Pergamon

Mathl. Comput. Modelling Vol. 21, No. 11, pp. 97–99, 1995
Copyright©1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0895-7177/95 $9.50 + 0.00

0895-7177(95)00084-4

Parallel Computation of a Krylov Matrix for a Sparse and Structured Input*

V. Y. PAN
Department of Mathematics and Computer Science
Lehman College, City University of New York
Bronx, NY 10468, U.S.A.
vpan@lcvax.bitnet

(Received and accepted September 1994)

Abstract—As in our previous work, we reduce parallel computation of a Krylov matrix to solving a parametrized linear system of equations. This time we show that such a method is effective in the cases of banded matrices, sparse and structured matrices and triangular matrices.

Keywords—Krylov sequences, Krylov matrices, Linear systems, Band matrices, Sparse matrices, Triangular matrices.

For a pair of natural m and n, an $n \times n$ matrix A, and an n-dimensional vector v, one may define the $n \times m$ Krylov matrix

$$K(A, v, m) = \left(v, Av, A^2v, A^3v, \dots, A^{m-1}v\right), \tag{1}$$

whose computation (for various m ranging from a small constant to order of n) is a major task of numerical linear algebra [1]. Sequential computation of $K(A, v, m)$ is straightforward: it amounts to $m - 1$ successive multiplications of the matrix A by vectors $A^i v$, $i = 0, 1, \dots, m-2$.

To evaluate the matrix $K(A, v, m)$ on a parallel computer (assuming $m = 2^h - 1$ for simplicity), one may recursively compute the matrices A^{2^i}, $i = 1, \dots, h-1$, and then

$$\begin{aligned} A^2(v, Av) &= \left(A^2v, A^3v\right), \\ A^4\left(v, Av, A^2v, A^3v\right) &= \left(A^4v, A^5v, A^6v, A^7v\right), \\ &\vdots \end{aligned} \tag{2}$$

This well-known algorithm [2] only needs $2h - 2$ matrix multiplications, which, however, do not preserve the structure of A. For structured (dense and sparse) matrices A, we will recall the distinct approach of [3], originally proposed for Toeplitz and Toeplitz-like matrices A, and will extend it to the cases of banded (block tridiagonal) and some other special matrices A.

According to the recipe of [3], one should introduce an auxiliary scalar parameter λ, define the matrix $B(\lambda) = I - \lambda A$, and then compute the column vectors $A^i v$ of $K(A, v, m)$ as the coefficients of Newman's expansion,

$$B(\lambda)^{-1}v = (I - \lambda A)^{-1}v = \sum_{i=0}^{\infty}(\lambda A)^i v,$$

reduced modulo λ^m.

*Supported by NSF Grant CCR 9020690 and PSC CUNY Awards Nos. 662478 and 664334.

Due to the latter equation, the computation of the Krylov matrix A of (1) reduces to the solution of the linear system

$$B(\lambda)x = v \bmod \lambda^m. \tag{3}$$

Our task, respectively, reduces to recalling some parallel algorithms for the system (3), for some selected classes of matrices A, and to estimating parallel arithmetic cost of performing these algorithms (in terms of parallel time and the number of processors used), and we will refer the reader to the relevant bibliography for the proofs and details.

To express this parallel cost, we hereafter let $O(t,p)$ denote the pair of simultaneous bounds $O(t)$ on the arithmetic parallel time and $O(p)$ on the number of processors supporting this time bound. Under this notation, $O(\log k, k)$ bounds the parallel cost of addition, subtraction and multiplication of a pair of polynomials in λ performed modulo λ^k [2]; furthermore, essentially the same cost bounds apply to computation modulo λ^k of the reciprocal of a polynomial in λ with a nonzero λ-free term.

Hereafter, we let $O(t_{I(k)}, p_{I(k)})$ denote the parallel complexity of $k \times k$ matrix inversion, where we may set $t_{I(k)} = \log^2 k$, $p_{I(k)} = k^d/\log k$, [2], assumed randomization and provided that $k \times k$ matrix multiplication costs $O(\log k, k^d/\log k)$. (Theoretically, one may set $d < 2.38$, but practically, d stays at the levels 3 or 2.81, [1,2].)

We note that algorithm (2) computes the Krylov matrix $K(A, v, m)$ at cost $O((\log m)\log n, n^d)$ for a general (dense and unstructured) matrix A. Next we will show improved estimates in the case of special matrices A.

In particular, the latter bound decreases to $O((\log m)\log n, mn/\log m)$ if A is a *Toeplitz or Hankel matrix* or, more generally, a *Toeplitz-like + Hankel-like matrix*, [2–4].

Next consider the cost of $s \times s$ *block tridiagonal matrices* A with $k \times k$ blocks, which is essentially the case of *banded matrices* A. Then we may solve the system (3) by applying the block cyclic reduction algorithm [1]. The cost of application of this algorithm to the scalar input matrix is bounded by

$$O\left(t_{I(k)} \log s, \frac{p_{I(k)} s}{\log s}\right)$$

(see [5]). Since we deal with polynomials reduced modulo λ^m, the cost of every arithmetic operation increases from $O(1,1)$ to $O(\log m, m)$, and we arrive at the overall cost $O((\log m) t_{I(k)} \log s$, $m p_{I(k)} s/\log s)$.

Next consider the case of *sparse matrices* A given with *their* $s(n)$-*separator families*, $s(n) = o(n)$, (see definitions of [6–9]). In this case, application of the parallel generalized nested dissection algorithm of [9] gives the cost bound $O(\log^3 n, (s(n)^d)/\log^2 n)$ for solving a linear system with scalar sparse input matrix. This is immediately extended to the cost bound $O((\log m)\log^3 n$, $(s(n))^d m/\log^2 n)$ for the solution of the system (3).

We conclude with the case of triangular matrices A. In this case, we may apply the algorithm of [10,11], which supports the cost bound $O(\sqrt{n}\,\log n, n^{3/2}/\log n)$ for solving a triangular linear system with a scalar coefficient matrix. For the system (3), this cost bound turns into $O((\log m)\sqrt{n}\,\log n, n^{3/2}m/\log n)$. Compared to the cost bound $O(n\log n, n^2/\log n)$ of the straightforward evaluation of $K(A,v,m)$ via $m-1$ successive multiplications of A by vectors, we note acceleration if $\sqrt{n}\,\log n = o(m)$.

REFERENCES

1. G.H. Colub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, (1989).
2. D. Bini and V.Y. Pan, *Polynomial and Matrix Computations: Vol. 1, Fundamental Algorithms*, Birkhäuser, Boston, (1994).
3. V.Y. Pan, Parametrization of Newton's iteration for computations with structured matrices and applications, *Computers Math. Applic.* **24** (3), 61–75 (1992).
4. D. Bini and V.Y. Pan, Improved parallel computation with Toeplitz-like and Hankel-like matrices, *Linear Algebra and Its Applications* **188** (189), 3–29 (1993).

5. V.Y. Pan, I. Sobze and A. Atinkpahoun, On parallel computation with band matrices, *Information and Computation* (to appear).
6. R.J. Lipton, D. Rose and R.E. Tarjan, Generalized nested dissection, *SIAM J. on Numer. Analysis* **16** (2), 346–358 (1979).
7. J.R. Gilbert and R.E. Tarjan, The analysis of nested dissection algorithm, *Numer. Math.* **50**, 377–404 (1987).
8. V.Y. Pan, Parallel solution of sparse and path systems, In *Synthesis of Parallel Algorithms*, (Edited by J.H. Reif), Chapter 14, pp. 621–678, Morgan Kaufmann, San Diego, CA, (1993).
9. V.Y. Pan and J.H. Reif, Fast and efficient solution of sparse linear systems, *SIAM J. on Computing* **22** (6), 1227–1250 (1993).
10. V.Y. Pan, Improved parallel solution of a triangular linear system, *Computers Math. Applic.* **27** (11), 41–43 (1994).
11. V.Y. Pan and F.P. Preparata, Work-preserving speed-up of parallel matrix computations, *SIAM J. on Computing* **24** (4) (1995).