Computations in Modules over Commutative Domains

Alkiviadis G. Akritas[1] and Gennadi I. Malaschonok[2]

[1] University of Thessaly, Department of Computer and Communication Engineering, GR-38221 Volos, Greece
akritas@uth.gr
[2] Tambov State University, Internatsionalnaya 33, Tambov 392000, Russia
malaschonok@ya.ru

Abstract. This paper is a review of results on computational methods of linear algebra over commutative domains. Methods for the following problems are examined: solution of systems of linear equations, computation of determinants, computation of adjoint and inverse matrices, computation of the characteristic polynomial of a matrix.

1 Introduction

Let $\mathbf{R}$ be a commutative domain with identity, $\mathbf{K}$ the field of quotients of $\mathbf{R}$. This paper is devoted to the review of effective matrix methods in the domain $\mathbf{R}$ for a solution of standard linear algebra problems. The problems are: solving linear systems in $\mathbf{K}$, computing the adjoint and inverse matrix, computing the matrix determinant and computing the characteristic polynomial of a matrix.

The standard used to tackle these problems in commutative domain $\mathbf{R}$ consists of the using the field of fractions $\mathbf{K}$ of this domain. The ring $\mathbf{R}$ may be canonically immersed in the field $\mathbf{K}$. To solve a problem in the commutative domain any algorithm that is applicable over the field of fractions of this domain you can be applied.

Unfortunately this way results in algorithms with suitable complexity *only* in the case where the cost of operations in the field does not depend on the value of the operands. As an example consider the finite fields. But in the general case the cost of operations in the field depends on the value of the operands. More over this cost, in general, grows very quickly. For example, Gauss' method in the ring of integer numbers results in an algorithm that has exponential growth of complexity — instead of cubic.

So the main aim in commutative domains is to construct algorithms with controlled intermediate results.

The algorithms presented here have two main features:

- The intermediate elements in the algorithms are minors of the initial matrix. So the growth of these elements is bounded by the maximal value of the minors of the initial matrix.

- With the exception of the last algorithm, which requires $O(n^3)$ operations, the number of operations in all other algorithms is the same as that of the algorithm for matrix multiplication.

We denote by $O(n^\beta)$ or by $\gamma n^\beta + o(n^\beta)$ the number of multiplication operations, necessary for the multiplication of square matrices of order n. For the standard matrix multiplication algorithm we have $\beta = 3$ and $\gamma = 1$, whereas for Strassen's algorithm [21] the values are $\beta = \log_2 7$ and $\gamma = 1$, when the order of the matrix is some power of 2. For the best algorithm today we have $\beta < 2.376$ and γ unknown [8].

In the second section we present methods for solving systems of linear equations and performing determinant computations. Included are: Dodgson's method [9], the method of Forward and Backward Direction [12], the One-pass method [14], [15] and the Recursive Method [17],[18]. Corresponding methods for determinant computations with some generalization are discussed in [16].

Methods for computing the adjoint and inverse matrices are presented in the third section [20].

In the forth section a method for computing the characteristic polynomial of a matrix is presented [19]. This method was developed in [5].

Finally, in the conclusion we present the best complexity bounds available today (in commutative domains) for the methods presented above.

2 System of Linear Equations

Let R be a commutative domain, F be the field of fractions of R, $A \in R^{n\times m}, c \in R^n, n \leq m, A^\star = (A, c) = (a_{ij})$ and,

$$Ax = c$$

be a system of linear equations over R.

Solving the above system with Cramer's rule we obtain

$$x_i = (\delta^n_{i,m+1} - \textstyle\sum_{j=n+1}^m x_j \delta^n_{ij})(\delta^n)^{-1},\ i = 1 \ldots n,$$

where $x_j, j = n+1, \ldots, m$, are the free variables, and the determinant $\delta^n \neq 0$.

We denote by $\delta^k, k = 1, \ldots, n$ the left upper corner minor of matrix A of order k, and by δ^k_{ij} the corner minor of matrix A where columns i and j have been interchanged. We assume that all corner minors $\delta^k, k = 1, \ldots, n$ are different from 0.

2.1 Dodgson's Algorithm

The determinant identity

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & k \end{vmatrix} = \begin{vmatrix} \begin{vmatrix} a & b \\ d & e \end{vmatrix} & \begin{vmatrix} b & c \\ e & f \end{vmatrix} \\ \\ \begin{vmatrix} d & e \\ g & h \end{vmatrix} & \begin{vmatrix} e & f \\ h & k \end{vmatrix} \end{vmatrix} \cdot e^{-1}$$

or in the more general form

$$\hat{a}_{ij}^{k+1} = \begin{vmatrix} \hat{a}_{i-1,j-1}^{k} & \hat{a}_{i-1,j}^{k} \\ \hat{a}_{i,j-1}^{k} & \hat{a}_{ij}^{k} \end{vmatrix} \cdot (\hat{a}_{i-1,j-1}^{k-1})^{-1}$$

where

$$\hat{a}_{ij}^{k+1} = |A_{j-k,\ldots,j:(columns)}^{i-k,\ldots,i:(rows)}|$$

is an instance of Sylvester's identity [2]. Dodgson [9] used it for the computation (condensation) of determinants and the solution of systems of linear equations computing a sequence of minors for

$$k = 2, \ldots, n-1,\ i = k+1, \ldots, n,\ j = k+1, \ldots, m.$$

Historical Note: As can be seen from the identity above, Dodgson liked to take the middle element e of the 4 corner minors as the leading minor (element). Later (in their 1945 paper [23]) Waugh and Dwyer took the top-left-corner element, a_{11} as the "middle " element of the 4 minors that are surrounding this a_{11} element.

Subsequent authors [22] and [6] used the same method without references to either Dodgson [9] or Waugh and Dwyer [23]. Other implementations of Dodgson's method can be found in [1] Other implementations of Dodgson method you can see in the book [1].

2.2 Method of Forward and Backward Direction

The forward direction part of this algorithm consists of computing the minors with Dodgson's method; the diagonal is the leading element in every step

$$a_{ij}^{k+1} = (a_{kk}^{k} a_{ij}^{k} - a_{ik}^{k} a_{kj}^{k})(a_{k-1,k-1}^{k-1})^{-1},$$

$$k = 2, \ldots, n-1,\ i = k+1, \ldots, n,\ j = k+1, \ldots, m,$$

where

$$a_{ij}^{k+1} = |A_{1,\ldots,k,j:(columns)}^{1,\ldots,k,i:(rows)}|$$

On one hand, this formula is a determinant identity and on the other hand it is the forward direction algorithm which is reminiscent of Gauss' elimination algorithm. (The only difference is that the leading element is one step behind.) As a result of the forward direction algorithm the matrix of the system becomes

$$\begin{pmatrix} a_{1,1}^1 & a_{1,2}^1 & \cdots & a_{1,n-1}^1 & a_{1,n}^1 & a_{1,n}^1 & \cdots & a_{1,m+1}^1 \\ 0 & a_{2,2}^2 & \cdots & a_{2,n-1}^2 & a_{2,n}^2 & a_{2,n+1}^2 & \cdots & a_{2,m+1}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1,n-1}^{n-1} & a_{n-1,n}^{n-1} & a_{n-1,n+1}^{n-1} & \cdots & a_{n-1,m+1}^{n-1} \\ 0 & 0 & \cdots & 0 & a_{n,n}^{n} & a_{n,n+1}^{n} & \cdots & a_{n,m+1}^{n} \end{pmatrix}$$

The leading elements $a_{k,k}^k$, $k = 1, \ldots, n-1$ cannot be zero.

The backward direction part of the algorithm consists of computing the minors δ_{ij}^n. The minor δ_{ij}^k is the corner minor of order k of the matrix A after column i has been interchanged with column j. The determinant identity of the backward direction algorithm is:

$$\delta_{ij}^n = \big(a_{nn}^n a_{ij}^i - \sum_{k=i+1}^{n} a_{ik}^i \delta_{kj}^n\big)(a_{ii}^i)^{-1}, \quad i = n-1, \dots 1, \quad j = n+1, \dots, m.$$

As a result of the backward direction algorithm the matrix of the system becomes:

$$\begin{pmatrix} a_{n,n}^n & 0 & \cdots & 0 & 0 & \delta_{1,n}^n & \cdots & \delta_{1,m+1}^n \\ 0 & a_{n,n}^n & \cdots & 0 & 0 & \delta_{2,n+1}^n & \cdots & \delta_{2,m+1}^n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n}^n & 0 & \delta_{n-1,n+1}^n & \cdots & \delta_{n-1,m+1}^n \\ 0 & 0 & \cdots & 0 & a_{n,n}^n & \delta_{n,n+1}^n & \cdots & \delta_{n,m+1}^n \end{pmatrix}$$

The number of operations, necessary for the procedure of forward and backward direction, is

$N^m = (9n^2m - 5n^3 - 3nm - 3n^2 - 6m + 8n)/6,$
$N^d = (3n^2m - n^3 - 3nm - 6n^2 + 13n - 6)/6$
$N^a = (6n^2m - 4n^3 - 6nm + 3n^2 + n)/6.$

2.3 The One-Pass Method

Another way of computing the minors δ_{ij}^k is given by the following two determinant identities:

$$\delta_{k+1,j}^{k+1} = a_{k+1,k+1}\delta_{kk}^k - \sum_{p=1}^{k} a_{k+1,p}\delta_{pj}^k, j = k+1 \dots m,$$

$$\delta_{ij}^{k+1} = (\delta_{k+1,k+1}^{k+1}\delta_{i,j}^k - \delta_{k+1,j}^{k+1}\delta_{i,k+1}^k)/\delta_{k,k}^k,$$
$$k = 1, \dots, n-1, \ i = 1, \dots, k, \ j = k+2, \dots, m.$$

At the k-th step the coefficient matrix looks like

$$\begin{pmatrix} a_{k,k}^k & 0 & \cdots & 0 & \delta_{1,k+1}^k & \cdots & \delta_{1,m+1}^k \\ 0 & a_{k,k}^k & \cdots & 0 & \delta_{2,k+1}^k & \cdots & \delta_{2,m+1}^k \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{k,k}^k & \delta_{k,k+1}^k & \cdots & \delta_{k,m+1}^k \\ a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} & a_{k+1,k+1} & \cdots & a_{k+1,m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,k} & a_{n,k+1} & \cdots & a_{n,m+1} \end{pmatrix}$$

The number of operations, necessary for the one-pass algorithm, is

$N^m = (9n^2m - 6n^3 - 3nm - 6m + 6n)/6,$
$N^d = (3n^2m - 2n^3 - 3nm - 6m + 2n + 12)/6$
$N^a = (6n^2m - 4n^3 - 6nm + 3n^2 + n)/6.$

When the number of equations and unknowns in the system is the same and equal to n, the last two algorithms can be compared

Number of Operations			
Method	Multiplications	Divisions	Add./Substr.
FB	$\frac{(4n^3+3n^2-n-6)}{6}$	$\frac{(2n^3-6n^2+10n-6)}{6}$	$\frac{(2n^3+3n^2-5n)}{6}$
OP	$\frac{(n^3+2n^2-n-2)}{2}$	$\frac{(n^3-7n+6)}{6}$	$\frac{(2n^3+3n^2-5n)}{6}$

2.4 The Recursive Method

The minors δ^k_{ij} and a^k_{ij} are elements of the following matrices

$$A^{r,l,(p)}_{k,c} = \begin{pmatrix} a^p_{r+1,k+1} & a^p_{r+1,k+2} & \cdots & a^p_{r+1,c} \\ a^p_{r+2,k+1} & a^p_{r+2,k+2} & \cdots & a^p_{r+2,c} \\ \vdots & \vdots & \ddots & \vdots \\ a^p_{l,k+1} & a^p_{l,k+2} & \cdots & a^p_{l,c} \end{pmatrix},$$

$$G^{r,l,(p)}_{k,c} = \begin{pmatrix} \delta^p_{r+1,k+1} & \delta^p_{r+1,k+2} & \cdots & \delta^p_{r+1,c} \\ \delta^p_{r+2,k+1} & \delta^p_{r+2,k+2} & \cdots & \delta^p_{r+2,c} \\ \vdots & \vdots & \ddots & \vdots \\ \delta^p_{l,k+1} & \delta^p_{l,k+2} & \cdots & \delta^p_{l,c} \end{pmatrix},$$

$G^{r,l,(p)}_{k,c}, A^{r,l,(p)}_{k,c} \in \mathbf{R}^{(l-r)\times(c-k)}$, $0 \le k < n$, $k < c \le n$, $0 \le r < m$, $r < l \le m$, $1 \le p \le n$.

We describe one recursive step reducing the matrix $\tilde{A} = A^{k,l,(k+1)}_{k,c}$ to the diagonal form

$$\tilde{A} \to (\delta^l I_{l-k}, \hat{G})$$

where

$$\tilde{A} = A^{k,l,(k+1)}_{k,c}, \quad \hat{G} = G^{k,l,(l)}_{l,c}$$

$0 \le k < c \le m$, $k < l \le n$, $l < c$. Note that if $k = 0$, $l = n$ and $c = m$, then we obtain the solution of the original system.

Description of One Step of the Recursive Method

$$\tilde{A} = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix} \to_1 \begin{pmatrix} \delta^s I_{s-k} & G^1_2 \\ A^2_1 & A^2_2 \end{pmatrix} \to_2 \begin{pmatrix} \delta^s I_{s-k} & G^1_2 \\ 0 & \hat{A}^2_2 \end{pmatrix} \to_3$$

$$\rightarrow_3 \begin{pmatrix} \delta^s I_{s-k} & G^1_{2'} & G^1_{2''} \\ 0 & \delta^l I_{l-s} & \hat{G}^2_{2''} \end{pmatrix} \rightarrow_4 \begin{pmatrix} \delta^l I_{s-k} & 0 & \hat{G}^1_{2''} \\ 0 & \delta^l I_{l-s} & \hat{G}^2_{2''} \end{pmatrix} = \begin{pmatrix} \delta^l I_{l-k} & \hat{G} \end{pmatrix}$$

We may choose arbitrary numbers s: $k < s < l$ and write the matrix $\tilde{A}$ as follows:

$$\tilde{A} = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix},$$

where $A^1 = A^{k,s,(k+1)}_{k,c}$ is the upper part of the matrix $\tilde{A}$ consisting of $s-k$ rows and $A^2 = A^{s,l,(k+1)}_{k,c}$ is the lower part of the matrix $\tilde{A}$.

$$A^1 \rightarrow (\delta^s I_{s-k}, G^1_2), \tag{I}$$

where $A^1 \in \mathbf{R}^{(s-k)\times(c-k)}$, $G^1_2 = G^{k,s,(s)}_{s,c}$.

Let $A^2 = (A^2_1, A^2_2)$ where $A^2_1 = A^{s,l,(k+1)}_{k,s}$ and $A^2_2 = A^{s,l,(k+1)}_{s,c}$ consisting of $s-k$ and $c-s$ columns respectively, $\delta^k \neq 0$. The matrix $\hat{A}^2_2 = A^{s,l,(s+1)}_{s,c}$ is computed with the help of the matrix identity

$$\hat{A}^2_2 = (\delta^s \cdot A^2_2 - A^2_1 \cdot G^1_2)(\delta^k)^{-1}. \tag{II}$$

$$\hat{A}^2_2 \rightarrow (\delta^l I_{l-s}, \hat{G}^2_{2''}), \tag{III}$$

where $\hat{A}^2_2 \in \mathbf{R}^{(l-s)\times(c-s)}$ and $\hat{G}^2_{2''} = G^{s,l,(l)}_{l,c}$.

Let $G^1_2 = (G^1_{2'}, G^1_{2''})$, where the blocks $G^1_{2'} = G^{k,s,(s)}_{s,l}$ and $G^1_{2''} = G^{k,s,(s)}_{l,c}$ contain $l-s$ and $c-l$ columns respectively, and $\delta^s \neq 0$.

The matrix $\hat{G}^1_{2''} = G^{k,s,(l)}_{l,c}$ is computed with the help of the matrix identity

$$\hat{G}^1_{2''} = (\delta^l \cdot G^1_{2''} - G^1_{2'} \cdot \hat{G}^2_{2''})(\delta^s)^{-1}. \tag{IV}$$

In the result we obtain δ^l and

$$\hat{G} = \begin{pmatrix} \hat{G}^1_{2''} \\ \hat{G}^2_{2''} \end{pmatrix}$$

Complexity of the Recursive Method is $O(mn^{\beta-1})$

We can obtain an exact estimate. For $n = 2^N, m = n+1$ and $\beta = \log_2 7$ the number of multiplication operations is

$$\frac{7}{15} n^{\log_2 7} + n^2(\log_2 n - \frac{2}{3}) + n(2\log_2 n + \frac{1}{5}).$$

For $n = 2^N,\ \beta = 3$ the number of multiplications and divisions is

$N^m = (6n^2m - 4n^3 + (6nm - 3n^2)\log_2 n - 6nm + 4n)/6,$
$N^d = ((6nm - 3n^2)\log_2 n - 6nm - n^2 + 6m + 3n - 2)/6.$

The number of multiplication operations for $m = n + 1$ is $(1/3)n^3 + O(n^2)$. The estimations for the previous two methods are, respectively, $n^3 + O(n^2)$ and $(2/3)n^3 + O(n^2)$.

3 Adjoint Matrix

The best method for computing the matrix determinant and adjoint matrix in the arbitrary commutative ring was suggested in the papers by Kaltofen [10] and Kaltofen and Villard [11]. Its complexity is $O(n^{\beta+1/3} \log n \log\log n)$; see also [3].

Here we describe the best method for computing adjoint matrices in commutative domains. Let $\mathcal{A} = \begin{pmatrix} A & C \\ B & D \end{pmatrix}$ be an invertible matrix and A an invertible block. Then

$$\mathcal{A}^{-1} = \begin{pmatrix} I & -A^{-1}C \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & (D - BA^{-1}C)^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & I \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & I \end{pmatrix}$$

is the factorization of the inverse matrix. This requires two multiplication operations and two inversions of blocks. In case $n = 2^p$ it will take 2^{p-1} inversions of 2×2 blocks and 2^{p-k} multiplications of $2^k \times 2^k$ blocks.

Overall, $n^{log7} - n/2$ multiplication operations will be needed, if we use Strassen's multiplication algorithm. In general, if the complexity of matrix multiplication is $O(n^\beta)$, then the computation of the factors of the inverse matrix can be done in time $O(n^\beta)$.

Let R be a commutative ring, and let $\mathcal{A} = (a_{i,j})$ be a square matrix of order n over the ring R. Let

$$\mathcal{A}_t^{(s)} = (a_{i,j}^s)_{j=s,\dots,t}^{i=s,\dots,t} \text{ and } \mathcal{G}_s^{(t)} = (\delta_{t(i,j)})_{j=t+1,\dots,n}^{i=s,\dots,t}$$

Theorem 1. *Let $\mathcal{A}$ be a square block matrix of order n over the ring R; that is,*

$$\mathcal{A} = \begin{pmatrix} A & C \\ B & D \end{pmatrix},$$

where A is a square block of order s, $(1 < s < n)$, the determinant of which, δ_s, is neither zero nor a zero divider in R. Then, the adjoint matrix $\mathcal{A}^*$ can be written as the product

$$\mathcal{A}^* = \begin{pmatrix} \delta_s^{-1}\delta_n I & -\delta_s^{-1}FC \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & \delta_s I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & I \end{pmatrix}, \qquad (*)$$

where $F = A^*$, $G = \delta_s^{-n+s+1}\mathcal{A}_n^{(s+1)*}$, I is the identity matrix and we have the identity

$$\mathcal{A}_n^{(s+1)} = \delta_s D - BFC.$$

Theorem 2. *Let $\mathcal{A}_n^{(s+1)}$ be a square block matrix of order $n-s$, $(s>0,\ n-s>2)$, over the ring R; that is,*

$$\mathcal{A}_n^{(s+1)} = \begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{B} & \mathbf{D} \end{pmatrix},$$

where $\mathbf{A}$ is a square block of order $t-s$, $(1<s<t<n)$, and δ_s and δ_t are neither zero nor zero dividers in R. Then, the matrix $\delta_s^{-n+s+1}\mathcal{A}_n^{(s+1)*}$ can be written as the product

$$\begin{pmatrix} \delta_t^{-1}\delta_n I & -\delta_t^{-1}\mathbf{FC} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \delta_s^{-1}\mathbf{G} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\mathbf{B} & \delta_t I \end{pmatrix} \begin{pmatrix} \mathbf{F} & 0 \\ 0 & I \end{pmatrix}, \qquad (**)$$

where $\mathbf{F} = \delta_s^{-t+s+1}\mathcal{A}_t^{(s+1)*}$, $\mathbf{G} = \delta_t^{-n+t+1}\mathcal{A}_n^{(t+1)*}$, I is the identity matrix and we have the identity

$$\mathcal{A}_n^{(t+1)} = \delta_s^{-1}(\delta_t\mathbf{D} - \mathbf{BFC}).$$

Remark 1. If $n=s+2$, then, $\mathcal{A}_n^{(s+1)*} = \begin{pmatrix} a_{n,n}^{s+1} & -a_{n-1,n}^{s+1} \\ -a_{n,n-1}^{s+1} & a_{n-1,n-1}^{s+1} \end{pmatrix}$. And if $n=s+1$, then $\mathcal{A}_n^{(s+1)*} = 1$.

3.1 Dichotomic Process

The dimensions of the upper left block A (of the initial square block matrix $\mathcal{A}$) may be chosen arbitrarily. The case will be examined when the dimensions of block A are powers of two.

Let n be the order of the matrix $\mathcal{A}$, $2^h < n \leq 2^{h+1}$ and assume that all minors δ_{2i}, $i=1,2,\dots$ are not zero or zero dividers of the ring R. According to Theorems 1 and 2 we are going to sequentially compute adjoint matrices for the upper left blocks of order $2,4,8,16,\dots$ of matrix $\mathcal{A}$.

1. For the block of order 2 we have:

$$A_{2,2}^2 = (a_{i,j})_{i,j=1,2},\ \delta_2 = \det A_{2,2}^1,$$

$$A_{2,2}^{2*} = \begin{pmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{pmatrix}.$$

2. For the block of order 4 we have:

$$A_{4,4}^{4*} = \begin{pmatrix} \delta_2^{-1}\delta_4 I & -\delta_2^{-1}FC \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & \delta_2 I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & I \end{pmatrix},$$

$F = A_{2,2}^{2*}$, $B = (a_{i,j})_{j=1,2}^{i=3,4}$,

$C = (a_{i,j})_{j=3,3}^{i=1,2}$, $D = (a_{i,j})_{i,j=3,4}$, $\mathcal{A}_4^{(3)} = \delta_2 D - BFC = (a_{i,j}^3)_{i,j=3,4}$, $G = \delta_2^{-1}\mathcal{A}_4^{(3)*}$, $\delta_4 = \det G$.

3. For the block of order 8 we have:

$$A_{8,8}^{8*} = \begin{pmatrix} \delta_4^{-1}\delta_8 I & -\delta_4^{-1}FC \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & \delta_4 I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & I \end{pmatrix},$$

$F = A_{4,4}^{4*}$, $B = (a_{i,j})_{j=1,\dots,4}^{i=5,\dots,8}$, $C = (a_{i,j})_{j=5,\dots,8}^{i=1,\dots,4}$, $D = (a_{i,j})_{i,j=5,\dots,8}$,

$$G = \begin{pmatrix} \delta_6^{-1}\delta_8 I & -\delta_6^{-1}\mathbf{FC} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \delta_4^{-1}\mathbf{G} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\mathbf{B} & \delta_6 I \end{pmatrix} \begin{pmatrix} \mathbf{F} & 0 \\ 0 & I \end{pmatrix},$$

$\mathcal{A}_8^{(5)} = \delta_4 D - BFC = (a_{i,j}^5)_{i,j=5,\dots,8}$, $\mathbf{F} = \delta_4^{-3}\mathcal{A}_6^{(5)*}$, $\delta_6 = \det \mathbf{F}$, $\mathbf{B} = (a_{i,j}^5)_{j=5,6}^{i=7,8}$, $\mathbf{C} = (a_{i,j}^5)_{j=7,8}^{i=5,6}$, $\mathbf{D} = (a_{i,j}^5)_{i,j=7,8}$, $\mathcal{A}_8^{(7)} = \delta_4^{-1}(\delta_6\mathbf{D} - \mathbf{BFC}) = (a_{i,j}^7)_{i,j=7,8}$, $\mathbf{G} = \delta_6^{-1}\mathcal{A}_8^{(7)*}$, $\delta_8 = \det \mathbf{G}$.

Complexity Estimation

Let $\gamma n^\beta + o(n^\beta)$ be an asymptotic estimation of the number of operations for multiplying two matrices of order n. Then the complexity of computing the adjoint matrix of order $n = 2^p$ is

$$F(n) = 6\gamma n^\beta \frac{1 - (n/2)^{1-\beta}}{2^\beta - 2} + o(n^\beta)$$

4 Characteristic Polynomial

In the case of an arbitrary commutative ring, the best algorithms for computing the characteristic polynomial are Chistov's algorithm [7] and the improved Berkowitz algorithm [4]. The complexity of these methods is $O(n^{\beta+1}\log n)$. We present the best method to date — for computations in commutative domains — which has complexity $O(n^3)$.

Let $A = (a_{ij})$ be an $n \times n$ matrix over the ring R. If all the diagonal minors δ_k $(k = 1, \dots, n-1)$ of matrix A are *not* zero, then the following identity holds

$$A_u = \tilde{L}A,$$

where A_u is an upper triangular matrix and $\tilde{L}$ is a lower triangular matrix with determinant different from zero, such that

$$\tilde{L} = \mathbf{D}_{n-2}^{-1}\tilde{\mathbf{L}}_{n-1} \cdots \mathbf{D}_1^{-1}\tilde{\mathbf{L}}_2\tilde{\mathbf{L}}_1$$

$\tilde{\mathbf{L}}_k = \operatorname{diag}(I_{k-1}, \tilde{L}_k)$, $\mathbf{D}_k = \operatorname{diag}(I_k, D_k)$, where I_k is the identity matrix of order k, $D_k = \delta_k I_{n-k}$,

$$L_k = \begin{pmatrix} \delta_k & 0 \\ v_k & I_{n-k} \end{pmatrix}, \tilde{L}_k = \begin{pmatrix} 1 & 0 \\ -v_k & \delta_k I_{n-k} \end{pmatrix}$$

$v_k = (a_{k+1,k}^k, \dots, a_{n,k}^k)^T$, $A_u = (a_{i,j}^{(n)})$ is an $n \times n$ matrix, and $a_{i,j}^{(n)} = a_{i,j}^i$, for $i \le j$, $a_{i,j}^{(n)} = 0$, for $i > j$.

The proof is based on Sylvester's identity

$$a^{k-1}_{k-1,k-1} a^{k+1}_{i,j} = a^{k}_{k,k} a^{k}_{i,j} - a^{k}_{i,k} a^{k}_{k,j}.$$

The factorization of matrix A into upper and lower triangular matrices is the result of the forward direction part — of the forward and backward direction algorithm.

Let $A_u^{(k)} = (a_{i,j}^{(k)})$ be an $n \times n$ matrix, $k = 1, \ldots, n$, with $a_{i,j}^{(k)} = a_{i,j}^{i}$ for $i \leq j < k$, $a_{i,j}^{(k)} = a_{i,j}^{k}$, $i \geq k$, $j \geq k$, and the remaining elements zero. Then $A_u = \tilde{L}A$ reduces to the identities

$$A_u^{(2)} = \tilde{\mathbf{L}}_1 A; \quad A_u^{(k+1)} = \mathbf{D}_{k-1}^{-1} \tilde{\mathbf{L}}_k A_u^{(k)}, \quad k = 2, \ldots, n-1,$$

which subsequently enable the computation of matrices $A_u^{(k)}$, $k = 2, 3, \ldots, n$, such that all the elements of the matrices $\mathbf{D}_k$ and $\tilde{\mathbf{L}}_k$ are elements of the matrix $A_u^{(k)}$.

The requirement that the diagonal minors δ_k $(k = 1, 2, \ldots, n-1)$ be different from zero may be weakened. If a diagonal minor δ_k of order k is equal to zero, and in column v_k there is a nonzero element $a_{i,k}^{k}$, then rows i and k must be interchanged; that is, multiply on the left the matrix of interchanges $P_k = P_{(i,k)} = I_n + E_{ik} + E_{ki} - E_{kk} - E_{ii}$, where E_{ik} denotes a matrix in which all elements are zero except element (i, k), which is equal to one.

And if $\delta_k = 0$ and $v_k = 0$, then necessarily $P_k = \tilde{\mathbf{L}}_k = \mathbf{D}_{k-1} = I_n$, $\mathbf{D}_k = \mathbf{D}_{k-1}$.

The factorization formula remains as before, only now

$$\tilde{\mathbf{L}}_k = \operatorname{diag}(I_{k-1}, \tilde{L}_k) P_k.$$

Note the following identities, which will be subsequently needed:

$$\tilde{L}_k L_k = D_k, \quad \tilde{L} L = T,$$

where

$$L = \mathbf{L}_1 \mathbf{L}_2 \cdots \mathbf{L}_{n-1},$$

$\mathbf{L}_k = P_k^{-1} \operatorname{diag}(I_{k-1}, L_k)$, T is a diagonal matrix defined by, $T = S_1 S_2$, where $S_1 = \operatorname{diag}(1, S)$, and $S_2 = \operatorname{diag}(S, 1)$, with $S = \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_{n-1})$.

To indicate the matrix A from which a given triangular or diagonal matrix was computed, we write $L = L(A), T = T(A)$.

4.1 Computation of Similar p-Trianular Matrix

Let $A = \begin{pmatrix} \mathbf{a}\,\mathbf{b} \\ \mathbf{c}\,\mathbf{d} \end{pmatrix}$ be a matrix over R with blocks $\mathbf{a}$ of order $p \times p$ and $\mathbf{d}$ of order $n \times n$. We will call matrix A upper p-triangular, if the block $(\mathbf{c}, \mathbf{d})$ looks like an upper triangular matrix.

We will denote with calligraphic letters block-diagonal matrices of order $(n + p) \times (n + p)$ of the type $\operatorname{diag}(I_p, G) = \mathcal{G}$, where G is a $p \times p$ matrix.

Let G be some $p \times p$ matrix and let $\tilde{L} = \tilde{L}((\mathbf{c}, \mathbf{d})\mathcal{G})$, and $T = T(((\mathbf{c}, \mathbf{d})\mathcal{G}))$. If we take now $G = L$, $\tilde{\mathcal{L}} = \mathrm{diag}(I_p, \tilde{L})$, $\mathcal{L} = \mathrm{diag}(I_p, L)$, then the matrix

$$A_u = \tilde{\mathcal{L}} A \mathcal{L}$$

will become an upper p-triangular matrix, and matrix $T^{-1}A_u$ will be similar to A.

The cofactors L and $\tilde{L}$ of the matrix can be computed sequentially. Since $((\mathbf{c}, \mathbf{d})\mathcal{G}) = (\mathbf{c}, \mathbf{d}L)$ and the first p of the columns of the matrix $(\mathbf{c}, \mathbf{d}L)$ constitute block $\mathbf{c}$ and are independent from L, then using them we can compute sequentially the first p cofactors of the matrix $\tilde{L} : \tilde{\mathbf{L}}_1, \mathbf{D}_1, \tilde{\mathbf{L}}_2, \ldots, \mathbf{D}_{p-1}, \tilde{\mathbf{L}}_p$. From these we can write the first p cofactors of matrix L , can compute p columns of the matrix $\mathbf{d}L$ and after that the following p cofactors of matrix $\tilde{L}$, etc. For $p = 1$ we obtain a quasi-triangular matrix, that is a matrix with zero elements under the second diagonal, which is obtained by the elements $a_{2,1}, a_{3,2}, \ldots, a_{n,n-1}$.

Let us denote by A_k $(1 \leq k \leq n)$ the corner minors of order k of the quasi-triangular matrix $A = (a_{i,j}), a_{i,j} = 0$ for $i \geq 2, j \leq i-1$, and assume $\mathrm{A}_0 = 1$. Then its determinant can be computed as shown

$$\det(\mathrm{A}_n) = a_{nn} \det(\mathrm{A}_{n-1}) + \sum_{i=1}^{n-1} a_{i,n} \det(\mathrm{A}_{i-1}) \prod_{j=i+1}^{n-1} (-a_{j,j-1}).$$

The complexity of this method is $\frac{5}{3}n^3 + O(n^2)$ — multiplicative operations.

5 Conclusion

For computations over commutative domains we have the following results:

- The complexity of the $O(n^3)$ methods (FB) and (OP) for solving systems of linear equations of size $n \times m$ is

$$M_{(FB)} = (1/2)(4n^2m - 2n^3 - 2nm - 3n^2 - 2m + 7n - 2),$$

$$M_{(OP)} = (1/6)(12n^2m - 8n^3 - 6nm - 9n^2 - 12m + 8n + 12).$$

  Suppose that the complexity of the given method for matrix multiplications is $\gamma n^\beta + o(n^\beta)$, where γ and β are constants, and n is the order of the matrix. Then, the complexity of the recursive methods for solving systems of size $n \times m$ is

$$S(n,m) = \gamma \frac{n^\beta}{2^\beta} \left[(4\frac{m}{n} - 2) \frac{1 - n^{2-\beta}}{1 - 2^{2-\beta}} - \frac{1 - n^{1-\beta}}{1 - 2^{1-\beta}} \right] + o(n^{\beta-1}m).$$

- The complexity of the method for the computation of the determinant of a matrix of order n is $S(n,n)$. The complexity of the method for the computation of the kernel of a linear operator is $S(n,m)$.

– The complexity of the method for the computation and the factorization of the adjoint matrix is

$$F(n) = 6\gamma n^{\beta} \frac{1-(n/2)^{1-\beta}}{2^{\beta}-2} + o(n^{\beta})$$

– Finally, the complexity of the best method we know today for the computation of the characteristic polynomial of a matrix of order n is $\frac{5}{3}n^3 + O(n^2)$.

References

1. Akritas, A.: Elements of Computer Algebra with Applications. John Wiley Interscience, New York (1989)
2. Akritas, A.G., Akritas, E.K., Malaschonok, G.I: Various proofs of Sylvester's (determinant) identity. Mathematics and Computations in Simulation 42, 585–593 (1996)
3. Akritas, A.G., Malaschonok, G.I.: Computation of the Adjoint Matrix. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2006. LNCS, vol. 3992, pp. 486–489. Springer, Heidelberg (2006)
4. Abdeljaoued, J.: Berkowitz Algorithm, Maple and computing the characteristic polynomial in an arbitrary commutative ring. Computer Algebra MapleTech 4(3), Birkhauser Boston (1997)
5. Abdeljaoued, J., Malaschonok, G.I.: Efficient Algorithms for Computing the Characteristic Polynomial in a Domain. Journal of Pure and Applied Algebra 156(2-3), 127–145 (2001)
6. Bareiss, E.H.: Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination. Math. Comp. 22(103), 565–578 (1968)
7. Chistov, A.L.: Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In: Budach, L. (ed.) FCT 1985. LNCS, vol. 199, pp. 147–150. Springer, Heidelberg (1985)
8. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation 9, 251–280 (1990)
9. Dodgson, C.L.: Condensation of determinants, being a new and brief method for computing their arithmetic values. Proc. Royal Soc. Lond. A15, 150–155 (1866)
10. Kaltofen, E.: On Computing Determinants of Matrices Without Divisions. In: ISSAC'92. Proc. Internat. Symp. Symbolic Algebraic Comput, pp. 342–349. ACM Press, New York (1992)
11. Kaltofen, E., Villard, G.: On the complexity of computing determinants. In: ASCM 2001. Proc. Fifth Asian Symposium on Computer Mathematics, pp. 13–27 (2001) (extended abstract)
12. Malashonok, G.I.: Solution of a system of linear equations in an integral domain. USSR Journal of computational Mathematics and Mathematical Physics 23(6), 1497–1500 (1983)
13. Malashonok, G.I.: On the solution of a linear equation system over commutative rung. Math. Notes of the Acad. Sci. USSR 42(4), 543–548 (1987)
14. Malashonok, G.I.: A new solution method for linear equation systems over commutative rung. In: International Algebraic Conference, Theses on the ring theory, algebras and modules, Novosibirsk, p. 82 (August 21-26, 1989)

15. Malashonok, G.I.: Algorithms for the solution of systems of linear equations in commutative rings; Effective methods in algebraic geometry (Castiglioncello, 1990), Progr. Math., Birkhauser Boston, Boston, 94, 289–298 (1991)
16. Malashonok, G.I.: Argorithms for Computing Determinants in commutative rings; Diskretnaya Matematika 7(4), 68–76 (1995) transl. in: Discrete Math. Appl. 5(6), 557–566 (1996)
17. Malaschonok, G.I.: On the solution of systems of linear equations; Computational Commutative Algebra, COCOA-IV, Abstracts, Genova, 32 (May 29-June 2, 1995)
18. Malaschonok, G.I.: Recursive Method for the Solution of Systems of Linear Equations; Computational Mathematics. In: Proc. of the 15th IMACS World Congress, I, Berlin, August 1997, Wissenschaft & Technik Verlag, Berlin, pp. 475–480 (1997)
19. Malaschonok, G.I.: A Computation of the Characteristic Polynomial of an Endomorphism of a Free Module; Zap. Nauchnyh Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov (POMI), 258 (1999). Teor. Predst. Din. Sist. Komb. i Algoritm. Metody 4, 101–114 (1999)
20. Malaschonok, G.I.: Effective Matrix Methods in Commutative Domains; Formal Power Series and Algebraic Combinatorics, pp. 506–517. Springer, Heidelberg (2000)
21. Strassen, V.: Gaussian Elimination is not optimal; Numerische Mathematik 13, 354–356 (1969)
22. Sasaki, T., Murao, H.: Efficient Gaussian elimination method for symbolic determinants and linear systems. A.C.M. Trans. Math. Software 8(4), 277–289 (1968)
23. Waugh, F.V., Dwyer, P.S.: Compact computation of the inverse of a matrix. Annals of Mathemaical Statistic 16, 259–271 (1945)