J. Korean Math. Soc. **59** (2022), No. 5, pp. 891–909
https://doi.org/10.4134/JKMS.j210701
pISSN: 0304-9914 / eISSN: 2234-3008

# THOMAS ALGORITHMS FOR SYSTEMS OF FOURTH-ORDER FINITE DIFFERENCE METHODS

Soyoon Bak, Philsu Kim, and Sangbeom Park

Abstract. The main objective of this paper is to develop a concrete inverse formula of the system induced by the fourth-order finite difference method for two-point boundary value problems with Robin boundary conditions. This inverse formula facilitates to make a fast algorithm for solving the problems. Our numerical results show the efficiency and accuracy of the proposed method, which is implemented by the Thomas algorithm.

## 1. Introduction

Finite difference method is one of the popular numerical methods to solve various ordinary and partial differential equations. It goes without saying that simple inverse formulas for finite difference matrices can be of great importance in developing efficient numerical algorithms. When the second-order central finite difference method is applied in a differential equation involving the second-order differential, the resulting in a system of equations is generally governed by a tridiagonal matrix. The analysis of the typical system of tridiagonal Jacobi matrices is well known, and Usmani [6] created its inverse formula represented by the principal minors in 1994. Further, the tridiagonal system can be solved numerically and quickly by the familiar Thomas algorithm [5].

The implementation of an approximate solution with high accuracy is of great important in numerical analysis, and one can find many finite difference methods with high accuracy in various problems. In spite of these many research results and their significance, to the best of the author's knowledge, few studies have focused on inversion formulas and fast numerical solvers for systems dominated by the standard higher-order finite difference matrices. The main purpose of this paper is to develop inverse formulas of the systems of

Received November 19, 2021; Revised April 26, 2022; Accepted June 15, 2022.
2020 *Mathematics Subject Classification.* 65N06, 65F15, 65F50.
*Key words and phrases.* Finite difference method, inverse of tridiagonal matrix, Thomas algorithm, boundary value problem.

equations governed by the fourth-order finite difference matrix of size $N$

$$\mathcal{D}_4 = \frac{1}{12h^2}\begin{bmatrix} -15 & -4 & 14 & -6 & 1 & & & & \\ 16 & -30 & 16 & -1 & & & & & \\ -1 & 16 & -30 & 16 & -1 & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & -1 & 16 & -30 & 16 & -1 & & \\ & & & -1 & 16 & -30 & 16 & -1 \\ & & & & -1 & 16 & -30 & 16 \\ & & & 1 & -6 & 14 & -4 & -15 \end{bmatrix}.$$

In addition, we find a fast solver for the systems of equations which consist of only the Thomas algorithm.

To achieve the purpose, we first provide an extended version of the formula for calculating the finite difference weights introduced by Fornberg [1]. Using this formula, we find a decomposition of $\mathcal{D}_4$ represented by

$$\mathcal{D}_4 = \mathcal{A}_2\mathcal{D}_2$$

and make an explicit inverse formula of $\mathcal{D}_4$ based on the inverse formula for the quasi-tridiagonal matrix $\mathcal{A}_2$ made by the results of Usmani [6] and the well-known Sherman-Morrison formula [3]. Here, $\mathcal{D}_2$ is the standard tridiagonal matrix made by the standard central finite difference matrix for the second derivative. It is also shown that the inverse formula enables us to make a fast Thomas algorithm for solving the system $\mathcal{D}_4\mathbf{u} = \mathbf{f}$. We extend these results to find both the inverse formula and the Thomas algorithm for the system governed by $\mathcal{D}_4$ that is induced from the fourth-order finite difference method for solving two-point boundary value problems with Robin boundary conditions.

The rest of the paper is structured as follows. In Section 2, the relation between a higher-order finite difference formula and the second-order one is derived. In addition, the explicit inverse formula of quasi-tridiagonal matrix $\mathcal{A}_2$ is derived. In Section 3, we extend the results to the system for two-point boundary value problems with Robin boundary conditions. Finally, in Section 4, we provide numerical results as evidence of the proposed fast algorithm based on the Thomas algorithm.

In the following discussion, we will use the formula for the inverse of a tridiagonal Jacobi matrix [6]. The inversion formula is described as follows. Let $A_n = [a_{ij}]$ be an $n \times n$ tridiagonal Jacobi matrix such that

$$a_{ij} = \begin{cases} b_i, & i = j, \\ c_i, & j = i+1, \\ a_i, & j = i-1, \\ 0, & |i-j| > 1. \end{cases}$$

**Theorem 1.1.** *Let $A_n^{-1} = [\alpha_{ij}]$. Then, each entry $\alpha_{ij}$ satisfies* [6]

$$\alpha_{ij} = \frac{(-1)^{i+j}}{\theta_n} \begin{cases} c_i c_{i+1} \cdots c_{j-1} \theta_{i-1} \phi_{j+1}, & i < j, \\ \theta_{i-1}\phi_{i+1}, & i = j, \\ a_{j+1}a_{j+2} \cdots a_i \theta_{j-1}\phi_{i+1}, & i > j, \end{cases}$$

*where $\{\phi_i\}$ is a sequence satisfying the recurrence formula*

$$\phi_i = b_i \phi_{i+1} - c_i a_{i+1} \phi_{i+2}, \quad i = n, n-1, \ldots, 2, 1,$$
$$\phi_{n+1} = 1, \quad \phi_{n+2} = 0,$$

*and $\theta_i$ is the principal minors satisfying*

$$\theta_i = b_i \theta_{i-1} - a_i c_{i-1} \theta_{i-2}, \quad i = 1, 2, \ldots, n.$$

*Here, $\theta_n = \det\big(A_n\big)$, $\theta_{-1} = 0$, and $\theta_0 = 1$, where $\det A_n$ denotes the determinant of the matrix $A_n$.*

## 2. Decomposition of $\mathcal{D}_4$ and its explicit inverse formula

The aim of this section is to derive the relation between the fourth-order and second-order finite differences for the $k^{th}$-derivative. To do this, let us introduce the uniform grid defined by

$$a = x_0 < x_1 < x_2 < \cdots < x_n = b, \quad x_i = x_0 + ih, \quad h = \tfrac{b-a}{n}.$$

We first recall that the Newton interpolation polynomial of degree $n$, that interpolates $f_i = f(x_i)$ $(i = 0, 1, \ldots, n)$, is given by

$$L_n(x) = f_0 + \sum_{i=1}^{n} f[x_0, \ldots, x_i] l_i(x),$$

where $f[x_0, \ldots, x_i]$ and $l_i(x)$ are the $i^{th}$-order Newton divided difference of $f$ and the Newton basis polynomial, respectively, defined by

$$f[x_i, \ldots, x_j] = \begin{cases} f_i, & i = j, \\ \frac{f[x_{i+1},\ldots,x_j] - f[x_i,\ldots,x_{j-1}]}{x_j - x_i}, & i < j, \end{cases} \qquad l_i(x) = \prod_{j=0}^{i} (x - x_j).$$

The following lemma gives a special form of the differentiation of $L_n(x)$ that will be used in deriving a decomposition of the finite difference matrix.

**Lemma 2.1.** *For a fixed positive integer $k$, if we write the $k^{th}$-derivative $L_n^{(k)}(x_j)$ of $L_n(x)$ as*

$$(1)\quad L_n^{(k)}(x_j) = \sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} k_0! f[x_{i-k_0}, \ldots, x_i], \quad j = 0, 1, 2, \ldots, n, \ 0 \le k_0 \le k,$$

*then, it can be seen that coefficients $w_{i-k_0}^{(k_0,j)}$ are implicitly defined by the polynomial*

$$\sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} y^{i-k_0} = h^{k_0-k} y^j \sum_{m=0}^{n-j-k_0} \frac{G^{(m)}(1)}{m!}(y-1)^m,$$

*where $G^{(m)}$ is $m^{th}$-derivative of a smooth function defined by*

$$G(y) := \frac{\Big(\ln y\Big)^k}{(y-1)^{k_0}}.$$

*Proof.* We first note that

$$\begin{aligned} f^{(k)}(x_j) &= L_n^{(k)}(x_j) + \mathcal{O}(h^{n-k+1}), \\ k_0! f[x_{i-k_0}, \dots, x_i] &= \frac{1}{h^{k_0}} \sum_{l=0}^{k_0} \binom{k_0}{l} (-1)^l f_{i-l}. \end{aligned} \tag{2}$$

Now, using the function $f(x) = e^{\mathbf{i}wx}$, we combine (1) and (2) to get

$$\begin{aligned} &(\mathbf{i}w)^k e^{\mathbf{i}wjh} e^{\mathbf{i}x_0} \\ &= \frac{1}{h^{k_0}} \sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} \sum_{l=0}^{k_0} \binom{k_0}{l} (-1)^l e^{\mathbf{i}w(i-l)h} e^{\mathbf{i}x_0} + \mathcal{O}(h^{n-k+1}) \\ &= \frac{1}{h^{k_0}} \sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} e^{\mathbf{i}w(i-k_0)h} \sum_{l=0}^{k_0} \binom{k_0}{l} (-1)^l e^{\mathbf{i}w(k_0-l)h} e^{\mathbf{i}x_0} + \mathcal{O}(h^{n-k+1}). \end{aligned}$$

Cancelling the fact $e^{\mathbf{i}x_0}$, and substituting $e^{\mathbf{i}wh} = y$, i.e., $\mathbf{i}wh = \ln y$, gives

$$h^{k_0-k} y^j (\ln y)^k = \sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} y^{i-k_0} (y-1)^{k_0} + \mathcal{O}(h^{n-k+1}).$$

Hence, the coefficient $w_{i-k_0}^{(k_0,j)}$ can be calculated with the equation

$$h^{k_0-k} y^j \frac{(\ln y)^k}{(y-1)^{k_0}} = \sum_{i=k_0}^{n} w_{i-k_0}^{(k_0,j)} y^{i-k_0} + \mathcal{O}(h^{n-k+1}).$$

Expanding $\ln y$ around $y = 1$ and bouncing the maximum degree of $y$, the proof is completed. □

*Remark* 2.2. When $k_0 = 0$, the formula of (1) is exactly same with the formula calculated by Fornberg [1]. Hence, the formula (1) is an extension of the formula in [1].

In the following, we give two special cases for calculating the weights $w_{i-k_0}^{(k_0,j)}$ by using a single line of Mathematica code:

```
CoefficientList[Normal[Series[y^{j}*Log[y]^{k}/(y-1)^{k0} h^{k0-k},{y,1,n-k0}]],y]
```

In the first case, we present the relation between higher-order and lower-order approximation of the first derivative. For $n = 4$, the fourth-order approximations of the first derivative at the grids $x_i$ $(i = 0, 1, \dots, 4)$ are defined as follows:

$$\begin{aligned} f'(x_j) &= \sum_{i=0}^{4} w_i^{(0,j)} f_i + \mathcal{O}(h^4) \\ &= \sum_{i=0}^{3} w_i^{(1,j)} f[x_i, x_{i+1}] + \mathcal{O}(h^4), \quad j = 0,1,2,3,4, \end{aligned} \tag{3}$$

where

$$\begin{aligned} w_i^{(0,0)} &= \frac{\{-25, 48, -36, 16, -3\}}{12h}, & w_i^{(1,0)} &= \frac{\{25, -23, 13, -3\}}{12}, \\ w_i^{(0,1)} &= \frac{\{-3, -10, 18, -6, 1\}}{12h}, & w_i^{(1,1)} &= \frac{\{3, 13, -5, 1\}}{12}, \\ w_i^{(0,2)} &= \frac{\{1, -8, 0, 8, -1\}}{12h}, & w_i^{(1,2)} &= \frac{\{-1, 7, 7, -1\}}{12}, \\ w_i^{(0,3)} &= \frac{\{-1, 6, -18, 10, 3\}}{12h}, & w_i^{(1,3)} &= \frac{\{1, -5, 13, 3\}}{12}, \\ w_i^{(0,4)} &= \frac{\{3, -16, 36, -48, 25\}}{12h}, & w_i^{(1,4)} &= \frac{\{-3, 13, -23, 25\}}{12}. \end{aligned}$$

The second case shows the relation between the fourth-order and second-order approximation of the second derivative. For $n = 5$, the fourth-order approximations of the second derivative at the grids $x_i$ ($i = 0, 1, \ldots, 5$) are defined as follows:

$$\begin{aligned} f''(x_j) &= \sum_{i=0}^{5} w_i^{(0,j)} f_i + \mathcal{O}(h^4) \\ &= \sum_{i=0}^{3} w_i^{(2,j)} 2f[x_i, x_{i+1}, x_{i+2}] + \mathcal{O}(h^4), \; j = 0,1,2,3,4,5, \end{aligned} \tag{4}$$

where

$$\begin{aligned} w_i^{(0,0)} &= \frac{\{45, -154, 214, -156, 61, -10\}}{12h^2}, & w_i^{(2,0)} &= \frac{\{45, -64, 41, -10\}}{12}, \\ w_i^{(0,1)} &= \frac{\{10, -15, -4, 14, -6, 1\}}{12h^2}, & w_i^{(2,1)} &= \frac{\{10, 5, -4, 1\}}{12}, \\ w_i^{(0,2)} &= \frac{\{-1, 16, -30, 16, -1\}}{12h^2}, & w_i^{(2,2)} &= \frac{\{-1, 14, -1, 0\}}{12}, \\ w_i^{(0,3)} &= \frac{\{0, -1, 16, -30, 16, -1\}}{12h^2}, & w_i^{(2,3)} &= \frac{\{0, -1, 14, -1\}}{12}, \\ w_i^{(0,4)} &= \frac{\{1, -6, 14, -4, -15, 10\}}{12h^2}, & w_i^{(2,4)} &= \frac{\{1, -4, 5, 10\}}{12}, \\ w_i^{(0,5)} &= \frac{\{-10, 61, -156, 214, -154, 45\}}{12h^2}, & w_i^{(2,5)} &= \frac{\{-10, 41, -64, 45\}}{12}. \end{aligned}$$

If we introduce the fourth-order difference matrices of size $N := n-1$, then the relation between coefficients in (4) gives the following decomposition of $\mathcal{D}_4$:

$$\mathcal{D}_4 = \mathcal{A}_2\mathcal{D}_2, \tag{5}$$

where both matrices $\mathcal{D}_2$ and $\mathcal{A}_2$ are of order $N$ defined by

$$\mathcal{D}_2 = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}, \quad \mathcal{A}_2 = \frac{1}{12}\begin{bmatrix} 10 & 5 & -4 & 1 & \\ -1 & 14 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 14 & -1 \\ & 1 & -4 & 5 & 10 \end{bmatrix}.$$

Let us introduce the matrices $\mathbf{q}$, $\mathbf{p}$, and $\mathcal{B}_2$ of sizes $N\times 2$, $N\times 2$, and $N\times N$ whose $(i,j)$th-entries are defined by

$$\Big(\mathcal{B}_2\Big)_{i,j} = \frac{1}{12}\begin{cases} 14 & i=j, \\ -1 & |i-j|=1, \\ 0 & \text{otherwise}, \end{cases} \qquad \Big(\mathbf{q}\Big)_{i,j} = \frac{1}{12}\begin{cases} 1 & (i,j)=(1,1),(N,2), \\ 0 & \text{otherwise}, \end{cases}$$

$$\Big(\mathbf{p}\Big)_{i,j} = \begin{cases} -4 & (i,j)=(1,1),(3,1),(N,2),(N-2,2), \\ 6 & (i,j)=(2,1),(N-1,2), \\ 1 & (i,j)=(4,1),(N-3,2), \\ 0 & \text{otherwise}, \end{cases}$$

respectively. Also, let $\mathbf{e}_j$ be the canonical unit vector of size $N$ with 1 in $j$th entry and zeros elsewhere. Then, we can see that

$$\begin{bmatrix} -4 & 6 & -4 & 1 & \\ 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 \\ & 1 & -4 & 6 & -4 \end{bmatrix} = \begin{aligned} &\mathbf{e}_1(-4\mathbf{e}_1 + 6\mathbf{e}_2 - 4\mathbf{e}_3 + \mathbf{e}_4)^T \\ &\quad + \mathbf{e}_N(-4\mathbf{e}_N + 6\mathbf{e}_{N-1} - 4\mathbf{e}_{N-2} + \mathbf{e}_{N-3})^T. \end{aligned}$$

Eventually, the quasi-tridiagonal matrix $\mathcal{A}_2$ is split into

$$\begin{aligned} \mathcal{A}_2 &= \frac{1}{12}\begin{bmatrix} 14 & -1 & 0 & 0 & \\ -1 & 14 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 14 & -1 \\ & 0 & 0 & -1 & 14 \end{bmatrix} + \frac{1}{12}\begin{bmatrix} -4 & 6 & -4 & 1 & \\ 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 \\ & 1 & -4 & 6 & -4 \end{bmatrix}, \\ &= \mathcal{B}_2 + \mathbf{q}\mathbf{p}^T. \end{aligned} \tag{6}$$

The tridiagonal matrix $\mathcal{B}_2$ is then symmetric diagonally dominant and hence invertible. In particular, the following theorem shows the invertibility of the $2\times 2$ matrix $\mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q}$.

**Theorem 2.3.** *The matrix* $\mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q}$ *has the explicit form*

$$\mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q} = \frac{144\lambda^{2N}}{\lambda^{2N+2}-1}\begin{bmatrix} 1-\lambda^{-4N} & 28\sqrt{48}\lambda^{1-N} \\ 28\sqrt{48}\lambda^{1-N} & 1-\lambda^{-4N}\end{bmatrix}, \quad \lambda = 7+\sqrt{48}.$$

*Proof.* If we let $\mathbf{r} = \mathcal{B}_2^{-1}\mathbf{q}$, then a direct calculation from the definition of $\mathbf{p}^{\mathrm{T}}$ gives

$$\begin{aligned} &\mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q} \\ &= \mathcal{I}_2 + \mathbf{p}^T\mathbf{r} \\ &= \begin{bmatrix} 1-4r_{1,1}+6r_{2,1}-4r_{3,1}+r_{4,1} & -4r_{1,2}+6r_{2,2}-4r_{3,2}+r_{4,2} \\ -4r_{N,1}+6r_{N-1,1}-4r_{N-2,1}+r_{N-3,1} & 1-4r_{N,2}+6r_{N-1,2}-4r_{N-2,2}+r_{N-3,2}\end{bmatrix}. \end{aligned}$$

Since the matrix $\mathbf{r}$ is the solution of the system $\mathcal{B}_2\mathbf{r} = \mathbf{q}$, the definition of $\mathbf{q}$ gives

$$(7)\qquad \left\{\begin{aligned} 14r_{1,1}-r_{2,1} &= 1, \\ -r_{1,1}+14r_{2,1}-r_{3,1} &= 0, \\ &\vdots \\ -r_{N-2,1}+14r_{N-1,1}-r_{N,1} &= 0, \\ -r_{N-1,1}+14r_{N,1} &= 0, \end{aligned}\right. \qquad \left\{\begin{aligned} 14r_{1,2}-r_{2,2} &= 0, \\ -r_{1,2}+14r_{2,2}-r_{3,2} &= 0, \\ &\vdots \\ -r_{N-2,2}+14r_{N-1,2}-r_{N,2} &= 0, \\ -r_{N-1,2}+14r_{N,2} &= 1. \end{aligned}\right.$$

From the left system of equations of (7), we simplify the $(1,1)$-entry of $\mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q}$ as

$$\begin{aligned} &1-4r_{1,1}+6r_{2,1}-4r_{3,1}+r_{4,1} \\ &= 1-4r_{1,1}+5r_{2,1}+10r_{3,1}+\gamma_1 \quad (\gamma_1 := r_{2,1}-14r_{3,1}+r_{4,1}=0) \\ &= 1-14r_{1,1}+145r_{2,1}+\gamma_2 \quad (\gamma_2 := 10(r_{1,1}-14r_{2,1}+r_{3,1})=0) \\ &= 144r_{2,1}. \end{aligned}$$

From similar procedures, the matrix of (6) can be simplified by

$$(8)\qquad \mathcal{I}_2 + \mathbf{p}^T\mathcal{B}_2^{-1}\mathbf{q} = \begin{bmatrix} 144r_{2,1} & 144r_{2,2} \\ 144r_{N-1,1} & 144r_{N-1,2}\end{bmatrix}.$$

According to the explicit formula for the inverse of the tridiagonal matrix given by Theorem 1.1, the two systems of (7) show that

$$(9)\qquad r_{2,1} = r_{N-1,2} = \phi_3/\theta_N, \quad r_{2,2} = r_{N-1,1} = 14/\theta_N,$$

where $\phi_i$ and $\theta_i$ are, respectively, the solutions of the three-term recurrence equations:

$$(10)\qquad \begin{aligned} &\theta_i = 14\theta_{i-1}-\theta_{i-2}, \quad \theta_0 = 1, \quad \theta_1 = 14, \\ &\phi_i = 14\phi_{i+1}-\phi_{i+2}, \quad \phi_{N+1} = 1, \quad \phi_N = 14. \end{aligned}$$

The solutions of (10) can be easily obtained using the standard methods, which are given by

$$(11)\qquad \theta_i = \phi_{N+1-i} = \frac{1}{2\sqrt{48}}\Big(\lambda^{i+1} - \Big(\frac{1}{\lambda}\Big)^{i+1}\Big), \quad i = 0,1,\ldots,N.$$

Hence, the proof can be completed using (8), (9), and (11). □

*Remark* 2.4. Since $\lambda^N \to \infty$ as $N \to \infty$, Theorem 2.3 shows that for sufficiently large $N$, the matrix $\mathcal{I}_2 + \mathbf{p}^T \mathcal{B}_2^{-1} \mathbf{q}$ is almost a scalar matrix. More precisely, we have

$$\mathcal{I}_2 + \mathbf{p}^T \mathcal{B}_2^{-1} \mathbf{q} = \frac{144}{97 + 56\sqrt{3}} \mathcal{I}_2 + \mathcal{T}, \quad \|\mathcal{T}\|_\infty = 9.99 \times 10^{-16}$$

for $N \geq 16$.

Remark 2.4 guarantees that the quasi-tridiagonal matrix $\mathcal{A}_2$ has an explicit inverse formula, which can be proved by the Sherman-Morrison formula [3].

**Corollary 1.** *The matrix $\mathcal{A}_2$ of* (6) *has an explicit inverse formula* [3]

$$\mathcal{A}_2^{-1} = \Big(\mathcal{I} - \mathcal{K}\Big) \mathcal{B}_2^{-1}, \tag{12}$$

*where $\mathcal{K}$ is a known matrix defined by*

$$\mathcal{K} := \mathcal{B}_2^{-1} \mathbf{q} \Big(\mathcal{I}_2 + \mathbf{p}^T \mathcal{B}_2^{-1} \mathbf{q}\Big)^{-1} \mathbf{p}^T.$$

Summarizing, if we apply the fourth-order finite difference method to the basic two-point boundary value problem with the Dirichlet boundary condition

$$u''(x) = f(x), \quad x \in (a,b); \quad u(a) = g_1, \quad u(b) = g_2, \tag{13}$$

the formulas (5) and (12) show that we can obtain the numerical solution by solving only two tridiagonal systems

$$\mathcal{B}_2 \mathbf{x} = \mathbf{f}, \qquad \mathcal{D}_2 \mathbf{u} = \mathbf{x} - \mathbf{r}\Big((\mathcal{I}_2 + \mathbf{p}^T \mathcal{B}_2^{-1} \mathbf{q})^{-1} (\mathbf{p}^T \mathbf{x})\Big), \tag{14}$$

where $\mathbf{r}$ is calculated only once while solving the system. In particular, assuming $N \geq 16$, the numerical solution can be quickly obtained by solving two tridiagonal system:

$$\mathcal{B}_2 \mathbf{x} = \mathbf{f}, \qquad \mathcal{D}_2 \mathbf{u} = \mathbf{x} - \frac{97 + 56\sqrt{3}}{144} \mathbf{r}(\mathbf{p}^T \mathbf{x}). \tag{15}$$

The systems (14) and (15) can be calculated with the Thomas algorithm. This skill will be directly applied the first system in (16) for solving the two-point boundary value problem with the Robin boundary condition.

## 3. Application to two-point boundary value problems

The aim of this section is to extend the discussion presented in Section 2 and derive the Thomas algorithm for solving two-point boundary value problems with Robin boundary condition

$$\begin{aligned} &u''(x) = f(x), \quad x \in (a,b), \\ &\alpha_1 u(a) + \beta_1 u'(a) = g_1, \quad \alpha_2 u(b) + \beta_2 u'(b) = g_2, \end{aligned} \tag{16}$$

where $|\alpha_k| + |\beta_k| \neq 0,\ k = 1, 2$. For the discretization of the Robin boundary condition, we introduce two vectors

$$\begin{aligned}\mathfrak{L} &= \Big[\alpha_1 - \frac{25\beta_1}{12h}, \frac{4\beta_1}{h}, -\frac{3\beta_1}{h}, \frac{4\beta_1}{3h}, -\frac{\beta_1}{4h}\Big],\\ \mathfrak{R} &= \Big[\frac{\beta_2}{4h}, -\frac{4\beta_2}{3h}, \frac{3\beta_2}{h}, -\frac{4\beta_1}{h}, \alpha_2 + \frac{25\beta_2}{12h}\Big].\end{aligned} \tag{17}$$

Then, the vectors of (17) and the fourth-order finite difference scheme for first derivative (3) for the boundary conditions in (16) give

$$g_1 = \sum_{k=1}^{5} \mathfrak{L}_k u_{k-1} + \mathcal{O}(h^4), \quad g_2 = \sum_{k=1}^{5} \mathfrak{R}_k u_{N-4+k} + \mathcal{O}(h^4), \tag{18}$$

where $\mathfrak{L}_k$ and $\mathfrak{R}_k$ are the $k^{th}$-components of $\mathfrak{L}$ and $\mathfrak{R}$, respectively.

Now, we introduce matrices $\mathbf{Q}$, $\mathbf{P}$ of size $N \times 2$, a vector $\mathbf{f}$ of size $N \times 1$ and $2 \times 1$ vector $\mathbf{g}$ defined by

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \\ q_{2,1} & \\ & \\ & \\ & q_{N-1,2} \\ & q_{N,2} \end{bmatrix}_{N\times 2}, \quad \mathbf{P} = \begin{bmatrix} \mathfrak{L}_2 & \\ \vdots & \\ \mathfrak{L}_5 & \\ & \\ & \mathfrak{R}_1 \\ & \vdots \\ & \mathfrak{R}_4 \end{bmatrix}_{N\times 2}, \quad \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}_{N\times 1}, \quad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}_{2\times 1},$$

where

$$[q_{1,1}, q_{2,1}] = \frac{1}{12h^2\mathfrak{L}_1}[-10, 1], \quad [q_{N,2}, q_{N-1,2}] = \frac{1}{12h^2\mathfrak{R}_5}[-10, 1].$$

Then, the fourth-order finite difference schemes of (5) and (18) give a fourth-order discretization system for solving (16) stated as

$$\begin{aligned} &\Big(\mathcal{D}_4 + \mathbf{Q}\mathbf{P}^T\Big)\mathbf{u} = \mathbf{f} + \mathbf{Q}\mathbf{g},\\ &u_0 = \Big(g_1 - \sum_{k=1}^{4} \mathfrak{L}_{k+1} u_k\Big)/\mathfrak{L}_1, \quad u_{N+1} = \Big(g_2 - \sum_{k=1}^{4} \mathfrak{R}_k u_{N-4+k}\Big)/\mathfrak{R}_5. \end{aligned} \tag{19}$$

To derive the Thomas algorithm for the system (19), the following lemma analyze the matrix $\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}$.

**Lemma 3.1.** *For the* $2 \times 2$ *matrix* $\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}$, *its determinant can be calculated as*

$$\det\big(\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}\big) = \frac{144h^2}{(12\alpha_1 h - 25\beta_1)(12\alpha_2 h + 25\beta_2)}\Big(\alpha_1\alpha_2 + \frac{\alpha_1\beta_2 - \alpha_2\beta_1}{b-a}\Big).$$

*Hence, the matrix* $\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}$ *is invertible, when* $|\alpha_1| + |\alpha_2| \neq 0$.

*Proof.* We first note that the exact solution of the two-point boundary value problem

$$u''(x) = 0, \quad x \in (a,b); \quad u(a) = \frac{1}{\mathfrak{L}_1}, \quad u(b) = 0 \tag{20}$$

is given by

$$u(x) = \frac{b-x}{(b-a)\mathfrak{L}_1}.$$

If we define $N \times 2$ matrix $\mathbf{r}$ by $\mathbf{r} = \mathcal{D}_4^{-1}\mathbf{Q}$, the definitions of $\mathcal{D}_4$ and $\mathbf{Q}$ give the relation

$$\mathfrak{L}_1 r_{i,1} = \mathfrak{R}_5 r_{N+1-i,2}, \quad i = 1,2,\ldots,N. \tag{21}$$

Further, the first column $\mathbf{r}_1$ of $\mathbf{r}$ is the solution of the system

$$\mathcal{D}_4\mathbf{r}_1 = \frac{1}{12h^2\mathfrak{L}_1}\begin{bmatrix} -10 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

which is obtained from the fourth-order finite difference method for the boundary value problem (20). Hence, the solution $\mathbf{r}_1$ can be exactly calculated by

$$r_{i,1} = \frac{1}{\mathfrak{L}_1}\Big(1 - \frac{i}{N+1}\Big), \quad i = 1,2,\ldots,N. \tag{22}$$

Combining (21) and (22) gives the explicit formula of $\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}$ as follows:

$$\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q} = 12h\begin{bmatrix} \frac{\alpha_1-\beta_1/(b-a)}{12\alpha_1 h-25\beta_1} & \frac{\beta_1/(b-a)}{12\alpha_2 h+25\beta_2} \\ \frac{-\beta_2/(b-a)}{12\alpha_1 h-25\beta_1} & \frac{\alpha_2+\beta_2/(b-a)}{12\alpha_2 h+25\beta_2} \end{bmatrix}. \tag{23}$$

Therefore, the determinant can be easily checked from a direct calculation and one can complete the proof. □

As discussed in Corollary 1, this lemma and the Sherman-Morrison formula [3] provide an explicit formula for the solution of the system of (19) defined by

$$\mathbf{u} = \Big(\mathcal{D}_4^{-1} - \mathcal{D}_4^{-1}\mathbf{Q}(\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q})^{-1}\mathbf{P}^T\mathcal{D}_4^{-1}\Big)\Big(\mathbf{f} + \mathbf{Q}\mathbf{g}\Big). \tag{24}$$

The explicit formula (24) indicates that we can quickly obtain the numerical solution by solving only two tridiagonal systems based on the Thomas algorithm. First we note that as derived the formula (22) and using the relation (21), one can explicitly solve the equation

$$\mathcal{D}_4\mathbf{r} = \mathbf{Q}$$

with

$$\mathbf{r} = \begin{bmatrix} \frac{N}{\mathfrak{L}_1(N+1)} & \frac{1}{\mathfrak{R}_5(N+1)} \\ \frac{N-1}{\mathfrak{L}_1(N+1)} & \frac{2}{\mathfrak{R}_5(N+1)} \\ \vdots & \vdots \\ \frac{1}{\mathfrak{L}_1(N+1)} & \frac{N}{\mathfrak{R}_5(N+1)} \end{bmatrix}. \tag{25}$$

Hence, the numerical solution can be obtained by solving

$$\mathcal{D}_4\mathbf{x} = \mathbf{f} + \mathbf{Q}\mathbf{g}, \quad \mathbf{y} = \mathbf{P}^T\mathbf{x}, \quad \mathbf{u} = \mathbf{x} - \mathbf{r}\Big(\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}\Big)^{-1}\mathbf{y}. \tag{26}$$

Here, the first equation of (26) can be solved with the Thomas algorithm discussed in (14) or (15). The pseudo code of (26) is described in Algorithm 1.

**Algorithm 1** Pseudo code of (26)

```
**Input**: $(a, b, \alpha_i, \beta_i, g_i, (i = 1, 2), h$ and $f)$
1: Discretize the spatial domain $(a, b)$ with an uniform step size $h$ and construct two vectors $\mathbf{f}$ and $\mathbf{g}$.
2: Explicitly construct two matrices $\mathbf{r}$ and $\Big(\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}\Big)^{-1}$ defined in (25) and (23), respectively.
3: Solve the tridiagonal system $\mathcal{B}_2\mathbf{z} = \mathbf{f} + \mathbf{Q}\mathbf{g}$ by using the Thomas algorithm.
4: Solve the tridiagonal system $\mathcal{D}_2\mathbf{x} = \mathbf{z} - \frac{97+56\sqrt{3}}{144}\mathbf{r}(\mathbf{P}^T\mathbf{z})$ by using the Thomas algorithm.
5: Compute $\mathbf{y} = \mathbf{P}^T\mathbf{x}$.
6: Compute $\mathbf{u} = \mathbf{x} - \mathbf{r}\Big(\mathcal{I}_2 + \mathbf{P}^T\mathcal{D}_4^{-1}\mathbf{Q}\Big)^{-1}\mathbf{y}$.
**Output**: $\mathbf{u}$
```

*Remark* 3.2. We observe that the banded system (19) of size $n = N - 1$ has the bandwidth $w = 4$ and hence a general banded matrix solver for the system (19) requires at least a total of $\Big(w^2 + 4w\Big)n = 32n$ multiplication [4]. While the proposed method solve only two tridiagonal systems and saves the multiplication operation of $22n$ since a total of $5n$ multiplications are required for solving a tridiagonal system for the Thomas algorithm.

*Remark* 3.3. Lemma 3.1 indicates that the equation (24) is not available when $\alpha_1 = \alpha_2 = 0$. To solve the pure Neumann boundary problem, we suggest the following procedure:

$$\begin{aligned} \widehat{\mathbf{u}} &= \Big(\mathcal{D}_4^{-1} - \mathcal{D}_4^{-1}\widehat{\mathbf{Q}}\mathbf{P}^T\mathcal{D}_4^{-1}\Big)\Big(\mathbf{f} + \widehat{\mathbf{Q}}\mathbf{g}\Big), \\ \widehat{u}_{N+1} &= 0, \quad \widehat{u}_0 = \Big(g_1 - \sum_{k=1}^{4}\mathfrak{L}_{k+1}\widehat{u}_k\Big)/\mathfrak{L}_1, \end{aligned} \tag{27}$$

where

$$\widehat{\mathbf{Q}}(:,1) = \mathbf{Q}(:,1), \quad \widehat{\mathbf{Q}}(:,2) = 0.$$

Finally, we set $u_k = \widehat{u}_k + u_{N+1}$, $k = 0, 1, \ldots, N+1$.

*Remark* 3.4. In this remark, we consider an application of the proposed algorithm for solving a general two-point boundary value problem described by

$$\begin{aligned} &\frac{d^2}{dx^2}u(x) + p(x)\frac{d}{dx}u(x) + q(x)u(x) = f(x), \quad x \in (a,b), \\ &\alpha_1 u(a) + \beta_1 u'(a) = g_1, \quad \alpha_2 u(b) + \beta_2 u'(b) = g_2. \end{aligned} \tag{28}$$

As a direct application, we propose a simple iterative algorithm such as

$$\frac{d^2}{dx^2}u^{(k+1)}(x) = f(x) - p(x)\frac{d}{dx}u^{(k)}(x) - q(x)u^{(k)}(x), \quad k = 0, 1, 2, \ldots.$$

To ensure a fast convergence of the iterative method while maintaining the efficiency of the proposed algorithm based on the Thomas algorithm, we suggest the central finite difference method to get the initial guess $u^{(0)}$. For the convenience of readers, we introduce the second-order central finite difference scheme for solving the problem (28) as follows:

$$\begin{aligned} &\left(\frac{\alpha_1(2 - hp(x_0))}{\beta_1 h} - \frac{2 - h^2 q(x_0)}{h^2}\right)u_0 + \frac{2}{h^2}u_1 = f(x_0) + \frac{2 - hp(x_0)}{h\beta_1}g_1, \\ &\frac{2 - hp(x_i)}{2h^2}u_{i-1} - \frac{2 - h^2 q(x_i)}{h^2}u_i + \frac{2 + hp(x_i)}{2h^2}u_{i+1} = f_i, \quad i = 2, 3, \ldots, N, \\ &\left(-\frac{\alpha_1(2 + hp(x_{N+1}))}{\beta_2 h} - \frac{2 - h^2 q(x_{N+1})}{h^2}\right)u_{N+1} + \frac{2}{h^2}u_N \\ &= f(x_{N+1}) - \frac{2 + hp(x_{N+1})}{h\beta_2}g_2. \end{aligned}$$

## 4. Numerical experiments

This section aims to present the numerical investigation of the proposed scheme for solving the two-point boundary value problems. For this, we measure the error defined as follows:

$$Error = \frac{\|u_{approx} - u_{exact}\|}{\|u_{exact}\|},$$

where $u_{exact}$ and $u_{approx}$ are the analytic and approximate solutions, respectively, and $\|\cdot\|$ is the $L_2$-norm.

To show the efficiency and superiority of the proposed method, its results are compared with those of Matlab's built-in function "mldivide." For a fair comparison, the codes for all programs are written in Matlab R2020a and run on a Windows 10 PC with 3.59 GHz Ryzen 5 3600 processor.

Since the matrix $\mathcal{D}_4 + \mathbf{Q}\mathbf{P}^T$ in (19) is a sparse and banded matrix, the "mldivide" uses a banded solver, which can be checked using the Matlab function"spparms('spumoni',2)". In particular, we check how much the proposed method improves the computational speed, for which we measure *Speed* defined as

$$Speed = cpu_m/cpu_p,$$

where $cpu_m$ and $cpu_p$ are the computational costs for "mldivide" and the proposed scheme, respectively.

**Example 4.1.** We consider two-point boundary value problem (16) with the analytic solution

$$u(x) = \sin(x), \quad -100 \leq x \leq 100$$

and the boundary condition

$$\begin{cases} \alpha_1 u(-100) + \beta_1 u'(-100) = g_1, \\ \alpha_2 u(100) + \beta_2 u'(100) = g_2, \end{cases}$$

where $g_k$ are chosen from the analytic solution.

This problem can deal with four boundary conditions, such as Dirichlet, Neumann, Robin, and Mixed boundaries, by selecting $\alpha_k$, and $\beta_k$ $(k = 1, 2)$ as 0 or 1. We consider the following for five cases.
- For the Dirichlet boundary condition, we choose $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 0)$.
- For the Neumann boundary condition, we choose $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (0, 1, 0, 1)$.
- For the Mixed boundary condition, we choose $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 1)$, and $(1, 0, 0, 1)$.
- For the Robin boundary condition, we choose $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 1, 1, 1)$.

For each case, we measured the error, computational cost, and *Speed* for the numerical results obtained by both the proposed method and "mldivide" by varying resolution $N$ from $2^{10}$ to $2^{14}$. These results are recorded in Tables 1-5. In all tables, it can be seen that both methods have fourth-order accuracy, and their obtained results are almost the same in the sense of both accuracy and the order of convergence. However, the proposed method is at least six times and on average 11.76-times faster than "mldivide," in terms of computational cost, which guarantee the effectiveness of the developed Thomas algorithm.

Table 1. Numerical performances of Example 4.1 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 0)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (26) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $1.91 \times 10^{-5}$ | – | $8.21 \times 10^{-4}$ | $1.91 \times 10^{-5}$ | – | $8.97 \times 10^{-5}$ | 9.16 |
| $2^{11}$ | $1.10 \times 10^{-6}$ | 4.11 | $1.57 \times 10^{-3}$ | $1.10 \times 10^{-6}$ | 4.11 | $1.13 \times 10^{-4}$ | 13.82 |
| $2^{12}$ | $6.81 \times 10^{-8}$ | 4.02 | $2.89 \times 10^{-3}$ | $6.81 \times 10^{-8}$ | 4.02 | $2.89 \times 10^{-4}$ | 9.99 |
| $2^{13}$ | $4.24 \times 10^{-9}$ | 4.00 | $6.10 \times 10^{-3}$ | $4.24 \times 10^{-9}$ | 4.00 | $5.25 \times 10^{-4}$ | 11.63 |
| $2^{14}$ | $2.70 \times 10^{-10}$ | 3.97 | $1.24 \times 10^{-2}$ | $2.65 \times 10^{-10}$ | 4.00 | $8.71 \times 10^{-4}$ | 14.20 |

TABLE 2. Numerical performances of Example 4.1 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (0, 1, 0, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (27) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $3.60 \times 10^{-2}$ | – | $9.09 \times 10^{-4}$ | $3.60 \times 10^{-2}$ | – | $1.32 \times 10^{-4}$ | 6.88 |
| $2^{11}$ | $2.33 \times 10^{-3}$ | 3.95 | $1.58 \times 10^{-3}$ | $2.33 \times 10^{-3}$ | 3.95 | $1.21 \times 10^{-4}$ | 13.13 |
| $2^{12}$ | $1.48 \times 10^{-4}$ | 3.98 | $3.12 \times 10^{-3}$ | $1.48 \times 10^{-4}$ | 3.98 | $3.36 \times 10^{-4}$ | 9.28 |
| $2^{13}$ | $9.34 \times 10^{-6}$ | 3.99 | $6.00 \times 10^{-3}$ | $9.34 \times 10^{-6}$ | 3.99 | $5.01 \times 10^{-4}$ | 11.98 |
| $2^{14}$ | $5.87 \times 10^{-7}$ | 3.99 | $1.22 \times 10^{-2}$ | $5.86 \times 10^{-7}$ | 3.99 | $8.29 \times 10^{-4}$ | 14.77 |

TABLE 3. Numerical performances of Example 4.1 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (26) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $1.85 \times 10^{-4}$ | – | $9.29 \times 10^{-4}$ | $1.85 \times 10^{-4}$ | – | $1.39 \times 10^{-4}$ | 6.70 |
| $2^{11}$ | $1.19 \times 10^{-5}$ | 3.97 | $1.55 \times 10^{-3}$ | $1.19 \times 10^{-5}$ | 3.97 | $1.18 \times 10^{-4}$ | 13.14 |
| $2^{12}$ | $7.52 \times 10^{-7}$ | 3.98 | $2.84 \times 10^{-3}$ | $7.52 \times 10^{-7}$ | 3.98 | $2.15 \times 10^{-4}$ | 13.17 |
| $2^{13}$ | $4.74 \times 10^{-8}$ | 3.99 | $6.56 \times 10^{-3}$ | $4.74 \times 10^{-8}$ | 3.99 | $4.45 \times 10^{-4}$ | 14.74 |
| $2^{14}$ | $2.97 \times 10^{-9}$ | 4.00 | $1.20 \times 10^{-2}$ | $2.99 \times 10^{-9}$ | 3.99 | $7.99 \times 10^{-4}$ | 15.01 |

TABLE 4. Numerical performances of Example 4.1 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 0, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (26) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $3.60 \times 10^{-2}$ | – | $8.40 \times 10^{-4}$ | $3.60 \times 10^{-2}$ | – | $8.66 \times 10^{-5}$ | 9.70 |
| $2^{11}$ | $2.33 \times 10^{-3}$ | 3.95 | $1.66 \times 10^{-3}$ | $2.33 \times 10^{-3}$ | 3.95 | $1.17 \times 10^{-4}$ | 14.11 |
| $2^{12}$ | $1.48 \times 10^{-4}$ | 3.98 | $3.13 \times 10^{-3}$ | $1.48 \times 10^{-4}$ | 3.98 | $3.42 \times 10^{-4}$ | 9.15 |
| $2^{13}$ | $9.34 \times 10^{-6}$ | 3.99 | $5.77 \times 10^{-3}$ | $9.34 \times 10^{-6}$ | 3.99 | $4.96 \times 10^{-4}$ | 11.65 |
| $2^{14}$ | $5.86 \times 10^{-7}$ | 3.99 | $1.21 \times 10^{-2}$ | $5.86 \times 10^{-7}$ | 3.99 | $8.55 \times 10^{-4}$ | 14.16 |

TABLE 5. Numerical performances of Example 4.1 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 1, 1, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (26) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $3.13 \times 10^{-4}$ | – | $9.36 \times 10^{-4}$ | $3.13 \times 10^{-4}$ | – | $1.52 \times 10^{-4}$ | 6.17 |
| $2^{11}$ | $2.02 \times 10^{-5}$ | 3.95 | $1.57 \times 10^{-3}$ | $2.02 \times 10^{-5}$ | 3.95 | $1.42 \times 10^{-4}$ | 11.09 |
| $2^{12}$ | $1.28 \times 10^{-6}$ | 3.98 | $2.95 \times 10^{-3}$ | $1.28 \times 10^{-6}$ | 3.98 | $3.58 \times 10^{-4}$ | 8.24 |
| $2^{13}$ | $8.09 \times 10^{-8}$ | 3.99 | $6.93 \times 10^{-3}$ | $8.10 \times 10^{-8}$ | 3.99 | $4.62 \times 10^{-4}$ | 15.00 |
| $2^{14}$ | $5.11 \times 10^{-9}$ | 3.99 | $1.24 \times 10^{-2}$ | $5.09 \times 10^{-9}$ | 3.99 | $7.55 \times 10^{-4}$ | 16.43 |

**Example 4.2.** We consider the two-point boundary value problem given by

$$u''(x) = -\sin(x), \quad x \in (a, b),$$
$$\begin{cases} \alpha_1 u(a) + \beta_1 u'(a) = g_1, \\ \alpha_2 u(b) + \beta_2 u'(b) = g_2, \end{cases}$$

whose analytic solution is calculated by

$$u(x) = \sin(x) + \frac{(\alpha_2 \widehat{g}_1 - \alpha_1 \widehat{g}_2)x + (\alpha_1 a + \beta_1)\widehat{g}_2 - (\alpha_2 b + \beta_2)\widehat{g}_1}{(\beta_1 + a\alpha_1)\alpha_2 - (\beta_2 + b\alpha_2)\alpha_1}$$

for the case $(\beta_1 + a\alpha_1)\alpha_2 \neq (\beta_2 + b\alpha_2)\alpha_1$. Here, $\widehat{g}_1 = g_1 - \alpha_1 \sin(a) - \beta_1 \cos(a)$, and $\widehat{g}_2 = g_2 - \alpha_2 \sin(b) - \beta_2 \cos(b)$. The computational domain is set as $(a, b) = (-100, 100)$.

In our experiment, $g_1$ and $g_2$ are set as 1 and 0, respectively. Additionally, we only consider the three cases, Robin, Dirichlet, and two mixed boundary conditions without the pure Neumann boundary, because the analytic solution does not exist when $\alpha_k = 0$ and $\beta_k = 1$, in this example. To examine the precision and superiority of the proposed method, the error, computational cost, and *Speed* are measured by both the proposed method and "mldivide" by varying resolution $N$ from $2^{10}$ to $2^{14}$. The results are listed in Tables 6-9. As with Example 4.1, the proposed method is at least seven-times and on average 11.51-times faster than "mldivide," in terms of computational costs. These tables show that the proposed method has the fourth-order accuracy in all cases. Further, it is worth observing that the solver "mldivide" has a phenomenon of order reduction for both cases, Dirichlet boundary and Mixed boundary conditions. In contrast, the proposed method does not have such phenomena.

Table 6. Numerical performances of Example 4.2 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 0)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (24) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $1.56 \times 10^{-5}$ | 4.64 | $8.88 \times 10^{-4}$ | $1.56 \times 10^{-5}$ | 4.64 | $8.22 \times 10^{-5}$ | 10.80 |
| $2^{11}$ | $9.02 \times 10^{-7}$ | 4.11 | $1.64 \times 10^{-3}$ | $9.02 \times 10^{-7}$ | 4.11 | $1.57 \times 10^{-4}$ | 10.41 |
| $2^{12}$ | $5.56 \times 10^{-8}$ | 4.02 | $3.54 \times 10^{-3}$ | $5.56 \times 10^{-8}$ | 4.02 | $2.74 \times 10^{-4}$ | 12.93 |
| $2^{13}$ | $3.49 \times 10^{-9}$ | 3.99 | $6.62 \times 10^{-3}$ | $3.47 \times 10^{-9}$ | 4.00 | $4.72 \times 10^{-4}$ | 14.03 |
| $2^{14}$ | $1.64 \times 10^{-9}$ | 1.09 | $1.24 \times 10^{-2}$ | $2.17 \times 10^{-10}$ | 4.00 | $8.01 \times 10^{-4}$ | 15.47 |

Table 7. Numerical performances of Example 4.2 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (24) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $1.73 \times 10^{-4}$ | 3.94 | $9.88 \times 10^{-4}$ | $1.73 \times 10^{-4}$ | 3.94 | $1.40 \times 10^{-4}$ | 7.04 |
| $2^{11}$ | $1.11 \times 10^{-5}$ | 3.97 | $1.67 \times 10^{-3}$ | $1.11 \times 10^{-5}$ | 3.97 | $2.00 \times 10^{-4}$ | 8.37 |
| $2^{12}$ | $7.03 \times 10^{-7}$ | 3.98 | $3.22 \times 10^{-3}$ | $7.03 \times 10^{-7}$ | 3.98 | $2.76 \times 10^{-4}$ | 11.65 |
| $2^{13}$ | $4.43 \times 10^{-8}$ | 3.99 | $6.26 \times 10^{-3}$ | $4.43 \times 10^{-8}$ | 3.99 | $5.00 \times 10^{-4}$ | 12.52 |
| $2^{14}$ | $2.66 \times 10^{-9}$ | 4.06 | $1.36 \times 10^{-2}$ | $2.80 \times 10^{-9}$ | 3.99 | $8.02 \times 10^{-4}$ | 16.98 |

Table 8. Numerical performances of Example 4.2 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 0, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (24) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $2.58 \times 10^{-4}$ | 3.87 | $9.41 \times 10^{-4}$ | $2.58 \times 10^{-4}$ | 3.87 | $8.94 \times 10^{-5}$ | 10.52 |
| $2^{11}$ | $1.66 \times 10^{-5}$ | 3.95 | $1.75 \times 10^{-3}$ | $1.66 \times 10^{-5}$ | 3.95 | $1.53 \times 10^{-4}$ | 11.44 |
| $2^{12}$ | $1.06 \times 10^{-6}$ | 3.97 | $3.23 \times 10^{-3}$ | $1.06 \times 10^{-6}$ | 3.98 | $3.19 \times 10^{-4}$ | 10.12 |
| $2^{13}$ | $7.02 \times 10^{-8}$ | 3.91 | $6.01 \times 10^{-3}$ | $6.67 \times 10^{-8}$ | 3.99 | $5.34 \times 10^{-4}$ | 11.25 |
| $2^{14}$ | $1.74 \times 10^{-8}$ | 2.01 | $1.30 \times 10^{-2}$ | $4.20 \times 10^{-9}$ | 3.99 | $7.71 \times 10^{-4}$ | 16.89 |

Table 9. Numerical performances of Example 4.2 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 1, 1, 1)$.

| $N$ | Scheme (19) with mldivide | | | Proposed scheme (24) | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^{10}$ | $2.78 \times 10^{-4}$ | 3.87 | $8.58 \times 10^{-4}$ | $2.78 \times 10^{-4}$ | 3.87 | $1.02 \times 10^{-4}$ | 8.37 |
| $2^{11}$ | $1.80 \times 10^{-5}$ | 3.95 | $1.68 \times 10^{-3}$ | $1.80 \times 10^{-5}$ | 3.95 | $1.80 \times 10^{-4}$ | 9.32 |
| $2^{12}$ | $1.14 \times 10^{-6}$ | 3.98 | $3.08 \times 10^{-3}$ | $1.14 \times 10^{-6}$ | 3.98 | $3.39 \times 10^{-4}$ | 9.09 |
| $2^{13}$ | $7.24 \times 10^{-8}$ | 3.98 | $6.03 \times 10^{-3}$ | $7.21 \times 10^{-8}$ | 3.99 | $6.14 \times 10^{-4}$ | 9.82 |
| $2^{14}$ | $5.70 \times 10^{-9}$ | 3.67 | $1.17 \times 10^{-2}$ | $4.53 \times 10^{-9}$ | 3.99 | $8.91 \times 10^{-4}$ | 13.14 |

**Example 4.3.** We consider a general two-point boundary value problem given by

$$-u''(x) + \frac{2}{x+1}u'(x) + \left(1 - \frac{2}{(1+x)^2}\right)u(x) = 4x(1+x)e^x, \quad x \in (a,b),$$
$$\begin{cases} \alpha_1 u(a) + \beta_1 u'(a) = g_1, \\ \alpha_2 u(b) + \beta_2 u'(b) = g_2, \end{cases}$$

whose analytic solution is calculated by [2]

$$u(x) = x(1-x^2)e^x.$$

The computational domain is set as $(a, b) = (0, 1)$.

Table 10. Numerical performances of Example 4.3 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 0)$.

| $N$ | Standard method with mldivide | | | Proposed scheme with 8 iteration | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^6$ | $3.60 \times 10^{-8}$ | 4.03 | $1.70 \times 10^{-4}$ | $3.60 \times 10^{-8}$ | 4.03 | $1.75 \times 10^{-4}$ | 0.97 |
| $2^7$ | $2.23 \times 10^{-9}$ | 4.01 | $2.29 \times 10^{-4}$ | $2.24 \times 10^{-9}$ | 4.01 | $2.04 \times 10^{-4}$ | 1.12 |
| $2^8$ | $1.40\times10^{-10}$ | 3.99 | $3.33 \times 10^{-4}$ | $1.40\times10^{-10}$ | 4.00 | $2.31 \times 10^{-4}$ | 1.44 |
| $2^9$ | $1.24\times10^{-11}$ | 3.50 | $5.91 \times 10^{-4}$ | $8.78\times10^{-12}$ | 3.99 | $3.28 \times 10^{-4}$ | 1.80 |
| $2^{10}$ | $1.14\times10^{-11}$ | 0.11 | $1.19 \times 10^{-3}$ | $7.49\times10^{-13}$ | 3.55 | $5.33 \times 10^{-4}$ | 2.22 |

TABLE 11. Numerical performances of Example 4.3 with $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 0, 1, 1)$.

| $N$ | Standard method with mldivide | | | Proposed scheme with 12 iteration | | | *Speed* |
|---|---|---|---|---|---|---|---|
| | *Error* | *rate* | $cpu_m$ | *Error* | *rate* | $cpu_p$ | |
| $2^6$ | $1.51 \times 10^{-6}$ | 3.98 | $1.69 \times 10^{-4}$ | $1.51 \times 10^{-6}$ | 3.98 | $2.17 \times 10^{-4}$ | 0.78 |
| $2^7$ | $9.53 \times 10^{-8}$ | 3.99 | $2.34 \times 10^{-4}$ | $9.53 \times 10^{-8}$ | 3.99 | $2.45 \times 10^{-4}$ | 0.96 |
| $2^8$ | $5.98 \times 10^{-9}$ | 3.99 | $3.25 \times 10^{-4}$ | $5.98 \times 10^{-9}$ | 3.99 | $2.84 \times 10^{-4}$ | 1.14 |
| $2^9$ | $3.81 \times 10^{-10}$ | 3.97 | $5.76 \times 10^{-4}$ | $3.75 \times 10^{-10}$ | 4.00 | $3.96 \times 10^{-4}$ | 1.46 |
| $2^{10}$ | $4.54 \times 10^{-11}$ | 3.07 | $1.17 \times 10^{-3}$ | $2.36 \times 10^{-11}$ | 3.99 | $6.47 \times 10^{-4}$ | 1.80 |

As mentioned in Remark 3.4, in this example, we investigate whether the proposed method is applicable to a general second-order problem with both Dirichlet and Robin boundary conditions. To examine the precision and superiority of the proposed method, the error, computational cost, and *Speed* are measured by both the proposed method and "mldivide" by varying resolution $N$ from $2^6$ to $2^{10}$. The results are listed in Tables 10-11. The proposed method requires 8 and 12 iteration procedures to achieve the required convergence rate according to the boundary conditions, respectively. First, the column of *Speed* shows that the proposed method is slightly faster than "mldivide" as the spatial resolution increases. Further, it is worth observing that the order reduction of the solver "mldivide" is more severe than the proposed method. However, the proposed method can improve the order reduction phenomenon by increasing the number of iterations.

## 5. Conclusions and perspectives

We developed a general formula to decompose the high-order finite-difference matrix with the lower-order finite-difference. This formula is used to propose both the inverse formula and the Thomas algorithm of the system induced by the fourth-order finite difference method for two boundary value problems with Robin boundary. The proposed algorithm can be applied in various problems and ways to simulation using the finite-difference methodology with higher-order accuracy. Furthermore, it is expected to provide a clue that can be solved remarkably quickly through the application of the Thomas algorithm by dividing the coefficient matrix of the high-order finite difference method into tridiagonal matrices.

**Acknowledgements.** This research was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.2020R1A2C1A01008506, No.2020R1A4A1018190). The Bak was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.2020R1C1C1A01004139).

## References

[1] B. Fornberg, *Calculation of weights in finite difference formulas*, SIAM Rev. **40** (1998), no. 3, 685–691. https://doi.org/10.1137/S0036144596322507

[2] M. F. Pettigrew and H. Rasmussen, *A compact method for second-order boundary value problems on nonuniform grids*, Comput. Math. Appl. **31** (1996), no. 9, 1–16. https://doi.org/10.1016/0898-1221(96)00037-5

[3] J. Sherman and W. J. Morrison, *Adjustment of an inverse matrix corresponding to a change in one element of a given matrix*, Ann. Math. Statistics **21** (1950), 124–127. https://doi.org/10.1214/aoms/1177729893

[4] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, MA, 1986.

[5] L. H. Thomas, *Elliptic problems in linear differential equations over a Network*, Watson Sci. Comput. Lab Report, Columbia University, New York, 1949.

[6] R. A. Usmani, *Inversion of a tridiagonal Jacobi matrix*, Linear Algebra Appl. **212/213** (1994), 413–414. https://doi.org/10.1016/0024-3795(94)90414-6

Soyoon Bak
Department of Mathematics
Kyungpook National University
Daegu 41566, Korea
*Email address*: jiya525@knu.ac.kr

Philsu Kim
Department of Mathematics
Kyungpook National University
Daegu 41566, Korea
*Email address*: kimps@knu.ac.kr

Sangbeom Park (earlier known as Xiangfan Piao)
Department of Mathematics
Kyungpook National University
Daegu 41566, Korea
*Email address*: piaoxf76@gmail.com