# Solving linear recurrence systems on a Cray Y-MP

Marcin Paprzycki
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762
e-mail: paprzycki_m@gusher.pb.utexas.edu

Przemysław Stpiczyński
Institute of Mathematics, Numerical Analysis Department
Marie Curie-Skłodowska University
Pl. Marii Curie-Skłodowskiej 1, 20-031 Lublin, POLAND
e-mail: przem@apollo.umcs.lublin.pl

**Abstract.** Three *divide–and–conquer* algorithms for solving linear recurrence systems are introduced. Two of them are algorithms for the general case whereas the third is designed to take full advantage of the constant coefficient case. All three algorithms were implemented on an 8-processor Cray Y-MP and the results of experiments are presented.

**Keywords.** Linear recurrence systems, divide–and–conquer algorithms, shared–memory multiprocessors, speedup.

**Acknowledgement.** Experiments on the Cray were performed in the Center for High Performance Computing in Austin.

## 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 [1–4, 6–9]. In [12] two efficient medium-grain divide-and-conquer parallel algorithms were introduced; they were later implemented on a Sequent [11]. The algorithms have very good numerical properties [13]. Recently, a modified algorithm has been proposed to handle the special case of a system with constant coefficients [14]. The major purpose of this paper is to study the behavior of these algorithms on a vector-parallel computer (Cray Y-MP).

Section 2 presents the algorithms. Section 3 contains their arithmetical complexity functions. In Section 4, the results of experiments are presented and discussed.

## 2 Algorithm Description

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

$$(I - A)\mathbf{x} = \mathbf{f} \qquad \text{where } I, A \in R^{n\times n} \text{ and } \mathbf{x}, \mathbf{f} \in R^n, \tag{2}$$

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 (2) can be written in the following block form

$$\begin{pmatrix} L_1 & & & & \\ U_2 & L_2 & & & \\ & U_3 & L_3 & & \\ & & \ddots & \ddots & \\ & & & U_p & L_p \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{p-1} \\ \mathbf{x}_p \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{p-1} \\ \mathbf{f}_p \end{pmatrix}, \tag{3}$$

where

$$L_j = \begin{pmatrix} 1 & & & & & \\ -a_{r+2,r+1} & 1 & & & & \\ \vdots & \vdots & \ddots & & & \\ -a_{r+m+1,r+1} & \cdots & -a_{r+m+1,r+m} & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & -a_{jq,jq-m} & \cdots & -a_{jq,jq-1} & 1 \end{pmatrix} \in R^{q\times q}, \tag{4}$$

$$\mathbf{x}_j = \left(x_{(j-1)q+1}, \ldots, x_{jq}\right)^T, \qquad \mathbf{f}_j = \left(f_{(j-1)q+1}, \ldots, f_{jq}\right)^T \in R^q \tag{5}$$

for $j = 1, \ldots, p$ and

$$U_j = \begin{pmatrix} 0 & \ldots & 0 & -a_{r+1,r-m+1} & \cdots & -a_{r+1,r} \\ \vdots & & & \ddots & \ddots & \vdots \\ & & & & 0 & -a_{r+m,r} \\ & & & \ddots & & 0 \\ & & & & & \vdots \\ 0 & & & & \ldots & 0 \end{pmatrix} \in R^{q\times q} \tag{6}$$

for $j = 2, \ldots, p$ where $r = (j-1)q$. Vectors $\mathbf{x}_j$ in (5) satisfy the following recurrence relation

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = L_j^{-1}\mathbf{f}_j - L_j^{-1}U_j\mathbf{x}_{j-1} \quad \text{for } j = 2, \ldots, p. \end{cases} \tag{7}$$

Let $U_j^k$ denote the $k$th 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:

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = \mathbf{z}_j - \sum_{k=0}^{m-1} x_{(j-1)q-k}\mathbf{y}_j^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{8}$$

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:

$$U_j = -\sum_{k=1}^{m}\sum_{l=k}^{m} \beta_j^{kl}\mathbf{e}_k\mathbf{e}_{q-m+l}^T, \tag{9}$$

where $\beta_j^{kl} = a_{(j-1)q+k,(j-1)q-m+l}$ and $\mathbf{e}_k$ denotes the $k$th unit vector of $R^q$. Substituting into (8) we obtain the following formula for Algorithm 2 (for a more detailed description of both algorithms see [12])

$$\begin{cases} \mathbf{x}_1 = L_1^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = \mathbf{z}_j + \sum_{k=1}^{m} \alpha_j^k\mathbf{y}_j^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{10}$$

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 $L_j\mathbf{z}_j = \mathbf{f}_j$.

Let us now consider a constant coefficient case of system (1):

$$x_k = \begin{cases} 0 & \text{for } k \le 0, \\ f_k + \sum_{j=1}^{m} a_j x_{k-j} & \text{for } 1 \le k \le n, \end{cases} \tag{11}$$



Observe that blocks (4) and (6) simplify to:

$$L = \begin{pmatrix} 1 & & & & & \\ -a_1 & \ddots & & & & \\ \vdots & \ddots & \ddots & & & \\ -a_m & & \ddots & \ddots & & \\ & \ddots & & \ddots & \ddots & \\ & & -a_m & \cdots & -a_1 & 1 \end{pmatrix} \in R^{q\times q}, \tag{12}$$

$$U = \begin{pmatrix} & -a_m & \cdots & -a_1 \\ & & \ddots & \vdots \\ & & & -a_m \\ \mathbf{0} & & & \end{pmatrix} \in R^{q\times q}, \tag{13}$$

and the recurrence relation (7) can be thus rewritten as:

$$\begin{cases} \mathbf{x}_1 = L^{-1}\mathbf{f}_1, \\ \mathbf{x}_j = L^{-1}\mathbf{f}_j - L^{-1}U\mathbf{x}_{j-1} \quad \text{for } j = 2, \ldots, p. \end{cases} \tag{14}$$

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

$$\mathbf{y}^1 = (1, y_2, \ldots, y_q)^T, \tag{15}$$

and

$$\mathbf{y}^k = (\underbrace{0, \ldots, 0}_{k-1}, 1, y_2, \ldots, y_{q-k+1})^T. \tag{16}$$

The method (10) can be now modified to define Algorithm 3:

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1, \\ \mathbf{x}_j = \mathbf{z}_j + \sum_{k=1}^{m} \alpha_j^k\mathbf{y}^k \quad \text{for } j = 2, \ldots, p, \end{cases} \tag{17}$$

where

$$L\mathbf{z}_j = \mathbf{f}_j \quad \text{for } j = 1, \ldots, p, \tag{18}$$

and $\mathbf{y}^k$ is given by (16) and

$$\begin{aligned} \alpha_j^1 &= a_m x_{(j-1)q-m+1} + a_{m-1}x_{(j-1)q-m+2} + \ldots + a_1 x_{(j-1)q} \\ \alpha_j^2 &= a_m x_{(j-1)q-m+2} + \ldots + a_2 x_{(j-1)q} \\ &\vdots \\ \alpha_j^m &= a_m x_{(j-1)q}. \end{aligned} \tag{19}$$

Methods (8), (10) and (17) 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 if 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. The speedup of Algorithm 3 should thus decrease only marginally for increasing values of $m$.

# 3 Arithmetical Complexity

The arithmetical complexity of the sequential algorithm (1) for the solution of the linear recurrence system is

$$T_1(n) = m\left[(n - \tfrac{m+1}{2})\right]\tau_a. \tag{20}$$

where $\tau_a$ denotes the time of the execution of the basic arithmetic operation (multiply and add). The complexity of Algorithm 1 is represented by:

$$T_{p,1}(n) = m\left[mp + (m+2)\tfrac{n}{p} - \tfrac{1}{2}(m^2+6m+1)\right]\tau_a. \tag{21}$$

(For parallel algorithms costs of communication are omitted as we plan to perform experiments on shared memory computers in which case we assume that the communication costs are negligible.) The complexity function of Algorithm 2 is:

$$T_{p,2}(n) = m\left[(\tfrac{3}{2}m+\tfrac{1}{2})p + (m+2)\tfrac{n}{p} - (m^2+\tfrac{9}{2}m+\tfrac{1}{2})\right]\tau_a, \tag{22}$$

whereas the complexity function of Algorithm 3 is

$$T_{p,3}(n) = m\left[(\tfrac{3}{2}m+\tfrac{1}{2})p + 3\tfrac{n}{p} - 2(2m+1)\right]\tau_a. \tag{23}$$

Note that functions (20), (22) and (23) have the same overall structure as arithmetical complexity functions for other divide-and-conquer algorithms (see for instance [5], [10]). In all cases the optimal number of processors is $P_{OPT} = O(\sqrt{n})$ so the solution to the problem can be obtained in $O(\sqrt{n})$ steps. Observe also that Algorithm 3 is much less sensitive to changes in the value of $m$.



# 4 Experimental Results

The experiments were performed on an 8-processor Cray Y-MP. We used CF77 Fortran compiler with autotasking facility (CMIC) which allows parallel execution on multiple CPUs. Each result presented here is an average of multiple runs on an empty machine where times were measured using the *timef* function. To simplify programming it was assumed that $n$ is divisible by $p$. The speedup of the algorithms was calculated against the sequential algorithm.

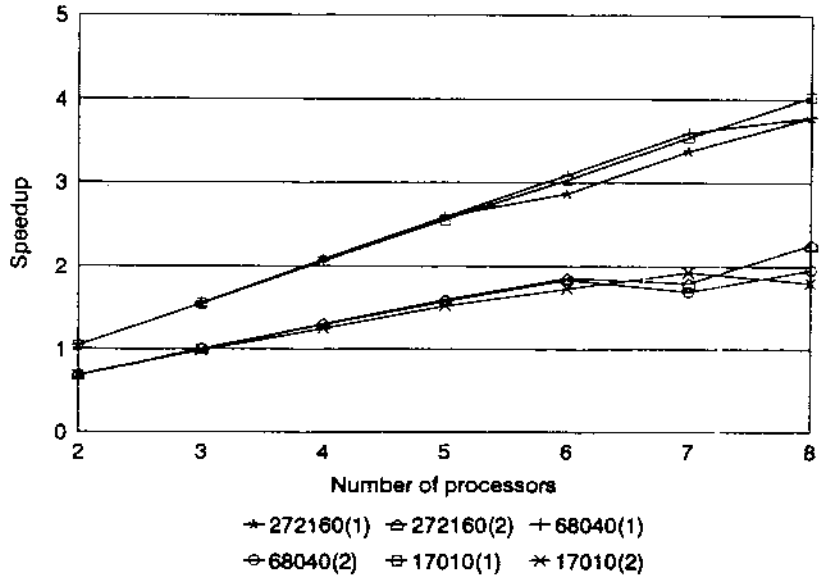


Figure 1. Performance of Algorithms 1 and 2 for the general case for 2 to 8 processors, $m=1$ and for various values of $n$.

Figure 1 compares the performance of Algorithms 1 and 2 applied to the general case problem for $m = 1$ and $n = 272160, 68040, 17010$. In each case Algorithm 2 outperforms Algorithm 1. Change in the value of $n$ has a minimal effect on the speedup of both algorithms. Similar results were obtained also for other values of $m$. The leveling-off effect related to the system overhead could not be observed for the small number of processors available. It is not clear what the reason for such a significant difference between the performance of Algorithms 1 and 2 is but it must be related to the Cray's hardware and software (in case of the Sequent this difference was much smaller [11]).

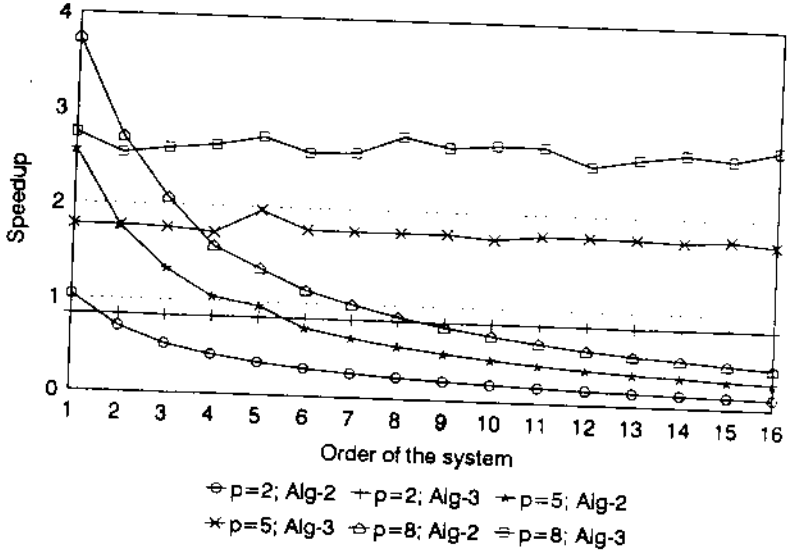


Figure 2. Performance of Algorithms 2 and 3 for the constant coefficient case for $n$=272160, $m$=1, ..., 16 and $p$=2, 5, 8.

Figure 2 compares the performance of Algorithms 2 and 3 for the constant-coefficient case for $n = 272160$, $p = 2, 5, 8$ and $m = 1, \ldots, 16$. As $m$ increases the performance of Algorithm 2 decreases rapidly whereas the performance of Algorithm 3 remains almost constant. These results follow closely the arithmetical complexity functions introduced in Section 4.

## 5 Conclusions

We have studied the behavior of three divide-and-conquer algorithms for the solution of a linear recurrence system of $n$ equations of order $m$. The experimental results approximate the predictions based on the theoretical model. At the same time more work needs to be done to properly explain the observed discrepancies. The proposed algorithms should be also tested on computers with a larger number of processors.



## References

[1] A. Borodin and I. Munro, *The computational complexity of algebraic and numerical problems* (American Elsevier, New York 1975).

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

[3] S.-C. Chen, Speedup of iterative programs in multiprocessing systems. Dissertation, University of Illinois at Urbana 1975.

[4] S.-C. Chen and D.J. Kuck, Time and parallel processor bounds for linear recurrence systems. *IEEE Trans. on Computers* 24(1975) 701-717.

[5] J. Dongarra and L. Johnsson, Solving Banded Systems on Parallel Processor. *Parallel Comput.* 5(1987) 219-246.

[6] D. Heller, A survey of parallel algorithms in numerical linear algebra. *SIAM Review* 20(1978) 740-777.

[7] A.C. Greenberg, R.E.Lander, M.S. Paterson and Z. Galil, Efficient parallel algorithms for linear recurrence computation, *Inf. Proc. Letters* 15(1982) 31-35.

[8] D.J. Kuck, *Structure of Computers and Computations* (Wiley, New York, 1978).

[9] J. Modi, *Parallel Algorithms and Matrix Computations* (Oxford University Press, Oxford, 1988).

[10] M. Paprzycki and I. Gladwell, Solving Almost Block Diagonal Systems on Parallel Computers. *Parallel Comput.* 17(1991) 133-153.

[11] M. Paprzycki and P. Stpiczyński, 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, 1994) – to appear.