REPORT 00-02

COMPARING GMRES AND P-GMRES IN DOMAIN DECOMPOSITION WITH APPROXIMATE SUBDOMAIN SOLUTION

K. Dekker

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2000
Comparing GMRES and P-GMRES in Domain Decomposition with approximate subdomain solution

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. In this paper P-GMRES, a partitioned modification of GMRES, is applied to such problems. It is shown that P-GMRES converges faster than GMRES if the subdomains are solved exactly, and that P-GMRES requires less communication in the computation of the inner products. Also, approximate solutions for the subdomains by an inner preconditioned GMRES iteration are considered, in combination with restarted versions of GMRES and P-GMRES. We investigate the effect of the tolerance in the subdomain problems on the convergence of the outer iteration, and on the total amount of work in numerical experiments. It turns out that rather crude tolerances are allowed, and that a good strategy is to vary the tolerance for the subdomains in the course of the outer iteration.

Keywords: Domain decomposition; Parallel GMRES methods; Approximate subdomain solution; Orthogonalisation methods

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. [23, 30], 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.

Frank and Vuik [11] address the parallel implementation of a domain decomposition method for the DeFT Navier-Stokes solver described in [23]. Their paper is a continuation of work by Brakkee, summarised in [5] and presented in [6], where a serial implementation of nonoverlapping, one-level additive Schwarz method with approximate subdomain solution gave promising
results. In [11] the GCR method in combination with inaccurate subdomain solution is tested for a Poisson problem on a square domain, which is representative for the system to be solved for the pressure correction method used in DeFT. Our present goal is to evaluate the partitioned method, P-GMRES, described in [9], in combination with accurate and inaccurate subdomain solution on such a problem.

Theoretical results on approximate solution of subproblems for Schur complement domain decomposition methods are given by Börgers [4], Haase et al. [12, 13, 14, 20]. Tan [25] and Brakke [5] give theoretical results for nonoverlapping Schwarz iterations with approximate subdomain solvers.

In this paper we demonstrate for a nonoverlapping, additive Schwarz method that P-GMRES converges faster than GMRES, if the subdomains are solved exactly, and that restarted versions of both methods can be applied if the subproblems are solved with moderate accuracy. We show that the computational work of the methods is about the same per iteration, and that P-GMRES requires less communication, so it can be more efficiently parallelised. On the other hand, the applicability of P-GMRES is restricted to the class of problems for which a red-black colouring of subdomains exists such that no adjacent subdomains have the same colour.

In Section 2 we briefly review the relevant mathematics and present the GMRES and P-GMRES algorithms. Much effort has focused on the efficient parallelisation of Krylov subspace methods. The computation and communications of the many inner products often limit the attainable speedup on many processors. Therefore, authors have tried to overlap inner product communication with computation [8], or to increase the number of inner products that can be computed with a single communication [2, 18, 8]. Frank and Vuik [11] suggest to increase the amount of computation to reduce the number of communications in GCR. We show that P-GMRES requires less communication than GCR or GMRES, and that the communications can be easily overlapped with computation.

In Section 2.4 we address the solution of the subdomain problems. We give evidence that the restarted versions of GMRES and P-GMRES are applicable in combination with an approximate subdomain solution. Moreover, we show that a variable precision in the solution of these subproblems is likely to be most efficient, which is confirmed by experiments in Section 4.

A performance model for the orthogonalisations in (P-)GMRES, derived from [11], is presented in Section 3. Theoretical speedup ratios for P-GMRES on a workstation cluster and a Cray T3E, based on this model, are given. We also develop a model for the costs of various subdomain solvers which is used for the evaluation of the results in Section 4.

In Section 4 we compare the convergence rate of GMRES and P-GMRES with various multiblock preconditioners on the test problem from [11]. We also report results for P-GMRES with approximate subdomain solvers, in combination with several strategies for the tolerance in the inner iterations. Our results suggest that a variable precision, decreasing during the course of the outer iterations, is most efficient. According to the performance model, however, exact solution will be cheapest for the relatively small subdomains (less than 22500 grid points) considered in the tests.
2 Mathematical background

2.1 Nonoverlapping domain decomposition

We consider an (elliptic) partial differential equation discretised using a finite difference or finite volume method on a computational domain \( \Omega \). By a computational domain we mean a set of unknown values to be approximated, together with their locations in space. We suppose that \( \Omega \) is the union of \( M \) nonoverlapping subdomains \( \Omega_m, m = 1, \ldots, M \).

Discretisation of the PDE results in a sparse linear system

\[
Ax = b
\]

with \( x, b \in \mathbb{R}^N \). The structure of the matrix \( A \) is determined by the stencil of the discretisation. Even if there is no overlap between the subdomains, there is an inter-subdomain coupling due to the stencil. Grouping together into blocks those unknowns which share a common subdomain will permute the system (1) to produce a block system:

\[
\begin{bmatrix}
A_{11} & \cdots & A_{1M} \\
\vdots & \ddots & \vdots \\
A_{M1} & \cdots & A_{MM}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
\vdots \\
x_M
\end{bmatrix}
= \begin{bmatrix}
b_1 \\
\vdots \\
b_M
\end{bmatrix}
\]

(2)

Here, the diagonal blocks \( A_{mm} \) express the coupling among unknowns defined on a common subdomain \( \Omega_m \), whereas the off-diagonal blocks \( A_{mn}, m \neq n \), represent coupling across subdomain boundaries. The only nonzero off-diagonal blocks are those corresponding to neighbouring subdomains. Moreover, we will assume in the sequel that a red-black colouring of the subdomains exists, such that adjacent subdomains have a different colour, i.e. there holds

\[
A_{mn} = 0, m \neq n,
\]

if \( \Omega_m \) and \( \Omega_n \) have the same colour. This restriction, which often can be satisfied in practice, is essential for the solver P-GMRES.

The additive Schwarz iteration introduces the block Jacobi preconditioner

\[
K = \begin{bmatrix}
A_{11} \\
\ddots \\
A_{MM}
\end{bmatrix},
\]

which, together with the residual \( b - Ax^{(i)} \), defines a system whose solution provides an approximation of the error \( x - x^{(i)} \). Because this system decouples into \( M \) independent systems, it can be solved efficiently on parallel computers.

This form of domain decomposition has also been considered by Frank and Vuik [11]. For a thorough discussion of domain decomposition see the book [24] and the review article [7], and the extensive bibliography therein. Roughly speaking, the convergence rate suffers proportionally to the number of subdomains in each direction. The convergence rate may be made independent of the grid size by using constant overlaps or by application of coarse subspace correction. We will accelerate the convergence by a Krylov subspace method, as in [11].
Algorithm: GMRES

1. Start: Let initial guess $x_0$ be given.
   $q_0 = b - Ax_0$, Solve $Kw = q_0$, $\beta = \|w\|_2$, $\nu_1 = w/\beta$.

2. Arnoldi process:
   for $k = 1,2,\ldots$ until convergence:
   - $q_k = Av_k$
   - Solve $Kw = q_k$
   - $[v_{k+1}, h_k] = \text{orthonorm}(w, v_j, j \leq k)$
   end

3. Form approximate solution:
   - Define $V_k = [v_1, v_2, \ldots, v_k]$
   - Define $\tilde{H}_k = [h_1, h_2, \ldots, h_k]$
   - Compute $x_k = x_0 + V_ky_k$, where $y_k = \arg\min_y \|\beta e_1 - \tilde{H}_k y\|_2$ and $e_1 = [1, 0, \ldots, 0]^T$.

4. Restart: Compute $Ax_k$ and check termination criterion. If satisfied stop, else set $x_0 = x_k$ and goto 1.

---

**2.2 GMRES acceleration**

In practice (2) is solved iteratively, using $K$ as a preconditioner for a Krylov subspace method, such as the conjugate gradient method for symmetric problems or the GMRES method [22] for nonsymmetric problems. In contrast with [11], where GCR is used, we consider GMRES, shown in Fig. 1, in order to facilitate the comparison with P-GMRES [9].

The function $\text{orthonorm}(\ )$ takes input vectors $w$, orthonormalises $w$ with respect to the $v_i, i \leq k$, and returns the modified vector $v_{k+1}$. In serial computations, the modified Gram-Schmidt method (MGS), shown in Fig. 2, is usually employed for the $\text{orthonorm}(\ )$ function.

In parallel computations MGS has serious disadvantages, because the inner products require global communications, and therefore do not scale. Moreover, these inner products must be computed successively, and their number increases by one in every iteration step. Various alternatives have been proposed for MGS, e.g. orthogonalising a number of vectors simultaneously [8, 17, 18], Householder transformations [29, 11] or two-fold application of the classical Gram-Schmidt method (CGS) [16, 11]. However, these alternatives have some drawbacks, varying from loss of stability with respect to rounding errors [3] to an increase of the number of floating point operations. Also, most alternatives are not applicable when a preconditioner is used.
Algorithm: Modified Gram-Schmidt

\[ v_{k+1}, h_k = \text{orthonorm}(w, v_j, j \leq k) \]

for \( j = 1, 2, \ldots, k \)

\[ h_{k,j} = \langle w, v_j \rangle \]
\[ w = w - h_{k,j}v_j \]

end

\[ h_{k,k+1} = \| w \| \]
\[ v_{k+1} = w/h_{k,k+1} \]

Figure 2: The modified Gram-Schmidt algorithm

that varies in each iteration. Frank and Vuik [11] conclude from a comparison with MGS and Householder that re-orthogonalised CGS (see Fig. 3) is the most attractive method.

2.3 P-GMRES acceleration

Dekker [9] has proposed a modification of GMRES, called Partitioned GMRES (P-GMRES), which is applicable to (2) if the subdomains can be partitioned into two groups, such that for each pair of different subdomains from the same group the corresponding blocks \( A_{mn} \) and \( A_{nm} \) are zero.

In [9] the trivial case \( M = 2 \) is considered, but this situation also occurs if a red-black colouring of the subdomains is possible where only adjacent subdomains lead to nonzero blocks. In the sequel we assume that such a colouring exists. Let the restriction to the red subdomains be denoted by \( R^r \), and to the black ones by \( R^b \). Then the following equations, which are essential for P-GMRES, hold:

\[
R^r + R^b = I,
R^r K^{-1}AR^r x = R^r x,
R^b K^{-1}AR^b x = R^b x.
\]

P-GMRES, described in Fig 4, offers several advantages when compared to GMRES, whereas the computational costs in an iteration are about the same. First, P-GMRES yields an optimal approximation in the affine space

\[ x_0 + \text{Span}\{V_k^r, V_k^b\} \]

which has a higher dimension than

\[ x_0 + \text{Span}\{V_k\} \]
Algorithm: Classical Gram-Schmidt
\[v_{k+1}, h_k = \text{orthonorm}(w, v_j, j \leq k)\]
\[\text{for } j = 1, 2, \ldots, k\]
\[h_{k,j} = \langle w, v_j \rangle\]
\[\text{end}\]
\[h_{k,k+1} = \sqrt{\|w\|^2 - \sum_{j=1}^{k} \{h_{k,j}\}^2}\]
\[v_{k+1} = (w - \sum_{j=1}^{k} \{h_{k,j}v_j\}) / h_{k,k+1}\]

Figure 3: The classical Gram-Schmidt algorithm

in which subspace GMRES searches for an approximation. Also, there holds
\[\text{Span}\{V_k\} \subset \text{Span}\{V^r_k, V^b_k\},\]
so P-GMRES converges at least as fast as GMRES.

Secondly, we have two independent orthogonalisation processes, one for the variables from the red subdomains and one for the black subdomains. This property allows several possibilities for parallelisation. On a 2-processor machine we could perform two modified Gram-Schmidt algorithms with only a communication step after termination of MGS. When many processors are available, we could divide the processors into two groups, each taking care of one orthogonalisation process, thereby slightly reducing the amount of communication, as not all processors are involved in the computation of an inner product. In this case it might even be more attractive to overlap computation and communication [8] in a natural way: computation of a vector update and a local inner product for the red subdomains can be done simultaneously with the accumulation of the inner products for the black subdomains, vice versa, when each processor is assigned to both a red and a black subdomain. Then, the costs of the global communication would be negligible, provided the subdomains are not too small.

Finally we note that the minimisation problem in P-GMRES can be cheaply solved using Givens rotations, just as in GMRES [22], because \(\tilde{H}^r_k\) and \(\tilde{H}^b_k\) are both Hessenberg matrices. Due to the larger size \(2k \times 2k\) of the coefficient matrix each iteration step requires \(8k\) Givens rotations, compared to only \(k\) rotations in a GMRES iteration. This additional amount of operations, however, is usually very small compared to the costs of the inner products.

Further, the computational work in the Arnoldi process of P-GMRES is just the same as in GMRES. The vectors \(v^r_k\) and \(v^b_k\) are both restricted to part of the subdomains, so the two matrix multiplications with \(A\) amount to just one multiplication of \(A\) with a full vector. In the preconditioning we need to solve
\[Kw^b = q^b_k, Kw^r = q^r_k\]
only for the black, viz. red subdomains.
Algorithm: P-GMRES

1. **Start**: Let initial guess $x_0$ be given.
   
   \[
   q_0^r = R^r(b - Ax_0), \text{ Solve } Kw^r = q_0^r, \quad \beta^r = \|w^r\|_2, \quad v_1^r = w^r/\beta^r, \]
   
   \[
   q_0^b = R^b(b - Ax_0), \text{ Solve } Kw^b = q_0^b, \quad \beta^b = \|w^b\|_2, \quad v_1^b = w^b/\beta^b. \]

2. **Arnoldi process:**
   
   for $k = 1, 2, \ldots$ until convergence:
   
   - $q_k^b = R^bAv_k^r, q_k^r = R^rAv_k^b,$
   
   - Solve $Kw^b = q_k^b, Kw^r = q_k^r,$
   
   - $[v_{k+1}^r, h_k^r] = \text{orthonorm}(w^r, v_j^r, j \leq k),$
   
   - $[v_{k+1}^b, h_k^b] = \text{orthonorm}(w^b, v_j^b, j \leq k).$

3. **Form approximate solution:**
   
   - Define $V_k^r = [v_1^r, v_2^r, \ldots, v_k^r], V_k^b = [v_1^b, v_2^b, \ldots, v_k^b].$
   
   - Define $H_k^r = [h_1^r, h_2^r, \ldots, h_k^r], H_k^b = [h_1^b, h_2^b, \ldots, h_k^b].$
   
   - Compute $x_k = x_0 + V_k^r y_k^r + V_k^b y_k^b,$ where $y_k^r, y_k^b$ minimise
     
     \[
     \| \begin{bmatrix} \beta e_1 - I_{k^r} & H_k^r y_k^b \\ \beta e_1 - I_{k^b} & H_k^b y_k^r \end{bmatrix} \|_2, \]

     $e_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{k+1}$ and $I_k$ is the identity matrix, extended with a row of zeros.

4. **Restart**: Compute $Ax_k$ and check termination criterion. If satisfied stop, else set $x_0 = x_k$ and goto 1.

---

Figure 4: The P-GMRES algorithm
2.4  Subdomain solution

Solution for \( w \) from the preconditioning equation \( K w = q \) in the GMRES algorithm requires the solution of \( M \) independent subdomain systems \( A_{mm} w_m = q_m, m = 1, \ldots, M \), and similarly we have to solve \( M \) subdomain systems to obtain the solutions \( w^r \) and \( w^b \) from the equations \( K w^r = q^r, K w^b = q^b \) in the P-GMRES algorithm. In the formulations of the algorithms we have assumed that these subdomains are solved exactly. However, in practical problems this might be too expensive, especially when the subsystems are large, so that solution by an iterative method might be a better alternative. It is generally thought that the solution obtained should be very accurate [6, 4], otherwise GMRES acceleration may no longer be applied. In case of inaccurate subdomain solutions the methods GCR [26] and FGMRES [21], which also allow variable preconditioners, are more appropriate. However, these methods require more storage and, in case of GCR, additional computations. Moreover, P-GMRES is not suitable for very inaccurate subdomain solutions. Therefore, we require that the subdomain problems are solved with a moderate accuracy, and then restarted versions of GMRES are also applicable, as the following analysis shows. For results obtained with GCR and an inaccurate subdomain solution we refer to [11].

Suppose that, instead of solving \( K w_k = q_k \) exactly, we obtain an approximate solution \( \tilde{w}_k \), satisfying,

\[
K \tilde{w}_k = q_k + K e_k, q_k = A v_k,
\]

where \( e_k = \tilde{w}_k - w_k \).

Define

\[
E_k = [e_1, e_2, \ldots, e_k]
\]

and let \( \tilde{w}_k, k = 0, 1, \ldots \) be used to generate the Krylov subspace. Then, there holds

\[
K^{-1} A V_k = V_{k+1} \tilde{H}_k - E_k.
\]

Consequently, after \( m \) outer iterations, using the inexact subdomain solutions, we obtain for the preconditioned residual

\[
\| K^{-1} (A x_m - b) \| = \| K^{-1} (A x_0 - b) + K^{-1} A m y_m \| = \| V_{m+1} \tilde{H}_m y_m - E_m y_m + K^{-1} q_0 \| = \| V_{m+1} \tilde{H}_m y_m - E_m y_m - \beta V_{m+1} e_1 + e_0 \| \leq \| V_{m+1} \tilde{H}_m y_m - \beta V_{m+1} e_1 \| + \| e_0 - E_m y_m \| \leq \| \tilde{H}_m y_m - \beta e_1 \| + \| e_0 \| + \sum_{k=1}^{m} \| e_k \| \| y_m(k) \|. \tag{3}
\]

As \( \| y_m \| = \| x_m - x_0 \| \) will be bounded, we observe that the inexact subdomain solutions do not affect the preconditioned residual dramatically, as long as the errors \( e_k, k = 0, 1, \ldots, m \) are small compared to the estimation of the preconditioned residual \( \| \tilde{H}_m y_m - \beta e_1 \| \), which is calculated in GMRES. In the experiments we will investigate the influence of the accuracy in the solution of the subdomain problems on the convergence of the outer iterations of GMRES and P-GMRES.
A second question which arises, addresses the value of the tolerance to which the subdomain iterations should converge. There seems to be some theoretical evidence [25] that a fixed relative tolerance is optimal. However, in such a model it is usually assumed that convergence is linear and independent of previous iterations. This assumption is obviously not valid for (P-)GMRES, where the convergence is superlinear and the iterations are intimately related. Moreover, even if a fixed tolerance would be optimal, this value is not known beforehand. If the outer iteration converges slowly, too strict a tolerance will not be necessary in a restarted method. Also, when the outer iteration has almost converged, the accuracy can be relaxed, as the components \( y_m(k) \) will get small for increasing \( k \), according to (extending \( y_{k-1} \in \mathbb{R}^{k-1} \) with additional zeros)

\[
|y_m(k)| = |y_m(k) - y_{k-1}(k)| \leq \|y_m - y_{k-1}\| \leq \|x_m - x_{k-1}\|.
\]

### 3 Performance models

To give insight into the costs of the orthogonalisation procedure in GMRES and P-GMRES, and of the subdomain solution we consider simple performance models. In the first subsection we discuss the orthogonalisation. The emphasis will then be on the communication costs for parallel platforms, as inner products distributed over the processors are to be calculated. In the second subsection we consider the sequential costs of subdomain solvers, as it is assumed that each processor takes care of the solution for one (or more) subdomains.

#### 3.1 Orthogonalisation

The cost of orthogonalisation is mainly determined by the inner products and the vector updates which occur both in the MGS and the CGS algorithm. Here, we distinguish between inner products that can be computed simultaneously (i.e., with a single communication), those that cannot and inner products for half the vector length, as occur in P-GMRES. Following [11], we denote \( k \) simultaneous inner products by SIP(\( k \)). Two inner products and vector updates of half the vector length are denoted by HSIP(2) and haxpy(2). Then, the modified and re-orthogonalised Gram-Schmidt can be broken down into components as given in Table 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMRES(MGS)</td>
<td>( k + 1 ) SIP(1) ( k ) haxpy</td>
</tr>
<tr>
<td>GMRES(CGSI2)</td>
<td>2 SIP(( k + 1 )) ( 2k ) haxpy</td>
</tr>
<tr>
<td>P-GMRES(MGS)</td>
<td>( k + 1 ) HSIP(2) ( k ) haxpy(2)</td>
</tr>
</tbody>
</table>

Table 1: Number of operations in the \( k \)-th iteration of GMRES and P-GMRES

Let the time for communication of a message of \( n \) floating point numbers be given by

\[
t_{\text{comm}} = t_0 + \beta n,
\]

where \( t_0 \) is the communication startup time, and \( \beta \) is the time per floating point number, depending on the bandwidth. Let the time for \( n \) floating point operations be given by

\[
t_{\text{comp}} = \phi n.
\]
<table>
<thead>
<tr>
<th>Operation</th>
<th>Communication</th>
<th>Computation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>send($k$)</td>
<td>$t_0 + \beta k$</td>
<td>$n\phi$</td>
<td>Send a message of length $k$</td>
</tr>
<tr>
<td>flop($n$)</td>
<td>$f(p)(t_0 + \beta k)$</td>
<td>$k\phi$</td>
<td>$n$ floating point operations</td>
</tr>
<tr>
<td>$B(p, k)$</td>
<td>$2f(p)(t_0 + \beta k)$</td>
<td>$2kn^2\phi$</td>
<td>Broadcast $k$ elements</td>
</tr>
<tr>
<td>SIP($k$)</td>
<td>$2f(p/2)(t_0 + \beta)$</td>
<td>$2n^2\phi$</td>
<td>$k$ simultaneous inner products</td>
</tr>
<tr>
<td>HSIP(2)</td>
<td>$2f(p/2)(t_0 + \beta)$</td>
<td>$2n^2\phi$</td>
<td>$2$ simultaneous half IPs</td>
</tr>
<tr>
<td>axpy</td>
<td>$2n^2\phi$</td>
<td>vector update</td>
<td></td>
</tr>
<tr>
<td>haxpy(2)</td>
<td>$2n^2\phi$</td>
<td>$2$ half vector updates</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Communication and computation times

<table>
<thead>
<tr>
<th>Operation</th>
<th>Communication</th>
<th>Computation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLIP</td>
<td>$n^2\phi$</td>
<td></td>
<td>Half local inner product</td>
</tr>
<tr>
<td>ALIP</td>
<td>$2f(p)(t_0 + \beta)$</td>
<td></td>
<td>Accumulation of HLIPs</td>
</tr>
<tr>
<td>haxpy</td>
<td>$n^2\phi$</td>
<td></td>
<td>Half vector update</td>
</tr>
</tbody>
</table>

Table 3: Communication and computation times

Such a model is used in e.g. [15, 8, 11].

Let $p$ denote the number of processes, and define a function $f(p)$ which gives the maximum number of non-simultaneous sends necessary for a broadcast to $p - 1$ processes. The function $f(p)$ is machine dependent and also depends on the distribution of the processes on the machine. Common values are, $f(p) = \log_2(p)$ for a hypercube structure, and $f(p) = p - 1$ for an Ethernet broadcast. Assuming that each processor is responsible for an $n \times n$ subdomain with $n^2$ unknowns, we arrive at the times for the basic operations as in Table 2.

Based on the communication model outlined in Tables 1 and 2 the orthogonalisation time required for $s$ iterations of GMRES (without restart) using modified Gram-Schmidt (MGS), re-orthogonalised classical Gram-Schmidt (CGS2) and P-GMRES using modified Gram-Schmidt (P-MGS), is given by

$$t_{MGS} = s(s+3)[2n^2\phi + f(p)(t_0 + \beta)] - 2sn^2\phi, \quad (4)$$
$$t_{CGS2} = s(s+3)[4n^2\phi + 2f(p)\beta] + 4(s+1)f(p)t_0 - 4sn^2\phi, \quad (5)$$
$$t_{P-MGS} = s(s+3)[2n^2\phi + f(p/2)(t_0 + \beta)] - 2sn^2\phi. \quad (6)$$

Comparing these expressions, we see that the orthogonalisation in P-GMRES is slightly cheaper than MGS in GMRES, unless 2 processors are used, in which case P-GMRES requires no communication. On many processors, with high communication startup time, CGS2 seems to be favourable. However, if it is possible to overlap computation and communication, an implementation of P-GMRES might be considered where each processor is assigned both a red and a black subdomain, each containing $n^2/2$ unknowns. Then, the MGS algorithm for the red and black subdomains can be performed as in Fig. 5 (cf. [8]).
Algorithm: Modified implementation of MGS in P-GMRES

for $i = 1, 2, \ldots, k + 1$

- Accumulate local inner products $i - 1$ for black subdomains, if $i \neq 1$
  Update vectors on red subdomains, if $i \neq 1$
  Compute local inner products $i$ for red subdomains

- Accumulate local inner products $i$ for red subdomains
  Update vectors on black subdomains, if $i \neq 1$
  Compute local inner products $i$ for black subdomains

end

- Accumulate local inner products $k + 1$ for black subdomains

Figure 5: Modified implementation of MGS

Using the costs of each basic operation as given in Table 3, we arrive at the orthogonalisation time for this modification (P-MGSM)

$$t_{P-MGSM} = s(s + 1)\max(2n^2\phi, 2f(p)(t_0 + \beta)) + s[n^2\phi + 2f(p)(t_0 + \beta) + \max(n^2\phi, 2f(p)(t_0 + \beta))]$$

From [11] we derive representative values for the parameters $t_0, \beta$ and $\phi$

$t_0 \approx 4.7 \times 10^{-4}, \beta \approx 7.5 \times 10^{-6}, \phi \approx 4.9 \times 10^{-8}$

for a cluster of HP workstations, and for Cray T3E using MPI communications

$t_0 \approx 2.4 \times 10^{-5}, \beta \approx 5.4 \times 10^{-8}, \phi \approx 5.8 \times 10^{-8}$.

Assuming the models (4-6,7), and $f(p) = p - 1$ for the HP-cluster, $f(p) = \lceil \log_2(p) \rceil$ for the Cray T3E, we compute the quantities

$$\mathcal{F}_{CGS2} = t_{MGSM}/t_{CGS2},$$
$$\mathcal{F}_{PMGS} = t_{MGSM}/t_{P-MGSM},$$
$$\mathcal{F}_{PMGSM} = t_{MGSM}/t_{P-MGSM},$$

denoting the predicted speedup with respect to modified Gram-Schmidt in GMRES.

In Fig. 6 the results are plotted as function of $n$ for $s = 60$ and $p = 4, 9$ (HP-cluster), resp. $p = 4, 25$ (Cray T3E). We observe that the model predicts that CGS2 is advantageous for small subdomain sizes, when communication is relatively expensive, as on the HP-cluster. This observation has been made before in [11], where CGS2 and MGS are compared for the GCR method.
Figure 6: Predicted speedup for P-GMRES compared to GMRES

and the predictions verified in actual experiments. More importantly, the model predicts that modified Gram-Schmidt is more efficient in P-GMRES than in GMRES, due to the reduced communication costs, both on the HP-cluster and on the Cray T3E. The speedup varies between 2.8 for small subdomains (400 unknowns per subdomain) with expensive communication (HP-cluster with $p=9$) and 1.0 for large subdomains (10000 unknowns) on the Cray T3E. Moreover, an additional speedup can be obtained by the modified version of P-GMRES, when communication and computation are balanced, i.e.

$$2n^2\phi \approx 2f(p)(t_0 + \beta).$$

### 3.2 Costs of subdomain solution

We assume that each subdomain is solved on one processor, so no communication between processors is necessary and we can restrict ourselves to the sequential costs, determined by the number of floating point operations.

First, we consider the exact solution of the subdomain problem using a forward-backward substitution, after an LU-decomposition has been made. Let $n^2$ be the number of unknowns in the subdomain, and $n$ half the bandwidth of the coefficient matrix. Then the average costs of a subdomain solution is approximately

$$C_{\text{exact}} = (4n^3 + 2n^4/s)\phi,$$

(8)
Table 4: Number of operations and costs for \( m \) inner iterations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number</th>
<th>Costs</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>matvec</td>
<td>( m )</td>
<td>( 9n^2\phi )</td>
<td>Matrix-vector multiply</td>
</tr>
<tr>
<td>prec</td>
<td>( m + 1 )</td>
<td>( 10n^2\phi )</td>
<td>ILU-preconditioning</td>
</tr>
<tr>
<td>IP</td>
<td>( \frac{1}{2}(m + 1)(m + 2) )</td>
<td>( 2n^2\phi )</td>
<td>Inner product</td>
</tr>
<tr>
<td>axpy</td>
<td>( \frac{1}{2}m(m + 1) )</td>
<td>( 2n^2\phi )</td>
<td>Vector update</td>
</tr>
<tr>
<td>scal</td>
<td>( m + 1 )</td>
<td>( n^2\phi )</td>
<td>Vector scaling</td>
</tr>
<tr>
<td>sol</td>
<td>1</td>
<td>( 2mn^2\phi )</td>
<td>Solution update</td>
</tr>
</tbody>
</table>

where \( s \) denotes the number of outer iterations in (P-)GMRES necessary for convergence of the global problem, and \( \phi \) the costs of one floating point operation.

Secondly, suppose that the subproblem is solved inexactly by \( m \) iterations of GMRES using an ILUD-preconditioner ([19]) or an RILUD-preconditioner ([11]). For the two-dimensional problem considered in this paper the number of nonzeros in a row of the coefficient matrix will be 5, and hence 3 for the incomplete L and U factors. Then, the costs for the matrix-vector multiplication and the preconditioner are

\[
C_{\text{matvec}} = 9n^2\phi, \\
C_{\text{prec}} = 10n^2\phi.
\]

Note that these costs might be slightly reduced using Eisenstat’s trick [10], but that is not essential here. In Table 4 we list the costs and the number of basic operations in \( m \) inner iterations.

Neglecting the costs of the construction of the preconditioner (approx. \( 10n^2\phi \)) and the solution of the Hessenberg system (about \( m^2\phi \)), we obtain for the costs of the inexact solution

\[
C_{\text{gmres}(m)} = (2m^2 + 26m + 13)n^2\phi. \quad (9)
\]

Comparison of (8) and (9) shows that for this type of problems solution by GMRES is only competitive if \( 2\sqrt{n} \) iterations are sufficient to obtain a reasonable accurate subdomain solution. Hence, the number of unknowns in the subdomain, \( n^2 \), should be quite large, e.g. \( n > 100 \), but in that case the inner iteration will probably converge slowly and 20 iterations might not be sufficient. Alternatively, the subdomain problem might be solved very inaccurately (cf. [11]), leading to a small value of \( m \). However, the number of outer iterations will increase then, and the exact solution might be the most efficient one after all. We will pursue this issue in the numerical experiments further. Here, we conclude that, for the 2D problem considered, exact subdomain solution will probably be computationally the most efficient. Only when memory limitations exclude the use of a full LU-decomposition, subdomain solution by an iterative method will be of value.

### 4 Numerical experiments

In this section we give numerical results which provide insight into the convergence behaviour of P-GMRES in comparison with GMRES. We also assess the performance of several subdo-
Table 5: Number of iterations for the $60 \times 60$ grid with $M \times M$ subdomains

<table>
<thead>
<tr>
<th>$M$</th>
<th>$\beta = 0$</th>
<th>$\beta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GMRES</td>
<td>P-GMRES</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

As a test example, we consider a Poisson problem, discretised with a finite difference method on a square domain. Such a problem is similar to the pressure correction matrix, which is solved in each time step of an incompressible Navier-Stokes simulation to enforce the divergence-free constraint [27], apart from some asymmetry in the pressure correction matrix. As the test example is meant to model such a pressure correction matrix, we do not exploit the symmetry in the experiments. The domain is decomposed into $M \times M$ subdomains, each containing $n \times n$ grid points. With $h = \Delta x = \Delta y = 1/(Mn + 1)$ the discretisation is

$$4u_{ij} - u_{i,j-1} - u_{i,j+1} - u_{i-1,j} - u_{i+1,j} = h^2 f_{ij}. \quad (10)$$

The right-hand side function is $f_{ij} = f(ih, jh)$, where

$$f(x, y) = x(1-x)\{2\beta x(1-3y) - 32\} + y(1-y)\{2\beta y(1-3x) - 32\}. \quad (11)$$

Homogeneous Dirichlet boundary conditions $u = 0$ are defined on $\partial \Omega$. Note that this example is almost identical to the one in [11], apart from a different discretisation at the boundaries. Moreover, we introduced an additional term in the right-hand side, as the formulation in [11], with $\beta = 0$, suffers from an 8-fold symmetry which usually speeds up the convergence considerably.

### 4.1 Convergence behaviour with exact subdomain solution

In this section we compare the speed of convergence of GMRES and P-GMRES. For all tests a fixed restart value of $s = 30$ was used, and the solution was computed after the initial (preconditioned) residual has been reduced by a factor of $10^6$. In all cases the subdomain problems were solved exactly.

In the first experiment we compare results for a fixed problem size on the $60 \times 60$ grid with $M \times M$ subdomains, $M = 2, \ldots, 6$, and two different values for $\beta$ in (11). In Table 5 we list the required number of iterations. It is interesting to observe that the symmetry in the problem
(β = 0) influences the convergence substantially. Therefore, we think that it is more appropriate to consider the case with β = 1 as a model for practical problems. Also note that P-GMRES for M even does not profit as much from the symmetry as GMRES does. This can be explained by the fact that the solutions on the black and red subdomains are identical in the symmetric case; consequently, P-GMRES and GMRES converge in exactly the same way. In all other cases P-GMRES requires about 20% less iterations than GMRES.

We considered grids of dimension 60, 120, 180, 240 and 300 in a second experiment. The number of required iterations for various M × M subdomain partitionings are plotted in Fig. 7. In all cases we took the problem with β = 1. One sees that P-GMRES converges faster than GMRES in all cases but one (n = 300, M = 4). A marked improvement is obtained for an odd number of subdomains and a relatively small grid. Here the difference between solutions on the red and the black subdomains is most pronounced, and P-GMRES profits from the discrimination between these two. For M even, there will be 2 black and 2 red subdomains in the corners of Ω, and the solutions on these two sets of subdomains will not behave very differently.

In a last experiment we chose different starting values for the problem with β = 1 on the 60 × 60 grid, viz.

\[
x_{ij}^{(0)} = \begin{cases} 
\frac{\max(i,j) - n - 1}{n} u_{ij}, & i \leq n, j \leq n, \\
u_{ij}, & \text{otherwise},
\end{cases}
\]

(12)

where \(u_{ij}\) denotes the exact solution of (10). Consequently, we start the iteration with the exact solution in all subdomains, but the first one. Table 6 shows that GMRES does not profit from such a good starting value at all, but P-GMRES has converged after 1 iteration, as could be

![Figure 7: # Iterations for various subdomains](image-url)
expected. This suggests that P-GMRES might be very efficient for problems whose solutions behave differently on various parts of the domain, such as layered problems [28].

4.2 Evaluation of approximate subdomain solvers

In this section we compare the performance of a number of approximate subdomain solvers to get an impression of which solvers might be an alternative for exact subdomain solution in (P-)GMRES. Again, we used a fixed restart value of $s = 30$ in the outer iteration, unless noted otherwise, and a relative tolerance of $10^{-6}$.

The subdomain approximations will be denoted as follows:

- EX = exact subdomain solution,
- GMR$k$ = (restarted) GMRES with a tolerance of $10^{-k}$, preconditioned with RILUD,
- GMR$k$F = (restarted) GMRES with a tolerance of $10^{-k}F$, preconditioned with RILUD.

The last method needs some explanation. The factor $F$ is given by

$$F = \min \left( 10^{k-1}, \max \left( 10^{k-7}, \frac{\text{res}(0)}{\text{res}(j-1)}, \frac{\text{res}(j_0)}{\text{res}(j-1)} \right) \right),$$

where $\text{res}(0), \text{res}(j_0), \text{res}(j-1)$ denote the norms of the initial residual, the residual at the beginning of the last restart and the residual after the previous outer iteration, resp. This means that we aim to reduce the residual in a cycle of restarted (P-)GMRES by a factor of $10^{-k}$, unless the residual at the last restart is already small. Moreover, the tolerance for the subdomain approximation is bounded above by 0.1. Because the inaccuracies from the subdomain approximations will persist during a cycle of outer iterations, we do not continue the outer iteration until $s = 30$, but restart as soon as the condition

$$\text{res}(j) < 2 \sqrt{(j - j_0)10^{-k}\text{res}(j_0)}$$

is satisfied.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$x^{(0)} = 0$</th>
<th>$x^{(0)}$ from (12)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GMRES</td>
<td>P-GMRES</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 6: Number of iterations for the $60 \times 60$ grid with $M \times M$ subdomains
4.2.1 Convergence on a small problem

We applied P-GMRES for the fixed problem on the 60×60 grid with $M \times M$ subdomains. The required number of outer iterations and the averaged number of inner iterations (in parentheses) is listed in Table 7. For the sake of comparison we also include the iteration count for the exact subdomain solution.

It is seen that the outer iteration does not suffer much from an approximate subdomain solution, if a sufficiently small tolerance is imposed for the subdomain problems (GMR8, GMR6). However, the number of inner iterations and thus the amount of work is rather high in these cases, even for the small subdomains considered here. Relaxing the tolerance for the inner loop (GMR4) reduces the number of inner iterations, but leads to a significant increase in the number of outer iterations. The methods using a flexible inner loop tolerance perform much better. GMR7F requires the same amount of inner iterations as the inaccurate solver GMR4, without much loss of accuracy in the outer iteration, whereas GMR4F is about twice as cheap as GMR4, see (9).

4.2.2 Convergence on a larger problem

In this subsection we consider the fixed problem on the 300×300 grid, which has also been used in [11] to assess the performance of subdomain solvers for GCR (although [11] has $\beta = 0$ in the right hand side function (11)). We applied P-GMRES with the flexible subdomain solver GMRkF for various $M \times M$ partitionings of $\Omega$, and list the number of outer and averaged inner iterations in Table 8. For the sake of comparison we also quote those numbers for GCR with GMR6 from [11].

Note that the flexible tolerance subdomain solvers perform quite satisfactory, even for the rather rude tolerances in GMR4F. The difference in inner iterations between the fixed strategy in GCR and GMRkF is striking, whereas the convergence of the outer iteration is comparable. This can be explained by the fact that inaccuracies introduced by the subdomain solves are soon detected by the restarts in the outer loop, so they do not have a chance to spoil too many subsequent iterations. Fig. 8 illustrates the convergence of P-GMRES with the GMR4F subdomain solver for the 6×6 subdomain case.

The peaks in the graph for GMR4F occur at the restarts, indicating that the calculated resid-
Table 8: Outer and inner iterations for various subdomain solvers, 300 × 300 grid

<table>
<thead>
<tr>
<th>Solver</th>
<th>2 × 2</th>
<th>3 × 3</th>
<th>4 × 4</th>
<th>5 × 5</th>
<th>6 × 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX</td>
<td>32</td>
<td>59</td>
<td>87</td>
<td>94</td>
<td>123</td>
</tr>
<tr>
<td>GCR</td>
<td>78(68.4)</td>
<td>83(38.7)</td>
<td>145(31.4)</td>
<td>168(26.4)</td>
<td></td>
</tr>
<tr>
<td>GMR6F</td>
<td>84(14.4)</td>
<td>105(13.0)</td>
<td>116(12.9)</td>
<td>114(12.8)</td>
<td>142(12.0)</td>
</tr>
<tr>
<td>GMR5F</td>
<td>62(17.3)</td>
<td>102(12.2)</td>
<td>123(11.8)</td>
<td>115(12.2)</td>
<td>170(10.6)</td>
</tr>
<tr>
<td>GMR4F</td>
<td>110(12.1)</td>
<td>144(10.8)</td>
<td>161(10.0)</td>
<td>163(10.1)</td>
<td>175(10.7)</td>
</tr>
</tbody>
</table>

Figure 8: Convergence of P-GMRES (GMR4F), 300 × 300 grid, 6 × 6 subdomains

ual is contaminated by inaccuracies from the subdomain solve. However, subsequent iterations quickly diminish the value of these residuals. The number of inner iterations, used in this example, is plotted in Fig. 9. It is seen that the required number of inner iterations rapidly decreases in the course of an outer cycle, using the flexible tolerance. Consequently, the averaged number of inner iterations is substantially reduced.

In some experiments we observed that the behaviour of the residual was quite erratic in the final stage of convergence (see Fig. 8). Then, the calculated residual in the outer iteration is small enough, hence the cycle is terminated, but the actual residual, calculated after the update of the solution, does not yet satisfy the stopping criterion, so a new cycle of outer iterations is started. This phenomenon is probably due to the crude tolerance \((tol \approx 0.1)\) with which the subproblems are solved. Requiring \(tol \leq 0.01\) gives a much more regular behaviour of the calculated residual, however at the costs of more inner iterations.

Finally we remark that in [11] GCR is also applied with the subdomain solvers GMR2, GMR1 and RILUD, the latter standing for just one forward backward substitution with the incomplete factors from the RILUD decomposition. We quote their results for the 5 × 5 subdomain case in Table 9.

They note that the inner loop tolerance of 0.1 is insufficient for fast global convergence,
although it is still an expensive subdomain approximation. We did not apply these crude toler-
ances, because the flexible strategy GMR4F already invokes large inner loop tolerances during
part of the outer iterations, see (13), and moreover (P-)GMRES is more sensitive to inaccurate
subdomain solutions than GCR. We also did not apply RILUD as a preconditioner. Although
GMRES can be combined with RILUD very well, P-GMRES will fail, as some accuracy in the
solution of the subdomain problem is required for this method.

4.2.3 Performance for the larger problem

Fig. 10 shows the performance of the P-GMRES method with various subdomain solvers for the
300×300 grid. The figure shows the number of floating point operations per subdomain required
for the convergence of the outer iteration. These numbers are computed from the costs of the
outer iteration and (8), (9). Increasing the number of subdomains reduces the computational costs
substantially, showing the potential of the domain decomposition method for parallelisation.
Note that the exact subdomain solver is the most efficient one for the relatively small grids
considered here.
Conclusions

For applications which require domain decomposition the partitioned iterative method P-GMRES offers advantages compared to GMRES, both with respect to speed of convergence and communication costs. These advantages are expected to be more pronounced for problems with large variations in the solution on the various subdomains, e.g. in layered problems ([28]). However, P-GMRES can only be applied if a red black colouring of the subdomains is possible.

Although monotone convergence of (P-)GMRES is only guaranteed in case of exact subdomain solution, it is also possible to solve the subdomain problems approximately, in combination with restarts. Our experiments indicate that a large reduction in computation time is obtained by a flexible tolerance strategy in the subdomain problems, which contradicts theoretical suggestions in literature ([25]). A rather crude tolerance \((tol \approx 0.1)\) is allowed in the subdomain solutions, but a stricter tolerance leads to a smoother convergence. However, for the size of the problems we considered, upto 22500 unknowns per subdomain, an exact subdomain solver still turned out to be the most efficient one. Nevertheless, it seems worthwhile to modify P-GMRES according to the ideas in [21] to recover the monotone convergence. This would only cost additional storage, which is usually not a problem on a parallel system.

Considering the computational work only, the domain decomposition method profits from a large number of subdomains. Notwithstanding the slower convergence, the work per subdomain decreases. This allows good opportunities for parallelisation if the communication costs are small. A performance model indicates that the orthogonalisation in P-GMRES is about twice as cheap as in GMRES for relatively small problems; using 9 processors and up to 5000 unknowns per processor for the HP-cluster. The advantage is about 20% on large problems, e.g. 10000 unknowns per subdomain on the Cray T3E. The model also indicates that a speedup of 1,5 can be obtained if computation and communication are balanced, and can be overlapped. For very small problems it might be advantageous to use the re-orthogonalised classical Gram-Schmidt orthogonalisation, as suggested in [11].
References


