ELSEVIER

Future Generation Computer Systems 17 (2001) 933–940

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.nl/locate/future

Coarse grid acceleration of a parallel block preconditioner

C. Vuik*, J. Frank[1]

Faculty of Information Technology and Systems, Department of Applied Mathematical Analysis, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract

A block preconditioner is considered in a parallel computing environment. This preconditioner has good parallel properties, however, the convergence deteriorates when the number of blocks increases. Two different techniques are studied to accelerate the convergence: overlapping at the interfaces and using a coarse grid correction. It appears that the latter technique is indeed scalable, so the wall clock time is constant when the number of blocks increases. Furthermore, the method is easily added to an existing solution code.

Keywords: Parallel Krylov subspace methods; Block preconditioner; Overlapping subdomains; Coarse grid correction

1. Introduction

Domain decomposition arises naturally in computational fluid dynamics applications on structured grids: complicated geometries are broken down into (topologically) rectangular regions and discretized in general coordinates, see e.g. [28], applying domain decomposition to iteratively arrive at the solution on the global domain. This approach provides easy exploitation of parallel computing resources, and additionally offers a solution to memory limitation problems.

In this paper, we present a parallel implementation of a Krylov accelerated block Gauss–Jacobi method for the DeFT Navier–Stokes solver. This research is a continuation of our work presented in [2,3,9,25,27]. For an overview of the literature of related parallel methods we refer to [9,27]. We report results for a Poisson problem on a square domain, which is representative of the pressure system which must be solved for the pressure correction method used in DeFT.

The main parallel operations required in Krylov subspace methods are distributed matrix–vector multiplications, vector updates, inner products, and preconditioner-vector multiplications. For many problems, the matrix–vector multiplications require only nearest neighbor communications, and are very efficient. Vector updates are also easy to parallelize. Inner products require global communication, so one has to be careful in their parallel implementation. This aspect of the Krylov subspace solver has been addressed in [9]. The parallelization of non-overlapping block preconditioner operations is also trivial. However, the convergence behavior of the preconditioner deteriorates considerably, when the number of blocks increases (compare [9]).

In this paper, we consider the acceleration of parallel block preconditioned Krylov subspace methods by overlapping and deflation. The parallel implementation is based on MPI subroutines [11]. The details of the block preconditioner are given in Section 2. In Section 3, overlapping of the subdomains is defined, which can be used to enhance the convergence

* Corresponding author.
E-mail address: c.vuik@math.tudelft.nl (C. Vuik).
[1] Present address: CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.

0167-739X/01/$ – see front matter
PII: S0167-739X(01)00035-8

of the block preconditioner. Coarse grid correction, presented in Section 4, is another promising technique to accelerate the block preconditioned method. Section 5 contains numerical experiments illustrating the convergence of the various parallel iterative methods.

2. Block preconditioning and Krylov subspace methods

2.1. The block Jacobi preconditioner

We consider an elliptic partial differential equation discretized using a cell-centered finite difference method on a computational domain Ω. Let the domain be the union of M non-overlapping subdomains Ω_m, $m = 1, \ldots, M$. Discretization results in a sparse linear system $Au = f$, with $u, f \in \mathbb{R}^N$. When the unknowns in a subdomain are grouped together one gets the block system

$$\begin{bmatrix} A_{11} & \cdots & A_{1M} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MM} \end{bmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_M \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_M \end{pmatrix}. \tag{1}$$

In this system, the diagonal blocks A_{mm} express coupling among the unknowns defined on Ω_m, whereas the off-diagonal blocks A_{mn}, $m \neq n$ represent coupling across subdomain boundaries. The only non-zero off-diagonal blocks are those corresponding to neighboring subdomains.

In order to solve system (1) with a Krylov subspace method, we use the block Jacobi preconditioner

$$K = \begin{bmatrix} A_{11} & & \\ & \ddots & \\ & & A_{MM} \end{bmatrix}.$$

When this preconditioner is used, systems of the form $Kv = r$ have to be solved. Since there is no overlap, the diagonal blocks $A_{mm}v_m = r_m$, $m = 1, \ldots, M$, can be solved in parallel. In our method, these systems are solved by an iterative method. Since the number of inner iterations may vary in each outer iteration, the effective preconditioner is non-linear and varies in each outer iteration.

We use RILU preconditioned GMRES [1,17,24] to solve the subdomain problems within a fixed tolerance. Additionally, a blockwise application of the RILU preconditioner is used.

2.2. The Krylov subspace methods

In this paper, we consider general linear systems so that the coefficient matrix may be symmetric or non-symmetric. For a symmetric matrix we use a parallel version of the preconditioned conjugate gradient method [6,18]. For this method, the preconditioner should be the same in every outer iteration. This means that only two choices for the preconditioner can be used: a block RILU preconditioner or solving the subdomain problems with a small tolerance. In the latter choice the preconditioner is close to K^{-1}.In our application, pressure correction, the pressure system resembles a discretized Poisson equation; however, the coefficient matrix may be non-symmetric. For the non-symmetric case we use the GCR method [8,22]. This is a Krylov subspace method which allows a variable preconditioner. Using the conclusions of [9] we choose the following orthogonalization methods: reorthogonalized classical Gram–Schmidt when the subdomain size is small, otherwise we take the modified Gram–Schmidt method.

3. Overlapping of the subdomains

It is well known that the convergence of an overlapping block preconditioner is nearly independent of the subdomain grid size when the physical overlap region is constant (see [7,16,19,20]).

To describe the overlapping block preconditioner we define the subdomains $\Omega_m^* \subset \Omega$. The domain Ω_m^* consists of Ω_m and n_{over} neighboring grid points (see Fig. 1). The matrix corresponding to this subdomain is denoted by A_{mm}^*. Application of the preconditioner goes as follows: given r compute v using the steps

1. r_m^* is the restriction of r to Ω_m^*,
2. solve $A_{mm}^* v_m^* = r_m^*$ in parallel,
3. form v_m, which is the restriction of v_m^* to Ω_m.

A related method is presented by Cai and co workers [4,5]. A drawback of overlapping subdomains is that the amount of work increases proportionally to n_{over}. Furthermore, it is not so easy to implement this approach on top of an existing software package.

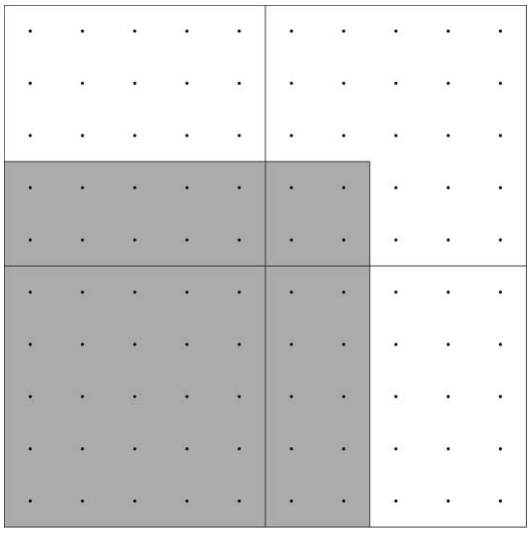

Fig. 1. The shaded region is subdomain Ω_1^* for $n_{\text{over}} = 2$.

4. Coarse grid correction

We present the coarse grid correction for a general system $Au = f$. In our implementation, we use deflated Krylov methods as defined in [10,26] (see also [12–15]). Let P and Q be given by

$$P = I - AZ(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}},$$
$$Q = I - Z(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}}A,$$

where Z is a suitable subspace of dimension $N \times M$. Note that for a symmetric matrix A, operator Q is equal to P^{T}. To solve the system $Au = f$ using deflation, note that u can be written as $u = (I - Q)u + Qu$ and that $(I - Q)u = Z(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}}Au = Z(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}}f$ can be computed immediately. In light of the identity $AQ = PA$, we can solve the deflated system

$$PA\tilde{u} = Pf \tag{2}$$

for $\tilde{u}$ and pre-multiplying the result with Q.

Deflation can be combined with preconditioning. Suppose K is a suitable preconditioner of A, then (2) can be replaced by: solve $\tilde{u}$ from $K^{-1}PA\tilde{u} = K^{-1}Pf$, and form $Q\tilde{u}$, or solve $\tilde{v}$ from $PAK^{-1}\tilde{v} = Pf$, and form $QK^{-1}\tilde{v}$. Both systems can be solved by one's favorite Krylov subspace solver, such as: GMRES [17], GCR [8,22], Bi-CGSTAB [21], etc.

For the coarse grid correction we choose the vectors z_m as follows:

$$z_m(i) = 1,\ \mathbf{x}_i \in \Omega_m, \qquad z_m(i) = 0,\ \mathbf{x}_i \notin \Omega_m. \tag{3}$$

We are able to give a sharp upperbound for the effective condition number of the deflated matrix, used with and without classical preconditioning [10]. This bound provides direction in choosing a proper decomposition into subdomains and a proper choice of classical preconditioner. If grid refinement is done keeping the subdomain resolutions fixed, the condition number can be shown to be independent of the number of subdomains.

To specify the extra costs of our coarse grid acceleration for a 2D problem, we consider a square domain, which is subdivided into M subdomains, where $\sqrt{M}$ is an integer number. The grid on each subdomain consists of $n \times n$ grid points, so $N = n^2M$. First the vectors Az_m are computed in parallel. The vectors z_m and Az_m are stored in memory which takes $N(2 + 4/n)$ memory positions. Then a band LU decomposition of the matrix $Z^{\mathrm{T}}AZ$ is computed on processor 1. This part has to be done only once.

In each iteration the extra work for the coarse grid correction is the calculation of the product

$$Pw = I - AZ(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}}w.$$

The part $\tilde{w} = Z^{\mathrm{T}}w$ costs n^2 flops on each processor, since only vector z_m is non-zero on the mth subdomain. The results are sent to processor 1, so M messages containing one floating point number are sent. On processor 1 the vector $\tilde{e}$ is computed from $Z^{\mathrm{T}}AZ\tilde{e} = \tilde{w}$ which takes $2M^{3/2}$ flops. The result is sent to all processors. Finally, the vector updates $Pw = w - AZ\tilde{e}$ are performed, which cost $2n^2 + 16n$ flops on each processor.

Summarizing, we need one gather-broadcast communication in which a set of M distributed floating point numbers are gathered from the participating processors and after some adaptations the whole set is returned to each processor. The maximal extra amount of work (on processor 1) is equal to $3n^2+16n+2M^{3/2}$ flops. This is less than the work to perform two vector updates.

5. Numerical experiments

5.1. Block preconditioner results

As a test example, we consider a Poisson problem, discretized with the cell-centered finite volume method on a square domain. We do not exploit the symmetry of the Poisson matrix in these experiments. The domain is composed of a $\sqrt{M} \times \sqrt{M}$ array of subdomains, each with an $n \times n$ grid. With $h = \Delta x = \Delta y = 1.0/(n\sqrt{M})$, the discretization is

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j}.$$

The right-hand side function is $f_{i,j} = f(ih, jh)$, where $f(x, y) = -32(x(1-x) + y(1-y))$. Homogeneous Dirichlet boundary conditions $u = 0$ are defined on $\partial\Omega$, implemented by adding a row of ghost cells around the domain, and enforcing the condition, e.g., $u_{0,j} = -u_{1,j}$ on boundaries. This ghost cell scheme allows natural implementation of the block preconditioner as well.

For the tests of this section, GCR is restarted after 30 iterations, and modified Gram–Schmidt was used as the orthogonalization method for all computations. The solution was computed to a fixed tolerance of 10^{-6}. The subdomain approximations will be denoted as follows:

- GMR6 = GMRES with a tolerance of 10^{-6} (preconditioned with RILU);
- GMR2 = GMRES with a tolerance of 10^{-2} (preconditioned with RILU);
- GMR1 = GMRES with a tolerance of 10^{-1} (preconditioned with RILU);
- RILU = one application of an RILU preconditioner.

We compare the results for a fixed problem size on the 300×300 grid using 4, 9, 16 and 25 subdomains, without overlapping of subdomains and coarse grid acceleration. In Table 1, the wall clock times on the Cray T3E are given together with the number of outer iterations (in parentheses). Note that for all preconditioners the number of outer iterations increases when the number of blocks grow. This implies that the parallel efficiency decreases when one uses more processors. In the following sections, we present two different approaches to diminish this drawback. The fastest solutions in each case are obtained with the least accurate subdomain approximation — namely, the block RILU preconditioner. Therefore, we use this choice in our timing measurements in Section 5.5.

Table 1
Wall clock times in seconds on the Cray T3E and number of outer iterations in parentheses

	M			
	4	9	16	25
GMR6	685 (78)	178 (83)	143 (145)	79 (168)
GMR2	167 (86)	102 (118)	63 (168)	37 (192)
GMR1	222 (139)	118 (225)	66 (287)	39 (303)
RILU	65 (341)	26 (291)	22 (439)	15 (437)

5.2. Block preconditioner and overlap

We consider a Poisson problem on a square domain with Dirichlet boundary conditions and a constant right-hand side function. The problem is discretized by cell-centered finite differences. We consider overlap of 0, 1 and 2 grid points and use A_{mm}^{-1} in the block preconditioner. Table 2 gives the number of iterations necessary to reduce the initial residual by a factor 10^6 using a decomposition into 3×3 blocks with subgrid dimensions given in the table. Note that the number of iterations is constant along the diagonals. This agrees with domain decomposition theory that the number of iterations is independent of the subdomain grid size when the physical overlap remains the same (see [7,16,19,20]).

In the second experiment, we take a 5×5 grid per subdomain. The results for various number of blocks are given in Fig. 2. Note that without overlap the number of iterations increases considerably, whereas the increase is much smaller when two grid points

Table 2
Iterations for various grid sizes

Grid size	Overlap		
	0	1	2
5×5	10	8	7
10×10	14	9	8
20×20	19	13	10
40×40	26	18	14

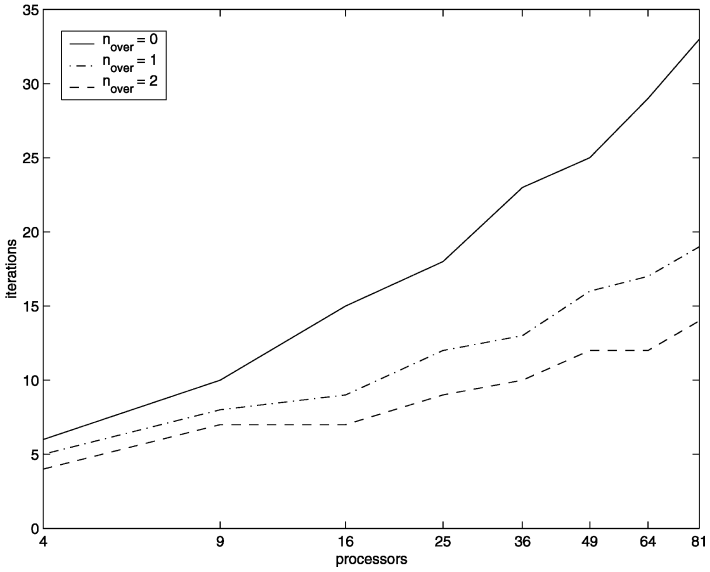

Fig. 2. Without coarse grid correction.

are overlapped. The large overlap (two grid points on a 5×5 grid) that has been used in this test is not affordable for real problems. In Section 5.5, we present results with large subdomain grids without overlap.

5.3. Coarse grid correction

We do the same experiments using the coarse grid correction (see Fig. 3). The subdomain grid size is 5×5. Initially, we see some increase of the number

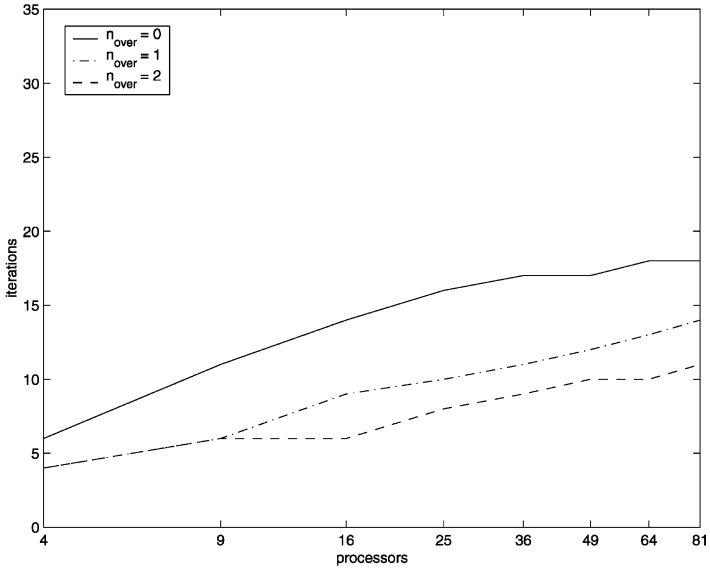

Fig. 3. With coarse grid correction.

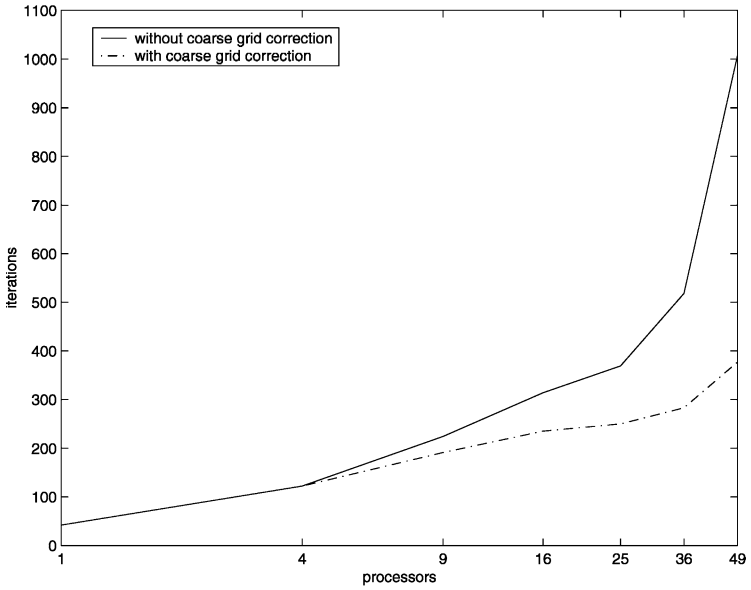

Fig. 4. Scalability for a non-symmetric problem, subdomain grid 50×50.

of iterations; however, for more than 16 blocks the increase levels off. This phenomenon is independent of the amount of overlap. The same conclusion holds when block RILU is used instead of fully solving the subdomain problems.

5.4. A non-symmetric example

We also illustrate the convergence of the coarse grid correction for a convection-dominated problem

$$\nabla \cdot (\mathbf{a}(x, y)\, u(x, y)) - \Delta u(x, y) = f$$

on $(0, 1) \times (0, 1)$ with recirculating wind field $a_1(x, y) = -80xy(1-x)$, $a_2(x, y) = 80xy(1-y)$ and boundary conditions $u(x, 0) \equiv u(y, 0) \equiv u(x, 1) \equiv 0$, $u_x(1, y) = 0$. The resulting system is solved using GCR truncated to a subspace of 20 vectors by dropping the vector most nearly orthogonal to the current search direction [23]. Classical preconditioning in the form of RILU(0.975) is incorporated. Fig. 4 compares the required number of GCR iterations as the number of subdomains is increased keeping the subdomain resolution fixed at $n = 50$. Although the number of iterations is not bounded using coarse grid correction, it grows much slowly than without coarse grid correction.

5.5. Timing results of coarse grid correction

Finally, we present some timing results on the Cray T3E for a problem on a 480×480 grid. The results are given in Table 3. In this experiment, we use GCR with the block RILU preconditioner combined with coarse grid correction. Note that the number of iterations decreases when the number of blocks increases. This leads to an efficiency larger than 1. The decrease in iterations is partly due to the improved

Table 3
Speedup of the iterative method using a 480×480 grid

M	Iterations	Time	Speedup	Efficiency
1	485	710	–	–
4	322	120	5	1.2
9	352	59	12	1.3
16	379	36	20	1.2
25	317	20	36	1.4
36	410	18	39	1.1
64	318	8	89	1.4

approximation of the RILU preconditioner for smaller subdomains. On the other hand, when the number of blocks increases more small eigenvalues are projected to zero which also accelerates the convergence (see [10]). We expect that there is some optimal value for the number of subdomains, because at the extreme limit there is only one point per subdomain and the coarse grid problem is identical to the original problem, so there is no speedup at all.

6. Conclusions

From the experiments presented in this paper, we conclude that the overlapping of the subdomains makes the parallel iterative method more or less independent of the subdomain grid size. A drawback is that overlapping is not so easy to implement on top of an existing software package.

Examples of coarse grid correction are given for symmetric and non-symmetric coefficient matrices. Experiments show that there is only a slow increase of the number of iterations when the subdomain grid size is constant and the number of subdomains increases. For a fixed global grid, it appears that the number of iterations decreases when the number of processors increases. This leads to efficiencies larger than 1. So we conclude that coarse grid correction is a very efficient technique to accelerate parallel block preconditioners. Finally, coarse grid correction implemented by deflation can easily be used in combination with existing software.

Acknowledgements

The authors thank HPαC for providing computing facilities on the Cray T3E.

References

[1] O. Axelsson, G. Lindskog, On the eigenvalue distribution of a class of preconditioning methods, Numer. Math. 48 (1986) 479–498.

[2] E. Brakkee, A. Segal, C.G.M. Kassels, A parallel domain decomposition algorithm for the incompressible Navier–Stokes equations, Simulat. Pract. Theory 3 (1995) 185–205.

[3] E. Brakkee, C. Vuik, P. Wesseling, Domain decomposition for the incompressible Navier–Stokes equations: solving subdomain problems accurately and inaccurately, Int. J. Numer. Meth. Fluids 26 (1998) 1217–1237.

[4] X.-C. Cai, C. Farhat, M. Sarkis, A minimum overlap restricted additive Schwarz preconditioner and applications in 3D flow simulations, in: J. Mandel, C. Farhat, X.-C. Cai (Eds.), Proceedings of the Tenth International Conference on Domain Decomposition Methods for Partial Differential Equations, AMS, Providence, RI, 1998, pp. 479–485.

[5] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (1999) 792–797.

[6] E. de Sturler, H.A. van der Vorst, Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers, Appl. Numer. Math. 18 (1995) 441–459.

[7] E. de Sturler, Incomplete block LU preconditioners on slightly overlapping subdomains for a massively parallel computer, Appl. Numer. Math. 19 (1995) 129–146.

[8] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, SIAM J. Numer. Anal. 20 (1983) 345–357.

[9] J. Frank, C. Vuik, Parallel implementation of a multiblock method with approximate subdomain solution, Appl. Numer. Math. 30 (1999) 403–423.

[10] J. Frank, C. Vuik, On the construction of deflation-based preconditioners, MAS-R 0009, CWI, Amsterdam, SIAM J. Sci. Comput. (2000), in press. http://ta.twi.tudelft.nl/nw/users/vuik/MAS-R0009.pdf.

[11] W. Gropp, E. Lusk, A. Skjellum, Using MPI, Portable Programming with the Message-passing Interface, 2nd Edition, Scientific and Engineering Computation Series, MIT Press, Cambridge, MA, 1999.

[12] C.B. Jenssen, P.Å. Weinerfelt, Coarse grid correction scheme for implicit multiblock Euler calculations, AIAA J. 33 (10) (1995) 1816–1821.

[13] L. Mansfield, On the conjugate gradient solution of the Schur complement system obtained from domain decomposition, SIAM J. Numer. Anal. 27 (6) (1990) 1612–1620.

[14] L. Mansfield, Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers, SIAM J. Sci. Stat. Comput. 12 (6) (1991) 1314–1323.

[15] R.A. Nicolaides, Deflation of conjugate gradients with applications to boundary value problems, SIAM J. Numer. Anal. 24 (2) (1987) 355–365.

[16] G. Radicati, Y. Robert, Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor, Parallel Comput. 11 (1989) 223–239.

[17] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.

[18] M.K. Seager, Parallelizing conjugate gradient for the Cray X-MP, Parallel Comput. 3 (1986) 35–47.

[19] K.H. Tan, Local coupling in domain decomposition, Ph.D. Thesis, University Utrecht, Utrecht, 1995.

[20] W.P. Tang, Generalized Schwarz splittings, SIAM J. Sci. Stat. Comput. 13 (1992) 573–595.

[21] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (1992) 631–644.
[22] H.A. van der Vorst, C. Vuik, GMRESR: a family of nested GMRES methods, Numer. Lin. Alg. Appl. 1 (1994) 369–386.
[23] C. Vuik, Further experiences with GMRESR, Supercomputer 55 (1993) 13–27.
[24] C. Vuik, Fast iterative solvers for the discretized incompressible Navier–Stokes equations, Int. J. Numer. Meth. Fluids 22 (1996) 195–210.
[25] C. Vuik, J. Frank, A parallel implementation of the block preconditioned GCR method, in: P. Sloot, M. Bubak, A. Hoekstra, B. Hertzberger (Eds.), Proceedings of the Seventh International Conference on High-Performance Computing and Networking (HPCN), Europe 1999, Amsterdam, The Netherlands, April 12–14, 1999, Springer, Berlin, Lecture Notes in Computer Science 1593, pp. 1052–1060.
[26] C. Vuik, A. Segal, J.A. Meijerink, An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients, J. Comput. Phys. 152 (1999) 385–403.
[27] C. Vuik, R.R.P. van Nooyen, P. Wesseling, Parallelism in ILU-preconditioned GMRES, Parallel Comput. 24 (1998) 1927–1946.
[28] P. Wesseling, A. Segal, C.G.M. Kassels, Computing flows on general three-dimensional nonsmooth staggered grids, J. Comput. Phys. 149 (1999) 333–362.

C. Vuik earned his Master's degree in Applied Mathematics from the Delft University of Technology in 1982. After a short stay at Philips Research Laboratories, he obtained his PhD at Utrecht University on the solution of moving boundary problems in 1988. In this year, he joined the Numerical Analysis group at the Delft University of Technology. His current interests are parallel iterative methods which are applied to linear systems originating from discretized partial differential equations.

J. Frank was born in Hutchinson, KS, USA, in 1970. He earned Bachelor of Science and Master of Science degrees in aerospace engineering from the University of Kansas in 1992 and 1994, respectively. Thereafter, he spent one-and-a-half-years as a visiting researcher at the CWI in Amsterdam, The Netherlands, before beginning his PhD research in numerical analysis at Delft University of Technology. After finishing his PhD in 2000, he has been working as a postdoctoral researcher at the CWI.