Solving systems of symmetric Toeplitz tridiagonal equations: Rojo's algorithm revisited

Antonio M. Vidal[1], Pedro Alonso[1,*]

Department of Information Systems and Computation.
Universidad Politécnica de Valencia.
Camino de Vera s/n. 46022 Valencia. Spain.

Abstract

More than 20 years ago, O. Rojo published [1] an algorithm for solving linear systems where the matrix is tridiagonal symmetric Toeplitz and diagonal dominant. The technique proposed by Rojo is very efficient, $O(n)$, and has been applied successfully in the solution of other similar problems: circulant tridiagonal systems, pentadiagonal Toeplitz systems, etc. In this article we extend Rojo's algorithm to the case of non–diagonal dominant matrices, thus completing a good tool in the aforementioned applications. Other algorithms that solve the same problem are also analysed and compared with the new version of Rojo's algorithm.

Keywords: Symmetric Tridiagonal Toeplitz, Rojo's Algorithm, circulant matrices

1. Introduction

More than 20 years ago, O. Rojo published [1] an algorithm for solving linear systems where the matrix is tridiagonal symmetric Toeplitz and diagonal dominant.

The fundamental idea of the algorithm is to decompose the Toeplitz matrix into a sum of a pair of matrices, so that one of them has a simple LU–

*Corresponding author: Pedro Alonso

Email addresses: `avidal@dsic.upv.es` (Antonio M. Vidal), `palonso@dsic.upv.es` (Pedro Alonso)

[1]Supported by Spanish Government (Projects TIN2008-06570-C04 and TEC2009-13741), and Generalitat Valenciana (Project PROMETEO/2009/013).

type Toeplitzlike decomposition, which can be computed with constant cost and allows to solve the triangular systems in $O(n)$. Then, using Sherman–Morrison's formula in $O(n)$, a modification to the partial solution can be calculated that transforms it into the solution of the Toeplitz system. In the algorithm proposed by Rojo, this decomposition can be carried out only when the initial matrix is diagonally dominant.

This same idea arose some years before and appears in several references. For example, Boisvert [2] presents a collection of algorithms that solve this problem and cites references [3, 4, 5] in connection with what he called Toeplitz factorization, and he indicates that Sherman–Morrison's formula is used in this factorization.

Unfortunately, the process of calculating LU decomposition requires the Toeplitz matrix to be diagonal dominant. This limits the possible applications of the algorithm and makes it less robust in comparison to other algorithms with a similar computational cost, such as LU, LDL^T or the use of techniques based on the implementation of the Discrete Sine Transform (DST).

This algorithm has been successfully applied as computational core in the resolution of other related matrix problems: in Rojo's article it is applied to solve circulant tridiagonal systems of linear equations, but it can also be used to solve Toeplitz pentadiagonal systems, block tridiagonal systems with circulant matrices, etc. [6, 7, 8]. Such algorithms can be used in applications in many fields of engineering such as multi–channel 3D sound applications, interpolation by splines in problems of partial differential equations discretization, etc. Let's see, e.g., the solution of the following partial differential equation

$$\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t) \ , \quad \text{for } 0 < x < l \quad \text{ and } t > 0 \ ,$$

with conditions

$$u(0,t) = u(l,t) = 0 \quad \text{with } t > 0 \ , \quad \text{and } u(x,0) = f(x) \ , \text{for } 0 \le x \le l \ ,$$

which arises in heat diffusion. A finite-difference method to approximate the above *heat equation* consists of the solution of a linear equation $Tx = b$ where the system matrix T is a tridiagonal symmetric indefinite Toeplitz matrix being $(1 - 2\lambda)$ and λ the diagonal and off–diagonal entries, respectively. In this case, $\lambda = \alpha^2(k/h^2)$ where $h = l/m$ for a given $m > 0$, and k is the time step size (see [9] for more details).

The technique proposed by Rojo, is efficient ($O(n)$) and robust, so it is worth trying to extend it to the case of non–diagonal dominant matrices, without changing substantially its cost. Also noteworthy is the numerical stability of the algorithm that compares favorably with other approaches based on LU decomposition.

In this paper, we extend Rojo's algorithm to the case of non–diagonal dominant matrices, thereby supplementing a good tool, useful in the applications mentioned above. The extension is accomplished without substantially increasing the computational cost, $O(n)$, of the algorithm proposed by Rojo, while maintaining the characteristics of robustness and stability. This article also analyzes other algorithms that solve the same problem. These serve as a reference for comparison with the new version of Rojo's algorithm from the point of view of computational complexity and numerical stability. The performance of the algorithms using a Linux environment where the algorithms have been implemented in Fortran and C has also been analyzed. The implementations have been executed on an Intel Xeon based computer.

The rest of the paper is organized as follows: In Section 2 we analyze the numerical behavior of symmetric Toeplitz tridiagonal matrices, by means of the singular value decomposition. In Section 3, we recall Rojo's algorithm and we present its extension to the case when the symmetric Toeplitz tridiagonal matrices are non–diagonal dominant. In Section 4 we show two algorithms based on LDL^T decomposition and on the properties of the DST. In Section 5, we compare the numerical stability of different algorithms, and discuss algorithm performance from the point of view of their experimental complexity. Section 6 shows how the new algorithm can also be used to solve symmetric tridiagonal circulant systems in the case of non–diagonal dominant matrices. We have taken the same example suggested by Rojo in his paper. Section 7 is devoted to drawing some conclusions.

2. Analysis of the SVD of symmetric tridiagonal Toeplitz matrices

The problem we consider is the resolution of a system of linear equations of the form $Tx = b$, where $T \in \Re^{n\times n}$ is a matrix of the form

$$T = \begin{pmatrix} t_0 & t_1 & & \\ t_1 & t_0 & t_1 & \\ & t_1 & \ddots & \ddots \\ & & \ddots & \end{pmatrix}, \tag{1}$$

and $b, x \in \Re^n$.

In order to check the robustness and stability of the methods that solve this linear system of equations it seems appropriate to consider some properties of the matrix T. In particular, the two–norm, the condition number and the invertibility of the system matrix. Such properties can be known from the Singular Value Decomposition (SVD) of matrix T, more specifically through knowledge of the minimum and maximum singular values. We denote these values as σ_{min} and σ_{max}, respectively.

It can be shown that extreme singular values of T have the form:

$$\sigma_{max} = \max\left\{\left||t_0| + 2|t_1|\cos\left(\frac{j\pi}{n+1}\right)\right|, j = 1, 2, \ldots, \frac{m}{2}\right\} ,$$
$$\sigma_{min} = \min\left\{\left| t_0 \ + \ 2t_1 \ \cos\left(\frac{j\pi}{n+1}\right)\right|, j = 1, 2, \ldots, \frac{m}{2}\right\} ,$$

where $m = \begin{cases} n & \text{if } n \text{ is even} \\ n+1 & \text{if } n \text{ is odd} \end{cases}$.

Indeed, simple consideration of the matrix for the normalized Sine Transform S [10] allows to diagonalize the displacement matrix

$$F + F^T, \text{ with F} = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \end{pmatrix} . \tag{2}$$

Thus, given matrix $S = \sqrt{\frac{2}{n+1}}\,(Z_{ij})_{i,j=1}^{n}$, with $Z_{ij} = \sin\left(\frac{ij\pi}{n+1}\right)$, it can be easily verified that $SS = I$, where I is the identity matrix, and $S(F + F^T)S = W$ being $W = \text{diag}(w_1, w_2, \ldots, w_n)$, with $w_i = 2\cos\left(\frac{i\pi}{n+1}\right)$.

Therefore, if we express the system matrix as $T = t_0 I + t_1(F + F^T)$, we have

$$STS = t_0 I + t_1 S(F + F^T)S = t_0 I + t_1 W .$$

Since S is orthogonal, symmetric and STS is diagonal, $(t_0 + t_1 w_1)$, $(t_0 + t_1 w_2)$, $\ldots$, $(t_0 + t_1 w_n)$ are the eigenvalues of T and $|(t_0 + t_1 w_1)|, |(t_0 + t_1 w_2)|, \ldots, |(t_0 + t_1 w_n)|$ are the singular values of T. In particular, $\sigma_{max} = \max\{|(t_0 + t_1 w_i)|, i = 1, \ldots, n\}$ and $\sigma_{min} = \min\{|(t_0 + t_1 w_i)|, i = 1, \ldots, n\}$.

In this way, it can be stated that $\sigma_{max} = \left||t_0| + 2|t_1|\cos\left(\frac{\pi}{n+1}\right)\right|$.

The value of σ_{min} can be determined by searching the ith value that minimizes $|t_0 + t_1 w_i|$, for $i = 1, 2, \ldots, n$.

Problem $Tx = b$ can be rewritten as $(-1/t_1)Tx = (-1/t_1)b$ obtaining an equivalent linear system. For the sake of convenience we use the same notation, $Tx = b$, to refer to this last linear system where

$$T = \begin{pmatrix} \alpha & -1 & & \\ -1 & \alpha & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & \end{pmatrix} , \quad \alpha = -\frac{t_0}{t_1} \quad \text{and} \quad b_i \leftarrow -\frac{b_i}{t_1} . \tag{3}$$

Assuming matrix T is defined as in (3), the maximum SVD has the form $\sigma_{max} = \left| |\alpha| + 2\cos\left(\frac{\pi}{n+1}\right) \right|$ and $\sigma_{min} = \min\{|\alpha + 2\cos\left(\frac{i\pi}{n+1}\right)|, i = 1, 2, \ldots, n\}$. Algorithm 1 shows how to find σ_{min}.

Algorithm 1 Computation of the minimum SVD (σ_{min}) of a tridiagonal symmetric Toeplitz matrix of the form (3).

```
Require: Value $\alpha$ as the quotient $-t_0/t_1$.
Ensure: $\sigma_{min}$.
 1: if $|\alpha| \geq 2$ then
 2:     $\sigma_{min} = \left| |\alpha| - 2\cos\left(\frac{\pi}{n+1}\right) \right|$
 3: else
 4:     $j = \frac{(n+1)\arccos\left(\frac{\alpha}{2}\right)}{\pi}$
 5:     $j_1 = \lfloor j \rfloor$
 6:     $j_2 = \lceil j \rceil$
 7:     $\beta_1 = \left| |\alpha| - 2\cos\left(\frac{j_1\pi}{n+1}\right) \right|$
 8:     $\beta_2 = \left| |\alpha| - 2\cos\left(\frac{j_2\pi}{n+1}\right) \right|$
 9:     $\sigma_{min} = \min(\beta_1, \beta_2)$
10: end if
```

Extreme singular values allow to analyze some of the properties of T.

2.1. The two–norm of T

The two–norm of T can be computed as

$$\|T\|_2 = \sigma_{max} = \left| |\alpha| + 2\cos\left(\frac{\pi}{n+1}\right) \right| .$$

It is seen that $\|T\|_2$ is bounded in the form $|\alpha| \leq \|T\|_2 \leq |\alpha| + 2$, for any $n > 1$.

2.2. Singularity of T

The condition of singularity for a matrix can be expressed as $\sigma_{min} = 0$. Thus, the singularity of T can be analyzed based on the value of parameter $|\alpha|/2$.

- If $\frac{|\alpha|}{2} > 1$,
$$\sigma_{min} = 0 \Rightarrow |\alpha| = 2\cos\left(\frac{\pi}{n+1}\right) \leq 2 \; .$$
Since $|\alpha|/2 > 1 \Rightarrow |\alpha| > 2$ it is obvious that T cannot be singular.

- If $\frac{|\alpha|}{2} = 1$,
$$\sigma_{min} = 0 \Rightarrow |\alpha| = 2\cos\left(\frac{\pi}{n+1}\right) \leq 2 \; .$$
Since $|\alpha|/2 = 1 \Rightarrow |\alpha| = 2$ it is obvious that T can only be singular if $\cos\left(\frac{\pi}{n+1}\right) = 1$. This only happens if $\left(\frac{\pi}{n+1}\right) = 0$, so T is singular only for large values of n, that is, if $|\alpha| = 2$ and n is large enough.

- If $\frac{|\alpha|}{2} < 1$, then $0 < \arccos\left(\frac{|\alpha|}{2}\right) \leq \frac{\pi}{2}$. Furthermore, for T to be singular the condition $|\alpha| - 2\cos\left(\frac{j_e\pi}{n+1}\right) = 0$, where j_e is a natural number between 1 and n, must be met. Let $D(n+1) = \{d_i, i = 1, 2, \ldots, r\}$ be the set of divisors of $n+1$, then the condition $\frac{\pi}{\arccos\left(\frac{|\alpha|}{2}\right)} = d_i \in D(n+1)$ must be accomplished so that $j_e = (n+1)\frac{\arccos\left(\frac{|\alpha|}{2}\right)}{\pi} \in \{1, 2, \ldots, n\}$. Then, $\frac{\pi}{d_i} = \arccos\left(\frac{|\alpha|}{2}\right)$ and $\frac{|\alpha|}{2} = \cos\left(\frac{\pi}{d_i}\right)$; thus $|\alpha| - 2\cos\left(\frac{j_e\pi}{n+1}\right) = 0$. Hence, T is singular if and only if
$$|\alpha| = 2\cos\left(\frac{\pi}{d_i}\right), d_i \in D(n+1), \quad \text{and} \quad \arccos\left(\frac{|\alpha|}{2}\right) \in \left]0, \pi/2\right] \; .$$
For the particular case of $|\alpha| = 0$, then $2 = d_i \in D(n+1)$ if n is odd, then $0 = |\alpha| = 2\cos\left(\frac{\pi}{2}\right)$. Thus, T is singular when n is odd.

2.3. Condition number of T

The two–condition number of a given matrix A is defined as $\kappa_2(A) = \sigma_{max}/\sigma_{min}$. This number tends to infinity when matrix A is singular. Large values for κ_2 show that small errors in input data of a linear system of equations $Ax = b$ can result in large errors in the computed solution.

In the case of matrix T as defined in (3), the value of $\kappa_2(T)$ will be analyzed as a function of $|\alpha|$.

If $|\alpha| > 2$,

$$\kappa_2(T) = \frac{\sigma_{max}}{\sigma_{min}} = \frac{|\alpha| + 2\cos\left(\frac{\pi}{n+1}\right)}{|\alpha| - 2\cos\left(\frac{\pi}{n+1}\right)} \le \frac{|\alpha| + 2}{|\alpha| - 2\cos\left(\frac{\pi}{n+1}\right)} \le \frac{|\alpha| + 2}{|\alpha| - 2} ,$$

that is, there is an upper bound for $\kappa_2(T)$. This upper bound approaches 1 as $|\alpha|$ increases and can be large for values of $|\alpha|$ close to 2.

If $|\alpha| = 2$,

$$\kappa_2(T) = \frac{\sigma_{max}}{\sigma_{min}} = \frac{2 + 2\cos\left(\frac{\pi}{n+1}\right)}{2 - 2\cos\left(\frac{\pi}{n+1}\right)} = \frac{1 + \cos\left(\frac{\pi}{n+1}\right)}{1 - \cos\left(\frac{\pi}{n+1}\right)} .$$

This value can be large if n is large, since $\cos\left(\frac{\pi}{n+1}\right) \to 1$ and $\frac{1+\cos\left(\frac{\pi}{n+1}\right)}{1-\cos\left(\frac{\pi}{n+1}\right)} \to \frac{1}{0}$.

For small values of n the value of $\kappa_2(T)$ is not expected to be very large. For example, see the following pairs: $(n, \lfloor\kappa_2(T)\rfloor)$:(10,48), (50,1053), (100,4133), (500,10172), (1000,406095).

If $|\alpha| < 2$,

$$\kappa_2(T) = \frac{\sigma_{max}}{\sigma_{min}} = \frac{|\alpha| + 2\cos\left(\frac{\pi}{n+1}\right)}{|\alpha| - 2\cos\left(\frac{j_e\pi}{n+1}\right)} ,$$

where j_e must be a natural number between 1 and n.

If $\left||\alpha| + 2\cos\left(\frac{j_e\pi}{n+1}\right)\right| > K > 0$, with a large K, then $\kappa_2(T)$ need not be large. That is, if $\frac{\pi}{\arccos\left(\frac{|\alpha|}{2}\right)}$ is not close to any divisor of $n+1$, then matrix T is not ill–conditioned.

If $\frac{\pi}{\arccos\left(\frac{|\alpha|}{2}\right)} = d_i \in D(n+1)$, then $j_e = (n+1)\frac{\arccos\left(\frac{|\alpha|}{2}\right)}{\pi} \in \{1, 2, \dots, n\}$.

Hence,

$$|\alpha| - 2\cos\left(\frac{j_e\pi}{n+1}\right) \to 0 ,$$

and therefore $\kappa_2(T) = \sigma_{max}/\sigma_{min} \to \infty$. Thus, the matrix will be ill–conditioned if $|\alpha| \to 2\cos\left(\frac{\pi}{d_i}\right)$, with $d_i \in D(n+1)$ and $\arccos\left(\frac{|\alpha|}{2}\right) \in]0, \pi/2]$.

In the particular case in which $|\alpha| = 0$, we have $2 = d \in D(n+1)$ if n is odd, so $0 = |\alpha| = 2\cos\left(\frac{\pi}{2}\right)$ and $\kappa_2(T) = \sigma_{max}/\sigma_{min} \to \infty$ if n is odd.

3. Extension of Rojo's algorithm to the non–diagonal dominant case

3.1. Rojo's Algorithm for diagonal dominant matrices

First, we show the original algorithm from Rojo [1] for diagonal dominant matrices. Although in Rojo's paper the algorithm is based on use of the LU decomposition, for the sake of coherence we use the LDL^T decomposition to develop the rest of this paper. The diagonal dominant feature of matrix T in this case makes both developments equivalent.

We consider the linear system of equations $Tx = b$, that is,

$$Tx = \begin{pmatrix} \alpha & -1 & & \\ -1 & \alpha & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \end{pmatrix} = b \ , \tag{4}$$

with $|\alpha| \geq 2$. Matrix T can be decomposed into the sum $T = C + \beta e_1 e_1^T$, so that C have a LDL^T decomposition of Toeplitz tridiagonal type, that is, $T = LDL^T + \beta e_1 e_1^T$ where

$$L = \begin{pmatrix} 1 & 0 & & \\ l & 1 & 0 & \\ & l & \ddots & \ddots \\ & & \ddots & \end{pmatrix} \ , \quad D = \begin{pmatrix} d & 0 & & \\ 0 & d & 0 & \\ & 0 & \ddots & \ddots \\ & & \ddots & \end{pmatrix} \ , \text{ and } \beta = \alpha - d \ , \tag{5}$$

where $e_1 \in \Re^n$ is the first column of the identity matrix.

Values l and d can be obtained from the equality $T = LDL^T + \beta e_1 e_1^T$ and they must verify $l = \frac{-1}{d}$ and $d + \frac{1}{d} = \alpha$, that is, they must be a solution of equation $d^2 - \alpha d + 1 = 0$. Hence, $d = \frac{\alpha}{2} \pm \sqrt{\left(\frac{\alpha}{2}\right)^2 - 1}$, and d can be chosen so that $|d| > 1 : d = \frac{\alpha}{2} + \text{sign}(\alpha)\sqrt{\left(\frac{\alpha}{2}\right)^2 - 1}$.

We assume the solution of $Tx = b$ can be expressed as a sum of vectors of unknowns $x = z + \eta$ where z is chosen in such a way that $LDL^T z = b$. Thus, the system $Tx = b$ can be expressed as $(LDL^T + \beta e_1 e_1^T)(z + \eta) = b$, and using the hypothesis $LDL^T z = b$ we obtain $(LDL^T + \beta e_1 e_1^T)\eta = -\beta(e_1^T z)e_1$, that is, $T\eta = -\beta z_1 e_1$.

To obtain the solution of $T\eta = -\beta z_1 e_1$, Rojo proposes to use of the Sherman–Morrison formula [11] that allows to express the inverse of the sum of two matrices, $(A + UV^T)$, of the form,

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1} \ , \tag{6}$$

with $A \in \Re^{n\times n}$, $U, V \in \Re^{n\times r}$, and $I \in \Re^{r\times r}$ the identity matrix.

Matrix $T = LDL^T + \beta e_1 e_1^T$ in the form (6) can be used to calculate the inverse of T^{-1},

$$T^{-1} = (LDL^T + \beta e_1 e_1^T)^{-1} =$$
$$(LDL^T)^{-1} - \beta(LDL^T)^{-1} e_1 (1 + \beta e_1^T (LDL^T)^{-1} e_1)^{-1} e_1^T (LDL^T)^{-1} \ ,$$

and thus η can be calculated as $\eta = -\beta z_1 T^{-1} e_1$, hence,

$$\eta = -\beta z_1 ((LDL^T)^{-1} e_1 - \beta (LDL^T)^{-1} e_1 (1 + \beta e_1^T (LDL^T)^{-1} e_1)^{-1} e_1^T (LDL^T)^{-1} e_1) \ .$$

If we call $u = (LDL^T)^{-1} e_1$, then we have,

$$\begin{aligned} \eta &= -\beta z_1 (u - \beta u (1 + \beta e_1^T u)^{-1} e_1^T u) \\ &= -\beta z_1 (u - \beta u (1 + \beta u_1)^{-1} u_1) \\ &= -\beta z_1 (1 - \beta (1 + \beta u_1)^{-1} u_1) u \ . \end{aligned}$$

Thus,

$$\eta = -\beta z_1 \left(\frac{1}{1 + \beta u_1} \right) u \ .$$

Computation of the solution vector $x = z + \eta$ involves the following steps:

1. Compute l and d that provide L and D.
2. Solve the system $LDL^T z = b$.
3. Solve the system $LDL^T u = e_1$.
4. Compute vector $\eta = -\beta z_1 \left(\frac{1}{1+\beta u_1} \right) u$.
5. Sum both vectors to have $x = z + \eta$.

Step 1 can be computed in $O(1)$ flops. Steps 2 and 3 are recurrences that can be carried out in $O(n)$ flops. Also, steps 4 and 5 have a cost of $O(n)$ flops. Thus, an algorithm (Algorithm 2) can be generated to solve system $Tx = b$ in $O(n)$ flops.

The special case of $\alpha = 2$ can be solved with the same operations and the following values: $d = 1$, $\beta = 1$ and $u_1 = n$.

The cost of Algorithm 2 is $9n+6$ flops, where the flop is defined as in [11].

Algorithm 2 (**Rojo's Algorithm**): Solution of linear system $Tx = b$ (4).

```
Require: $\alpha$ so $|\alpha| \geq 2$, $n > 1, b \in \Re^n$ and $T = \alpha I - (F + F^T)$.
Ensure: Solution $x$ of the system $Tx = b$ in the form of (4).
  1. $d = \frac{\alpha}{2} + \text{sign}(\alpha)\sqrt{\left(\frac{\alpha}{2}\right)^2 - 1}$.
  2. Solve system $LDL^T z = b$:
    2.1. Solve $Lw = b$:
      $w_1 = b_1$.
      for $i = 2 : n$ do
        $w_i = b_i + \frac{w_{i-1}}{d}$.
      end for
    2.2. Solve $L^T w = D^{-1} w$:
      $z_n = w_n / d$.
      for $i = n - 1 : -1 : 1$ do
        $z_i = \frac{w_i + z_{i+1}}{d}$.
      end for
  3. Solve system $LDL^T u = e_1$:
    3.1. Solve $Lv = e_1$:
      $v_1 = 1$.
      for $i = 2 : n$ do
        $v_i = \frac{v_{i-1}}{d}$.
      end for
    3.2. Solve $L^T u = D^{-1} v$:
      $u_n = v_n / d$.
      for $i = n - 1 : -1 : 1$ do
        $u_i = \frac{v_i + u_{i+1}}{d}$.
      end for
  4. Compute $\eta = -\beta z_1 \left(\frac{1}{1+\beta u_1}\right) u$.
  5. Compute $x = z + \eta$.
```

3.2. Yan and Chun variants for diagonal dominant matrices

Step 3 of Algorithm 2 consists of successive divisions of a number, initially set to 1, by number d, whose modulus is greater than 1. When the iteration from 1 to n reaches a certain value t, the quotient $v_i = \frac{v_{i-1}}{d}$ is almost 0. This allows to build an approximated algorithm in such a way that the number of iterations in recurrences 3.1 and 3.2 in Algorithm 2 can be limited to t. The value of t depends on $|\alpha|$ and the precision. For example, working in double precision in Matlab, if $|\alpha| = 3$ then $v_{36} = 0.00000000000000$, that is $t = 36$; if $|\alpha| = 6$ then $v_{20} = 0.00000000000000$, that is $t = 20$. A suitable value for t can be obtained in the form $t = \left|\frac{\log_{10}(bound)}{\log_{10}(d)}\right|$, where *bound* is the minimum floating point number capable of representation. This approximation results, for example, for $bound = 10^{-14}$ in the following pairs: (α, t): (2.1,103), (3,34), (6,19), ...

With these ideas, step 3 of Rojo's Algorithm can be modified as follows:

```
$bound = 10^{-14}$ (or the desired value of the precision)
$t = \lceil -\log_{10}(bound) / \log_{10}(t) \rceil$.
3. Solve system $LDL^T u = e_1$:
  3.1. Solve system $Lv = e_1$:
    $v_1 = 1$
    for $i = 2 : t$ do
      $v_i = \frac{v_{i-1}}{d}$.
    end for
  3.2. Solve $L^T u = D^{-1} v$:
    $u_{t+1:n} = 0$.
    for $i = t - 1 : -1 : 1$ do
      $u_i = \frac{v_i + u_{i+1}}{d}$.
    end for
```

The cost of Rojo's Algorithm with this variant is $5n + 4t + 10$ flops.

This approximation was presented in [12]. In that work, step 3 of the algorithm is based on this approach, being possible to build a vector p with the last $n - t$ components equal to 0 so that $Tp = e_1$ and the solution can be approximated as $x = z + \gamma p$.

3.3. Rojo's Algorithm for non–diagonal dominant matrices

In the following, we consider the general case in which $\alpha \in \Re$ has no bound on its absolute value, that is, without imposing the condition of being diagonal dominant to matrix T.

Obviously, the case $|\alpha| \geq 2$ has already been studied, but it may also happen that $|\alpha| < 2$. This case may be dealt with as follows. Since the linear system $Tx = b$ is real defined, it has a real solution if T is invertible. The use of Rojo's Algorithm involves working with complex arithmetic that, after carrying out the operations, leads to a real solution in which the imaginary part will be null after such operations. Therefore, an algorithm can be implemented that involves only the use of real arithmetic with only searching for the real component of the solution, by splitting the real and imaginary parts in the successive operations that arise in the process.

Note that matrix T can be decomposed as before in the sum $T = C + \beta e_1 e_1^T$ so that C has a tridiagonal Toeplitz type LDL^T decomposition, that is, $T = LDL^T + \beta e_1 e_1^T$ with the L and D factors defined in (5).

As in the former case, values l and d can be obtained from the equality $T = LDL^T + \beta e_1 e_1^T$ and they must verify $l = \frac{-1}{d}$ and $d + \frac{1}{d} = \alpha$, that is, they must be a solution of equation $d^2 - \alpha d + 1 = 0$. Hence, $d = \frac{\alpha}{2} \pm \sqrt{\left(\frac{\alpha}{2}\right)^2 - 1}$. But now, $\left(\frac{\alpha}{2}\right)^2 - 1 < 0$, so the solution will be a complex number of the form:

$$d = \frac{\alpha}{2} \pm \sqrt{1 - \left(\frac{\alpha}{2}\right)^2}\, i \ , \text{ with } i = \sqrt{-1}\ .$$

Let us choose d as $d = \frac{\alpha}{2} + \operatorname{sign}(\alpha)\sqrt{1 - \left(\frac{\alpha}{2}\right)^2}\, i$. Given the form of d, it is accomplished that $|d| = 1$.

We denote $c = \frac{\alpha}{2}$ and $s = \operatorname{sign}(\alpha)\sqrt{1 - \left(\frac{\alpha}{2}\right)^2}$. Thus $d = c + si$ and $d^* = c - si$, where * denotes the complex conjugated.

Since l and d are now complex numbers, the components of matrices L and D are complex too and also the scalar $\beta = \alpha - d$. Hence, to tackle the linear systems $LDL^T z = b$ and $LDL^T u = e_1$, we must take into account that z and u will also be vectors with complex components. In the same way, the scalar $-\beta z_1 \left(\frac{1}{1+\beta u_1}\right)$ will be a complex number and therefore vector η will have complex components. However, the solution vector computed as $x = z + \eta$ will have all its components real, that is, $x = \Re(x) = \Re(z) + \Re(\eta)$ (in the sequel we denote $\Re(a)$ and $\Im(a)$ as the real and imaginary part, respectively, of a scalar or a complex array).

The operations planned for the real case ($|\alpha| \geq 2$) can be modified now in a simple way to be affordable by using real arithmetic, although the components of vectors and matrices will be complex.

Thus, the system $LDL^T z = b$ can be solved using the same recurrences used in the previous section but ensuring that $z = \Re(z) + \Im(z)i$. If we deal with the system $Lw = b$, with $w = \Re(w) + \Im(w)i$, the recurrences that produce the solution are:

```
$w_1 = b_1$
**for** $i = 2 : n$ **do**
```

```
    $w_i = b_i + w_{i-1}d^*$
  end for
```

and

```
  $z_n = w_n d^*$
  for $i = n-1 : -1 : 1$ do
    $z_i = (w_i + z_{i+1})d^*$
  end for
```

The corresponding recurrences by separating the real and imaginary parts have the form

```
  $\Re(w_1) = b_1$, $\Im(w_1) = 0$.
  for $i = 2 : n$ do
    $\Re(w_i) = b_i + \Re(w_{i-1})c - \Im(w_{i-1})(-s)$
    $\Im(w_i) = \Re(w_{i-1})(-s) + \Im(w_{i-1})c$
  end for
```

and

```
  $\Re(z_n) = \Re(w_n)c - \Im(w_n)(-s)$
  $\Im(z_n) = \Im(w_n)c + \Re(w_n)(-s)$
  for $i = n-1 : -1 : 1$ do
    $\Re(z_i) = (\Re(w_i) + \Re(z_{i+1}))c - (\Im(w_i) + \Im(z_{i+1}))(-s)$
    $\Im(z_i) = (\Im(w_i) + \Im(z_{i+1}))c + (\Re(w_i) + \Re(z_{i+1}))(-s)$
  end for
```

In the same way, the solution to system $LDL^T u = e_1$ can be obtained by solving first $Lv = e_1$ and later $L^T u = D^{-1}v$.

The following recurrences solve these systems taking into account that $u = \Re(u) + \Im(u)i$ and $v = \Re(v) + \Im(v)i$.

Recurrence for v:

```
  $\Re(v_1) = I$, $\Im(v_1) = 0$.
  for $i = 2 : n$ do
    $\Re(v_i) = \Re(v_{i-1})c - \Im(v_{i-1})(-s)$
    $\Im(v_i) = \Re(v_{i-1})(-s) + \Im(v_{i-1})c$
  end for
```

Recurrence for u:

```
  $\Re(u_n) = \Re(v_n)c - \Im(v_n)(-s)$
  $\Im(u_n) = \Im(v_n)c + \Re(v_n)(-s)$
  for $i = n-1 : -1 : 1$ do
    $\Re(u_i) = (\Re(u_i) + \Re(u_{i+1}))c - (\Im(v_i) + \Im(u_{i+1}))(-s)$
    $\Im(u_i) = (\Im(u_i) + \Im(u_{i+1}))c + (\Re(v_i) + \Re(u_{i+1}))(-s)$
end for
```

Now it is necessary to evaluate the constant $\chi = -\beta z_1 \left(\frac{1}{1+\beta u_1}\right)$ to compute

$$\eta \;=\; -\beta z_1 \left(\frac{1}{1+\beta u_1}\right) u \; .$$

As

$$\chi \;=\; -\beta z_1 \left(\frac{1}{1+\beta u_1}\right) = \frac{-z_1}{\frac{1}{\beta} + u_1} = \frac{-z_1}{\beta^* + u_1} \; ,$$

if we denote $\beta_u = |\beta^* + u_1|^2$, then we have $\chi = \left(\frac{-1}{\beta_u}\right) z_1(\beta^* + u_1)^*$ and, thus,

$$\Re(\chi) = \left(\frac{-1}{\beta_u}\right) (\Re(z_1)\Re(\beta + u_1^*) - \Im(z_1)\Im(\beta + u_1^*))$$

and

$$\Im(\chi) = \left(\frac{-1}{\beta_u}\right) (\Im(z_1)\Re(\beta + u_1^*) + \Re(z_1)\Im(\beta + u_1^*))$$

with $\Re(\beta + u_1^*) = (\alpha - c) + \Re(u_1)$ and $\Im(\beta + u_1^*) = -s - \Im(u_1)$.

Hence, $\eta = \chi u$ with $\Re(\eta) = \Re(\chi)\Re(u) - \Im(\chi)\Im(u)$. The solution vector is expressed now as $x = \Re(z) + \Re(\eta)$.

The algorithm for non–diagonal dominant matrices can be expressed as shown in Algorithm 3.

The cost of Algorithm 3 is $36n + 20$ flops.

3.4. The special cases $\alpha = 1$ and $\alpha = 0$

The algorithm works fine when matrix T is invertible, that is, in the case $\alpha = 1$ it is required that $n \in \{3k, 3k+1; k \in \mathcal{Z}^+\}$, and in the case $\alpha = 0$ it is required that n be even.

4. Other algorithms of reference

In this section we present two algorithms which can be used to solve the linear system (1). These algorithms also exploit the special structure of the symmetric tridiagonal Toeplitz matrix. The first algorithm is based on the LDL^T factorization of the system matrix (LDL^T method) and the second one is based on Discrete Sine Transformation (DST method).

Algorithm 3 (**Modified Rojo's Algorithm**): Solution of linear system $Tx = b$ (4).

```
Require: $\alpha \in \Re$, $n > 1$, $b \in \Re^n$ and $T = \alpha I - (F + F^T)$.
Ensure: Solution $x$ of the system $Tx = b$ in the form of (4).
  1. Compute the real and imaginary parts of $l$ and $d$ in order to compute $L$ and $D$:
        $c = \frac{\alpha}{2}\ , s = \ \mathrm{sign}\,(\alpha)\sqrt{1-\left(\frac{\alpha}{2}\right)^2}\ .$
  2. Solve system $LDL^T[\ z \quad u\ ] = [\ b \quad e_1\ ]$:
    2.1. $L[\ w \quad v\ ] = [\ b \quad e_1\ ]$:
      $rw_1 = b_1$, $iw_1 = 0$.
      $rv_1 = 1$, $iv_1 = 0$.
      for $i = 2 : n$ do
        $rw_i = b_i + rw_{i-1} * c + iw_{i-1} * s$
        $iw_i = -rw_{i-1} * s + iw_{i-1} * c$
        $rv_i = rv_{i-1} * c + iv_{i-1} * s$
        $iv_i = -rv_{i-1} * s + iv_{i-1} * c$
      end for
    2.2. $L^T[\ z \quad u\ ] = D^{-1}[\ w \quad v\ ]$:
      $rz_n = rw_n * c + iw_n * s$
      $iz_n = iw_n * c - rw_n * s$
      $ru_n = rv_n * c + iv_n * s$
      $iu_n = iv_n * c - rv_n * s$
      for $i = n - 1 : -1 : 1$ do
        $rz_i = (rw_i + rz_{i+1}) * c + (iw_i + iz_{i+1}) * s$
        $iz_i = (iw_i + iz_{i+1}) * c - (rw_i + rz_{i+1}) * s$
        $ru_i = (ru_i + ru_{i+1}) * c + (iv_i + iu_{i+1}) * s$
        $iu_i = (iu_i + iu_{i+1}) * c - (rv_i + ru_{i+1}) * s$
      end for
  3. Compute real part of $\eta = -\beta z_1 \left(\frac{1}{1+\beta u_1}\right) u = \chi u$.
    3.1. Compute $\chi$:
      $r\delta = (\alpha - c) + ru_1$
      $i\delta = -s - iu_1$
      $\beta_u = r\delta^2 + i\delta^2$
      $r\chi = \left(\frac{-1}{\beta_u}\right)(rz_1 * r\delta - iz_1 * i\delta)$
      $i\chi = \left(\frac{-1}{\beta_u}\right)(iz_1 * r\delta + rz_1 * i\delta)$
    3.2. Compute real part of $\eta$:
      $\eta = r\chi * ru - i\chi * iu$
  4. Obtain the solution $x = rz + \eta$.
```

4.1. The LDL^T method

This method consists of the factorization of the Toeplitz matrix $T = LDL^T$, where L is unit lower triangular and D is diagonal, to solve afterwards the associated triangular systems. The factorization algorithm lets to exploit the tridiagonality of the Toeplitz matrix so the lower triangular matrix L is real bidiagonal. Since all the elements of the diagonal of L are 1, only the subdiagonal must be stored. Let d be the array where the diagonal of D is stored and l the array where the subdiagonal of L is stored, then the following recurrence allows to obtain the LDL^T factorization of T.

```
$d_1 = t_0$
**for** $i = 1 : n-1$ **do**
  $l_i = \frac{t_1}{d_1}$
  $d_{i+1} = t_0 - t_1 l_i$
**end for**
```

The following two recurrences that solve the two associated triangular linear systems, $Ly = b$:

```
$y_1 = b_1$
**for** $i = 2 : n$ **do**
  $y_i = b_i - l_{i-1} y_{i-1}$
**end for**
```

and $L^T x = D^{-1} y$:

```
$x_n = y_n / d_n$
**for** $i = n-1 : -1 : 1$ **do**
  $x_i = y_i / d_{i-l} - l_i x_{i+1}$
**end for**
```

allow to obtain the solution of (1).

An important note about this method is that it can only be used when the leading principal submatrices of T are non–singular [11].

The cost of this algorithm is $8n - 7$ flops.

4.2. The DST method

This method is based on the fact that a symmetric tridiagonal Toeplitz matrix T is diagonalizable by the matrix that defines the Discrete Sine Transform (DST). Let S be the matrix that defines the DST [10], then, the columns of S are the eigenvectors of T. Hence, the linear system $Tx = b$ can be translated to an equivalent linear system as follows,

$$STSSx = Sb \rightarrow Dy = z \ ,$$

where D is a diagonal matrix. The diagonal elements of D are the eigenvalues of T. Recall from Section 2 that S is symmetric and orthogonal.

Matrix D can be obtained as follows,

$$STS = S(t_0 + t_1(F + F^T))S = t_0 I + t_1 S(F + F^T)S = t_0 I + t_1(SFS + SF^T S) \ ,$$

where I the identity matrix and F the one position down shift matrix as defined in (2).

Matrix $(SFS + SF^T S)$ is diagonal. Assuming n is even, it can be shown that the diagonal of SFS is an array of the form $\begin{pmatrix} s \\ -\bar{s} \end{pmatrix}$, where $\bar{s}$ is vector s in the reverse order. Vector s can be constructed as follows,

```
for i=1:n/2 do
    $s_i = S_{2i,1}/S_{i,1}$
end for
```

Algorithm 4 shows the process of computing the solution of the linear system. The algorithm is valid for both even or odd problem sizes.

To calculate the cost of the algorithm it must be noted that computation of the DST of an array is an FFT related operation that can be applied rapidly [10], that is, there are fast algorithms that enable this computation in $O(n \log_2 n)$ in the best case. The best case is for a size n such that the quantity $n+1$ is a power of 2. So, the cost of the algorithm in the best case is $5 + 3n\log_2(n) + \frac{9}{2}n$ flops. In the worst case, that is, $n+1$ is a prime number, the algorithm might be of $O(n^2)$.

5. Experimental study: analysis of algorithm precision and performance

In order to carry out this study we implemented a Linux application that allows to solve system (1) by means of the three described methods. The results have been obtained in a workstation Fujitsu Siemens Celsius R650 with a Intel Quad-Core Xeon E5430 processor at 2.66 GHz.

We study two error types, forward and backward error. To evaluate the forward error we compute

$$\frac{\|x - \hat{x}\|_2}{\|x\|_2} \ ,$$

Algorithm 4 (**DST method**) Computation of the linear system $Tx = b$

```
Require: $t_0, t_1 \in \Re$ and $b \in \Re^n$.
Ensure: Solution $x$ of the system $Tx = b$.
 1: $m = \lfloor \frac{n}{2} \rfloor$
 2: $k = \frac{1}{2(n+1)}$
 3: $s = Se_1$
 4: for $i = 1 : m$ do
 5:     $v_i = \frac{s_{2i}}{s_i}$
 6: end for
 7: $y = Sb$
 8: for $i$=1:$m$ do
 9:     $x_i = \frac{ky_i}{(t_0+t_1v_i))}$
10: end for
11: if mod$(n, 2)$=1 then
12:     $x_{m+1} = \frac{kx_{m+1}}{t_0}$
13:     $m = m + 1$
14: end if
15: for $i = m + 1 : n$ do
16:     $x_i = \frac{ky_i}{(t_0+t_1v_{n/2-i+(m+1)}))}$
17: end for
18: $x \leftarrow Sx$.
```

where $x = e_1$ is the real solution and $\hat{x}$ is the computed solution. Backward error is assessed by computing

$$\frac{\|T\hat{x} - b\|_2}{\|T\|_2\|\hat{x}\|_2} .$$

We use σ_{max} as $\|T\|_2$.

The study is divided into different cases. In the first one (Table 1) we deal with the diagonal dominant case. Results in Table 1 have been obtained with $t_0 = 3.0$ and $t_1 = 1.0$. These results are representative for the entire spectrum of diagonal dominant matrices. For this range of matrices, the accuracy of the solution obtained with the three methods is similar. It is slightly better in the case of the LDL^T method for the forward error. The time obtained is also better for the same method as it is foreseen by the theoretical cost analysis. The DST method is somewhat lower in precision and clearly inferior in performance. We used the Intel MKL package (Version

10.1) to compute the DST. This package does not include a routine for the direct computation of the DST, so we used the routine to compute the FFT to implement the DST [10]. The routine for the application of the FFT included in this package is very efficient in the target machine with regard to efficiency and precision. The DST method is more efficient the smaller the largest prime number of the prime numbers into which it is decomposed $n+1$. For the size used in the table it is $n+1 = 853 \times 3517$. However, we obtained a time of 1.5 sec. with a more favorable problem size, $n = 2999999$ so that $n+1 = 3000000 = 2^6 \times 3 \times 5^6$, a time still larger than the one obtained with the other methods.

Table 1: Diagonal dominant case. $n = 3000000$, $\sigma_{max} = 5.0$, $\sigma_{min} = 1.0$.

Method	Time (sec.)	Forward error	Backward error
Modified Rojo	0.22	4.42×10^{-17}	6.25×10^{-17}
DST	4.82	1.24×10^{-15}	2.83×10^{-13}
LDL^T	0.12	2.03×10^{-19}	6.07×10^{-17}

The following case (Table 2) is a special case where the matrix is diagonal dominant but it is in the border with the non–diagonal case, that is, $t_0 = 2.0$ and $t_1 = 1.0$. In addition, the matrix is close to singularity since $\kappa_2 = 3.64 \times 10^{12}$.

Table 2: Diagonal dominant case. $n = 3000000$, $\sigma_{max} = 4.0$, $\sigma_{min} = 1.1 \times 10^{-12}$.

Method	Time (sec.)	Forward error	Backward error
Modified Rojo	0.18	0.00	1.71×10^{-16}
DST	4.80	4.91×10^{-8}	2.06×10^{-11}
LDL^T	0.13	6.23×10^{-7}	3.01×10^{-17}

The next study shows the non–diagonal dominant case (Table 3) where $t_0 = 1.5$ and $t_1 = 1.0$. The DST and LDL^T methods remain unchanged in time. The Modified Rojo method is more expensive dealing with the non–diagonal dominant case as foreseen in the theoretical cost analysis. The precision of the three methods is similar, and has even been improved in the

Table 3: Non–diagonal dominant case. $n = 3000000$, $\sigma_{max} = 3.5$, $\sigma_{min} = 5.56 \times 10^{-7}$.

Method	Time (sec.)	Forward error	Backward error
Modified Rojo	0.31	6.60×10^{-10}	6.06×10^{-17}
DST	4.80	6.15×10^{-13}	4.47×10^{-16}
LDL^T	0.14	1.59×10^{-10}	3.77×10^{-14}

case of the DST method. The results show that, in general, the forward error obtained with the DST method is lower.

Table 4 shows a special case for non–diagonal dominant matrices ($t_0 = t_1 = 1.0$). The Modified Rojo method is backward stable in all cases even for singular matrices ($n = 2999999$). The DST method is stable for non–singular matrices resulting in a good accuracy of the results. The drawback of the DST method is that it relies on the disparity of computation time. The LDL^T is useless for this case since its effectiveness depends on the non–singularity of the leading principal submatrices. In this example the LDL^T method fails due to the existing singular principal minors. Pivoting techniques need to be included in the method, but that would make the method lose its competitiveness in execution time.

Table 4: Non–diagonal dominant case, $\sigma_{max} = 3.0$ (time in sec.).

n	Method	Time	Forward err.	Backward err.	Min. SVD
	M. Rojo	0.30	1.50×10^{-12}	5.42×10^{-17}	
2999998	DST	4.81	1.05×10^{-12}	2.94×10^{-16}	6.05×10^{-07}
	LDL^T	0.12	nan	nan	
	M. Rojo	0.31	8.13×10^{1}	3.76×10^{-17}	
2999999	DST	1.51	nan	nan	2.22×10^{-16}
	LDL^T	0.11	nan	nan	
	M. Rojo	0.30	1.57×10^{-12}	6.01×10^{-17}	
3000000	DST	4.81	9.53×10^{-13}	2.70×10^{-16}	6.05×10^{-07}
	LDL^T	0.11	nan	nan	

A similar behavior has been obtained in the case $t_0 = 0.0$ and $t_1 = 1.0$. The LDL^T method fails for the same reason. The precision obtained with the DST method is acceptable. The Modified Rojo method offers the exact solution.

6. Applications: Solution of a circulant system

We now apply the Modified Rojo's Algorithm to solve a special circulant system $Mx = f$.

Let M be a matrix that can be expressed as $M = M_1 + a(e_n e_1^T + e_1 e_n^T)$, with $M_1 = cI + a(F + F^T)$, where F is the one position down shift matrix defined in (2), and e_1 and e_n are the first and last columns of the identity matrix of order n, respectively. It should be noted that matrix M does not need to be diagonal dominant. We only set the obvious condition $a \neq 0$.

If we take $\lambda = \frac{c}{-a}$, the equation system $Mx = f$ should be equivalent to the equation system $Ax = b$, with $A = \lambda I - (F + F^T) - (e_n e_1^T + e_1 e_n^T)$ and $b = -f/a$. Thus, we propose the solution of the system $Ax = b$.

6.1. Solution using the Sherman–Morrison formula with rank–one modifications

Let us consider the following partition of matrix A:

$$A = \begin{pmatrix} T & -e \\ -e^T & \lambda \end{pmatrix} ,$$

where $e = e_1 + e_{n-1}$, being now e_1 and e_{n-1} the first and last columns of the identity matrix of order $n-1$, respectively, and

$$T = \begin{pmatrix} \lambda & -1 & & \\ -1 & \lambda & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & \end{pmatrix} \in \Re^{(n-1)\times(n-1)} . \tag{7}$$

System $Ax = b$ can be written as

$$\begin{pmatrix} T & -e \\ -e^T & \lambda \end{pmatrix} x = b \leftrightarrow \begin{cases} Tx_{1:n-1} - x_n e = b_{1:n-1} \\ -e^T x_{1:n-1} + \lambda x_n = b_n \end{cases} .$$

Thus,

$$x_n = \lambda^{-1}(b_n + e^T x_{1:n-1}) ,$$

and

$$Tx_{1:n-1} - \lambda^{-1}(b_n + e^T x_{1:n-1})e = b_{1:n-1} \ .$$

Therefore, there is a linear system with $n-1$ unknowns that, once known, allow to calculate also x_n.

Regrouping terms gives

$$(T - \lambda^{-1}ee^T)x_{1:n-1} = b_{1:n-1} + \lambda^{-1}b_n e \ . \tag{8}$$

Using again the Sherman–Morrison formula (6) we obtain

$$(T - \lambda^{-1}ee^T)^{-1} =$$
$$T^{-1} + T^{-1}(\lambda^{-1}e)(1 - e^T T^{-1}(\lambda^{-1}e))^{-1}e^T T^{-1} \ .$$

Defining $g = b_{1:n-1} + \lambda^{-1}b_n e$ the solution of (8) can be expressed as

$$x_{1:n-1} = T^{-1}g + T^{-1}(\lambda^{-1}e)(1 - e^T T^{-1}(\lambda^{-1}e))^{-1}e^T T^{-1}g \ .$$

If we take $z = T^{-1}g$ and $w = T^{-1}e$,

$$x_{1:n-1} = z + \lambda^{-1}w(1 - \lambda^{-1}e^T w)^{-1}e^T z \ .$$

After some changes,

$$x_{1:n-1} = z + \lambda^{-1}(1 - \lambda^{-1}(w_1 + w_{n-1}))^{-1}(z_1 + z_{n-1})w \ .$$

The system can be simplified as $x_{1:n-1} = z + \chi w$, with $\chi = \lambda^{-1}(1 - \lambda^{-1}(w_1 + w_{n-1}))^{-1}(z_1 + z_{n-1})$ and $x_n = \lambda^{-1}(b + (x_1 + x_{n-1}))$.

Algorithm 5 shows how to perform the solution of $Ax = b$.

Steps 2 and 3 of Algorithm 5 can be tackled by means of the Modified Rojo's Algorithm even in the case of matrix A being non– diagonal dominant. The cost of Algorithm 5 can be evaluated as $74n + 53$ flops.

6.2. Solution by means of Sherman–Morrison formula with rank–two modifications

The original approach in [1] utilized the Serman–Morrison formula with rank–two modifications. For the sake of completeness, we show here how to solve system $Ax = b$ with this approach in the general case where A can be non–diagonal dominant.

Algorithm 5 Solution of the system $Ax = b$ for a Circulant matrix by the Modified Rojo's Algorithm.

```
Require: $\lambda \in \Re$ and $b \in \Re^n$.
Ensure: The solution vector $x$ of $Ax = b$.
1: Compute $g = b_{1:n-1} + \lambda^{-1} b_n e$, with $e_1, e_{n-1} \in \Re^{n-1}$ and $e = e_1 + e_{n-1}$.
2: Solve system $Tz = g$.
3: Solve system $Tw = e$.
4: Compute $\chi = \lambda^{-1}(1 - \lambda^{-1}(w_1 + w_{n-1}))^{-1}(z_1 + z_{n-1})$.
5: Compute $x_{1:n-1} = z + \chi w$.
6: Compute $x_n = \lambda^{-1}(b + (x_1 + x_{n-1}))$.
```

Let us consider again system $Ax = b$ and let $\mu \in \Re$ verifying $\mu - \lambda = \frac{1}{\mu}$, that is,

$$\mu = \begin{cases} \frac{\lambda}{2} + \operatorname{sign}(\lambda) \sqrt{\left(\frac{\lambda}{2}\right)^2 - 1}\,, & \text{if } |\lambda| \geq 2 \\ \frac{\lambda}{2} + \operatorname{sign}(\lambda) \sqrt{1 - \left(\frac{\lambda}{2}\right)^2} i\,, & \text{if } |\lambda| < 2 \end{cases} .$$

Matrix T (7) can be decomposed as $T = C + \beta e_1 e_1^T$, with $C = LDL^T$ and $\beta = \lambda - \mu$ and the former system can be expressed as

$$\left(C + \beta e_1 e_1^T - \lambda^{-1} e e^T\right) x_{1:n-1} = b_{1:n-1} + \lambda^{-1} b_n e \,,$$

being $e_1, e_{n-1} \in \Re^{n-1}$ and $e = e_1 + e_{n-1}$. Alternatively, the system can be expressed as

$$\left(C + \beta e_1 e_1^T - \lambda^{-1} e e^T\right) x_{1:n-1} = g \,,$$

that is,

$$x_{n-1} = \left(C + \beta e_1 e_1^T - \lambda^{-1} e e^T\right)^{-1} g \,,$$

similarly to Section 6.1.

Note that

$$\begin{aligned} H &= \left(C + \beta e_1 e_1^T - \lambda^{-1} e e^T\right) = \\ & C + \begin{bmatrix} \beta e_1 & -\lambda^{-1} e \end{bmatrix} \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} . \end{aligned}$$

Using again the Sherman–Morrison formula (6),

$$\begin{aligned} H^{-1} &= C^{-1} - C^{-1} \begin{bmatrix} \beta e_1 & -\lambda^{-1} e \end{bmatrix} \\ & \left(I_2 + \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} C^{-1} \begin{bmatrix} \beta e_1 & -\lambda^{-1} e \end{bmatrix}\right)^{-1} \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} C^{-1} \,, \end{aligned}$$

so $x_{1:n-1} = H^{-1}g$ can be expressed as a linear combination of the solution of the following three systems

$$\begin{aligned} z = C^{-1}g &\leftrightarrow LDL^T z = g \\ u = C^{-1}e_1 &\leftrightarrow LDL^T u = e_1 \\ w = C^{-1}e &\leftrightarrow LDL^T w = e \ , \end{aligned}$$

that is,

$$x_{1:n-1} = z - \begin{bmatrix} \beta u & -\lambda^{-1}w \end{bmatrix} \left(I_2 + \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} \begin{bmatrix} \beta u & -\lambda^{-1}w \end{bmatrix} \right)^{-1} \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} z \ ;$$

since

$$\left(I_2 + \begin{bmatrix} e_1^T \\ e^T \end{bmatrix} \begin{bmatrix} \beta u & -\lambda^{-1}w \end{bmatrix} \right)^{-1} = \begin{bmatrix} 1+\beta u_1 & -\lambda^{-1}w_1 \\ \beta(u_1+u_{n-1}) & 1-\lambda^{-1}(w_1+w_{n-1}) \end{bmatrix}^{-1} ,$$

and

$$\begin{bmatrix} e_1^T \\ e^T \end{bmatrix} z = \begin{bmatrix} z_1 \\ z_1 + z_{n-1} \end{bmatrix} ,$$

calling

$$\begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} = \begin{bmatrix} 1+\beta u_1 & -\lambda^{-1}w_1 \\ \beta(u_1+u_{n-1}) & 1-\lambda^{-1}(w_1+w_{n-1}) \end{bmatrix}^{-1} \begin{bmatrix} z_1 \\ z_1 + z_{n-1} \end{bmatrix} ,$$

we have

$$\begin{aligned} x_{1:n-1} &= z - \begin{bmatrix} \beta u & -\lambda^{-1}w \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} = z - \varphi_1 \beta u + \varphi_2 \lambda^{-1} w \\ x_n &= \lambda^{-1}(b_n + (x_1 + x_{n-1})) \ . \end{aligned}$$

Algorithm 6 summarizes the steps to solve the circulant linear system $Ax = b$ with rank–two modifications in the Serman–Morrison formula.

Note that with this approach, vectors z, u and w might have complex components. The same happens with scalars φ_1, φ_2 and β. In order to compute vector $x_{1:n-1} = z - \varphi_1\beta u + \varphi_2\lambda^{-1}w$ only the real part of the second term is required since the solution of $Ax = b$ is real if it exits.

Using the Sherman–Morrison formula with rank–two modifications proposed in the article from Rojo means more complications to avoid complex

Algorithm 6 Solution of the system $Ax = b$ for a Circulant matrix by the Serman–Morrison Formula.

```
Require: $\lambda, \in \Re$ and $b, \in \Re^n$.
Ensure: The solution vector $x$ of $Ax = b$.
1: Compute $g = b_{1:n-1} + \lambda^{-1} b_n e$, with $e_1, e_{n-1} \in \Re^{n-1}$ and $e = e_1 + e_{n-1}$.
2: Compute $l$ and $d$ of the factors $L$ and $D$.
3: Solve system $LDL^T z = g$.
4: Solve system $LDL^T u = e_1$.
5: Solve system $LDL^T w = e$.
6: Solve system
   $$\begin{pmatrix} 1+\beta u_1 & -\lambda^{-1} w_1 \\ \beta(u_1 + u_{n-1}) & 1 - \lambda^{-1}(w_1 + w_{n-1}) \end{pmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_1 + z_{n-1} \end{pmatrix}.$$
7: Compute $x_{1:n-1} = z - \varphi_1 \beta u + \varphi_2 \lambda^{-1} w$.
8: Compute $x_n = \lambda^{-1} = (b_n + (x_1 + x_{n-1}))$.
```

arithmetic than if done as proposed in the current algorithm with modifications of rank–one. Since the Rojo article only applies to diagonally dominant matrices, it makes sense to use Rojo's algorithm there, as it never presents complex arithmetic. The algorithm based on rank–one modifications is more efficient for the non–diagonal dominant case as the problems of avoiding complex arithmetic are confined to solving a tridiagonal Toeplitz system, where complex arithmetic has already been avoided.

6.3. Using the LDL^T

In order to have a reference algorithm we describe the solution of system $Ax = b$ with $A = \lambda I - (F + F^T) - (e_n e_1^T + e_1 e_n^T)$, with $b = -f/\lambda$, by means of the utilization of the LDL^T decomposition.

Note that the structure of A determines the structure of matrix L in the LDL^T decomposition too. In this case we have,

$$A = \begin{pmatrix} T & -e \\ -e^T & \lambda \end{pmatrix} = \begin{pmatrix} L_1 & \\ z^T & 1 \end{pmatrix} \begin{pmatrix} D_1 & \\ & d_n \end{pmatrix} \begin{pmatrix} L_1^T & z \\ & 1 \end{pmatrix},$$

where $L_1 \in \Re^{(n-1)\times(n-1)}$ is unit lower triangular, $D_1 = \mathrm{diag}(d_1, d_2, \ldots, d_{n-1}) \in \Re^{(n-1)\times(n-1)}$, $d_n \in \Re$, $z \in \Re^{n-1}$ and $e = e_1 + e_{n-1}$, being now e_1 and e_{n-1} the first and the last columns of the $(n-1)$ identity matrix, respectively.

Therefore, L_1 and D_1 can be obtained by computing the LDL^T decomposition of T, vector z by solving the lower triangular system $L_1 D_1 z = -e$, and $d_n = \lambda - z^T D_1 z$.

From this decomposition the solution of the system can be obtained from the solution of triangular systems and some scalar operations in $O(n)$ flops.

Algorithm 7 allows to obtain the solution of $Ax = b$ with the same restrictions of the LDL^T presented in Section 4.

Algorithm 7 Solution of the system $Ax = b$ for a Circulant matrix by the LDL^T decomposition.

```
Require: $\lambda \in \Re$ and $b \in \Re^n$.
Ensure: The solution vector $x$ of $Ax = b$, with
        $$A = \begin{pmatrix} T & -e \\ -e^T & \lambda \end{pmatrix}.$$
    and $e = e_1 + e_{n-1}$ with $e_1, e_{n-1} \in \Re^{n-1}$.
 1: Compute the LDL$^T$ of $T$, $T = L_1 D_1 L_1^T$.
 2: Solve system $L_1 y = -e$.
 3: Compute $z = D_1^{-1} y$.
 4: Compute $d_n = \lambda - z^T y$.
 5: Solve system $L_1 u_{1:n-1} = b_{1:n-1}$.
 6: Compute $u_n = b_n - z^T u_{1:n-1}$.
 7: Compute $x_n = u_n / d_n$.
 8: for $i = 1 : n-1$ do
 9:     $w_i = u_i / d_i$.
10: end for
11: Solve system $L_1^T x_{1:n-1} = w_{1:n-1} - x_n z$.
```

The cost of Algorithm 7 can be evaluated as $18n - 22$ flops.

7. Conclusions

In this work we have extended Rojo's algorithm to the case of non–diagonal dominant matrices. The benefits of this method are good both in terms of runtime and in terms of backward stability. Its application to special systems of equations has been also analyzed. It has been applied to solve circulant tridiagonal linear systems with non–diagonal dominant circulant matrices. Also, it is possible to apply it to other kinds of special systems such as pentadiagonal, heptadiagonals, etc.

References

[1] O. Rojo. A new method for solving symmetric circulant tridiagonal systems of linear equations. *Computers & Mathematics with Applications*, 20(12):61–67, 1990.

[2] Ronald F. Boisvert. Algorithms for special tridiagonal systems. *SIAM J. Sci. Stat. Comput.*, 12(2):423–442, March 1991.

[3] D. J. Evans. An algorithm for the solution of certain tridiagonal systems of linear equations. *The Computer Journal*, 15(4):356–359, 1972.

[4] D. Fischer, G. Golub, O. Hald, C. Leiva, and O. Widlund. On fourier–toeplitz methods for separable elliptic problems. *Mathematics of Computation*, 28(126):349–368, April 1974.

[5] Donald J. Rose. An algorithm for solving a special class of tridiagonal systems of linear equations. *Commun. ACM*, 12(4):234–236, 1969.

[6] Jeffrey Mark McNally. A fast algorithm for solving diagonally dominant symmetric pentadiagonal toeplitz systems. *J. Comput. Appl. Math.*, 234(4):995–1005, 2010.

[7] Salah M. El-Sayed. A direct method for solving circulant tridiagonal block systems of linear equations. *Applied Mathematics and Computation*, 165(1):23–30, 2005.

[8] S.M. El-Sayed, I.G. Ivanov, and M.G. Petkov. A new modification of the rojo method for solving symmetric circulant five-diagonal systems of linear equations. *Computers and Mathematics with Applications*, 35(10):35 – 44, 1998.

[9] J.D. Faires and R.L. Burden. *Numerical Methods.* 4 edition. Cengage Learning, 2012.

[10] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM Press, Philadelphia, 1992.

[11] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[12] W. M. Yan and K. L. Chung. A fast algorithm for solving special tridiagonal systems. *Computing*, 52(2):203–211, 1994.