Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational and Applied Mathematics 212 (2008) 260–271

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

# A communication-less parallel algorithm for tridiagonal Toeplitz systems

Jeffrey M. McNally[a,*], L.E. Garey[b], R.E. Shaw[b]

[a] *Department of Mathematics, Statistics and Computer Science, Saint Francis Xavier University, Antigonish, Nova Scotia, Canada B2G 2W5*
[b] *Department of Computer Science and Applied Statistics, University of New Brunswick, Saint John, New Brunswick, Canada E2L 4L5*

Received 26 October 2006

**Abstract**

Diagonally dominant tridiagonal Toeplitz systems of linear equations arise in many application areas and have been well studied in the past. Modern interest in numerical linear algebra is often focusing on solving classic problems in parallel. In McNally [Fast parallel algorithms for tri-diagonal symmetric Toeplitz systems, MCS Thesis, University of New Brunswick, Saint John, 1999], an $m$ processor Split & Correct algorithm was presented for approximating the solution to a symmetric tridiagonal Toeplitz linear system of equations. Nemani [Perturbation methods for circulant-banded systems and their parallel implementation, Ph.D. Thesis, University of New Brunswick, Saint John, 2001] and McNally (2003) adapted the works of Rojo [A new method for solving symmetric circulant tri-diagonal system of linear equations, Comput. Math. Appl. 20 (1990) 61–67], Yan and Chung [A fast algorithm for solving special tri-diagonal systems, Computing 52 (1994) 203–211] and McNally et al. [A split-correct parallel algorithm for solving tri-diagonal symmetric Toeplitz systems, Internat. J. Comput. Math. 75 (2000) 303–313] to the non-symmetric case. In this paper we present relevant background from these methods and then introduce an $m$ processor scalable communication-less approximation algorithm for solving a diagonally dominant tridiagonal Toeplitz system of linear equations.


*MSC:* 65F05; 65Y05; 68M14

*Keywords:* Parallel computing; Communication-less algorithms; Toeplitz; Tridiagonal; Distributed; Linear systems; Fast algorithms; Perturbation analysis

## 1. Introduction

Consider the tridiagonal Toeplitz system of linear equations $A\mathbf{x} = \mathbf{b}$ given by

$$\begin{bmatrix} d & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & & & & \\ & & \alpha & d & 1 & \\ & & & \alpha & d & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}, \tag{1}$$

* Corresponding author. Tel.: +1 902 867 3341; fax: +1 902 867 3302.
*E-mail address:* jmcnally@stfx.ca (J.M. McNally).
*URL:* http://www.people.stfx.ca/JMCNALLY/ (J.M. McNally).

0377-0427/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2006.12.001

where strict diagonal dominance is given by the condition $|d| > 1 + |\alpha|$. It has been well established in McNally [4], McNally et al. [6] and Nemani [7] that this is a well motivated and frequently occurring problem which arises in such application areas as second order differential equations, signal/image processing and numerical solutions to integral equations [3,9].

In Rojo [8] an algorithm was presented for exactly solving system (1) when $\alpha = 1$ based upon the perturbation $A = LU + P$ and in Yan and Chung [10] a fast algorithm was presented for approximating the solution to within a given tolerance. McNally [4] and McNally et al. [6] presented fast parallel algorithms for approximating the solution to system (1) in parallel when $\alpha = 1$ through the use of a Split & Correct algorithm. This algorithm performs well on both shared and distributed memory systems as it incorporates only minimal inter-processor communication. Nemani [7] and McNally [5] both adapted these methods for solving the general non-symmetric problem of system (1).

Here we present relevant background from these methods and introduce a fast $m$ processor parallel scalable communication-less approximation algorithm for solving a diagonally dominant tridiagonal Toeplitz system of linear equations. This algorithm, based upon two perturbations $A = LU + P$ and $A = UL + P'$, eliminates the correction step required by all previous algorithms, and in doing so, also eliminates the need for any inter-processor communication. An error analysis will be presented which results in a tighter error bounds that previously given for this problem [5].

## 2. Background

Rojo [8], Yan and Chung [10], and McNally et al. [6] previously dealt with solving symmetric diagonally dominant Toeplitz system of linear equations. In Garey and Shaw [1,2], Nemani [7] and McNally [5] the non-symmetric case was addressed and solved in similar fashion.

The initial step used in the aforementioned works was to determine a matrix splitting which allowed for a fast solution due to a Toeplitz LU factorization of the perturbed matrix. Here we consider the splitting $A = LU + P$ given by

$$\begin{bmatrix} 1 & & & & \\ r_1 & 1 & & & \\ & \ddots & & & \\ & & r_1 & 1 & \\ & & & r_1 & 1 \end{bmatrix} \begin{bmatrix} r_2 & 1 & & & \\ & r_2 & 1 & & \\ & & \ddots & & \\ & & & r_2 & 1 \\ & & & & r_2 \end{bmatrix} + \begin{bmatrix} d - r_2 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix},$$

where

$$r_2 = \frac{d + \sqrt{d^2 - 4\alpha}}{2} \quad \text{or} \quad r_2 = \frac{d - \sqrt{d^2 - 4\alpha}}{2}.$$

In Nemani [7], it was shown that, for the given system, $r_2$ can always be chosen such that $r_2$ is real with $|r_2| > 1$. Also shown is $r_1 = d - r_2$, $r_1$ is real and $|r_1| < 1$.

The system $LU\mathbf{z} = \mathbf{b}$ is readily solved in $4n + \mathrm{O}(1)$ operations using forward/backward substitution. The method used by Rojo [8] is to determine a vector of length $n$ which exactly corrects $\mathbf{z}$ using the Sherman–Morrison formula for a total of $8n + 2\log n + \mathrm{O}(1)$ operations. Yan and Chung [10] determined a vector $\mathbf{p}$ of length $t$ such that $\mathbf{z}$ could be corrected to within given tolerance resulting in an approximate solution $\hat{\mathbf{x}}$ to the system $A\mathbf{x} = \mathbf{b}$. Provided $t < n$ (in general we see that $t \ll n$) the method requires only $4n + 2t + \mathrm{O}(1)$ operations.

In McNally [4] and McNally et al. [6] algorithms were presented which use a matrix splitting to create a set of $m$ sub-systems which could then be solved in parallel. A set of correction vectors $\mathbf{p}_i$ and $\mathbf{q}_i$, $i = 1(1)m$, each of length $t$ were then used to correct these solutions to within acceptable tolerance levels. Once again, the value of $t$ was easily determined and provided that $2mt < n$ these methods could be applied. On two processors this took only $2n + 3t + \mathrm{O}(1)$ operations, and on $m$ processors it required $4\frac{n}{m} + 4t + \mathrm{O}(1)$ operations.

These methods, referred to as Split & Correct methods, consist of a matrix splitting and a correction step. In requiring a correction step, the parallel methods also required a small communication step where each processor was required to send and receive a value to and from each of its adjacent processors.

## 3. The Stacked method—2 processors

Given system (1), the Split & Correct algorithm introduced in McNally [4] divided the system into two independent systems which could be solved in parallel using the following steps:

(1) Solve $LU\mathbf{x}' = \mathbf{b}'$ on each processor.
(2) Communicate a single value to each adjacent processors.
(3) Perform a correction on $\mathbf{x}'$ resulting in an approximate solution $\tilde{\mathbf{x}}$.

Here we show that by over-extending the two sub-systems, we not only eliminating the requirements for the correction steps, but also the requirements for communication between processors. Consider solving in parallel the following two systems of equations, each of size $(\frac{n}{2}+t)\times(\frac{n}{2}+t)$

$$\begin{bmatrix} d & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & r_2 \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_{\frac{n}{2}+t-1} \\ x'_{\frac{n}{2}+t} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{\frac{n}{2}+t-1} \\ b_{\frac{n}{2}+t} \end{bmatrix} \tag{2}$$

and

$$\begin{bmatrix} r_2 & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & d \end{bmatrix} \begin{bmatrix} x''_{\frac{n}{2}+1-t} \\ x''_{\frac{n}{2}+2-t} \\ \vdots \\ x''_{n-1} \\ x''_n \end{bmatrix} = \begin{bmatrix} b_{\frac{n}{2}+1-t} \\ b_{\frac{n}{2}+2-t} \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}. \tag{3}$$

These systems, ($A'\mathbf{x}' = \mathbf{b}'$ and $A''\mathbf{x}'' = \mathbf{b}''$, respectively) can be solved using fast Toeplitz $LU$ decomposition with forward/backwards substitution, and a fast Toeplitz $UL$ decomposition with backward/forwards substitution, respectively.

Examining system (2) in Yan and Chung [10] tells us that only the last $t$ equations of $\mathbf{x}'$ differ from the true solution $\mathbf{x}$ by more than the given tolerance $\varepsilon$. Therefore, assuming $2t < n$ the first $\frac{n}{2}$ equations do not require correcting and are acceptable as approximate solutions to the true solution without being corrected. For system (3) we see that only the first $t$ equations of $\mathbf{x}''$ are not within the given acceptable tolerance levels (easily shown from a $UL$ implementation of the algorithm in Yan and Chung [10]) implying that the last $\frac{n}{2}$ solutions are within the given tolerance and do not require correction. Combining these results, we present the following theorem:

**Theorem 1.** *The solution* $\tilde{\mathbf{x}}$ *defined as*

$$\tilde{\mathbf{x}} = [x'_1, x'_2, \ldots, x'_{\frac{n}{2}}, x''_{\frac{n}{2}+1}, x''_{\frac{n}{2}+2}, \ldots, x''_{n-1}, x''_n]$$

*is a valid approximate solution to the original problem* $A\mathbf{x} = \mathbf{b}$ *provided* $t < \frac{n}{2}$ *where*

$$t \geqslant \frac{\ln \varepsilon - \ln\left(\dfrac{1+|r_2|}{|r_2 - r_1|}\dfrac{\left(\dfrac{|r_1|}{|r_2|}+1\right)}{|d|-|\alpha|-1}\right)}{\ln\, g}$$

*and* $g = \max\{|r_1|, |\frac{1}{r_2}|\}$. *The total operation count for this algorithm is* $2n + 2t + \mathrm{O}(1)$ *with no correction step or communication between processors.*

**Proof.** The first $\frac{n}{2}+t$ equations of the original system $A\mathbf{x}=\mathbf{b}$ can be written as

$$\begin{bmatrix} d & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & r_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{\frac{n}{2}+t-1} \\ x_{\frac{n}{2}+t} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{\frac{n}{2}+t-1} \\ b_{\frac{n}{2}+t} - r_1 x_{\frac{n}{2}+t} - x_{\frac{n}{2}+t+1} \end{bmatrix}$$

and recall that system (2) is

$$\begin{bmatrix} d & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & r_2 \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_{\frac{n}{2}+t-1} \\ x'_{\frac{n}{2}+t} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{\frac{n}{2}+t-1} \\ b_{\frac{n}{2}+t} \end{bmatrix}.$$

Subtracting these two equations and letting $h'_j = x_j - x'_j$ results in the 3-term recurrence relation

$$\alpha h'_{j-1} + dh'_j + h'_{j+1} = 0, \quad j = 2 \ldots \left(\frac{n}{2} + t - 1\right)$$

with the boundary conditions

$$dh'_1 + h'_2 = 0$$

and

$$\alpha h'_{\frac{n}{2}+t-1} + r_2 h'_{\frac{n}{2}+t} = -r_1 x_{\frac{n}{2}+t} - x_{\frac{n}{2}+t+1}.$$

The solution to this difference equation is well known to us from both Rojo [8] and Yan and Chung [10] as

$$h'_j = c_1(-r_1)^j + c_2(-r_2)^j.$$

Substituting this solution into the two boundary conditions gives us

$$c_2 = -c_1 = (-r_2)^{-\frac{n}{2}-t} \frac{-r_1 x_{\frac{n}{2}+t} - x_{\frac{n}{2}+t+1}}{r_2 - r_1}$$

which results in

$$x_j - x'_j = \frac{r_1 x_{\frac{n}{2}+t} + x_{\frac{n}{2}+t+1}}{r_2 - r_1}(-r_1)^j(-r_2)^{-\frac{n}{2}-t} - \frac{r_1 x_{\frac{n}{2}+t} + x_{\frac{n}{2}+t+1}}{r_2 - r_1}(-r_2)^{j-\frac{n}{2}-t}.$$

Taking the sup-norm of the above equations results in

$$|x_j - x'_j| \leqslant \left|\frac{1}{r_2}\right|^t \frac{1+|r_1|}{|r_2 - r_1|} \left|(-r_1)^j(-r_2)^{-\frac{n}{2}} - (-r_2)^{j-\frac{n}{2}}\right| \|\mathbf{x}\|.$$

We are only interested in this equation for $j = 1 \ldots \frac{n}{2}$. Observing

$$\left|\frac{(-r_1)^j}{(-r_2)^{\frac{n}{2}}} - \frac{(-r_2)^j}{(-r_2)^{-\frac{n}{2}}}\right| \leqslant \frac{|r_1|^j}{|r_2|^{\frac{n}{2}}} + \frac{|r_2|^j}{|r_2|^{\frac{n}{2}}} < \frac{|r_1|}{|r_2|} + 1$$

results in

$$|x_j - x'_j| < \left|\frac{1}{r_2}\right|^t \frac{1+|r_1|}{|r_2 - r_1|} \left(\frac{|r_1|}{|r_2|} + 1\right) \|\mathbf{x}\|.$$

From Yan and Chung [10] it is shown that

$$\|\mathbf{x}\| < \frac{1}{|d| - |\alpha| - 1}\|\mathbf{b}\|$$

resulting in

$$|x_j - x'_j| < \left|\frac{1}{r_2}\right|^t \frac{1 + |r_1|}{|r_2 - r_1|} \frac{\left(\frac{|r_1|}{|r_2|} + 1\right)}{|d| - |\alpha| - 1}\|\mathbf{b}\|.$$

Consider now system (3). The final $\frac{n}{2} + t$ equations of the original system $A\mathbf{x} = \mathbf{b}$ can be written as

$$\begin{bmatrix} r_2 & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & d \end{bmatrix} \begin{bmatrix} x_{\frac{n}{2}-t+1} \\ x_{\frac{n}{2}-t+2} \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_{\frac{n}{2}-t+1} - \alpha x_{\frac{n}{2}-t} - r_1 x_{\frac{n}{2}-t+1} \\ b_{\frac{n}{2}-t+2} \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}.$$

Subtracting from this system (3) and letting $h''_j = x_j - x''_j$ results in the difference equation

$$\alpha h''_{j-1} + d h''_j + h''_{j+1} = 0, \quad j = \left(\frac{n}{2} - t + 2\right) \cdots (n-1)$$

with the boundary condition

$$r_2 h''_{\frac{n}{2}-t+1} + h''_{\frac{n}{2}-t+2} = -\alpha x_{\frac{n}{2}-t} - r_1 x_{\frac{n}{2}-t+1}$$

and

$$\alpha h''_{n-1} + d h''_n = 0.$$

Again the solution to this difference equation is well known to us as

$$h''_j = c_1(-r_1)^j + c_2(-r_2)^j$$

which we choose to write as

$$h''_j = w_1(-r_1)^{j-\frac{n}{2}} + w_2(-r_2)^{j-\frac{n}{2}}.$$

Substituting this into our two boundary conditions results in

$$w_1 = \frac{r_2 x_{\frac{n}{2}-t} + x_{\frac{n}{2}-t+1}}{(r_2 - r_1)}(-r_1)^t$$

and

$$w_2 = -\frac{(-r_1)^{\frac{n}{2}+1}}{(-r_2)^{\frac{n}{2}+1}} \frac{r_2 x_{\frac{n}{2}-t} + x_{\frac{n}{2}-t+1}}{(r_2 - r_1)}(-r_1)^t.$$

This gives the difference between this approximate solution $\mathbf{x}''$ and the true solution $\mathbf{x}$ for the given indices as

$$x_j - x''_j = \frac{r_2 x_{\frac{n}{2}-t} + x_{\frac{n}{2}-t+1}}{(r_2 - r_1)}(-(-r_1)^{j-\frac{n}{2}+t} + (-r_1)^{\frac{n}{2}+1+t}(-r_2)^{j-n-1}).$$

Taking the sup-norm of both sides and considering only the cases $j = \frac{n}{2} \ldots n$ results in

$$|x_j - x_j''| < |r_1|^t \frac{1 + |r_2|}{|r_2 - r_1|} \frac{\left(\frac{|r_1|}{|r_2|} + 1\right)}{|d| - |\alpha| - 1} \|\mathbf{b}\|.$$

Defining $\tilde{\mathbf{x}}$ as before, letting $g = \max\{|r_1|, \left|\frac{1}{r_2}\right|\}$, and recalling that $|r_2| > |r_1|$ results in

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| < g^t \frac{1 + |r_2|}{|r_2 - r_1|} \frac{\left(\frac{|r_1|}{|r_2|} + 1\right)}{|d| - |\alpha| - 1} \|\mathbf{b}\|.$$

Setting this less than the desired level of tolerance $\varepsilon\|\mathbf{b}\|$ gives an equation for $t$ as

$$t \geqslant \frac{\ln \varepsilon - \ln\left(\frac{1 + |r_2|}{|r_2 - r_1|}\left(\frac{|r_1|}{|r_2|} + 1\right)\frac{1}{|d| - |\alpha| - 1}\right)}{\ln\, g},$$

where $t < \frac{n}{2}$ is also necessary. □

The following table demonstrates some $t$ values for given values of $|d| - |\alpha| - 1$ and $\varepsilon$ for the cases of $\alpha = 10$, 1 and 0.05, respectively.

| $\|d\| - \|\alpha\| - 1/\varepsilon$ | $10^{-4}$ | $10^{-8}$ | $10^{-16}$ |
|---|---|---|---|
| 0.001 | 147737 | 230635 | 396433 |
| 0.01 | 12709 | 21804 | 27594 |
| 0.1 | 1070 | 1905 | 3574 |
| 0.5 | 190 | 361 | 704 |
| 2 | 45 | 92 | 185 |
| 4 | 23 | 49 | 100 |
| 6 | 16 | 34 | 72 |
| 0.001 | 641 | 932 | 1515 |
| 0.01 | 168 | 260 | 445 |
| 0.1 | 43 | 72 | 130 |
| 0.5 | 16 | 29 | 56 |
| 2 | 7 | 14 | 28 |
| 4 | 5 | 10 | 21 |
| 6 | 4 | 9 | 18 |
| 0.001 | 16075 | 24830 | 42340 |
| 0.01 | 1396 | 2276 | 4036 |
| 0.1 | 123 | 216 | 401 |
| 0.5 | 26 | 48 | 92 |
| 2 | 8 | 17 | 33 |
| 4 | 5 | 11 | 23 |
| 6 | 4 | 9 | 19 |

We observe that as the system loses symmetry, the calculated values of $t$ increase. $T$ values of $t$, however, are not significant enough to cause overall operational concern when $N$ is of sufficient size such as in problems with very small step sizes demanding precise accuracy. We also note the effect that diagonal dominance has on $t$.

## 4. The Stacked method—*m* processors

To expand the 2 processor Stacked method to $m$ processors we propose a new set of systems to solve. Consider the following $m$ systems of equations (the first and last are each of size $(\frac{n}{m}+t)\times(\frac{n}{m}+t)$ and the middle $m-2$ of which are of size $(\frac{n}{m}+2t)\times(\frac{n}{m}+2t)$)

$$\begin{bmatrix} d & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & r_2 \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_{\frac{n}{m}+t-1} \\ x'_{\frac{n}{m}+t} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{\frac{n}{m}+t-1} \\ b_{\frac{n}{m}+t} \end{bmatrix}, \tag{4}$$

$$\begin{bmatrix} r_2 & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & d \end{bmatrix} \begin{bmatrix} x''^{i}_{\frac{in}{m}+1-t} \\ x''^{i}_{\frac{in}{m}+2-t} \\ \vdots \\ x''^{i}_{\frac{(i+1)n}{m}+t-1} \\ x''^{i}_{\frac{(i+1)n}{m}+t} \end{bmatrix} = \begin{bmatrix} b_{\frac{in}{m}+1-t} \\ b_{\frac{in}{m}+2-t} \\ \vdots \\ b_{\frac{(i+1)n}{m}+t-1} \\ b_{\frac{(i+1)n}{m}+t} \end{bmatrix} \tag{5}$$

and

$$\begin{bmatrix} r_2 & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & d \end{bmatrix} \begin{bmatrix} x'''_{\frac{n}{m}+1-t} \\ x'''_{\frac{n}{m}+2-t} \\ \vdots \\ x'''_{n-1} \\ x'''_{n} \end{bmatrix} = \begin{bmatrix} b_{\frac{n}{m}+1-t} \\ b_{\frac{n}{m}+2-t} \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}. \tag{6}$$

Each of these systems (4), (5) [for $i=1,\ldots,m-2$], and (6) can be solved using fast Toeplitz $LU$ (or $UL$) decomposition as described earlier.

In system (4) Yan and Chung [10] tells us that only the last $t$ value of $\mathbf{x}'$ differ from the true solution $\mathbf{x}$ by more than the given tolerance $\varepsilon$. This gives us that the first $\frac{n}{m}$ equations are acceptable as approximate solutions to within given tolerance. In system (6), we observe that only the first $t$ values of $\mathbf{x}'''$ are not within the given acceptable tolerance levels implying that the last $\frac{n}{m}$ solutions are acceptable as approximate solutions to within given tolerance.

For each of the middle systems (5) [$i=1,\ldots,m-2$] we will show that only the first $t$ and last $t$ values of $\mathbf{x}''^{i}$ are not within the given acceptable tolerance levels and as such that the middle $\frac{n}{m}$ solutions are. From these results we state the following:

**Theorem 2.** *The solution* $\tilde{\mathbf{x}}$ *defined as*

$$\tilde{\mathbf{x}} = \left[ x'_1,\ldots,x'_{\frac{n}{m}}, x''^{1}_{\frac{n}{m}+1},\ldots,x''^{1}_{\frac{2n}{m}}, x''^{2}_{\frac{2n}{m}+1},\ldots,x''^{m-2}_{\frac{(m-1)n}{m}}, x'''_{\frac{(m-1)n}{m}+1},\ldots,x'''_{n} \right]$$

*is a valid approximate solution to the original problem* $A\mathbf{x}=\mathbf{b}$ *provided* $2mt<n$ *where*

$$t > \frac{\ln\varepsilon - \ln\left(\frac{1+|r_2|}{|r_2-r_1|}\left(1+\frac{|r_1|}{|r_2|}+|r_1|\right)\frac{1}{(|d|-|\alpha|-1)}\right)}{\ln g}$$

*and* $g=\max\{|r_1|,|\frac{1}{r_2}|\}$ *and the total operation count for this algorithm is* $4\frac{n}{m}+6t+\mathrm{O}(1)$ *with no correction step or communication between processors.*

**Proof.** From Section 3 of this paper, we see that the first and last systems both have errors which can be stated as

$$|x_j - \tilde{x}_j| < g^t \frac{1+|r_2|}{|r_2 - r_1|} \frac{\left(\frac{|r_1|}{|r_2|} + 1\right)}{|d| - |\alpha| - 1} \|\mathbf{b}\|$$

for $j = 1, \ldots, \frac{n}{m}$ and $j = \frac{(m-1)n}{m+1}, \ldots, n$ where $g = \max\{|r_1|, |\frac{1}{r_2}|\}$. The middle $m-2$ systems all have the same form given by

$$\begin{bmatrix} r_2 & 1 & & & & \\ \alpha & d & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \alpha & d & 1 \\ & & & & \alpha & d \end{bmatrix} \begin{bmatrix} x''^i_{\frac{in}{m}+1-t} \\ x''^i_{\frac{in}{m}+2-t} \\ \vdots \\ x''^i_{\frac{(i+1)n}{m}+t-1} \\ x''^i_{\frac{(i+1)n}{m}+t} \end{bmatrix} = \begin{bmatrix} b_{\frac{in}{m}+1-t} \\ b_{\frac{in}{m}+2-t} \\ \vdots \\ b_{\frac{(i+1)n}{m}+t-1} \\ b_{\frac{(i+1)n}{m}+t} \end{bmatrix}.$$

Subtracting this system from the corresponding equations in the original system and setting $h_j = x_j - x''^i_j$ results in the difference equation

$$\alpha h_{j-1} + d h_j + h_{j+1} = 0$$

with two boundary conditions

$$r_2 h_{\frac{in}{m}+1-t} + h_{\frac{in}{m}+2-t} = -\alpha x_{\frac{in}{m}-t} - r_1 x_{\frac{in}{m}+1-t}$$

and

$$\alpha h_{\frac{(i+1)n}{m}+t-1} + d h_{\frac{(i+1)n}{m}+t} = -x_{\frac{(i+1)n}{m}+t+1}.$$

The solution to this difference equation is well known, and can be written as

$$h_j = w_1(-r_1)^{j-\frac{in}{m}} + w_2(-r_2)^{j-\frac{in}{m}}.$$

Substituting this into the two boundary conditions gives

$$w_1 = (-r_1)^t \frac{r_2 x_{\frac{in}{m}-t} + x_{\frac{in}{m}+1-t}}{r_2 - r_1}$$

and

$$\begin{aligned} w_2 &= \frac{x_{\frac{(i+1)n}{m}+t+1} - (-r_1)^{\frac{n}{m}+t+1} w_1}{(-r_2)^{\frac{n}{m}+t+1}} \\ &= \frac{(r_2 - r_1) x_{\frac{(i+1)n}{m}+t+1} - (-r_1)^{\frac{n}{m}+2t+1}(r_2 x_{\frac{in}{m}-t} + x_{\frac{in}{m}+1-t})}{(r_2 - r_1)(-r_2)^{\frac{n}{m}+1+t}} \end{aligned}$$

resulting in

$$\begin{aligned} x_j - x''^i_j = (-r_1)^t &\frac{r_2 x_{\frac{in}{m}-t} + x_{\frac{in}{m}+1-t}}{r_2 - r_1} (-r_1)^{j-\frac{in}{m}} \\ &+ \frac{(r_2 - r_1) x_{\frac{(i+1)n}{m}+t+1} - (-r_1)^{\frac{n}{m}+2t+1}(r_2 x_{\frac{in}{m}-t} + x_{\frac{in}{m}+1-t})}{(r_2 - r_1)(-r_2)^{\frac{(i+1)n}{m}+1+t-j}}. \end{aligned}$$

Letting $g = \max\{|r_1|, |\frac{1}{r_2}|\}$ and taking the absolute value of each side results in

$$|x_j - x''^i{}_j| \leqslant g^t \frac{1+|r_2|}{|r_2 - r_1|} \left( |r_1|^{j-\frac{in}{m}} + \frac{(1+|r_1|^{\frac{n}{m}+2t+1})}{|r_2|^{\frac{j-(i+1)n}{m}-1}} \right) \|\mathbf{x}\|$$

which is bounded for each $i$ (for the values $j$ of interest) by

$$|x_j - x''^i{}_j| \leqslant g^t \frac{1+|r_2|}{|r_2 - r_1|} \left( 1 + \frac{|r_1|}{|r_2|} + |r_1| \right) \|\mathbf{x}\|.$$

Again recalling the bounds of $\|\mathbf{x}\|$ from Yan and Chung [10] results in

$$|x_j - x''_j| \leqslant g^t \frac{1+|r_2|}{|r_2 - r_1|} \frac{\left(1 + \frac{|r_1|}{|r_2|} + |r_1|\right)}{|d| - |\alpha| - 1} \|\mathbf{b}\|, \quad j = \frac{in}{m} + 1, \ldots, \frac{(i+1)n}{m}.$$

which is necessarily larger than the error bounds for system (4) and (6) giving us the overall result

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| \leqslant g^t \frac{1+|r_2|}{|r_2 - r_1|} \frac{\left(1 + \frac{|r_1|}{|r_2|} + |r_1|\right)}{|d| - |\alpha| - 1} \|\mathbf{b}\|.$$

Setting this error to be less than a given tolerance $\varepsilon\|\mathbf{b}\|$ results in a $t$ value of

$$t > \frac{\ln \varepsilon - \ln \left( \frac{1+|r_2|}{|r_2 - r_1|} \left( 1 + \frac{|r_1|}{|r_2|} + |r_1| \right) \frac{1}{|d| - |\alpha| - 1} \right)}{\ln g},$$

where of course we also require that $2mt < n$. The following table demonstrates some $t$ values for given values of $|d| - |\alpha| - 1$ and $\varepsilon$ when $\alpha = 10, 1, 0.05$.

| $\lvert d\rvert - \lvert\alpha\rvert - 1/\varepsilon$ | $10^{-4}$ | $10^{-8}$ | $10^{-16}$ |
|---|---|---|---|
| 0.001 | 153356 | 236455 | 402252 |
| 0.01 | 13291 | 21586 | 38176 |
| 0.1 | 1128 | 1963 | 3632 |
| 0.5 | 201 | 373 | 715 |
| 2 | 48 | 94 | 188 |
| 4 | 24 | 50 | 102 |
| 6 | 17 | 35 | 73 |
| 0.001 | 654 | 945 | 1528 |
| 0.01 | 172 | 264 | 449 |
| 0.1 | 44 | 73 | 132 |
| 0.5 | 17 | 30 | 56 |
| 2 | 7 | 14 | 28 |
| 4 | 5 | 10 | 21 |
| 6 | 4 | 9 | 18 |
| 0.001 | 16119 | 24874 | 42384 |
| 0.01 | 1400 | 2280 | 4040 |
| 0.1 | 124 | 216 | 401 |
| 0.5 | 26 | 48 | 92 |
| 2 | 8 | 17 | 33 |
| 4 | 5 | 11 | 23 |
| 6 | 4 | 9 | 19 |

Here, we note that the same observations as made in Section 3 about $t$ can be made here. □

## 5. Implementation

The above algorithms have been coded and executed on a variety of systems such as a 16 processor IBMSP2, an 8 processor BEOWOLF cluster, and a Sun Solaris cluster grid of 34 Sun servers. The results to be shown below are for the Sun cluster grid, however, the algorithms described in this paper are machine independent and can be easily executed on any parallel machine, regardless of memory structure or communication architecture.

The two parallel algorithms utilized here are the Split & Correct method $[4\frac{n}{p} + 4t + \mathrm{O}(1) + \text{communication}]$ and the stacked method $[4\frac{n}{p} + 6t + \mathrm{O}(1)]$. The following table (for various $p$ values) shows the average run times for each of the algorithm\processor combinations. The results were obtained for the case of $d = 14$, $\alpha = -10$ and a tolerance level of $10^{-8}$. The executions presented herein used a system size of $N = 4, 324, 320$. Sequentially, we compare our algorithm to the method of Yan and Chung [10] $[4n + 2t + \mathrm{O}(1)]$ which is currently the fastest known sequential algorithm for approximating the solution to our given system. Table 1 presents the execution times for the algorithms considered here.

### 5.1. Speed-up and efficiency

Often, when validating parallel results we consider the metrics of speed-up ($S_p$) and efficiency ($E_p$) given by the standard formulas

$$S_p = \frac{T(N, 1)}{T(N, p)} \quad \text{and} \quad E_p = \frac{S_p}{p},$$

where $T(N, p)$ is the time to solve a problem of $N$ equations on $p$ processors. For the Split & Correct algorithm the theoretical speed-up equates to

$$S_p = \frac{4n + 2t + \mathrm{O}(1)}{4\frac{n}{p} + 4t + \mathrm{O}(1) + \text{communication}}$$

and assuming $N \gg t$ we can approximate this by

$$S_p \approx \frac{4n}{4\frac{n}{p} + \text{communication}}.$$

Table 1
Execution times: Stacked Algorithm and Split & Correct

| $P$ | $N = 4, 324, 320$ | | |
|---|---|---|---|
| | Stacking | Split & Correct | Increase |
| 1* | 1.60406 | | |
| 2 | 0.699201 | 0.709434 | 0.010233 |
| 3 | 0.544647 | 0.567661 | 0.023014 |
| 4 | 0.407737 | 0.410452 | 0.002715 |
| 5 | 0.326613 | 0.331708 | 0.05095 |
| 6 | 0.269354 | 0.270606 | 0.01252 |
| 7 | 0.231125 | 0.250168 | 0.01903 |
| 8 | 0.202472 | 0.213142 | 0.01067 |
| 12 | 0.135721 | 0.175783 | 0.01088 |
| 16 | 0.106286 | 0.117166 | 0.040062 |
| 20 | 0.081853 | 0.103298 | 0.01088 |
| 24 | 0.067882 | 0.093136 | 0.021445 |
| 28 | 0.058006 | 0.090508 | 0.025254 |
| 32 | 0.050695 | 0.080505 | 0.032502 |

Table 2
Speedups: Stacked Algorithm and Split & Correct

| $P$ | $N = 4,324,320$ | |
|---|---|---|
| | Stacking | Split & Correct |
| 1* | 1.60406 | |
| 2 | 2.294132875 | 2.261041901 |
| 3 | 2.945136942 | 2.825735782 |
| 4 | 3.934055531 | 3.908033095 |
| 5 | 4.911194594 | 4.835759162 |
| 6 | 5.955211358 | 5.927658662 |
| 7 | 6.94022715 | 6.411931182 |
| 8 | 7.922379391 | 7.525780935 |
| 12 | 11.81880475 | 9.125228264 |
| 16 | 15.0919218 | 13.69049042 |
| 20 | 19.59683823 | 15.52847103 |
| 24 | 23.63012286 | 17.222771 |
| 28 | 27.65334621 | 17.72285323 |
| 32 | 31.64138 | 19.92497 |

Dependent upon the architectures' communication costs, this speed-up is less than $p$. For the Stacking algorithm we get

$$S_p = \frac{4n + 2t + \mathrm{O}(1)}{4\frac{n}{p} + 6t + \mathrm{O}(1)}$$

which if $N \gg t$ is approximately $p$. Table 2 shows the results for speed-up from implementing these algorithms on our Sun cluster grid.

It is clear from the results that the new Stacking algorithm is out-performing the previously derived Split & Correct algorithm. It can also be observed that the Stacking algorithm maintains a speedup much closer to linear speed up ($S_p = p$) than does the Split & Correct algorithm which requires more communication as the number of processors increases. These results, naturally, are very much system dependent, relying on the communication protocols and architectures in place within the parallel machine.

The efficiency of the Stacking algorithm, from the results in Table 2, show a stable efficiency of approximately 98–99% with the exception of low values of $p$.

### 5.2. Scalability results

The running times presented above can easily be examined for scalability. Perfect scalability is defined as

$$\frac{T(kp, kn)}{T(p, n)} = k \quad \text{for any positive integer value of } k.$$

Here we see that the presented stacking algorithm achieves near perfect scalability; a result better than that presented by the previous Split & Correct algorithm.

### 5.3. Actual error

The algorithms in this paper all require a pre-determined acceptable level of error or tolerance ($\varepsilon$). The examples cases presented above all utilized a maximum allowable error of $10^{-8}$. The actual implementation, however, of these algorithms often produces a much smaller actual measure of error that which is requested. For the three algorithms considered here, the following table shows an example of the actual obtained largest error in the solution when $\alpha = 1$.

These errors are significantly improved over the desired error of $10^{-8}$ due to the nature of rounding in our error analysis\proof (Table 3).

Table 3
Actual obtained error

| Program | Obtained error |
|---|---|
| Yan and Chung | $4.02 \times 10^{-11}$ |
| Split & Correct | $2.70 \times 10^{-11}$ |
| Stacking | $3.68 \times 10^{-10}$ |

## 6. Conclusion

The Stacked algorithm presented here provides an efficient scalable algorithm for solving a linear system of equations whose coefficient matrix is both diagonally dominant and Toeplitz. It does so by eliminating the correction step inherent within previous works of Rojo [8], Yan and Chung [10], McNally [4], McNally et al. [6] and Nemani [7], while also removing all necessary inter-processor communication. Experimental results of this new algorithm show near optimal levels of efficiency and significant improvement in speed-up over previous algorithms.

Another key effect of this algorithm is the ability to solve for any particular value (or range of values) within a large system provided there are $t$ equations above and below the desired element(s) (or less should the element be near the first or final elements of the system). This ability to solve for a particular value to within a given level of tolerance could greatly assist in solution solving when looking for maxima or minima within a known interval of values.

Future work for this class of algorithms is to examine other Toeplitz systems whose band size is larger, and also to examine other similar systems such as block Toeplitz and $m$-Toeplitz systems. In the case of solving a pentadiagonal system, a non-linear system of equations arises which will introduce a sub-problem which needs to be solved to proceed.

## References

[1] L.E. Garey, R.E. Shaw, A parallel algorithm for solving Toeplitz linear systems, J. Appl. Math. Comput. 100 (1999) 241–245.
[2] L.E. Garey, R.E. Shaw, A parallel method for linear equations with tri-diagonal Toeplitz coefficient matrices, Internat. J. Math. Comput. 42 (2001) 1–11.
[3] Li. Kuiyuan, A fully parallel method for tri-diagonal eigenvalue problem, Internat. J. Math. Math. Sci. 17 (1994) 741–752.
[4] J.M. McNally, Fast parallel algorithms for tri-diagonal symmetric Toeplitz systems, MCS Thesis, University of New Brunswick, Saint John, 1999.
[5] J.M. McNally, A scalable communicationless parallel algorithm for tri-diagonal Toeplitz systems, Ph.D. Thesis, University of New Brunswick, Saint John, December 2002.
[6] J.M. McNally, L.E. Garey, R.E. Shaw, A split-correct parallel algorithm for solving tri-diagonal symmetric Toeplitz systems, Internat. J. Comput. Math. 75 (2000) 303–313.
[7] S.S. Nemani, Perturbation methods for circulant-banded systems and their parallel implementation, Ph.D. Thesis, University of New Brunswick, Saint John, 2001.
[8] O. Rojo, A new method for solving symmetric circulant tri-diagonal system of linear equations, Comput. Math. Appl. 20 (1990) 61–67.
[9] X.-H. Sun, Application and accuracy of the parallel diagonal dominant algorithm, Parallel Comput. (1995), 1241–1267.
[10] W.M. Yan, K.L. Chung, A fast algorithm for solving special tri-diagonal systems, Computing 52 (1994) 203–211.