Van der Vorst, H.A.; Chan, T.F.

Parallel Preconditioning for Sparse Linear Equations

Abstract

A popular class of preconditioners is known as incomplete factorizations. They can be thought of as approximating the exact LU factorization of a given matrix A (e.g. computed via Gaussian elimination) by disallowing certain fill-ins. As opposed to other PDE-based preconditioners such as multigrid and domain decomposition, this class of preconditioners are primarily algebraic in nature and can in principle be applied to any sparse matrices. In this paper we will discuss some new viewpoints for the construction of effective preconditioners.

In particular, we will discuss parallelization aspects, including re-ordering, series expansion and domain decomposition techniques. Generally, this class of preconditioner does not possess a high degree of parallelism in its original form. Re-ordering and approximations by truncating certain series expansion will increase the parallelism, but usually with a deterioration in convergence rate. Domain decomposition offers a compromise.

1. Introduction

The general problem of finding a preconditioner for a linear system $Ax = b$ is to find a matrix M (the *preconditioner*) with the properties that M is a good approximation to A in some sense and that the system $Mx = b$ is much easier to solve than the original system. We will then solve the *preconditioned system* $M^{-1}Ax = M^{-1}b$ (or the symmetric version $(M^{-1/2}AM^{-1/2})(M^{1/2}x) = M^{-1/2}b$ when both A and M are symmetric positive definite) by standard iterative methods such as the conjugate gradient method, in which only the actions of A and M^{-1} are needed.
The choice of M varies from purely "black box" algebraic techniques which can be applied to general matrices to "problem dependent" preconditioners which exploits special features of a particular problem class. Obviously, the more special features we know and can exploit in a problem, the better we can construct a preconditioner for it. However, there is still a practical need for preconditioning techniques that can be applied to a general matrix. Incomplete factorization preconditioners are among the candidates for this. They can be thought of as modifications of Gaussian Elimination in which certain fill-ins are disallowed in order to ensure that the action of M^{-1} is inexpensive. These preconditioners can also be viewed as a bridge between direct methods such as Gaussian Elimination and classical relaxation methods such as Jacobi, Gauss-Seidel and SOR.
We will discuss some approaches that have been proposed in the literature to efficiently implement the incomplete factorization methods on parallel computers. A shorter survey can also be found in [8].

There are two general approaches for parallelizing numerical methods:

1. extract maximal parallelism from a method which works well on sequential computer, *without* changing its numerical properties,

2. modify or approximate a good sequential method to increase the parallelism available, thus possibly degrading its numerical properties.

There is a fundamental difficulty when applying the above general principles to incomplete factorization methods. The major parallel bottleneck lies in the backsolves involving the triangular LU factors. The same bottleneck arises in computing the LU factors themselves but this occurs only once in the beginning of the iteration. Even though these backsolves possess some degree of parallelism which can be exploited, this is often not sufficient to efficiently exploit many parallel architectures, especially massively parallel ones. On the other hand, modifying or approximating the sequential method in order to increase the amount of parallelism invariably leads to slower convergence rates. This should not be too surprising; it is just an instance of the fundamental trade-off between parallelism (which prefers locality) and fast convergence rate (which prefers global dependence) which governs many genuinely globally coupled systems (e.g. elliptic PDEs). The goal is to make the right trade-off for a given architecture/algorithm configuration.

There are three basic methodologies to extract or increase the parallelism in ILU methods: re-ordering, series expansions (including polynomial preconditioners), and domain decomposition. We shall briefly discuss them next.

We shall assume that A comes from finite difference or finite element discretization of a pde over a regular grid, and our discussions will focus on point ILU as our example. We note that the issue of parallelization are the same for both ILU and MILU since the data dependence are identical.

2. Re-ordering

The Hyperplane Ordering: Assume that the ILU factors have been computed and consider now the task of computing the product $y = L^{-1}v$. The goal is to find an ordering with which the components of y can be computed with maximal parallel efficiency. This is accomplished by the so-called hyperpane or wavefront ordering. A similar ordering can be used for computing $U^{-1}v$.

It is obvious that if one generalizes this idea to a $d-$dimensional grid with n grid points in each direction, we can extract $O(n^{d-1})$ degree of parallelism. If the number of parallel processors is of the same order, then one can achieve good parallel efficiency. Thus, for a fixed number of processors, higher dimensional problems are easier to parallelise. On the other hand, potential degradation in performance can be caused by memory addressing with unequal stride for $d > 2$ and cache problems with the indirect addressing needed to access data on the hyperplanes when the grid is stored as a 2D array.

The use of hyperplane ordering has been investigated by Radicati and Vitaletti [12] for the IBM 3090. Numerical experiments on the CM2 can be found in Berryman et al [2]. The hyperplane ordering for regular 3D grids has been described in detail for vector computers in van der Vorst [19]. Recently, Barszcz et al [1] gave a data mapping for the hyperplane ordering in 3D useful for distributed memory architectures and compare the performance on an 8-processor Cray Y-MP, a 128 processor Intel iPSC/860 and a 32K processor CM-2.

Multi-color Orderings: Since the degree of parallelism for ILU methods are limited in the natural ordering, a popular alternative is to use orderings that are designed to be more parallel. However, it must be emphasized that most of these orderings are *not* equivalent to the natural ordering, in the sense that the ILU factors computed using these are generally different from those generated using the natural ordering. Thus, the goal is to tradeoff the relatively fast and well-understood convergence rate of the natural ordering for orderings with a high degree of parallelism.

An example is the well-known red-black ordering for 5-point stencils in 2D. Because the red points depend only on the black points but not on each other, they can all be updated in parallel. Thus the degree of parallelism is $n^2/2$, a substantial increase from $O(n)$ for the natural ordering. However, since the data dependence are completely local and there is no global sharing of information, the convergence rate is poor. In fact, Kuo and Chan [10] proved that the condition number of the preconditioned system in the red-black ordering is only about 1/4 that of the *unpreconditioned* system for ILU, MILU and SSOR, with no asymptotic improvement as h tends to zero.

One way to strike a better balance between parallelism and fast convergence is to use more colors [6]. In principle, since the different colors are updated sequentially, using more colors decreases the parallelism but increases the global dependence and hence the convergence. The key is to choose the number of colors to match the architecture. For example, Doi and Hoshi [7] used up to 75 colors for a 76^2 grid on the NEC SX-3/14 and achieved 2 Gflops performance, which is much better than that for the hyperplane ordering.

Multi-wavefront Orderings: A different approach to increase parallelism is to use several hyperplane wavefronts to sweep through the grid, the idea being that all wavefronts can be updated in parallel. For example, van der Vorst [16] considered starting wavefronts from each of the four corners in a 2D rectangular grid, or from the eight corners in a 3D grid. Earlier, Meurant [11] used a similar idea in which the grid is divided into equal parts (e.g. halves or quadrants) and each part is ordered in its own natural ordering.

3. Series Expansions

Instead of using an ordering with more parallelism, a quite different approach to increase the parallelism in the naturally ordered ILU method is to replace it by an *approximation* which can be evaluated more efficiently in parallel.

In order to illustrate this, consider the computation of $(I - L)^{-1}v$, which is needed in applying the preconditioner. Here we have assumed without loss of generality that the diagonal entries of the lower triangular factor has been scaled to unity. It can be easily proved that if the spectral radius $\rho(L)$ satisfies $\rho(L) < 1$ and L is n by n strictly lower triangular, then we have the following two finite expansions:

Neumann Expansion: $(I - L)^{-1}v = (I + L + L^2 + ... + L^{n-1})v,$

Euler Expansion: $(I - L)^{-1}v = (I + L^{2^s})(I + L^{2^{s-1}})...(I + L)v,$

where $s = \lceil \log_2 n - 1 \rceil$. Note that $L^n = 0$. Each of the terms on the right-hand-side of the above expansions can be evaluated in parallel efficiently because they only involve repeated multiplication of v by sparse matrices. The idea is to then truncate the expansions but keeping enough terms so that the convergence rate is not too adversely affected.
Van der Vorst [15] used this idea when he applied a truncated Neumann expansion to the diagonal blocks (which correspond to grid lines) in the point ILU factorization in order to increase the degree of vectorization. In the same paper, he also used the Euler expansion (of low order).

Finally, a related method is the class of *polynomial preconditioners* in which A^{-1} is approximated by a low degree polynomial in A, chosen in some optimal manner. In [9] it is shown how GMRES can often be effectively preconditioned by a Chebyshev matrix polynomial, for which the coefficients are obtained from eigenvalue approximations from a limited number of GMRES steps. In particular, the harmonic Ritz values have been employed as useful approximations and a surprisingly simple algorithm is presented in [9] for the computation of the Chebyshev parameters of the Chebyshev polynomial over a piecewise linear contour that encloses the eigenvalue approximations. Using this type of relatively expensive polynomial preconditioners often leads to a significant reduction in GMRES steps and hence the required communication-intensive innerproducts have a lesser degrading effect on the parallel performance of the preconditioned GMRES algorithm on distributed memory machines. For an analysis of the effects of communication in GMRES, see [5].

4. Domain Decomposition

In this general approach, the physical domain or grid is decomposed into a number of overlapping or non-overlapping subdomains on each of which an independent incomplete factorization can be computed and applied in parallel. The main idea is to obtain more parallelism at the subdomain level rather than at the grid point level. Usually, the interfaces or overlapping region between the subdomains must be treated in a special manner. The advantage of this approach is that it is quite general and can be used with different methods used within different subdomains.
Radicati and Robert [13] used an algebraic version of this approach by computing ILU factors within overlapping block diagonals of a given matrix A. When applying the preconditioner to a vector v, the values on the overlapped region is averaged from the two values computed from the two overlapping ILU factors.
The approach of Radicati and Robert has been further perfectioned by De Sturler [4], who studies the effects of overlap from the point of view of geometric domain decompositioning. He introduces artificial mixed boundary conditions on the internal boundaries of the subdomains. In [4]:Table 5.8 experimental results are shown for a decomposition in 20×20 slightly overlapping subdomains of a 200×400 mesh for a discretized convection-diffusion equation (5-point stencil). When taking ILU preconditioning for each subdomain, it is shown that the complete linear system can be solved by GMRES on a 400-processor distributed memory Parsytec system with an efficiency in the order of 80% (compared with $\frac{1}{400}$-th of the CPU time of ILU preconditioned GMRES for the unpartitioned system on 1 single processor).

In [14], Tan studies the interface conditions along subdomains and forces continuity for the solution at the interface up to some degree. He proposes to include also mixed derivatives in these relations. The involved parameters can be determined locally by mmeans of normal mode analysis, and they are adapted to the discretized problem. It is shown that the resulting domain decomposition method defines a standard iterative method for some splitting $A = M - N$, and the llocal coupling aimes at minimizing the largest eigenvalues of $I - AM^{-1}$. Of course, this method can be accelerated and impressive results for GMRES acceleration are shown in [14]. Some attention is paid to the case where the solutions for the subdomains are obtained in only modest accuracy per iteration step.

Chan and Goovaert [3] showed that the domain decomposition approach can actually lead to *improved* convergence rates, at least when the number of subdomains is not too large. The reason derives from the well-known divide and conquer effect when applied to methods with superlinear complexity such as ILU: it is more efficient to applied such methods to smaller problems and piece the global solution together.

Recently, Washio and Hayami [17] employed a domain decomposition approach for a rectangular grid by which one step of SSOR is done for the interior part of each subdomain. In order to make this domain-decoupled SSOR more resemble the global SSOR, the SSOR iteration matrix for each subdomain is modified by premultiplying it with a matrix $(I - X_L)^{-1}$ and postmultiplying it by $(I - X_U)^{-1}$. The matrices X_L and X_U depend on the couplings between adjacent subdomains. In order to further improve the parallel performance, the inverses are approximated by low-order truncated Neumann series. A similar approach is suggested in [17] for a block MILU preconditioner. Experimental results have been shown for a 32-processor NEC-Cenju distributed memory computer.

5. References

1 E. Barszcz, R. Fatoohi, V. Venkatakrishnan, and S. Weeratunga: Triangular systems for CFD applications on parallel architectures; Technical report, NAS Applied Research Branch, NASA Ames Research Center, 1994.

2 H. Berryman, J. Saltz, W. Gropp, and R. Mirchandaney: Krylov methods preconditioned with incompletely factored matrices on the CM-2; Technical Report 89-54, NASA Langley Research Center, ICASE, Hampton, VA, 1989.

3 T.F. Chan and D. Goovaerts: A note on the efficiency of domain decomposed incomplete factorizations; SIAM J. Sci. Stat. Comp., **11** (1990), 794–803.

4 E. de Sturler: Iterative methods on distributed memory computers; Ph.D. Thesis, Delft University, the Netherlands, 1994.

5 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; to appear in J. Appl. Num. Math.

6 S. Doi: On parallelism and convergence of incomplete LU factorizations; App. Numer. Math., **7**, (1991), 417–436.

7 S. Doi and A. Hoshi: Large numbered multicolor MILU preconditioning on SX-3/14; Int'l J. Computer Math., **44** (1992), 143–152.

8 J.J. Dongarra, I.S. Duff, D.C. Sorensen, and Henk A. van der Vorst: Solving Linear Systems on Vector and Shared Memory Computers; SIAM, Philadelphia, PA, 1991.

9 M.B. van Gijzen: Iterative solution methods for linear equations in finite element computations; Ph.D. Thesis, Delft University, the Netherlands, 1994.

10 J.C.C. Kuo and T.F. Chan: Two-color Fourier analysis of iterative algorithms for elliptic problems with red/black ordering; SIAM J. Sci. Stat. Comp., **11** (1990), 767–793.

11 G. Meurant: Domain decomposition methods for partial differential equations on parallel computers; Int. J. Supercomputing Appls., **2** (1988), 5–12.

12 G. Radicati di Brozolo and M. Vitaletti: Sparse matrix-vector product and storage representations on the IBM 3090 with Vector Facility; Technical Report 513-4098, IBM-ECSEC, Rome, July 1986.

13 G. Radicati di Brozolo and Y. Robert: Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor; Parallel Comput., **11** (1989), 223–239.

14 K.H. Tan: Local coupling in domain decomposition; Ph.D. Thesis, Utrecht University, the Netherlands, 1995.

15 H.A. van der Vorst: A vectorizable variant of some ICCG methods; SIAM J. Sci. Stat. Comput., **3** (1982), 350–356.

16 H.A. van der Vorst: High performance preconditioning; SIAM J. Sci. Stat. Comput., **10** (1989), 1174–1185.

17 T. Washio and K. Hayami: Parallel block preconditioning based on SSOR and MILU; Numer. Lin. Alg. with Applic., **2** (1994).

Addresses: Prof. Dr. Henk van der Vorst, Mathematical Institute, University of Utrecht, P.O. Box 80010, 3508 TA Utrecht, the Netherlands.

Prof. Dr. Tony F. Chan, Department of Mathematics, University of California at Los Angeles, CA 90024–1555, USA.