J Math Chem (2015) 53:1617–1633
DOI 10.1007/s10910-015-0509-3

ORIGINAL PAPER

A structure-preserving algorithm for linear systems with circulant pentadiagonal coefficient matrices

Qiong-Xiang Kong[1] · **Ji-Teng Jia**[2,3]

Received: 19 December 2014 / Accepted: 19 April 2015 / Published online: 28 April 2015

Abstract Circulant pentadiagonal (CP) systems of linear equations arise in many application areas and have been thoroughly studied in the past decades. In the current paper, a novel algorithm is presented for solving CP linear systems based upon a structure-preserving factorization for the coefficient matrix. Meanwhile, we show that the proposed algorithm is competitive with some already existing algorithms in terms of arithmetic operations. In addition, a symmetric case of the CP linear systems is also considered. Finally, two examples are provided in order to demonstrate the validity and efficiency of our algorithm and its competitiveness with other algorithms. All of the numerical experiments are performed on a computer with the aid of programs written in Matlab.

Keywords Circulant pentadiagonal matrices · Circulant tridiagonal matrices · Toeplitz matrices · Matrix factorization · Linear systems · Computational costs

Mathematics Subject Classification 15A15 · 15B05 · 65F30 · 65F50

This work was supported by the Natural Science Foundation of China (NSFC) under Grant 11371287 and the International Science and Technology Cooperation Program of China under Grant 2010DFA14700.

✉ Ji-Teng Jia
lavenderjjt@163.com

1 Department of Building Environment and Services Engineering, Xi’an Jiaotong University, Xi’an 710049, Shaanxi, China

2 School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an 710049, Shaanxi, China

3 Present Address: Department of Computer Science, University of Regina, Regina, SK, Canada

1 Introduction

Complex mathematical models are frequently used in the modern computer age to deal with many difficult problems which arise in several mathematical chemistry as well as scientific and engineering investigations. During the last 40 or 50 years, the size of the mathematical models and the number of linear equations which have to be handled after their discretization were greatly increased in these areas. For example, the work with the mathematical model discussed in [1], which is used for investigating the long-range transport of air pollutants in a large spatial domain containing the whole of Europe, was initiated in 1980. The gradually increased computational complexity of this large-scale model during the period 1980–2014 (due to the computational grid-points and to the numbers of involved chemical species) has been summarized in [2], see Table 1.

From Table 1, we note that the number of the equations that are to be handled at each time-step is dramatically increased from 2048 up to 389376000. Therefore, the studies related to the development of novel and more efficient numerical (or symbolic) algorithms as well as to the improvement of the already existing algorithms are extremely important for researchers. However, the choice of a computational algorithm for large-scale models is always a result of some compromise. The need of a suitable compromise can be illustrated by the following two aspects:

(I) Applying divide and conquer (D & C) techniques may facilitate the choice of algorithms, however, the accuracy of these techniques is normally low.
(II) Increasing the accuracy of the numerical solutions may lead to more time-consuming algorithms.

In this paper, we mainly consider an n-by-n circulant pentadiagonal (CP) system of linear equations defined by

$$A\mathbf{x} = \mathbf{f}, \tag{1.1}$$

Table 1 The increase of the computational complexity of an air pollution model in the period 1980–2014

No.	Dimensionality	Number of grid-point	Chemical species	Number of equations
1	2-D	32×32	2	2048
2	2-D	32×32	35	35,840
3	3-D	$32 \times 32 \times 10$	35	358,400
4	2-D	96×96	35	322,560
5	3-D	$96 \times 96 \times 10$	35	32,25,600
6	2-D	480×480	35	8064000
7	3-D	$480 \times 480 \times 10$	35	8,06,40,000
8	2-D	480×480	56	1,29,02,400
9	3-D	$480 \times 480 \times 10$	56	12,90,24,000
10	2-D	480×480	169	3,89,37,600
11	3-D	$480 \times 480 \times 10$	169	38,93,76,000

where

$$A := \begin{bmatrix} d & a & b & & & e & c \\ c & d & a & b & & & e \\ e & c & d & a & b & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & b \\ b & & & e & c & d & a \\ a & b & & & e & c & d \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{1.2}$$

$\mathbf{x} := [x_1, x_2, \ldots, x_n]^T$, $\mathbf{f} := [f_1, f_2, \ldots, f_n]^T$ are unknown and known vectors of length n, respectively. Without loss of generality, we assume that $be \neq 0$. This type of system often appears in quintic spline problems, boundary value problems (BVPs), fluid mechanics, Hückel theory, parallel computing, and numerical solution of ordinary differential equations (ODEs), especially because the discretization of second-order linear differential equations with periodic boundary conditions, transforming them into finite-difference equations, often results in the CP linear systems, see [3–9]. Two important examples of the second-order differential equations that frequently arise in chemical engineering are Bessel's equation

$$x^2y'' + xy' + \left(x^2 - n^2\right) y = 0,$$

and the confluent hypergeometric equation

$$x\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + (c - x)\frac{\mathrm{d}y}{\mathrm{d}x} - ay = 0,$$

see [10] for details. Moreover, in paper [11], a sixth-order uniform mesh difference scheme using sextic splines for solving a self-adjoint singularly perturbed two-point boundary-value problem arising in the study of chemical reactor theory, of the form

$$\begin{cases} -\varepsilon u'' + p(x)u = f(x), \ p(x) > 0, \\ u(0) = \alpha_0, \ u(1) = \alpha_1, \end{cases}$$

is derived. And, the proposed scheme leads to a nearly pentadiagonal Toeplitz linear systems which is a special case of the CP linear systems (1.1).

A standard solution of the CP linear systems can be obtained if we know the inverse of the coefficient matrix A, which would reduce the original problem of computing the solution for such systems to one of matrix multiplication, i.e. $\mathbf{x} = A^{-1}\mathbf{f}$. However, methods of implementing an algorithm to compute the solution by using the procedure described above, are extremely inefficient for large-scale systems since the number of required operations grows very fast. Therefore, more specific algorithms have been devised for solving CP and pentadiagonal block circulant (PBC) linear systems in recent years, see [12–20]. In addition, some authors have presented fast numerical algorithms for periodic pentadiagonal Toeplitz (PPT) linear systems which are special case of our problem, see e.g. [21–24] and references therein. In this study, our main objective is to construct a structure-preserving algorithm for solving system (1.1) without imposing any restrictive conditions.

The remainder of this paper is organized as follows: in the next section, we show that a reliable symbolic algorithm with less computation costs is derived from the use of a structure-preserving matrix factorization and the circulant tridiagonal linear solver described in our framework [25], that can be regarded as a generalization of the algorithm given in [21]. In addition, we derive an algorithm for solving a symmetric case of the CP linear systems, and give the feasibility and stability analysis of the algorithm. In Sect. 3, the proposed algorithm is demonstrated by some simple numerical experiments. Finally, we make some concluding remarks of the present study in Sect. 4.

2 Main results

In this section, we will develop an efficient structure-preserving algorithm for the CP linear systems as in (1.1). Before describing the detail, let us review two existing algorithms (Jia, Kong and Sogabe's algorithm [26] and Navon's algorithm [27]), and give the computational costs of these algorithms.

2.1 Existing algorithms for solving CP linear systems

Recently, in their paper [26], Jia, Kong and Sogabe presented a recursive algorithm for solving CP linear systems in linear time. The algorithm is based on the Sherman-Morrison-Woodbury formula [28] and a matrix factorization that represents the coefficient matrix A as a sum of pentadiagonal matrix and rank-two matrix, thus

$$A = P + UV, \tag{2.1}$$

where

$$P := \begin{bmatrix} d-b & a & b & & & & \\ c-a & d-b & a & b & & & \\ e & c & d & a & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & c & d & a & b \\ & & & e & c & d-e & a-c \\ & & & & e & c & d-e \end{bmatrix} \in \mathbb{R}^{n\times n},$$

$$U := [\mathbf{u}_1, \mathbf{u}_2] = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^{n\times 2},$$

and

$$V := \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} b & 0 & \cdots & 0 & e & c \\ a & b & 0 & \cdots & 0 & e \end{bmatrix} \in \mathbb{R}^{2\times n}.$$

It follows from the Sherman-Morrison-Woodbury formula that the matrix A is invertible if and only if $I_2 + VP^{-1}U$ is invertible, and

$$A^{-1} = P^{-1} - P^{-1}U(I_2 + VP^{-1}U)^{-1}VP^{-1}. \tag{2.2}$$

Here, I_2 denotes the 2-by-2 identity matrix. On both sides of Eq. (2.2), multiply by $\mathbf{f}$ on the right to obtain

$$\begin{aligned} \mathbf{x} &= \mathbf{y} - [\mathbf{z}, \mathbf{w}] \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix}^{-1} \begin{bmatrix} by_1 + ey_{n-1} + cy_n \\ ay_1 + by_2 + ey_n \end{bmatrix} \\ &= \mathbf{y} - d_1\mathbf{z} - d_2\mathbf{w}, \end{aligned}$$

where $\mathbf{y} = [y_1, y_2, \dots, y_n]^T$, $\mathbf{z} = [z_1, z_2, \dots, z_n]^T$, $\mathbf{w} = [w_1, w_2, \dots, w_n]^T$ are solutions of the following linear systems with multiple right-hand sides

$$P[\mathbf{y}|\mathbf{z}|\mathbf{w}] = [\mathbf{f}|\mathbf{u}_1|\mathbf{u}_2], \tag{2.3}$$

and

$$m_i = \begin{cases} bz_1 + ez_{n-1} + cz_n + 1 & \text{if } i = 1 \\ bw_1 + ew_{n-1} + cw_n & \text{if } i = 2 \\ az_1 + bz_2 + ez_n & \text{if } i = 3 \\ aw_1 + bw_2 + ew_n + 1 & \text{if } i = 4, \end{cases} \tag{2.4}$$

$$d_i = \begin{cases} \dfrac{(bm_4 - am_2)y_1 + (cm_4 - em_2)y_n + em_4y_{n-1} - bm_2y_2}{m_1m_4 - m_2m_3} & \text{if } i = 1 \\ \dfrac{(am_1 - bm_3)y_1 + (em_1 - cm_3)y_n - em_3y_{n-1} + bm_1y_2}{m_1m_4 - m_2m_3} & \text{if } i = 2. \end{cases} \tag{2.5}$$

The resulting algorithm can be summarized as follows:

Algorithm 2.1 Jia, Kong and Sogabe's algorithm [26]

```
Step 1 Input $P$, $U$, $V$, $\mathbf{f}$ and order $n$.
Step 2 Solve linear systems (2.3) by using the pentadiagonal linear solver given in [26, p. 1240].
Step 3 Compute $m_1$, $m_2$, $m_3$ and $m_4$ by using (2.4).
Step 4 Compute $d_1$ and $d_2$ by using (2.5).
Step 5 Output the solution of system: $\mathbf{x} = \mathbf{y} - d_1\mathbf{z} - d_2\mathbf{w}$.
```

If we measure the computational costs of the above algorithm in terms of total number of operations, where each operation represents one of the four arithmetic floating point operations, then Jia, Kong and Sogabe's algorithm requires $48n - 24$ operations for solving the CP linear system (1.1). For the validity of Algorithm 2.1, the conditions are that P is invertible and $m_1m_4 - m_2m_3 \neq 0$.

Since the difference between the pentadiagonal matrix and the circulant pentadiagonal matrix is only the $(1, n-1)$, $(1, n)$, $(2, n)$, $(n-1, 1)$, $(n, 1)$ and $(n, 2)$ entries, i.e., an n-by-n circulant pentadiagonal matrix $A = [a_{ij}]$ with $a_{1,n-1} = a_{1,n} = a_{2,n} = a_{n-1,1} = a_{n,1} = a_{n,2} = 0$ reduces to a pentadiagonal Toeplitz matrix, Navon proposed an approach for solving CP linear systems based on the use of pentadiagonal linear solvers, see [27]. In this approach, the original coefficient matrix A is split into the following 2-by-2 block matrix form

$$A = \begin{bmatrix} \hat{P} & \hat{U} \\ \hat{V} & D \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{2.6}$$

where

$$\begin{aligned}
\hat{U} &:= [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2] = \begin{bmatrix} e & 0 & \cdots & 0 & b & a \\ c & e & 0 & \cdots & 0 & b \end{bmatrix}^T \in \mathbb{R}^{(n-2)\times 2}, \\
\hat{V} &:= \begin{bmatrix} \hat{\mathbf{v}}_1 \\ \hat{\mathbf{v}}_2 \end{bmatrix} = \begin{bmatrix} b & 0 & \cdots & 0 & e & c \\ a & b & 0 & \cdots & 0 & e \end{bmatrix} \in \mathbb{R}^{2\times(n-2)}, \\
D &:= \begin{bmatrix} d & a \\ c & d \end{bmatrix} \in \mathbb{R}^{2\times 2},
\end{aligned}$$

and $\hat{P}$ is the $(n-2)$-th leading principal submatrix of A. Remarkably, the matrix $\hat{P}$ inherits some nice properties from the matrix A as shown below.

Proposition 2.1 *Let A be an n-by-n circulant pentadiagonal matrix and $\hat{P}$ its $(n-2)$-th leading principal submatrix. Then we have*

1. *If A is symmetric positive definite, then $\hat{P}$ is symmetric positive definite;*
2. *If A is symmetric negative definite, then $\hat{P}$ is symmetric negative definite and $(-1)^{n-2}det(\hat{P}) > 0$;*
3. *If A is totally positive (nonnegative) matrix, then $\hat{P}$ is totally positive (nonnegative) matrix;*
4. *If A is row (column) diagonally dominant, then $\hat{P}$ is row (column) diagonally dominant.*

Proof Since $\hat{P}$ is the $(n-2)$-th leading principal submatrix of A, we readily obtain the above properties by using the corresponding results given in [28]. □

Based on the partition given in (2.6), the original CP linear system (1.1) can be rewritten as

$$\begin{bmatrix} \hat{P} & \hat{U} \\ \hat{V} & D \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \tilde{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{f}} \\ \tilde{\mathbf{f}} \end{bmatrix}, \tag{2.7}$$

where

$$\hat{\mathbf{x}} := [x_1, x_2, \ldots, x_{n-2}]^T \in \mathbb{R}^{(n-2)\times 1},\ \tilde{\mathbf{x}} := [x_{n-1}, x_n]^T \in \mathbb{R}^{2\times 1},$$

and

$$\hat{\mathbf{f}} := [f_1, f_2, \ldots, f_{n-2}]^T \in \mathbb{R}^{(n-2)\times 1},\ \tilde{\mathbf{f}} := [f_{n-1}, f_n]^T \in \mathbb{R}^{2\times 1}.$$

Thus, we have

$$\begin{cases} \hat{P}\hat{\mathbf{x}} + \hat{U}\tilde{\mathbf{x}} = \hat{\mathbf{f}}, \\ \hat{V}\hat{\mathbf{x}} + D\tilde{\mathbf{x}} = \tilde{\mathbf{f}}. \end{cases} \tag{2.8}$$

From the first equation of (2.8), we can deduce that

$$\hat{\mathbf{x}} = \hat{P}^{-1}\hat{\mathbf{f}} - \hat{P}^{-1}\hat{U}\tilde{\mathbf{x}}. \tag{2.9}$$

And, if we substitute (2.9) into the second equation of (2.8), it yields

$$\hat{V}\hat{P}^{-1}\hat{\mathbf{f}} - \hat{V}\hat{P}^{-1}\hat{U}\tilde{\mathbf{x}} + D\tilde{\mathbf{x}} = \tilde{\mathbf{f}},$$

thus

$$\left(D - \hat{V}\hat{P}^{-1}\hat{U}\right)\tilde{\mathbf{x}} = \tilde{\mathbf{f}} - \hat{V}\hat{P}^{-1}\hat{\mathbf{f}}$$

which permits us to obtain the two components of the unknown vector $\tilde{\mathbf{x}}$ explicitly as

$$\tilde{\mathbf{x}} = \left(D - \hat{V}\hat{P}^{-1}\hat{U}\right)^{-1}\left(\tilde{\mathbf{f}} - \hat{V}\hat{P}^{-1}\hat{\mathbf{f}}\right). \tag{2.10}$$

Let $\hat{\mathbf{w}} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{n-2}]^T$, $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n-2}]^T$, $\hat{\mathbf{z}} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_{n-2}]^T$ are solutions of the systems

$$\hat{P}\hat{\mathbf{w}} = \hat{\mathbf{u}}_1,\ \ \hat{P}\hat{\mathbf{y}} = \hat{\mathbf{u}}_2,\ \ \hat{P}\hat{\mathbf{z}} = \hat{\mathbf{f}}, \tag{2.11}$$

respectively. Then, we have

$$\tilde{\mathbf{x}} = \hat{D}^{-1}\bar{\mathbf{f}}, \tag{2.12}$$

where

$$\hat{D} = \begin{bmatrix} d - b\hat{w}_1 - e\hat{w}_{n-3} - c\hat{w}_{n-2} & a - b\hat{y}_1 - e\hat{y}_{n-3} - c\hat{y}_{n-2} \\ c - a\hat{w}_1 - b\hat{w}_2 - e\hat{w}_{n-2} & d - a\hat{y}_1 - b\hat{y}_2 - e\hat{y}_{n-2} \end{bmatrix},$$

and

$$\bar{\mathbf{f}} = \begin{bmatrix} f_{n-1} - b\hat{z}_1 - e\hat{z}_{n-3} - c\hat{z}_{n-2} \\ f_n - a\hat{z}_1 - b\hat{z}_2 - e\hat{z}_{n-2} \end{bmatrix}.$$

Finally, the sought values of $\hat{\mathbf{x}}$ can be calculated by using (2.9). The corresponding algorithm is given below.

Algorithm 2.2 Navon's algorithm [27]

```
Step 1 Input $\hat{P}$, $\hat{U}$, $\hat{V}$, $D$, $\hat{\mathbf{f}}$, $\tilde{\mathbf{f}}$ and $n$.
Step 2 Based on the $LU$ factorization of matrix $\hat{P}$, solve equations (2.11) by using forward substitution and back substitution.
Step 3 Compute $\tilde{\mathbf{x}} = [x_{n-1}, x_n]^T$ by using (2.12).
Step 4 Compute $\hat{\mathbf{x}} = [x_1, x_2, \ldots, x_{n-2}]^T$ by using (2.9).
Step 5 Output the solution of system: $\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}} \\ \tilde{\mathbf{x}} \end{bmatrix}$.
```

From the above we can see that it is important to solve three $(n-2)$-by-$(n-2)$ pentadiagonal Toeplitz linear systems $\hat{P}\hat{\mathbf{w}} = \hat{\mathbf{u}}_1$, $\hat{P}\hat{\mathbf{y}} = \hat{\mathbf{u}}_2$ and $\hat{P}\hat{\mathbf{z}} = \hat{\mathbf{f}}$ in the algorithm. Therefore, for validity of the algorithm, the condition is that the matrix $\hat{P}$ is invertible. Moreover, we note that the computational costs of Algorithm 2.2 is $41n - 29$ and this leads to the result that Algorithm 2.2 is about 15 % faster than Algorithm 2.1 when the system order n is large enough.

2.2 A structure-preserving algorithm based on circulant tridiagonal linear solvers

In this section, we focus on the construction of a novel computational algorithm for solving CP linear systems. First, let us define two circulant tridiagonal matrices as follows

$$T_1 := \begin{bmatrix} s & 1 & & & \frac{e}{p} \\ \frac{e}{p} & s & 1 & & \\ & \frac{e}{p} & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & \frac{e}{p} & s \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad T_2 := \begin{bmatrix} t & \frac{b}{p} & & & 1 \\ 1 & t & \frac{b}{p} & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{b}{p} \\ \frac{b}{p} & & & 1 & t \end{bmatrix} \in \mathbb{R}^{n\times n},$$

where the parameters p, s and t satisfy the following nonlinear equations

$$\begin{cases} et + sp - c = 0, \\ bs + tp - a = 0, \\ (st+1)p^2 - dp + be = 0. \end{cases} \tag{2.13}$$

Then, we can readily confirm that $A = pT_1T_2$ where A is the circulant pentadiagonal matrix defined in (1.2). This is a structure-preserving factorization of the matrix A. Consequently, solving the linear system $A\mathbf{x} = \mathbf{f}$ can be achieved by solutions of two circulant tridiagonal (CT) linear systems

$$T_1\mathbf{y} = \frac{\mathbf{f}}{p}, \quad T_2\mathbf{x} = \mathbf{y}. \tag{2.14}$$

It follows from (2.13) that in order to calculate p, s, and t, we should solve the following sextic equation at first:

$$p^6 - dp^5 + (ac - be)p^4 + \left(2bde - a^2e - c^2b\right) p^3 + \left(abce - b^2e^2\right) p^2 - b^2e^2dp + b^3e^3 = 0. \tag{2.15}$$

Proposition 2.2 *If we set* $\alpha = \frac{d}{3}$, $\beta = \frac{d^2+12be-3ac}{9}$, $\gamma = \frac{3a^2e+3c^2b-8bde-acd}{6}$ *and* $\eta = \sqrt[3]{\alpha^3 + \gamma + \sqrt{(\alpha^3+\gamma)^2 - \beta^3}}$, *then the analytic solutions of Eq. (2.15) can be given as*

$$p_{2i-1} = \frac{\lambda_i + \sqrt{\lambda_i^2 - 4ae}}{2} \ \textit{and} \ p_{2i} = \frac{\lambda_i - \sqrt{\lambda_i^2 - 4ae}}{2}, \ i = 1, 2, 3,$$

where

$$\lambda_1 = \alpha + \eta + \frac{\beta}{\eta},$$
$$\lambda_2 = \alpha - \frac{\eta}{2} - \frac{\beta}{2\eta} + i\frac{\sqrt{3}}{2}\left(\eta - \frac{\beta}{\eta}\right),$$
$$\lambda_3 = \alpha - \frac{\eta}{2} - \frac{\beta}{2\eta} - i\frac{\sqrt{3}}{2}\left(\eta - \frac{\beta}{\eta}\right).$$

Proof In order to calculate the analytic solutions, we first divide the both sides of Eq. (2.15) by p^3, and define $\lambda = p + \frac{be}{p}$. Then, Eq. (2.15) can be reformulated as

$$\lambda^3 - d\lambda^2 - (4be - ac)\lambda + 4bde - c^2b - a^2e = 0.$$

Here, we use the Cardano's method to solve the above cubic equation, then we have

$$\lambda_1 = \alpha + \sqrt[3]{\alpha^3 + \gamma + \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}} + \sqrt[3]{\alpha^3 + \gamma - \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}}$$

$$\lambda_2 = \alpha - \frac{1-\sqrt{3}i}{2}\sqrt[3]{\alpha^3 + \gamma + \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}} - \frac{1+\sqrt{3}i}{2}\sqrt[3]{\alpha^3 + \gamma - \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}}$$

$$\lambda_3 = \alpha - \frac{1+\sqrt{3}i}{2}\sqrt[3]{\alpha^3 + \gamma + \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}} - \frac{1-\sqrt{3}i}{2}\sqrt[3]{\alpha^3 + \gamma - \sqrt{\left(\alpha^3+\gamma\right)^2 - \beta^3}}$$

where α, β, γ are defined above. Finally, the solutions of Eq. (2.15) can be calculated and simplified to the following form

$$p_{1,2} = \frac{\lambda_1 \pm \sqrt{\lambda_1^2 - 4be}}{2}, \ p_{3,4} = \frac{\lambda_2 \pm \sqrt{\lambda_2^2 - 4be}}{2}, \ p_{5,6} = \frac{\lambda_3 \pm \sqrt{\lambda_3^2 - 4be}}{2}.$$

That completes the proof. □

Throughout this paper, we pre-assume that the sextic equation (2.15) has real roots with a, b, c, d, e. Then, after determining p, we can compute s and t by using Eq. (2.13). In particular, if $p^2 - be \neq 0$, it follows from the first two equations of Eq. (2.13) that we have

$$s = \frac{cp - ae}{p^2 - be}, \quad t = \frac{ap - bc}{p^2 - be}. \tag{2.16}$$

The resulting algorithm can be summarized as follows:

Algorithm 2.3

Step 1 Input A, $\mathbf{f}$ and order n.
Step 2 Compute the circulant tridiagonal matrices T_1 and T_2 by using Proposition 2.2 and Eq. (2.13).
Step 3 Solve systems $T_1\mathbf{y} = \frac{\mathbf{f}}{p}$ and $T_2\mathbf{x} = \mathbf{y}$ by using any circulant tridiagonal linear solver.
Step 4 Output the solution of system: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$.

Remark 2.1 It should be mentioned that even if a complex root of Eq. (2.15) is chosen. Algorithm 2.3 will produce the solution of system (1.1) by permitting complex operations. Thus, the algorithm actually works without the pre-assumption that Eq. (2.15) has real roots. Such pre-assumption, however, will be required, if all computation are required to be done in real number.

2.3 A symbolic algorithm for solving periodic tridiagonal linear systems

From the previous subsection, we note that using a reliable circulant tridiagonal linear solver makes our algorithm (Algorithm 2.3) robust. Therefore, an efficient algorithm will be described for solving circulant tridiagonal (CT) linear systems in this subsection.

Without imposing any restrictive conditions, in their paper [25], Jia and Kong have developed a symbolic algorithm (SPT algorithm) for periodic tridiagonal systems of equations

$$T\mathbf{x} = \mathbf{f}, \tag{2.17}$$

where T is a periodic tridiagonal matrix of the form

$$T := \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

The algorithm is based on the *LMU* matrix factorization that represents the coefficient matrix T as a product of three matrices, i.e., $T = LMU$, where L is lower bidiagonal matrix, M is nearly upper unitriangular matrix, and U is upper unitriangular matrix, for details, see [25, p. 2226]. Finally, solving $T\mathbf{x} = \mathbf{f}$ can be achieved by solutions of three linear systems $L\mathbf{z} = \mathbf{f}$, $M\mathbf{y} = \mathbf{z}$, and $U\mathbf{x} = \mathbf{y}$. The corresponding algorithm is given below.

Algorithm 2.4 SPT algorithm [25]

```
Step 1 Input $a_i, b_i, c_i, f_i$ and $n$.
Step 2 $LMU$ factorization stage:
        Set $d_1 = b_1$, If $d_1 = 0$ then $d_1 = \tau$ end if. Set $e_1 = \frac{c_1}{d_1}$, $g'_1 = \frac{a_1}{d_1}$,
          For $i = 2, 3, \ldots, n-1$ compute and simplify
            $d_i = b_i - e_{i-1}a_i$, If $d_i = 0$ then $d_i = \tau$ end if.
            $e_i = \frac{c_i}{d_i}$, $g'_i = -\frac{a_i g'_{i-1}}{d_i}$,
          End.
        Set $g'_{n-1} = \frac{c_{n-1} - a_{n-1}g'_{n-2}}{d_{n-1}}$, $g_{n-1} = g'_{n-1}$,
          For $i = n-2, n-3, \ldots, 1$ compute and simplify
            $g_i = g'_i - e_i g_{i+1}$,
          End.
        Set $d_n = b_n - c_n g_1 - a_n g_{n-1}$, If $d_n = 0$ then $d_n = \tau$ end if.
Step 3 Solution stage:
        Set $z_1 = \frac{f_1}{d_1}$,
          For $i = 2, 3, \ldots, n$ compute and simplify
            $z_i = \frac{f_i - a_i z_{i-1}}{d_i}$,
          End.
        Set $y_{n-1} = z_{n-1}$,
          For $i = n-2, n-3, \ldots, 1$ compute and simplify
            $y_i = z_i - e_i y_{i+1}$,
          End.
        Set $g_n = \frac{c_n}{d_n}$, $x_n = z_n - g_n y_1$,
          For $i = n-1, n-2, \ldots, 1$ compute and simplify
            $x_i = y_i - g_i x_n$,
          End
```

Since the *LMU* factorization depends on a formal parameter τ which can be regarded as a symbolic name whose actual value is 0, such factorization always exists even if the coefficient matrix T is singular, and this leads to the reliable solution of the periodic tridiagonal linear system (2.17).

On the other hand, we note that the SPT algorithm can be directly applied to the CT linear systems $T_1\mathbf{y} = \frac{\mathbf{f}}{p}$ and $T_2\mathbf{x} = \mathbf{y}$ given in (2.14), by setting $a_i = \frac{e}{p}$, $b_i = s$,

$c_i = 1$ and $a_i = 1, b_i = t, c_i = \frac{b}{p}$, respectively. We now state a symbolic algorithm for CP linear systems below.

```
Algorithm 2.5 Symbolic algorithm for $A\mathbf{x} = \mathbf{f}$
Step 1 Input $a, b, c, d, e, n$ and $\mathbf{f}$.
Step 2 Solve Eq. (2.15) by using Proposition 2.2, and choose a root $p$.
Step 3 Compute $s$ and $t$ by using Eq. (2.13).
Step 4 Solve systems $T_1\mathbf{y} = \frac{\mathbf{f}}{p}$ and $T_2\mathbf{x} = \mathbf{y}$ by using Algorithm 2.4.
Step 5 Evaluate $\mathbf{y}$ and $\mathbf{x}$ with $\tau = 0$.
Step 6 Output the solution of system: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$.
```

The above algorithm will be referred to as the SCP algorithm. In general, it is difficult to know the total costs a priori when complex numbers and symbolic names arise during the computation. Therefore, we only count operations under the assumption that Eq. (2.13) has real solutions. In this case, the computational costs for Algorithm 2.5 are $29n + 94$, since costs for the steps 2, 3, and 4 are 109, 11, and $29n - 26$, respectively. For convenience, we assume that the square (or cube) root evaluation is of the same complexity as the other basic arithmetical operations.

Below, we compare the computational costs among Jia, Kong and Sogabe's algorithm (Algorithm 2.1), Navon's algorithm (Algorithm 2.2) and our algorithm (Algorithm 2.5) in Table 2.

Comparing the results given in Table 2, we can see that our algorithm reduces the computational costs by using less number of arithmetic operations.

2.4 A symmetric case

In this section, we consider a symmetric case of the CP linear system $A\mathbf{x} = \mathbf{f}$, where

$$A := \begin{bmatrix} d & a & 1 & & & 1 & a \\ a & d & a & 1 & & & 1 \\ 1 & a & d & a & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ 1 & & & 1 & a & d & a \\ a & 1 & & & 1 & a & d \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{2.18}$$

and matrix A is strictly diagonally dominant.

Table 2 Total operations for solving the circulant pentadiagonal Toeplitz linear systems under the assumption that Eq. (2.13) has real solutions and the symbolic name τ does not emerge

	Algorithm 2.1 [26]	Algorithm 2.2 [27]	Our algorithm
Operations	$48n - 24$	$41n - 29$	$29n + 94$

As a generalization of the symmetric pentadiagonal Toeplitz (SPT) linear systems [29,30], symmetric circulant pentadiagonal (SCP) linear systems often appear in many applications. For details, the interested reader may refer to paper [31] and the references therein.

From the results given in Sect. 2.2, we note that the SCP matrix A as in (2.18) can be factorized into a product of two n-by-n symmetric circulant tridiagonal matrices, i.e., $A = T_1 T_2$, where

$$T_1 := \begin{bmatrix} s & 1 & & & 1 \\ 1 & s & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & 1 & s \end{bmatrix}, \quad T_2 := \begin{bmatrix} t & 1 & & & 1 \\ 1 & t & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & 1 & t \end{bmatrix}.$$

And, the parameters s and t satisfy the following equations

$$\begin{cases} s + t = a, \\ st + 2 = d. \end{cases}$$

It is easy to deduce that

$$s_{1,2} = \frac{a \pm \sqrt{a^2 - 4d + 8}}{2}, \quad t_{1,2} = \frac{a \mp \sqrt{a^2 - 4d + 8}}{2}.$$

Below, we state an algorithm for solving the SCP linear system.

Algorithm 2.6

```
Step 1 Input $a, d, n$ and $\mathbf{f}$.
Step 2 Compute $s = \frac{a+\text{sign}(a)\sqrt{a^2-4d+8}}{2}$, $t = \frac{a-\text{sign}(a)\sqrt{a^2-4d+8}}{2}$.
Step 3 Solve systems $T_1\mathbf{y} = \mathbf{f}$ and $T_2\mathbf{x} = \mathbf{y}$ by using Algorithm 2.4 (or any circulant tridiagonal linear solver).
Step 4 Evaluate $\mathbf{x}$ with $\tau = 0$.
Step 5 Output the solution of system: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$.
```

The following proposition shows that under certain assumptions the above algorithm is feasible and stable.

Proposition 2.3 *Let the scalar a and b be such that*

$$|d| > 2|a| + 2, \tag{2.19}$$

and

$$a^2 - 4d + 8 > 0, \tag{2.20}$$

then

$$|s_{1,2}| = \frac{|a \pm \sqrt{a^2 - 4d + 8}|}{2} > 2.$$

Proof **Case (i)** $d < 0$.
From Eq.(2.19), we get $a^2 - 4d + 8 > |a|^2 + 8|a| + 16 = (|a| + 4)^2$.
Therefore, $\sqrt{a^2 - 4d + 8} > |a| + 4$.
Subcase (i) $a > 0$.
Since $a + |a| = 2a > 0$, we get $s_1 = \frac{a+\sqrt{a^2-4d+8}}{2} > \frac{a+(|a|+4)}{2} > 2$.
Since $a - |a| = 0$, we get $s_2 = \frac{a-\sqrt{a^2-4d+8}}{2} < \frac{a-(|a|+4)}{2} = -2$.
Subcase (ii) $a < 0$.
Since $a + |a| = 0$, we get $s_1 = \frac{a+\sqrt{a^2-4d+8}}{2} > \frac{a+(|a|+4)}{2} = 2$.
Since $a - |a| = 2a < 0$, we get $s_2 = \frac{a-\sqrt{a^2-4d+8}}{2} < \frac{a-(|a|+4)}{2} < -2$.
Case (ii) $d > 0$.
Using conditions (2.19) and (2.20), we get $2|a| + 2 < d < \frac{a^2+8}{4}$.
Therefore, $|a| > 8$. Thus $a^2 - 4d + 8 < |a|^2 - 8|a| < (|a| - 4)^2$. This means that $\sqrt{a^2 - 4d + 8} < |(|a| - 4)| = |a| - 4$.
Subcase (i) $a > 0$.
$s_2 = \frac{a-\sqrt{a^2-4d+8}}{2} > \frac{a-(|a|-4)}{2} = 2$, since $s_1 > s_2$, we get $s_1 > 2$.
Subcase (ii) $a < 0$.
$s_1 = \frac{a+\sqrt{a^2-4d+8}}{2} < \frac{a+(|a|-4)}{2} = -2$, since $s_2 < s_1$, we get $s_2 < -2$.
Thus by combining all cases, we have $|s_{1,2}| > 2$. □

The feasibility and stability of Algorithm 2.6 depends on the third step that solving the circulant tridiagonal Toeplitz linear systems $T_1\mathbf{y} = \mathbf{f}$ and $T_2\mathbf{x} = \mathbf{y}$. Since the coefficient matrix A in (2.18) is strictly diagonally dominant, we have $|d| > 2|a| + 2$. With the restriction that $a^2 - 4d + 8 > 0$, it follows from Proposition 2.3 that we can obtain $|s| > 2$ and $|t| > 2$, i.e., matrices T_1 and T_2 are both strictly diagonally dominant. This property guarantees the numerical stability of the process of solving $T_1\mathbf{y} = \mathbf{f}$ and $T_2\mathbf{x} = \mathbf{y}$, see [32].

3 Numerical experiments

In this section, we present some numerical results to demonstrate the performance and effectiveness of the proposed algorithm. Meanwhile, we compare our algorithm (Algorithm 2.5) with Jia, Kong and Sogeba's algorithm [26], Navon's algorithm [27], and also with Matlab back-slash operator. All numerical experiments were performed in MATLAB 7 environment (using double precision arithmetic) and run on a HP Compaq 6280 Pro Microtower PC with Inter (R) Core (TM) i5-2400 processor.

Example 3.1 In this example, we consider an n-by-n nonsymmetric CP linear system $A\mathbf{x} = \mathbf{f}$ given by

Table 3 Numerical results of Example 3.1

Algorithms	n	2500	5000	7500	10,000
Algorithm 2.1	$\Vert\mathbf{x}-\hat{\mathbf{x}}\Vert_2$	1.0886e−014	1.5549e−014	1.9106e−014	2.2097e−014
	CPU time (s)	0.0933	0.1254	0.1672	0.2186
Algorithm 2.2	$\Vert\mathbf{x}-\hat{\mathbf{x}}\Vert_2$	1.0908e−014	1.5565e−014	1.9118e−014	2.2108e−014
	CPU time (s)	0.0294	0.0610	0.1075	0.1539
Back-slash (\)	$\Vert\mathbf{x}-\hat{\mathbf{x}}\Vert_2$	6.2705e−015	8.8198e−015	1.0785e−014	1.2445e−014
	CPU time (s)	0.0323	0.0589	0.0961	0.1318
Our algorithm	$\Vert\mathbf{x}-\hat{\mathbf{x}}\Vert_2$	5.5942e−015	7.8810e−015	9.6398e−015	1.1124e−014
	CPU time (s)	0.0155	0.0467	0.0623	0.0780

$$\begin{bmatrix} 4 & -2 & -2 & & & 0.5 & 3 \\ 3 & 4 & -2 & -2 & & & 0.5 \\ 0.5 & 3 & 4 & -2 & -2 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & -2 \\ -2 & & & 0.5 & 3 & 4 & -2 \\ -2 & -2 & & & 0.5 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 3.5 \\ 3.5 \\ 3.5 \\ \vdots \\ \vdots \\ 3.5 \\ 3.5 \end{bmatrix}.$$

It can be readily verified that $\hat{\mathbf{x}} = [1, 1, \ldots, 1]^T$ is the exact solution of the above system. Below, the results (absolute errors $\Vert\mathbf{x}-\hat{\mathbf{x}}\Vert_2$ and CPU times) with Algorithm 2.1, Algorithm 2.2, Algorithm 2.5 (with $p = 1$), and Matlab back-slash operator are shown in Table 3.

We can see from Table 3 that our algorithm slightly outperforms other algorithms for every dimension n. In the next example, another CP linear system which originates from an ordinary differential equation subject to periodic boundary conditions will be used to further illustrate the competitiveness of our algorithm.

Example 3.2 Consider the following linear ordinary differential equation

$$f''(x) + f(x) = \left(1 - 4\pi^2\right)\sin(2\pi x),\ \ x \in [0, 1].$$

Subject to periodic boundary conditions

$$f(0) = f(1),\ \ f'(0) = f'(1),$$

where $h = \Delta x$ is the mesh width and $h = \frac{1}{n}$. It is easy to see that the exact solution of the above equation is

$$f(x) = \sin(2\pi x).$$

Table 4 Numerical results of Example 3.2

Algorithms	n	1000	2500	5000	10,000
Algorithm 2.1	$\|\mathbf{x}-\tilde{\mathbf{x}}\|_2$	7.6887e−010	4.4635e−009	1.0199e−008	2.6550e−008
	CPU time (s)	0.0622	0.2035	0.3143	0.4856
Algorithm 2.2	$\|\mathbf{x}-\tilde{\mathbf{x}}\|_2$	7.7053e−010	4.4711e−009	1.0067e−008	2.6624e−008
	CPU time (s)	0.0320	0.0671	0.1090	0.1872
Back-slash (\)	$\|\mathbf{x}-\tilde{\mathbf{x}}\|_2$	4.8784e−010	9.2087e−010	1.8717e−009	1.1550e−008
	CPU time (s)	0.0324	0.0665	0.1019	0.1538
Our algorithm	$\|\mathbf{x}-\tilde{\mathbf{x}}\|_2$	2.1258e−010	4.6566e−010	9.9774e−010	1.8397e−009
	CPU time (s)	0.0261	0.0577	0.0873	0.1224

If we use a fourth order finite difference to approximate the second derivative, i.e.,

$$f''(x) \approx \frac{1}{12h^2}(-f(x-2h)+16f(x-h)-30f(x)+16f(x+h)-f(x+2h)),$$

then the original ordinary equation can be discretized as the following n-by-n CP linear system

$$\begin{bmatrix} d & a & b & & & b & a \\ a & d & a & b & & & b \\ b & a & d & a & b & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & b \\ b & & & b & a & d & a \\ a & b & & & b & a & d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix},$$

where $d = -30 + 12h^2$, $a = 16$, $b = -1$, and $f_i = 12h^2(1-4\pi^2)\sin(2\pi(i-1)h)$ for $i = 1, 2, \ldots, n$. Here, we also use Algorithm 2.1, Algorithm 2.2, Algorithm 2.5, and Matlab back-slash operator to compute the solution $\mathbf{x}$. The errors $\|\mathbf{x}-\tilde{\mathbf{x}}\|_2$ ($\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n]^T$, $\tilde{x}_i = \sin(2\pi(i-1)h)$) and the mean value of CPU times (after 100 tests) are shown in Table 4.

Similar to the results shown in Example 3.1, our algorithm slight outperforms other existing algorithms for every dimension.

4 Concluding remarks

In this paper, we mainly considered the solution of an n-by-n circulant pentadiagonal Toeplitz linear system (1.1). First, two existing algorithms for solving CP linear systems were reviewed. Then, we developed a symbolic algorithm (Algorithm 2.5) for robust computation which is based on factoring the coefficient matrix into the product of two circulant tridiagonal Toeplitz matrices. Also, we showed that the computational costs of our proposed algorithm is much less than those of other existing algorithms, i.e. Jia, Kong and Sogabe's algorithm [26] and Navon's algorithm [27]. Moreover, a symmetric case of the CP linear systems was considered. Finally, some numerical examples were given in order to demonstrate the performance and efficiency of our algorithm.

Acknowledgments The authors would like to thank the anonymous referees whose comments substantially enhanced the quality of the paper.

References

1. V. Alexandrov, W. Owczarz, P.G. Thomsen, Z. Zlatev, Math. Comput. Simul. **65**, 557 (2004)
2. I. Faragó, K. Georgiev, Á. Havasi, Z. Zlatev, Comput. Math. Appl. **67**, 2085 (2014)
3. A. Bjorck, G.H. Golub, SIAM Rev. **19**, 5 (1977)
4. S.S. Nemani, L.E. Garey, Int. J. Comput. Math. **79**, 1001 (2002)
5. J. Monterde, H. Ugail, Comput. Aided Geom. Design. **23**, 208 (2006)
6. F. Iachello, A. Del Sol Mesa, J. Math. Chem. **25**, 345 (1999)
7. M.E. Kanal, J. Supercomput. **59**, 1071 (2012)
8. S. Barnett, *Matrices: Methods and Applications* (Oxford University Press, New York, 1990)
9. M.B. Allen III, E.L. Isaacson, *Numerical Analysis for Applied Science* (Wiley, New Jersey, 1997)
10. N.W. Loney, *Applied Mathematical Methods for Chemical Engineers*, 2nd edn. (CRC Press, New York, 2000)
11. Arshad Khan, Islam Khan, Tariq Aziz, Appl. Math. Comput. **181**, 432 (2006)
12. M.E. Kanal, N.A. Baykara, M. Demiralp, J. Math. Chem. **50**, 289 (2012)
13. T.A. Davis, *Direct Methods for Sparse Linear Systems* (SIAM, Philadelphia, 2006)
14. L.E. Garey, R.E. Shaw, Int. J. Math. Comput. **42**, 1 (2001)
15. L.E. Garey, R.E. Shaw, J. Appl. Math. Comput. **100**, 241 (1999)
16. M. El-Mikkawy, E. Rahmo, Comput. Math. Appl. **59**, 1386 (2010)
17. J.T. Jia, T. Sogabe, J. Math. Chem. **51**, 881 (2013)
18. J.T. Jia, Y.L. Jiang, Numer. Algor. **63**, 357 (2013)
19. B.V. Minchev, I.G. Ivanov, LSSC. (2003). doi:10.1007/978-3-540-24588-9_55
20. M. Batista, A.A. Karawia, Appl. Math. Comput. **210**, 558 (2009)
21. J.T. Jia, Y.L. Jiang, Comput. Math. Appl. **66**, 965 (2013)
22. T. Sogabe, Appl. Math. Comput. **202**, 850 (2008)
23. A.A. Karawia, Appl. Math. Comput. **174**, 613 (2006)
24. M. El-Mikkawy, E. Rahmo, Appl. Math. Comput. **207**, 164 (2009)
25. J.T. Jia, Q.X. Kong, J. Math. Chem. **52**, 2222 (2014)
26. J.T. Jia, Q.X. Kong, T. Sogabe, Comput. Math. Appl. **63**, 1238 (2012)
27. I.M. Navon, Commun. Appl. Numer. Methods **3**, 63 (1987)
28. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, 1996)
29. J.M. McNally, J. Comput. Appl. Math. **234**, 995 (2010)
30. S.S. Nemani, Appl. Math. Comput. **215**, 3830 (2010)
31. S.M. El-Sayed, I.G. Ivanov, M.G. Petkov, Comput. Math. Appl. **35**, 35 (1998)
32. Q. Li, N.C. Wang, Mini-Micro System **23**, 1393 (2002)