MATEMATIKA, 2019, Volume 35, Number 1, 25–38
Penerbit UTM Press. All rights reserved

Modified Cramer's Rule and its Application to Solve Linear Systems in WZ Factorization

[1]Olayiwola Babarinsa* and [2]Hailiza Kamarulhaili
[1,2]School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia
[1]Department of Mathematical Sciences, Federal University Lokoja, P.M.B 1154, Nigeria
*Coresponding author: olayiwola.babarinsa@fulokoja.edu.ng

Article history
Received: 2 May 2018
Received in revised form: 27 August 2018
Accepted: 15 November 2018
Published on line: 1 April 2019

Abstract The proposed modified methods of Cramer's rule consider the column vector as well as the coefficient matrix concurrently in the linear system. The modified methods can be applied since Cramer's rule is typically known for solving the linear systems in WZ factorization to yield Z-matrix. Then, we presented our results to show that there is no tangible difference in performance time between Cramer's rule and the modified methods in the factorization from improved versions of MATLAB. Additionally, the Frobenius norm of the modified methods in the factorization is better than using Cramer's rule irrespective of the version of MATLAB used.

Keywords Linear systems; Cramer's rule; WZ factorization; Z-matrix; Frobenius norm

Mathematics Subject Classification 5A06, 15A23, 15B99.

1 Introduction

If a *linear system* is defined by

$$Bx = c, \tag{1}$$

then the unique solutions $x = (x_1, x_2, ..., x_n)^T$, where B is the coefficient matrix of the system and c the column vector, to equation (1) can be obtained from an old method called *Cramer's rule* [1]. Cramer's rule has many drawbacks: fails when the coefficient matrix is singular, requires $(n+1)$ determinants in its computation and has high computational time [2]. Cramer's rule and Gaussian elimination (GE) requires about the same amount of arithmetic for finding the solution of 2×2 linear systems. Moler [3] expressed that Cramer's rule is unsatisfactory even for 2×2 linear systems because of round off errors. However, Dunham [4] gives a counter example to the statement to show that Cramer's rule is satisfactory. Thus, accurate methods to evaluate determinants make Cramer's rule numerically stable [5]. Notwithstanding its high computational complexity, Cramer's rule is truly intriguing and it is of hypothetical significance for solving linear systems [6]. One of the advantages of Cramer's rule is its application in

WZ factorization or quadrant interlocking factorization (QIF) to check if the matrix being factorized has non-singular inner submatrix (centro-nonsingular) and to solve its linear systems [7, 8].

WZ *factorization* is first proposed by Evans and Hatzopoulos [9]. During WZ factorization of non-singular matrix B, Z*-matrix* exists together with W-matrix [10], such that

$$B = WZ. \tag{2}$$

The matrix norm of WZ factorization is the *Frobenius norm* of matrix B given as

$$||B||_F = \|B - WZ\|.$$

WZ factorization is known for the adaptability of its direct method for solving linear systems on shared memory parallel computers with many integrated core, see [11–14] and the references therein. WZ factorization has been shown to be better than the GE and LU factorization irrespective of the number of processors used and is also better on Intel processors than AMD processors, see [15, 16]. WZ factorization is extensively applied in finding the numerical solutions of Markov chains [17, 18]. Though, the efficiency of WZ factorization depends on an efficacious use of the memory echelon because computational cost often relies on both the total number of arithmetic operations used and the data transferring time between different memory levels [19]. For the WZ factorization, we compute $w^{*(k)}_{i,k}$ and $w^{*(k)}_{i,n-k+1}$ from equation (3) by solving its 2×2 linear systems via Cramer's rule for every update of matrix B,

$$\begin{cases} z^{(k-1)}_{k,k} w^{*(k)}_{i,k} + z^{(k-1)}_{n-k+1,k} w^{*(k)}_{i,n-k+1} = -z^{(k-1)}_{i,k} \\ z^{(k-1)}_{k,n-k+1} w^{*(k)}_{i,k} + z^{(k-1)}_{n-k+1,n-k+1} w^{*(k)}_{i,n-k+1} = -z^{(k-1)}_{i,n-k+1}, \end{cases} \tag{3}$$

where $k = 1, 2, ..., \lfloor \frac{n}{2} \rfloor$; $i = k+1, ..., n-k$. Then, we update matrix B for every computed ith to the $(n-1)$th element of the $(i-1)$th and $(n-i+1)$th column of W-matrix, see [20].

In Section 2, we propose two modified methods from Cramer's rule and show how they are equivalent to classical Cramer's rule. While Section 3 applies the proposed methods of Cramer's rule in WZ factorization. To obtain the matrix norm of the factorization, we evaluate the Frobenius norm of the applied modified methods and Cramer's rule. Furthermore, the performance time of the proposed methods and Cramer's rule in the factorization were compared. Due to the lack of parallel computer or mesh multiprocessors, the MATLAB codes application of this article are limited to Intel processor (Core i7-4600U 2.1GHz).

2 Modified Cramer's Rule

Theorem 1 (Cramer's rule) *Let $Bx = c$ be an $n \times n$ system of linear equation and B an $n \times n$ non-singular matrix, then the unique solution $x = (x_1, x_2, ..., x_n)^T$ to the linear system is given by*

$$x_i = \frac{det(B_{i|c})}{det(B)}. \tag{4}$$

Where $B_{i|c}$ is the matrix obtained from B by substituting the column vector c to the ith column of B, for $i = 1, 2, ..., n$.

It is a well-established theorem that if the ith column of matrix B is the sum or difference of the ith column of matrix C and the ith column of matrix D and all other columns in C and D are equal to the corresponding columns in B. Then

$$det(B) = det(C) \pm det(D). \tag{5}$$

In addition, if for $i = 1, 2, ..., n$, the column matrix B is replaced with the row sum of its matrix to obtain a new matrix B^{α_i} with all other columns in B and B^{α_i} remain the same, then the determinant of the matrix and the obtained matrix are equal [21]. That is,

$$det(B) = det(B^{\alpha_i}). \tag{6}$$

Now, we can deduce that if column vector c is added to or subtracted from the ith column of matrix B^{α_i} (i.e the ith column of matrix B where its row sum replaced), then we can re-write equation (5) via (6) as

$$det(B^{\alpha_i}_{i\pm c}) = det(B^{\alpha_i}) \pm det(B^{\alpha_i}_{i|c}), \tag{7}$$

where $B^{\alpha_i}_{i\pm c}$ is the matrix obtained from B^{α_i} by adding (or subtracting) the column vector c to (or from) the ith column of B^{α_i}, $B^{\alpha_i}_{i|c}$ is the matrix obtained from B^{α_i} by substituting column vector c to the ith column of B^{α_i}. While B^{α_i} is the matrix obtained from matrix B, where the ith column of matrix B is replaced by its row sum. It is important to note note that if $det(B) = det(B^{\alpha_i})$, then

$$det(B_{i|c}) = det(B^{\alpha_i}_{i|c}). \tag{8}$$

2.1 Method I

Corollary 1 *Let $Bx = c$ be an $n \times n$ system of linear equation and B a square matrix of x, then the ith entry x_i of the unique solution $x = (x_1, x_2, ..., x_n)^T$ to the linear system is given by*

$$x_i = \frac{det(B^{\alpha_i}_{i+c})}{det(B^{\alpha_i})} - 1, \tag{9}$$

where $B^{\alpha_i}_{i+c}$ is the matrix obtained from B^{α_i} by adding the column vector c to the ith column of B^{α_i} and B^{α_i} is the matrix obtained from B with its ith column being replaced by the row sum of B, for $i = 1, 2, ..., n$.

Proof First, we assume that matrix B is non-singular. Now, we consider only the positive part of equation (7) to prove Corollary 1. We have

$$det(B^{\alpha_i}_{i+c}) = det(B^{\alpha_i}) + det(B^{\alpha_i}_{i|c}). \tag{10}$$

Then, we solve for $det(B^{\alpha_i}_{i|c})$ from equation (10) and substitute it in equation (8) to have

$$det(B_{i|c}) = det(B^{\alpha_i}_{i+c}) - det(B^{\alpha_i}). \tag{11}$$

Now, substitute equation (11) and equation (6) in equation (4) to get

$$\begin{aligned} x_i &= \frac{det(B^{\alpha_i}_{i+c}) - det(B^{\alpha_i})}{det(B^{\alpha_i})} \\ &= \frac{det(B^{\alpha_i}_{i+c})}{det(B^{\alpha_i})} - 1. \end{aligned}$$

□

2.2 Method II

Corollary 2 *Let $Bx = c$ be an $n \times n$ system of linear equation and B a square matrix of x, then the ith entry x_i of the unique solution $x = (x_1, x_2, ..., x_n)^T$ to the linear system is given by*

$$x_i = 1 - \frac{det(B^{\alpha_i}_{i-c})}{det(B^{\alpha_i})}, \tag{12}$$

where $B^{\alpha_i}_{i-c}$ is the matrix obtained from B^{α_i} by subtracting the column vector c from the ith column of B^{α_i} and B^{α_i} is the matrix obtained from B with its ith column being replaced by the row sum of B for $i = 1, 2, ..., n$.

Proof Now, we consider the negative part of equation (7) based on Corollary 2 assumption to have

$$det(B^{\alpha_i}_{i-c}) = det(B^{\alpha_i}) - det(B^{\alpha_i}_{i|c}). \tag{13}$$

Then, we solve for $det(B^{\alpha_i}_{i|c})$ from equation (13) and substitute it in equation (8) to get

$$det(B_{i|c}) = det(B^{\alpha_i}) - det(B^{\alpha_i}_{i-c}). \tag{14}$$

Then, we substitute equation (14) and equation (6) in equation (4) to have

$$\begin{aligned} x_i &= \frac{det(B^{\alpha_i}) - det(B^{\alpha_i}_{i-c})}{det(B^{\alpha_i})} \\ &= 1 - \frac{det(B^{\alpha_i}_{i-c})}{det(B^{\alpha_i})}. \end{aligned}$$

□

We give the steps for implementing Method I and Method II in Algorithm 1, while their MATLAB codes are provided in Listings 1 and 2 respectively.

Listing 1: MATLAB Code of Method I

```
function x=methodI(B,c)
B=input('matrix B =');
c=input('vector c =');
n=size(B,1);
m=size(B,2);
if n~=m
    Error('The matrix is not square.');
    x=[];
else
    if det(B)~=0
        x=zeros(n,1);
        k1=sum(B,2);
        k2=k1+c;
        for j=1:n
            if j~=1 && j~=n
                Bc=[B(:,1:j-1) k2 B(:,j+1:n)];
                D=[B(:,1:j-1) k1 B(:,j+1:n)];
```

Algorithm 1 Modified Cramer's rule algorithm

```
1:  procedure MODIFIED CRAMER'S RULE
2:      $B \leftarrow n \times n$ coefficient matrix
3:      $c \leftarrow$ column vector
4:      $x_i \leftarrow$ solutions of linear system
5:      for i do1n
6:          $D \leftarrow$ row sum of $B$
7:          $E \leftarrow D \pm c$.
8:          $F_i \leftarrow$ replace $i$th column of $B$ with $E$.
9:          $S_i \leftarrow$ replace $i$th column of $B$ with $D$.
10:         $det(S) \leftarrow$ determinant of $S_i$ .
11:         $det(F) \leftarrow$ determinant of $F_i$.
12:         if $E \leftarrow D + c$ then
13:             $x_i \leftarrow 1 - (det(F)/det(S))$
14:         else
15:             $x_i \leftarrow (det(F)/det(S)) - 1$
16:         end if
17:     end for
18: end procedure
```

```
              elseif j==1
                  Bc=[k2 B(:,2:n)];
                  D=[k1 B(:,2:n)];
              elseif j==n
                  Bc=[B(:,1:n-1) k2];
                  D=[B(:,1:n-1) k1];
              end
               detD=det(D);
              x(j)=(det(Bc)/detD)-1;
          end
      else
          Error('Matrix B is singular.');
           x=[];
      end
  end
```

Listing 2: MATLAB Code of Method II

```
function x=methodII(B,c)
B=input('matrix B =');
c=input('vector c =');
n=size(B,1);
m=size(B,2);
if n~=m
    Error('The matrix is not square.');
    x=[];
else
    if det(B)~=0
```

```
        x=zeros(n,1);
        k3=sum(B,2);
        k4=k3-c;
        for j=1:n
            if j~=1 && j~=n
                Bc=[B(:,1:j-1) k4 B(:,j+1:n)];
                E=[B(:,1:j-1) k3 B(:,j+1:n)];
            elseif j==1
                Bc=[k4 B(:,2:n)];
                E=[k3 B(:,2:n)];
            elseif j==n
                Bc=[B(:,1:n-1) k4];
                E=[B(:,1:n-1) k3];
            end
            detE=det(E);
            x(j)=1-(det(Bc)/detE);
        end
    else
        Error('Matrix B is singular.');
         x=[];
    end
end
```

Proposition 1 *Let $Bx = c$ be an $n \times n$ system of linear equation, where B is an $n \times n$ non-singular matrix of x for the distinct solution of $x = (x_1, x_2, ..., x_n)^T$ and c the column vector. If $x_i = 1 - \frac{det(B^{\alpha_i}_{i-c})}{det(B^{\alpha_i})}$ when the column vector c is subtracted from the column of matrix B^{α_i} and $x_i = \frac{det(B^{\alpha_i}_{i+c})}{det(B^{\alpha_i})} - 1$ when the column vector c is added to the column of matrix B^{α_i}. Then*

$$1 - \frac{det(B^{\alpha_i}_{i-c})}{det(B^{\alpha_i})} = \frac{det(B_{i|c})}{det(B)} = \frac{det(B^{\alpha_i}_{i+c})}{det(B^{\alpha_i})} - 1, \tag{15}$$

where $B_{i|c}$ is the matrix obtained from B by substituting the column vector c to the ith column of B and B^{α_i} is the matrix obtained from B from its ith column being replaced by the row sum of B for $i = 1, 2, ..., n$.

Proof We begin by substituting equation (10) in equation (9) of Corollary 1 to obtain

$$x_i = \frac{det(B^{\alpha_i}) + det(B^{\alpha_i}_{i|c})}{det(B^{\alpha_i})} - 1. \tag{16}$$

Substitute equation (6) and equation (8) in equation (16) to have Cramer's rule

$$x_i = \frac{det(B_{i|c})}{det(B)}. \tag{17}$$

Similarly, substitute equation (13) in equation (12) to get

$$x_i = 1 - \frac{det(B^{\alpha_i}) - det(B^{\alpha_i}_{i|c})}{det(B^{\alpha_i})}. \tag{18}$$

We will obtain Cramer's rule if equation (6) and equation (8) are substituted in equation (18). Thus, equations (16) and (18) are equivalent. □

It is apparent that Cramer's rule has better computational time than its modifications due to less computation operations. The modified methods may only be compared with Cramer's rule, such as their residual error measurements, but our objective is to apply the methods in WZ factorization. Moreover, the advantage of these methods like Cramer's rule over direct methods (such as GE or LU decomposition) is that they indicate if a system is incompatible or indeterminate without completely solving the systems. This specific advantage is useful in solving the 2×2 linear systems in WZ factorization.

3 Application of Modified Cramer's Rule in *WZ* Factorization

The MATLAB code to compute the elements of W-matrix and Z-matrix is given in [20]. However, for future research of interested readers, we give the complete MATLAB code of WZ factorization in Listing 3.

Listing 3: MATLAB Code for WZ Factorization

```
function Z = WZfactorization(B,W,Z)
% step of elimination - from B to Z
B=input('matrix B =');
n = size(B, 1);
W = zeros(n);
for k = 1:ceil((n-1)/2)
   k2 = n - k + 1 ;
   determinant = B(k,k) * B(k2,k2) - B(k2,k) * B(k,k2);
   if determinant == 0
       exitflag = 0;
       for i1 = k:k2
           for i2 = i1:k2
               determinant = B(i1,k) * B(i2,k2) - B(i2,k) * B(i1,k2);
               if determinant ~= 0
                   tmp = B(i1,k:k2);
                   B(i1,k:k2) = B(k,k:k2);
                   B(k,k:k2) = tmp;
                   tmp = B(i2,k:k2);
                   B(i2,k:k2) = B(k2,k:k2);
                   B(k2,k:k2) = tmp;
                   exitflag = 1;
                   break
               end  % end if determinant ~= 0
           end   % end of i2
       end    % end of i1
       if exitflag == 0
           Z = B;
           return
       end
   end  % end if determinant == 0
 % finding elements of W
 W(k+1:k2-1,k)=(B(k2,k2)*B(k+1:k2-1,k)-B(k2,k)*B(k+1:k2-1,k2))/determinant;
```

```
33   W(k+1:k2-1,k2)=(B(k,k2)*B(k+1:k2-1,k)-B(k,k)*B(k+1:k2-1,k2))/determinant;
34    for m=1:n
35       W(m,m)=1;
36       W(m,n+1-m);
37   % updating B
38   B(k+1:k2-1,k) = 0;
39   B(k+1:k2-1,k2) = 0;
40   B(k+1:k2-1,k+1:k2-1) = B(k+1:k2-1,k+1:k2-1) + W(k+1:k2-1,k)* B(k,k+1:k2-1)
41              + W(k+1:k2-1,k2) * B(k2,k+1:k2-1);
42     Z = B;
43 end
```

To apply the modified methods of Cramer's rule in WZ factorization, we begin by expressing equation (3) like equation (1) as

$$\begin{bmatrix} z_{k,k}^{(k-1)} & z_{n-k+1,k}^{(k-1)} \\ z_{k,n-k+1}^{(k-1)} & z_{n-k+1,n-k+1}^{(k-1)} \end{bmatrix} \begin{bmatrix} w_{i,k}^{*(k)} \\ w_{i,n-k+1}^{*(k)} \end{bmatrix} = \begin{bmatrix} -z_{i,k}^{(k-1)} \\ -z_{i,n-k+1}^{(k-1)} \end{bmatrix} \tag{19}$$

and then compute $w_{i,k}^{*(k)}$ and $w_{i,n-k+1}^{*(k)}$ using method I and method II respectively. To do this, we first apply Corollary 1 to compute $w_{i,k}^{*(k)}$ and $w_{i,n-k+1}^{*(k)}$ in equation (19) as

$$w_{i,k}^{*(k)} = \frac{det\left(\left[w_{i,k}^{*(k)}\right]_{1+c}^{\alpha_1}\right)}{det\left(z^{\alpha_1}\right)} - 1 \qquad \text{and} \qquad w_{i,n-k+1}^{*(k)} = \frac{det\left(\left[w_{i,n-k+1}^{*(k)}\right]_{2+c}^{\alpha_2}\right)}{det\left(z^{\alpha_2}\right)} - 1, \tag{20}$$

where

$$\begin{aligned}
det\left(z^{\alpha_1}\right) = det\left(z^{\alpha_2}\right) &= -\, z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} \\
det\left(\left[w_{i,k}^{*(k)}\right]_{1+c}^{\alpha_1}\right) &= z_{n-k+1,k}^{(k-1)} z_{i,n-k+1}^{(k-1)} + z_{n-k+1,k}^{(k-1)} z_{n-k+1,n-k+1}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} \\
&\quad + z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} - z_{n-k+1,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} - z_{n-k+1,n-k+1}^{(k-1)} z_{n-k+1,k}^{(k-1)} \\
&= z_{n-k+1,k}^{(k-1)} z_{i,n-k+1}^{(k-1)} - z_{n-k+1,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} \\
det\left(\left[w_{i,n-k+1}^{*(k)}\right]_{2+c}^{\alpha_2}\right) &= z_{k,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} - z_{i,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} \\
&\quad + z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} - z_{k,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} + z_{k,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} \\
&= z_{k,n-k+1}^{(k-1)} z_{i,k}^{(k-1)} - z_{i,n-k+1}^{(k-1)} z_{k,k}^{(k-1)} - z_{n-k+1,k}^{(k-1)} z_{k,n-k+1}^{(k-1)} + z_{n-k+1,n-k+1}^{(k-1)} z_{k,k}^{(k-1)}.
\end{aligned}$$

The W^*-matrix obtained from equation (20) will be referred to as W^{m_1}-matrix and its counterpart Z-matrix as Z^{m_1}-matrix. The $W^{m_1}Z^{m_1}$ factorization is the factorization obtained from using Corollary 1. For the MATLAB code of $W^{m_1}Z^{m_1}$ factorization, we replace line 32 and line 33 in Listing 3 with line 2 to line 5 of Listing 4.

Listing 4: MATLAB Code for $W^{m_1}Z^{m_1}$ Factorization

```
1    %finding elements of W
2    W(k+1:k2-1,k)=((B(k2,k)*B(k+1:k2-1,k2)-B(k2,k2)*B(k+1:k2-1,k)
                  −B(k2,k)*B(k,k2)+B(k2,k2)*B(k,k))/determinant)−1;
   W(k+1:k2−1,k2)=((B(k,k2)*B(k+1:k2−1,k)−B(k+1:k2−1,k2)*B(k,k)
                     −B(k2,k)*B(k,k2) +B(k2,k2)*B(k,k))/determinant)−1;
```

Furthermore, if we apply Corollary 2 to compute $w^{*(k)}_{i,k}$ and $w^{*(k)}_{i,n-k+1}$ in equation (19) then

$$w^{*(k)}_{i,k} = 1 - \frac{det\left(\left[w^{*(k)}_{i,k}\right]^{\alpha_1}_{1-c}\right)}{det\left(z^{\alpha_1}\right)} \qquad \text{and} \qquad w^{*(k)}_{i,n-k+1} = 1 - \frac{det\left(\left[w^{*(k)}_{i,n-k+1}\right]^{\alpha_2}_{2-c}\right)}{det\left(z^{\alpha_2}\right)}, \tag{21}$$

where

$$\begin{aligned}
det\left(z^{\alpha_1}\right) = det\left(z^{\alpha_2}\right) = &- z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{k,n-k+1} + z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{k,k} \\
det\left(\left[w^{*(k)}_{i,k}\right]^{\alpha_1}_{1-c}\right) =& z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{n-k+1,k} + z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{k,k} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{i,n-k+1} \\
&+ z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{i,k} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{n-k+1,n-k+1} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{k,n-k+1} \\
=& z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{i,k} + z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{k,k} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{i,n-k+1} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{k,n-k+1} \\
det\left(\left[w^{*(k)}_{i,n-k+1}\right]^{\alpha_2}_{2-c}\right) =& z^{(k-1)}_{k,n-k+1} z^{(k-1)}_{k,k} + z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{k,k} - z^{(k-1)}_{k,n-k+1} z^{(k-1)}_{i,k} \\
&+ z^{(k-1)}_{i,n-k+1} z^{(k-1)}_{k,k} - z^{(k-1)}_{k,k} z^{(k-1)}_{k,n-k+1} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{k,n-k+1} \\
=& z^{(k-1)}_{i,n-k+1} z^{(k-1)}_{k,k} + z^{(k-1)}_{n-k+1,n-k+1} z^{(k-1)}_{k,k} - z^{(k-1)}_{k,n-k+1} z^{(k-1)}_{i,k} - z^{(k-1)}_{n-k+1,k} z^{(k-1)}_{k,n-k+1}.
\end{aligned}$$

The W^*-matrix obtained from equation (21) will be referred to as W^{m_2}-matrix and its Z-matrix as Z^{m_2}-matrix. The $W^{m_2}Z^{m_2}$ factorization is the factorization obtained from using Corollary 2. For the MATLAB code of $W^{m_2}Z^{m_2}$ factorization, we replace line 32 and line 33 in Listing 3 with line 2 to line 5 of Listing 5.

Listing 5: MATLAB Code for $W^{m_2}Z^{m_2}$ Factorization

```
1  %finding elements of W
2    W(k+1:k2−1,k)=1−((B(k2,k2)*B(k+1:k2−1,k)+B(k2,k2)*B(k,k)
3                 −B(k2,k)*B(k+1:k2−1,k2) −B(k2,k)*B(k,k2))/determinant);
4    W(k+1:k2−1,k2)=1−(B(k+1:k2−1,k2)*(B(k,k)+B(k2,k2)*B(k,k)
5                       −B(k,k2)*B(k+1:k2−1,k)−B(k2,k)*B(k,k2))/determinant);
```

The algorithms of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization are implemented on Intel processor (Core i7-4600U 2.1GHz) with standard hardware. Then, we investigate the performance time of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization on different versions of MATLAB (R2013b, R2015b and R2017b respectively) and the results were recorded in Table 1.

In Figure 1, 2 and 3, the performance time of $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization are similar but higher than performance time of WH factorization for all the versions of MATLAB used. However, the performance time of WH, $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization reduces as the versions of MATLAB improve. The performance time of WH, $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization decreases about 16% for MATLAB R2015b and decreases about 32% for MATLAB R2017b. Figure 3 shows the performance time of $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization

Table 1: Performance Time of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization

Matrix size	MATLAB R2013b			MATLAB R2015b			MATLAB R2017b		
	WZ	$W^{m_1}Z^{m_1}$	$W^{m_2}Z^{m_2}$	WZ	$W^{m_1}Z^{m_1}$	$W^{m_2}Z^{m_2}$	WZ	$W^{m_1}Z^{m_1}$	$W^{m_2}Z^{m_2}$
500×500	7.26	9.65	9.45	5.29	6.56	7.06	2.25	2.76	3.15
1000×1000	19.95	28.24	27.45	17.01	19.51	18.93	14.28	13.88	14.26
1500×1500	48.93	63.98	70.26	41.43	42.56	41.91	40.12	40.40	40.21
2000×2000	135.84	198.15	205.88	117.33	123.84	127.10	88.32	89.66	86.22
2500×2500	271.03	342.54	351.46	226.01	249.14	253.93	163.83	173.24	181.21
3000×3000	498.27	552.89	563.01	423.54	441.34	448.34	281.44	294.72	301.69
3500×3500	673.90	749.30	762.91	593.56	623.16	631.90	439.30	458.97	462.56
4000×4000	968.43	1032.40	1051.13	882.23	921.57	927.67	656.59	672.19	679.41
4500×4500	1643.64	1730.84	1753.39	1394.23	1430.55	1436.01	998.10	1013.20	1019.08
5000×5000	2239.65	2305.10	2329.12	1739.87	1809.54	1820.47	1263.74	1274.60	1281.69

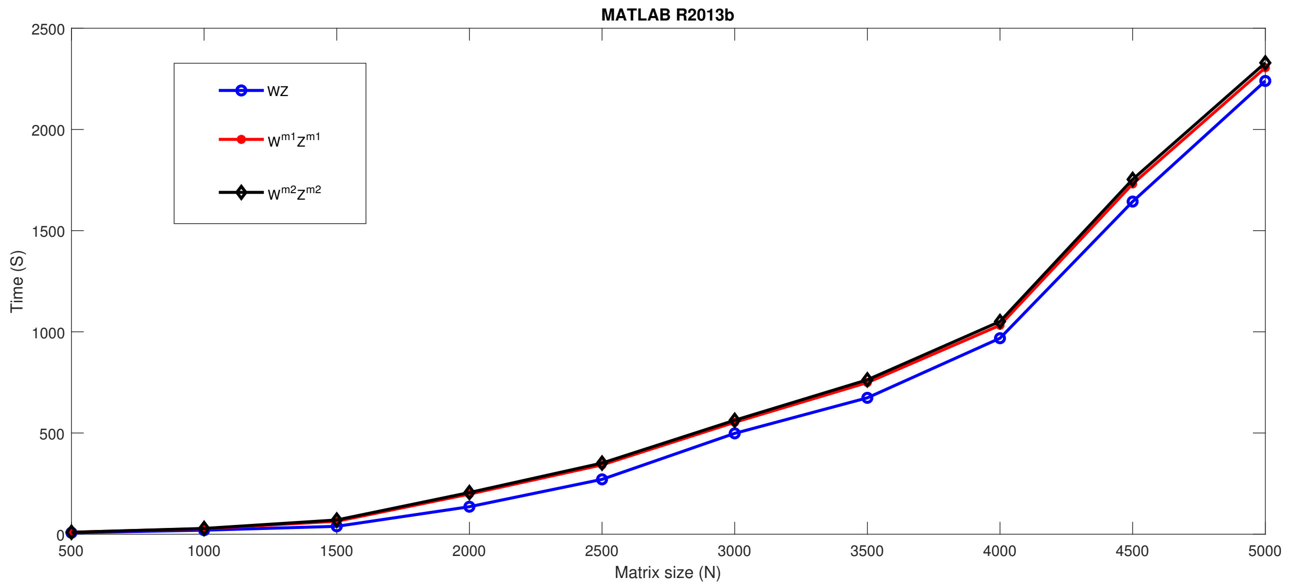

Figure 1: Execution Time of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2013b

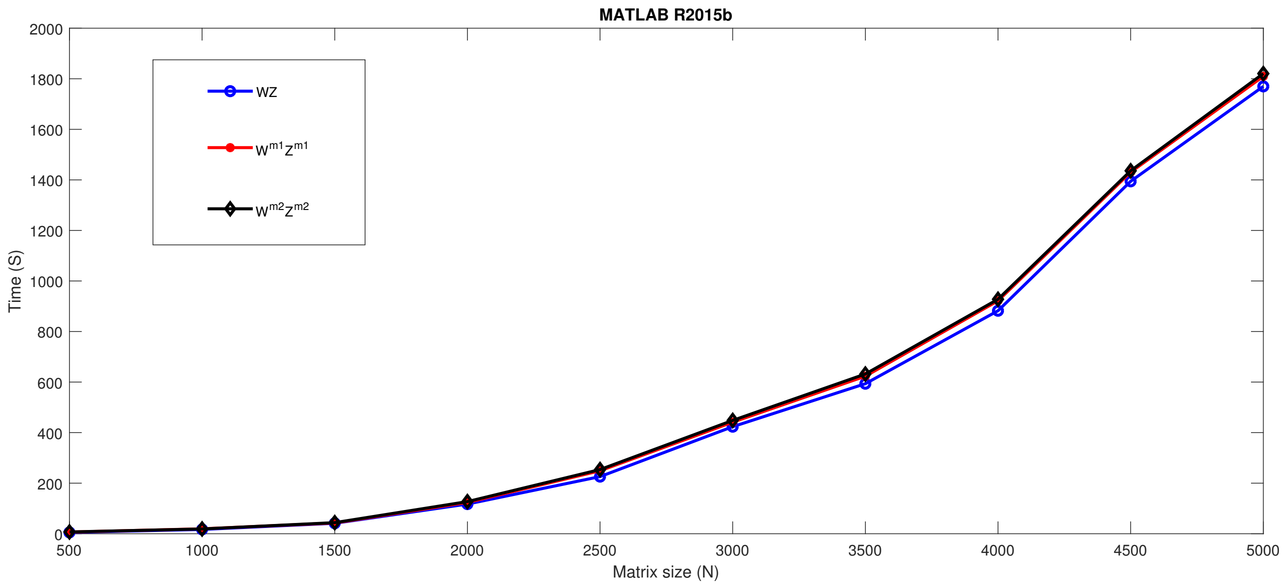

Figure 2: Execution Time of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2015b

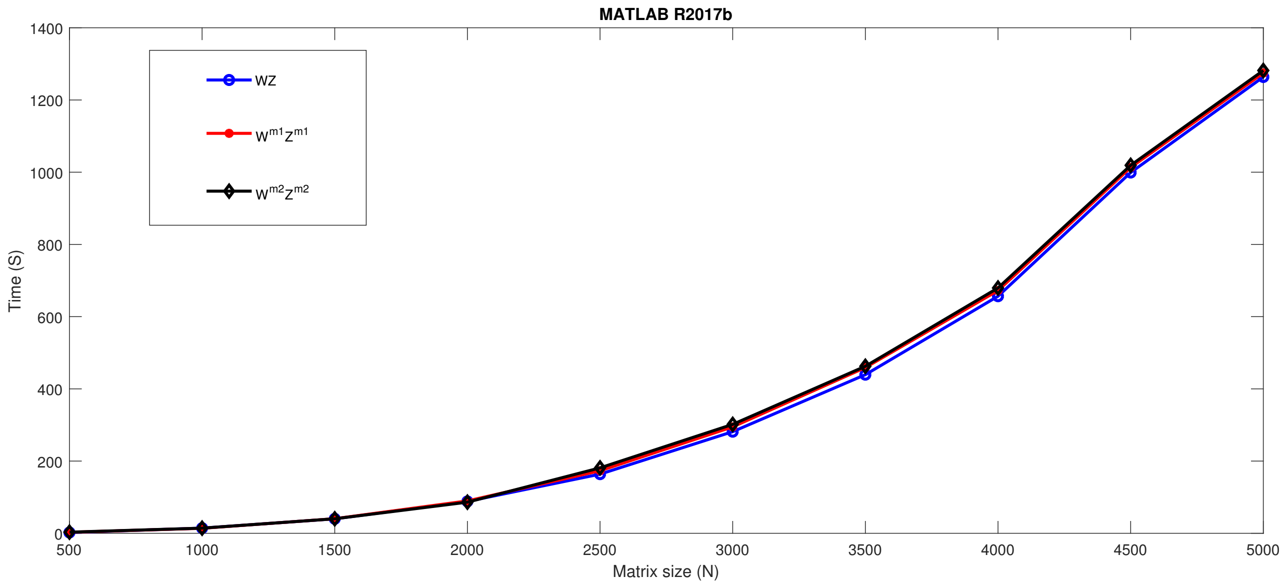

Figure 3: Execution Time of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b

approaches WH factorization. In spite of more arithmetic operations used in $W^{m_1}H^{m_1}$ and $W^{m_2}H^{m_2}$ factorization as a result of applying Corollary 1 and Corollary 2 respectively, we can deduce that a better version of MATLAB in the near future will show that there would be no tangible difference in performance time between WH, $W^{m_1}H^{m_1}$ and the $W^{m_2}H^{m_2}$ factorization.

In Table 2, we only display our result from MATLAB R2017b because our background analysis shows that irrespective of the version of MATLAB used, the Frobenius norms of $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization are better than WZ factorization, see Figure 4. More so, the Frobenius norms of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization increase as the size of their matrices increase.

Table 2: Matrix Norm of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b

Matrix size (N)	$\|B - WZ\|$	$\|B - W^{m_1}Z^{m_1}\|$	$\|B - W^{m_2}Z^{m_2}\|$
500 × 500	5.33E-14	4.92E-14	4.91E-14
1000 × 1000	1.18E-13	0.92E-13	0.90E-13
1500 × 1500	5.67E-13	4.76E-13	4.76E-13
2000 × 2000	7.89E-13	6.73E-13	6.71E-13
2500 × 2500	1.28E-12	0.97E-12	0.94E-12
3000 × 3000	1.39E-12	0.95E-12	0.96E-12
3500 × 3500	3.06E-12	2.36E-12	2.36E-12
4000 × 4000	3.93E-12	2.71E-12	2.73E-12
4500 × 4500	5.51E-12	4.83E-12	4.80E-12
5000 × 5000	6.52E-12	5.25E-12	5.26E-12

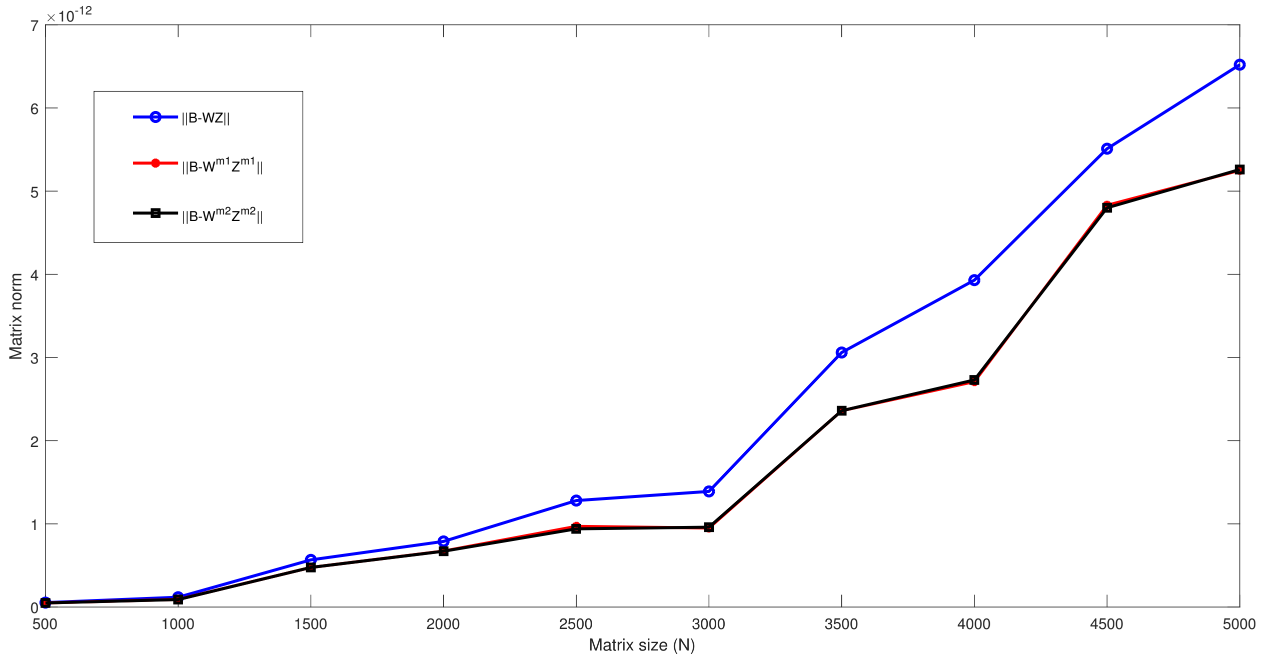

Figure 4: Matrix Norms of WZ, $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ Factorization on MATLAB R2017b

4 Conclusion

Like Cramer's rule, the modified methods may be impracticable and inappropriate for higher linear systems. However, we pointed out their advantage over Cramer's rule in quadrant interlocking factorization even though they are theoretically equivalent. The Frobenius norm of the applied modified Cramer's rule over classical Cramer's rule in the factorization shows that $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization would be better in numerical solution of Markov chains. Therefore, we passionately advocate that $W^{m_1}Z^{m_1}$ and $W^{m_2}Z^{m_2}$ factorization should be compared with WZ factorization on parallel computer or mesh multiprocessors such as Intel Xeon Phi, Sunway Taihulight or OLCF-4.

Acknowledgments

This research is funded by RU (Research University) Grant, Universiti Sains Malaysia, Grant number 1001/PMATHS/811337.

References

[1] Babarinsa, O. and Kamarulhaili, H. Modification of cramer's rule. *J. Fundam. Appl. Sci.*. 2017. 9(5): 556–567.

[2] Debnath, L. A brief historical introduction to matrices and their applications. *Internat. J. Math. Ed. Sci. Tech.*. 2013. 45(3): 360–377.

[3] Moler, C. Cramer's rule on 2-by-2 systems. *ACM SIGNUM Newsletter*. 1974. 9(4): 13–14.

[4] Dunham, C. Cramer's rule reconsidered or equilibration desirable. *ACM SIGNUM Newsletter*. 1980. 15(4): 9–9.

[5] Habgood, K. and Arel, I. A condensation-based application of cramer's rule for solving large-scale linear systems. *Journal of Discrete Algorithms*. 2012. 10: 98–109.

[6] Brunetti, M. and Renato, A. Old and new proofs of cramer's rule. *Appl. Math. Sci.*. 2014. 8(133): 6689–6697.

[7] Levin, D. and Evans, D. The inversion of matrices by the double-bordering algorithm on mimd computers. *Parallel Comput.*. 1991. 17(4): 591–602.

[8] Babarinsa, O. and Kamarulhaili, H. Quadrant interlocking factorization of hourglass matrix. In *AIP Conference Proceedings of the 25th National Symposium on Mathematical Sciences*. AIP Publishing. 2018. 030009:1–9.

[9] Evans, D. and Hatzopoulos, M. A parallel linear system solver. *Int. J. Comput. Math.* 1979. 7(3): 227–238.

[10] Rhofi, K., Ameur, M. and Radid, A. Double power method iteration for parallel eigenvalue problem. *Int. J. Pure Appl. Math.* 2016. 108(4): 945–955.

[11] Shanehchi, J. and Evans, D. Further analysis of the quadrant interlocking factorisation (qif) method. *Int. J. Comput. Math.* 1982. 11(1): 49–72.

[12] Chudik, J., David, G., Kotov, V., Mirenkov, N., Ondas, J., Plander, I. and Valkovskii, V. *Algorithms, software and hardware of parallel computers*. Springer Science and Business Media. 2013.

[13] Golpar-Raboky, E. Abs algorithms for integer wz factorization. *Malaysian J. Math. Sci.* 2014. 8(1): 69–85.

[14] Heinig, G. and Rost, K. Fast algorithms for toeplitz and hankel matrices. *Linear Algebra Appl.* 2011. 435(1): 1–59.

[15] Evans, D. and Abdullah. The parallel implicit elimination (pie) method for the solution of linear systems. *Parallel Algorithms Appl.* 1994. 4(1): 153–162.

[16] Bylina, B. and Bylina, J. Mixed precision iterative refinement techniques for the wz factorization. In *Federated Conference on Computer Science and Information Systems.* IEEE. 2013. 425–431.

[17] Bylina, B. The inverse iteration with the wz factorization used to the markovian models. *Annales UMCS Informatica AI.* 2015. 2(1): 15–23.

[18] Von Hilgers, P. and Langville, A. N. The five greatest applications of markov chains. In *Proceedings of the Markov Anniversary Meeting.* Boston Press. 2006. 155–168.

[19] Bylina, B. The block wz factorization. *J. Comput. Appl. Math.* 2018. 331: 119–132.

[20] Bylina, B. and Bylina, J. The wz factorization in matlab. In *Federated Conference on Computer Science and Information Systems.* IEEE. 2014. 561–568.

[21] Aitken, A. *Determinants and matrices.* Edinburgh; New York: Oliver and Boyd; Interscience Publishers. 1956.