IMACS '91

13TH WORLD CONGRESS
ON
COMPUTATION AND APPLIED MATHEMATICS

JULY 22 - 26, 1991
TRINITY COLLEGE DUBLIN
IRELAND

PROCEEDINGS

IN FOUR VOLUMES

VOLUME 2

LU DECOMPOSITION ON A SHARED MEMORY MULTIPROCESSOR

Paul A. Farrell
Dept. of Mathematics & Computer Science
Kent State University
Kent, OH 44242, U.S.A.

Arden Ruttan
Dept. of Mathematics & Computer Science
Kent State University
Kent, OH 44242, U.S.A.

Abstract:- We propose an algorithm for the parallel LU decomposition of an upper Hessenberg matrix on a shared memory multiprocessor. We consider the general case of p processors, where p is not related to the size of the matrix problem. We show that the LU decomposition of an $(m+1)$-banded Hessenberg matrix can be achieved in $O(\frac{3nm^2}{p})$ operations, where n is the dimension of the matrix and p is the number of processors. For tridiagonal matrices this algorithm has a lower operation count than those in the literature and yields the best existing algorithm for the solution of tridiagonal systems of equations.

1. Introduction

A number of authors over the last two decades have written on parallel algorithms for solving tridiagonal systems. These articles have considered the problem of solving tridiagonal systems for the form $Ax = b_i$, $1 \le i \le k$ where A and all of the b_i, are known at the start of the process. In such cases, the computations can be arranged to produce highly efficient parallel solutions to all m systems simultaneously. It should be noted, however, that there are a number of common numerical situations, for example the ADI method, where one needs to solve tridiagonal systems where A is known *ab initio* but the b_i's are not all known at the start of the computation but rather arise as a result of an iteration process.

2. LU Decomposition Algorithm

We shall, in fact, consider the LU decomposition of an $n \times n$ upper Hessenberg matrix, since the analysis is not significantly more difficult and the additional generality leads to insights, which produce a more efficient algorithm. Let $A = (a_{ij})$ be a banded $n \times n$ upper Hessenberg matrix with band width $m+1$, i.e., $a_{ij} \neq 0$ only when $\min\{1, i-1\} \le j \le \max\{n, m+i-1\}, 1 \le i \le n$. It suffices to consider the case where $a_{i+1,i} \neq 0, 1 \le i \le n-1$, since otherwise the matrix is reducible, and we may consider the LU decomposition of the subproblems resulting from the reduction. Throughout this paper we will use the convention that any element with a nonpositive index has value zero.

As in most algorithms for shared memory multiprocessors, the object here is to partition the problem into a number of subproblems suitable for solution by tasks running on the available processors. We shall consider the general case of p processors, where p is not related to the size of the matrix problem. Clearly A has an LU factorization, $A = LU$, where L is a unit lower bi-diagonal $n \times n$ matrix and $U = (u_{ij})$ is a banded $n \times n$ upper triangular, with m non-zero diagonals, including the main diagonal. The special form of L allows one to readily determine L^{-1}. One finds that $L^{-1} = (\hat{\ell}_{ij})$ is an $n \times n$ lower triangular matrix given by

$$\hat{\ell}_{ij} := \begin{cases} \prod_{t=j+1}^{i} (-\ell_t) & i \ge j \\ 0 & i < j. \end{cases} \tag{1}$$

Thus the elements of $U = L^{-1}A$, satisfy $1 \le i, j \le n$

$$u_{ij} = \sum_{s=j-m+1}^{\min\{i,j+1\}} \hat{\ell}_{is} a_{sj} = \sum_{s=j-m+1}^{\min\{i,j+1\}} \prod_{t=s+1}^{i} (-\ell_t) a_{sj}. \tag{2}$$

As in the tridiagonal case, the well known substitution (*cf.* [2], pp. 473 – 474)

$$\begin{aligned} y_1 &= 1 \\ \ell_i &= y_{i-1}/y_i \qquad i = 2, 3, \cdots, n \end{aligned} \tag{3}$$

can be used to simplify (2). Since $u_{j+1,j} = 0$, for $1 \le j \le n-1$, (2) yields the following linear systems for the unknowns y_i:

$$\begin{aligned} y_j &= 0, \;\; j \le 0 \quad , \quad y_1 = 1 \\ y_{j+1} a_{j+1,j} &= \sum_{s=j-m+1}^{j} (-1)^{j-s} y_s a_{sj}, \quad 1 \le j \le n-1. \end{aligned} \tag{4}$$

It is clear that (4) defines an $m+1$ banded triangular linear system

$$Ty = e. \tag{5}$$

where

$$t_{ij} = \begin{cases} 1 & \text{if } \; i = j = 1 \\ (-1)^{i-j} a_{j,i-1} & j \le i \;,\; i > 1 \\ 0 & j > i \end{cases}$$

Thus the problem of finding an LU factorization of an upper Hessenberg matrix reduces to solving the banded triangular system described in (5) to obtain the y_i's and then using the solution of that system to evaluate L and U. In practice, L may be determined from equation (3). To determine U, note first that the elements of the $(m-1)^{st}$ diagonal, $u_{i,i+m-1}$ satisfy $u_{i,i+m-1} = a_{i,i+m-1}$, for $i = 1, \ldots, n-m+1$. Also $u_{1,j} = a_{1,j}$ for $j = 1, \ldots, m$. Thus these elements do not require any calculation. The j^{th} super-diagonal is given in terms of the $(j+1)^{st}$ by

$$\ell_i u_{i-1,j+i} + u_{i,j+i} = a_{i,j+i}, i = 2, \ldots, n-j. \tag{6}$$

Note that each element depends only on a single element of the next superdiagonal and on known values from L and A. Thus, the calculation of each super-diagonal of U is perfectly parallelizable. In fact, the main diagonal may be obtained using 1 division rather than the multiplication and subtraction in (6) by

$$u_{i,i} = a_{i+1,i}/\ell_{i+1}, i = 2, \ldots, n. \tag{7}$$

Further, the calculation of the super-diagonals can be chained.

Thus the calculation of L using (3) requires $n-1$ divisions. The total computations for U is

$$(n-1) + 2\sum_{j=1}^{m-2} (n-j-1) = n(2m-3) - (m^2 - m - 1). \tag{8}$$

The latter term is negative for $m \ge 1$. Hence we get the following upper bound for the complexity of calculating U

$$n(2m-3).$$

Note that in the case of a tridiagonal system, $m = 2$, (8) reduces exactly to $n-1$. Hence using p processors L and U can be calculated in the general case in

$$\frac{2n(m-1)}{p} \tag{9}$$

operations and in the special tridiagonal case in

$$2(n-1)/p \tag{10}$$

operations. There remains only the solution of the triangular system $Ty = e$.

3. Algorithm for the Triangular System

In [3], Lakshmivarahan and Dhall present an algorithm for calculating the LU factorization of a tridiagonal matrix. Their algorithm used the substitution given in (3) to produce a linear system, which is equivalent to that described by (5) with $m = 2$. Our algorithm is a generalization to the upper Hessenberg case of the algorithm presented in [3]. We remark that the improvement produced in the generalization leads also to a more efficient algorithm for the tridiagonal case.

In order to partition the problem we set

$$z_j = (y_{j-m+1}, y_{j-m+2}, \cdots, y_j)^T, \quad 1 \le j \le n,$$

and let $B_j, 1 \le j \le n-1$, be the $m \times m$ matrix

$$B_j := \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \\ b_1^j & \cdots & & b_{m-1}^j & b_m^j \end{bmatrix} \tag{11}$$

where $b_i^j := (-1)^{m-i} a_{j-m+i,j}/a_{j+1,j} \quad j = 1, 2, \cdots, n-1$. We obtain from (4) the m-vector iteration

$$z_{j+1} = B_j z_j \quad j = 1, 2, \cdots, n-1. \tag{12}$$

In order to evaluate $y_1, y_2, \cdots, y_n$, one must compute

$$z_{km+1} = \left(\prod_{i=1}^{k} C_i\right) z_1, \quad k = 1, 2, 3, \cdots, N := \lceil (n-1)/m \rceil. \tag{13}$$

where $C_i := \prod_{j=(i-1)m+1}^{im} B_j,\ 1 \le i \le N-1$ and $C_N := \prod_{j=(N-2)m+1}^{n-1} B_j$.

To calculate the required products $\prod_{i=1}^{k} C_i,\ k = 1, 2, \cdots, N$, one uses a variant of recursive doubling. Let $Z_1 = C_1 z_1$ and $Z_i = C_i,\ i = 2, \ldots, N$, then, assuming we have g processor groups, each group first calculates

$$D_{l,k} = \prod_{i=(k-1)M+2}^{(k-1)M+l} Z_i,\ k = 1, \ldots, g,\ l = 2, \ldots, M = N/g.$$

Then the g processor groups execute the following algorithm:

```
for i := 0 thru log(g) - 1 do
{distribute the g/2 independent calculations found in the}
{ j and k loops below among the g/2 groups of processors }
for j := 2^i thru g - 2^i step 2^(i+1) do
      for k := j + 1 thru j + 2^i do
            {using 2 groups calculate }
            for l = 1 thru M do
                  D_{l,k} := D_{l,k} D_{M,j}
```

It is easily seen that after the execution of the above algorithm $D_{l,k} = (\prod_{j=1}^{(k-1)M+l} C_j) z_1$. For the purpose of simplifying the complexity analysis, assume that $n = Nm + 1$, $p = g(2m-1)$ where g is a power of 2, and $N = gM$.

An analysis, the details of which appear in [1], yields the following time complexity estimate. For a general upper Hessenberg matrix with band width $m+1$, the time required to calculate its y_i's in this fashion is

$$\frac{n}{2p}\left[6m^2 - 2m + 1 + m(2m-1)\log\left(\frac{p}{2m-1}\right)\right] + O(m). \tag{14}$$

In the tridiagonal case $m = 2$, and this reduces to

$$\frac{n}{p}\left[\frac{21}{2} + 3\log(\frac{p}{3})\right]. \tag{15}$$

Thus, from (9) and (14), the total cost of an LU factorization is

$$\frac{n}{2p}\left[6m^2 + 2m - 3 + (2m^2 - m)\log\left(\frac{p}{2m-1}\right)\right] + O(m). \tag{16}$$

In the tridiagonal case, by (10) and (15), the cost of producing an LU factorization is

$$\frac{n}{p}\left[\frac{25}{2} + 3\log\left(\frac{p}{3}\right)\right] + O(1). \tag{17}$$

If one's goal is to solve the linear system $Ax = b$, in addition to solving (5), one must also perform the backsolve by solving the banded triangular systems $Lz = b$ and $Ux = z$. In the tridiagonal case, this may be done by casting it as the solution of two linear recurrences, similar to (2.3) and (2.4) in [3]. The recurrences may then be cast in the form $z_i = \alpha_i z_{i-1} + \beta_i$ and solved using *Algorithm A* and *Algorithm Y* from [3]. The complexity involved, in the tridiagonal case, is $2n/p$, to cast the analogue of (2.3) in [3] in the appropriate form, and $3n(2 + \log(2p/3))/p$ to solve the two recurrences, using *Algorithm A* and *Algorithm Y*. Adding these gives a total of

$$\frac{n}{p}(8 + 3\log(\frac{2p}{3})). \tag{18}$$

The cost of solving for one righthand side, given by (17) and (18), is thus

$$\frac{n}{p}\left[\frac{47}{2} + 6\log\left(\frac{p}{3}\right)\right] + O(1). \tag{19}$$

4. Conclusions

In the tridiagonal case, the algorithm is not only better than existing algorithms in the literature for LU decomposition, but also has better computational complexity for the solution of a single tridiagonal system, as indicated in Table 1, where $n\prime = n + 1 = 2^t$.

Method	Processors	Time
Serial Gaussian Elimination	1	8n
Recursive Doubling [2]	n	$24\log n$
Odd-Even Reduction [2]	$n\prime/2$	$19\log n\prime - 14$
Odd-Even Elimination [2]	$n\prime$	$14\log n\prime + 1$
Lakshmivarahan Dhall [3]	$n/2$	$18\log n$
Lakshmivarahan Dhall [3]	p $3 \le p \le \frac{3n}{4}$	$(n/p)[25 + 9\log p/3] - 3$
Algorithm	p	$(n/p)[47/2 + 6\log p/3]$

Table 1: Complexity of the solution of a single linear system for tridiagonal matrices.

Further let us consider again cases, such as the ADI method discussed in the introduction, where A is known in advance but the b_i are not. Comparing this algorithm with methods, such as Recursive-Doubling, which do not perform the LU decomposition, a further improvement in computational efficiency results, since one need only perform the forward and back solves, for each right-hand side, rather than performing the full elimination.

References

[1] J.Buoni, P.A. Farrell, A. Ruttan, *Algorithms for LU Decomposition on a Shared Memory Multiprocessor*, Technical Report CS-90-10-28, Department of Mathematics and Computer Science, Kent State University, 1990.

[2] R.W. Hockney, C.R. Jesshope, *Parallel Computers 2 - Architecture, Programming and Algorithms* (Hilger, Bristol, 1988).

[3] S. Lakshmivarahan, S.K. Dhall, A New Class of Parallel Algorithms for Solving Tridiagonal Systems, *IEEE Fall Joint Computer Conference* (1986) 315-324.

[4] A.H. Sameh, R.P. Brent, Solving Triangular Systems on a Parallel Computer, *SIAM JNA* 14(6) (1977) 1101-1113.