, J. Périaux and O.B. Widlund (eds), *Proc. 2nd Int. Symp. on Domain Decomposition Methods*, SIAM, Philadelphia, PA, 1989.
37. T.F. Chan, R. Glowinski, J. Periaux and O.B. Widlund (eds), *Proc. 3rd Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1990.
38. R. Glowinski, Yu.A. Kuznetsov, G. Meurant, J. Périaux and O.B. Widlund (eds), *Proc. 4th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1991.
39. D.E. Keyes, T.F. Chan, G. Meurant, J.S. Scroggs and R.G. Voigt (eds), *Proc. 5th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1992.
40. P. Bjørstad and M.D. Skogen, 'Domain decomposition algorithms of Schwarz type, designed for massively parallel computers', in D.E. Keyes, T.F. Chan, G. Meurant, J.S. Scroggs and R.G. Voigt (eds), *Proc. 5th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1992, pp. 362–375.
41. C. Vuik, 'New insights in GMRES-like methods with variable preconditioners', *J. Comput. Appl. Math.*, **61**, 189–204 (1995).
42. Y. Saad, 'A flexible inner–outer preconditioned GMRES algorithm', *SIAM J. Sci. Stat. Comput.*, **14**, 461–469 (1993).
43. C. Börgers, 'The Neumann–Dirichlet domain decomposition method with inexact solvers on the subdomains' *Numer. Math.*, **55**, 123–136 (1989).
44. I. Gustafsson, 'A class of first order factorization methods', *BIT*, **18**, 142–156 (1978).
45. R. di Brozolo and Y. Robert, 'Parallel conjugate gradient-like algorithms for solving sparse non-symmetric linear systems on a vector multiprocessor', *Parallel Comput.*, **11**, 223–239 (1989).
46. Wang Jin-xiau, 'The parallel block preconditioned conjugate gradient algorithms', in D.E. Keyes, T.F. Chan, G. Meurant, J.S. Scroggs and R.G. Voigt (eds), *Proc. 5th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1992, pp. 339–345.
47. 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 (eds), *6th Copper Mountain Conf. on Multigrid Methods*, NASA Conf. Publ. 3224, part 1, NASA Langley Research Center, Hampton, VA, 1993, pp. 111–125.
48. E. de Sturler, 'IBLU preconditioners for massively parallel computers', in D.E. Keyes and J. Xu (eds), *Domain Decomposition Methods in Science and Engineering* (*Proc. 7th Int. Conf. on Domain Decomposition*, October 27–30, 1993, The Pennsylvania State University), AMS, Providence, RI, 1995, pp. 395–400.
49. D. Hall, 'Measurements of the mean force on a particle near a boundary in turbulent flow', *J. Fluid Mech.*, **187**, 451–466 (1988).
50. A.M. Mollinger, 'Particle entrainment: measuring the fluctuating lift force', *Ph.D. thesis*, Delft University of Technology, Delft, Netherlands, 1994.
51. C. Vuik, 'Fast iterative solvers for the discretized incompressible Navier–Stokes equations', *Int. j. numer. methods fluids*, **22**, 195–210 (1996).
52. J.A. Meijerink and H.A. Van der Vorst, 'An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix', *Math. Comput.*, **31**, 148–162 (1977).
53. O. Axelsson and G. Lindskog, 'On the eigenvalue distribution of a class of preconditioning methods', *Numer. Math.*, **48**, 479–498 (1986).
54. 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).
55. 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 (eds), *Massively Parallel Processing Applications and Development*, Elsevier, Amsterdam, 1994, pp. 743–752.
56. E. Brakkee, A. Segal and C.G.M. Kassels, 'A parallel domain decomposition algorithm for the incompressible Navier–Stokes equations', *Simul. Pract. Theory*, **3**, 185–205 (1995).

