J. Parallel Distrib. Comput. 87 (2016) 102–108

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Research note

A highly scalable parallel algorithm for solving Toeplitz tridiagonal systems of linear equations

Andrew V. Terekhov *

Institute of Computational Mathematics and Mathematical Geophysics, 630090, Novosibirsk, Russia
Novosibirsk State University, 630090, Novosibirsk, Russia

HIGHLIGHTS

- A parallel algorithm for solving Toeplitz tridiagonal systems is proposed.
- The algorithm allows an effective use of the 2D/3D fast Poisson solvers.
- The dependence of the speedup on the number of processors is almost linear.
- A large number of processors can be used (up to 16 384).

ARTICLE INFO

Article history:
Received 16 June 2014
Received in revised form
27 May 2015
Accepted 14 October 2015
Available online 24 October 2015

Keywords:
Parallel dichotomy algorithm
Thomas algorithm
Poisson equation
Toeplitz matrices

ABSTRACT

Based on a modification of the dichotomy algorithm, we propose a novel parallel procedure for solving tridiagonal systems of equations with Toeplitz matrices. Taking the structure of the Toeplitz matrices, we may substantially reduce the number of the preliminary calculations of the dichotomy algorithm, which makes possible to efficiently solve systems of linear equations with both one and several right-hand sides. On examples of solving the 2D Poisson equation by the variable separation method and the 3D Poisson equation by a combination of the alternating direction implicit and the variable separation methods we show that the computation accuracy is comparable with the sequential version of the Thomas method, the dependence of the speedup on the number of processors being almost linear. The proposed modification is aimed at parallel implementation of a broad class of numerical methods including the Toeplitz tridiagonal matrices inversion.

1. Introduction

The implementation of many numerical algorithms, such as the methods of multigrid line relaxation [32], alternating direction implicit [17,36], separation variable and cyclic reduction [10] for solving elliptic equations and also the problems of constructing splines [25], etc. requires solving tridiagonal systems of linear algebraic equations (SLAEs) with the Toeplitz matrices

$$T = \begin{pmatrix} t_0 & t_1 & & & 0 \\ t_{-1} & t_0 & t_1 & & \\ & \ddots & \ddots & \ddots & \\ & & t_{-1} & t_0 & t_1 \\ 0 & & & t_{-1} & t_0 \end{pmatrix} \equiv \mathrm{tridiag}\{\dots, t_{-1}, t_0, t_1, \dots\}. \tag{1}$$

Since in solving modern problems of mathematical modeling, the size and the number of such problems can reach several tens of thousands, such computations must be performed on a supercomputer.

Parallel algorithms for solving SLAEs with tridiagonal Toeplitz matrices were investigated in [14,28,37,12,18,2]. These procedures are particular cases of the algorithms for solving tridiagonal SLAEs such as the cyclic reduction method [11,10], partitioning methods [35,13,38,30,29] as well as various parallel implementations of the Gauss elimination method [27,21,16,19]. One of the main issues that is taken into account in all the methods proposed is decreasing the time taken for inter-processor communications. Another difficulty of solving the SLAEs with tridiagonal matrices is associated with strictly determined order of calculation, and algorithms used do not admit asynchronous parallel implementations. A standard example is the cyclic reduction method [2,11], where the main problem of this class of algorithms is that when there are a large number of processors, many of the processors stand idle at the end of the forward elimination process and at the beginning of the backward elimination process. In this case, to

* Correspondence to: Institute of Computational Mathematics and Mathematical Geophysics, 630090, Novosibirsk, Russia.
E-mail address: andrew.terekhov@mail.ru.

http://dx.doi.org/10.1016/j.jpdc.2015.10.004
0743-7315/

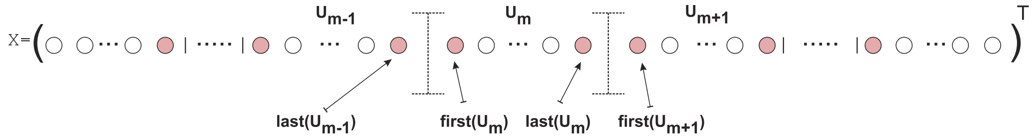

Fig. 1. The *first* and *last* components of the solution vector.

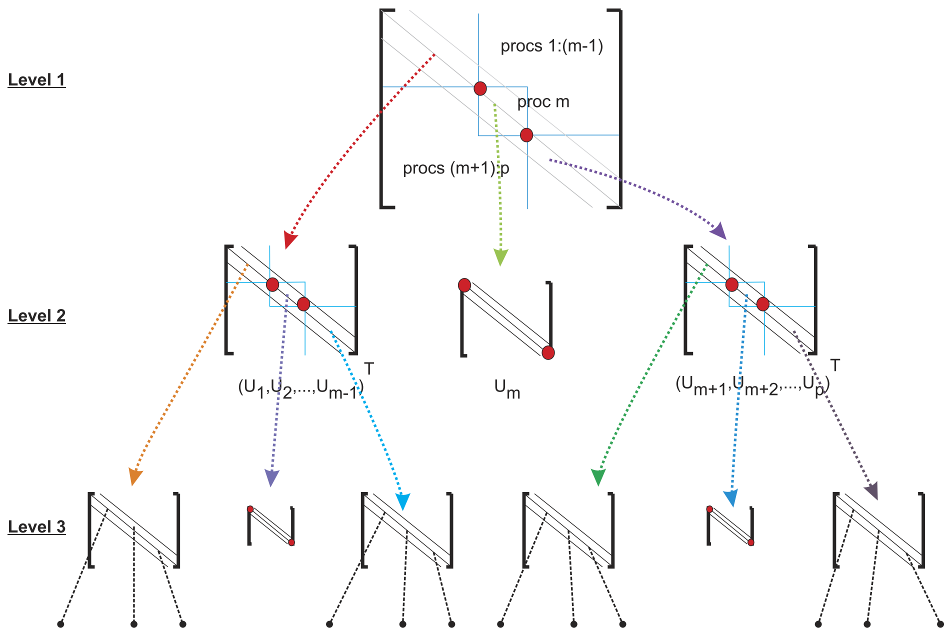

Fig. 2. Partitioning of a tridiagonal SLAE into independent subsystems by the dichotomy algorithm.

overcome such a difficulty for the dominant diagonal matrices it is possible to use the incomplete cyclic reduction algorithm [8] allowing a decrease in the number of the reduction levels thus diminishing the idle time of a greater number of processors. Also, the property of the diagonal dominance is applied for decreasing the time of inter-processor communications in the algorithms [30] based on the Sherman and Morrison matrix modification formula [24]. Many of the existing approaches are successfully implemented for GPU-accelerated computing [39,40,5,9], however for distributed-memory supercomputers, combining thousands of processors, the efficiency considerably decreases.

For solving this problem, a method called the dichotomy algorithm [31] was proposed for solving a series of problems with a constant matrix and different right-hand sides

$$A\mathbf{X}_i = \mathbf{F}_i, \quad i = 1, 2, \ldots, M,$$
$$A = \mathrm{tridiag}\{\ldots, c_j, b_j, a_j, \ldots\}_{j=1}^{N}, \tag{2}$$

where M is the number of the right hand sides, N is the order of the matrix. The dichotomy algorithm is a representative of the class of algorithms known as “Divide and Conquer” and a sufficient condition for the applicability of the dichotomy algorithm is the diagonal dominance of the matrix of a SLAE. The advantage of the dichotomy algorithm consists in that it makes possible to attain a thousandfold speedup for problem (2) not only in theory but also in practice, that has been verified [7]. A high efficiency of the dichotomy algorithm for distributed-memory supercomputers is provided by the fact that solving a tridiagonal SLAE is made in two stages. At the first stage named preliminary, four auxiliary vectors independent of the right-hand sides $\mathbf{F}_i$ are calculated. At the second stage, using the auxiliary vectors on each dichotomy level for computing the *first*$\{\mathbf{U}_m\}$ and the *last*$\{\mathbf{U}_m\}$ components of the vector solution (Fig. 1), the tridiagonal SLAE obtained at the previous level is partitioned into three independent subsystems of lesser dimensions (Fig. 2). Thus, all the *first*$\{\mathbf{U}_m\}$, *last*$\{\mathbf{U}_m\}$ components are calculated in $\lceil \log_2 p \rceil$ steps, where p is the number of processors. After partitioning the original SLAE into p independent subsystems the rest components of the solution vector can be calculated by any sequential algorithm for solving a tridiagonal SLAE.

As for the number of arithmetic operations, the dichotomy algorithm is comparable with the cyclic reduction method. However, the time of interprocess or interactions is much smaller in the dichotomy algorithm comparable with other available algorithms. This is because the implementation of the dichotomy process on a supercomputer is reduced to calculating the sum of series for distributed data. The property of commutability and the commutative and associative properties of addition enable a considerable reduction in the total computation time using the interprocess or interaction optimization algorithms [4,3,6,33,26].

But as a matter of fact, the preliminary stage with $O(N)$ arithmetic operations costs, where N is the dimension of the SLAE, does not make possible to efficiently use the dichotomy algorithm for solving the SLAE with one right-hand side. The number of the right-hand sides should be sufficient for the costs of the preliminary stage be less than those of the second stage of the dichotomy algorithm. However, for the Toeplitz matrices, an efficient preliminary procedure can be constructed with the number of the arithmetic operations of order $O(N/p+\log_2 p)$. Thus, a modification of the preliminary stage in the dichotomy algorithm enables us to effectively solve SLAEs with both one and several right-hand sides.

2. The parallel dichotomy algorithm for the Toeplitz matrices

Before discussing of the dichotomy algorithm, consider the question of mapping the data of the problem onto many processors.