DELFT UNIVERSITY OF TECHNOLOGY

REPORT 00-05

PARALLEL GMRES AND DOMAIN DECOMPOSITION

K. DEKKER

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2000

Copyright © 2000 by Department of Applied Mathematical Analysis, Delft, The Netherlands.

No part of the Journal may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission from Department of Applied Mathematical Analysis, Delft University of Technology, The Netherlands.

Parallel GMRES and Domain Decomposition

K. Dekker*

Abstract

Solution of large linear systems encountered in computational fluid dynamics often leads to some form of domain decomposition, especially when it is desired to use parallel machines. To solve such problems we introduce a partitioned modification of GMRES, which has the property of minimising the the norm of the residual vector over a Krylov subspace of larger dimension than the one in GMRES. We prove that the new method converges faster than GMRES, if the subdomain problems are solved exactly. Moreover, less communication is required in the computation of inner products in a modified Gram-Schmidt orthogonalisation process, which makes the method suitable for parallel computing. The additional computational work is negligible, as a resulting least squares can be solved efficiently using Givens rotations. Numerical experiments for two fundamental test problems show that the new method requires about 30% less iterations than GMRES.

Keywords: Domain decomposition; Krylov subspaces; Parallel GMRES methods; Orthogonalisation methods; Givens rotations

1 Introduction.

Domain decomposition arises naturally in computational fluid dynamics applications on structured grids: complicated geometries are broken down into (topologically) rectangular regions and discretised, see e.g. [12, 16], and by solving subproblems on these regions one arrives 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 his thesis [2] Brakkee describes a serial implementation of a nonoverlapping, one-level additive Schwarz method, which is accelarated by the Krylov subspace method GMRES [11]. On a distributed memory platform, however, this approach has a drawback, because the modified Gram-Schmidt process in GMRES requires the global communication of many separate inner products. As the other main operations, preconditioning and distributed matrix-vector multiplication, usually require only nearest neighbour communication, and hence may be efficient on

*Faculty of Mathematics and Informatics, Delft University of Technology, Mekelweg 4, 2628CD Delft, The Netherlands; E-mail: K.Dekker@math.tudelft.nl

a parallel system, much effort has focused on the efficient parallelisation of Krylov subspace methods. Several authors have tried to overlap inner product communication with computation [4], others increase the number of inner products that can be computed with a single communication [1, 9, 4], or apply the classical Gram-Schmidt method twice [8, 7]. However, these approaches usually increase the amount of computation or diminish the numerical stability of the orthogonalisation process.

In the present paper we propose a partitioned version of GMRES, called P-GMRES, in which the solution space is constructed from two orthonormal Krylov subspaces. As independent modified Gram-Schmidt processes can be performed to obtain orthonormal bases for these subspaces, the amount of communication is substantially reduced compared with GMRES. We will show that the approximate solution in P-GMRES minimises the norm of the residual vector over the solution space, which has dimension twice as large as the Krylov subspace in GMRES. Consequently, faster convergence can be expected. The approximate solution is obtained from the least squares solution of a block system, composed of two Hessenberg matrices and two identity matrices, which system can be solved efficiently using Givens rotations.

In Section 2 we briefly review the Schwarz domain decomposition method restricted to two subdomains. In Section 3 we recall some properties of GMRES and introduce the P-GMRES method. We show in Section 4 that the minimisation problem in P-GMRES can be solved by a QR-decomposition using Givens rotations. The computation and communication costs of the domain decomposition method, combined with GMRES or P-GMRES, are discussed in Section 5. In Section 6 we compare the convergence behaviour of both methods applied to a Poisson problem and an advection-diffusion equation.

2 Schwarz domain decomposition

Consider a large system of linear equations,

$$Au = b \tag{1}$$

which e.g. originates from the semi-discretisation of an advection-diffusion equation on a domain Ω. Hence, the coefficient matrix A is large and sparse, and an iterative solution method may be appropriate. In a domain decomposition approach, the domain Ω is divided into several subdomains $\Omega_1, \cdots, \Omega_p$. Grouping together into blocks those unknowns which share a common subdomain, produces a block system

$$\begin{bmatrix} A_{11} & \dots & A_{1p} \\ \vdots & \ddots & \vdots \\ A_{p1} & \dots & A_{pp} \end{bmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_p \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix}, \tag{2}$$

where u_j denotes the vector of variables defined on Ω_j. This system may be solved in an iterative way by solving each of the subsystems separately for u_j, while taking the remaining vectors from the previous iteration:

$$A_{jj}u_j^{(i+1)} = b_j - \sum_{k\neq j} A_{jk}u_k^{(i)}, \quad j = 1, \cdots, p.$$

This so-called additive Schwarz domain decomposition approach (cf. [13]) is particularly suitable for a distributed computing environment, as all subsystems can be solved in parallel.

In [17, 18] it is shown that the large system (2) can be reduced to a much smaller system for the so-called interface variables. Each vector u_j is divided into two parts: the first part z_j contains the variables which appear in the subsystem for Ω_j only, whereas the remaining variables form the second part x_j. For the sake of simplicity we restrict ourselves to the case of two subdomains. Let us introduce the operators Q_i and P_i, which restrict the vector u_i to the vector of interface variables x_i and internal variables z_i, respectively. System (2) then can be reformulated as

$$\left[\begin{array}{cc|cc} & A_{11} & A_{12}Q_2^T & 0 \\ 0 & A_{21}Q_1^T & A_{22} & \end{array}\right] \begin{pmatrix} z_1 \\ x_1 \\ x_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{3}$$

A subdomain solution can now be regarded as a multiplication of (3) by the preconditioner (cf. [18])

$$K = \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{pmatrix},$$

yielding the preconditioned system

$$\begin{bmatrix} I & 0 & P_1A_{11}^{-1}A_{12}Q_2^T & 0 \\ 0 & I & Q_1A_{11}^{-1}A_{12}Q_2^T & 0 \\ 0 & Q_2A_{22}^{-1}A_{21}Q_1^T & I & 0 \\ 0 & P_2A_{22}^{-1}A_{21}Q_1^T & 0 & I \end{bmatrix} \begin{pmatrix} z_1 \\ x_1 \\ x_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} P_1A_{11}^{-1}b_1 \\ Q_1A_{11}^{-1}b_1 \\ Q_2A_{22}^{-1}b_2 \\ P_2A_{22}^{-1}b_2 \end{pmatrix}, \tag{4}$$

Obviously, we can rewrite (4) as a separate system for the interface vectors only

$$\begin{bmatrix} I & Q_1A_{11}^{-1}A_{12}Q_2^T \\ Q_2A_{22}^{-1}A_{21}Q_1^T & I \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} Q_1A_{11}^{-1}b_1 \\ Q_2A_{22}^{-1}b_2 \end{pmatrix}, \tag{5}$$

and a system for the internal variables,

$$\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} P_1A_{11}^{-1}(b_1 - A_{12}Q_2^Tx_2) \\ P_2A_{22}^{-1}(b_2 - A_{21}Q_1^Tx_1) \end{pmatrix}, \tag{6}$$

which can be solved after the interface vectors have been computed. In the next section we consider the solution of (4,5) by a Krylov subspace method.

3 Krylov subspace acceleration

In this section we present a new iterative method to solve the system (5) efficiently. We will show that this method requires at most the number of iterations of a GMRES-accelerator. In section 4 it will be shown that the computational costs per iteration are comparable, whereas the proposed method requires much less communication on a distributed platform.

First we describe a standard method to solve the systems (4) or (5). Both systems can be written as

$$Bu = f \tag{7}$$

for which a stationary iteration method reads

$$u_{k+1} = f + (I - B)u_k.$$

3.1 GMRES acceleration

It is usual to accelerate the stationary iteration by a Krylov subspace method, e.g. GMRES [11]. Let u_0 be the starting vector for the iteration, and $r_0 = f - Bu_0$ the initial residual. Then we construct approximations $u_k, k = 1, 2, \ldots,$ such that u_k is the member of the (shifted) Krylov subspace

$$u_k \in u_0 + K_k \equiv u_0 + \mathrm{Span}\{r_0, Br_0, \ldots, B^{k-1}r_0\} \tag{8}$$

which minimises the residual norm

$$\| f - Bu\|, \qquad u \in u_0 + K_k. \tag{9}$$

This method, applied to (5) is called a Krylov-Schwarz domain decomposition [2]. In [11] it is shown that the approximation u_k can be obtained efficiently using the GMRES algorithm. We briefly recall some properties of GMRES, in order to facilitate the description of the new method in the next section.

Let $v_1, \ldots, v_k$ denote an orthonormal basis for the Krylov subspace K_k. This basis can be formed by a standard modified Gram-Schmidt process. In each iteration step a new vector Bv_i is calculated, and orthogonalised against the previous basis vectors. This leads to the relation

$$AV_k = V_{k+1}H_k \tag{10}$$

where

$$V_k = [v_1, \cdots, v_k]$$

and H_k is a $(k+1) \times k$ Hessenberg matrix.

Now, in order to solve the least squares problem (9), we set $u = u_0 + V_k y$, and consider the function

$$J(y) = \| f - B[u_0 + V_k y]\| = \|r_0 - BV_k y\|.$$

Using $\beta = \|r_0\|$, $r_0 = \beta v_1$, (10) and the orthonormality of V_{k+1}, this function can be rewritten as

$$J(y) = \|V_{k+1}[\beta e_1 - H_k y\| = \|\beta e_1 - H_k y\|. \tag{11}$$

$$
\begin{aligned}
&w = Bv_k,\\
&\textbf{for} \quad i = 1,\ldots,k\\
&\qquad h_{i,k} = w^T v_i\\
&\qquad w = w - h_{i,k} v_i\\
&\textbf{end}\\
&h_{k+1,k} = \|w\|\\
&v_{k+1} = w/h_{k+1,k}
\end{aligned}
$$

Figure 1: The modified Gram-Schmidt algorithm

Hence, the solution of the least squares problem (9) is given by

$$u_k = u_0 + V_k y_k,$$

where y_k minimizes the function $J(y)$, for $y \in \mathbb{R}^k$. Due to the special form of the Hessenberg matrix H_k, the least squares system (11) can be solved very efficiently, using plane rotations (see [11]).

Although the GMRES-acceleration is very efficient on sequential machines, the implementation on a parallel platform has disadvantages. The main drawback here lies in the fact that the preconditioning is very well parallelisable (the two subdomain solutions can be computed independently), but the orthogonalisation of the basis vectors for the Krylov subspace requires excessive global communication, and therefore do not scale. In the kth iteration step of the GMRES method we have to perform the computations for the modified Gram-Schmidt process (cf. [6]) as given in Fig. 1. Hence, we must compute $k+1$ inner products of vectors which are distributed, After the subdomain solution, over the processors. Parallel computation of the inner products then requires $k+1$ communication steps, in which partial sums are exchanged between the processors. In case inter processor communication is slow compared to the computation speed, the speed up might deteriorate substantially.

In the literature several approaches have been suggested to alleviate this problem (cf. [6]). One strategy is to compute all inner products simultaneously, and use the classical Gram-Schmidt process. This approach, however, is less stable, and leads to loss of orthogonality of the basis vectors [3, 8]. Then, the orthogonalisation process should be performed twice, increasing the amount of computation [7]. Another approach is to generate a number of successive Krylov vectors and orthogonalise these as a block [1, 4, 9]. The algorithm we propose in the next section contains two independent orthogonalisation processes which can be distributed over the processors, thereby reducing the amount of communication.

We conclude our description of the GMRES-acceleration by observing that application to (5) is cheaper than application to (4). In the first case the dimension of the solution space is equal to the number of interface variables, which is usually (much) less than the total number of variables. Hence, the modified Gram-Schmidt process requires much less operations for (5) than for (4). However, when we restrict ourselves to the interface system (5), we have to calculate the

complete solution, using (6), after convergence.

3.2 P-GMRES acceleration

In the partitioned method the solution of (7) is approximated by

$$u_k = u_0 + u_k^{(1)} + u_k^{(2)},$$

where $u_k^{(1)}$ and $u_k^{(2)}$ belong to two orthogonal subspaces $K_k^{(1)}$ and $K_k^{(2)}$. Recall that (7) can be written in the block form

$$\begin{bmatrix} I & B_{12} \\ B_{21} & I \end{bmatrix} \begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix} = \begin{pmatrix} f^{(1)} \\ f^{(2)} \end{pmatrix},$$

and that the initial residual can also be decomposed as

$$r_0 = \begin{pmatrix} r_0^{(1)} \\ r_0^{(2)} \end{pmatrix}. \tag{12}$$

Then, we define these subspaces recursively by

$$\begin{aligned} K_1^{(1)} &= \mathrm{Span}\{r_0^{(1)}\}, && (13a) \\ K_1^{(2)} &= \mathrm{Span}\{r_0^{(2)}\}, && (13b) \\ K_i^{(1)} &= K_{i-1}^{(1)} \oplus B_{12} K_{i-1}^{(2)}, \quad i = 2, \ldots && (13c) \\ K_i^{(2)} &= K_{i-1}^{(2)} \oplus B_{21} K_{i-1}^{(1)}, \quad i = 2, \ldots && (13d) \end{aligned}$$

Orthonormal bases for these subspaces can be constructed recursively, like in GMRES. Let $v_1^{(i)}, \ldots, v_k^{(i)}$ denote an orthonormal basis for the Krylov subspace $K_k^{(i)}, i = 1, 2$. In each iteration step a new vector $B_{ij} v_k^{(j)}, j \neq i$ is calculated, and orthogonalised against the previous basis vectors. This leads to the relations

$$\begin{aligned} B_{12} V_k^{(2)} &= V_{k+1}^{(1)} H_k^{(1)}, \\ B_{21} V_k^{(1)} &= V_{k+1}^{(2)} H_k^{(2)}, \end{aligned} \tag{14}$$

where

$$V_k^{(i)} = \left[v_1^{(i)}, \ldots, v_k^{(i)} \right]$$

and $H_k^{(i)}, i = 1, 2$ are $(k+1) \times k$ Hessenberg matrices.

Now the new iterate u_k will be the member of the (shifted) subspace

$$u_0 + K_k^{(1)} \oplus K_k^{(2)}$$

which minimises the residual norm

$$\|f - Bu\|, \qquad u \in u_0 + K_k^{(1)} \oplus K_k^{(2)} \tag{15}$$

If we set $u = u_0 + V_k^{(1)} y_1 + V_k^{(2)} y_2$, we can view the norm to be minimised as the following function of y_1, y_2:

$$J(y_1, y_2) = \|f - B[u_0 + V_k^{(1)} y_1 + V_k^{(2)} y_2]\| = \|r_0 - B[V_k^{(1)} y_1 + V_k^{(2)} y_2]\|,$$

which is written, using the block decomposition of B, as

$$\left\| \begin{pmatrix} r_0^{(1)} \\ r_0^{(2)} \end{pmatrix} - \begin{bmatrix} I & B_{12} \\ B_{21} & I \end{bmatrix} \begin{pmatrix} V_k^{(1)} y_1 \\ V_k^{(2)} y_2 \end{pmatrix} \right\|.$$

Introducing $\beta_i = \|r_0^{(i)}\|$ for convenience, we obtain, using (14) and the orthonormality of $V_{k+1}^{(i)}, i = 1, 2$

$$\begin{aligned} J(y_1, y_2) &= \left\| \begin{pmatrix} \beta_1 V_{k+1}^{(1)} e_1 - V_k^{(1)} y_1 - V_{k+1}^{(1)} H_k^{(1)} y_2 \\ \beta_2 V_{k+1}^{(2)} e_1 - V_{k+1}^{(2)} H_k^{(2)} y_1 - V_k^{(2)} y_2 \end{pmatrix} \right\| = \\ &= \left\| \begin{pmatrix} \beta_1 e_1 \\ \beta_2 e_1 \end{pmatrix} - \begin{pmatrix} I_{k+1,k} & H_k^{(1)} \\ H_k^{(2)} & I_{k+1,k} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\| \end{aligned} \tag{16}$$

Here, $I_{k+1,k}$ denotes a $(k+1) \times k$ matrix, obtained from the identity matrix by adding a zero last row. Hence, the solution of the least squares problem is given by

$$u_k = u_0 + \begin{pmatrix} V_k^{(1)} y_k^{(1)} \\ V_k^{(2)} y_k^{(2)} \end{pmatrix}, \tag{17}$$

where $y_k^{(1)}, y_k^{(2)}$ minimises the function $J(y_1, y_2)$ defined by (16), for $y_1, y_2 \in \mathbb{R}^k$. This minimisation problem has dimension $(2(k+1) \times 2k)$, and can be solved efficiently, as described in Section 4.

In the partitioned method the dimension of the subspace increases by 2 in each iteration step. Hence, it is more expensive to solve the resulting least squares problem (16) then (11) in the GMRES method. On the other hand one might expect that the kth iterate in P-GMRES is a better approximation to the solution of (7) than the corresponding iterate in GMRES, as we search in a subspace of larger dimension. We confirm this expectation in the next section.

Finally, we observe that each iteration of P-GMRES can be implemented on two processors, almost without any communication. All inner products involved in the modified Gram-Schmidt process can be computed locally, and we only need communication for the exchange of the basis vectors $v_k^{(1)}$ and $v_k^{(2)}$. The kth iteration then consists of the computations given in Fig. 2.

Processor 1	Processor 2
$w_1 = B_{21} v_k^{(2)}$	$w_2 = B_{12} v_k^{(1)}$
for $i = 1, \ldots, k$	**for** $i = 1, \ldots, k$
$\quad h_{i,k}^{(1)} = w_1^T v_i^{(1)}$	$\quad h_{i,k}^{(2)} = w_2^T v_i^{(2)}$
$\quad w_1 = w_1 - h_{i,k}^{(1)} v_i^{(1)}$	$\quad w_2 = w_2 - h_{i,k}^{(2)} v_i^{(2)}$
end	**end**
$h_{k+1,k}^{(1)} = \|w_1\|$	$h_{k+1,k}^{(1)} = \|w_1\|$
$v_{k+1}^{(1)} = w_1 / h_{k+1,k}^{(1)}$	$v_{k+1}^{(2)} = w_2 / h_{k+1,k}^{(2)}$

Figure 2: Modified Gram-Schmidt processes in P-GMRES

3.3 Convergence properties of P-GMRES

Let the Krylov subspaces $K_k^{(1)}, K_k^{(2)}$ be defined by (13a- 13d), and the subspace K_k by (8). Then, from (12) we have

Lemma 1

$$K_1 \subset K_1^{(1)} \oplus K_1^{(2)}. \qquad \square$$

Next, we will show that a similar relation holds for the higher dimensional subspaces K_k. To that end we need

Proposition 1 *Let $x_1 \in K_k^{(1)}$, $x_2 \in K_k^{(2)}$. Define*

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{bmatrix} I & B_{12} \\ B_{21} & I \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Then $z_1 \in K_{k+1}^{(1)}$, $z_2 \in K_{k+1}^{(2)}$.

Proof: We have $z_1 = x_1 + B_{12} x_2$, where $x_1 \in K_k^{(1)}$ and $x_2 \in K_k^{(2)}$. Hence, z_1 is an element of $K_{k+1}^{(1)}$, by definition (13c). Similarly, it is proved that z_2 is an element of $K_{k+1}^{(2)}$. $\square$

Lemma 2

$$K_k \subset K_k^{(1)} \oplus K_k^{(2)}.$$

Proof: (by induction) Suppose the assertion holds for $k = m$. Let $y \in K_{m+1}$ be arbitrary, and decompose y as $y = u + Bx$, with $u, x \in K_m$. Then, by assumption,

$$u \in K_m^{(1)} \oplus K_m^{(2)}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, x_1 \in K_m^{(1)}, x_2 \in K_m^{(2)}.$$

From Proposition 1 it follows that $Bx \in K_{m+1}^{(1)} \oplus K_{m+1}^{(2)}$, and consequently $y \in K_{m+1}^{(1)} \oplus K_{m+1}^{(2)}$. $\square$

As a consequence, the norm of the residual vector in P-GMRES will be less or equal to the one in GMRES,

$$\min_{u-u_0 \in K_k^{(1)} \oplus K_k^{(2)}} \|f - Bu\| \leq \min_{u-u_0 \in K_k} \|f - Bu\|.$$

4 Efficient QR-decomposition

We now describe a few important details concerning the efficient solution of the minimisation problem (15). The approach is quite similar to the method used in GMRES [11], although more complicated because of the structure of the problem (16):

$$\min_{y_1, y_2} \left\| \begin{pmatrix} \beta_1 e_1 - y_1 - H_k^{(1)} y_2 \\ \beta_2 e_1 - H_k^{(2)} y_1 - y_2 \end{pmatrix} \right\|$$

A classical way of solving this problem is to factor the matrix

$$H_k = \begin{bmatrix} I_{k+1,k} & H_k^{(1)} \\ H_k^{(2)} & I_{k+1,k} \end{bmatrix}$$

into $Q_k R_k$ using plane rotations, where Q_k is orthogonal and R_k is a (permuted) upper triangular matrix. However, like in GMRES, it is desirable to update the factorisation of H_k progressively as each pair of columns appears, i.e. at every iteration step. This enables us to obtain the residual norm of the approximate solution without actually computing the solution u_k.

We now show in detail how such a factorisation can be carried out. Let $F(i, j, 0, 0, c, s)$ represent a Givens rotation which rotates the unit vectors e_i and e_j by the angle θ:

$$F(i, j, 0, 0, c, s) = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & c & & s & & & \\ & & & \ddots & & & & \\ & & -s & & c & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & I_{k+1} \end{bmatrix}$$

where $c = cos(\theta)$ and $s = sin(\theta)$. Such a rotation will transform the upper part of H_k. Let us similarly denote by $F(0, 0, i, j, c, s)$ a rotation which acts on the lower part of H_k, rotating the unit vectors e_{k+1+i} and e_{k+1+j}, and by $F(i, 0, j, 0, c, s)$ a transformation which rotates the unit vectors e_i and e_{k+1+j}.

Assume that rotations have been previously applied to H_{k-1} to produce

$$R_{k-1} = \begin{bmatrix} R_{k-1}^{11} & R_{k-1}^{12} \\ R_{k-1}^{21} & R_{k-1}^{22} \end{bmatrix}$$

where the R_{k-1}^{ij} are all upper triangular matrices of dimension $k \times (k-1)$, and R_{k-1}^{21} is strictly upper triangular. Hence, e.g., for $k = 4$

$$R_{k-1} = \left[\begin{array}{ccc|ccc} x & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ 0 & 0 & x & 0 & 0 & x \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & x & x & x & x & x \\ 0 & 0 & x & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The letter x stands for a (possibly) nonzero element. At the next step, two columns and rows of H_k appear, and are inserted in the above matrix, yielding

$$\tilde{R}_k = \left[\begin{array}{ccc|c|ccc|c} x & x & x & 0 & x & x & x & h_{1,k}^{(1)} \\ 0 & x & x & 0 & 0 & x & x & h_{2,k}^{(1)} \\ 0 & 0 & x & 0 & 0 & 0 & x & h_{3,k}^{(1)} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & h_{k,k}^{(1)} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{k+1,k}^{(1)} \\ \hline 0 & x & x & h_{1,k}^{(2)} & x & x & x & 0 \\ 0 & 0 & x & h_{2,k}^{(2)} & 0 & x & x & 0 \\ 0 & 0 & 0 & h_{3,k}^{(2)} & 0 & 0 & x & 0 \\ 0 & 0 & 0 & h_{k,k}^{(2)} & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & h_{k+1,k}^{(2)} & 0 & 0 & 0 & 0 \end{array}\right].$$

In order to obtain R_k we first premultiply the new columns by the previous rotations. Note that the updates of both columns are independent of each other, so they can be applied in parallel. Now, we obtain a $(2k+2) \times (2k)$ matrix of the form

$$\left[\begin{array}{ccc|c|ccc|c} x & x & x & x & x & x & x & x \\ 0 & x & x & x & 0 & x & x & x \\ 0 & 0 & x & x & 0 & 0 & x & x \\ 0 & 0 & 0 & g_3 & 0 & 0 & 0 & h_1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_2 \\ \hline 0 & x & x & x & x & x & x & x \\ 0 & 0 & x & x & 0 & x & x & x \\ 0 & 0 & 0 & x & 0 & 0 & x & x \\ 0 & 0 & 0 & g_1 & 0 & 0 & 0 & h_3 \\ \hline 0 & 0 & 0 & g_2 & 0 & 0 & 0 & 0 \end{array}\right].$$

The large $k \times (k-1)$ submatrices are nothing but R_{k-1}^{ij}, and h_2, g_2 stand for $h_{k+1,k}^{(1)}$, $h_{k+1,k}^{(2)}$, resp., which are uneffected by the previous rotations. The next step consists in eliminating

the elements g_2, g_1, 0 and h_2 in the positions $(2k+2,k)$, $(2k+1,k)$, $(2k+2,2k)$, $(k+1,2k)$, respectively. This is achieved by four rotations defined by ($\tilde{x}$ denotes an element changed by a previous rotation.)

$$F_k^{(1)} = F(0,0,k,k+1,g_1/(g_1^2+g_2^2)^{1/2},g_2/(g_1^2+g_2^2)^{1/2}), \tag{18a}$$

$$F_k^{(2)} = F(k,0,k,0,g_3/(\tilde{g_1}^2+g_3^2)^{1/2},\tilde{g_1}/(\tilde{g_1}^2+g_3^2)^{1/2}), \tag{18b}$$

$$F_k^{(3)} = F(0,0,k,k+1,\tilde{h_3}/(\tilde{h_3}^2+\tilde{0}^2)^{1/2},\tilde{0}/(\tilde{h_3}^2+\tilde{0}^2)^{1/2}), \tag{18c}$$

$$F_k^{(4)} = F(k+1,0,k,0,\tilde{h_3}/(\tilde{h_3}^2+h_2^2)^{1/2},-h_2/(\tilde{h_3}^2+h_2^2)^{1/2}). \tag{18d}$$

Note, that in the computation of the new rotations only 3 elements of each of the new columns are involved. After application of these four rotations to the new columns, we obtain the matrix R_k, which is of the appropriate form. Next, the successive rotations should also be applied to the right hand side $[\beta_1 e_1;\beta_2 e_1]$. This can also be achieved in a step by step fashion, applying the four newly computed rotations in each iteration step to the modified right hand side.

Thus, after k steps of the above process, we have the following decomposition of H_k

$$Q_k H_k = R_k$$

where Q_k is $(2k+2)\times(2k+2)$, being the accumulated product of the $4k$ rotation matrices, while R_k is a (permuted) upper triangular matrix of dimension $(2k+2)\times 2k$, whose $(k+1)$st and last row are zero. Since Q_k is orthogonal, we have

$$\begin{aligned} J(y_1,y_2) &= \left\| \begin{pmatrix} \beta_1 e_1 \\ \beta_2 e_1 \end{pmatrix} - H_k \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\| = \\ &= \left\| Q_k \left[\begin{pmatrix} \beta_1 e_1 \\ \beta_2 e_1 \end{pmatrix} - H_k \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right] \right\| = \left\| f_k - R_k \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\|, \end{aligned} \tag{19}$$

where $f_k = Q_k[\beta_1 e_1;\beta_2 e_1]$ is the transformed right hand side. Since two rows of R_k are zero, the minimisation of (19) is achieved by solving the (permuted) upper triangular system which results from removing the zero rows from R_k and the corresponding components of f_k. This provides $y_k^{(1)}, y_k^{(2)}$ and the approximate solution u_k is then formed from (17).

Just like in the GMRES [11] it is possible to obtain the residual norm $\|f - Bu_k\|$ of the approximate solution u_k while performing the above factorisation, without explicitly computing u_k. From the definition of $J(y_1,y_2)$ it follows that the residual norm is nothing but $J(y_1,y_2)$, which, from (19), is in turn equal to $\|f_k - R_k[y_k^{(1)};y_k^{(2)}]$. But, by construction, this is equal to the norm of the two components of f_k corresponding to the zero rows of R_k. Hence we have

Proposition 2 *The residual norm of the approximate solution u_k is equal to the norm of the $(k+1)$st and $(2k+2)$nd component of f_k,*

$$\|r_k\| = \sqrt{\{f_k(k+1)\}^2 + \{f_k(2k+2)\}^2},$$

where f_k is obtained by premultiplying $[\beta_1 e_1;\beta_2 e_1]$ by the $4k$ successive rotations (18a-18d), which transform H_k into the (permuted) upper triangular matrix R_k. □

5 Computation and communication costs

From the previous description of P-GMRES it is clear that the method requires only few communication. However, P-GMRES is computationally more expensive than GMRES, as a least square problem of larger size has to be solved. In [11] it is claimed that the costs of solving the least square problem of size k in GMRES is negligible compared to the amount of work in the matrix-vector multiplication and in the Gram-Schmidt process, as the least squares problem is usually of much smaller dimension than the original system. Nevertheless, we will also take these computations into account, to demonstrate the effect of the increased size of the least squares problem. The total costs of computing the approximate solution, then, can be divided in four tasks, three of which are to be performed in each iteration step:

- The matrix-vector multiplication, involving the interface vectors, and the subsequent solution of the subdomain problems. Here, we first have to communicate the two interface vectors (both of dimension m). Then, all computations can be performed in parallel for the subdomains. Of course, the amount of computation depends on the size of the problem, and the structure of the matrix A.

- The computation of the othonormal basis vectors for the Krylov subspaces, using the modified Gram-Schmidt process. In the kth iteration of GMRES this amounts to k inner products and vector updates (of dimension $2m$), and one norm and scaling. These computations can be performed in parallel, although each inner product then requires the communication of a partial sum. In the P-GMRES method we have twice as many inner products and updates, but each for vectors of dimension m, so the number of floating point operations is exactly the same, and no communication is necessary.

- The factorisation of the matrix H_k. In P-GMRES, the update of the factorisation requires the application of $4k$ plane rotations on two newly added columns, to be performed in parallel, whereas we need only k rotations on one column in GMRES. Moreover, one or four new rotations have to be constructed and applied to the right hand side in GMRES, viz. P-GMRES. In the latter method we also need communication, if these calculations are to be performed in parallel.

Finally, upon convergence, we have to solve the least squares problem by back substitution with R_k, which has size $k \times k$ in case of GMRES, and size $2k \times 2k$ for P-GMRES. With the least squares solution y_k, resp. $y_k^{(1)}, y_k^{(2)}$, we can construct the approximate solution $u_0 + V_k y_k$, resp. $u_0 + [V_k^{(1)} y_k^{(1)}; V_k^{(2)} y_k^{(2)}]$. Using these interface vectors, we obtain the solution of (6) by an additional matrix-vector multiplication, followed by a parallel solution of the subdomain problems.

In Table 1 we compare the number of sequential and parallel floating point operations in the k-th iteration of GMRES and P-GMRES. We have restricted ourselves to two simple model problems, such that the matrix-vector multiplication and subdomain solves do not completely dominate the computation. In both cases we consider a Poisson problem on a square grid, with N unknowns and $2m$ interface variables. In the first problem, denoted by matvec-full, we simulate a ”larger” problem by performing all operations on full matrices, whereas in the second problem

	GMRES		P-GMRES	
	parallel	seq	parallel	seq
matvec-full	$N(N+4+2m)+2m$		$N(N+4+2m)$	
matvec-sparse	$N(3m+\frac{7}{2})+16m-5m^2$		$N(3m+\frac{7}{2})+14m-5m^2$	
Gram-Schmidt	$8km+10m$		$8km+10m$	
rotations		$10k$	$64(k-1)$	$4k+64$

Table 1: Number of parallel and sequential floating point operations in the kth iteration.

	GMRES		P-GMRES	
	parallel	seq	parallel	seq
LSQ solve		k^2+2k		$4k^2+4k$
Compute u_k	$4km$		$4km$	
matvec-full	$N(N+4+2m)+2m$		$N(N+4+2m)$	
matvec-sparse	$N(3m+\frac{7}{2})+16m-5m^2$		$N(3m+\frac{7}{2})+14m-5m^2$	

Table 2: Number of parallel and sequential floating point operations to obtain the complete solution after the kth iteration.

a sparse (banded) LU-factorisation is used for the solution of the subdomain problems. The number of operations to obtain the complete solution after the k-th iteration is given in Table 2.

From the tables above it is clear that most of the computations in GMRES and P-GMRES can be distributed over the processors. As we consider only the Schwarz method for two subdomains, we will restrict ourselves to two processors. In order to assess the time spent in communication, it is assumed that the time for a message of length l is given by (cf. [14])

$$T_{mes} = s_0 + s_1 l$$

In Table 3 the communication time required in the iterations of GMRES and P-GMRES is summarised. Note, that in case of parallel computation each processor has only part of the factorised matrix R_k at its disposal. In order to obtain the least square solution $y_k^{(1)}, y_k^{(2)}$, we have to distribute the remaining part of R_k. We do this after the iteration has converged. Alternatively, it is possible to communicate the two new columns of R_k in each iteration, after the rotations have been performed. Finally, it is necessary to distribute the approximate interface solutions, in

	GMRES	P-GMRES
matvec	$2s_0+2s_1m$	$2s_0+2s_1m$
Gram-Schmidt	$2(k+1)(s_0+s_1)$	
rotations		$2s_0+6s_1$

Table 3: Communication time in the kth iteration.

	GMRES	P-GMRES
LSQ solve		$2s_0 + 2k(k-1)s_1$
matvec	$2s_0 + 2s_1 m$	$2s_0 + 2s_1 m$

Table 4: Communication time to obtain the complete solution after the kth iteration.

Problem ($N = m^2$)	GMRES (full)	GMRES (interface)	P-GMRES
full, $m = 6$	45	27	30
full, $m = 10$	266	154	157
full, $m = 20$	3662	2130	2133
sparse, $m = 6$	24	13	16
sparse, $m = 10$	83	44	47
sparse, $m = 20$	507	306	309

Table 5: Number of operations for 10 iterations (in Kflops).

order to compute the solution of (6). In Table 4 we summarise the communication times required to obtain the complete solution.

The total number of floating point operations, as measured by MATLAB [10], for 10 iterations of GMRES and P-GMRES and the subsequent computation of the complete solution, are given in Table 5. For the sake of completeness we also present the number for GMRES applied to system (4), which method is obviously less efficient as the Gram-Schmidt process is applied to vectors of dimension N.

These results indicate that P-GMRES is computationally slightly more expensive than the GMRES method applied to the interface equations. The relative performance, however, depends not only on the complexity of the (preconditioned) matrix-vector multiplication and the size of the problem, but also on the number of iterations to be performed. To assess the influence of this factor we present the average amount of work per iteration, which depends on the total number of iterations, for a small problem ($m = 6$) in Fig. 3. The number of flops in the (preconditioned) matrix-vector multiplication is taken as unit of work. In the sparse case we can clearly observe that the computational overhead in P-GMRES increases faster than in GMRES. As the full matrix-vector multiplication is rather expensive, the influence of the overhead is less pronounced in the full case. Note, however, that both problems are unrealistically small, and that these results are presented here only to show the computational overhead in the parallel method. For more realistic problems the overhead becomes negligible, as Table 5 shows.

The average communication time per iteration is presented in Fig. 4. Here, we assume that the latency of the communication s_0 is equal to the time for the transmission of 4 double precision numbers (cf. [15], formula (1)). Again, the time necessary for communication in the matrix-vector multiplication, i.e. the time to communicate the interface variables (two vectors of length m), is taken as unit of time.

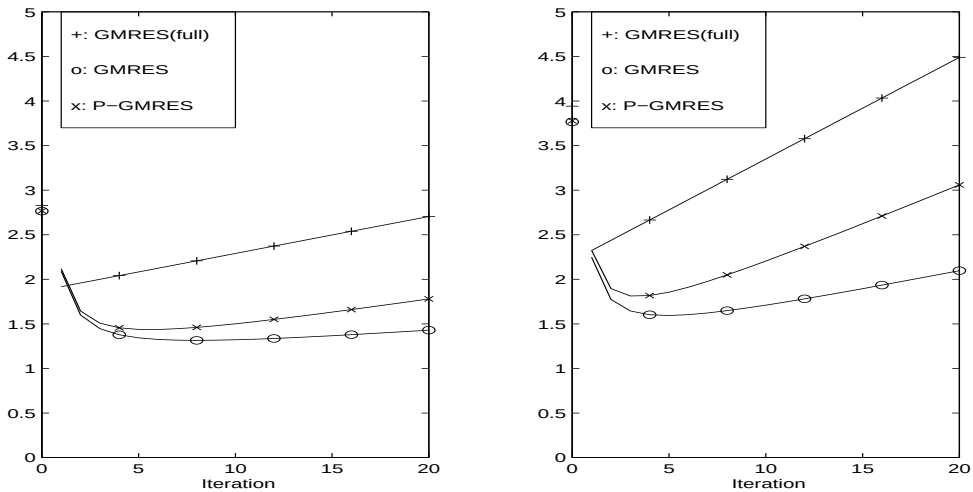

Figure 3: Average #flops per iteration, scaled by the #flops in matvec, using a full (left) and a sparse matrix-vector multiplication (for $it = 0$ the overhead in initialisation and computation of the final residual is given).

Finally, we compare the performance of P-GMRES with respect to GMRES, applied to the interface equations. We assume that in both methods the computations are distributed over 2 processors, whenever possible. We also assume that the cost of communication is given by

$$s_0 = \alpha t_{fl}$$

where t_{fl} is the time for one floating point operation. Of course, for small values of α the performance is completely determined by the computation, whereas the communication costs dominate for large values of α. In Fig. 5 we present the relative performance of P-GMRES for various values of α for the larger problem ($m = 20$). We note that even for the rather small value $\alpha = 100$ P-GMRES is more efficient than GMRES, whereas in case of more expensive communication the iterations in P-GMRES are almost twice as cheap as those in GMRES.

6 Numerical experiments

In this section we report a few numerical experiments in order to compare the convergence of P-GMRES with GMRES accelaration. The first problem is a Poisson equation with Dirichlet boundary conditions:

$$\begin{aligned} -u_{xx} - u_{yy} &= 0, \qquad (x,y) \in \Omega = [0,1]^2, \\ u(x,y) &= \begin{cases} 1, & y = 0 \vee (x = 1 \wedge y > 0.5), \\ 0, & y = 1 \vee x = 0 \vee (x = 1 \wedge y \leq 0.5). \end{cases} \end{aligned} \tag{20}$$

A uniform mesh with $(m+2) \times (m+2)$ gridpoints is used. Taking second order central differences, and dividing the domain Ω into two subdomains

$$\Omega = \Omega_1 + \Omega_2, \quad \Omega_1 = [0,1] \times [0,0.5], \quad \Omega_2 = [0,1] \times (0.5,1],$$

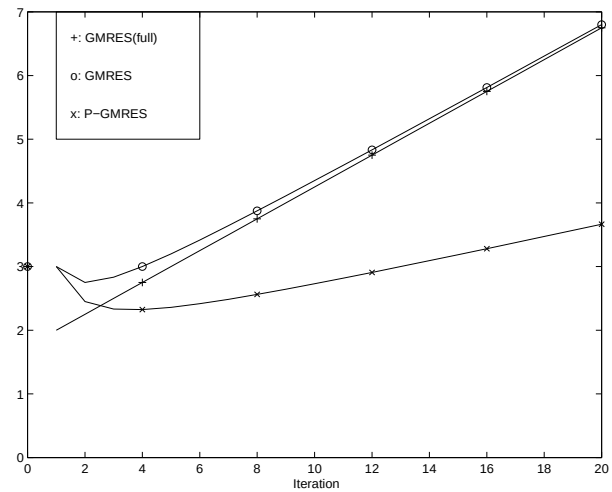

Figure 4: Average communication time per iteration, scaled by the communication time for the matrix-vector multiplication.

yields two subsystems of dimension $m^2/2$ each. In the Schwarz method these subsystems can be reduced to the smaller system (5) of dimension $2m$ for the $2m$ interface variables only. We solved this system for various values of m, using GMRES and P-GMRES acceleration. In Table 6 we list the number of iterations necessary to reduce the initial residue by a factor ε. In Fig. 6

	GMRES		P-GMRES	
m	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-6}$
6	6	10	4	6
10	8	12	6	8
20	10	17	7	12
40	13	23	10	16

Table 6: Number of iterations to reduce the initial residual by ε for (20), $m \times m$ grid.

the convergence behaviour of GMRES and P-GMRES is shown for the smallest grid ($m = 6$) and the largest grid ($m = 40$). The sudden drop of the residual norm after 6 iterations of P-GMRES for the smallest grid follows easily from the theory: after 6 iterations the subspace $K_k^{(1)} \oplus K_k^{(2)}$ has dimension 12 and thus equals the complete solution space. For the same reason we obtain the exact solution after 12 iterations of GMRES, too. The second figure shows that doubling the dimension of the subspace also leads to a faster convergence on the largest grid.

As a second problem we consider the 2D advection-diffusion equation (cf. [2]):

$$\begin{aligned} -u_{xx} - u_{yy} + vu_y &= 1, \qquad (x,y) \in \Omega = [-1,1]^2, \\ u(x,y) &= 1, \qquad y = -1 \vee x = -1, \\ u_y(x,y) &= 0, \qquad y = 1 \vee x = 1. \end{aligned} \tag{21}$$

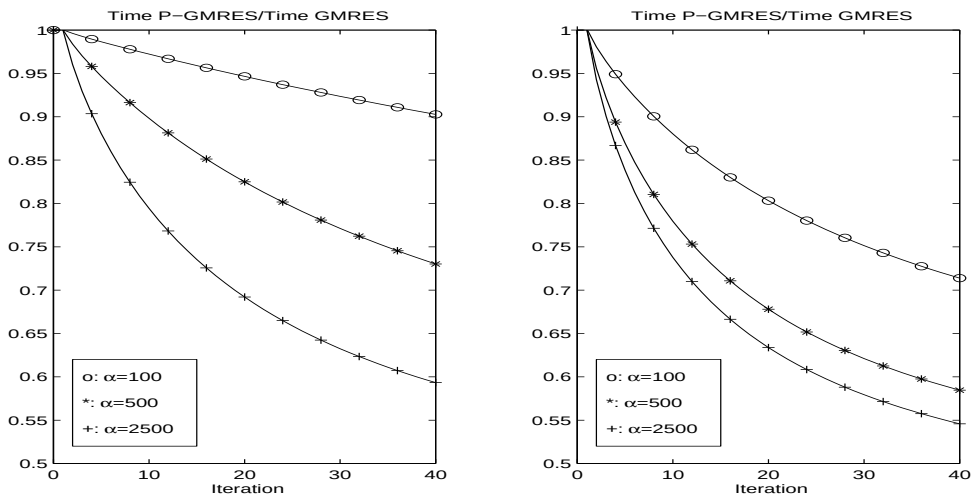

Figure 5: Relative performance of P-GMRES compared to GMRES, 20×20 grid, using full(left) and sparse matrix-vector multiplication.

A uniform mesh of 40×40 cells on Ω is used. Again we took second order central differences and decomposed Ω in two subdomains,

$$\Omega = \Omega_1 + \Omega_2, \quad \Omega_1 = [-1,1] \times [-1,0], \quad \Omega_2 = [-1,1] \times (0,1].$$

The flow magnitude v in the y-direction is given by the dimensionless mesh-Péclet number $p = vh$, which is varied in the tests. In Table 6 we list the average reduction factors after 10 iterations,

$$\rho = \{\|r_{10}\|/\|r_0\|\}^{0.1}$$

for $p = 0,1,3,5,10$. We did not include the case $p = 2$, because then the system is decoupled ($B_{12} = \mathbf{0}$), so the solution is obtained in 2 iterations of P-GMRES. Again, the partitioned method

Péclet	0	1	3	5	10
GMRES	0.59	0.29	0.18	0.22	0.21
P-GMRES	0.36	0.16	0.08	0.09	0.09

Table 7: Average reduction factors for different mesh-Péclet ranges in problem (21).

converges faster than GMRES applied to the interface equations. We also observe that the convergence rate of the partitioned method is comparable to the results in [2] for the Schwarz-Schwarz coupling. However, there a multiplicative domain decomposition method (block Gauss-Seidel) is used to solve (5), which is not parallelisable. We show the convergence behaviour of GMRES and P-GMRES for mesh-Péclet numbers $p = 1$ and $p = 10$ in Fig. 7.

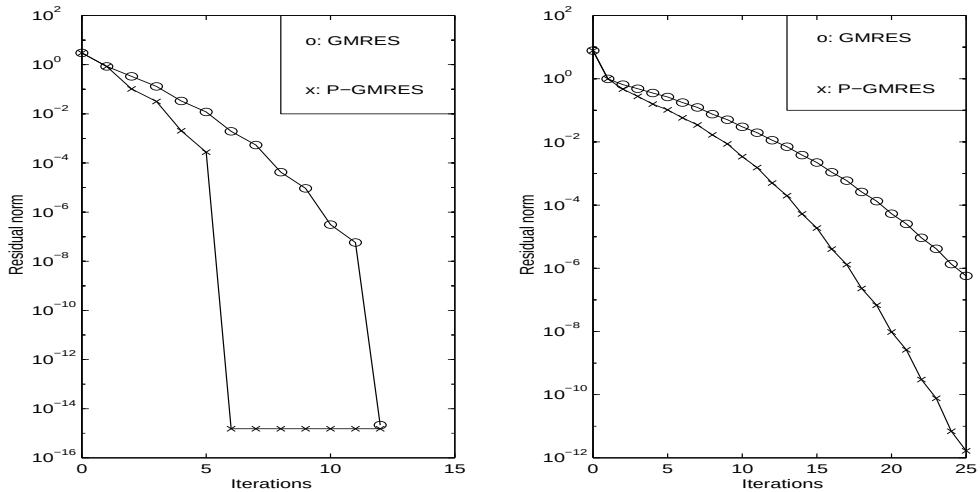

Figure 6: Convergence behaviour for problem (20), 6×6 (left) and 40×40 (right) grid

.

7 Conclusions

For applications which require domain decomposition we have introduced a partitioned GMRES-type method, called P-GMRES. The new method offers advantages compared to GMRES, both with respect to convergence and to communication costs. It is shown that P-GMRES does not require any communication for the inner products in the modified Gram-Schmidt orthogonalisation on two processors. Therefore, the communication time in a P-GMRES iteration might be considerably less than in GMRES as a performance model in Section 5 indicates.

The main additional computations in P-GMRES stem from solving a least squares problem of higher dimension than in GMRES. The solution can be obtained efficiently, using 4 Givens rotations in each iteration step, followed by a back substitution. Therefore, the difference in computation costs compared to GMRES will be negligible, if the subdomains are not too small; we observed a 2% increase for a problem consisting of 100 grid points.

We proved that P-GMRES converges as least as fast as GMRES, due to its higher dimensional solution space. In the numerical experiments the convergence behaviour of both methods are compared for a 2D Poisson problem and an advection-diffusion equation. In all tests P-GMRES converged considerably faster, e.g. 30% less iterations for the Poisson problem with 1600 grid points.

In the present paper we have assumed that the two subdomain problems in the additive Schwarz method are solved exactly, using an LU-decomposition. In a forthcoming paper [5] we show that the method is also applicable in the more general case, with many subdomains and inaccurate subdomain solvers.

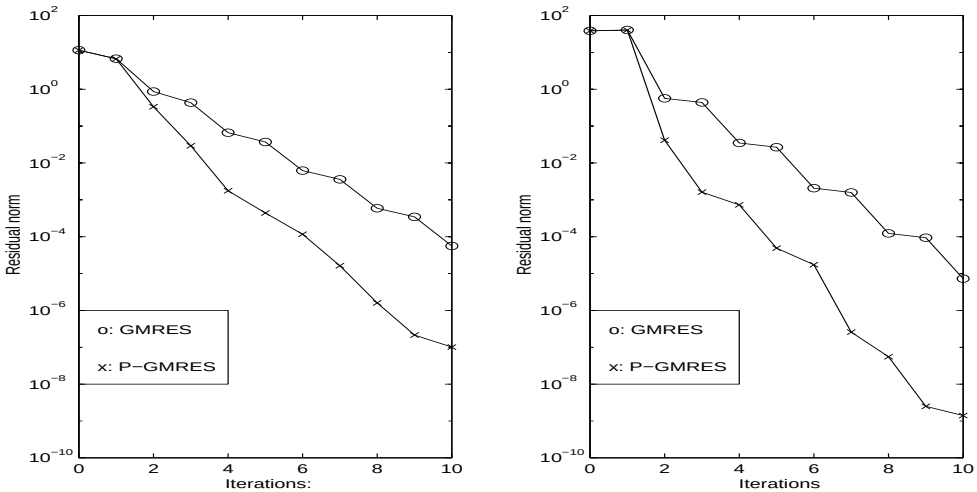

Figure 7: Convergence behaviour for problem (21), Péclet=1 (left) and Péclet=10(right).

References

[1] Z. Bai, D. Hu, and L. Reichel. A Newton-basis GMRES implementation. *IMA J. Numer. Anal.*, 14:563–581, 1994.

[2] E. Brakkee. *Domain decomposition for the incompressible Navier-Stokes equations*. PhD thesis, Delft University of Technology, Delft, The Netherlands, April 1996.

[3] E. F. D'Azevedo and C. H. Romine. Reducing communication costs in the conjugate gradient algorithm on distributed memory computers. Report ORNL/TM-12192, Oak Ridge National Lab., 1992.

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

[5] K. Dekker. Comparing GMRES and P-GMRES in domain decomposition with approximate subdomain solution. Report 00-02, Delft University of Technology, Delft, 2000.

[6] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H.A. van der Vorst. *Solving linear systems on vector and shared memory computers*. SIAM, Philadelphia, 1991.

[7] J. Frank and C. Vuik. Parallel implementation of a multiblock method with approximate subdomain solution. *Appl. Numer. Math.*, 30:403–423, 1999.

[8] W. Hoffmann. Iterative algorithms for Gram-Schmidt orthogonalization. *Computing*, 41:335–348, 1989.

[9] G. Li. A block variant of the GMRES method on massivily parallel processors. *Parallel Computing*, 23:1005–1019, 1997.

[10] C.B. Moler, J. Little, S. Bangert, and S. Kleiman. *ProMatlab user's guide*. MathWorks, Sherborn, 1987.

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

[12] A. Segal, P. Wesseling, J. van Kan, C.W. Oosterlee, and C. Kassels. Invariant discretization of the incompressible Navier-Stokes equations in boundary fitted co-ordinates. *Int. J. Num. Meth. Fluids*, 15:411–426, 1992.

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

[14] M. van Gijzen. Parallel iterative solution methods for linear finite element computations on the CRAY T3D. In *Proceedings of the 5th International Conference, HPCN Europe 1995, Amsterdam, The Netherlands, 1995*, Lecture Notes in Computer Science 919, Berlin, 1995. Springer.

[15] M. van Gijzen. GMRES-like methods on distributed memory computers. In S.D. Margenov and P.S. Vassilevski, editors, *Iterative methods in linear algebra I, Proceedings of the Second IMACS International Symposium on Iterative Methods in Linear Algebra, Blagoevgrad, Bulgaria, June 17-20, 1995*, IMACS series in computational and applied mathematics 3, pages 45–56, Piscataway, 1996. IMACS.

[16] P. Wesseling, A. Segal, C.G.M. Kassels, and H. Bijl. Computing flows on general two-dimensional nonsmooth staggered grids. *J. Eng. Math.*, 34:21–44, 1998.

[17] P. Wilders and E. Brakkee. Schwarz and Schur: a note on finite volume domain decomposition for advection-diffusion. Report 95-59, Delft University of Technology, Delft, 1995.

[18] P. Wilders and G. Fotia. One level Krylov-Schwarz domain decomposition for finite volume advection diffusion. Report 96-134, Delft University of Technology, Delft, 1996.