British Journal of Mathematics & Computer Science
18(1): 1-8, 2016, Article no.BJMCS.27679
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

On Solving Comrade Linear Systems Via Transformations

A. A. Karawia[1,2*] **and S. S. Askar**[1,3]

[1] *Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt.*
[2] *Computer Science Unit, Deanship of Educational Services, Qassim University, P.O.Box 6595, Buraidah 51452, Saudi Arabia.*
[3] *Department of Statistics and Operations Researches, College of Science, King Saud University, P.O.Box 2455, Riyadh 11451, Saudi Arabia.*

Authors' contributions

This work was carried out in collaboration between both authors. Both authors designed the study, performed the statistical analysis, wrote the protocol, wrote the first draft of the manuscript, and managed the analyses of the study and literature searches. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/27679
Editor(s):
(1) Sergio Serrano, Department of Applied Mathematics, University of Zaragoza, Spain.
Reviewers:
(1) Abdalah Rababah, Jordan University of Science and Technology, Jordan.
(2) Zhaolin Jiang, Linyi University, China.
(3) Alessandro Danielis, Institute of Information Science and Technology (ISTI), National Research Council (CNR) of Pisa, Italy.
(4) Yanpeng Li, National University of Defense Technology, Changsha, China.
(5) Anonymous, Atilim University, Turkey.
Complete Peer review History: http://www.sciencedomain.org/review-history/15648

Original Research Article

Received: 14th June 2016
Accepted: 27th July 2016
Published: 3rd August 2016

Abstract

Recently, a new symbolic algorithm for solving comrade linear systems is given by Karawia [A. A. Karawia, Symbolic algorithm for solving comrade linear systems based on a modified Stair-Diagonal Approach, Appl. Math. Lett. 2013; 26:913-918.]. This paper introduces different numerical and symbolic algorithms for solving comrade linear systems via transformations. Our symbolic algorithm removes the cases where the numerical algorithms fail. The computational cost of these algorithms is given. It gives better absolute error than the other methods. The

**Corresponding author: E-mail: abibka@mans.edu.eg;*

algorithm is implementable to the Computer Algebra System (CAS) such as MAPLE and MATLAB. Some illustrative examples are presented for the sake of illustration.

Keywords: Comrade matrices; tridiagonal matrices; determinants; Computer Algebra Systems (CAS).

2010 Mathematics Subject Classification: 15A15, 15A23, 33F10, F.2.1, G.1.0, 68W30, 11Y05.

1 Introduction

Comrade linear system of size n takes the form:

$$CX = Y, \tag{1.1}$$

where

$$C = \begin{pmatrix} -\frac{\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ \frac{\gamma_2}{\alpha_2} & -\frac{\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & 0 & \dots & \dots & \dots & 0 & 0 \\ 0 & \frac{\gamma_3}{\alpha_3} & -\frac{\beta_3}{\alpha_3} & \frac{1}{\alpha_3} & 0 & \dots & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \vdots & \vdots & \frac{\gamma_{n-2}}{\alpha_{n-2}} & -\frac{\beta_{n-2}}{\alpha_{n-2}} & \frac{1}{\alpha_{n-2}} & 0 \\ 0 & 0 & 0 & \dots & \dots & 0 & \frac{\gamma_{n-1}}{\alpha_{n-1}} & -\frac{\beta_{n-1}}{\alpha_{n-1}} & \frac{1}{\alpha_{n-1}} \\ -\frac{a_n}{\alpha_n} & -\frac{a_{n-1}}{\alpha_n} & -\frac{a_{n-2}}{\alpha_n} & \dots & \dots & \frac{-a_4}{\alpha_n} & \frac{-a_3}{\alpha_n} & \frac{\gamma_n - a_2}{\alpha_n} & \frac{-\beta_n - a_1}{\alpha_n} \end{pmatrix}, n \geq 3, \tag{1.2}$$

where $X = (x_1, x_2, ..., x_n)^t$, $Y = (y_1, y_2, ..., y_n)^t$ and $\alpha_i \neq 0, i = 1, 2, 3, ..., n$.

Matrix C is called comrade matrix and it can be stored in $4n - 4$ memory locations. Comrade matrix is a generalization of the companion matrix and is associated with a polynomial expressed as a linear combination of an arbitrary orthogonal basis. This matrix appears frequently in many areas of science and engineering, for example in linear multivariable systems theory [1], computing the greatest common divisor of polynomials [2] and division of generalized polynomials [3].

Comrade linear system is widely used in solving problems in many areas of science, for example solving differential equations using finite differences, parallel computing, and telecommunication system analysis [4]-[6]. Finding the solution of such a linear system is usually required in these fields. This problem has been investigated by many authors (see for instance, [7]-[10]).In [7], the author has introduced efficient numerical and symbolic algorithms for solving comrade linear system using LU decomposition. Two algorithms are presented for solving the comrade linear systems based on the use of conventional fast tridiagonal solvers, and an efficient way of evaluating the determinant of the comrade matrix is given [8]. The author in [9] has described a symbolic computational algorithm for solving comrade linear systems based on a modified Stair-Diagonal Approach. Recently, the authors have derived a novel elementary algorithm for solving comrade linear systems based on matrix reordering and the Sherman-Morrison-Woodbury formula [10]. In this paper, we introduce more efficient algorithms based on transformations. Our proposed algorithms are completely different

from the algorithms in [9]. We used transformation to convert the comrade system (1.1) to equivalent simple system which we can solve it easily.

The paper is organized as follows: In Section 2, new algorithms for solving comrade linear systems are given. Some illustrative examples are introduced in Section 3. In Section 4, conclusions of the work are given.

2 Main Results

In this section, we shall concentrate on the construction of new algorithms for computing the solution of comrade linear system. For this goal, it is suitable to give five vectors $\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_n)$, $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_{n-1})$, $\boldsymbol{\psi} = (\psi_1, \psi_2, ..., \psi_{n-1})$, $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_{n-1})$, and $\boldsymbol{Z} = (z_1, z_2, ..., z_n)$, where

$$\mu_i = \begin{cases} -\frac{\beta_1}{\alpha_1} & i = 1, \\ \frac{-\beta_i - \gamma_i \phi_{i-1}}{\alpha_i} & i = 2, 3, ..., n-1, \\ -\sigma_{n-1}\phi_{n-1} - \frac{a_1 + \beta_n}{\alpha_n} & i = n, \end{cases} \tag{2.1}$$

$$\phi_i = \frac{1}{\alpha_i \mu_i}, \qquad i = 1, 2, ..., n-1, \tag{2.2}$$

$$\sigma_i = \begin{cases} -\frac{a_n}{\alpha_n} & i = 1, \\ -\sigma_{i-1}\phi_{i-1} - \frac{a_{n-i+1}}{\alpha_n} & i = 2, 3, ..., n-2, \\ -\sigma_{n-2}\phi_{n-2} + \frac{\gamma_n - a_2}{\alpha_n} & i = n-1, \end{cases} \tag{2.3}$$

$$z_i = \begin{cases} \frac{y_1}{\mu_1} & i = 1, \\ \frac{1}{\mu_i}(y_i - \frac{\gamma_i}{\alpha_i} z_{i-1}) & i = 2, 3, ..., n-1, \\ \frac{1}{\mu_n}(-\sigma_{n-1} z_{n-1} + \psi_{n-1}) & i = n, \end{cases} \tag{2.4}$$

and

$$\psi_i = \begin{cases} y_n & i = 1, \\ -\sigma_{i-1} z_{i-1} + \psi_{i-1} & i = 2, 3, ..., n-1. \end{cases} \tag{2.5}$$

By using these vectors together with appropriate elementary row operations, one can see that the system (1.1) may be transformed to the equivalent linear system(see Appendix A for more details):

$$\begin{pmatrix} 1 & \phi_1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \phi_2 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 1 & \phi_3 & 0 & \dots & \dots & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & 0 & 1 & \phi_{n-2} & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 & \phi_{n-1} \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ \vdots \\ \vdots \\ z_{n-2} \\ z_{n-1} \\ z_n \end{pmatrix} \tag{2.6}$$

By using a backward substitution, it is easy to solve the transformed system (2.6). As a result, the comrade linear system (1.1) can be solved by using the following numerical algorithm:

Algorithm 2.1. Numerical algorithm for solving comrade linear system

To solve the comrade linear system (1.1), we carried out the procedures as follows:
INPUT: Order of the matrix n and the components $\alpha_i, \beta_i, \gamma_i, a_i, \quad i = 1, 2, ..., n, (\gamma_1 = 0)$.
OUTPUT: The solution vector $X = (x_1, x_2, ..., x_n)^t$.
Step 1: Set $\mu_1 = -\beta_1/\alpha_1$, $\phi_1 = 1/(\alpha_1\mu_1)$, $z_1 = y_1/\mu_1$, $\sigma_1 = -a_n/\alpha_n$, and $\psi_1 = y_n$.
Step 2: Compute and simplify:
For $i = 2, 3, ..., n-2$ do
$\mu_i = (-\beta_i - \gamma_i\phi_{i-1})/\alpha_i$
$z_i = (y_i - \frac{\gamma_i}{\alpha_i} z_{i-1})/\mu_i$
$\phi_i = 1/(\alpha_i\mu_i)$
$\sigma_i = -\sigma_{i-1}\phi_{i-1} - a_{n-i+1}/\alpha_n$
$\psi_i = -\sigma_{i-1}z_{i-1} + \psi_{i-1}$
End do
Step 3: Compute and simplify:
$\mu_{n-1} = (-\beta_{n-1} - \gamma_{n-1}\phi_{n-2})/\alpha_{n-1}$
$\phi_{n-1} = 1/(\alpha_{n-1}\mu_{n-1})$
$\sigma_{n-1} = -\sigma_{n-2}\phi_{n-2} + (\gamma_n - a_2)/\alpha_n$
$\psi_{n-1} = -\sigma_{n-2}z_{n-2} + \psi_{n-2}$
$z_{n-1} = (y_{n-1} - \frac{\gamma_{n-1}}{\alpha_{n-1}} z_{n-2})/\mu_{n-1}$
$\mu_n = -\sigma_{n-1}\phi_{n-1} - (a_1 + \beta_n)/\alpha_n$
$z_n = (-\sigma_{n-1}z_{n-1} + \psi_{n-1})/\mu_n$
Step 4: Compute $det(C) = \prod_{i=1}^{n} \mu_i$.
Step 5: If $det(C) = 0$, then Exit and Print Message ("No solutions") end if.
Step 6: Set $x_n = z_n$.
Step 7: Compute and simplify:
For $i = n-1, n-2, ..., 1$ do
$x_i = z_i - \phi_i x_{i+1}$
End do

Algorithm 2.1, will be referred to as **COMTRANS** in the sequel. The computational cost of **COMTRANS** is $15n - 14$ operations. The sufficient conditions for its validity are $\mu_i \neq 0, i = 1, 2, ..., n$.

The following symbolic version algorithm is developed with a view to remove the cases where the numerical algorithm **COMTRANS** fails. The parameter $''s''$ in the following symbolic algorithm is just a symbolic character. It is a dummy argument and its actual value is zero.

Algorithm 2.2. Symbolic version algorithm for **COMTRANS** algorithm

To solve the comrade linear system (1.1), we carried out the procedures as follows:
INPUT: Order of the matrix n and the components $\alpha_i, \beta_i, \gamma_i, a_i, \quad i = 1, 2, ..., n, (\gamma_1 = 0)$.
OUTPUT: The solution vector $X = (x_1, x_2, ..., x_n)^t$.
Step 1: Set $\mu_1 = -\beta_1/\alpha_1$. If $\mu_1 = 0$ then $\mu_1 = s$ end if. $\phi_1 = 1/(\alpha_1\mu_1)$, $z_1 = y_1/\mu_1$, $\sigma_1 = -a_n/\alpha_n$, and $\psi_1 = y_n$.
Step 2: Compute and simplify:
For $i = 2, 3, ..., n-2$ do

$\mu_i = (-\beta_i - \gamma_i\phi_{i-1})/\alpha_i$. If $\mu_i = 0$ then $\mu_i = s$ end if.
$z_i = (y_i - \frac{\gamma_i}{\alpha_i} z_{i-1})/\mu_i$
$\phi_i = 1/(\alpha_i\mu_i)$
$\sigma_i = -\sigma_{i-1}\phi_{i-1} - a_{n-i+1}/\alpha_n$
$\psi_i = -\sigma_{i-1}z_{i-1} + \psi_{i-1}$
End do

Step 3: Compute and simplify:
$\mu_{n-1} = (-\beta_{n-1} - \gamma_{n-1}\phi_{n-2})/\alpha_{n-1}$. If $\mu_{n-1} = 0$ then $\mu_{n-1} = s$ end if.
$\phi_{n-1} = 1/(\alpha_{n-1}\mu_{n-1})$
$\sigma_{n-1} = -\sigma_{n-2}\phi_{n-2} + (\gamma_n - a_2)/\alpha_n$
$\psi_{n-1} = -\sigma_{n-2}z_{n-2} + \psi_{n-2}$
$z_{n-1} = (y_{n-1} - \frac{\gamma_{n-1}}{\alpha_{n-1}} z_{n-2})/\mu_{n-1}$
$\mu_n = -\sigma_{n-1}\phi_{n-1} - (a_1 + \beta_n)/\alpha_n$. If $\mu_n = 0$ then $\mu_n = s$ end if.
$z_n = (-\sigma_{n-1}z_{n-1} + \psi_{n-1})/\mu_n$

Step 4: Compute $det(C) = \left(\prod_{i=1}^{n} \mu_i\right)_{s=0}$.
Step 5: If $det(C) = 0$, then Exit and Print Message ("No solutions") end if.
Step 6: Set $x_n = z_n$.
Step 7: Compute and simplify:
For $i = n-1, n-2, ..., 1$ do
$x_i = z_i - \phi_i x_{i+1}$
End do
Step 8: Substitute $s = 0$ in all expressions of the solution vector $X = (x_1, x_2, ..., x_n)^t$.

Algorithm 2.2, will be referred to as **SCOMTRANS** in the sequel. In [[7]-[10]], the authors introduced numerical algorithms for finding the solution of comrade linear system (1.1) under conditions. On the other hand, **SCOMTRANS** algorithm is given with no conditions but the matrix C is assumed to be a non-singular matrix.

3 Illustrative Examples

In this section, we give three examples for the sake of illustration. All experiments were carried out in MATLAB 2014a with an Intel(R) Core(TM) i7-4700MQ Quad Core.

Example 3.1. *Case I:* $\mu_i \neq 0, \quad i = 1, 2, ..., n$.
Solve the following comrade linear system of size 7[9]

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{3} & -\frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{1} & -\frac{3}{-1} & \frac{1}{-1} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{5} & -\frac{4}{5} & \frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{2} & -\frac{6}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{7} & -\frac{5}{7} & \frac{1}{7} \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{2}{3} & -\frac{4}{3} & -\frac{3}{3} & \frac{4-1}{3} & \frac{-3-1}{3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{2}{3} \\ \frac{3}{1} \\ \frac{-1}{1} \\ \frac{-6}{1} \\ \frac{-13}{7} \\ \frac{-50}{3} \end{pmatrix} \tag{3.1}$$

Solution: *We have*
$n = 7$, $\alpha = (2, 3, -1, 5, 2, 7, 3)^t$, $\beta = (1, 2, 3, 4, 6, 5, 3)^t$, $\gamma = (0, 3, 1, 2, 3, 2, 4)^t$, $a = (1, 1, 3, 4, 2, 1, 1)^t$, *and* $Y = (1/2, 2/3, 3/1, -1/1, -6/1, -13/7, -50/3)^t$.

Applying the ***COMTRANS*** *algorithm, it yields*
X=*Comtrans_I*$(\alpha, \beta, \gamma, a, Y, n) = (1, 2, 3, 4, 5, 6, 7)^t$ *and* $det(C) = -0.9286$.

Example 3.2. *Case II: At least one of* $\mu_i = 0, \quad i = 1, 2, ..., n.$
Solve the following comrade linear system of size 4

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{3}{3} & -\frac{3}{3} & \frac{1}{3} & 0 \\ 0 & \frac{-1}{-1} & -\frac{3}{-1} & \frac{1}{-1} \\ \frac{-4}{5} & \frac{-2}{5} & \frac{-3-2}{5} & \frac{-4-3}{5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{3}{1} \\ \frac{-18}{5} \end{pmatrix} \tag{3.2}$$

Solution: *We have*
$n = 4$, $\alpha = (2, 3, -1, 5)^t$, $\beta = (1, 3, 3, 4)^t$, $\gamma = (0, 3, -1, -3)^t$, $a = (3, 2, 2, 4)^t$, *and* $Y = (0, 1/3, 3/1, -1/5)^t$.

The numerical algorithm ***COMTRANS*** *fails to solve the comrade linear system (4.2) since* $\mu_2 = 0$.

Applying the ***SCOMTRANS*** *algorithm, it yields*
$X = SCOMTRANS(\alpha, \beta, \gamma, a, Y, n) = (-\frac{1}{6s-1}, -\frac{1}{6s-1}, \frac{9s-1}{6s-1}, \frac{9s-1}{6s-1})_{s=0} = (1, 1, 1, 1)^t$

Example 3.3. *With a view to show the efficiency and competitiveness of the proposed algorithm described in this paper, we now consider an n-by-n comrade linear system* $Cx = y$ *of the form [9]*

$$\begin{pmatrix} \frac{1}{2} & 1 & 0 & \cdots & \cdots & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\frac{1}{2} & \frac{1}{2} & 1 \\ -1 & -1 & \cdots & -1 & -\frac{3}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ 1 \\ \vdots \\ \vdots \\ \vdots \\ 1 \\ -n \end{pmatrix} \tag{3.3}$$

The exact solution of linear system (3.3) is $x = (1, 1, ..., 1)^t$. *We give some comparisons between our proposed algorithm and the algorithms in literature to compute* $\overline{x}$. *The absolute errors* $\varepsilon = \|x - \overline{x}\|_\infty$, *and CPU times with these algorithms are shown in Table 1.*

Table 1. Errors and CPU times in seconds for our proposed algorithm and the other algorithms(KCOM,ECLS, Sogabe[8], and SCMSDA) for different values of n.

Algorithms	n	100	500	1000	5000
COMTRANS	$\|x - \overline{x}\|_\infty$	7.7716×10^{-16}	7.7716×10^{-16}	7.7716×10^{-16}	7.7716×10^{-16}
	$CPUtime(s)$	0.0065	0.0076	0.0086	0.0341
KCOM	$\|x - \overline{x}\|_\infty$	1.1102×10^{-15}	1.1102×10^{-15}	1.1102×10^{-15}	1.1102×10^{-15}
	$CPUtime(s)$	0.0055	0.0066	0.0071	0.0280
ECLS	$\|x - \overline{x}\|_\infty$	2.8422×10^{-14}	1.1369×10^{-13}	1.1369×10^{-13}	1.8190×10^{-12}
	$CPUtime(s)$	0.0048	0.0080	0.0086	0.0117
Sogabe[8]	$\|x - \overline{x}\|_\infty$	2.8422×10^{-14}	1.7053×10^{-13}	9.0949×10^{-13}	3.0923×10^{-10}
	$CPUtime(s)$	0.0081	0.0433	0.0476	0.0525
SCMSDA	$\|x - \overline{x}\|_\infty$	1.4211×10^{-13}	2.8422×10^{-13}	8.6402×10^{-11}	3.1041×10^{-9}
	$CPUtime(s)$	0.0261	0.0843	0.1506	0.6437

The obtained results show that the algorithm ***COMTRANS*** *gives better absolute error than the other algorithms used in comparison for large values of n. Moreover, the obtained results indicate that the value of the running times for* ***KCOM*** *and* ***ECLS*** *algorithms are small in comparison with other algorithms for large sizes.*

4 Conclusions

In this work, new algorithms, **COMTRANS** and **SCOMTRANS**, have been developed for solving comrade linear systems via transformations. If $det(C) \neq 0$, then the comrade linear system(1.1) has a solution. But with at least one of $\mu_i = 0 (i = 1, 2, 3, ..., n)$ **COMTRANS** algorithm will break down. So the symbolic algorithm, **SCOMTRANS** is developed in order to remove the cases where **COMTRANS** algorithm fails by assigning a $\mu_i = s$ if $\mu_i = 0$. After calculating the symbolic solution, we set $s = 0$ to get the final solution. **SCOMTRANS** algorithm is reliable, computationally efficient and will not fail. Three numerical examples were introduced to explain the effectiveness of our algorithms and competitiveness with other algorithms in literature.

Acknowledgement

The authors would like to express their extreme gratefulness to the referees and editor for their useful comments and suggestions that improved this paper.

Competing Interests

Author have declared that no competing interests exist.

References

[1] Maroulas J, Barnett S. Applications of the comrade matrix to linear multivariable systems theory. Internat. J. Control. 1978;28:129-145.

[2] Aris N, Abd Rahman A. Computing the greatest common divisor of polynomials using the Comrade Matrix In: Lecture Notes Series in Computer Science. Lecture Notes in Artificial Intelligence, Deepak Kapur (Ed.). 2008;5081:87-96.

[3] Barnett S. Division of generalized polynomials using the comrade matrix. Linear Algebra Appl. 1984;60:159-175.

[4] Paprzycki M, Gladwell I. A parallel chopping algorithm for ODE boundary value problems. Parallel Comput. 1993;19(6):651-666.

[5] Golub GH, Van Loan CF. Matrix computations, third ed., The Johns Hopkins University Press. Baltimore and London; 1996.

[6] Rosen KH. Discrete mathematics and its applications. McGraw-Hill, New York; 2007.

[7] Karawia AA. Two algorithms for solving comrade linear systems. Appl. Math. Comput. 2007;189:291-297.

[8] Sogabe T. Numerical algorithms for solving comrade linear systems based on tridiagonal solvers. Appl. Math. Comput. 2008;198:117-122.

[9] Karawia AA. Symbolic algorithm for solving comrade linear systems based on a modified Stair-Diagonal Approach. Appl. Math. Lett. 2013;26:913-918.

[10] Jia J, Kong Q. A novel elementary algorithm for solving comrade linear systems. Appl. Math. Comput. 2014;236:642-646.

Appendix A

In this appendix, We will introduce the proof that the systems (1.1) and (2.6) are equivalent. One can get system(2.6) from system (1.1) by the following proof:
Proof. Assume that $\mu_1 = -\frac{\beta_1}{\alpha_1}$ in the first equation of system (1.1). Then divide the first equation by μ_1. So it will appear as follows:

$$x_1 + \phi_1 x_2 + 0x_3 + ... + 0x_n = z_1$$

where $\phi_1 = \frac{1}{\alpha_1 \mu_1}$ and $z_1 = \frac{y_1}{\mu_1}$.

Now we will use the pivot element in the first equation to eliminate x_1 from second and last equations of system (1.1) as follows:

$$\begin{aligned} Row_2 &\longleftarrow -\tfrac{\gamma_2}{\alpha_2} Row_1 + Row_2 \\ Row_n &\longleftarrow -\sigma_1 Row_1 + Row_n \end{aligned}$$

where $\sigma_1 = -\frac{a_n}{\alpha_n}$ and the symbol $\longleftarrow$ indicates a replacement. So the second and last equations will appear as follows:

$$\begin{aligned} \mu_2 x_2 + \tfrac{1}{\alpha_2}\ x_3 + 0x_4 + ... + 0x_{n-2} + 0x_{n-1} + 0x_n &= -\tfrac{\gamma_2}{\alpha_2} z_1 + y_2 \\ \sigma_2 x_2 - \tfrac{a_{n-2}}{\alpha_n} x_3 - \tfrac{a_{n-3}}{\alpha_n} x_4 - ... - \tfrac{a_3}{\alpha_n} x_{n-2} + \tfrac{\gamma_n - a_2}{\alpha_n} x_{n-1} + \tfrac{-\beta_n - a_1}{\alpha_n} x_n &= \psi_2 \end{aligned}$$

where $\mu_2 = (-\beta_2 - \gamma_2\phi_1)/\alpha_2$, $\sigma_2 = -\sigma_1\phi_1 - \frac{a_{n-1}}{\alpha_n}$, $\psi_2 = -\sigma_1 z_1 + \psi_1$, and $\psi_1 = y_n$.

By the same manner, we will use the pivot element of eqn_i to eliminate x_i from eqn_{i+1} and eqn_n, $i = 2, 3, ..., n-2$. In the final step, we will use eqn_{n-1} to eliminate x_{n-1} from eqn_n. By the end of this step, we get the system (2.6). During this proof, we used the five vectors $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, and $\boldsymbol{Z}$. □

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/15648