Domain decomposition for the incompressible Navier-Stokes equations: solving subdomain problems accurately and inaccurately

Report 95-37

Erik Brakkee
Kees Vuik
Piet Wesseling

TU Delft
Technische Universiteit Delft
Delft University of Technology

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

ISSN 0922-5641

Copies of these reports may be obtained from the bureau of the Faculty of Technical Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, phone +3115784568.
A selection of these reports is available in PostScript form at the Faculty's anonymous ftp-site, ftp.twi.tudelft.nl. They are located in directory /pub/publications/tech-reports. They can also be accessed on the World Wide Web at:
http://www.twi.tudelft.nl/TWI/Publications/Overview.html

Domain decomposition for the incompressible Navier-Stokes equations: solving subdomain problems accurately and inaccurately

Erik Brakkee,
Kees Vuik, Piet Wesseling

May 31, 1995

Abstract

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, such as the GMRES method, are used to accelerate convergence. With accurate subdomain solution, the amount of time spent in solving these problems may be quite large. To reduce computing time, inaccurate solution of subdomain problems is considered, which requires a different, 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.

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 [38, 17, 51, 43, 54, 55] describe the discretization in detail and [39, 54] discuss the capability of the method to accurately solve a number of laminar and turbulent flows. A Schwarz type domain decomposition iteration [42] in combination with GMRES [41] acceleration is used. In [10], significant reductions in computing time can be obtained using the GMRES acceleration procedure, see [11] and [10].

However, since the method described in [10] requires accurate solution of subdomain problems, it appears that the computing time can be much larger than with single-block solution for the same number of unknowns. Also, it is not known beforehand how accurate the subdomain problems must be solved. The required subdomain solution accuracy may be quite high, especially when grid cells are very much 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 solution and to allow (very) inaccurate subdomain solution. Since the preconditioner may now vary in each iteration, GMRES acceleration may no longer be applied. Instead, a method based on GCR [23] is used.

Considerable reductions in computing time can be obtained in this way for a 2-dimensional advection-diffusion equation, see [9]. Approximate subdomain solution using a single iteration with ILUD factorization reduced multi-block 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 solution of subdomain problems for the incompressible Navier-Stokes equations.

Parallel computing is of increasing importance. Therefore it is important to compare the parallel (additive) domain decomposition algorithms with the best multiplicative algorithms, which are known to be faster than additive algorithms. Therefore, we pay much attention to multiplicative algorithms.

2 Discretization

For the spatial discretization, we use a finite volume method employing a staggered grid and central discretization. The normal velocity components are located at the centers of the faces of the cell and the pressure unknowns are located at the centers of the cells, see Figure 1.

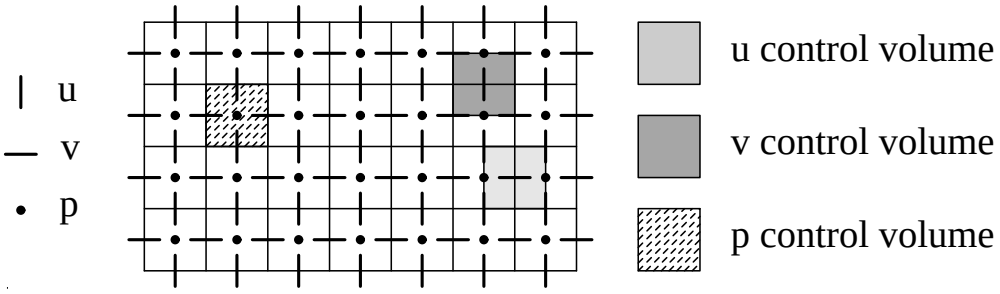


Figure 1: Arrangement of unknowns in a staggered grid

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, we get

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

$$DV^{n+1} = 0, \quad (2)$$

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. The pressure correction method [29, 16, 47] is used to solve (1) and (2). The pressure correction method consists of three steps. In the first step, an estimate $V^\star$ of V^{n+1} is computed by solving (1) with the pressure fixed at the old time level:

$$\frac{V^\star - V^n}{\Delta t} = M(V^n, P^n)V^\star - GP^n. \quad (3)$$

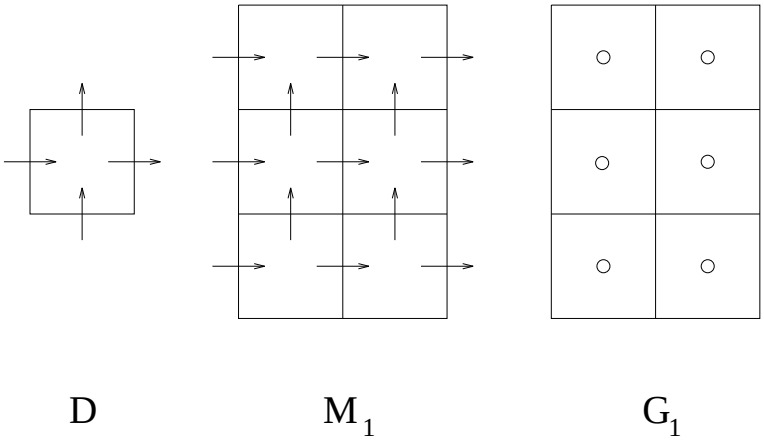


Figure 2: Discretization stencils: discretization of divergence operator D, x-component of momentum equations M_1 and x component of gradient matrix G_1.

In the second step, the pressure correction ΔP is solved from

$$DG\Delta P = \frac{DV^\star}{\Delta t}. \tag{4}$$

The last step consists of correcting the pressure: $P^{n+1} = P^n + \Delta P$ and computing V^{n+1} satisfying the incompressibility condition (2)

$$V^{n+1} = V^\star - \Delta t G \Delta P. \tag{5}$$

3 Domain decomposition

The basic approach to handle geometrically complex domains is to develop a domain decomposition version of (3)–(5). Subdomains are assumed to intersect regularly, so that grid lines are continuous across block-interfaces.

The domain decomposition algorithm considered here uses a minimal overlap and no coarse grid correction. It is known from theory [52] and experiment [13] that both constant overlap in physical space and a multi-level 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 [30, 44, 53]. However, as observed in [26, 22, 13], algorithms with small overlap can be quite effective, even for large and ill-conditioned problems. Although the number of GMRES iterations is typically higher with small overlap, this is compensated for by the fact that there is less duplication of work in the overlap regions. Methods with small overlap are also much easier to implement for practical complicated problems, and tend to dominate engineering applications.

A coarse grid correction [35, 12, 3, 4] can be quite effective for improving 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. Our present aim is to optimize efficiency of domain decomposition with minimal overlap.

3.1 General description

The pressure correction algorithm (3)–(5) is used for Navier-Stokes solution on the global domain. The equations (3) and (4) are solved using domain decomposition. In this paper, we assume that the subdomains intersect regularly, that is 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

$$S(V^n, P^n) = \frac{I}{\Delta t} - M(V^n, P^n) \tag{6}$$

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

$$T = DG \tag{7}$$

with D the global divergence and G the global gradient operator. Equations (6) and (7) are solved using domain decomposition. The correction of $V^\star$ is carried out in all blocks independently.

Both the pressure equations (4) and the momentum equations (3) can be written as

$$Av = f \tag{8}$$

with either $A = S$ from (6) and $v = V$ for the momentum equations or $A = T$ from (7) 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

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{9}$$

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 (8) is of the following form

$$v^{m+1} = (I - N^{-1}A)v^m + N^{-1}f, \tag{10}$$

with N^{-1} 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 [42, 34]. Similar to Schwarz domain decomposition, in each iteration, subdomain problems are solved using values from neighboring blocks. For instance, formula (10) interpreted for domain 1 becomes

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

where A_{11}^{-1} represents subdomain solution and v_2^m are the values from the neighboring block. The subdomain problems, $A_{11}x_1 = \ldots$ and $A_{22}x_2 = \ldots$ are solved using GMRES [41] with

appropriate preconditioners [50]. 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 3 sets as in Figure 3.

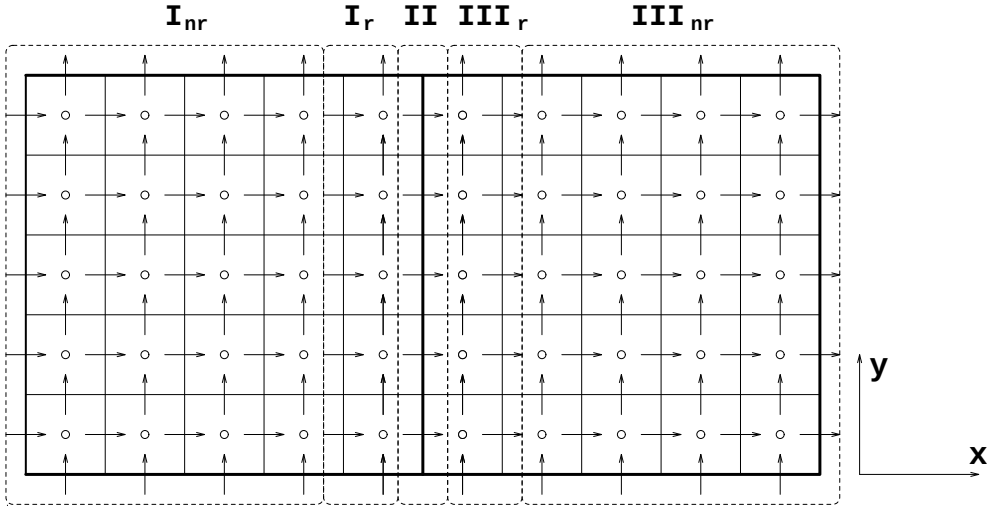


Figure 3: Definition of three sets of unknowns. I_{nr} and I_r consistute set I and III_{nr} and III_r consitute set III

- 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:

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

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

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

The solution of (13) always satisfies $\bar{V}_2^\star = \bar{V}'^\star_2$ if S_{22} is invertible (see [45]) and therefore the system (13) is equivalent to the original system of equations $S(V^n, P^n)V^\star = f$. In view of (9), we have

$$A_{11} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}, \quad A_{22} = \begin{bmatrix} S_{22} & S_{23} \\ S_{32} & S_{33} \end{bmatrix}. \tag{14}$$

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

In (11), v_1^{m+1} only depends on $A_{12}v_2^m$. Since A_{12} only has non-zero coefficients for unknowns in III_r, see Figures 2 and 3, the left-hand side v_1^{m+1} only depends on these components. Analogously, v_2^{m+1} only depends on the components of v_1^m in region I_r. The components in I_r and 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 (5) of the pressure-correction algorithm is carried out in all blocks independently. The above discussion can be easily extended to the general multi-domain case. Also extensions to irregular intersections are possible, see for example [2, 30, 53].

3.2 Accurate subdomain solution

In [11] 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

$$N = N_{gs} = \begin{bmatrix} A_{11} & \emptyset \\ A_{21} & A_{22} \end{bmatrix} \text{ or } N = N_{jac} = \begin{bmatrix} A_{11} & \emptyset \\ \emptyset & A_{22} \end{bmatrix} \tag{15}$$

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 from Figure 2 and 3 that the left-hand side of (10) only depends on the values of u^m in regions I_r and III_r in Figure 3. The unknowns u are ordered in such a way that $u = \begin{bmatrix} w \\ v \end{bmatrix}$, where v are the interface unknowns (regions I_r, III_r), and w are remaining unknowns. We have

$$(I - N^{-1}A)u = (I - N^{-1}A)Q\,v \tag{16}$$

with $Q = \begin{bmatrix} 0 \\ I \end{bmatrix}$ an injection operator such that $Q\,v = \begin{bmatrix} 0 \\ v \end{bmatrix}$. By substituting (16) into (10) and by premultiplying with Q^T we get

$$v^{m+1} = Q^T u^{m+1} = Q^T(I - N^{-1}A)Q\,v^m + Q^T N^{-1} f. \tag{17}$$

Since we are interested in the stationary solution v of (17) we get

$$v = Q^T(I - N^{-1}A)Q\,v + Q^T N^{-1} f. \tag{18}$$

which is equivalent to

$$Q^T N^{-1} A Q v = Q^T N^{-1} f. \tag{19}$$

In this way, accurate solution of subdomain problems finally leads to a system involving only the interface equations. Accelerated domain decomposition in [11] amounts to solving the interface equations (19) using GMRESR [46]. In the present paper, we use GMRES: the required matrix-vector product can be computed by doing one domain decomposition iteration, see [11] for details.

3.3 Inaccurate subdomain solution

Domain decomposition iteration (10) is typically implemented as

$$\tilde{N} u^{m+1} = (N - A)u^m + f, \tag{20}$$

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 solution of the subdomain problems using some type of solver, which was assumed accurate enough in the previous section.

In general, the stationary solution of (20) satisfies the perturbed equations $(A+\tilde{N}-N)u = f$ instead of $Au = f$. Since with inaccurate subdomain solution, the difference between $\tilde{N}$ and N may be quite large, the computed solution u may have a very large error. Since the algorithm of the previous section relies on (20), we may not use this procedure with inaccurate solution of subdomain problems. Instead we must use

$$u^{m+1} = u^m + \tilde{N}^{-1}(f - Au^m), \tag{21}$$

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

With inaccurate subdomain solution, we have

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

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

$$\begin{aligned} p_1 &= \tilde{A}_{11}^{-1} t_1 \\ p_2 &= \tilde{A}_{22}^{-1}(t_2 - A_{21} t_1) \end{aligned} \tag{23}$$

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 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, GMRES acceleration cannot be used since the preconditioner $\tilde{N}$ varies in each step. Only for the case $\tilde{A}_{ii} = L_i U_i$ we may apply GMRES acceleration, but we still apply GCR in this case.

3.4 Theoretical motivation

Inaccurate solution 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 ($\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ for CG) For unsymmetric problems the condition number is less closely linked to convergence.

Each iteration involves solving

$$N u = g \tag{24}$$

with N the matrix from (10). With inaccurate solution of subdomains, we solve a problem

$$\tilde{N}\tilde{u} = g \tag{25}$$

with $\tilde{N}$ as in (22) and (21). All subproblems are solved using a relative accuracy.

Condition 1 *Each subproblem $A_{ii}u_i = g_i$ is solved using initial guess 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 sides g_i, then*

(a) $\|I - N_{gs}\tilde{N}_{gs}^{-1}\| \leq C\epsilon$, *for some constant* $C > 0$.

(b) $\|I - N_{jac}\tilde{N}_{jac}^{-1}\| \leq \epsilon$.

Proof:

Proof of (a): 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 (15) and (22) respectively. We get*

$$I - N\tilde{N}^{-1} = \begin{bmatrix} I - A_{11}\tilde{A}_{11}^{-1} & \emptyset \\ -(I - A_{22}\tilde{A}_{22}^{-1})A_{21}\tilde{A}_{11}^{-1} & I - A_{22}\tilde{A}_{22}^{-1} \end{bmatrix} \tag{26}$$

Partition $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ *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* $\|I - N\tilde{N}^{-1}\| = \sup_{\|x\|\leq 1} \|(I - N\tilde{N}^{-1})x\| \leq \sup_{\|x\|\leq 1} \left(\|(I - A_{11}\tilde{A}_{11}^{-1})x_1\| + \|(I - A_{22}\tilde{A}_{22}^{-1})A_{21}\tilde{A}_{11}^{-1}x_1\| + \|(I - A_{22}\tilde{A}_{22}^{-1})x_2\| \right)$.
Furthermore, for the Euclidean norm $\|x\| \leq 1$ *implies* $\|x_1\| \leq 1$ *and* $\|x_2\| \leq 1$ *so that finally (a) follows with* $C = 2 + \|A_{21}\tilde{A}_{11}^{-1}\|$.
Proof of (b): For any block diagonal matrix $B = diag(D_1, D_2, \ldots, D_n)$*, we have:* $\|B\| = \sqrt{\rho(B^T B)} = \sqrt{\rho(diag(D_1^T D_1, \ldots, D_n^T D_n))} = \max\{\sqrt{\rho(D_1^T D_1)}, \ldots, \sqrt{\rho(D_n^T D_n)}\} = \max\{\|D_1\|, \ldots \|D_n\|\}$*. 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* $\|I - N\tilde{N}^{-1}\| = \max_i \|I - A_{ii}\tilde{A}_{ii}^{-1}\| \leq \epsilon$*. Therefore, (b) holds.*

□

Theorem 1 enables us to give a relation between the condition numbers of $A\tilde{N}^{-1}$ and AN^{-1}.

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

$$\kappa(A\tilde{N}^{-1}) \leq \frac{1 + C\epsilon}{1 - C\epsilon} \cdot \kappa(AN^{-1}). \tag{27}$$

Proof:

Application of Theorem 1, and noting that $\| \star \|$ *is a least upper bound norm, gives* $\|N\tilde{N}^{-1}\| = \|N\tilde{N}^{-1} - I + I\| \leq 1 + C\epsilon$ *and* $\|(N\tilde{N}^{-1})^{-1}\| = \|(N\tilde{N}^{-1})^{-1}(I - N\tilde{N}^{-1}) + I\| \leq 1 + \|(N\tilde{N}^{-1})^{-1}\|C\epsilon$.
Since $C\epsilon < 1$, $\kappa(N\tilde{N}^{-1}) = \|N\tilde{N}^{-1}\| \cdot \|(N\tilde{N}^{-1})^{-1}\| \leq \frac{1+C\epsilon}{1-C\epsilon}$.
Inequality (27) follows from $\kappa(A\tilde{N}^{-1}) = \kappa(AN^{-1}N\tilde{N}^{-1}) \leq \kappa(AN^{-1}) \cdot \kappa(N\tilde{N}^{-1})$.

□

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 1 for the additive algorithm, independently of the number of subdomains. 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 $\tilde{N}$, but the results in Section 6 show that the conclusions also hold in case $\tilde{N}$ 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 for example [5] and many of the papers on *iterative substructuring* methods in [24, 14, 15, 25, 33]. The acceleration procedure used with accurate solution of subdomain problems is GMRES applied to the interface equations (19) and is described in detail in [11, 10]. This section describes the procedure used with inaccurate subdomain solution.

The GCR [23] method for solving $Ax = f$ can be easily adapted to cope with variable preconditioners. Because of its simplicity and for completeness, we describe the GCR method here. GCR 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 $As_i = v_i$. 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,

$$(f - Ax_k, v_i) = 0. \tag{28}$$

Since $Ax_k \in V_k$ we have

$$Ax_k = \sum_{j=1\ldots k} \lambda_j v_j \tag{29}$$

and by substituting (29) into (28) we get $\lambda_i = (f, v_i)$ so that

$$Ax_k = \sum_{i=1\ldots k} (f, v_i) v_i. \tag{30}$$

Since $As_i = v_i$, we have

$$Ax_k = \sum_{i=1\ldots k} (f, v_i) As_i = A \sum_{i=1\ldots k} (f, v_i) s_i \tag{31}$$

so that $x_k = \sum_{i=1\ldots k} (f, v_i) s_i$. 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.

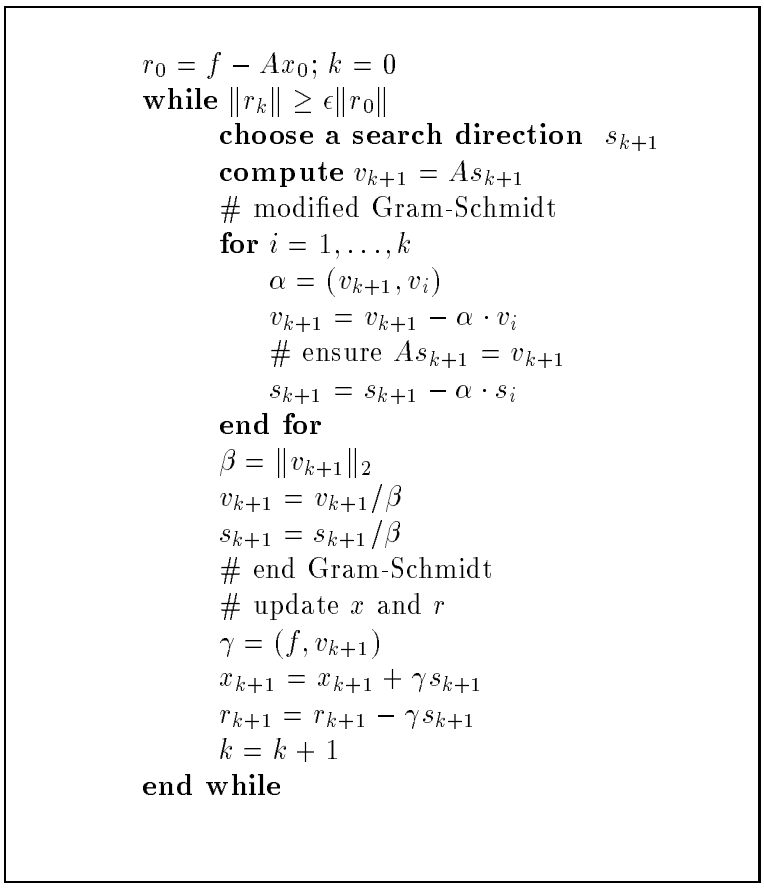


Figure 4: The GCR algorithm with general search directions without restart and with a relative stopping criterion [46]

For the special case of the search direction $s_{k+1} = r_k$, we obtain the classical GCR algorithm, which is equivalent to GMRES [41]. 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 3/2 times the number of floating point operations for orthogonalization. However, GCR can be combined with truncation strategies, for instance the Jackson & Robinson [31] strategy, whereas GMRES can only be restarted. Because of this, truncated GCR may converge faster than GMRES, see for example Section 6.2. Furthermore, restarted GCR can be optimized [49], which makes GCR just as efficient as GMRES. Both optimized restarted GCR and truncated

GCR will be considered in our numerical experiments.

Recent developments have led to a more flexible GMRES algorithm which allows more general search directions, so called FGMRES [40]. The FGMRES method is used in [6] to investigate the Neumann-Dirichlet method with inexact subdomain solution. The emphasis in [6] 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 the present paper, we use $s_{k+1} = \tilde{N}^{-1} r_k$, which corresponds to a single iteration of (21) with initial guess $u^m = 0$. The case of multiple iterations of (21) 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 [46] 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) [27] 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 for example [21, 32, 18, 19]. The present paper also investigates the multiplicative version of the RIBLU(α) postconditioner. The GMRES acceleration procedure may be applied with RIBLU(α), which is equivalent to GCR acceleration in this case.

The stopping criterion for accurate solution of subdomain problems differs from that for inaccurate solution. With accurate solution, the stopping criterion is based on the preconditioned residual $r = Q^T N^{-1} f - Q^T N^{-1} A Q v$ 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. 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 [9]. With inaccurate subdomain solution, 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 for instance, the cleaning of surfaces by water jets, vacuum cleaners and 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 [28, 37]. The problem also requires large computing times, about 7 hours 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 multi-block and single-block grids. Decompositions into more blocks are obtained by further

decomposing the two blocks into subblocks.

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

$$\mathrm{Re} = \frac{a u^\star}{\nu} \tag{32}$$

with

$$u^\star = \sqrt{\frac{\tau_0}{\rho}} \tag{33}$$

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 $\mathrm{Re} = 1 - 5$. Our results are given for $\mathrm{Re} = 2$. In the computation we have used $L = H = 10$. The boundary conditions are as follows:

- **ABGFIBC:** $u = 0$, $v = 0$
- **AE:** $u = \frac{\tau_0}{\mu} \cdot y$, $v = 0$
- **DC:** $\sigma_{xx} = 0$, $v = 0$
- **ED:** $\sigma_{xy} = \tau_0$, $v = 0$

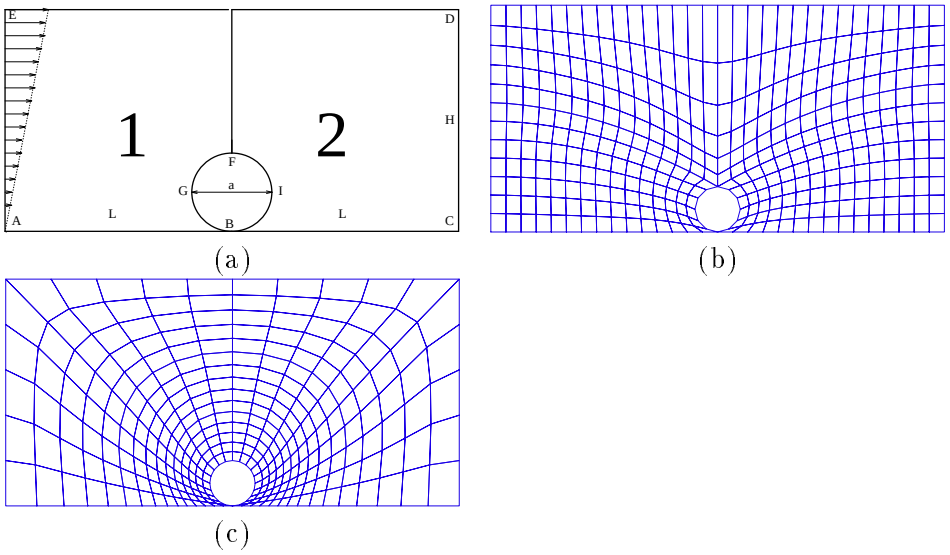


Figure 5: (a) Geometry and decomposition of the domain. (b) multi-block grid (c) single-block grid

The stationary solution is computed using the implicit Euler time integration scheme with start time $t = 0$, end time $t = 10$ and with time step 0.02. The time measurements in the next section are given for only the first 10 time steps to avoid excessive computing times. Figure 6 shows the streamlines for the stationary solution. For more details on this computation, the reader is referred to [10].

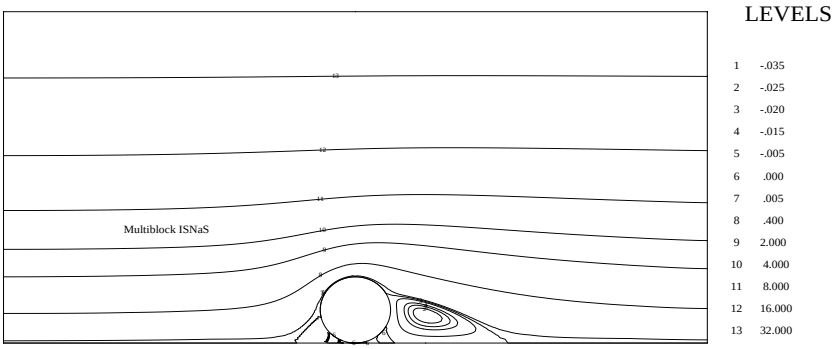


Figure 6: Streamlines of the stationary solution

6 Results

This section compares accurate with inaccurate solution of subdomain problems. We consider both the cylinder problem from the previous section and a Poisseuille flow in a rectangle $[0,1] \times [0,1]$. The global grid for the cylinder problem consists of 12240 grid cells. The single-block cylinder grid in Figure 5 consists of 10800 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 [50, 48] and a relative stopping criterion. For $\alpha = 0$ we get the standard ILU preconditioner [36] and for $\alpha = 1$ we get the Modified ILU preconditioner [27]. RILU(α) [1] lies in between these two. With RILUD(α) we mean 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). As a short-hand, we will use RILU(α) to mean RILUD(0.95) whenever the momentum, and RILU(0.95) whenever the pressure equations are involved. The subdomain solution accuracy is varied. As a special case the subdomain solution is approximated by means of the inverse of the RILU(α) [20, 50, 48] preconditioner, thereby omitting GMRES for subdomain solution.

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. GMRES always uses a restart after 20 iterations. With GCR, we investigate both optimized restart, denoted by GCR (restart), and truncation, denoted by GCR (trunc). Iteration counts and computing times are given in the tables in the form *time(iteration count)*. The iteration counts and times are summed over all time steps taken (10 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 solution on the number of iterations and total computing time. Section 6.2 compares single-block solution time with multi-block 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 1 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 multi-block 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 solution of subdomain problems, method II uses GCR for the outer loop and uses subdomain solution (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.

	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 1: Results with varying accuracy of subdomain solution for the cylinder problem, multiplicative algorithm

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 (here approximately a factor two). This is in accordance with Theorem 2.

The use of the RIBLU(α) postconditioner (method III) leads to small 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 [9], 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 number iterations with respect to subdomain RILU(α) for α close to 1.0. This increase is not present with $\alpha = 0$, see [9] and Section 6.3. The use of optimized restarted GCR instead of Jackson & Robinson truncation gives only a small reduction in computing time. For the momentum equations the total number of iterations is the same which is because the number of iterations per time step is below 20: the dimension of the Krylov space.

6.2 Single domain versus multi domain

One of the main reasons for investigating inaccurate solution of subdomain problems is to reduce the excessive computing times observed in the multi-block incompressible Navier-Stokes solver [10], and to bring them closer to single-block solution time. This also gives better prospects for parallel computing.

Table 2 lists the number of iterations and computing times for single-block solution of the Poisseuille flow on an 80×80 grid and the single-block cylinder grid with 10800 cells from Figure 5. The results are given for GMRES subdomain solution using RILU(α) postconditioning. Table 3 shows a comparison of single-block solution and multi-block solution of the

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

Table 2: Single-block solution using GMRES with RILU(α) postconditioning

momentum equations for different decompositions of the domain. Table 4 shows a comparison of single-block solution and multi-block solution 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 4), the maximum dimension of the Krylov space had to be increased to 40 to obtain convergence within 200 iterations/time step of the pressure equation.

Comparing Tables 2 and 3, we see that for small numbers of subdomains, computing time for the momentum equations can be reduced to below that of single-block solution for 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 further on. 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 for the Poisseuille flow (0.1) which increases the diagonal of the momentum matrix considerably and improves convergence.

Poisseuille flow				
		decomposition		
	ϵ	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	RIBLUD(0.95) post + GCR (trunc)	51.5(241)	78.8(281)	92.5(295)
	RIBLUD(0.95) post + GCR (restart)	42.8(241)	65.8(281)	86.4(295)

Cylinder problem				
		no. of blocks		
	ϵ	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	RIBLUD(0.95) post + GCR (trunc)	28.4(85)	30.8(86)	30.8(86)
	RIBLUD(0.95) post + GCR (restart)	26.7(85)	29.1(86)	29.0(86)

Table 3: Results with various decompositions into subdomains for the momentum equations, multiplicative algorithm.

Comparing Tables 2 and 4, we see that for small numbers of subdomains, the computing time with method II with inaccurate subdomain solution is still a factor $2-3$ larger than with single block solution. Also, for the Poisseuille flow in Table 4 inaccurate subdomain solution does not always provide a speedup. Method III leads to growing iteration count 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 much worse with respect to single domain RILU(α) than for $\alpha = 0.0$, see Section 6.3. We also see that the Jackson & Robinson truncation strategy is quite effective in reducing iteration count compared to 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 the following. For larger numbers of blocks the subdomains are smaller and therefore superlinear convergence of the subdomain GMRES solver can occur earlier. For example, in 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 6 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.

Poisseuille flow				
		decomposition		
	ϵ	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				
		no. of blocks		
	ϵ	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(339)
	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)

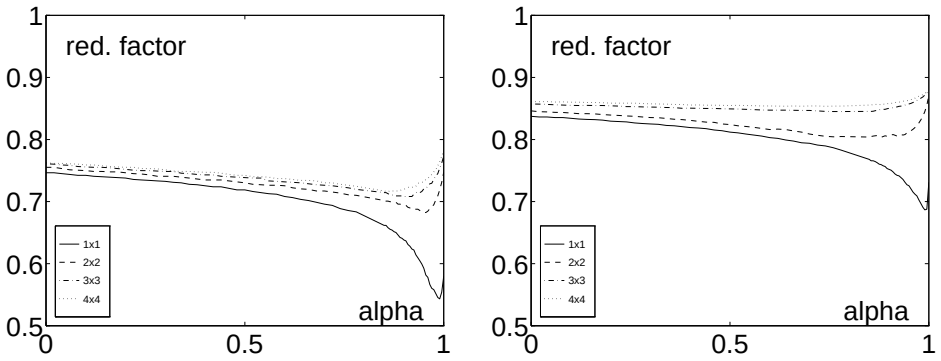
Table 4: Results with various decompositions into subdomains for the pressure equations, multiplicative algorithm.

The results in Table 3 and Table 4 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 $\epsilon = 10^{-1}$ does not increase as the domain is decomposed into more subdomains.

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 confirms the observation in [9] that with standard ILU preconditioning (RILU(0.0)), 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 multi-domain problems has a much lower influence on convergence than for single-domain problems. Also, the optimal value of α is lower for multi-domain problems.



(a) (b)

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

6.4 Prospects for parallel implementation

In [7, 8] parallelization of domain decomposition for the incompressible Navier-Stokes equations using accurate solution of subdomain problems is investigated. The method performs well on a cluster of workstations. The reason is that with accurate solution of subdomain problems the parallelization is rather coarse grained. Furthermore, the reduction to a system of interface equations (19) 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 5 shows a comparison between the multiplicative and additive algorithms. We see that the penalty of going from the multiplicative to the additive algorithm is between $1.5 - 2$ for method II, which is more than for method III.

	Multiplicative		*Additive*	
ϵ	*momentum*	*pressure*	*momentum*	*pressure*
10^{-4}	53.0(47)	283.2(339)	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)+GCR (trunc)	30.8(86)	183.0(811)	37.0(101)	213.2(998)
RIBLU(α+GCR (restart)	29.0(86)	225.2(1332)	34.5(101)	332.3(1965)

Table 5: Comparison between the multiplicative (Gauss-Seidel) and additive (Jacobi) algorithm for a decomposition into 8 blocks.

Table 5 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 & Robinson truncation is quite effective for method III compared to optimized restarted GCR.

The results in Table 5 show that for this problem, the multiplicative algorithm is more sensitive to the subdomain solution accuracy than the additive algorithm, probably because errors made in solving subproblems propagate to other subdomains within a single iteration.

7 Conclusions

It is possible to obtain significant reductions in computing time by inaccurate solution 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.

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.

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 actual reductions obtained by inaccurate subdomain solution are very much problem dependent. Significant reductions in computing time can be obtained for the momentum equations. For the pressure equations these reductions are typically much less.

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 solution but its effect on convergence of multi-block solution is much less. As shown in [9] and Figure 7, the number of iterations needed with multiplicative RIBLU(0) postconditioners is only slightly larger than with single-block solution. 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 especially to the disadvantage of the RIBLU(α) algorithms. These disadvantages of the current RIBLU(α) postconditioner prevent a reduction of computing time to almost that of single-block solution. The optimized restarted GCR method does not give significant reductions in computing time because of an increased number of iterations compared to Jackson & 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 solution, so that reduced computing time is almost certain. Only for very inaccurate subdomain solution, for instance when the RIBLU(α) postconditioner is used, we get a significant increase in the number of iterations and therefore communication.

Inaccurate solution of subdomain problems combined with GCR acceleration removes the restriction inherent in GMRES solution of interface equations (19) that subdomain problems

should be solved accurately (enough). The GCR based algorithm is therefore in general more reliable than the GMRES algorithm for solving interface equations.

References

[1] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numerische Mathematik*, 48:479–498, 1986.

[2] M.J. Berger. On conservation at grid interfaces. *SIAM Journal of Numerical Analysis*, 24:967–984, 1987.

[3] 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.

[4] P. Bjørstad and M.D. Skogen. Domain decomposition algorithms of Schwarz type, designed for massively parallel computers. 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 362–375, SIAM, Philadelphia, 1992.

[5] 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.

[6] Christoph Börgers. The Neumann-Dirichlet domain decomposition method with inexact solvers on the subdomains. *Numer. Math.*, 55:123–136, 1989.

[7] 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.

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

[9] E. Brakkee, C. Vuik, and P. Wesseling. An investigation of Schwarz domain decomposition using accurate and inaccurate solution of subdomains. Report 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, Submitted to Numerical Linear Algebra with Applications.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] A.J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Maksymillian Dryja and Olof B. Widlund. Some recent results on Schwarz type domain decomposition algorithms. In A. Quarteroni, J. Periaux, Yu.A. Kuznetsov, and O.B. Widlund, editors, *Proc. of the Sixth International Symposium on Domain Decomposition methods in Science and Engineering*, pages 53–61, AMS, Providence, 1992.

[23] 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.

[24] 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.

[25] 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.

[26] William D. Gropp and Barry F. Smith. Experiences with domain decomposition in three dimensions: Overlapping Schwarz methods. In A. Quarteroni, J. Periaux, Yu.A. Kuznetsov, and O.B. Widlund, editors, *Proc. of the Sixth International Symposium on Domain Decomposition methods in Science and Engineering*, pages 323–333, AMS, Providence, 1992.

[27] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[28] D. Hall. Measurements of the mean force on a particle near a boundary in turbulent flow. *J. Fluid Mechanics*, 187:451–466, 1988.

[29] F.H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids*, 8:2182–2189, 1965.

[30] W.D. Henshaw and G. Chesshire. Multigrid on composite meshes. *SIAM J. Sci. Stat. Comp.*, 8:914–923, 1987.

[31] 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.

[32] 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.

[33] 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.

[34] P.L. Lions. On the Schwarz alternating method, I. In R. Glowinski, G.H. Golub, G.A. Meurant, and J. Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42, SIAM, Philadelphia, 1988.

[35] Jan Mandel and Steve McCormick. Iterative solution of elliptic equations with refinement: The two-level case. In Tony F. Chan, Roland Glowinski, Jacques Périaux, and Olof B. Widlund, editors, *Proc. of the Second International Symposium on Domain Decomposition methods*, pages 81–92, SIAM, Philadelphia, 1989.

[36] 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. Comp.*, 31:148–162, 1977.

[37] A. M. Mollinger. *Particle entrainment: Measuring the fluctuating lift force*. PhD dissertation, Delft University of Technology, 1994.

[38] 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.

[39] 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.

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

[41] 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.

[42] H.A. Schwarz. Über einige Abbildungsaufgaben. *Journal für Reine und Angewandte Mathematik*, 70:105–120, 1869.

[43] 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 coordinates. *International Journal for Numerical Methods in Fluids*, 15:411–426, 1992.

[44] John C. Strikwerda and Carl D. Scarbnick. A domain decomposition method for incompressible flow. *SIAM J. Sci. Comput.*, 14:49–67, 1993.

[45] Wei Pai Tang. Generalized Schwarz splittings. *SIAM J. Sci. Stat. Comput.*, 13:573–595, 1992.

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

[47] J.J.I.M. Van Kan. A second-order accurate pressure correction method for viscous incompressible flow. *SIAM J. Sci. Stat. Comp.*, 7:870–891, 1986.

[48] C. Vuik. Fast iterative solvers for the discretized incompressible Navier-Stokes equations. Reports of the Faculty of Technical Mathematics and Informatics 93–98, Delft University of Technology, Delft, 1993. Available from anonymous ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1993/DUT-TWI-93-98.ps.gz. To appear in Int. J. Num. Meth. in Fluids.

[49] 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. Available from anonymous ftp://ftp.twi.tudelft.nl/TWI/publications/tech-reports/1993/DUT-TWI-93-10.ps.gz.

[50] 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.

[51] 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.

[52] O.B. Widlund. Some Schwarz methods for symmetric and nonsymmetric elliptic problems. 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*, SIAM, Philadelphia, 1992.

[53] J.A. Wright and W. Shyy. A pressure-based composite grid method for the Navier-Stokes equations. *Journal of Computational Physics*, 107:225–238, 1993.

[54] M. Zijlema, A. Segal, and P. Wesseling. Finite volume computation of incompressible turbulent flows in general coordinates on staggered grids. *Int. J. of Num. Meth. in Fluids*, 20:621–640, 1995.

[55] M. Zijlema, A. Segal, and P. Wesseling. Invariant discretization of the k-ε model in general coordinates for prediction of turbulent flows in complicated geometries. *Computers and Fluids*, 24:209–225, 1995.