IMPLEMENTATION AND EVALUATION OF VAIDYA'S PRECONDITIONERS

DORON CHEN AND SIVAN TOLEDO

1. Introduction

A decade ago Pravin Vaidya proposed an intriguing family of preconditioners for M-matrices [4]. He presented his ideas in a scientific meeting but never published a paper on the topic. His preconditioners were never implemented or tested experimentally. We have implemented Vaidya's preconditioners. We experimentally compare the effectiveness of Vaidya's preconditioners to that of incomplete-factorization preconditioners, including no-fill and drop-tolerance preconditioners, both modified and unmodified.

Our results indicate that Vaidya's preconditioners are often superior to drop-tolerance incomplete-factorization preconditioners, including modified and unmodified, and with various matrix orderings. Vaidya's preconditioners are particularly effective on difficult 2D problems (e.g. isotropic elliptic problems with Neumann boundary conditions). Vaidya's preconditioners are also more robust. They are not as effective on 3D problems.

Vaidya proposed constructing a preconditioner M for a symmetric M-matrix A by dropping off-diagonal nonzeros from A and factoring M (completely). An input parameter t controls how many edges are dropped, and hence, the effectiveness and cost of the resulting preconditioner. John Gilbert coined the term *complete factorization of incomplete matrices* to describe such preconditioners, as opposed to conventional incomplete factorizations of the complete matrix. Vaidya proposed a sophisticated dropping algorithm that balances the amount of fill in M with the condition number of the preconditioned matrix $M^{-1/2}AM^{-1/2}$. Perhaps the most remarkable aspect of Vaidya's preconditioners is that for many classes of M-matrices, the condition number of the preconditioned matrix depends only on the size of A and is independent of the condition number $\kappa(A)$ of A.

Vaidya stated (without proofs; for proofs, see [1]) several theoretical condition-number bounds for his preconditioners. We omit them from this abstract due to lack of space.

We have evaluated Vaidya's preconditioners by experimentally comparing their performance to that of drop-tolerance incomplete-Cholesky preconditioners (ICC), in the context of a conjugate gradients iterative solver. Both families of preconditioners accept a parameter that indirectly influences the sparsity of the preconditioner: t in the case of Vaidya and the drop tolerance in the case of ICC. We always compare preconditioners with the same amount of fill.

The rest of this paper is organized as follows. Section 2 describes the setup, methodology, and results of our experiments. We summarize our conclusions in Section 3. The details of the construction of Vaidya's preconditioners are omitted from this conference abstract due to lack of space. The full paper includes them.

2. Experimental Results

2.1. Methodology.

Both Vaidya's preconditioners and drop-tolerance incomplete-Cholesky preconditioners accept a parameter that indirectly affects the sparsity of the preconditioner. We always compare preconditioners with similar levels of fill (i.e., preconditioners that take up similar amounts of memory and similar amounts of work to construct and to apply).

We measure the amount of fill in the factors of preconditioners in terms of *fill ratios*. We define the fill ratio of a preconditioner $M = LL^T$ to be the ratio between the number of nonzeros in L and $2n-1$, which is the minimal number of nonzeros in Vaidya's preconditioners.

2.2. Experimental Setup.

The experiments were conducted on a dual-processor 600Mhz Pentium III computer with 2GBytes of main memory. We only use one processor.

We use METIS version 4.0 [2, 3] to find fill-reducing orderings. Since our experiments use matrices whose graphs are regular meshes in 2 and 3 dimensions, we also run incomplete Cholesky with the natural ordering of the mesh. On modified ICC preconditioners, we use only the natural ordering; METIS ordering breaks down the modified incomplete factorization.

We implemented a sparse Cholesky factorization algorithm specifically for this project. The code is roughly 6 times slower than a state-of-the-art code. The full paper explains why we used this code and estimates the running times with a state-of-the-art factorization code. We omit the analysis from this abstract.

The iterative solver that we use is preconditioned conjugate gradients.

The matrices that we use for our experimental analysis are discretizations of elliptic PDEs on regular 2- and 3-dimensional meshes. The matrices all arise from finite-differences discretizations of the equation

$$c_x \frac{\partial^2 u}{\partial x^2} + c_y \frac{\partial^2 u}{\partial y^2} = f \qquad \text{in } \Omega =]0,1[\times]0,1[$$

with either Dirichlet or Neuman boundary conditions. We solve isotropic problems ($c_x = c_y = 1$) and unisotropic problems in which either $c_x = 100$ and $c_y = 1$ or vice

This research was supported by Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (grant number 572/00 and grant number 9060/99) and by the University Research Fund of Tel-Aviv University.

versa. We also solve similar problems in 3D. We use a five-point discretization in 2D and a seven-point discretization in 3D, which lead to a pentadiagonal matrix when a 2D mesh is ordered row-by row (the so-called natural order) or to a septadiagonal matrix in 3D.

2.3. **Experimental Analysis.** Both ICC and Vaidya solve problems fastest when the preconditioner is allowed to fill somewhat. If little or no fill is allowed, most of the solution time is spent performing a large number of iterations. If the preconditioner is allowed to fill considerably, most of the solution time is spent factoring the preconditioner. Figure 2.1 clearly shows that low and high fill ratios lead to slower solution times than medium fill ratios. The figure also shows that the solution time of the faster preconditioners, Vaidya and MICC, is not sensitive to the fill ratio within the range 3–6. Therefore, we used fill ratio 5 in the rest of the experiments.

In particular, Vaidya's maximum spanning tree preconditioner (with no additional edges) is ineffective. It requires a huge number of iterations, as demonstrated by our experiments and shown in Figures 2.1 and 2.3.

Figure 2.2 shows that when both Vaidya and ICC preconditioners are allowed to fill appropriately (that is, to achieve near-optimal solution times), Vaidya's preconditioners converge in similar or smaller numbers of iterations. The next best preconditioner in our experiments is always modified ICC with the natural ordering. (We always use METIS to order Vaidya's preconditioners unless their graph is a tree.) Unmodified ICC preconditioners are significantly worse than Vaidya's and MICC.

Vaidya's preconditioners are not sensitive to the boundary conditions that we impose, but ICC preconditioners are. Vaidya's preconditioners converge in roughly the same number of iterations on both Neumann and Dirichlet boundary conditions, but ICC preconditioners are less effective on Neumann boundary conditions. This is shown both by Figure 2.2 and by Figure 2.3.

The scaling behavior of Vaidya's preconditioner is remarkable. Figure 2.2 shows that the number of iterations grows very slowly with the size of the matrix. Note that the size of the preconditioner (i.e., the number of nonzeros in its factors) remains a constant fraction of the size of A. Therefore, these preconditioners combine linear scaling of the work per iteration with a slow growth in the number of iterations, a highly desirable behavior.

Vaidya's preconditioners are unaffected by the original ordering of the matrix and are capable of automatically exploiting numerical features. On unisotropic problems, Vaidya's preconditioners are almost completely unaffected by whether the direction of strong influence is the x or the y direction, as shown in Figure 2.3. The construction of Vaidya's preconditioners starts with a maximum spanning tree, which always include all the edges (nonzeros) along the direction of strong influence. They are, therefore, capable of exploiting the fact that there is a direction of strong influence and they lead to faster convergence than on an isotropic problem. ICC preconditioners, on the other hand, are sensitive to the interaction between the elimination ordering and the direction of strong influences. Figure 2.3 shows that naturally-ordered MICC converges faster for one direction of strong influence than for the other when the matrix ordering is fixed. Therefore, to achieve fast convergence with MICC the user would need to tailor the ordering to the numerics, whereas in Vaidya's preconditioners this adaptation occurs automatically.

Vaidya's preconditioners are not as effective as ICC preconditioners on 3D problems; the full paper contains experimental results, which we omit here.

3. Conclusions

Vaidya's preconditioners are efficient and robust. More specifically, we draw the following conclusions from this research:

1. Vaidya's preconditioners are robust when used to solve linear systems arising from finite-differences discretizations of 2D elliptic problems. Their construction and convergence are not affected by the boundary conditions. They effectively and automatically exploit unisotropy. They are not affected by the original ordering of the unknowns. In all of these respects, they are better than all the drop-tolerance preconditioners that we have tested.
2. On some of these 2D problems, Vaidya's preconditioners lead to faster solution times than all the ICC preconditioners, including modified ICC. Specifically, Vaidya's preconditioners are faster on 2D problems with Neumann boundary conditions.
3. Vaidya's preconditioners do not appear to be competitive with ICC preconditions on 3D problems.
4. Vaidya's maximum-spanning-tree preconditioners are sparser than ICC(0) preconditioners, but they converge more slowly. We recommend some fill in Vaidya's preconditioners.

The full paper is available at `www.tau.ac.il/~stoledo/pubs.html`. We will also make our code publicly available.

References

[1] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. Technical report, School of Computer Science, Tel-Aviv University, 2001.

[2] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[3] George Karypis and Vipin Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, 48:71–85, 1998.

[4] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis.

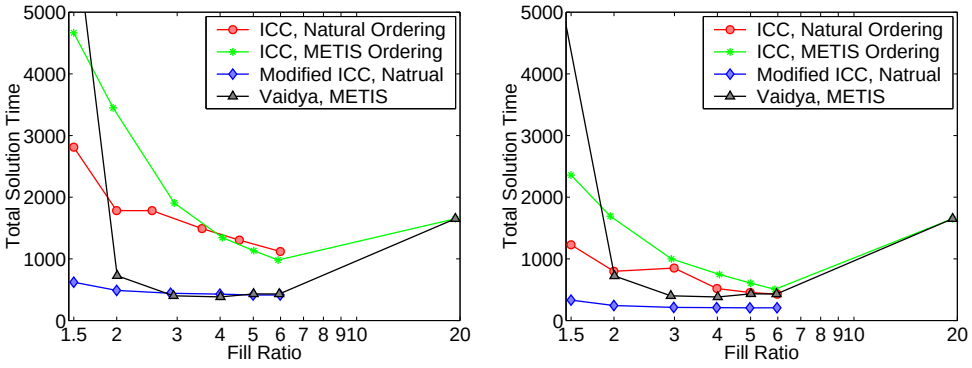

FIGURE 2.1. Total solution times with ICC and Vaidya's preconditioners on 1200-by-1200 2D isotropic problems with Neumann (left) and Dirichlet (right) boundary conditions. The graphs show the total solution times (construction, factorization, ordering, and iterations) as a function of the fill ratio of the preconditioners. The iterative solver stops when the 2-norm of the residual drops by a factor of 10^8. The solution times with a fill-ratio-1 Vaidya's preconditioner are high, outside the scale of the graphs. The rightmost data point in both graphs represents a complete factorization of A with METIS ordering.

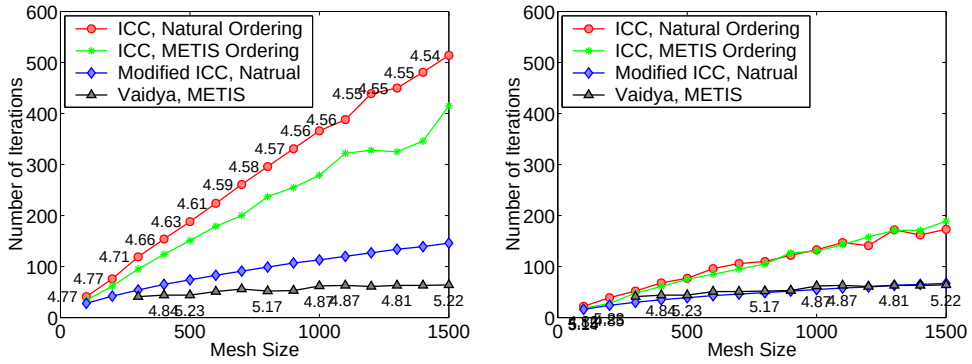

FIGURE 2.2. Convergence of fill-ratio-5 ICC and Vaidya's preconditioners on 2D isotropic problems with Neumann (left) and Dirichlet (right) boundary conditions. The matrix of a size-$\sqrt{n}$ mesh is n-by-n. The numbers near the graphs show the actual fill ratio of the preconditioners when it deviated from 5 by more than 2.5%.

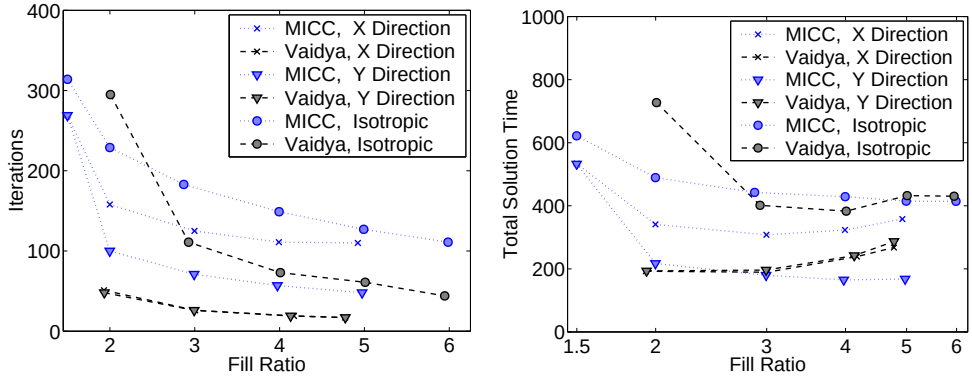

FIGURE 2.3. Numbers of iterations (left) and solution times (right) of Vaidya and MICC on unisotropic 1200-by-1200 2D model problems with Neumann boundary conditions. Vaidya's preconditioners are ordered using METIS whereas MICC uses the same natural ordering for all the problems.

SCHOOL OF COMPUTER SCIENCE, TEL-AVIV UNIVESITY, TEL-AVIV 69978, ISRAEL
E-mail address: stoledo@tau.ac.il
URL: http://www.tau.ac.il/~stoledo