On the influence of the orthogonalization scheme on the parallel performance of GMRES

Valérie Fraysse[1], Luc Giraud[1] and Hatim Kharraz-Aroussi[2]

[1] CERFACS, 42 av. Gaspard Coriolis, 31057 Toulouse Cedex 1, France.
Email : `fraysse,giraud@cerfacs.fr`.
[2] ENSIAS, BP713, Agdal, Rabat, Morocco.

CERFACS Technical Report - TR/PA/98/07

Abstract. In Krylov-based iterative methods, the computation of an orthonormal basis of the Krylov space is a key issue in the algorithms because the many scalar products are often a bottleneck in parallel distributed environments. Using GMRES, we present a comparison of four variants of the Gram-Schmidt process on distributed memory machines. Our experiments are carried on an application in astrophysics and on a convection-diffusion example. We show that the iterative classical Gram-Schmidt method overcomes its three competitors in speed and in parallel scalability while keeping robust numerical properties.

1 Introduction

Krylov-based iterative methods for solving linear systems are attractive because they can be rather easily integrated in a parallel distributed environment. This is mainly because they are free from matrix manipulations apart from matrix-vector products which can often be parallelized. The difficulty is then to find an efficient preconditioner which is good at reducing the number of iterations without degrading too much the parallel performance. We consider the solution of a large sparse linear system

$$Ax = b$$

where A is a nonsingular $n \times n$ matrix, b and x are two vectors of length n. Given a starting vector x_0, the GMRES method [10] consists in building an orthonormal basis V_m for the Krylov space

$$\mathcal{K}(A) = \left\{x_0, Ax_0, \ldots, A^{m-1}x_0\right\}.$$

The integer m is called the projection size. GMRES produces a solution whose restriction to this Krylov space has a minimal 2-norm residual. When one wants to limit the amount of storage for V_m, m is kept fixed and the method is restarted with a better initial guess: m is then called the restart parameter and the restarted GMRES method is denoted by GMRES(m). The construction of the basis V_m is an important step of GMRES. Several variants of the Gram-Schmidt (GS) process are available; they have different efficiencies and different numerical properties in finite precision arithmetic [1]. Classical GS (CGS) is the most efficient but is numerically unstable, modified GS (MGS) improves on CGS but can still suffer from a loss of orthogonality. The iterative MGS (IMGS) method and the iterative CGS (ICGS) have been designed to reach the numerical quality of the Householder (or Givens) factorization with a reasonable computational cost. From a point of view of computational efficiency, CGS and ICGS are the best because the scalar products can be gathered and implemented in a matrix-vector product form. However, the iterative procedures ICGS and IMGS may require more than twice as much work than their counterparts CGS and ICGS, when reorthogonalization is needed. Our implementations are the ones described in [1]. From a numerical point of view, it has been proved that MGS, IMGS and ICGS are able to produce a good enough computed Krylov basis to ensure the convergence of GMRES [4, 6].

In this paper, we wish to compare the efficiency of GMRES(m) with these four orthogonalization processes in a parallel distributed environment. It is well-known that the many scalar products arising in the orthogonalization phase are the bottleneck of Krylov based methods. Our tests are based on a restarted GMRES code developed at CERFACS which implements the four GS variants and uses reverse communication for the preconditioner, the matrix-vector products and dot products [5]. The numerical experiments have been performed on the 128 node CRAY T3D available at CERFACS, using the MPI message passing library. We present results concerning an application in astrophysics developed in collaboration of M. Rieutord (Observatoire Midi-Pyrénées, Toulouse) and L. Valdettaro (Politecnico di Milano). In order to explain in depth these results, we study a model problem deriving from a convection-diffusion equation, on which we can test the scalability of GMRES.

2 An application in astrophysics

2.1 Description of the application

The application we are interested in belongs to the class of flow stability problems and arises in astrophysics, when modelling the internal structure of stars and planets, to study for instance the electromagnetic field in the earth kernel [8, 9]. The study of the stability of a solution of a nonlinear problem begins by solving the eigenvalue problem verified by small perturbations of the original solution. Therefore, the inertial modes (or eigenmodes) for an incompressible viscous fluid located between two concentric spheres are obtained by solving the linearized Navier Stokes continuity equation

$$\begin{cases} E\Delta\boldsymbol{\nabla} \times \mathbf{u} - \boldsymbol{\nabla} \times (\mathbf{e_z} \times \mathbf{u}) = \lambda \boldsymbol{\nabla} \times \times \mathbf{u}, \\ \text{div } \mathbf{u} = 0 \end{cases}$$

where $\mathbf{u}$ is the scaled velocity of the fluctuations, and $(\mathbf{e_z} \times \mathbf{u})$ is the non dimensional Coriolis force. The Eckman number E is equivalent to the inverse of a scaled Reynolds number and is intended to be very small. When using a spherical geometry and after projecting on the space of Chebyshev polynomials, one obtains a generalized eigenproblem $Ax = \lambda Bx$ where A and B are two non hermitian matrices, and B may be singular. The smaller E, the finer must be the discretization and thus A and B must be larger.

All the eigenvalues have an imaginary part between -1 and 1. The eigenvalues of interest are those closest to the imaginary axis. The eigenproblem is solved by the Arnoldi method with a Chebyshev acceleration [2]. The strategy for selecting the interesting eigenvalues is based on a shift and invert technique, with several imaginary shifts σ in the interval $[-i, i]$. We then solve for μ the eigenproblem

$$(A - \sigma B)^{-1} Bx = \mu x$$

with $\mu = 1/(\lambda - \sigma)$ being the largest eigenvalue of $C = (A - \sigma B)^{-1}B$. The Arnoldi method, like any Krylov-based iterative method, requires only the application of C to a vector. Clearly in this case, this operation involves the solution of linear systems such as $(A - \sigma B)x = y$. Even if A and B are sparse, the use of direct methods for solving this system imposes severe limitations on the size of A and B, and consequently limits the range of possible values of the Eckman number E. We have therefore investigated the use of GMRES(m) in a parallel distributed memory environment.
After detailing some choices about the data distribution and the preconditioning, we give some results on the performance of GMRES(m) on this example, where, for sake of simplicity, we have assumed the shift σ to be zero. In the rest of this paper, the acronym GMRES will always refer to the restarted method.

2.2 The data distribution

The matrix A is tridiagonal by blocks. The size of the diagonal blocks is usually twice the size of the off-diagonal blocks (also called coupling blocks). For example, the test matrix A_{5850} used in our experiments is 5850×5850, with 45 diagonal blocks of size 130×130 and 88 off-diagonal blocks of size 61×61. Only the blocks are stored. They are dense enough so that there is no substantial gain to hope from a sparse storage. Note that the matrix B is diagonal by blocks, with a block size compatible with the storage of A: therefore the matrices $A - \sigma B$ can be stored identically as A.

The matrix is distributed over the processors so that each processor possesses complete blocks only. The advantage of this distribution is that it allows an easy implementation of the preconditioner but the drawback is that it may induce some load imbalance when the number of blocks is not a multiple of the number of processors. In the case of two processors for instance, processor 1 will receive the first 23 diagonal blocks of A_{5850} and processor 2 the remaining 22 blocks. In other words, processor 1 (resp., processor 2) will treat a subproblem of size $130 \times 23 = 2990$ (resp., $130 \times 22 = 2860$). But out of 32 processors, 13 will have two blocks whereas 19 processors will have half of the load that is one block only. Let $n(p)$ be the size of the subproblem treated by the first processor when the matrix is distributed over p processors. If the data distribution were equilibrated, $n(p)$ would be kept proportional to $1/p$. Figure 1 shows the evolution of the ratio $n(2)/n(p)$, where the optimal distribution (dotted line) would give $n(2)/n(p) = p/2$. We see that our distribution is reasonably balanced whenever $p \leq 16$. When $p \geq 32$, the first processor is penalized by a larger amount of data than expected with the increase in processors. The opposite phenomena arises for the last processor which sees its load decrease faster than the increase in processors.

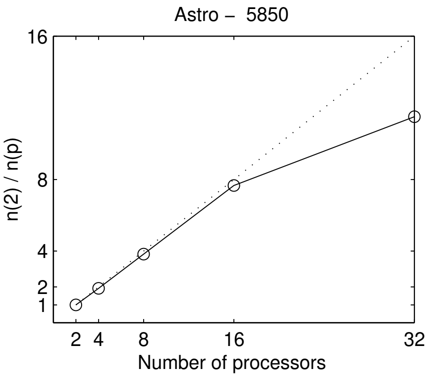


Fig. 1. Load of the first processor

2.3 Preconditioning

The restarted GMRES method applied on this problem does not converge even with large restarts. We have chosen one of the simplest preconditioner: the block Jacobi preconditioner. The preconditioner consists in the diagonal blocks of A and is well adapted to the data distribution. Each processor computes once a dense LU factorization of its blocks at the beginning of the computation. Applying the preconditioner consists in performing two triangular solves per block locally on each processor. There is no communication involved when building or applying the preconditioner.

2.4 Performance results

We describe now the results obtained for the matrix of size 5850. Since the number of processors on the CRAY T3D has to be a power of 2, we show results for 2, 4, 8, 16 and 32 processors (which is the maximum allowed for 45 blocks). The sequential code could not be run due to memory limitations: consequently, our reference for speed-ups will be the time obtained for 2 processors.

From a numerical point of view, the matrix A is very ill-conditioned (with a condition number larger than 10^{13}). GMRES with a CGS reorthogonalization does not converge on this example. In our tests, we stop the iterations when the backward error on the preconditioned system becomes lower than 10^{-7}. The original system then has a normwise backward error $\|A\tilde{x}-b\|_2/(\|A\|_2\|\tilde{x}\|_2+\|b\|_2)$ smaller than 10^{-10} (here $\tilde{x}$ denotes the computed solution). We will show the results obtained with a restart of 100 since we obtained similar behaviors for other values.

Because the matrix is ill-conditioned, the number of iterations may vary significantly with the orthogonalization strategy and the number of processors (see Figure 2). However, these variations are not monotonous and one cannot predict the preeminence of one method over the other. The variation with the number of processors is due to the different orderings of the floating-point operations in the parallel matrix-vector and dot products. Because of this sensitivity, we have chosen to evaluate the performance of the code mainly on one iteration rather than on the complete solution: the times measured over the solution are then scaled by the number of iterations required for this solution. However, to be fair, we first look at the total computational time needed to obtain the solution (see Figure 3). Regardless of the number of iterations, the fastest method is GMRES-ICGS and the slowest is GMRES-IMGS. The success of GMRES-ICGS is mainly due to the better scalability properties of the ICGS orthogonalization scheme, as we will see in the next section.

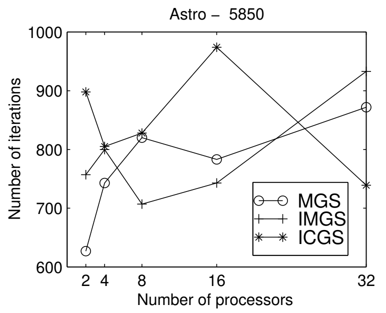


Fig. 2. Number of iterations required for convergence

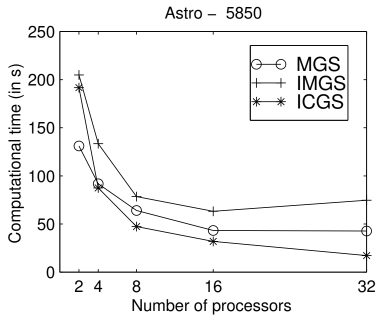


Fig. 3. Time for convergence

Speed-up for the preconditioner. The speed-up for the preconditioner is taken as the ratio

$$\frac{\text{preconditioning time on } p \text{ processors}}{\text{preconditioning time on 2 processors}}$$

and is plotted on Figure 4. We cannot compare with the sequential time because the problem is too large to fit on one processor. It departs from its optimal value (dotted line) when $p \geq 16$. This only reflects the imbalance of the processors load (see Figure 1) and, because there is no communication overhead, is the best one can expect from this data distribution.

Speed-up for the matrix-vector product. The matrix-vector product in parallel involves *a local part* corresponding to the contribution of the diagonal blocks and *a communication part* with the left and right neighbours to take into account the contribution of the coupling blocks. It is implemented so as to overlap communication and computation as much as possible. The speed-up, computed as for the preconditioning is shown on Figure 5. Here again, it reflects rather faithfully the distribution of the load on the processors (see Figure 1), which proves that the overhead due to communications does not penalize the computation.

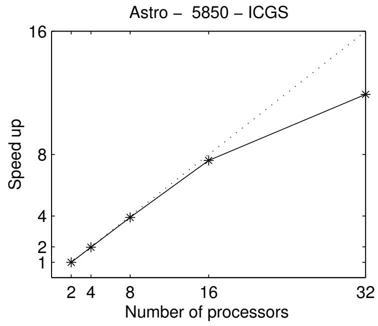


Fig. 4. Preconditioning

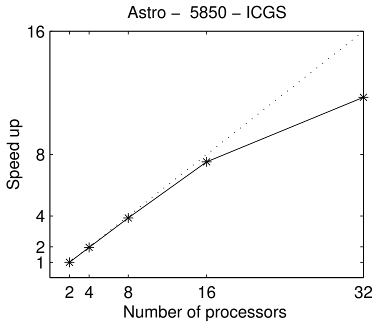


Fig. 5. Matrix-vector product

Influence of dot products. The dot products are the well-known bottleneck for Krylov methods in a distributed environment. Figure 6 shows the time spent, per iteration, in the dot products and Figure 7 gives the percentage of the solution time spent in the dot products. Both figures indicate clearly that GMRES with ICGS is the best method for avoiding the degradation of the performance generally induced by the scalar products. We even see on Figure 6 that, when $p \geq 32$, the time spent in iteration in the dot products starts increasing for IMGS and MGS whereas it continues to decrease with ICGS: this happens when the communication time overcomes the computation time in the dot products for IMGS and MGS. By gathering the scalar products, ICGS ensures more computational work to the processors.

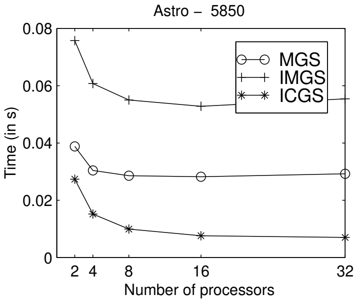


Fig. 6. Time spent in dot products per iteration

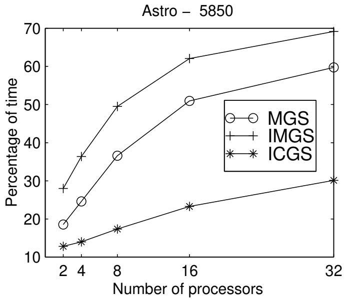


Fig. 7. Percentage of the solution time spent in dot products

Finally, Figure 8 gives the speed-up, for an average iteration, of the complete solution. When comparing with the best possible curve with the given processor load on Figure 1, we see that GMRES-MGS and GMRES-IMGS give bad performance whereas GMRES-ICGS is less affected by scalar products.

We now present a test problem for which it is easier to vary the size in order to test the parallel scalability properties of GMRES.

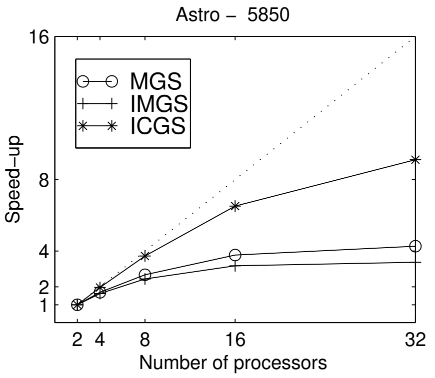


Fig. 8. Speed-up for an average iteration

3 Scalability on a model problem

We intend to investigate the parallel scalability of the GMRES code and the influence of the orthogonalization schemes. For this purpose we consider the solution, via two classic domain decomposition techniques, of an elliptic equation

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = f \tag{1}$$

on the unit square $\Omega = (0,1)^2$ with Dirichlet boundary conditions.

Assume that the original domain Ω is triangulated by a set of non-overlapping coarse elements defining the N sub-domains Ω_i.

We first consider an additive Schwarz preconditioner that can be briefly described as follows. Each substructure Ω_i is extended to a larger substructure $\Omega_i^{'}$, within a distance δ from Ω_i, where δ refers to the amount of overlap. Let $A_i^{'}$ denote the discretizations of the differential operator on the sub-domain $\Omega_i^{'}$. Let R_i^T denote the extension operator which extends by zero a function on Ω_i onto Ω, and R_i the corresponding pointwise restriction operator. With these notations the additive Schwarz preconditioner, M_{AD}, can be compactly described as

$$M_{AD}u = \sum R_i^T A_i^{'-1} R_i u.$$

We also consider the class of domain decomposition techniques that use non-overlapping sub-domains. The basic idea is to reduce the differential operator on the whole domain to an operator on the interfaces between the sub-domains.

Let I denote the union of the interior nodes in the sub-domains, and let B denote the interface nodes separating the sub-domains. Then grouping the unknowns corresponding to I in the vector u_I and the unknowns corresponding to B in the vector u_B, we obtain the following reordering of the problem:

$$Au = \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} u_I \\ u_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix}. \tag{2}$$

For standard discretizations, A_{II} is a block diagonal matrix where each diagonal block, A_i, corresponds to the discretization of (1) on the sub-domain Ω_i.

Eliminating u_I in the second block row of (2) leads to the following reduced equation for u_B:

$$Su_B = g_B = f_B - A_{BI}A_{II}^{-1}f_I, \tag{3}$$

where

$$S = A_{BB} - A_{BI}A_{II}^{-1}A_{IB}.$$

S is referred to as the Schur complement matrix (or also the capacitance matrix). In our experiments, Equation (1) is discretized using finite elements. The Schur complement matrix can then be written as

$$S = \sum_{i=1}^{N} S^{(i)} \tag{4}$$

where $S^{(i)}$ is the contribution from the i^{th} sub-domain with $S^{(i)} = A^{(i)}_{BB} - A^{(i)}_{BI}(A^{(i)}_{II})^{-1}A^{(i)}_{IB}$ and $A^{(i)}_{XX}$ denotes submatrices of A_i

Without more sophisticated preconditioners, these two domain decomposition methods are not numerically scalable [12]; that is the number of iterations required grows significantly with the number of sub-domains. However, in this section we are only interested in the study of the influence of the orthogonalization schemes on the scalability of the GMRES iterations from a computer science point of view. This is the reason why we selected those two domain decomposition methods that exhibit a large amount of parallelism and we intend to see how their scalability is affected by the GMRES solver. For a detailed overview of the domain decomposition techniques, we refer to [12].

For both parallel domain decomposition implementations, we allocate one sub-domain to one processor of the target distributed computer. To reduce as much as possible the time per iteration we use an efficient sparse direct solver from the Harwell library [7] for the solution of the local Dirichlet problems arising in the Schur and Schwarz methods.

To study the scalability of the code, we keep constant the number of nodes per sub-domain when we increase the number of processors. In the experiments reported in this paper, each sub-domain contains 64×64 nodes. For the additive Schwarz preconditioner, we selected an overlap of one element (i.e. $\delta = 1$) between the sub-domains that only requires one communication after the local solution A'^{-1} while one more communication would be necessary before the solution for a larger overlap.

In Figure 9 are displayed the elapsed time for both the Schur and the Schwarz approaches observed on a 128 node Cray T3D. If the parallel code were perfectly scalable, the elapsed time would remain constant when the number of processors is increased.

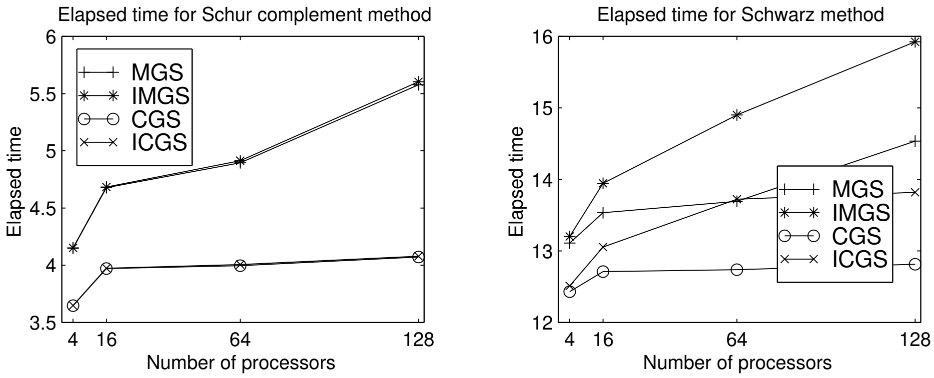
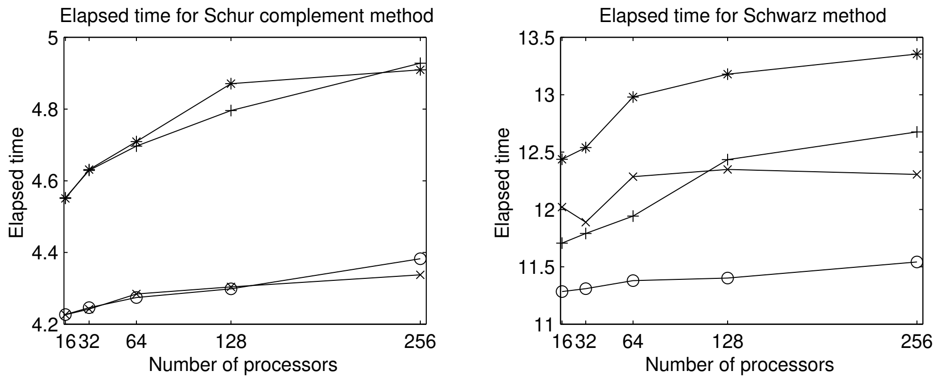


Fig. 9. Elapsed time



$+$: MGS - $*$: IMGS - $\circ$: CGS - $\times$: ICGS

Fig. 10. Elapsed time

We define also the scaled speed-up by

$$SU_p = p \times \frac{T_4}{T_p}$$

where T_ℓ is the elapsed time to perform a complete restart step of GMRES(50) on ℓ processors. For both Schur and the Schwarz approaches we report in Figure 11 the scaled speed-ups associated with the elapsed time displayed in Figure 9.

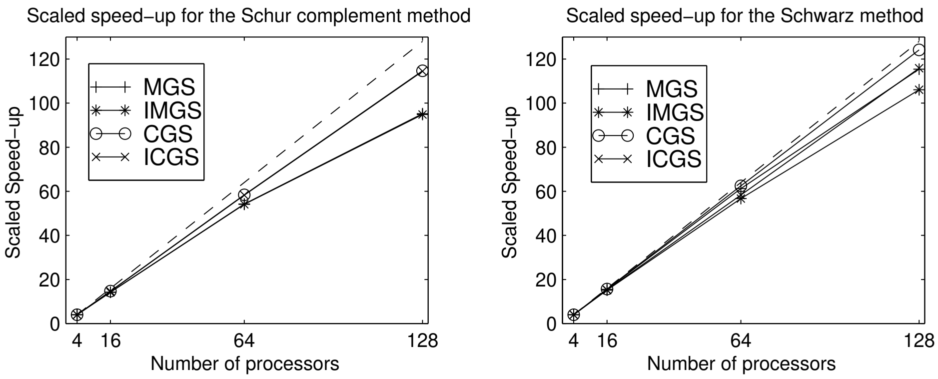
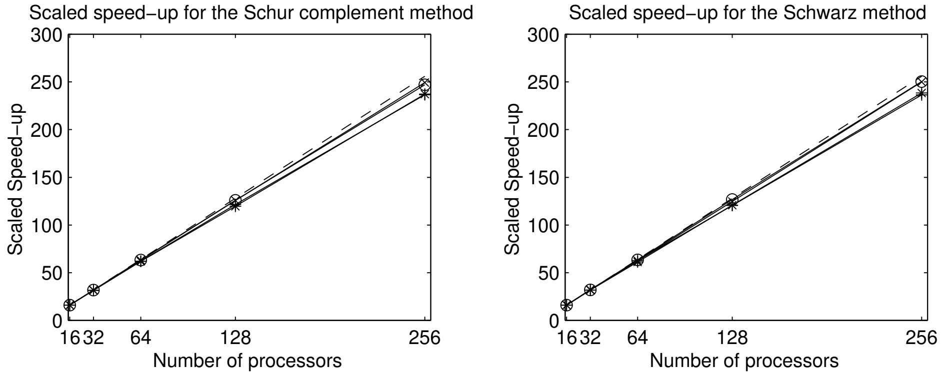


Fig. 11. Scaled speed-ups

As it can be seen in Figure 9, the solution of the Schur complement system does not require iterative orthogonalization; the curves associated with CGS/ICGS and MGS/IMGS perfectly overlap each other. In such a situation, the CGS/ICGS orthogonalizations are more attractive than MGS/IMGS. They are



$+$: MGS - $*$: IMGS - $\circ$: CGS - $\times$: ICGS

Fig. 12. Scaled speed-ups

faster and exhibit a better parallel scalability as it can be seen in the left picture of Figure 11. On 128 nodes the scaled speed-up is equal to 114 for CGS/ICGS and only 95 for MGS/IMGS. When iterative orthogonalization is required, as for the Schwarz method on our example for instance, CGS is the fastest and the most scalable but may lead to a loss of orthogonality in the Krylov basis resulting in a poor and even a loss of convergence. In that case ICGS offers the best trade-off between numerical robustness and parallel efficiency. Among the numerically reliable orthogonalization schemes, ICGS gives rise to the fastest and the most scalable iterations.

4 Conclusion

The choice of the orthogonalization scheme is crucial to obtain good performance from Krylov-based iterative methods in a parallel distributed environment. From a numerical point of view, CGS should be discarded together with MGS in some eigenproblem computations [3]. Recent works have proved that for linear systems, MGS, IMGS and ICGS ensure enough orthogonality to the computed basis so that the method converges. Finally, in a parallel distributed environment, ICGS is the orthogonalization method of choice because, by gathering the dot products, it reduces significantly the overhead due to communication.

References

1. Å. Björck. Numerics of Gram-Schmidt orthogonalization. *Linear Algebra Appl.*, 197–198:297–316, 1994.
2. T. Braconnier, V. Frayssé, and J.-C. Rioual. ARNCHEB users' guide : Solution of large non symmetric or non hermitian eigenvalue problems by the Arnoldi-Tchebycheff method. Tech. Rep. TR/PA/97/50, CERFACS, 1997.

3. F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations.* SIAM, Philadelphia, 1996.
4. J. Drkošová, A. Greenbaum, Z. Strakoš, and M. Rozložnik. Numerical stability of GMRES. *BIT*, 35, 1995.
5. V. Frayssé, L. Giraud, and S. Gratton. A set of GMRES routines for real and complex arithmetics. Technical Report TR/PA/97/49, CERFACS, 1997.
6. A. Greenbaum, Z. Strakoš, and M. Rozložnik. Numerical behaviour of the modified Gram-Schmidt GMRES implementation. *BIT*, 37:707–719, 1997.
7. HSL. *Harwell Subroutine Library. A Catalogue of Subroutines (Release 12).* AEA Technology, Harwell Laboratory, Oxfordshire, England, 1995.
8. M. Rieutord. Inertial modes in the liquid core of the earth. *Phys. Earth Plan. Int.*, 90:41–46, 1995.
9. M. Rieutord and L. Valdettaro. Inertial waves in a rotating spherical shell. *J. Fluid Mech.*, 341:77–99, 1997.
10. Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.
11. J. N. Shadid and R. S. Tuminaro. A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine. *SIAM J. Sci. Comp.*, 14(2):440–459, 1994.
12. B.F. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1st edition, 1996.