Journal of Algorithms **35,** 122–126 (2000)
doi:10.1006/jagm.1999.1069, available online at http://www.idealibrary.com on IDEAL®

NOTE

Processor Efficient Parallel Solution of Linear Systems of Equations

Gilles Villard

LMC–IMAG, B.P. 53 F38041 Grenoble Cedex 9, France

Received February 10, 1998

We present a deterministic parallel algorithm that solves a n-dimensional system $Ax = b$ of linear equations over an ordered field or over a subfield of the complex numbers. This algorithm uses $O(\log^2 n)$ parallel time and $O(\max\{\mathrm{M}(n), n^2 (\log\log n)/\log n\})$ arithmetic processors if $\mathrm{M}(n)$ is the processor complexity of fast parallel matrix multiplication.

Key Words: fast parallel algorithm; processor-efficient algorithm; linear system solution; Krylov subspace; Toeplitz matrix.

1. INTRODUCTION

Solving an $n \times n$ linear system $Ax = b$ over a field F is a major computational problem. It is still of interest to know its parallel complexity and especially to decrease the work of known algorithms. The work of a parallel algorithm is the product of its running time by the number of processors it utilizes. For an n-dimensional input, an algorithm that has a running time in $\log^{O(1)} n$ is called *processor efficient* if its work is within a polylogarithmic factor from the record sequential time for the same problem [13]. We suppose that the product of two $n \times n$ matrices over F can be computed in $O(\log n)$ parallel time using $\mathrm{M}(n)$ processors. Each time unit in the algorithms represents an arithmetic operation in F.

If F is of characteristic zero or greater than n, the processor count measures of the best known *deterministic algorithms* to solve $Ax = b$, exceed by a factor slightly less than $\sqrt{n}$ the processor complexity of matrix multiplication. These algorithms are improvements of the method of Csanky [5, 17] and are based on the reduction of the problem to rectangu-

0196-6774/00 $35.00

lar matrix multiplication [8]. The solution x can be computed in $O(\log^2 n)$ time using $O(n^{2.837})$ processors [9]. Over a field of any characteristic, the exceeding factor is n [1, 4]. These results hold if A is invertible. When A is singular, to test whether the system is consistent and to possibly compute a solution over any field leads to the even greater exceeding factor n^4 [2, p. 333].

Over an abstract field, Kaltofen and Pan have discovered the only known approach to handle the problem of processor efficiency for linear system solution. This approach leads to the following class of *Las Vegas randomized* algorithms. One can solve $Ax = b$, for any $n \times n$ matrix A, in randomized time $O(\log^2 n)$ using $O(\mathrm{M}(n))$ processors if the characteristic of F is zero or greater than n [10, 11]. Over any field if A is invertible, the time increases to $O(\log^3 n)$ using the algorithm in [11] combined with those in [12, 16]; the processor complexity remains in $O(\mathrm{M}(n))$ using the improvement of Eberly [6]. For A singular, the time used increases to $O(\log^4 n)$ with the same processor complexity [11, 12, 16].

We will now assume that F is an ordered field (-1 is not a finite sum of squares) or a subfield of the complex numbers, a direct application of known results allows us to present a *deterministic* version of above processor efficient algorithms. For a matrix A or a vector over F, A^* will denote the Hermitian transpose.

2. THE ALGORITHM

The previously cited algorithms are nontrivial parallelizations of the Wiedemann sequential method [18]. They are thus strongly related to Lanczos biorthogonalization [14]. The solution of $Ax = b$ is computed in the Krylov subspace span$\{b, Ab, A^2b, \ldots\}$, using an auxiliary vector u and the associated Krylov subspace span$\{u^*, u^*A, u^*A^2, \ldots\}$. It is a classical fact that—as called by Wilkinson [19]—the algorithm may "seriously break down." The randomizations proposed in [18] and used, as seen above, for parallelization [6, 11, 12], precisely avoid this problem especially over finite fields.

Our observation is simply that since—as well known—the *Hermitian Lanczos* method [14] i.e. when $A = A^*$ and with $u = b$, does not seriously break down, then the corresponding parallelization proposed in [11] must be deterministic. The algorithm follows immediately.

2.1

We begin with A invertible. Forming if necessary the system $AA^*y = b$, we will work with a Hermitian matrix. Following the parallelization of

Kaltofen and Pan [11] of the Wiedemann method [18], the parallel algorithm is:

Algorithm 1.
Input: A invertible in $\mathscr{M}_{n,n}(\mathsf{F})$; $b \neq 0$ a n-dimensional vector.
Compute $C = AA^*$.
Form the $(n+1) \times (n+1)$ Hankel matrix $H = \left[h_{i,j}\right]$ where $h_{i,j} = b^*C^{i+j}b, 0 \leq i,j \leq n$.
$d :=$ rank H.
Form the $d \times d$ sub-matrix $H_d = \left[h_{i,j}\right]_{0 \leq i,j \leq d-1}$ and the vector $z = \left[-h_{i,d}\right]_{0 \leq i \leq d-1}$.
Solve $H_d[g_0, g_1, \ldots, g_{d-1}]^t = z$.
{Here, $\lambda^d + g_{d-1}\lambda^{d-1} + \cdots + g_1\lambda + g_0$ is the minimal polynomial of b with respect to C.}
$y := (-1/g_0)\left(C^{d-1}b + g_{d-1}C^{d-2}b + \cdots + g_1 b\right)$.
$x = A^*y$.
Output: x the n-dimensional vector such that $Ax = b$.

To show that the algorithm is correct it is sufficient to prove that rank H_d = rank $H(= d)$ and that the corresponding polynomial $g(\lambda) = \lambda^d + g_{d-1}\lambda^{d-1} + \cdots + g_1\lambda + g_0$ is the minimal polynomial of b with respect to C. These facts are easily shown. Let δ be the degree of this latter minimal polynomial, δ is the dimension of span$\{b, Cb, C^2b, \ldots\}$ thus rank $H \leq \delta$. In addition, $H_\delta = B^*B$, where $B = [b, Cb, \ldots, C^{\delta-1}b]$, thus H_δ is invertible. Indeed, $H_\delta x = 0$ implies that $x^*B^*Bx = 0$ thus $Bx = 0$ and $x = 0$. It follows that rank $H \geq$ rank $H_\delta = \delta$ and $d = \delta$. By uniqueness of the solution of $H_d[g_0, g_1, \ldots, g_{d-1}]^t = z$, $g(\lambda)$ is the minimal polynomial of b.

The matrix C is computed in $O(\log n)$ time using $\mathrm{M}(n)$ processors. From [3, p. 128], the $2n+1$ vectors $C^k b$ and consequently the matrix H are computed using $O(\log^2 n)$ time with $O(\mathrm{M}(n))$ processors. We know that computing the rank of the Hankel matrix H [16, §1.4] and solving the corresponding system [15] can be done in time $O(\log^2 n)$ with $O(n^2(\log\log n)/\log n)$ processors. From there, x is computed within the same bounds. Hence the whole algorithm takes $O(\log^2 n)$ time using $O(\max\{\mathrm{M}(n), n^2(\log\log n)/\log n\})$ processors.

2.2

If A is square and singular, range A = range AA^* thus the system $Ax = b$ is consistent if and only if $Cy = b$ is consistent. Since C is Hermitian, these systems are consistent if and only if the minimal polynomial of b with respect to C has nonzero constant term. Indeed, C is

similar to a diagonal matrix D over an extension field of F: $C = P^{-1}DP$. The system $Cy = b$ is consistent if and only if $DPy = Pb$ is consistent. This is equivalent to the fact that Pb has nonzero entries at places corresponding to the nonzero entries of D, thus to the fact that the minimal polynomial of Pb with respect to D has nonzero constant term. The claim follows since the latter minimal polynomial is equal to the one of b with respect to C.

A solution is thus computed as follows.

`Algorithm 2.`
`Input:` $A \in \mathcal{M}_{n,n}(\mathsf{F})$; $b \neq 0$ a n-dimensional vector.
Compute $g(\lambda)$ using `Algorithm 1`.
If $g_0 = 0$ then `output` "system inconsistent".
else $y := (-1/g_0)\left(C^{d-1}b + g_{d-1}C^{d-2}b + \cdots + g_1 b\right)$.
$x = A^* y$.
`Output:` x an n-dimensional vector such that $Ax = b$.

The algorithm is correct since the arguments used in the regular case still hold when A is singular, `algorithm 1` actually computes the minimal polynomial of b. The complexity measures remain unchanged.

CONCLUSION

As a consequence of some known results, we have given a deterministic processor efficient algorithm for a particular class of fields. The question of deterministic processor efficiency for any field is still open. For a randomized version of the *symmetric Lanczos method* over finite fields, the reader may refer to [7]. In the same way, the solution of the system is computed using the minimal polynomial of a particular vector, thus even with restrictions on the field, we do not solve the problem of matrix inversion.

REFERENCES

1. S. J. Berkowitz, On computing the determinant in small parallel time using a small number of processors, *Inform. Process. Lett.* **18** (1984), 147–150.
2. D. Bini and V. Pan, "Polynomial and Matrix Computations," Birkhäuser, Basel, 1994.
3. A. Borodin and I. Munro, "The Computational Complexity of Algebraic and Numeric Problems," Elsevier, New York, 1975.
4. A. L. Chistov, Fast parallel computation of the rank of matrices over a field of arbitrary characteristic, *in* "Proc. FCT'95," LNCS 199, pp. 63–69, Springer Verlag, Berlin, 1985.

5. L. Csanky, Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5** (4) (1976), 618–623.
6. W. Eberly, Processor-efficient parallel matrix inversion over abstract fields: Two extensions, *in* "Second International Symposium on Parallel Symbolic Computation (PASCO'97)," Maui, HI, July 1997, pp. 38–45.
7. W. Eberly and E. Kaltofen, On randomized Lanczos algorithms, *in* "International Symposium on Symbolic and Algebraic Computation," Maui, HI, July 1997, ACM Press, pp. 176–183.
8. Z. Galil and V. Y. Pan, Parallel evaluation of the determinant and of the inverse of a matrix, *Inform. Process. Lett.* **30** (1989), 41–45.
9. X. Huang and V. Pan, Fast rectangular matrix multiplications and improving parallel matrix computations, *in* "Second International Symposium on Parallel Symbolic Computation (PASCO'97)," Maui, HI, July 1997, pp. 11–23.
10. E. Kaltofen and V. Pan, Processor efficient parallel solution of linear systems over an abstract field, *in* "Proc. 3rd Annual ACM Symposium on Parallel Algorithms and Architecture," ACM Press, 1991.
11. E. Kaltofen and V. Pan, Processor efficient parallel solution of linear systems II: The general case, *in* "Proc. 33rd IEEE Symp. Foundations of Computer Science," Pittsburgh, 1992.
12. E. Kaltofen and V. Pan, Parallel solution of Toeplitz and Toeplitz-like linear systems over fields of small positive characteristic, *in* "First International Symposium on Parallel Symbolic Computation (PASCO'94)," Lecture Notes Series in Computing, Vol. 5, pp. 225–233, World Scientific, Singapore, 1994.
13. R. M. Karp and V. Ramachandran, Parallel algorithms for shared-memory machines, *in* "Handbook of Theoretical Computer Science Vol. A" (J. van Leuwen, Ed.), pp. 869–941, North-Holland, Amsterdam, 1990.
14. C. Lanczos, Solutions of systems of linear equations by minimized iterations, *J. Res. Bur. Standards Sect. B* **49** (1952), 33–53.
15. V. Pan, Parametrization of Newton's iteration for computations with structured matrices and applications, *Comput. Math.* **24** (3) (1992), 61–75.
16. V. Pan, Parallel computation of polynomial GCD and some related parallel computations over abstract fields, *Theoret. Comput. Sci.* **162** (1996), 173–223.
17. F. P. Preparata and D. V. Sarwate, An improved parallel processor bound in fast matrix inversion, *Inform. Process. Lett.* **7** (1978), 148–150.
18. D. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory* **32** (1986), 54–62.
19. J. H. Wilkinson, "The Algebraic Eigenvalue Problem," Clarendon Press, Oxford, 1965.