The Adaptive Reducing Methods of Calculating Determinant

Puttha Sakkaplangkul and Nattaporn Chuenjarern

***Abstract*—This paper introduces a method for determinant computation in square matrices. Our approach utilized recursion and the Schur formula to partition the matrix into submatrices. Determinant calculations were performed using the condensation method. To evaluate its computational efficiency, we conducted a floating-point operation per second (FLOPS) analysis, using FLOPS to compare the efficiency of all reduction methods. Pseudocode was provided to demonstrate the computational efficiency of our method in terms of flops and execution time for square matrix determinant calculations.**

***Index Terms*—Determinant, Gaussian elimination method, Floating-point operations per second, Execution time.**

I. Introduction

THE determinant is a fundamental concept in linear algebra and matrix theory, playing a pivotal role in numerous scientific and engineering applications. It is widely used in multivariate function integration, solving complex linear systems, and calculating the area and volume of geometric shapes. While Leibniz's formula and the Laplace expansion can frequently be used to evaluate the determinant of small matrices, the situation drastically changes when dealing with large matrices. In such cases, difficulties with permutation selection and matrix multiplications, which are necessary to identify determinants by conventional methods, arise. These challenges result in increased computational complexity. To address these issues, several determinant computation methods have been extensively examined, as documented in [1]. This includes the Cholesky decomposition method, the LU decomposition, the QR decomposition, and the Gaussian elimination method.

Calculating the determinants of the matrix is made simpler by the condensation method which reduces a large matrix into smaller submatrices. The two most popular methods are Chio's condensation method and Dodgson's condensation method [2]–[5]. The main idea of these two methods is to reduce the size of the original matrix to a matrix consisting of a matrix of size 2×2. The benefit of this idea is that we do not have to calculate the large size of a matrix. Recently, the new method called generalized Dodgson's condensation method reduces the order of a matrix [2]. This method is based on the basic idea of Chio's condensation method and Dodgson's condensation method. The idea of generalized Dodgson's condensation method is to reduce the original matrix of size $n\times n$ to the 2×2 matrices whose elements are determinants of matrices of size $(n-1)\times(n-1)$. However, Chio's condensation method and Dodgson's condensation method reduce the original matrix of size $n\times n$ to the $(n-1)\times(n-1)$ matrix involving determinants of a matrix of size 2×2 while the generalized Dodgson's condensation method computes the determinant of a matrix only size 2×2.

Different methods can have different performance characteristics. To determine the performance of the method finding the determinant, one approach to evaluating the performance is counting floating-point operations per second (FLOPS). In computing, FLOPS is a computer performance measure that is frequently used to compare the processing ability of various systems by measuring the number of basic arithmetic operations required to perform a calculation. In recent years, researchers have used FLOPS as a metric to compare the performance of algorithms and schemes presented in [6]–[9]. In the case of matrix determinant calculation, the traditional method of finding the determinant of a matrix, known as Gaussian elimination, requires $\mathcal{O}(n^3)$ flops for an $n\times n$ matrix. In 2007, Rezaifar introduced [1] the reduction method to calculate the determinant of a matrix. In this approach, the matrix is divided into four submatrices, and its determinant is calculated by using the determinants of those submatrices. This method is more effective than Laplace's method since it reduces the size of the matrix being considered, making the calculation more manageable. However, Rezaifar's method might be less effective if it is applied to the large size of a matrix.

In this paper, we present an adapted method for determinant calculation, which involves reducing the matrix size through a recursive procedure. This adaptation is inspired by PN's method [10] and is aimed at enhancing the performance of Rezaifar's method. The adapted method divides the matrix into four submatrix blocks, with the initial main diagonal consisting of a matrix of size 1×1. To calculate the determinant, we employ Chio's condensation method for submatrix determinants within the Schur formula [11], [12]. We provide an analysis of the computational complexity, particularly in terms of flops. Our findings indicated that this method requires fewer flops than Gaussian elimination but slightly more than Chio's method, with only 2 flops, regardless of the matrix size. Furthermore, we performed a numerical example, measuring execution time and evaluating determinant accuracy to verify the correctness and assess the method's efficiency.

The rest of the paper is organized as follows: First, a method to calculate the determinant of a matrix is described in the next section. In section 3, the approaches to measure calculating performance are discussed in the sense of counting the number of operations. Next, we prove the main theorems showing the closed form of FLOPS for computing

Manuscript received June 30, 2023; revised January 2, 2024. This work was supported by King Mongkut's Institute of Technology Ladkrabang [2565-02-05-028].

P. Sakkaplangkul is an assistant professor of the Department of Mathematics, School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand (e-mail: puttha.sa@kmitl.ac.th.)

N. Chuenjarern is a lecturer at the Department of Mathematics, School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand (Corresponding author; phone: +660837672304; e-mail:nattaporn.ch@kmitl.ac.th)

the determinant of a matrix. Also, the efficiency of determinant computation methods is compared. The summary of this work is discussed in the conclusion section.

II. Methods of Matrix Determinants

In this section, determinant calculation methods are described. Let $A = [a_{ij}]$ be a square matrix, and its determinant, denoted by $\det(A)$ or $|A|$, is a constant determined by the definition found in [13].

The conventional approach to compute the determinant is the Gaussian elimination method, which involves a series of operations on the coefficient matrix. For determinant computation, the matrix reduction method has been widely used. Also, Chio's condensation method, Dodgson's condensation method, PN's method, and Rezaifar's method are all included in the methods.

A. Gaussian elimination method

Gaussian elimination is a widely recognized technique for determinant calculation in square matrices. This method transforms the matrix into a triangular form through row operations and derives the determinant from the product of the diagonal elements of the triangular matrix.

Algorithm 1: An algorithm of Gaussian elimination method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A. If the matrix is 2×2, calculate its determinant directly.

Step 2: Performing elementary row operations to transform the matrix into an upper triangular form

Step 2.1: Finding the pivot element in the i-th column. If it is zero, the determinant is zero. If the pivot element is not in the i-th row, swap the row containing the pivot element with the i-th row.

Step 2.2: For each row below the pivot row, subtract a multiple of the pivot row from the current row to eliminate the element below the pivot in the same column. Set all elements below the pivot element in the same column to zero.

Step 3: Multiply the diagonal elements of the upper triangular matrix, the determinant of A.

B. Chio's condensation method

In 1853, Chio [3] proposed an interesting recursive method to evaluate the determinant of a square matrix. This approach reduces the size of the matrix to an $(n-1)\times(n-1)$ matrix composed of the determinants of 2×2 matrix as its entries.

Theorem 2.1: Let $A = [a_{ij}]$ be an $n \times n$ matrix. Without loss of generality, suppose $a_{11} \neq 0$. If B is an $(n-1)\times(n-1)$ matrix produced from A by

$$B = \begin{bmatrix} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1n} \\ a_{21} & a_{2n} \end{vmatrix} \\ \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1n} \\ a_{31} & a_{3n} \end{vmatrix} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{vmatrix} a_{11} & a_{12} \\ a_{n1} & a_{n2} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{n1} & a_{n3} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1n} \\ a_{n1} & a_{nn} \end{vmatrix} \end{bmatrix}, \quad (1)$$

then the determinant of matrix A by Chio's condensation method is

$$\det(A) = \frac{1}{a_{11}^{n-2}} \det(B). \quad (2)$$

Algorithm 2: An algorithm of Chio's condensation method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A. If the matrix is 2×2, calculate its determinant directly.

Step 2: Exchange rows if $A(1,1) = 0$ to find nonzero elements.

Step 3: Calculate a submatrix $B = [b_{ij}]$ where $b_{i,j} = A(1,1)A(i+1,j+1) - A(1,j+1)A(i+1,1)$.

Step 4: Calculate the determinant in step 3 using with the formula (2), while $\det(B)$ can be a recursive function obtained from **Step 1** - **Step 4**.

C. Dodgson's condensation method

In 1866, Dodgson [14] developed a method for calculating the determinant of an $n \times n$ matrix based on the concept of Chio's condensation method. Let $A = [a_{ij}]$ be an $n \times n$ matrix and denote the original matrix $A = A^{(n)}$. Define $A^{(n-1)}$ is an $(n-1) \times (n-1)$ matrix produced from A by

$$A^{(n-1)} = \begin{bmatrix} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} & \cdots & \begin{vmatrix} a_{1(n-1)} & a_{1n} \\ a_{2(n-1)} & a_{2n} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{2,2} \\ a_{31} & a_{32} \end{vmatrix} & \ddots & \begin{vmatrix} a_{2(n-1)} & a_{,n} \\ a_{3(n-1)} & a_{3n} \end{vmatrix} \\ \vdots & & \vdots \\ \begin{vmatrix} a_{(n-1)1} & a_{(n-1)2} \\ a_{n1} & a_{n2} \end{vmatrix} & \cdots & \begin{vmatrix} a_{(n-1)(n-1)} & a_{(n-1)n} \\ a_{n} & a_{nn} \end{vmatrix} \end{bmatrix} \quad (3)$$

and $\mathbf{int}A^{(n)}$ is an $(n-2) \times (n-2)$ matrix produced from A by

$$\mathbf{int}A^{(n)} = \begin{bmatrix} a_{22} & a_{23} & \cdots & a_{2(n-1)} \\ a_{32} & a_{33} & \cdots & a_{3(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1)2} & a_{(n-1)3} & \cdots & a_{(n-1)(n-1)} \end{bmatrix}. \quad (4)$$

In other words, $\mathbf{int}A^{(n)}$ is obtained by deleting the first and last row and column of the matrix A. Without loss of generality, we assume that $\mathbf{int}A^{(n)}$ is a non-zero matrix.

Algorithm 3: An algorithm of Dodgson's condensation method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A. If the matrix is 2×2, calculate its determinant directly.

Step 2: Reduce matrix A of size $n \times n$ to matrix $A^{(n-1)}$ of size $(n-1) \times (n-1)$ composing of the 2×2 determinant for every four adjacent terms by using (3).

Step 3: Find a matrix $\mathbf{int}A^{(n)}$ of size $(n-2) \times (n-2)$ by (4).

Step 4: Construct $B^{(n-2)}$, an $(n-2) \times (n-2)$ matrix, from the matrix $A^{(n-1)}$ by (3).

Step 5: Construct the matrix $A^{(n-2)}$ by dividing the matrix $B^{(n-2)}$ by $\mathbf{int}A^{(n)}$ in Step 3 component-wise.

Step 6: Repeat Step 3 – 5 to find matrices $A^{(n-k)}$ where $3 \leq k \leq n-1$ by using previous results $A^{(n-k+2)}$ and $A^{(n-k+1)}$ until we obtain $A^{(1)}$, a determinant of A.

Theorem 2.2: Let $A = [a_{ij}]$ be an $n \times n$ matrix, then the determinant of matrix A by Dodgson's condensation method is

$$\det(A) = A^{(1)}, \tag{5}$$

where $A^{(1)}$ is obtained by the above Dodgson's condensation process.

D. Rezaifar's method

The determinant based on Rezaifar's approach [1] for a square matrix is given by

$$\det(A) = \frac{1}{|A_{11,nn}|} \cdot \det \begin{bmatrix} |A_{11}| & |A_{1n}| \\ |A_{n1}| & |A_{nn}| \end{bmatrix}, \tag{6}$$

where A_{ij} and $A_{ii,jj}$ are given by

$$A_{ij} = \begin{bmatrix} * & * & \dots & * & | & * & \dots & * \\ * & * & \dots & * & | & * & \dots & * \\ - & - & - & - & \oplus_{ij} & - & - & - \\ * & * & \dots & * & | & * & \dots & * \\ * & * & \dots & * & | & * & \dots & * \end{bmatrix}, \tag{7}$$

$$A_{ii,jj} = \begin{bmatrix} * & | & * & \dots & * & | & * & * \\ - & \oplus_{ii} & * & - & - & | & - & - \\ * & | & * & \dots & * & | & * & * \\ - & - & - & - & - & \oplus_{jj} & - & - \\ * & | & * & \dots & * & | & * & * \end{bmatrix}. \tag{8}$$

Remark A_{ij} is a matrix obtained by deleting the $i-$th row and $j-$th column and $A_{ii,jj}$ is a matrix obtained by deleting $i-$th row, $i-$th column, $j-$th row and $j-$th column. It can be seen that the right-hand side of (6) involves a scalar number involving determinants of an $(N-2) \times (N-2)$ matrix and a 2×2 matrix. The 2×2 matrix on the right side of (6) consists of the determinant of the $(N-1) \times (N-1)$ matrices.

Algorithm 4: An algorithm of Rezaifar's method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A.

Step 2: Construct $A_{11}, A_{1n}, A_{n1}, A_{nn}$ and $A_{11,nn}$ using (7) and (8).

Step 3: Calculation determinants of matrices in Step 2 using Gaussian elimination method.

Step 4: Calculating determinant using (6).

E. PN's method

The method to evaluate the determinant introduced in [10] involves reducing the size of the $n \times n$ matrix, A, by dividing the original matrix into four blocks

$$A = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, \tag{9}$$

where P, Q, R, and S are block matrices having sizes $k \times k, k \times (n-k), (n-k) \times k, (n-k) \times (n-k)$, respectively. The determinant of the matrix A is calculated by using the formula of Schur [12]

$$\det(A) = \det(P) \det(S - RP^{-1}Q), \tag{10}$$

where the Gauss elimination method is utilized to compute the determinants of submatrices on the right side (10).

Algorithm 5: An algorithm of PN's method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A and a size of the submatrices defined in (9).

Step 2: Create a loop for calculating a submatrix $S - RP^{-1}Q$.

Step 3: Applying Gaussian elimination method to calculate $\det(S - RP^{-1}Q)$ and $\det(P)$.

Step 4: Calculating determinant using (10).

III. Evaluation of the Quantity of Operations over the Matrix Elements

This section presents the methodology used to evaluate the effectiveness of a determinant calculation method.

A. Floating-point operations per second (FLOPS)

A floating-point operation per second (FLOPS) serves as a metric to measure the computer performance based on the number of floating-point arithmetic calculations that the processor can perform within a second. In essence, FLOPS quantifies the number of operations required to execute a command in a program. Fewer flops correspond to reduced computation time. The approach for counting flops in mathematical operations involves considering basic arithmetic

operations such as addition, subtraction, multiplication, and division. Each of these operations is counted as one flops. For instance, $2+3$ is equivalent to 1 flops, while $4(5+2)$ accounts for 2 flops. When it comes to matrix algebra, the flops count is usually higher because a single matrix operation involves multiple entries. More examples of FLOPS related to vector and matrix operations can be found in [15].

B. Closed form of FLOPS.

In this section, the closed-form FLOPS is determined for the methods described in the previous section using the principle of counting operations.

Theorem 3.1: Let A be an $n \times n$ matrix. The number of flops required for the Gaussian elimination method can be defined by

$$\text{FLOPS}_{\text{Gau}}(n) = \frac{1}{6}(4n^3 - 3n^2 + 5n - 6). \tag{11}$$

Proof: See [13]. ■

Theorem 3.2: Let A be an $n \times n$ matrix. The number of flops required for Chio's condensation method can be defined by

$$\text{FLOPS}_{\text{Chio}}(n) = \sum_{k=1}^{n-1} 2(n-k)^2 + (n-k). \tag{12}$$

Proof: We prove this theorem using mathematical induction on the size of a square matrix A. Without loss of generality, let's assume $a_{11} \neq 0$. To begin, for a matrix of size 2×2, the number of flops required to compute the determinant is 3 flops. We can express this as:

$$\text{FLOPS}_{\text{Chio}}(2) = \sum_{k=1}^{1} 2(2-k)^2 + (2-k) = 3.$$

Next, Let n be a positive integer where $n \geq 2$ and assume that the number of flops required to compute the determinant of an $n \times n$ matrix is denoted as $\text{FLOPS}_{\text{Chio}}(n)$. For a square matrix A of size $(n+1) \times (n+1)$, the process of reducing A to B, which is of size $n \times n$, using Chio's condensation method involves the following steps:

1) Scaling the entry a_{11} in the first row of A to 1, which requires n flops.
2) Calculating the determinants of n^2 submatrices of size 2×2, which requires $2n^2$ flops.

Therefore, the total number of flops is $2n^2+n$. According to the mathematical induction hypothesis, the number of flops required to compute the determinant of B is $\text{FLOPS}_{\text{Chio}}(n)$. Consequently, the number of flops needed to compute the determinant of A is

$$2n^2 + n + \sum_{k=1}^{n-1} \left(2(n-k)^2 + (n-k)\right). \tag{13}$$

By (13), we have

$$\begin{aligned} &2n^2 + n + \sum_{k=1}^{n-1} \left(2(n-k)^2 + (n-k)\right) \\ &= \sum_{k=1}^{n} 2(n+1-k)^2 + (n+1-k) \\ &= \text{FLOPS}_{\text{Chio}}(n+1). \end{aligned}$$

By mathematical induction, we obtain the number of flops for calculating the determinant by Chio's condensation method satisfies (12). ■

The total number of operations for calculating the determinant using Chio's condensation method, as presented in Theorem 3.2, aligns with the results obtained by Habgood in [16]. Furthermore, we can express the flops as follows:

$$\begin{aligned} \text{FLOPS}_{\text{Chio}}(n) &= \sum_{k=1}^{n-1} 2(n-k)^2 + (n-k) \\ &= \frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}. \end{aligned} \tag{14}$$

Theorem 3.3: Let A be an $n \times n$ matrix. The number of flops required for Dodgson's condensation method can be defined by

$$\text{FLOPS}_{\text{Dodg}}(n) = \sum_{k=1}^{n-1} 3(n-k)^2 + (n-k-1)^2. \tag{15}$$

Proof: The proof of this theorem may be followed by the concept of the proof of Theorem 3.2 using mathematical induction. ■

The number of floating-point operations (flops) as described in Theorem 3.3 corresponds to the results presented by Malaschonok in [17].

Theorem 3.4: Let A be an $n \times n$ matrix. The number of flops required for the method in (6) is given by

$$\text{FLOPS}_{\text{Rezaifar}}(n) = \frac{1}{6}\left(20n^3 - 87n^2 + 157n - 108\right). \tag{16}$$

Proof: First, we calculate the number of flops of computing determinants of submatrices, A_{11}, A_{1n}, A_{n1} and A_{nn}, by using Gaussian elimination method. From (11), the determinant of each submatrix of size $(n-1) \times (n-1)$ requires $\frac{2}{3}(n-1)^3 - \frac{1}{2}(n-1)^2 + \frac{5}{6}(n-1) - 1$ flops and the submatrix $A_{11,nn}$ of size $(n-2) \times (n-2)$ requires $\frac{2}{3}(n-2)^3 - \frac{1}{2}(n-2)^2 + \frac{5}{6}(n-2) - 1$ flops. Moreover, it requires 3 flops for calculating the determinant of 2×2 matrix and 1 flops for the product between the determinant of 2×2 matrix and the scalar. Therefore, the total number of flops for Rezaifar's method satisfies (16). ■

Theorem 3.5: Let A be an $n \times n$ matrix. Suppose that the matrix P in (9) of size $k \times k$ is invertible. The number of flops required for the method in [10] is given by

$$\text{FLOPS}_{\text{PN}}(n) = \begin{cases} \dfrac{1}{6}\alpha & \text{if } k = 1 \\ \dfrac{1}{6}(-6 + 18k^3 - 6k + \alpha) & \text{if } 2 \leq k \leq n. \end{cases} \tag{17}$$

where $\alpha = 5n - 3n^2 + 4n^3$.

Proof: See in [10]. ■

IV. Main Results

In this section, we present and analyze a new approach to evaluate the determinant of a square matrix. This method is based on an adaptation of the PN's method [10]. As shown in Section 2, Chio's condensation method requires a relatively low number of floating-point operations to calculate the

determinant. We also note that when the submatrix P, as mentioned in Algorithm 5 of PN's method, has a size of 1×1, it requires the minimum number of operations to compute the determinant using the PN's method [10]. Building upon this observation, we have extended these advantages by incorporating Chio's condensation method for determinant computation using the Schur formula instead of the Gaussian elimination method. The algorithm for this method is presented in Algorithm 6.

Algorithm 6: An algorithm of adapted PN's method

input : Matrix A
output: $\det A$

Step 1: Determine a size of A and a size of the submatrices defined in (9) where a submatrix on the first main diagonal, P, of size 1×1.

Step 2: Create a loop for calculating a submatrix $S - RP^{-1}Q$.

Step 3: Applying Chio's condesation method to calculate $\det(S - RP^{-1}Q)$ and $\det(P)$.

Step 4: Calculating determinant using (10).

Theorem 4.1: Let A be an $n \times n$ matrix. Suppose that the matrix P in (9) of size 1×1 is invertible. The number of flops of the adapted method is given by

$$\text{FLOPS}_{\text{AdaptedPN}}(n) = \frac{1}{6}(12 - n - 3n^2 + 4n^3). \qquad (18)$$

Proof: In order to calculate the total number of flops in this approach, we divide the process into the following steps:

- Step 1: As P is of size 1×1, it takes 1 flops to find the inverse of P.
- Step 2: Calculating the matrix $S - RP^{-1}Q$ consists of three steps:
  - Calculation $P^{-1}Q$ requires $n - 1$ flops.
  - Calculation $RP^{-1}Q$ requires $(n-1)^2$ flops.
  - Calculation $S - RP^{-1}Q$ requires $(n-1)^2$ flops.

  In this process, $(n-1)(2n-1)$ flops are utilized altogether.
- Step 3: Calculating the determinant $S-RP^{-1}Q$ of size $(n-1) \times (n-1)$ by using the Chio's condensation method (14) requires $\frac{1}{6}(4n^3 - 15n^2 + 17n - 6)$ flops.
- Step 4: Applying the formula of Schur (10) needs 1 flops to calculate the product.

Therefore, the total number of flops from the above-mentioned steps satisfies (18). ■

V. Experiments

In this section, we demonstrate the performance of a method for calculating determinants. We begin by presenting a numerical example to provide an initial assessment of the method. To further evaluate our approach, we will use both flops and execution time to compute determinants for randomly chosen matrices varying in size from 3×3 to 15×15. Additionally, we will illustrate the accuracy of our proposed method.

A. Numerical example

In this section, we provide a numerical example to demonstrate the usage of our method. Let's assume that a matrix A is given as:

$$A = \begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 2 & 3 & 4 \\ 2 & 1 & -1 & 1 \\ 3 & 1 & 4 & 1 \end{bmatrix}.$$

The common method for finding the determinant can be applied as follows: $\det(A) = 1 \cdot 2 \cdot 10 + (-1) \cdot 1 \cdot 42 + 1 \cdot (-1) \cdot (-2) + 1 \cdot 0 \cdot (-20) = -20$.

The determinant of A was calculated using the adapted PN's method as follows:

- Step 1: We obtain the four submatrices $P = \begin{bmatrix} 2 \end{bmatrix}$, $Q = \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}$, $R = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, and $S = \begin{bmatrix} 2 & 3 & 4 \\ 1 & -1 & 1 \\ 1 & 4 & 1 \end{bmatrix}$.
- Step 2: We have $S - RP^{-1}Q = \begin{bmatrix} 3/2 & 7/2 & 4 \\ 0 & 0 & 1 \\ -1/2 & 11/2 & 1 \end{bmatrix}$.
- Step 3: By Chio's condensation method, we obtain

$$\det(S - RP^{-1}Q) = \frac{1}{3/2} \begin{vmatrix} \begin{vmatrix} 3/2 & 7/2 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} 3/2 & 4 \\ 0 & 1 \end{vmatrix} \\ \begin{vmatrix} 3/2 & 7/2 \\ -1/2 & 11/2 \end{vmatrix} & \begin{vmatrix} 3/2 & 4 \\ -1/2 & 1 \end{vmatrix} \end{vmatrix}$$

$$= \frac{2}{3} \begin{vmatrix} 0 & 3/2 \\ 10 & 7 \end{vmatrix} = -10.$$

- Step 4: $\det(A) = 2 \cdot (-10) = -20$.

Both methods provided the same value.

B. FLOPS comparison

In this section, a comparative analysis is performed to assess the computational efficiency of different determinant calculation methods. Initially, we investigated the number of flops required for determining the determinant of matrices.

Table I shows that FLOPS for determinant calculation for square matrices of size $n \times n$ where $n \leq 6$ have a relatively similar trend.

TABLE I: Flops of the methods to calculate the determinant of an $n \times n$ matrix with $n = 3$ to $n = 6$

Methods	Flops of a matrix of size $n \times n$			
	$n = 3$	$n = 4$	$n = 5$	$n = 6$
Gaussian	15	37	74	130
Chio	13	34	70	125
Dodgson	16	47	104	195
Rezaifar	20	68	167	337
PN with P is 1×1	16	38	75	131
Adapted PN	15	36	72	127

However, for larger matrices, both our approach and Chio's condensation method demonstrated comparable and lower

flops compared to other methods. This observation highlights the efficiency advantages provided by our method and Chio's condensation method, which are especially noticeable with larger matrices.

Next, the performance of a large-sized square matrix was examined. As shown in Figure 1, Rezaifar's method requires the highest number of flops for determinant calculation. Nevertheless, the flops decreased when we employed Dodgson's condensation method, PN's method with a submatrix block of size 1×1, the Gaussian elimination method, adapted PN's method, and Chio's condensation method.

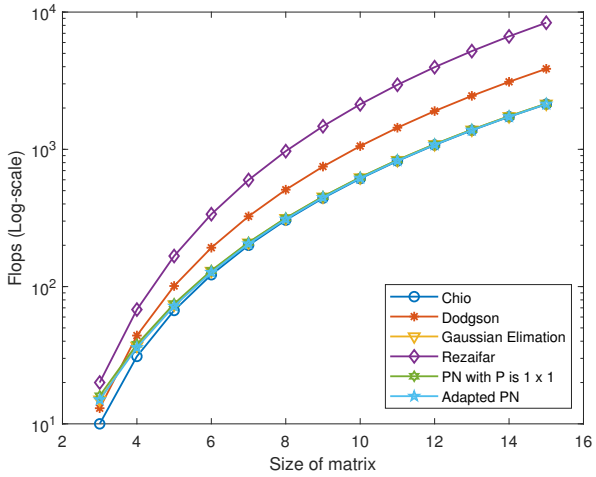

Fig. 1: Semi-logarithmic graph shows the relationship between flops of each method and sizes of matrices

C. Execution time comparison

Time complexity analysis offers an estimation of how the computational time for determinant calculation scales with the matrix size. Various determinant calculation methods exhibited different time complexities. This analysis enables us to discern the efficiency levels among these methods.

In this section, we utilize the software detailed in Table II to compare the execution time of determinant computation, and Table III presents the computer hardware used for this simulation.

TABLE II: Computer software used for evaluating execution time

OS	Windows 10 Education Version 21H1 OS build 19043.1706
Software	MATLAB, Version 9.3.0.713579 (R2017b), 64- bit (win64)

In Table IV and Figure 2, we present a comprehensive overview of the computational costs associated with our determinant calculation approach. Through extensive computational testing, we systematically compare the performance of each algorithm, including the Gaussian elimination method, Chio's condensation method, Dodgson's condensation method, Rezaifar's method, PN's method with P is 1×1 (PN1) , and the Adapted PN's method (APN). These comparisons are performed on randomly generated square matrices ranging in sizes from 3×3 to 15×15. The elapsed time taken to calculate the determinant for each method is measured using Matlab.

TABLE III: Computer hardware used to evaluating the determinant

Name	HP
Model	Prodesk 400 G7 Microtower PC
CPU	Intel(R) Core(TM) i5-10600 CPU @ 3.30GHz 3.31 GHz
RAM	8.00 GB
HDD	SSD 512GB M.2 2280 PCIe NVMe

TABLE IV: Execution time using MATLAB functions to calculate determinants of a square matrix of size from 3×3 to 15×15 presented in seconds.

Execution time of determinant calculation (seconds)						
Size	**Gaussian**	**Chio**	**Dodgson**	**Rezaifar**	**PN1**	**APN**
3×3	0.001417	0.000964	0.002823	0.002949	0.001244	0.001071
4×4	0.001894	0.001089	0.004345	0.003514	0.001306	0.001053
5×5	0.002008	0.001277	0.004996	0.005811	0.001628	0.001239
6×6	0.002228	0.001956	0.005302	0.007043	0.002220	0.001978
7×7	0.002324	0.002110	0.006105	0.007835	0.002202	0.002353
8×8	0.002223	0.002098	0.007154	0.008337	0.002239	0.002210
9×9	0.002293	0.002566	0.007130	0.008756	0.002232	0.002288
10×10	0.002633	0.002155	0.007977	0.008916	0.002693	0.002616
11×11	0.002905	0.002223	0.007988	0.009079	0.002796	0.002529
12×12	0.002651	0.002285	0.008471	0.009292	0.002715	0.002596
13×13	0.002745	0.002302	0.008857	0.009451	0.002766	0.002630
14×14	0.002482	0.002464	0.008089	0.009162	0.002588	0.002437
15×15	0.002589	0.002502	0.008115	0.009240	0.002655	0.002512

In this comparison, it is evident that all methods exhibit a consistent trend where the execution time for all methods is directly linked to the matrix size. Specifically, Chio's method, Gaussian method, PN1 method, and APN method show similar execution times. On the other hand, Dodgson and Rezaifar methods take longer compared to the previously mentioned ones. Given that each operation in determinant calculation demands computation time, methods with a higher number of operations naturally require more computation time. This observation is consistent with the results obtained from the number of flops in Section V(B), providing additional confirmation.

D. Methods' accuracy

In this section, we will evaluate and compare the accuracy of each method. This evaluation aims to confirm the relia-

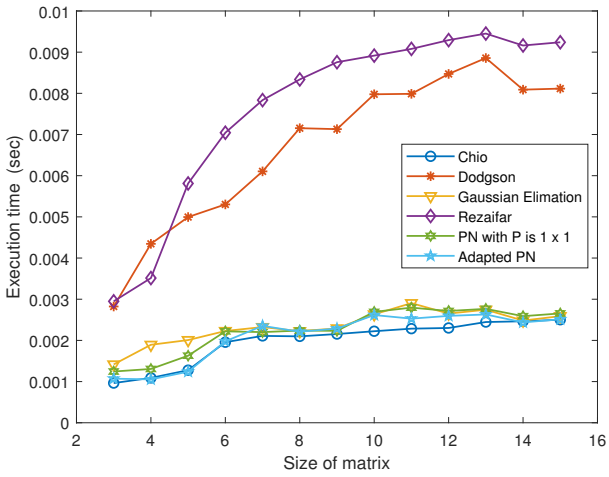

Fig. 2: Execution time using MATLAB function

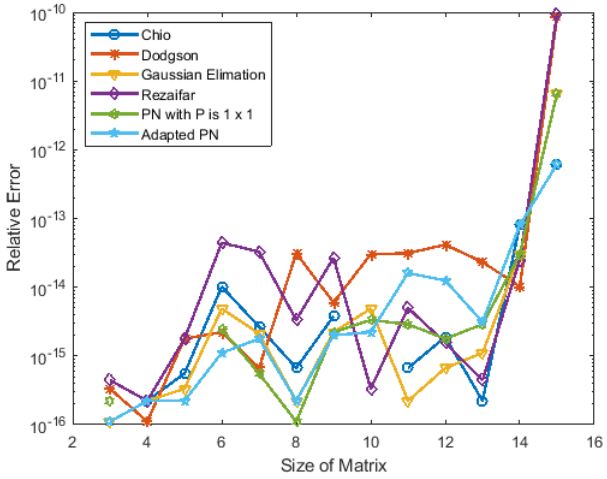

Fig. 3: Semi-logarithmic plot of the methods' accuracy

bility and precision of our approach, ensuring its consistent performance across a range of matrices.

We conducted accuracy tests to calculate the determinants of randomly generated square matrices ranging in sizes from 3×3 to 15×15. Each determinant method was applied to compute the determinant of these matrices, and the results were compared with the determinant values obtained from the MATLAB `det()` function. Specifically, we evaluated the accuracy of determining the determinant of matrix A by measuring the relative error with the formula:

$$\text{Relative Error} = \left|1 - \frac{NumDet}{\det(A)}\right|, \tag{19}$$

where $NumDet$ signifies the determinant value obtained from each method, and $\det(A)$ denotes the determinant value acquired through the `det()` function.

TABLE V: Assessment of determinant calculation accuracy for square matrices of sizes ranging from 3×3 to 7×7

	Relative error ($\times 10^{-16}$)					
Size	**Gaussian**	**Chio**	**Dodgson**	**Rezaifar**	**PN1**	**APN**
3×3	1.1102	0	3.3307	4.4409	2.2204	1.1102
4×4	2.2204	2.2204	1.1102	2.2204	0	2.2204
5×5	3.3307	5.5511	1.7764	1.7764	0	2.2204
6×6	48.849	102.14	22.204	442.98	24.424	11.102
7×7	21.094	26.645	6.6613	328.63	5.5511	17.764

From Table V and Figure 3, it is evident that the discrepancies in determining the determinant using each method exhibit a consistent trend and are very small, on the order of 10^{-16} to 10^{-10}, which is close to zero. However, it's important to note that these errors do not impact the size of the matrices but can influence the precision of floating-point calculations and the performance of the computer. This implies that our proposed method provides accuracy comparable to conventional approaches.

VI. Discussion and Conclusion

A. Efficiency of the method

1. Asymptotic analysis. In comparing the efficiency of determinant calculation methods with the general approach, we utilized asymptotic analysis to derive the explicit FLOPS formula for each method. We can quantify the operations involved in determining determinants of any square matrices using the explicit FLOPS formula. The advantage of asymptotic analysis lies in its ability to clarify the limiting behavior of determinant calculation methods as the matrix size increases.

While the flops for Gaussian, Chio, PN1, and APN methods are $\mathcal{O}(n^3)$, the analysis from the results in Section III and Section IV revealed that our APN method requires fewer flops compared to PN1 and Gaussian. Specifically, the APN method incurs significantly fewer flops as the matrix size increases. However, when compared to the Chio method, our APN method adds only an additional 2 flops, regardless of the matrix size.

2. Elapsed time computation. As outlined in the asymptotic analysis section, our method demands fewer flops than the PN1 and Gaussian methods, particularly for larger matrix sizes. This results in a decreased computation time for determinant calculations. However, the method's time complexity aligns with the Chio method, with only a marginal 2-flop difference which is not considered significant in computation time.

3. Accuracy. Our method also yields determinants' results consistent with the general method for determinant calculation. The experimental results presented in Section IV confirm the accuracy and correctness of the proposed method.

4. Simplicity. The method involves calculating determinants by employing the reduced PN1 method in conjunction with Chio's technique. This can be manually computed with relative ease as it involves matrix size reduction and determinant calculation of submatrices using 2×2 determinants. This stands as another viable option for computing matrices of larger sizes beyond the traditional method.

B. Conclusion and future works

This paper introduces a recursive procedure for determinant calculation, utilizing Schur's formula and a condensed

technique on submatrices. We derive and establish the closed-form expression for the flops of the adapted method, comparing its performance with other existing methods in terms of flops, execution time, and accuracy. The findings demonstrate consistent results across all methods, with larger matrices demanding more flops and execution time. Notably, our approach and Chio's condensation method exhibit improved performance for larger matrices, despite the $\mathcal{O}(n^3)$ complexity associated with calculating determinants of $n \times n$ matrices.

It's worth noting that our method, based on Schur's formula and Chio's condensation, simplifies determinant calculations using 2×2 matrices without compromising correctness compared to other methods. However, the entries of the matrix influence determinant computation, particularly when encountering zero elements or a zero determinant during Chio's condensation method. In such cases, rearranging the matrix using elementary row operations slightly affects execution time but has no impact on flops.

In future research, we could consider the possibility of dividing the matrix into more than four submatrices to achieve a greater reduction in the size of the original matrix. This would allow us to apply the APN method for determinant calculation on smaller matrices. Furthermore, we can expand our methodology to develop a technique for determining determinants for matrices of any size, employing matrix partitioning or matrix decomposition methods.

REFERENCES

[1] O. Rezaifar and H. Rezaeeb, "A new approach for finding the determinant of matrices," *Applied Mathematics and Computation*, pp. 1455–1454, 2007.

[2] M. Bayat and H. Faal, "New method for computing determinants by reducing the orders by two," *Caspian Journal of Mathematical Sciences (CJMS)*, vol. 7, no. 1, pp. 16–24, 2018.

[3] C. Felice, *Mémoire sur les fonctions connues sous le nom De Résultantes Ou De Déterminans*. A. Pons, 1853.

[4] H.-B. L. H. Li and T.-Z. Huang, "A note on dodgson's determinant condensation algorithm," *ArXiv abs/1907.12010*, 2019.

[5] A. Rise and E. Torrence, "Shutting up like a telescope: Lewis carroll's curious condensation method for evaluating determinants," *The College Mathematics Journal*, vol. 38, pp. 85–95, 2006.

[6] G.Alyami and I. Kostanic, "A low complexity user selection scheme with linear precoding for massive mimo systems," *IAENG International Journal of Computer Science*, vol. 43, no. 3, pp. 374–379, 2016.

[7] S. K. S. K. Mousami Turuk, R Sreemathy and V. Kulkarni, "Cnn based deep learning approach for automatic malaria parasite detection," *IAENG International Journal of Computer Science*, vol. 49, no. 3, pp. 745–753, 2022.

[8] H. T. X. Zhang and W. Lv, "A low-complexity detection algorithm for generalized space shift keying systems," *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 364–368, 2022.

[9] A. M. A. S. J.-F. D. N. Boumaaz, H. Semlali and A. Ghammaz, "Low complexity spectrum sensing approach applying random sampling in cognitive radio networks," *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 489–494, 2022.

[10] P. Sakkaplangkul and N. Chuenjarern, "Computational efficiency for calculation determinants of block matrices," *RMUTP Research Journal: Science and Technology*, vol. 18, no. 1, p. In press, 2024.

[11] R. W. Cottle, "Manifestations of the schur complement," *Linear Algebra and its Applications*, vol. 8, no. 3, pp. 189–211, 1974.

[12] J. Schur, "*Ü*ber potenzreihen, die im innern des einheitskreises beschränkt sind," *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 148, pp. 122–145, 1918.

[13] G. Williams, *Linear Algebra with Applications Alternate Edition*. Jones & Bartlett Learning, 2009.

[14] C. Dodgson, "Condensation of determinants, being a new and brief method for computing their arithmetical values," *Proceedings of the Royal Society of London*, vol. 15, pp. 150–155, 1866.

[15] R. Hunger, *Floating Point Operations in Matrix-Vector Calculus*. Germany: Technical University of Munich.

[16] K. Habgooda and I. Arel, *A condensation-based application of Cramer's rule for solving large-scale linear systems*. The University of Tennessee, Knoxville, TN, USA, 2017.

[17] G. Malaschonok, "Algorithms for the computing determinants in commutative rings," *arXiv preprint arXiv:1711.11472*, 2017.

Puttha Sakkaplangkul is an assistant professor of the Department of Mathematics, School of Science, KMITL, Bangkok, Thailand

Nattaporn Chuenjarern is a lecturer at the Department of Mathematics, School of Science, KMITL, Bangkok, Thailand