PAPRZYCKI, M.; STPICZYŃSKI, P.

Parallel Solution of Linear Recurrence Systems

The paper describes some recently proposed divide-and-conquer parallel algorithms for solving linear recurrence systems. Such systems arise in many computational problems. A special case – solving linear systems with constant coefficients – is also discussed. As an example of an application of such linear systems, parallel algorithms for finding trigonometric sums are presented.

1. Introduction

Let us consider a linear recurrence system of order m for n equations:

$$x_k = \begin{cases} 0 & \text{if } k \le 0, \\ f_k + \sum_{j=k-m}^{k-1} a_{kj} x_j & \text{if } 1 \le k \le n. \end{cases} \tag{1}$$

Many parallel algorithms for solving this problem have been proposed [1–9]. In [14] two efficient medium-grain divide-and-conquer parallel algorithms were introduced and later implemented on a Sequent [11]. These algorithms have very good numerical properties [15]. Recently, a modified algorithm has been proposed to handle the special case of the system (1) – a system with constant coefficients [12]:

$$x_k = \begin{cases} 0 & \text{for } k \le 0, \\ f_k + \sum_{j=1}^{m} a_j x_{k-j} & \text{for } 1 \le k \le n, \end{cases} \tag{2}$$

2. Parallel Algorithms

First, let us consider the general linear recurrence system (1). Computation of values x_k can be represented as a solution of a system of linear equations:

$$(I - A)\mathbf{x} = \mathbf{f} \qquad \text{where } I, A \in \mathbb{R}^{n\times n} \text{ and } \mathbf{x}, \mathbf{f} \in \mathbb{R}^n, \tag{3}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{f} = (f_1, \ldots, f_n)^T$ and $A = (a_{ik})$ with $a_{ik} = 0$ for $i \le k$ or $i - k > m$.

Without loss of generality we can assume that n is divisible by p (where 'p' is the number of processors) and $q = n/p > m$. The system (3) can be written in the following block form

$$\begin{pmatrix} L_1 & & & & \\ U_2 & L_2 & & & \\ & U_3 & L_3 & & \\ & & \ddots & \ddots & \\ & & & U_p & L_p \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{p-1} \\ \mathbf{x}_p \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{p-1} \\ \mathbf{f}_p \end{pmatrix}, \tag{4}$$

where $L_j \in \mathbb{R}^{q\times q}$, $\mathbf{x}_j \in \mathbb{R}^q$, $\mathbf{f}_j \in \mathbb{R}^q$ for $j = 1, \ldots, p$ and $U_j \in \mathbb{R}^{q\times q}$ for $j = 2, \ldots, p$. Vectors $\mathbf{x}_j$ in (4) satisfy the following recurrence relation

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = L_j^{-1}\mathbf{f}_j - L_j^{-1}U_j\mathbf{x}_{j-1} & \text{for } j = 2, \ldots, p. \end{cases} \tag{5}$$

Let U_j^k denote the kth column of the matrix U_j. Since $U_j^k = 0$ for $j = 2, \ldots, p$ and $k = 1, \ldots, q - m$ we obtain the following formula for Algorithm 1:

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = \mathbf{z}_j - \sum_{k=0}^{m-1} x_{(j-1)q-k}\mathbf{y}_j^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{6}$$

where $L_j\mathbf{z}_j = \mathbf{f}_j$ and $L_j\mathbf{y}_j^k = U_j^{q-k}$ for $k = 0, \ldots, m-1$.

Let us now observe that each matrix U_j can be rewritten as:

$$U_j = -\sum_{k=1}^{m}\sum_{l=k}^{m} \beta_j^{kl}\mathbf{e}_k\mathbf{e}_{q-m+l}^T, \tag{7}$$

where $\beta_j^{kl} = a_{(j-1)q+k,(j-1)q-m+l}$ and $\mathbf{e}_k$ denotes the kth unit vector of R^q. Substituting into (6) we obtain the following formula for Algorithm 2 (for a more detailed description of both algorithms see [14]):

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = \mathbf{z}_j + \sum_{k=1}^{m} \alpha_j^k\mathbf{y}_j^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{8}$$

where $\alpha_j^k = \sum_{l=k}^{m} \beta_j^{kl}\mathbf{e}_{q-m+l}^T\mathbf{x}_{j-1}$, $L_j\mathbf{y}_j^k = \mathbf{e}_k$ for $k = 1, \ldots, m$, and $L_j\mathbf{z}_j = \mathbf{f}_j$.

In the constant coefficient system (2) observe that blocks L_j and U_j simplify to L and U respectively and the recurrence relation (5) can be thus rewritten as:

$$\begin{cases} \mathbf{x}_1 = L^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = L^{-1}\mathbf{f}_j - L^{-1}U\mathbf{x}_{j-1} \quad \text{for } j = 2, \ldots, p. \end{cases} \tag{9}$$

Note also that to compute vectors $\mathbf{y}^k$, $k = 1, \ldots, m$, we need to find only the solution of the system

$L\mathbf{y}^1 = \mathbf{e}_1$, where

$$\mathbf{y}^1 = (1, y_2, \ldots, y_q)^T, \tag{10}$$

and

$$\mathbf{y}^k = (\underbrace{0, \ldots, 0}_{k-1}, 1, y_2, \ldots, y_{q-k+1})^T. \tag{11}$$

The method (8) can be now modified to define Algorithm 3:

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1, \\ \mathbf{x}_j = \mathbf{z}_j + \sum_{k=1}^{m} \alpha_j^k\mathbf{y}^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{12}$$

where $L\mathbf{z}_j = \mathbf{f}_j$ for $j = 1, \ldots, p$, and $\mathbf{y}^k$ is given by (11) and

$$\begin{aligned} \alpha_j^1 &= a_m x_{(j-1)q-m+1} + a_{m-1}x_{(j-1)q-m+2} + \ldots + a_1 x_{(j-1)q} \\ \alpha_j^2 &= a_m x_{(j-1)q-m+2} + \ldots + a_2 x_{(j-1)q} \\ &\vdots \\ \alpha_j^m &= a_m x_{(j-1)q}. \end{aligned} \tag{13}$$

Methods (6), (8) and (12) are examples of divide-and-conquer algorithms. First, several linear recurrence systems of order m for q equations are solved separately (in parallel). Second, a set of values is calculated in each block and communicated to the next block thus effectively decoupling the original system. This is the sequential part of the algorithms. Finally, the solution to the original problem is calculated independently in each block (in parallel). Let us also observe that when Algorithm 3 is applied to the constant coefficient problem only $p+1$ subsystems of order m for q equations must be solved in parallel, so the number of subsystems does not depend on the order of the recurrence system.

For all these algorithms the optimal number of processors is $P_{OPT} = O(\sqrt{n})$ and the solution to the problem can be obtained in $O(\sqrt{n})$ steps. Algorithm 3 is much less sensitive to changes in the value of m.

3. Application of Algorithms

A slight modification of the proposed parallel algorithms can be easily applied to the problem of computing trigonometric sums

$$C(x) = \sum_{k=0}^{n} b_k \cos kx, \qquad S(x) = \sum_{k=1}^{n} b_k \sin kx. \tag{14}$$

There are two well-known sequential algorithms for finding solutions of (14): Reinsch's algorithm [13], which works for any value of x, and Goertzel's algorithm [13], which can be applied for $|x|$ not too close to zero. These algorithms transform the original problem (14) to the solution of a linear recurrence system of order 2:

$$x_k = \begin{cases} 0 & \text{if } k \le 0 \\ f_k + a_{k,k-2}x_{k-2} + a_{k,k-1}x_{k-1} & \text{if } 1 \le k \le N. \end{cases} \tag{15}$$

For Goertzel's algorithm we need to compute the following linear recurrence system with constant coefficients (thus we apply Algorithm 3):

$$S_k = \begin{cases} 0 & \text{for } k = n+1, n+2, \\ b_k + 2S_{k+1}\cos x - S_{k+2} & \text{for } k = n, n-1, \ldots, 1, \end{cases} \tag{16}$$

and then we compute $C(x) = b_0 + S_1 \cos x - S_2$ and $S(x) = S_1 \sin x$. In Reinsch's algorithm, we set $S_{n+2} = D_{n+1} = 0$ and if $\cos x > 0$ then we solve

$$\begin{cases} S_{k+1} = D_{k+1} + S_{k+2} \\ D_k = b_k + eS_{k+1} + D_{k+1}, \end{cases} \tag{17}$$

for $k = n, n-1, \ldots, 0$, where $e = -4\sin^2 \frac{x}{2}$. If $\cos x \le 0$ then we solve

$$\begin{cases} S_{k+1} = D_{k+1} - S_{k+2} \\ D_k = b_k + eS_{k+1} - D_{k+1}, \end{cases} \tag{18}$$

where $e = 4\cos^2 \frac{x}{2}$. Finally, we compute $C(x) = D_0 - \frac{e}{2}S_1$ and $S(x) = S_1 \sin x$.

Let p be the number of available parallel processors. For the parallel version of Goertzel's algorithm, we have to solve p linear recurrence systems with constant coefficients for q equations (it is assumed that $q = n/p$ is an integer) in parallel, and compute the last two terms of the final solution using a *recursive doubling* scheme [9]. For the parallel version of Reinsch's algorithm we have to solve $p+2$ linear recurrence systems for q equations (where $2n = pq$) in parallel, and then find the last two terms of the solution of system (17) or (18) using a similar *recursive doubling* scheme.

Although the parallel version of Goertzel's algorithm reaches the solution faster than the parallel version of Reisch's algorithm, Goertzel's algorithm has a smaller speedup. This is a typical example of Amdahl's Effect as Reisch's algorithm solves $2n$ equations whereas Goertzel's algorithm solves only n equations. For both parallel algorithms the speedup increases as the size of the problem (n) increases [16]. The following table [16] presents the speedup computed for $n = 6390$.

# proc.	2	3	5	6	7	9	10	11	14	15	18
Goertzel	0.88	1.31	2.13	2.54	2.93	3.53	3.79	4.17	4.89	5.34	5.75
Reinsch	1.01	1.51	2.47	2.92	3.37	4.30	4.70	5.05	6.26	6.48	7.38

It was also observed that both algorithms scale well for more than 100 equations per processor. Finally, numerical tests [17] show that the rounding errors of parallel algorithms are of the same order as the rounding errors of the corresponding sequential algorithms.

4. Concluding remarks

Two parallel algorithms for solving linear recurrence systems were presented and their modification for the case of recurrence systems with constant coefficients was discussed. The latter algorithm can be easily applied to the problem of finding trigonometric sums. The experiments performed so far suggest that the presented algorithms behave well on shared memory computers with limited number of processors. In the next step we plan to experiment with these algorithms on message passing computers.

5. References

1 BORODIN, A., MUNRO, I.: The computational complexity of algebraic and numerical problems, American Elsevier, New York 1975.

2 CARLSON, D.A.: Solving linear recurrence systems on mesh-connected computers with multiple global buses, J. Parallel and Distrib. Comput. **8** (1990) 89–95.

3 CHEN, S.-C.: Speedup of iterative programs in multiprocessing systems, University of Illinois at Urbana 1975.

4 CHEN, S.-C., KUCK, D.J.: Time and parallel processor bounds for linear recurrence systems, IEEE Trans. on Computers, **24** (1975) 701–717.

5 DONGARRA, J., JOHNSSON, L.: Solving Banded Systems on Parallel Processor, Parallel Comput. **5** (1987) 219–246.

6 HELLER, D.: A survey of parallel algorithms in numerical linear algebra, SIAM Review **20** (1978) 740–777.

7 GREENBERG, A., LANDER, R., PATERSON, M., GALIL, Z.: Efficient parallel algorithms for linear recurrence computation, Inf. Proc. Letters **15** (1982) 31–35.

8 KUCK, D.J.: Structure of Computers and Computations, Wiley, New York 1978.

9 MODI, J.: Parallel Algorithms and Matrix Computations, Oxford University Press, Oxford 1988.

10 PAPRZYCKI, M., GLADWELL, I.: Solving Almost Block Diagonal Systems on Parallel Computers, Parallel Comput. **17** (1991) 133–153.

11 PAPRZYCKI, M., STPICZYŃSKI, P.: Solving linear recurrence systems on parallel computers, Proceedings of the Mardi Gras '94 Conference, Baton Rouge, Feb. 10–12, 1994, Nova Science Publishers, New York 1995, (in press).

12 PAPRZYCKI, M., STPICZYŃSKI, P.: Solving linear recurrence systems on a Cray Y-MP, in: J.Dongarra, J. Waśniewski eds., Lecture Notes in Computer Science **879**, Springer-Verlag, Berlin 1994, 416–424.

13 STOER, J., BULIRSCH, R.: Introduciton to Numerical Analysis, Springer Verlag, New York 1980.

14 STPICZYŃSKI, P.: Parallel algorithms for solving linear recurrence systems, in: L. Bougé et al. eds., Lecture Notes in Computer Science **634**, Springer–Verlag, Berlin 1992, 343–348.

15 STPICZYŃSKI, P.: Error analysis of two parallel algorithms for solving linear recurrence systems, Parallel Comput. **19** (1993) 917–923.

16 STPICZYŃSKI, P., PAPRZYCKI, M.: Parallel algorithms for finding trigonometric sums, in: D. H. Bailey et al. eds., Parallel Processing for Scientific Computing, SIAM. Philadelphia 1995, 291–292.

17 STPICZYŃSKI: Efficient parallel algorithms for solving linear algebra problems, PhD dissertation (in Polish), MCS University, Lublin 1994.

Addresses: DR. MARCIN PAPRZYCKI, Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762

DR. PRZEMYSLAW STPICZYŃSKI, Institute of Mathematics, Numerical Analysis Department, Marie Curie-Sklodowska University, Pl. Marii Curie-Skłodowskiej 1, 20-031 Lublin, POLAND