A Parallel Implementation of the Block Preconditioned GCR Method

C. Vuik[1] and J. Frank[1 2]

[1] Delft University of Technology, Faculty of Information Technology and Systems, Department of Technical Mathematics and Informatics, P.O. Box 5031 2600 GA Delft, The Netherlands, c.vuik@math.tudelft.nl

[2] Center for Mathematics and Computer Science (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, frank@math.tudelft.nl

Abstract. The parallel implementation of GCR is addressed, with particular focus on communication costs associated with orthogonalization processes. This consideration brings up questions concerning the use of Householder reflections with GCR. To precondition the GCR method a block Gauss-Jacobi method is used. Approximate solvers are used to obtain a solution of the diagonal blocks. Experiments on a cluster of HP workstations and on a Cray T3E are given.

Keywords: approximate subdomain solution; parallel Krylov subspace methods; orthogonalization methods

1 Introduction

This paper addresses the parallel implementation of a Krylov accelerated block Gauss-Jacobi method for the DeFT Navier-Stokes solver described in [15], and is the continuation of work summarized in [5]. Results from a parallel implementation of a Krylov-accelerated Schur complement domain decomposition method are presented in [3]. We report results for a Poisson problem on a square domain, which is representative of the system which must be solved for the pressure correction method used in DeFT.

Aside from the preconditioning, the main parallel operations required in these methods are distributed matrix-vector multiplications and inner products. For many problems, the matrix-vector multiplications require only nearest neighbor communications, and are very efficient. Inner products, on the other hand, require global communications; therefore, the focus has been on reducing the number of inner products [8, 16], overlapping inner product communications with computation [6], or increasing the number of inner products that can be computed with a single communication [2, 13].

The block Gauss-Jacobi preconditioner is described in Section 1.1. Parallel implementations of orthogonalization procedures for the GCR (Generalized Conjugate

Residual) method are investigated in Section 2. Experiments done on a cluster of workstations and a Cray T3E are given in Section 3.

1.1 The block Gauss-Jacobi preconditioner

We consider an elliptic partial differential equation discretized using a finite volume or finite difference method on a computational domain Ω. Let the domain be the union of M nonoverlapping subdomains Ω_m, $m = 1, \ldots, M$.Discretization of the PDE results in a sparse linear system $Ax = b$, with x, $b \in \mathrm{R}^N$. When the unknowns which share a common subdomain are grouped together into blocks one gets the block system:

$$\begin{bmatrix} A_{11} & \ldots & A_{1M} \\ \vdots & \ddots & \vdots \\ A_{M1} & \ldots & A_{MM} \end{bmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_M \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_M \end{pmatrix}. \tag{1}$$

In this system, one observes that the diagonal blocks A_{mm} express coupling among the unknowns defined on a common subdomain (Ω_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 neighboring subdomains.

In order to solve system (1) we use the block Gauss-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. An important point is the required tolerance of these inner iterations (see [9]). Since the number of inner iterations may vary from one subdomain to another, and in each outer iteration, the effective preconditioner is nonlinear and varies in each outer iteration.

Our choice of approximate solution methods is motivated by the results obtained in [4]. In that paper, GMRES was used as to approximately solve subdomain problems to within fixed tolerances of 10^{-4}, 10^{-3}, 10^{-2} and 10^{-1}. Additionally, a blockwise application of the RILUD preconditioner was used. RILUD, a diagonal-restricted variant of the preconditioner introduced in [1], is a weighted average of an ILUD preconditioner [14] and an MILUD preconditioner [10]. The weighting parameter ω, was assigned a value of 0.95 in our experiments. For small problems GCR without a preconditioner converges in a reasonable amount of time. However, when the gridsize increases the CPU time for GCR without preconditioning is much higher than for the preconditioned GCR method.

2 Orthogonalization methods for the GCR method

2.1 The GCR method

One of the Krylov subspace methods which allows a variable preconditioner is the GCR method [7], [17]. In this paper the Euclidean inner product $\langle x, y\rangle = x^T y$ and associated norm $||x|| = (x^T x)^{1/2}$ are used.

Algorithm: GCR

```
Given: initial guess $x_0$
$r_0 = b - Ax_0$
for $k = 1, \ldots,$ convergence
    Solve $K\tilde{v} = r_{k-1}$ (approximately)
    $\tilde{q} = A\tilde{v}$
    $[q_k, v_k] =$ orthonorm $(\tilde{q}, \tilde{v}, q_i, v_i, i < k)$
    $\gamma = q_k^T r_{k-1}$
    Update: $x_k = x_{k-1} + \gamma v_k$
    Update: $r_k = r_{k-1} - \gamma q_k$
end
```

The function **orthonorm()** takes input vectors $\tilde{q}$ and $\tilde{v}$, orthonormalizes $\tilde{q}$ with respect to the q_i, $i < k$, updating $\tilde{v}$ as necessary to preserve the relation $\tilde{q} = A\tilde{v}$, and returns the modified vectors q_k and v_k.

The primary challenges to parallelization of GCR are parallelization of the preconditioning and parallel computation of the inner products. Inner products require global communication and therefore do not scale. Much of the literature on parallel Krylov subspace methods and parallel orthogonalization methods is focused on orthogonalizing a number of vectors simultaneously. However, this is not possible using a preconditioner which varies in each iteration. For this reason, we need a method for orthogonalizing one new vector against an orthonormal basis of vectors.

2.2 Orthogonalization methods

The modified Gram-Schmidt method suffers from the fact that the inner products must be computed using successive communications, and the number of these inner products increases proportional to the iteration number. This is not the case if one uses the classical Gram-Schmidt method. In this algorithm all necessary inner products can be computed with a single global communication. Unfortunately, the classical Gram-Schmidt method is unstable with respect to rounding errors, so this method is rarely used. On the other hand, Hoffmann [11] gives experimental evidence indicating that a two-fold application of the classical Gram-Schmidt method is stable. A third method which has been suggested is the parallel implementation of Householder transformations, introduced by Walker [18]. We shall reformulate that method for GCR in the following section. Additionally, we will present a simple parallel performance analysis for comparison of these three orthogonalization procedures.

2.3 Householder orthogonalization

In the following discussion we use the notion a_k to represent the kth column of a matrix A and $a^{(i)}$ to represent the ith component of a vector a. Let a matrix $A \in \mathbb{R}^{n \times m}$, $m \leq n$ with linearly independent columns be factored as QZ, where Q is orthogonal and Z is upper triangular. Then the kth column of A is given by $a_k = Qz_k$ and the columns of Q form an orthonormal basis for the span of the columns of A.

We construct Q as the product of a series of Householder reflections, $Q = P_1 \cdots P_m$, used to transform A into Z. The matrices $P_i = I - 2\frac{w_i w_i^T}{w_i^T w_i}$, with $w_i^{(j)} = 0$ for $j < i$ have the property: $P_i(P_{i-1} \cdots P_1)a_i = z_i$.

Suppose one has already produced k orthogonal basis vectors. To compute w_{k+1} one must first apply the previous reflections to a_{k+1} as described in [18]: $\tilde{a} = P_k \cdots P_1 a_{k+1} = (I - 2W_k L_k^{-1} W_k^T)a_{k+1}$, where W_k is the matrix whose columns are $w_1, \ldots, w_k$, and where

$$L_k = \begin{bmatrix} 1 & & & \\ 2w_2^T w_1 & 1 & & \\ \vdots & & \ddots & \\ 2w_k^T w_1 & \ldots & 2w_k^T w_{k-1} & 1 \end{bmatrix}.$$

Note especially that in the $(k+1)$th iteration one must compute the last row of L_k, which is the vector $(2w_k^T W_{k-1}, 1)$, as well as the vector $W_k^T a_{k+1}$. This requires $2k-1$ inner products, but they may all be computed using only a single global communication.

Let $\hat{a}$ be the vector obtained by setting the first k elements of $\tilde{a}$ to zero. The vector w_{k+1} is chosen as: $w_{k+1} = \hat{a} + \text{sign}\,(\hat{a}^{(k+1)})||\hat{a}||e_{k+1}$. In practice, the vectors w_k are normalized to length one. The length of w_{k+1} can be expressed as $||w_{k+1}|| = \sqrt{2\alpha^2 - 2\alpha\hat{a}^{(k+1)}}$ where $\alpha = \text{sign}\,(\hat{a}^{(k+1)})||\hat{a}||$. The $(k+1)$th column of Q is the new orthonormal basis vector:

$$q_{k+1} = \frac{1}{\alpha}\left[a_{k+1} - \sum_{i=1}^{k} \tilde{a}^{(i)} q_i\right].$$

Within the GCR algorithm, the same linear combination must be applied to the v_i to obtain v_{k+1}.

2.4 Performance analysis

In this section the costs of the orthogonalization methods are considered (for the details we refer to [9]).

Re-orthogonalized Classical Gram-Schmidt (CGS2)

The kth iteration of this method costs $3k$ vector updates and $2k$ inner products. To compute the inner products 2 global messages of length k are sent.

Modified Gram-Schmidt (MGS)
The kth iteration of MGS costs $2k$ vector updates and k inner products. For the inner products k global messages of length 1 are required.

Householder (HH)
In the kth iteration of the Householder method, $3k$ vector updates and $2k$ inner products are done. In every iteration 3 communications are necessary: two global messages one with length k and the other with length 1, and a broadcast of k elements.

Comparing the costs we expect that the wall-clock time for CGS2 and HH are comparable. When communication is slow (large latency) with respect to computation one expects that these methods are faster than MGS. Otherwise MGS may be the fastest method because MGS needs less floating point operations.

3 Numerical experiments

Firstly we present some time measurements to compare the various orthogonalization methods. Secondly we investigate the scalability of the parallel block preconditioned GCR method. Each processor is responsible for an $n \times n$ subdomain with n^2 unknowns.

3.1 Measurements of the orthogonalization methods

Tests were performed on a cluster of HP workstations and on a a Cray T3E using MPI communication subroutines. The wall clock times in the orthogonalization part are measured when 60 GCR iterations are performed.

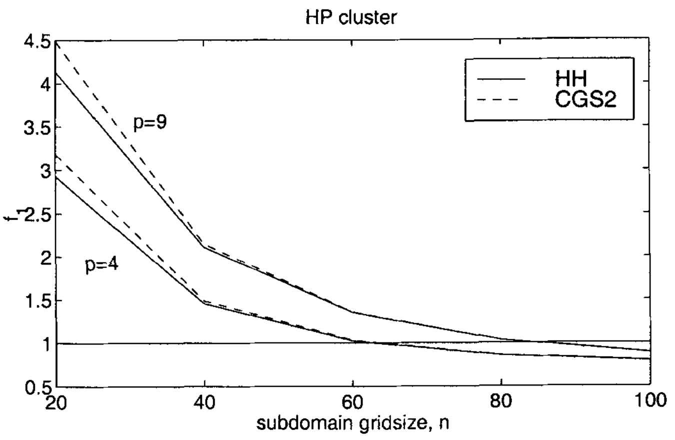
In Figure 1 the parameters

$$\mathcal{F}_{\text{HH}} = \frac{\text{orthog. time MGS}}{\text{orthog. time HH}}$$

and

$$\mathcal{F}_{\text{CGS2}} = \frac{\text{orthog. time MGS}}{\text{orthog. time CGS2}}$$

are plotted as functions of n. In each subdomain an $n \times n$ grid is used. The number of subdomains is equal to the number of processors. On the workstation cluster (HH) and (CGS2) are only advantageous when the number of unknowns is less than 3600 on 4 processors and less than 6400 on 9 processors. On the Cray T3E, the number of unknowns per processor should be fewer than 1000 for 9 or even 25 processors. For larger problems the smaller amount of work involved in modified Gram-Schmidt orthogonalization outweighs the increased communication cost.



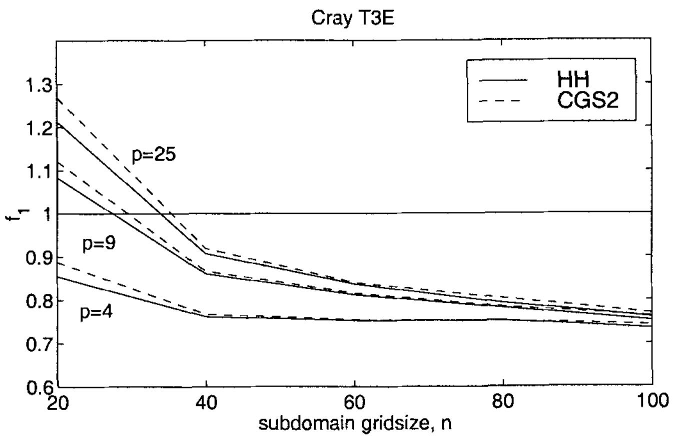


Fig. 1. Measured speedup with Householder (HH) orthogonalization and Re-orthogonalized Classical Gram-Schmidt (CGS2)

3.2 Evaluation of approximate subdomain solvers

As a test example, we consider a Poisson problem, discretized with the finite volume method on a square domain. The pressure correction matrix, which we solve in each time step of an incompressible Navier-Stokes simulation to enforce the divergence-free constraint [12], is similar to a Poisson problem, but with asymmetry arising from the use of curvilinear coordinates. Solution of this system requires about 75% of the computing effort. So that we can obtain a useful indication of the performance of our method on the pressure correction matrix, we do not exploit the symmetry of the Poisson matrix in these experiments. The domain is composed of a $\sqrt{p} \times \sqrt{p}$ array of subdomains, each with an $n \times n$ grid.

With $h = \Delta x = \Delta y = 1.0/(\sqrt{p}n)$ the discretization is

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

The right hand side function is $f_{ij} = f(ih, jh)$, 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, for example, $u_{0j} = -u_{1j}$ on boundaries. This ghost cell scheme allows natural implementation of the block Gauss-Jacobi preconditioner as well.

We compare speedups obtained with a number of approximate subdomain solvers

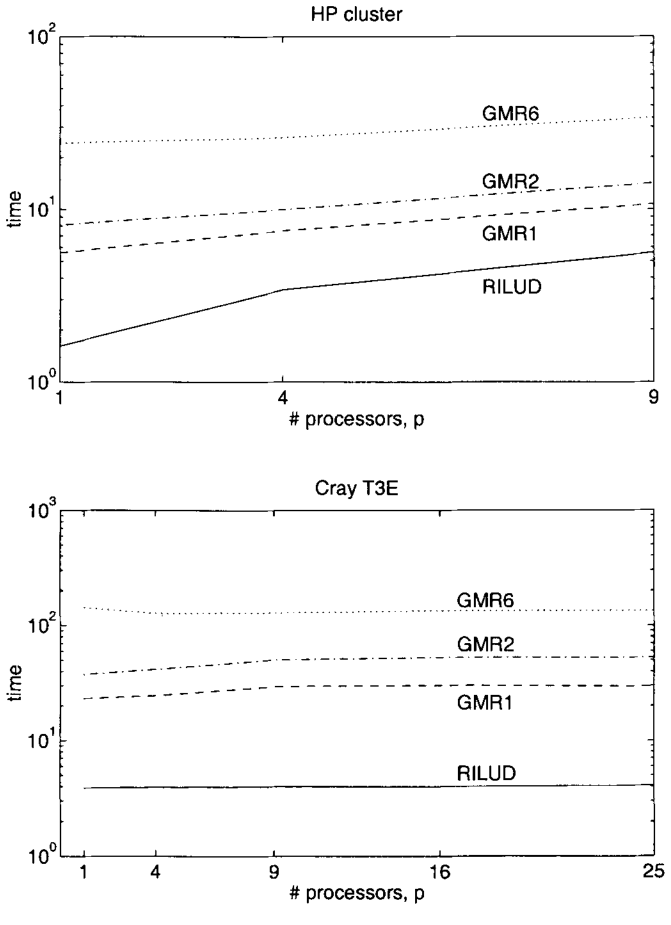


Fig. 2. Computation time for fixed subdomain size of 120 × 120

to get an impression of which solvers might be effectively used with the Navier-Stokes equations. For these tests a fixed number of GCR iterations (30) are done, and modified Gram-Schmidt was used as the orthogonalization method. The performance measure is wall clock time, after initialization, taken as the minimum achieved over three runs.

The subdomain approximations will be denoted as follows:

- GMR6 = restarted GMRES with a tolerance of 10^{-6},
- GMR2 = restarted GMRES with a tolerance of 10^{-2},
- GMR1 = restarted GMRES with a tolerance of 10^{-1},
- RILUD = one application of an RILUD preconditioner.

Figure 2 shows a comparison of the parallel scalability of the domain decomposition method with approximate subdomain solution. The figure shows computation times on 1, 4 and 9 processors (1, 4, 9, 16 and 25 processors for the Cray T3E) with a fixed subdomain size of 120×120. Note that the method scales almost perfectly on the Cray for this range of processors. On the workstation cluster, the scaling is somewhat poorer. The scaling for the GMR variants is reasonable, whereas the scaling of RILUD is bad.

4 Conclusions

Wall clock measurements for the modified Gram-Schmidt, re-orthogonalized classical Gram-Schmidt, and Householder orthogonalization methods indicates that classical Gram-Schmidt and Householder require approximately the same amount of work and communication, making the re-orthogonalized classical Gram-Schmidt more attractive, since it is easier to implement and more stable. The Householder and re-orthogonalized classical Gram-Schmidt methods are most effective for relatively small problems: using nine processors, up to about 900 unknowns per processor for a Cray T3E, or 8000 unknowns per processor for a cluster of workstations.

For this type of problem, the best subdomain approximation method in parallel is a simple incomplete factorization restricted to the diagonal: the RILUD factorization. With this preconditioner used as a subdomain approximation, the approximate solves become so cheap (and yet sufficiently accurate) that they offset the increased number of GCR iterations resulting from inaccurate subdomain solution.

References

1. O. Axelsson and G. Linskog. On the eigenvalue distribution of a class of preconditioning methods. *Numerische Mathematik*, 48:479–498, 1986.
2. Z. Bai, D. Hu, and L. Reichel. A Newton-basis GMRES implementation. *IMA Journal of Numerical Analysis*, 14:563–581, 1994.
3. E. Brakkee, A. Segal, and C. G. M. Kassels. A parallel domain decomposition algorithm for the incompressible Navier-Stokes equations. *Simulation Practice and Theory*, 3:185–205, 1995.

4. E. Brakkee, C. Vuik, and P. Wesseling. Domain decomposition for the incompressible Navier-Stokes equations: Solving subdomain problems accurately and inaccurately. *International Journal for Numerical Methods in Fluids*, 26:1217–1237, 1998.
5. Erik Brakkee. *Domain Decomposition for the Incompressible Navier-Stokes Equations*. PhD thesis, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands, April 1996.
6. 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. *Applied Numerical Mathematics*, 18:441–459, 1995.
7. Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, April 1983.
8. J. Erhel. A parallel GMRES version for general sparse matrices. *Electronic Transactions on Numerical Analysis (http://etna.mcs.kent.edu)*, 3:160–176, 1995.
9. J. Frank and C. Vuik. Parallel implementation of a multiblock method with approximate subdomain solution. *App. Num. Math.*, 1998. to appear.
10. Ivar Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.
11. Walter Hoffman. Iterative algorithms for Gram-Schmidt orthogonalization. *Computing*, 41:335–348, 1989.
12. J. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal on Scientific and Statistical Computing*, 7(3):870–891, 1986.
13. G. Li. A block variant of the GMRES method on massively parallel processors. *Parallel Computing 23*, 23:1005–1019, 1997.
14. J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31:148–162, 1977.
15. A. Segal, P. Wesseling, J. van Kan, C.W. Oosterlee, and K. Kassels. Invariant discretization of the incompressible Navier-Stokes equations in boundary-fitted coordinates. *International Journal for Numerical Methods in Fluids*, 15:411–426, 1992.
16. R. B. Sidje. Alternatives for parallel Krylov subspace basis computation. *Numerical Linear Algebra with Applications*, 4(4):305–331, 1997.
17. Henk A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.
18. Homer F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 9(1):152–163, 1988.