UNIVERSITY OF SILESIA
IN KATOWICE

Wydział Nauk Ścisłych i Technicznych
Instytut Matematyki

RAZAK OLAMIDE KOLAWOLE
Numer albumu 359320

Samuelson-Berkowitz and Modular Computation Algorithm

Imię i nazwisko promotora
dr hab. Przemysław Koprowski, prof. UŚ

Katowice 2024

Abstract

Gaussian elimination method is a fast and widely-used algorithm for efficient computation of the determinant of square matrices. However, it involves division operation, which can result in coefficients that are not integers. In fact, it forces us to work in the rational domain $\mathbb{Q}$ instead of the integer domain $\mathbb{Z}$, when we so much desired to have computations in $\mathbb{Z}$.

This project considers two algorithms, first, a divison-free algorithm for obtaining the characteristics polynomial and the determinant of square matrices whose coefficients are in any unital commutative ring $\mathcal{R}$ which is not an integral domain.

The second, a modular computation algorithm for calculating the determinant of square matrices with integers coefficients. Modular reduction is an important technique for speeding up computations. The modular computation algorithm is a fast method which rely on both division-free and non division-free methods.

Contents

1 Introduction

The study of determinant has a rich and long history because of it's importance in linear algebra, geometry, computational mathematics etc.. It's background can be traced back to the work of Cardano, Leibnitz, Crammer, Vandermode, Binet, Cauchy, Jacobi, Gauss and many others. Given it's importance in physical sciences and engineering, it is not surprising that a galaxy of great mathematicians investigated the determinant from different point of views. Similarly, the concept of characteristic polynomials has a rich history spanning several centuries and various mathematical developments, namely, Viète, Newton, Cayley, Frobenius to mention but a few. The characteristics polynomial have applications in diverse areas such as quantum mechanics for understanding the possible outcomes of measurements and the evolution of quantum states over time and in data analysis to extract meaningful features or reduce the dimension of the data while preserving important structural information (see, [1], [2]).

The most common algorithm for computing the determinant of square matrices is the Gaussian elimination method (GEM for short) and it requires $O(n^3)$ additions, multiplications and divisions. GEM is fast but requires division, so when applied to matrices with integers coefficients, it forces us to work in the field of rationals $\mathbb{Q}$, it is also affected by the growth sizes of coefficients. Division-free methods avoid these problems but are asymptotically slower. The explicit definition of determinants as the sum of $n!$ products shows that determinant can be obtained without divisions. Moreover, avoiding divisions seems attractive when working over a commutative ring which is not a field. The notion of division free method birth the Samuelson–Berkowitz algorithm which is used for computing the characteristics polynomial and the determinant of square matrices (constant term of the characteristics polynomial) whose entries belong to a unital commutative ring.

Modular reduction is an important tool for speeding up computations in computer arithmetic, symbolic computation, and elsewhere. The technique allows us to reduce a problem that involves large integer or polynomial coefficients to one or more similar problems that only involve small modular coefficients (see, [3]).

Modular computation algorithm is a method employs both division-free and non division-free algorithm in the the computation of the determinant of a square matrix whose coefficients are in $\mathbb{Z}$.

The project is organized as follows. Section 2 contains essential tools that are explored later. In particular, in Section 2.1 we recall some basic definitions from linear algebra, while in Section 2.2 we state some theorems from linear algebra that are needed for this project. The description, formulation with an example, implementation and complexity of Samuelson Berkowitz algorithm is discussed in Section 3.1, Section 3.2, Section 3.3 and Section 3.4 of Section 3 respectively. We proceed to Section 4 where we describe the modular computation algorithm in Section 4.1, while formulation of the algorithm with an example, implementation and the description of its complexity is done in Section 4.2, Section 4.3 Section 4.4 respectively.

2 Background

This section collects the basic definitions and results needed to write this project.

2.1 Definition of Terms

We shall recall some basic definitions and facts from linear algebra.

An *integral domain* $\mathcal{D}$ is a commutative ring with unity but without a zero divisor

A $n \times n$ *Toeplitz matrix* M is a matrix whose each entries depend on the difference $i - j$ and hence they are constant down all the diagonals.

$$M_{i,j} = \begin{pmatrix} a_0 & a_{-1} & \cdots & a_{-(n-1)} \\ a_1 & a_0 & \cdots & a_{-(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1)} & a_{(n-2)} & \cdots & a_0 \end{pmatrix}$$

We say that a matrix is *lower triangular* if all the values above the main diagonal are zero

Let M be a square matrix of size $n \times n$ with coefficients in some ring $\mathcal{R}$ and I_n denotes the unit matrix of size n. The polynomial $P_M(x) \in \mathcal{R}[x]$ is called the *characteristic polynomial* of M.

$$P_M(x) = det(xI_n - M) = (-1)^n(x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_n)$$

A polynomial $p(x) = \sum_{i=0}^{n} b_i x^i$ is called a *monic polynomial* of degree n if $b_n = 1$.

Let M be a square matrix of size $n \times n$ with coefficients in some ring $\mathcal{R}$. The *adjoint matrix* (also known as adjugate) of a square matrix M is a matrix $adj(M) = ((-1)^{i+j} \cdot detM_{ij})^T$ for $i, j \leq n$

Let M be a matrix of size $n \times n$ with coefficients in some ring $\mathcal{R}$. The *determinant* of a square matrix M is defined by the formula

$$detM = \sum_{\sigma} sgn(\sigma) \cdot m_{1\sigma(1)} \cdots m_{n\sigma(n)}$$

Let M be an $n \times n$ matrix with coefficients in some ring $\mathcal{R}$. A *principal submatrix* of a square matrix M is the matrix obtained by deleting any k rows and the corresponding k columns of the matrix M.

The $k \times k$ sub-matrix obtained from matrix M by deleting any $(n-k)$ columns of M and the corresponding $(n-k)$ rows of M is called *k-order principal sub-matrix of M.* That is,

$$M = \left[\begin{array}{c|c} M_{k,k} & C_k \\ \hline R_k & m_{k+1,k+1} \end{array}\right]$$

where R_k, C_k and $m_{k+1,k+1}$ are $k \times (n-k)$, $(n-k) \times k$ and $(n-k) \times (n-k)$ $(k = 0, 1, 2, \ldots, n)$ sub-matrices, respectively.

Let n be a positive integer, we say the integers a and b are congruent modulo n, and write $a \equiv b(mod\ n)$, if they have the same remainder on division by n.

2.2 Basic Theorems

In this part, several theorems about determinants and congruence relation are stated and proved. The first theorem is a useful tool for calculating the determinant of a matrix.

Theorem 2.1. *(Laplace expansion). Let* $M = (m_{ij})_{1\leq i,j\leq n}$ *be a square matrix. Denote by* M_{ij} *a matrix obtained from* M *after deleting its* $i-th$ *row and* $j-th$ *column. Then for every* i *and* j *the determinant of* M *can be expanded into a sum*

$$detM = \begin{cases} \sum_{j=1}^{n}(-1)^{i+j}m_{ij}detM_{ij}, & \textit{(expansion with respect to } j-th \textit{ row)} \\ \sum_{i=1}^{n}(-1)^{i+j}m_{ij}detM_{ij} & \textit{(expansion with respect to } i-th \textit{ row)} \end{cases}$$

Proof. See [4] for a proof to this theorem. □

Colorally 2.2. *If* M *is a triangular matrix (either upper or lower-triangular), then det* M *is the product of elements on the main diagonal of* M

The following establish the fact that there is a relationship between the characteristics polynomial of a square matrix M and it's determinant.

Remark 2.3. The constant term of the characteristic polynomial of a matrix M of size $n \times n$ equals $(-1)^n \cdot detM$.

Since, $P_M(x) = det(xI_n - M)$, then substituting $x = 0$ gives

$$P_M(0) = det(-M) = (-1)^n \cdot detM$$

The next result is a useful tool in the formulation of the Samuelson-Berkowitz algorithm as it provides an alternative expression for the inverse of a matrix in terms of it's adjugate and determinant.

Theorem 2.4. *Let* M *be a square* $n \times n$ *matrix, then*

$$M \cdot adj(M) = adj(M) \cdot M = det(M) \cdot I_n \tag{2.1}$$

Proof. If M is a square and non-singular matrix then there is nothing prove because

$$M^{-1} = \frac{adj(M)}{det(M)}$$

If on the other hand M is any matrix, then from the definition of determinants and the coefficient of adjoint matrix we write

$$detM = \sum_{i}^{n}(-1)^{i+j}m_{ij}detM_{ij},$$

$$(adjM)_{ij} = (-1)^{i+j}detM_{ji}$$

Multiplication of the coefficient of this adjoint by M gives

$$(M \cdot adj(M))_{kl} = \sum_{i}^{n}(-1)^{i+l}m_{ki}detM_{li}$$

if $k = l$ then

$$(M \cdot adj(M))_{kl} = (adj(M) \cdot M)_{kl} = \sum_{i}^{n} (-1)^{i+l} m_{li} det M_{li} = det(M) \cdot I$$

If $k \neq l$, then the result will be zero because it's the determinant of a matrix with two equal rows.

The proof is complete. □

Theorem 2.5. *(Cayley–Hamilton).Every square matrix satisfy its own characteristics equation. That is, if M is a square matrix and $P_M(x)$ is it's characteristic polynomial, then $P_M(M) = 0$*

Proof. From (2.1) we have

$$(M - xI) \cdot adj(M - xI) = adj(M - xI) \cdot (M - xI) = det(M - xI) \cdot I,$$

and the adjoint has the form $adj(M - xI) = P_{ij}(x)$, where $p_{ij}(x)$ are polynomial of degree at most $n - 1$ for $1 \leq i, j \leq n$.

Therefore, the adjoint matrix can be written as

$$adj(M - xI) = B_0 + B_1 x + \ldots + B_{n-1} x^{n-1} \quad = (-1)^n (x^n + a_1 x^{n-1} + \ldots + a_n) \cdot I, \tag{2.2}$$

for some $n \times n$ matrices $B_0, B_1, \ldots, B_{n-1}$. Then it follows that

$$(M - xI)(B_0 + B_1 x + \ldots + B_{n-1} x^{n-1}) = (-1)^n (x^n + a_1 x^{n-1} + \ldots + a_n) \cdot I$$

Equating the coefficient of like powers of x we have

$$\begin{aligned} MB_0 &= (-1)^n a_n I \\ -B_0 + MB_1 a &= (-1)^n a_{n-1} I \\ &\vdots \\ -B_{n-2} + MB_{n-1} &= (-1)^n a_1 I \\ -B_{n-1} &= (-1)^n I \end{aligned} \tag{2.3}$$

Multiplying the left hand side of (2.3) by $I, M, M^2, \ldots, M^n$ respectively we obtain

$$\begin{aligned} MB_0 &= (-1)^n a_n I \\ -MB_0 + M^2 B_1 a &= (-1)^n a_{n-1} M \\ &\vdots \\ -M^{n-1} B_{n-2} + M^n B_{n-1} &= (-1)^n a_1 M^{n-1} \\ -M^n B_{n-1} &= (-1)^n M^n \end{aligned} \tag{2.4}$$

Clearly, the left hand side of (2.4) telescope, so adding all the equations gives

$$P_M(M) = (-1)^n (M^n + a_1 M^{n-1} + \ldots + M_n) = 0$$

□

An important consequence of the Cayley-Hamilton theorem is that any polynomial in a $n \times n$ matrix M can be rewritten as a polynomial whose degree is at most $n-1$.

The next theorem is the famous Chinese remainder theorem which is important in the computation of determinant through modular computation algorithm.

Theorem 2.6. *(Chinese remainder theorem) Let $t_1, t_2, \ldots, t_k$ be pairwise relatively prime positive integers. Denote $T = t_1 \cdot t_2 \cdots t_k$. Then, for every integers $a_1, a_2, \cdots, a_k$ the system of congruence's*

$$\begin{cases} a \equiv a_1 (mod\ t_1) \\ a \equiv a_2 (mod\ t_2) \\ \vdots \\ a \equiv a_k (mod\ t_k). \end{cases} \tag{2.5}$$

has a unique solution a in the set $\{0, 1, \ldots, T-1\}$.

Proof. See (page 5, [5]) for a concise proof of Chinese remainder theorem. □

Lemma 2.7. *Let $m_1, m_2 \in \mathcal{R}$ be two relative prime moduli and $b_1, b_2 \in \mathcal{R}$ two arbitrary elements. The following are equivalent*

1. *$b_1 \equiv b_2 (mod\ m_1)$ and $b_1 \equiv b_2 (mod\ m_2)$*
2. *$b_1 \equiv b_2 (mod\ m_1 \cdot m_2)$*

Proof. For the proof to this lemma (see, [6]). □

The next Lemma which express the adjoint of a matrix as a product of the coefficients and powers of a matrix, is a useful tool in the proof of Samuelson's formula

Lemma 2.8. *Let M be a $n \times n$ matrix and $P_M(x) = a_0 + a_1 x + \ldots + a_n x^n (a_n = 1)$ its characteristic polynomial. Then*

$$adj(xI_n - M) = \sum_{j=1}^{n} \left(\sum_{i=1}^{j} a_{n-j+i} \cdot M^i \right) \cdot x^{n-j}$$

Proof. For a concise proof of this lemma (see, page 137 [6]) □

Theorem 2.9. *(Cauchy theorem). Let M and N be two matrices of the same size $n \times n$. Then*

$$det(M \cdot N) = det(M). \cdot det(N)$$

Proof. If one of the two matrices is singular, then the product AB is singular because

$$rank(MN) \leq min(rank(M), rank(N))$$

Therefore, $det(AB) = 0$ and at least one of $det(A) = 0$ or $det(MN) = 0$ is zero, so that $det(M) \cdot det(N) = 0$

Thus, the theorem holds if at least one of the two matrices is singular. Without any loss of generality, suppose neither of the matrices M and N is singular, then we can write them as products of elementary matrices as follows.

$M = U_1, U_2 \cdots U_n$ and $N = V_1, V_2 \cdots V_n$ where, $U_1, U_2 \cdots U_n$ and $V_1, V_2 \cdots V_n$ are elementary matrices. We recall that the determinant of a product of elementary matrices is equal to the products of their determinants. Therefore, we obtain

$$\begin{aligned} det(MN) &= det(U_1 \cdot U_2 \cdots U_n \cdot V_1 \cdot V_2 \cdots V_n) \\ &= det(U_1) \cdot det(U_2) \cdots det(U_n) \cdot det(V_1) \cdot det(V_2) \cdots det(V_n) \\ &= det(U_1 \cdot U_2 \cdots U_n) \cdot det(V_1 \cdot V_2 \cdots V_n)) \\ &= det(M) \cdot det(N) \end{aligned}$$

This completes the proof.

□

3 Samuelson-Berkowitz Algorithm

In this section we describe and formulate the Samuelson-Berkowitz algorithm and for finding the characteristics polynomial and the determinant of any square matrix whose coefficients in a unital commutative ring $\mathcal{R}$ which is not a field.

3.1 Description of Samuelson-Berkowitz Algorithm

The Samuelson-Berkowitz algorithm is an efficient method for computing the characteristic polynomial of a square matrix. It is particularly useful in our case because the Samuelson- Berkowitz algorithm is well behaved when the entries of the matrix belong to a unital commutative ring $\mathcal{R}$ without zero divisors. Given an $n \times n$ matrix, it recursively partitions the matrix into principal sub-matrices until it reaches the 1×1 sub-matrix in the upper left hand corner. It then assembles the coefficients of the characteristic polynomial by taking successively larger vector-matrix products.

The main idea behind Berkowitz's algorithm is Samuelson's formula, which relates the characteristics polynomial of a matrix to the characteristics polynomial of its principal sub-matrix. Thus, the coefficients of the characteristics polynomial of an $n \times n$ matrix M below are computed in terms of the coefficients of the characteristics polynomial of M.

3.2 Formulation of Samuelson-Berkowitz Algorithm

In this part of our project, we shall provide a comprehensive explanation of the formulation of Samuelson-Berkowitz algorithm (adapted from [6], [7]).

Theorem 3.1. *(Samuelson Identity). Assume that the characteristics polynomial of sub-matrix k is $P_k = a_0 + a_1 x + \ldots + a_k x^k$, with $a_k = 1$, then*

$$P_{k+1} = (x - m_{k+1,k+1})P_k - \sum_{j=1}^{k}(\sum_{i=1}^{j} a_{k-j+1} R_k . M_k^{i-1}) \cdot x^{k-j} \tag{3.6}$$

Proof. By definition $P_{k+1} = det(xI_{k+1} - M_{k+1})$, then using the proposed subdivision of M_{k+1} we may write

$$P_{k+1} = det\left(\begin{pmatrix} xI_k & 0 \\ 0 & x \end{pmatrix} - \begin{pmatrix} M_k & C_k \\ R_k & m_{k+1,k+1} \end{pmatrix} \right)$$

Expanding the determinants along the last row using Laplaces's formula yield

$$= \sum_{j}^{k} (-1)^{j+k+1} \cdot (-m_{k+1,j} \cdot detD_j + (x - m_{k+1,K+1} \cdot det(xI_k - M_k)$$

Where, D_j is a $k \times k$ matrix obtained from $(xI_k - M_k|C_k)$ by dropping the $j-th$ column. Expanding each determinant along the last column, again using Laplace's expansion in Theorem 2.1 yield

$$= \sum_{j=1}^{k} \left((-1)^{j+k+1} \cdot (-m_{k+1,j}) \cdot \sum_{i=1}^{k} (-1)^{i+k} \cdot detM_{ij} \right) + (x - m_{k+1,K+1}) \cdot P_k$$

Next, the matrix M_{ij} is nothing else but $(xI_k - M_k)$ with the $i-th$ row and $j-th$ column deleted. Rearranging the terms we obtain

$$= (-1) \sum_{j=1}^{k} \left(m_{k+1,j} \cdot \sum_{i=1}^{k} (-1)^{i+j} \cdot detM_{ij} \cdot m_{i,k+1} \right) + (x - m_{k+1,K+1}) \cdot P_k$$

Observe that, $(-1)^{i+j} M_{ij}$ is the entry of $adj(xI_k - M_k)$ located in the $j-th$ row and $i-th$ column. The double sum can now be interpreted as a matrix multiplication so that we have

$$= (x - m_{k+1,k+1}) \cdot P_k - R_k \cdot adj(xI_k - M_k) \cdot C_k$$

It then follows by Lemma 2.8 that,

$$= (x - m_{k+1,k+1}) \cdot P_k - R_k \cdot \sum_{j=1}^{k} \left(\sum_{i=1}^{j} a_{k-j+1} M_k^{i-1} \right) \cdot x^{k-j} \cdot C_k$$

This completes the proof. □

Example 3.2. Compute the determinant and characteristics polynomial of the matrix

$$M = \begin{pmatrix} 2 & 1 & 3 \\ 5 & -7 & 1 \\ 3 & 0 & -6 \end{pmatrix}$$

using Samuelson's identity

Employing the Samuelson's formula discussed above we have:
The first principal minor M_1 consists of a single entry at the top-left corner of M, that is $M_1 = 2$. The characteristic polynomial $p_1 = x - 2$.
We subdivide the second principal minor to obtain

$$\left[\begin{array}{c|c} 2 & 1 \\ \hline 5 & -7 \end{array} \right]$$

From which, $R_1 = (5)$ and $C_1 = -7$, then compute p_2 using the result in Theorem 3.1 and let $p_{i,j}$ denotes the coefficient at x^j in p_i.

$$p_2 = (x - m_{22}) \cdot p_1 - p_{1,1} \cdot R_1 M_1^0 C_1 \cdot x^0 = (x + 7)(x - 2) - 5 = x^2 + 5x - 19$$

Finally, we compute p_3

$$\left[\begin{array}{cc|c} 2 & 1 & 3 \\ 5 & -7 & 1 \\ \hline 3 & 0 & -6 \end{array}\right]$$

where, $R_2 = (3, 0), \quad C_2 = \begin{bmatrix} -3 \\ 1 \end{bmatrix}, \quad m_{33} = -6, \;\; p_{2,1} = 5$ and $p_{2,2} = 1$

$$\begin{aligned} p_3 &= (x - m_{33}) \cdot p_2 - (p_{2,2} \cdot R_2 M_2^0 C_2 x + (p_{2,1} \cdot R_2 M_2^0 C_2 + p_{2,2} \cdot R_2 M_2^1 C_2) \cdot x^0) \\ &= (x + 6)(x^2 + 5x - 19) - (9x + (45 + 21)) = x^3 + 11x^2 - 2x - 180 \end{aligned}$$

The characteristics polynomial of M is $P_M(x) = x^3 + 11x^2 - 2x - 180$.
It follows by Remark 2.3 that

$$detM = (-1)^3 P_M(0) = 180$$

Berkowitz Algorithm

Given an $n \times n$ matrix M over a commutative ring $\mathcal{R}$, Berkowitz algorithm computes an $(n+1) \times 1$ column vector P_M (characteristics polynomial). That is P_M is $(p_n, p_{n-1}, \ldots, p_0)$. The p_i are the coefficients of the $n-th$ degree polynomial given by $det(xI - M)$.

The main idea in the standard proof of Berkowitz's algorithm (see, [8]) is Samuelson's identity, which relates the characteristics polynomial of a matrix to the characteristics polynomial of its principal sub-matrix.

Let $M = (m_{ij})$ for $1 \leq i, j \leq n$ be a fixed $n \times n$ matrix and M_k be the principal minor of M of size k, for any $k \in \{1, 2, \ldots, n\}$. That is, M_k is the sub-matrix of M consisting of elements located in the first k rows and columns. In particular, $M_1 = (m_{11})$ and $M_n = M$. Subdivide the minor M_{k+1} into four cells as shown below:

$$\left[\begin{array}{c|c} M_k & C_k \\ \hline R_k & m_{k+1,k+1} \end{array}\right]$$

where, $C_k = (m_{k+1,1}, \ldots, m_{k+1,k})$ is a row vector and $R_k = (m_{1,k+1}, \ldots, m_{k,k+1})^T$ is a column vector. Let $P_k = P_{M_k}$ be the characteristic polynomial of the sub-matrix M_k. The idea behind Berkowitz algorithm is to express P_{k+1} in terms of P_k. This way we can recursively build all the successive characteristic polynomials. Let T_k be an $(K+2) \times (k+1)$ Toeplitz and lower-triangular matrix, where the entries in the $i - th$ row and $j - th$ column equals:

- $R_k \cdot M_k^{j-i-1} \cdot C_k$ if $i < j - 1$,
- $-m_{k+1,k+1}$ if $i = j - 1$,
- 1 if $i = j$,
- 0 if $i > j$.

The Toeplitz matrix T_k whose the description of it's entries explained above is of the form

$$T_k = \begin{pmatrix} -m_{k+1,k+1} & -R_kC_k & \cdots & R_kM_k^{k-1}C_k \\ 1 & -m_{k+1,k+1} & \cdots & R_kM_k^{k-2}C_k \\ 0 & 1 & \ddots & R_kM_k^{k-3}C_k \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{3.7}$$

We define the coefficients of the characteristics polynomial, for a given matrix M, to be the output of Berkowitz's algorithm, i.e., to be the entries of the column vector $P_M = T_1 \cdot T_2 \cdots T_n$.

In other words, if $(l_0 \dots l_{k+1} = 1)$ are the coefficients of the characteristic polynomial P_{k+1} of sub-matrix M_{k+1} and $(w_0, \dots, w_{k+1} = 1)$ the coefficients of the characteristic polynomial P_k of sub-matrix M_{k+1}. The Samuelson's formula can now be interpreted in terms of matrix multiplication as

$$\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_{k+1} \end{bmatrix} = T_k \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{k+1} \end{bmatrix}$$

3.3 Implementation of Samuelson-Berkowitz Algorithm

Given a $n \times n$ square matrix M with coefficients in a commutative ring $\mathcal{R}$, this algorithm computes the characteristic polynomial P_M of M.

1. Initialize $V = \begin{bmatrix} -m_{11} \\ 1 \end{bmatrix}$
2. for every $k \in \{1, \dots, n-1\}$ proceed as follows:
   a. let M_k be the $k-th$ principal minor of M
   b. set $R_k = (m_{k+1,1}, \dots, m_{k+1,k})$ and $C_k = (m_{1,k+1}, \dots, m_{k,k+1})^T$
   c. compute the products
   $$-R_kM_k^0C_k, -R_kM_k^1C_k \cdots - R_kM_k^{k-1}C_k$$
   d. construct a Toeplitz matrix in (3.7)
   e. update V setting $V = T_k \cdot V$
3. return $P_M(x) = (1, x, \dots, x^{n-1}, x^n) \cdot V$

3.4 Complexity of Samuelson-Berkowitz Algorithm

It is of utmost importance to expatiate on the computations of step (2) in Section 3.3 being the most important step that speed up the algorithm. It can be performed much more efficiently when executed serially by using matrix vector multiplication rather than matrix-matrix multiplications.

More precisely, instead of computing M_k^{k-1} for $1 \leq k \leq n$ we first compute matrix-vector product $M_k^{k-1}C_k$ and then the dot product $R_k M_k^{k-1} C_k$ for $1 \leq k \leq n$. In this case the algorithm is shown to involves less than $\frac{1}{2}n^4 - n^3 + \frac{5}{2}n^2$ arithmetic operations (additions, subtraction and multiplication) of coefficients in the commutative ring $\mathcal{R}$ (see, [9]).

4 Modular Computation Algorithm

This section provide a comprehensive explanation of the formulation of modular algorithm for calculating the determinant of any $n \times n$ matrix with integral coefficients.

4.1 Description of Modular Computation Algorithm

Modular computation algorithm is a method centred around the methods of division-free algorithm and fast division algorithm, for example, Gaussian elimination method. The main idea behind modular computation algorithm is to compute the determinant of square matrix M with coefficients in $\mathbb{Z}$ modulo an odd number of distinct primes p_i, and then use the Gaussian elimination method or any method to find the determinant d_i of the systems obtained and finally use Chinese remainder theorem (see, Theorem 2.6) to obtain the determinant d of matrix M.

In other words, the determinant of M as a matrix over $\mathbb{Z}$ is equal to the determinant of M regarded as a matrix over $\mathbb{Z}/p\mathbb{Z}$, which provides a control on the intermediate computations, since we work over a finite field $p = \{0, \ldots, p-1\}$. The immediate question is: **how do we choose $\mathbf{p_i}$?** For choosing p_i (for odd i) we need an a-priori bound for the determinant of M which is obtained through the Hadamard's inequality.

4.2 Formulation of Modular Computation Algorithm

Let $M = (m_{ij})$ be a matrix of size $n \times n$ and fix a distinct primes p_i (for odd i). For every primes p_i we denote the matrix M modulo p_i by $\overline{m_{ij}} = (m_{ij} mod\ p_i)$. We compute the determinant of $\overline{M} = (\overline{m_{ij}})$ using the fact that congruence's preserves sums and products. By definition of determinants we have

$$det(\overline{M}) = \sum_{\sigma} sgn(\sigma) \cdot \prod_{i=1}^{n} \overline{m_{i\sigma(i)}} = \sum_{\sigma} sgn(\sigma) \cdot \prod_{i=1}^{n} m_{i\sigma(i)} = detM(mod\ p_i)$$

The following theorem is a step close to knowing how to determine the primes p_i which is the centre of focus of the modular computation algorithm.

Theorem 4.1. *(Hadamard inequality). Let $M = (m_{ij})$ be a matrix of size $n \times n$ with real coefficients, then*

$$|detM| \leq \prod_{j=1}^{n} \sqrt{\sum_{i=1}^{n} m_{ij}^2} \tag{4.8}$$

Proof. A very beautiful proof of this theorem which rely on QR decomposition of a non-singular matrix M combined with Cauchy theorem (see, Theorem 2.9) is provided in (page 144, [6]). $\square$

The following corollary helps to know the bound of the determinant (Hadamard bound) of the given square matrix.

Colorally 4.2. *Let there be a constant $C > 0$ such that $|m_{ij}| \leq C$ for every $i, j \in \{1, \ldots, n\}$, then*

$$|det(M)| \leq C^n \cdot \sqrt{n^n}$$

Proof. Since m_{ij} are real coefficients then Theorem 4.1 holds and also we have $m_{ij} \leq |m_{ij}| \leq C$, and $m_{ij}^2 \leq C^2$, for every $i, j \leq n$.
Then

$$|detM| \leq \prod_{j=1}^{n} \sqrt{\sum_{i=1}^{n} m_{ij}^2} \leq \prod_{j=1}^{n} \sqrt{\sum_{i=1}^{n} C^2} \leq \prod_{j=1}^{n} \sqrt{n \cdot C^2} = C^n \cdot \sqrt{n^n}$$

This completes the proof. $\square$

Next, we present an approach whose goal is to control the growth of the intermediate computations when calculating the determinant of M . Let $detM = d$ and suppose $m = p_1. \cdot p_2 \cdots p_i$ be such that $m > 2 \cdot C^n \cdot \sqrt{n^n}$

We have already observed that a congruence $det(M \ mod \ p_i) \equiv detM(mod \ p_i)$ holds for every prime p_i . Lemma 2.7 asserts that $detM \equiv d \ (mod \ m)$. Hadamard's inequality says that if $d < \frac{m}{2}$, then the congruence $d - detM \equiv 0(mod \ m)$ implies that $d = detM > 0$. If on the other hand, we have $d > \frac{m}{2}$ (it cannot be equal $\frac{m}{2}$, since the number of primes p_i are odd), then $detM = d - m < 0$.

4.3 Implementation of Modular Computation Algorithm

Given a matrix $M = (m_{ij})$ with integer entries, this algorithm computes the determinant $detM$.

1. find a bound C such that $C \geq |m_{ij}|$ for all $i, j \leq n$;
2. find odd primes $p_1 \cdots p_s$, whose product is greater than twice the Hadamard's bound, i.e.
$$m = p_1 \cdots p_s > 2 \cdot C^n \cdot \sqrt{n^n}$$
3. for every $k \in \{1, 2, \ldots, s\}$ compute the determinant over $\mathbb{F}_{p_s}$ using e.g. Gaussian elimination or any other method;
4. use Chinese remainder theorem in Theorem 2.6 to find a solution d to the system of congruence's $d \equiv d_k \ (mod \ p_k)$ for every $k \in \{1, 2, \ldots, s\}$;
5. if $d > \frac{m}{2}$, then replace it by $d - m$;
6. return the determinant $detM = d$.

Example 4.3. Compute the determinant of a matrix

$$M = \begin{pmatrix} 9 & 2 & 1 \\ 5 & -1 & 6 \\ 4 & 0 & -2 \end{pmatrix}$$

using algorithm in Section 4.3.

Clearly, from the matrix M, we have, $n = 3$, $C = 9$ and the Hadamard's inequality gives the bound

$$|detM| \leq 9^3 \cdot \sqrt{3^3} = 3787.995$$

To this end, we consider five primes: $2, 3, 5, 7$ and 37, since the product $m = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 37 = 7,770$ is greater than twice the Hadamard's bound. We will successively compute the determinants modulo each prime using GEM, and then use Chinese remainder theorem in Theorem 2.6 to incrementally solve the resulting system of congruence's.

Firstly, we compute $det(M\ mod\ 2)$ as follows:

$$d_1 = det(M\ mod\ 2) = det \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = 0.$$

Next, we compute for the remaining primes

$$d_2 = det(M\ mod\ 3) = det \begin{pmatrix} 0 & 2 & 1 \\ 2 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix} = -6,$$

$$d_3 = det(M\ mod\ 5) = det \begin{pmatrix} 4 & 2 & 1 \\ 0 & 4 & 1 \\ 4 & 0 & 3 \end{pmatrix} = 40,$$

$$d_4 = det(M\ mod\ 7) = det \begin{pmatrix} 2 & 2 & 1 \\ 5 & 6 & 6 \\ 4 & 0 & 5 \end{pmatrix} = 34,$$

$$d_2 = det(M\ mod\ 37) = det \begin{pmatrix} 9 & 2 & 1 \\ 5 & 36 & 6 \\ 4 & 0 & 35 \end{pmatrix} = 10,894.$$

Therefore, it follows that

$$\begin{cases} d \equiv & 0\ (mod\ 2) \\ d \equiv & -6\ (mod\ 3) \\ d \equiv & 40\ (mod\ 5) \\ d \equiv & 34\ (mod\ 7) \\ d \equiv & 10,894\ (mod\ 37), \end{cases}$$

which can be reduced to

$$\begin{cases} d \equiv & 6\ (mod\ 7) \\ d \equiv & 16\ (mod\ 37), \end{cases}$$

from which we obtain,

$$d \equiv 90 \ (mod\ 259)$$

Then, $d = 90$ is a unique solution of the system and it is less than half times m. Therefore, $detM = 90$

4.4 Complexity of Modular Computation Algorithm

Prime numbers are frequent enough to find one with a word length in the same order of magnitude as $O(n^2)$ while computing the determinant of an $n \times n$ matrix over a finite field for each prime p_i using Gaussian elimination has a complexity of $O(n^3)$ and using the Chinese Remainder Theorem has a complexity of $O(n)$ involving simple arithmetic operations.

The complexities of each step demonstrate that the overall complexity of the algorithm is dominated by the step with the highest complexity, which is typically computing determinants $mod\ p_i$ (Step 3) with $O(n^3)$ complexity. For more explanations on the complexity of this algorithm (see, [10]). Therefore, the cost of computations in modular computation algorithm is $0(n^3)$.

References

[1] José E Moyal. Quantum mechanics as a statistical theory. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 45, pages 99–124. Cambridge University Press, 1949.

[2] Svante Wold. Spline functions in data analysis. *Technometrics*, 16(1):1–11, 1974.

[3] Joris van Der Hoeven. Fast chinese remaindering in practice. In *Mathematical Aspects of Computer and Information Sciences: 7th International Conference, MACIS 2017, Vienna, Austria, November 15-17, 2017, Proceedings 7*, pages 95–106. Springer, 2017.

[4] James B Carrell. An introduction to the theory of determinants. In *Groups, Matrices, and Vector Spaces: A Group Theoretic Approach to Linear Algebra*, pages 113–134. Springer, 2017.

[5] Dingyi Pei, Arto Salomaa, and Cunsheng Ding. *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.

[6] Przemysław Koprowski. Lcm: Lectures on computational mathematics. 2022.

[7] Michael Soltys-Kulinicz. *The complexity of derivations of matrix identities*. Citeseer, 2001.

[8] Stuart J Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information processing letters*, 18(3):147–150, 1984.

[9] Jounaidi Abdeljaoued. The berkowitz algorithm, maple and computing the characteristic polynomial in an arbitrary commutative ring. *MapleTech*, 4(3):21–32, 1997.

[10] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.