Efficient algorithm for solving tridiagonal quasi-Toeplitz linear systems

Skander Belhaj
University of Tunis El Manar, ENIT-LAMSIN
Hcini Fahd (hcini.fahd@enit.utm.tn)
University of Tunis El Manar, ENIT-LAMSIN
Yulin Zhang
Universidade do Minho

Research Article

Keywords: Quasi-Toeplitz matrix, Diagonally dominant, LU decomposition

Posted Date: November 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2301272/v1

Additional Declarations: No competing interests reported.

Efficient algorithm for solving tridiagonal quasi-Toeplitz linear systems

Skander Belhaj[a], Hcini Fahd[a,*], Yulin Zhang[b]

[a] *University of Tunis El Manar, ENIT-LAMSIN, BP 37, 1002, Tunis, Tunisia*
[b] *Centro de Matemática, Universidade do Minho, 4710-057 Braga, Portugal*

Abstract

In this paper, a fast algorithm for solving the special tridiagonal quasi-Toeplitz system is presented where the bandwidth of a quasi-Toeplitz is larger than the one of Toeplitz. Our algorithm is quite competitive with the classic LU method. Some examples demonstrate the good efficiency and stability of our algorithm.

Keywords: Quasi-Toeplitz matrix, Diagonally dominant, LU decomposition.
2019 MSC: 15A23, 35L30

1. Introduction

A Toeplitz matrix has the property that the entries are constant along diagonals parallel to the main diagonal. If we define the sequence

$$a_{-p}, a_{-p+1}, \cdots, a_0, \cdots, a_{q-1}, a_q; \quad \text{where} \quad a_{-p}, a_q \neq 0 \tag{1}$$

and p and q are specified positive integers, then the elements of a banded square Toeplitz matrix of order J and bandwidth $p+q+1$, $(\leq J)$ are given by

$$a_{ij} = a_{j-i}.$$

If a_{j-i} is a member of the sequence (1) and zero otherwise, the matrix becomes a tridiagonal Toeplitz matrix, for its properties and applications, see [1].

For this last matrix, several researchers have been developing fast serial and parallel algorithms to solve this kind of systems [2, 3, 4] and [5, 6, 7, 8] . Now if we make some disturbances in the first p and the last q rows of our matrix, for example $(p = q = 1)$, our matrix becomes tridiagonal quasi-Toeplitz. Recently, in [9, 10, 11] , the class of quasi-Toeplitz and their computational problems had been investigated by Bini et al., also its applications in quasi birth-and-death processes, option pricing, numerical solution of ordinary and partial differential equations (ODE and PDE), interpolation problems, boundary value problems (BVP), and signal processing had been studied.

Du et al. [12] presented a method for finding the solution of tridiagonal QT linear systems and prove the efficiency of the new method by numerical results. and more general, the numerical solution of block quasi-tridiagonal Toeplitz matrix was studied by [13]. In [12], it was assumed the bandwidth of a quasi-Toeplitz matrix equals to bandwidth of Toeplitz but, the bandwidth of a quasi-Toeplitz matrix may be larger than the bandwidth of its pure Toeplitz. That is the reason for which, we tackle the problem where the bandwidth of a quasi-Toeplitz is larger than the one of Toeplitz. In this paper, we will focus on the problem of solving n-by-n nonsingular tridiagonal quasi-Toeplitz linear system

$$Tx = f, \tag{2}$$

*Corresponding author
Email address: hcini.fahd@enit.utm.tn (Hcini Fahd)

where the matrix A is defined as

$$T = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \dots & \dots & & \alpha_n \\ c & a & b & & & & & \\ & c & a & b & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & c & a & b & \\ & & & & & c & a & b \\ \beta_1 & \beta_2 & \dots & \dots & \dots & & \beta_{n-1} & \beta_n \end{bmatrix},$$

and a, b, c, α_i and β_i, $i = 1, ..., n$ are real numbers and satisfy the relations $|a| \geq |b| + |c|$, $|\alpha_1| \geq \sum_{i=2}^{n} |\alpha_i|$, $|\beta_n| \geq \sum_{i=1}^{n-1} |\beta_i|$, i.e., the matrix A is diagonally dominant. We suppose that $\alpha_1 \neq a$, $\alpha_2 \neq b$ and $\beta_{n-1} \neq c$, $\beta_n \neq a$ and $b \neq 0$, $c \neq 0$. We can see that

$$T = T' + e_1 u^T + e_n v^T, \tag{3}$$

where e_1 is the first column vector of the identity matrix I and

$$u^T = \begin{bmatrix} \alpha_1 - a & \alpha_2 - b & \alpha_3 & \dots & \alpha_n \end{bmatrix},$$
$$v^T = \begin{bmatrix} \beta_1 & \beta_2 & \dots & \beta_{n-1} - c & \beta_n - a \end{bmatrix},$$

and

$$T' = \begin{bmatrix} a & b & & & & & \\ c & a & b & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & c & a & b & \\ & & & & c & a & \end{bmatrix}.$$

The article is organized as follows. In section 2, we give a new form of decomposition of the coefficient matrix T'. Based on this form of matrix decomposition and combined with the Sherman-Morrison formula, we propose an efficient algorithm for solving tridiagonal quasi-Toeplitz linear systems. In section 3, we present some numerical results to show the efficiency of our algorithm. Finally, we make some conclusions in Section 4.

2. Algorithms for solving tridiagonal quasi-Toeplitz linear systems

In this section we exploit the structure of the matrix T' to give a new decomposition of this matrix, then we use Sherman–Morrison formula to obtain a fast algorithm to solve system (2).

2.1. Special decomposition for matrix T'

The matrix

$$T' = \begin{bmatrix} a & b & & & & & \\ c & a & b & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & c & a & b & \\ & & & & c & a & \end{bmatrix},$$

is a tridiagonal Toeplitz matrix can be decomposed by LU decomposition. For this LU decomposition, $2n-1$ values of $l_{i,j},\ u_{i,j}$ should be determined and saved in memory, but if we see the structure of this matrix we can introduce a new decomposition other than LU.
Let

$$L = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & u_1 & & \\ & \ddots & \ddots & \\ & & 1 & u_1 \\ & & & 1 \end{bmatrix},$$

and $D_{n\times n} = diag(s, s, \ldots, s)$.
We see that the product LDU has the same structure with T', then the matrix T' can be represented as

$$T' = LDU + (a-s)e_1e_1^T,$$

where l_1, s and u_1 verify the following relations:

$$su_1 = b \tag{4}$$
$$sl_1u_1 + s = a \tag{5}$$
$$sl_1 = c. \tag{6}$$

These equations give the unknowns l_1, s and u_1 as follows

$$l_1 = \frac{a-\sqrt{a^2-4bc}}{2b},\ s = \frac{a+\sqrt{a^2-4bc}}{2},\ u_1 = \frac{a-\sqrt{a^2-4bc}}{2c}, \tag{7}$$

or

$$l_1 = \frac{a+\sqrt{a^2-4bc}}{2b},\ s = \frac{a-\sqrt{a^2-4bc}}{2},\ u_1 = \frac{a+\sqrt{a^2-4bc}}{2c}. \tag{8}$$

2.2. Fast algorithm based on a new matrix decomposition

According to (3) the matrix T can be written as

$$T = LDU + e_1z^T + e_nv^T \tag{9}$$

where $z = \begin{bmatrix}\alpha_1 - s & \alpha_2 - b & \alpha_3 & \ldots & \alpha_n\end{bmatrix}$.

Let $R = LDU + e_1z^T$, then

$$T = R + e_nv^T.$$

The inverse of R can be written in terms of matrices L, D, U by using the well-known Sherman–Morrison formula as follows

$$R^{-1} = (LDU)^{-1} - \frac{(LDU)^{-1}e_1z^T(LDU)^{-1}}{1 + z^T(LDU)^{-1}e_1}, \tag{10}$$

therefore the inverse of T can be written

$$T^{-1} = R^{-1} - \frac{R^{-1}e_nv^TR^{-1}}{1 + v^TR^{-1}e_n}. \tag{11}$$

In order to obtain the solution of $Tx = f$ by (10), three linear systems with the same coefficient matrix LDU and different right-hand side vectors f, e_1 and e_n should be solved, i.e., $LDU[r, g, w] = [f, e_1, e_n]$, which can be solved efficiently by Algorithm 1.

Algorithm 1 An algorithm for solving $LDUx = y$

Input: l_1, u_1, s, and y.

1. $z_1 = y_1;\ z_i = y_i - l_1 z_{i-1},\ i = 2, \ldots, n;$ % Forward substitution: $Lz = \boldsymbol{y}$.
2. $q_i = \frac{z_i}{s},\ i = 1, \ldots, n;$ $\%Dq = z$.
3. $x_n = q_n;\ x_i = q_i - u_1 x_{i+1},\ i = n-1, \ldots, 1;$ % Backwalrd substitution: $U\boldsymbol{x} = \boldsymbol{q}$.

Output: $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$;

Finally, we can summarize discussions in this section and give our algorithm in Algorithm 2.

Algorithm 2 Algorithm for solving tridiagonal quasi-Toeplitz linear systems $Tx = f$

1. Compute $r = (LDU)^{-1}f$, $g = (LDU)^{-1}e_1$ and $w = (LDU)^{-1}e_n$ by using Algorithm 1;
2. Compute $\sigma_1 = R^{-1}f = r - \frac{(z^T r)g}{1+z^T g}$, $\sigma_2 = R^{-1}e_n = w - \frac{(z^T w)g}{1+z^T g}$;
3. Compute $T^{-1}f = R^{-1}f - \frac{R^{-1}e_n v^T R^{-1} f}{1+v^T R^{-1} e_n} = \sigma_1 - \frac{(v^T \sigma_1)\sigma_2}{1+v^T \sigma_2}$;

2.3. Stability analysis of our algorithm

The stability of our Algorithm 2 depends on the first step that solving the upper and lower triangular linear systems $LDU[r, g, w] = [f, e_1, e_n]$. More precisely, two recursive iteration steps such as $z_i = y_i - l_1 z_{i-1}$ and $x_{n+1-i} = q_{n+1-i} - u_1 x_{n+2-i}$ for $i = 2, \ldots, n$ corresponding to the forward and backward substitutions as in Algorithm 1 are essential for Algorithm 2. When using finite precision arithmetic, we should avoid roundoff error propagation. If all the roots of their characteristic equations $\lambda + l_1 = 0$ and $\lambda + u_1 = 0$ are all less than unity in magnitude, errors of z_i and x_{n+1-i} will smaller than errors of previous values z_{i-1} and x_{n+2-i}, respectively, in which case that Algorithm 2 is stable.

3. Numerical experiments

In this section, we present some numerical results to illustrate the effectiveness of our proposed algorithm. We have performed the experiments in MATLAB R2017a with an Intel(R) Pentium(R) CPU B960, 2.20GHz 2.20 GHz processor and double precision arithmetic. We initialized the exact solution solution $x^* = [1, 1, \ldots, 1]^T$. The vector on the right-hand side was defined as $f = Ax^*$ and the Relative error $RErr = \frac{\|x - x^*\|_2}{\|x^*\|_2}$ and computation time in seconds will be listed.

3.1. Experiment 1

We consider the scalar hyperbolic equation [14]

$$u_t = cu_x, \quad 0 \le x \le L, \quad t > 0, \tag{12}$$

where $u = u(x, t)$ and c is a real constant. For a well-posed IBVP on a finite domain one must specify initial data, $u(x, 0) = f(x),\ 0 \le x \le L$, and an analytical boundary condition at $x = L$.

$$u(L, t) = g(t), \text{ for } c > 0. \tag{13}$$

To obtain a difference approximation of the model equation (12). We introduce the mesh in the space (x, t) with increments Δx and Δt and an indexation defined by $x = j\Delta x$ and $t = n\Delta t$. we divide the spatial domain $0 \le x \le L$ in J equally spaced increments, i.e. $J\Delta x = L$. As a prototype (explicit) finite-difference approximation for the model equation (12), we consider the Lax-Wendroff scheme

$$u_j^{n+1} = u_j^n + \frac{\nu}{2}\left(u_{j+1}^n - u_{j-1}^n\right) + \frac{\nu^2}{2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right), \tag{14}$$

where $\nu = c\Delta t/\Delta x$ is defined to be the Courant number, this number usually is less than 1. The boundary condition to (13) is homogeneous, i.e.,

$$u_J^n = 0. \tag{15}$$

To compute u_0^{n+1}, it is necessary to introduce additional numerical boundary scheme (NBS). We choose the spatially one-sided scheme :

$$u_0^{n+1} = u_0^n + \nu\left[-\alpha u_2^n + (1+2\alpha)u_1^n - (1+\alpha)u_0^n\right], \tag{16}$$

where α is a (real) parameter. If $\alpha = 0$, then (17) is simply

$$u_0^{n+1} = u_0^n + \nu\left(u_1^n - u_0^n\right). \tag{17}$$

The Lạx-Wendroff scheme (14) together with the analytical boundary condition (15) and the NBS (17) is called a discrete IBVP. Then the Lax-Wendroff scheme (14) with the analytical boundary condition (15) and NBS (17) can be written in vector-matrix form

$$u^{n+1} = Tu^n, \tag{18}$$

where

$$T = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & & & \\ c & a & b & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & c & a & b \\ & & & & c & a \end{bmatrix}, u^n = \begin{bmatrix} u_0^n \\ u_1^n \\ \cdot \\ \cdot \\ \cdot \\ u_{J-2}^n \\ u_{J-1}^n \end{bmatrix}$$

and

$$c = -\nu(1-\nu)/2, \quad a = 1-\nu^2, \quad b = \nu(1+\nu)/2,$$
$$\alpha_1 = 1-\nu(1+\alpha), \quad \alpha_2 = \nu(1+2\alpha), \quad \alpha_3 = -\nu\alpha.$$

We know that, the Courant number usually is small, when $\nu \leq 0.6$, we have that $|a| \geq |b| + |c|$ and l_1, s and u_1 are reals.
After we fixed ν, we chose α such that $|\alpha_1| \geq |\alpha_2| + |\alpha_3|$ to get diagonally dominant matrix T.
Numerical tests have been introduced in Tables 1 to 6 to show that our algorithm is more fast and efficient than other well-known existing methods.

Table 1: Numerical results for $\nu = 0.5$, $\alpha = -0.1$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	5.32e-4	7.2802e-17	2.40e-4	1.5384e-16
10^3	5.04e-4	2.6037e-17	4.56e-4	1.5669e-16
10^4	3.03e-3	8.2336e-18	5.37e-3	1.5698e-16
10^5	2.10e-2	2.6037e-18	3.95e-2	1.5701e-16
10^6	2.40e-1	8.2336e-19	4.16e-1	1.5701e-16

Table 2: Numerical results for $\nu = 0.5$, $\alpha = -0.5$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	4.02e-4	6.1815e-17	1.89e-4	1.5263e-16
10^3	4.61e-4	1.9547e-17	5.66e-4	1.5658e-16
10^4	3.69e-3	6.1815e-18	5.56e-3	1.5697e-16
10^5	2.17e-2	1.9547e-18	8.12e-2	1.5700e-16
10^6	2.08e-1	6.1815e-19	4.09e-1	1.5701e-16

Table 3: Numerical results for $\nu = -0.3$, $\alpha = 1$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	4.88e-4	4.5776e-17	4.27e-4	1.0991e-16
10^3	4.33e-4	1.4476e-17	5.44e-4	1.1091e-16
10^4	2.76e-3	4.5776e-18	5.35e-3	1.1101e-16
10^5	2.05e-2	1.4476e-18	4.50e-2	1.1102e-16
10^6	2.21e-1	4.5776e-19	4.10e-1	1.1102e-16

Table 4: Numerical results for $\nu = 0.48$, $\alpha = -0.8$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	5.35e-4	4.7103e-17	2.29e-4	1.0591e-16
10^3	6.34e-4	1.4895e-17	6.74e-4	1.1052e-16
10^4	2.28e-3	4.7103e-18	6.75e-3	1.1097e-16
10^5	2.20e-2	1.4895e-18	6.70e-2	1.1102e-16
10^6	2.09e-1	4.7103e-19	4.09e-1	1.1102e-16

Table 5: Numerical results for $\nu = 0.25$, $\alpha = 0.4$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	2.51e-4	8.5278e-17	1.47e-4	1.1644e-16
10^3	6.73e-4	7.9208e-17	8.06e-4	1.1158e-16
10^4	2.70-3	7.8575e-17	3.99e-3	1.1108e-16
10^5	2.16e-2	7.8512e-17	4.02e-2	1.1103e-16
10^6	2.30e-1	7.8505e-17	4.12e-1	1.1102e-16

Table 6: Numerical results for $\nu = 0.42$, $\alpha = -0.4$.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	5.99e-4	7.6919e-17	2.57e-4	8.5278e-17
10^3	5.67e-4	2.4324e-17	5.92e-4	7.9208e-17
10^4	2.76e-3	7.6919e-18	5.23e-3	7.8575e-17
10^5	2.23e-2	2.4324e-18	4.97e-2	7.8512e-17
10^6	2.28e-1	7.6919e-19	4.21e-1	7.8505e-17

In all examples, the CPU time required for our algorithms are significantly reduced, it saved at least 50% CPU time, and the precision is higher in almost all cases. This shows the efficiency of our proposed algorithm.

3.2. Experiment 2

Three artificial examples were used in this experiment. Values of a, b, c, α_i and β_i corresponding to each example are presented in Table 7.

Table 7: Test examples.

	a	b	c	α_i	β_i
Example 1	4	1	0.5	$\alpha_1 = 4$, $\alpha_2 = 2$, $\alpha_3 = 0.5$, $\alpha_{i4\leq i\leq n} = 0$	$\beta_{n-2} = 0.5$, $\beta_{n-1} = 1$, $\beta_n = 2$, $\beta_{i1\leq i\leq n-4} = 0$
Example 2	6	-1.2	-0.65	$\alpha_1 = -5.2$, $\alpha_2 = 4$, $\alpha_3 = -1$, $\alpha_4 = -0.4$, $\alpha_{i5\leq i\leq n} = 0$	$\beta_{n-3} = -0.6$, $\beta_{n-2} = -0.5$, $\beta_{n-1} = 1.5$ $\beta_n = 6$, $\beta_{i1\leq i\leq n-5} = 0$
Example 3	9.5	2.3	-3.2	$\alpha_1 = 10$, $\alpha_2 = 4.5$, $\alpha_3 = 2$, $\alpha_4 = 0.5$, $\alpha_5 = 0.6$, $\alpha_{i6\leq i\leq n} = 0$	$\beta_{n-4} = 4$, $\beta_{n-3} = 2$, $\beta_{n-2} = -0.5$ $\beta_{n-1} = 1$, $\beta_n = 11$, $\beta_{i1\leq i\leq n-5} = 0$

The compared results between our proposed algorithm and the LU method for all examples are presented in Tables 8-10.

Table 8: Numerical results for Example 1.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	3.04e-4	6.5682e-17	1.64e-4	1.1213e-16
10^3	8.02e-4	2.0770e-17	8.27e-4	1.1113e-16
10^4	2.64e-3	6.5682e-18	4.35e-3	1.1103e-16
10^5	1.86e-2	2.0770e-18	3.94e-2	1.1102e-16
10^6	2.16e-1	6.5682e-19	4.12e-1	1.1102e-16

Table 9: Numerical results for Example 2.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	5.96e-4	8.3081e-17	2.59e-4	1.1484e-16
10^3	4.12e-4	2.6273e-17	8.07e-4	1.1141e-16
10^4	2.91e-3	8.3081e-18	4.49e-3	1.1106e-16
10^5	2.09e-2	2.6273e-18	3.94e-2	1.1103e-16
10^6	2.29e-1	8.3081e-19	4.21e-1	1.1102e-16

Table 10: Numerical results for Example 2.

n	Our algorithm		LU method	
	Time [s]	RErr	Time [s]	RErr
10^2	2.28e-4	1.1484e-16	1.37e-4	1.5060e-16
10^3	4.76e-4	3.6316e-17	8.34e-4	1.5638e-16
10^4	2.86e-3	1.1484e-17	9.65e-3	1.5695e-16
10^5	1.96e-2	3.6316e-18	4.76e-2	1.5700e-16
10^6	2.23e-1	1.1484e-18	4.30e-1	1.5701e-16

Based on the numerical results, all the numerical results in Tables 8-10 show that the proposed algorithm takes less CPU time and is more efficient than the LU method especially when n increases.

4. Conclusions

In this paper, we have proposed a new algorithm for solving tridiagonal quasi-Toeplitz linear system such that the bandwidth of a quasi-Toeplitz is larger than the one of Toeplitz. The numerical results show the effectiveness of our method. The error and computation time of our algorithm are lower than other well-known existing methods. The efficiency of our algorithm is justified by numerical experiments.

Funding

The last author was partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the Projects UIDB/00013/2020 and UIDP/00013/2020.

Declarations

Ethical Approval

Not Applicable.

Availability of supporting data

Not Applicable.

Competing interests

The author has no relevant financial or non-financial interests to disclose.

Authors' contributions

The authors confirm sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation

Acknowledgments

Not applicable.

References

[1] Silvia Noschese, Lionello Pasquini, and Lothar Reichel. Tridiagonal toeplitz matrices: properties and novel applications. *Numerical linear algebra with applications*, 20(2):302–326, 2013.

[2] James R Bunch and Roummel F Marcia. A pivoting strategy for symmetric tridiagonal matrices. *Numerical Linear Algebra with Applications*, 12(9):911–922, 2005.

[3] Michael A Malcolm and John Palmer. A fast method for solving a class of tridiagonal linear systems. *Communications of the ACM*, 17(1):14–17, 1974.

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

[5] LE Garey and RE Shaw. A parallel method for linear equations with tridiagonal toeplitz coefficient matrices. *Computers & Mathematics with Applications*, 42(1-2):1–11, 2001.

[6] LE Garey, RE Shaw, and J Zhang. Parallel projection algorithms for tridiagonal toeplitz systems. In *High Performance Computing Systems and Applications*, pages 75–86. Springer, 2003.

[7] Hyoung Joong Kim and Jang Gyu Lee. A parallel algorithm solving a tridiagonal toeplitz linear system. *Parallel computing*, 13(3):289–294, 1990.

[8] Maryam Majedi, Ruth E Shaw, and Lawrence E Garey. A parallel sewing method for solving tridiagonal toeplitz strictly diagonally dominant systems. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8. IEEE, 2008.

[9] Dario Bini, Stefano Massei, and Beatrice Meini. Semi-infinite quasi-toeplitz matrices with applications to qbd stochastic processes. *Mathematics of Computation*, 87(314):2811–2830, 2018.

[10] Dario A Bini, Stefano Massei, and Leonardo Robol. Quasi-toeplitz matrix arithmetic: a matlab toolbox. *Numerical Algorithms*, 81(2):741–769, 2019.

[11] Dario A Bini and Beatrice Meini. On the exponential of semi-infinite quasi-toeplitz matrices. *Numerische Mathematik*, 141(2):319–351, 2019.

[12] Lei Du, Tomohiro Sogabe, and Shao-Liang Zhang. A fast algorithm for solving tridiagonal quasi-toeplitz linear systems. *Applied Mathematics Letters*, 75:74–81, 2018.

[13] Skander Belhaj, Fahd Hcini, and Yulin Zhang. A fast method for solving a block tridiagonal quasi-toeplitz linear system. *Portugaliae Mathematica*, 76(3):287–299, 2020.

[14] Robert F Warming and Richard M Beam. An eigenvalue analysis of finite-difference approximations for hyperbolic ibvps. In *Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 564–573. Springer, 1990.